International Series in
Operations Research & Management Science

Christo El Morr
Manar Jammal
Hossam Ali-Hassan
Walid El-Hallak

Machine Learning for Practical Decision Making

A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics

Springer

Christo El Morr • Manar Jammal •
Hossam Ali-Hassan • Walid El-Hallak

Machine Learning for Practical Decision Making

A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics

Springer

Christo El Morr
School of Health Policy and Management
York University
Toronto, ON, Canada

Manar Jammal
School of Information Technology
York University
Toronto, ON, Canada

Hossam Ali-Hassan
Department of International Studies
York University, Glendon Campus
Toronto, ON, Canada

Walid El-Hallak
Ontario Health
Toronto, ON, Canada

ISSN 0884-8289 ISSN 2214-7934 (electronic)
International Series in Operations Research & Management Science
ISBN 978-3-031-16989-2 ISBN 978-3-031-16990-8 (eBook)
https://doi.org/10.1007/978-3-031-16990-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To our families for their love and support. To our students, they are our inspiration.

Preface

This book fills a gap in the machine learning literature. It presents the machine learning concepts in a very simple and practical way starting with a tangible example and building on it the theory. Each machine learning chapter starts with a presentation of the problem to solve, followed by a practical example on how the algorithm that solves the problem works, then comes the presentation of the machine learning algorithm theory and closing remarks (e.g., pitfalls, advantages limitations). This is followed by a set of key terms, a set of questions to test your understanding of the chapter material, a set of references to read more about the subject and hands-on lab exercises. The lab exercises allow readers to apply the chapter's concepts and gain needed skills. To maximize the benefit for readers and to expose them to a myriad of machine learning languages and frameworks, the lab exercises (and sometimes the chapter's material) are built around Python mainly, followed by R (and R Studio), and Weka. One of the strengths of this book is that it can be used by people who are exposed to programming or would like to learn how to program and those who prefer not to program but to solve decision-making problems with machine learning using simple graphical user interface. The former can use Python, the machine learning language par excellence, throughout all chapters, or R (limited to Chaps. 1–4, and 6). The use of Tableau, a visual analytics platform, is reserved to Chap. 5, Data Visualization, while Weka is explained and used in Chaps. 4 and 6–12; given the simplicity of Weka, we believe that there was no necessity to add Weka-based lab exercises for Chaps. 13–15.

This introductory textbook to machine learning for decision making can be used by students in Computer Science, Information Technology, Health Informatics, and Business fields. Depending on the students' level of study and exposure to technology, either Weka or Python can be used. However, given the pervasive use of Python in the market we advise students of all sorts to get exposure to Python and how it works. Teaching the whole Python language is beyond the scope of this book; however, we cover Python's libraries related to machine leaning (e.g., Scikit-Learn, TensorFlow, Keras) and many Python programming concepts.

Another strength of the book is its focus on the necessary content for an introductory course to machine learning while providing enough complexity without being complex or introducing heavy mathematical formulation; the exception being neural networks where we considered that providing less simple mathematical formulations was necessary as an illustration but skipping them would not be a problem to understand the algorithm. There is no mathematical knowledge needed to read and use this book. Chapter 2 provides, in a simple manner coupled with many examples, the main mathematical concepts needed to understand the chapters.

A final strength of this book is the use of a variety of datasets from several domains (e.g., health, business, social media, census, survey) which provide a good exposure to the myriad types of applications in which machine learning can be used.

The book is organized in three parts: Part 1 is an introduction section that encompasses Chaps. 1–5, it introduces machine learning fundamentals and allows for installing the different software tools and the introduction to Python, R, and Weka. The machine learning algorithms and corresponding lab exercises are covered in Chaps. 6–15, and future perspectives are provided in Chap. 16.

Professors who adopt this book have flexibility in the way they want to teach the material; it all depends on the objectives of their course. Some can use to teach machine learning using Python and hence need to cover Chaps. 1, 4 and 6–16; this could be true for courses related to practical machine learning. Professors who are interested in teaching Analytics and (some) machine learning can cover Chaps. 1–5 and then some of the Chaps. 6–16. It is still feasible to cover all chapters in 12 weeks, Chaps. 1 and 2 are an introduction and a quick overview that can be covered with their labs in one session; Chaps. 3 (or some of it) and Chap. 4 are possible to combine in one session. Chapters 5 and 16 can be covered in one session; then each Chaps. 6–14 in one session; and final Chaps. 15 and 16 in one session. This is not to impose a single way of approaching the textbook but to provide examples of alternatives and demonstrate flexibility. The book provides you with flexibility to be adopted in several contexts. The datasets used in the lab exercise touch upon many domains: health, information technology, business, and engineering.

We hope that this first edition of the book will enrich the readers' knowledge and skills and we welcome their comments and suggestions. Readers can access the Python code for the Chap. 6 to Chap. 15 lab exercises on GitHub https://github.com/christoelmorr/ML-4-Practical-DM.git.

Toronto, ON, Canada

Christo El Morr
Manar Jammal
Hossam Ali-Hassan
Walid El-Hallak

Contents

Chapter 1
Introduction to Machine Learning

1.1 Introduction to Machine Learning

The last two decades have seen a quiet but important revolution in computer science. Now more than ever, computers and algorithms are leading to more prosperous and more accurate insights with software that learns from experience and adapts automatically to match the needs of its tasks [1]. Formerly, the programmer decided how the system would work by manually writing the code. Today, we do not write programs but rather collect data consisting of instruction insights, and develop the algorithms changes that manipulate it as necessary to extract patterns and insights. Today, we have programs that can recognize faces and fingerprints, understand speech, translate, navigate, drive a car, recommend movies, and many more [1]. This is possible now because of artificial intelligence (AI) and its fields, mainly machine learning.

Artificial intelligence reflects a computer's ability to recognize patterns to then use and apply those patterns based on available data [2]. Artificial intelligence mimics human cognition by accessing data from a variety of sources and systems to make decisions and learn from their results and patterns [3]. Artificial intelligence was inspired by the human brain, given that computers were once known as "electronic brains" [1]. The human undoubtedly has incredible processing capabilities that humans have long aimed to understand in order to use and create an artificial version known as artificial intelligence. Artificial intelligence has been overgrowing since the Turing Test (originally named the imitation game) was conducted in 1950 by Alan Turing, which suggested that computers do have the ability to think intelligently as artificial entities [3]. Alan Turing's research revolutionized how the world perceived artificial intelligence and its use in daily life [2]. As artificial intelligence has evolved from being a purely academic field, it has become a significant part of many social and economic sectors, including speech recognition, medical diagnosis, vehicles, and voice-activated assistance [3].

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_1

Machine learning (ML) is a type of artificial intelligence that branches from computer science [4]. Machine learning is not just a data storing, processing, or training problem; it is instead a means to achieve artificial intelligence by either training it on a dataset or using repeated trials to train a computer program to maximize intelligent performance [1]. Machines are great at making smart decisions because of the enormous datasets. On the other hand, humans are much better at making decisions with limited information. This combination is highly effective in leveraging both human and machine intelligence in creating machine learning models. Combining machine learning and human intelligence provides remarkably high levels of accuracy, leading us to artificial intelligence [3].

1.2 Origin of Machine Learning

Machine learning (ML) is often credited to a psychologist from Cornell University named Frank Rosenblatt, who, based on his theories about the workings of the human nervous system, developed a machine capable of recognizing letters of the alphabet in the 1960s [4]. The machine, called the "perceptron," converted analog signals into discrete ones, becoming the prototype for modern artificial neural networks (ANNs). Further studies of the structures and learning abilities of neural networks took place in the 1970s and 1980s. Even so, the Novikoff theorem (1962), which states that a perceptron learning algorithm can be converged in a finite number of steps, has become more widely known and credited for machine learning. In 1979, students at Stanford University created a notable invention known as the "Stanford cart," which could navigate different obstacles in a room [4]. The invention of the Stanford cart is an important part of the history of artificial intelligence and machine learning, as it paved a pathway for robotics research within the area. Today, rover robotic cleaning vacuums use a similar method to avoid obstacles in a room and pick up foreign materials such as dirt.

1.3 Growth of Machine Learning

Machine learning has transformed the twenty-first century through its progress and growth, increasing computer competence in various fields, including the automotive industry, healthcare, commerce, banking, and manufacturing [5]. The first decade of the twenty-first century marked a turning point in machine learning history, which can be attributed to three trends that worked collaboratively [4]. The first trend is **big data**, which refers to a large volume of data that is complicated and requires specialized methods to process [4]. This very large and mostly accessible data includes weather data, business transactions data, medical test results, social media posts, security camera recordings, GPS locations from smartphones, sensor data, and many more. Big data tomorrow will be bigger than today, and with more data, trained models will get more intelligent [1]. The second trend is the **reduced cost** of parallel computing and memory by distributing the processing of high volumes of data between simple processors [4]. In addition to that, more complex and powerful yet affordable processors like graphics processing units (GPU) are produced. These resources are available for data scientists and organizations today via cloud computing without the need for huge investments in hardware. The reduced costs and increased processing power and storage capability allowed for increased data capturing and storage and more efficient and faster coding and analysis. The third trend is the development of new algorithms of **machine learning** [4], many of which are readily available in open-source communities. The third trend is by far the most important, as it aided in the creation of artificial neural networks that support higher-level functions for data processing [4]. ANNs are crucial algorithms within machine learning (ML), serving the important purpose of solving complex problems.

There is more data than our sensors or brains can handle or process. The information available online today contains massive amounts of digital text and is now so vast that manual processing is impossible. The use of machine learning for this is much more efficient and is known as machine reading. The basic advantage of machine learning is that it can be applied to a wide range of tasks without explicitly being programmed to learn. Using machine learning, we can build systems that are capable of learning and adapting to their environment on their own with minimal supervision and maximal user satisfaction [1].

1.4 How Machine Learning Works

Machine learning works by utilizing many algorithms that make intelligent predictions based on the datasets being used. These datasets can be enormous, consisting of millions of data that cannot be processed by the human mind alone [5]. Machine learning has four variants: supervised, unsupervised, semi-supervised, and reinforcement learning [6]. In order to understand these variants, it is important to understand “labels.” A “label” in machine learning is the dependent variable and is a specified value of the outcome [6]. In supervised learning, labeled datasets are used by machine learning professionals to train algorithms by setting parameters to make accurate predictions about data [3]. Regression is one example of supervised learning [1]. On the other hand, unsupervised learning consists of multiple unlabeled datasets which are used to detect structure and patterns using the algorithm [3]. Data clustering is one example of unsupervised learning, and it is also much faster, as there are fewer labeled data [1]. Semi-supervised learning fits the models to both labeled and unlabeled data [6]. The main goal of semi-supervised learning is to understand how combining labeled and unlabeled data can change learning behavior and the design algorithms that use this combination [7]. Reinforcement learning is a machine learning algorithm that allows machines and software to automatically evaluate optimal behavior in specified contexts for improved efficiency [8]. Reinforced learning is usually used for training complicated artificial intelligence models to increase automation. All four of these variants are an important component of machine learning outcomes and play a significant role per their learning capabilities [8].

1.5 Machine Learning Building Blocks

Statistics, data mining, analytics, business intelligence, artificial intelligence, and machine learning are concepts, methods, and techniques used to understand data and explore it to find valuable information, relationships, trends, patterns, and anomalies and ultimately to make predictions. In the following section, we will introduce data, its types, and how it is managed and explored. We will also introduce business intelligence and data analytics. Statistics will be covered in detail in Chap. 2, and in Chap. 3, we will introduce data mining and explore machine learning and the different algorithms in more depth.

1.5.1 Data Management and Exploration

1.5.1.1 Data, Information, and Knowledge

Data and information are comparable but not the same. While many believe that data and information represent the same concept and can be used interchangeably, these two terms are quite distinct. Data are streams of raw, specific, and objective facts or observations generated from events such as business transactions, inventory, or medical examinations. Standing alone, data have no intrinsic meaning. Data is generally broken into two categories, structured and unstructured. Structured data, like patient records, sales transactions, and warehouse inventory, has a predefined format and can be easily processed and analyzed. It is commonly stored and managed in a relational database. Unstructured data, like free text, videos, images, audio files, tweets, and portable medical device outputs, are complex in their form and are more difficult to manage, process, and analyze [9, 10].

Once processed (e.g., filtered, sorted, aggregated, assembled, formatted, or calculated), data becomes endowed with relevance and purpose and is put in context. Data thus turns into information. Information is a cluster of facts that are meaningful and useful to human beings in processes such as making decisions [11, 12]. For instance, patients' IDs, names, dates of birth, home addresses, postal codes, phone numbers, emails, and diagnoses are examples of data that can be collected in a community center, clinic, or hospital, while a bar chart presenting the percentage of patients in different age groups, a pie chart representing the number of patients per type of disease, or a map representing the patients' distribution in a geographic area are examples of information.

Consider a simple example that we can all relate to, how purchases are processed at checkout at a grocery store. Scanning the barcodes of the products at a store generates or accesses data in the form of a product number, a short description of the product, and a price. When these data are processed, an invoice is generated, and the store's inventory is updated. This generated information helps the store determine how much to charge the customer and process the payment. This new information also lets the store manager know how much inventory is left for each product and helps him decide when to order new supplies [13].

In summary, data is the new oil; data is simply a collection of facts. Once data are processed, organized, analyzed, and presented in a way that assists in understanding reality and ultimately making a decision, it is called information. Information is ultimately used to make a decision and take a course of action.

When processed further and internalized by humans, information becomes knowledge. Knowledge can be defined as understanding, awareness, or experience. It can be learned, discovered, perceived, inferred, or understood [10]. In the grocery store example, knowledge would be the awareness of which products sell the most during specific times of the year, which translates into the decision to order additional supplies to avoid out of the stock issue [13].

As we move from data to information and then to knowledge, we see more human contribution and greater value and, traditionally, a decreasing role of technology. Data are easily captured, generated, structured, stored, and transmitted by information and communication technology (ICT). Information, which is data endowed with relevance and purpose [14], requires analysis, a task increasingly being done by technology but also by human mediation and interpretation. Finally, knowledge, valuable information from the human mind, is difficult to capture electronically, structure and transfer and is often tacit and personal to the source [12, 15].

1.5.1.2 Big Data

There is no unique or universal definition of big data, but there is a general agreement that there has been an explosion in data generation, storage, and usage [9]. Big data is a common term used to refer to the massive size structured and unstructured data generated, made available, and being used [16]. These data come from daily business transactions at banks and retailers, for example, from sensors such as security cameras and monitoring systems, the GPS systems on every mobile phone, content posted on social media such as YouTube videos, and from many more ubiquitous sources [13]. Big data in the healthcare field comes from medical devices such as MRI scanners and X-ray machines, sensors such as heart monitors, patient electronic medical and health records, insurance providers' records, doctors' notes, genomic research studies, wearable devices, and many more [17]. An example of big data is what is collected by Fitbit, a manufacturer of wearable activity trackers. In 2018, it was announced that Fitbit had collected 150 billion hours' worth of heart rate data from tens of millions of people from all over the world. These data also

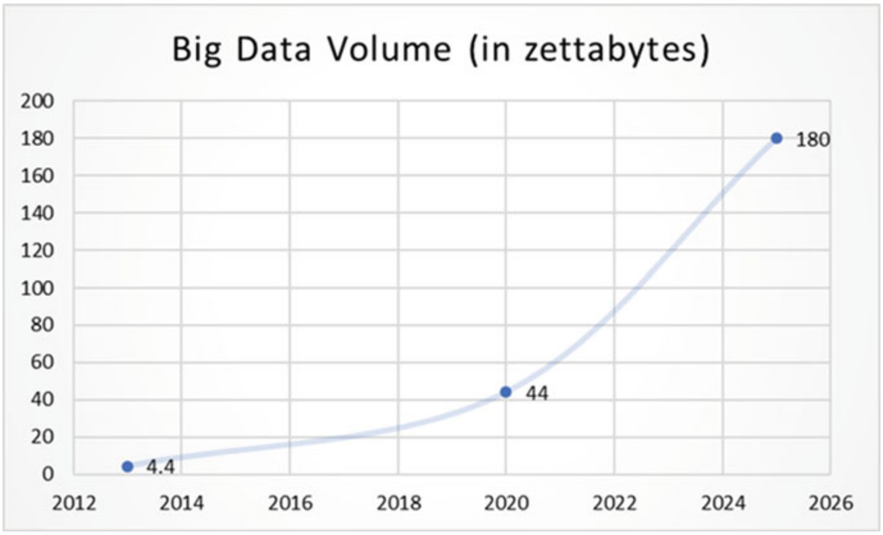


Fig. 1.1 Exponential increase in the volume of big data (actual and projected) based on an IDC study (adapted from [13])

include sex, age, location, height, weight, activity levels, and sleep patterns. Moreover, Fitbit has 6 billion nights' worth of sleep data [18].

There are multiple factors behind the emergence and growth of big data, and they include technological advances in the field of information and communication technology (ICT), where computing power and data storage capacity are continuously increasing while their cost is decreasing. The increased connectivity to the Internet is another major factor. Today, most people have a mobile device, and many modern pieces of equipment are connected to the Internet [13].

Big data is generally characterized by the four Vs: volume, variety, velocity (introduced originally by the Gartner Group in 2001), and veracity (added later by IBM) [19]. Multiple additional Vs were introduced later, including validity, viability, variability, vulnerability, visualization, volatility, and value [9, 19, 20]. Volume is the most defining characteristic of big data. The volume of data generated is increasing exponentially, and new units of measure have been created, such as zettabytes (10^{21}), to accommodate this increasing volume of data. According to IDC, a market-research firm, the data created and copied in 2013 was 4.4 zettabytes, and this number is projected to exponentially increase to 44 zettabytes in 2020 and 180 zettabytes in 2025 (Fig. 1.1) [19, 21]. Examples of large volumes of data are the 20 terabytes (10^{12}) of data produced by Boeing jets every hour and the 1 terabyte of data uploaded on YouTube every 4 minutes [22].

Variety refers to the different forms of big data, such as videos, pictures, social media posts, images from X-ray machines, location data from GPS systems, data from sensors like security devices and wearable wireless health monitors, and many more. **Velocity** refers to the very high speed at which big data are continuously being

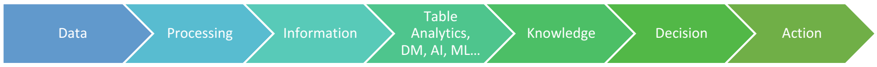


Fig. 1.2 Data-to-action value chain

generated, for example, from medical devices and monitors in hospitals' intensive care units or security cameras. Such data must be generated and analyzed in real-time, particularly when the outcome has a direct impact on someone's safety in the case of driverless cars or their financial situation in the case of the stock market. Finally, **veracity** represents the high level of uncertainty and low levels of reliability and truthfulness of big data [9, 10, 13, 19]. Data can be biased, incomplete, or filled with noise, and data scientists and analysts spend more than 60% of their time cleaning data [19]. These characteristics of big data represent challenges for any company or industry. Some of the challenges are technical, such as being able to analyze the large volume of data, which is generated very rapidly and in many different formats. Other challenges may be administrative, such as the reliability of the data [13].

The increasing volume and complexity of data, which is very rapidly generated in different formats, have made it practically impossible for humans to analyze without sophisticated analytics techniques. Therefore, techniques like data analytics, data mining, artificial intelligence, and machine learning are playing an increasing role in transforming data or information into knowledge and helping humans make decisions and take action (Fig. 1.2).

1.5.1.3 OLAP Versus OLTP

A significant amount of data is produced by daily business transactions, be it a purchase of a product, such as an airline ticket or a book; withdrawing money from a bank; admitting a patient to a hospital; generating medical imagery, such as X-rays; updating patient records after a medical examination; and so on [13]. These transactions are managed by transaction processing systems (TPS) or online transaction processing systems (OLTP), which are computerized systems, such as payroll systems, order processing systems, reservations systems, or enterprise resource planning (ERP) systems, that perform and record the transactions that are necessary to conduct a business, such as employee record keeping, payroll, sales order entry, and shipping [11, 23]. At a bank, OLTP can be used to create new accounts, deposit and withdraw funds, process checks, transfer funds to other accounts, withdraw cash, pay bills, calculate and apply fees, and generate a report on all transactions performed during a period of time. OLTP systems function at the operational level of an organization and are mainly responsible for acquiring and storing data related to day-to-day automated business transactions, running everyday real-time analyses, and generating reports [10]. The value of the data generated and maintained by an OLTP system goes beyond supporting an organization's operations and generating reports.

Table 1.1 A comparison between OLTP and OLAP (adapted from Sharda et al. (2015) [10])

Criteria	OLTP	OLAP
Purpose	Support and automate day-to-day business functions and transactions	Support decision-making and provide analytical capabilities and answers to business and management queries
Data sources	Transaction database (a data repository formatted to maximize efficiency and consistency)	Data warehouse or data mart (a data repository formatted to maximize accuracy and completeness)
Reporting	Routine, periodic, narrowly focused reports	Ad-hoc, multidimensional, broadly focused reports and queries
Resource requirement	Ordinary relational databases	Large capacity, specialized databases
Execution speed	Fast (recording business transactions and routine reports)	Relatively slow (resource-intensive, complex, large-scale queries, multiple data sources)

These data, coming from multiple sources or systems, can be further analyzed to support organizational decision-making using online analytical processing (OLAP). OLAP can manipulate and analyze large volumes of data from different perspectives and answer ad-hoc inquiries by executing multidimensional analytical queries [20, 23]. An OLAP system is a computer system with advanced query and analytical functionality, such as ad-hoc and what-if analysis capabilities [20, 24]. At a bank, an OLAP system can be used to predict which customers may quit, an exercise called churn analysis. OLAP can predict which customers are most susceptible to certain new services to develop targeted marketing campaigns instead of blanket or mass marketing, which is more expensive and less efficient and effective. Table 1.1 presents a brief comparison between OLTP and OLAP [10].

1.5.1.4 Databases, Data Warehouses, and Data Marts

Today, most data is stored, organized, manipulated, and managed inside databases. A database is a collection of data formatted and organized into records that facilitates accessing, updating, adding, deleting, and querying those records [25]. A database can be perceived as a collection of files that are viewed as a single storage area of organized data records that are available to a wide range of users [22].

The most common type of database is a relational database, which consists of tables (called relations, hence the name "relational database") that are connected via relationships (not to be confused with relations or tables). Each table, not different from a spreadsheet, represents an entity of interest for which we collect and manage data, for example, a customer table or a student table. Each table consists of multiple fields related to the entity it represents, such as the customer's last name, first name, social security number, phone number, and address. An example of an employee relation or table is in Table 1.2.

Table 1.2 An employee table divided into rows (i.e., records) and columns (i.e., fields)

Employee ID	Last name	First name	Department	Home Address	Employment date
1	Gonzales	Valentina	Marketing	55 Main Street	March 17, 2010
2	Singh	Peter	Accounting	12 Lakeshore East	November 5, 2016
3	Watson	Brian	Sales	921 Holmes Avenue	April 7, 2019
4	Jones	Peter	Accounting	6 McGill Crescent	June 24, 2020
5	Smith	Paul	Personnel	1025 Bay Avenue	April 28, 2005

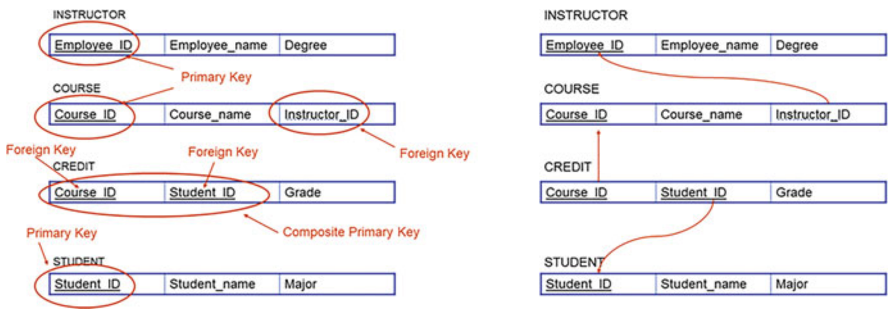


Fig. 1.3 Relationships in a relational database

In a relational database, tables are connected via relationships. Relationships are created by linking primary keys and foreign keys in different tables. In a database, each table has a primary key, which is a field or attribute that is used to uniquely identify each record in the table, such as a student ID, a patient medical health card number, a product code, or a customer phone number. A primary key can consist of multiple fields as long as their combination is unique for every record in the table, such as the combination of a shipment number and product ID. Such a key is called a composite primary key. A table also has foreign keys, which point to primary keys in other tables and confirm the presence of relationships between these tables. Figure 1.3 presents an overview of the relationships in a university database.

Relational databases are designed to quickly access the data for transaction processing (via TPS/OLTP systems), such as admitting a new patient to the hospital or performing a sales transaction. In addition to daily transactions, the database can be queried for occasional reports or information, such as the account balance of a customer; the report can then be used for decision-making, such as providing a line of credit or a loan. However, as the size of a database grows due to day-to-day transactions generating additional new records in the tables, it becomes very time-consuming to generate any analytics using the data. Moreover, analytics or OLAP would slow down the system, making routine transactions handled by the OLTP too slow. Transferring funds between customer bank accounts could take minutes instead of seconds. A solution would be to extract the data from the different

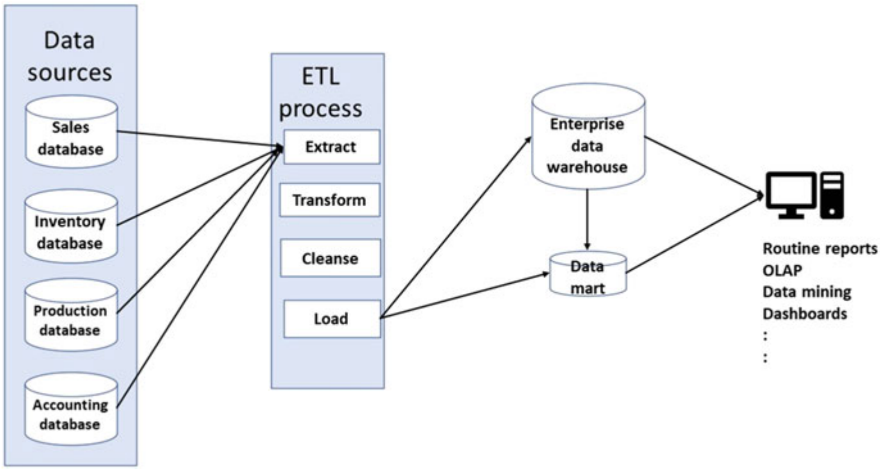


Fig. 1.4 Data warehouse overview (adapted from Sharda et al. (2013) [26])

databases, transform it into an appropriate format, and load it into a special database specifically designed for querying and OLAP. Data that is redundant or has no value would be cleansed. This process is called "extract, transform, and load," or ETL for short [13]. The databases suitable for querying, OLAP, and decision-making are referred to as "data warehouses" and "data marts." A data warehouse is a physical repository where current and historical data are specifically organized to provide an enterprise-wide cleansed data in a standardized format [25, 26]. The data in a warehouse is structured to be available in a form ready for OLAP, data mining, querying, reporting, and other decision support applications [26]. A data mart is a data warehouse subset usually focused on a single subject or department [22, 26]. Figure 1.4 presents an overview of a data warehouse.

Data inside a data warehouse or data mart is designed based on the concept of dimensional modeling, where high-volume complex queries are needed. The most common style of dimensional modeling is the star schema. While in an OLTP environment, the database consists of tables representing entities of interest, such as patients and their attributes (name, phone, address, etc.), the star schema has a fact table with a large number of attributes (mainly numbers) that are most needed for analysis and queries, while the rest of the valuable data are stored in attached dimension tables [24, 26, 27]. Figure 1.5 provides a visual representation of an OLTP and an OLAP-based database structure.

While an OLTP system is designed for operational purposes and thus it is detailed, continuously and easily updated, has very current data, has to be always available, and is designed for transactional speed, an OLAP-based system is more informational and has summary data, is not continuously updated, has mainly historical and integrated data, and is designed for complex queries and analytics [24, 26, 27].

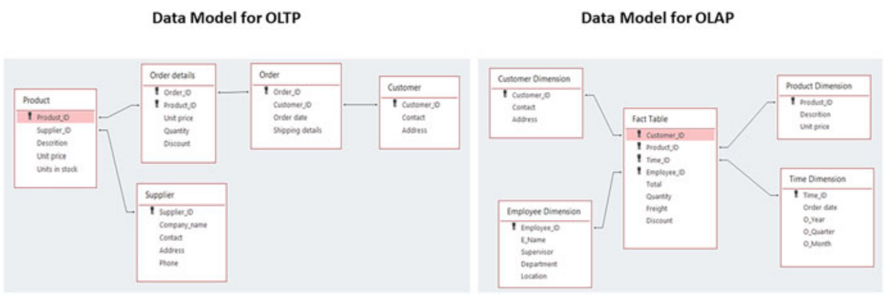


Fig. 1.5 Data models for OLTP and OLAP (adapted from Mailvaganam (2007) [27])

To perform the analytical processing, a data cube is created, which is a multidimensional data structure that is generated out of the star schema and allows for fast analysis of data. The cube is multidimensional but is commonly represented as three-dimensional for ease of viewing and understanding. Each side in a cube represents a dimension, such as a patient, procedure, or time, and the cells are populated with data from the fact table. The cube is optimally designed for common OLAP operations, such as filtering, slicing, dicing, drilling up and down, rolling up, and pivoting [24, 26, 27], which will be explored next.

1.5.1.5 Multidimensional Analysis Techniques

Data to be reported can be manipulated in many cases with simple arithmetic and statistical operations, such as summing up (e.g., total sales in a year), counting (e.g., the number of sales transactions), calculating the mean (e.g., the average profit of sales), filtering (e.g., extracting names of customers in a certain region who made the highest purchases), sorting, ranking, and so on. To extract the data from multiple tables in a relational database of a TPS system, one can issue an SQL query command that pulls out the data from multiple tables by performing a "join" operation (i.e., joining related data from different tables). To perform OLAP on a multidimensional data structure, similar to a cube in a data warehouse, several operations or techniques may be needed, such as slicing, dicing, and pivoting [9, 24, 28].

For simplicity, assume that we have a three-dimensional dataset of sales, where the dimensions are product, region, and year, which can be represented as a cube, where each axis is a dimension, and the cells contain sales data in thousands of dollars (Fig. 1.6). In this figure, we find that the sale of chairs in QC in 2021 was worth $110,000.

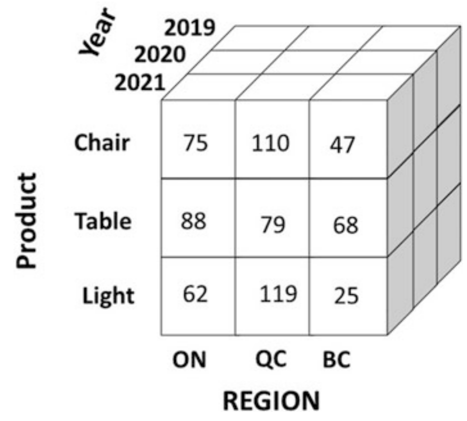


Fig. 1.6 Example of an OLAP cube containing sales figures of three products in three regions over a period of 3 years

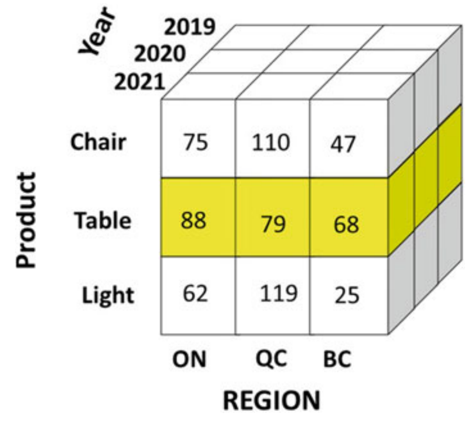


Fig. 1.7 Representation of slicing data to extract the sales of tables in all regions and years

1.5.1.5.1 Slicing and Dicing

Slicing and dicing operations are used to make large amounts of data easier to understand and work with. Slicing is a method to filter a large dataset into smaller datasets of interest while dicing these datasets creates even more granularly defined datasets [9]. Slicing is taking a single slice out of the cube, representing one dimension, showing, for example, the sales of tables for each region and year (Fig. 1.7).

Another example of slicing is in Table 1.3, where the sales of each product are summed up for all regions and years.

Dice is a slice on more than two dimensions of the cube [28]. Dicing is putting multiple side-by-side members from a dimension on an axis with multiple related members from a different dimension on another axis, allowing the viewing and analysis of the interrelationship among different dimensions [24]. Two examples of

Table 1.3 Example of slicing

Sales for all regions and years	
Product	Sales
Chair	$694,000
Table	$710,000
Light	$575,000
Total sales	$1,979,000

dicing are depicted in Table 1.4, showing the sales of all products per region per year and sales per region per product for all years.

1.5.1.5.2 Pivoting

A pivot table is a cross-tabulated structure (crosstab) that displays aggregated and summarized data based on the ways the columns and rows are sorted. Pivoting means swapping the axes or exchanging rows with columns and vice versa or changing the dimensional orientation of a report [9, 24, 28] (Table 1.5).

1.5.1.5.3 Drill-Down, Roll-Up, and Drill-Across

Drilling down or rolling up is where the user navigates among levels of the data ranging from the most summarized (roll-up) to the most detailed (drill-down) [28] and happens when there is a multilevel hierarchy in the data (e.g., country, province, city, neighborhood) and the users can move from one level to another [24]. Figure 1.8 shows an example of drilling down on the product dimension. When you roll up, the key data, such as sales, are automatically aggregated, and when you drill down, the data are automatically disaggregated [9].

Drilling across is a method where you drill from one dimension to another, but where the drill-across path must be defined [24]. Figure 1.9 shows an example of a drill-across from the store CA to the product dimension.

1.5.2 The Analytics Landscape

Analytics is the science of analysis—using data for decision-making [26]. Analytics involves the use of data, analysis, and modeling to arrive at a solution to a problem or to identify new opportunities. Data analytics can answer questions such as (1) what has happened in the past and why, (2) what could happen in the future and with what certainty, and (3) what actions can be taken now to control events in the future [9, 10].

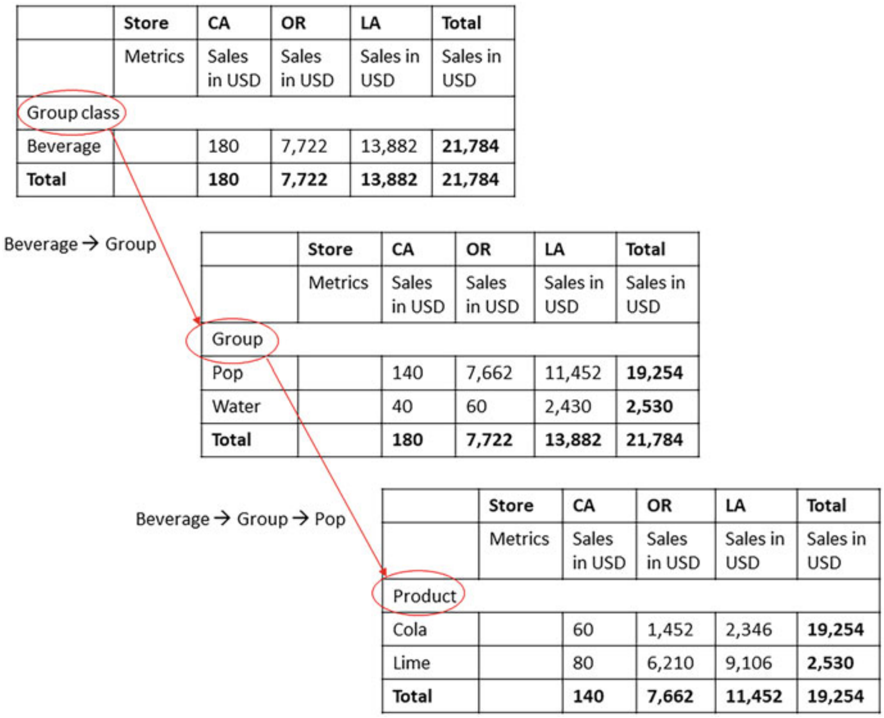
Data analytics have traditionally fallen under the umbrella of a larger concept called "business intelligence," or BI. BI has been defined as "the integration of data

Table 1.4 Dicing for region/year (left) and region/product (right)

	Year					Product			
	2019	2020	2021	Total		Chair	Table	Light	Total
Region					Region				
ON	$220,000	$229,000	$225,000	$674,000	ON	$240,000	$239,000	$190,000	$669,000
QC	$290,000	$303,000	$299,000	$892,000	QC	$249,000	$246,000	$201,000	$696,000
BC	$130,000	$143,000	$140,000	$413,000	BC	$205,000	$225,000	$184,000	$614,000
Total	$640,000	$675,000	$664,000	$1,979,000	Total	$694,000	$710,000	$575,000	$1,979,000

Table 1.5 Pivoting region and year

	Year						Region			
	2019	2020	2021	Total			ON	QC	BC	Total
Region						Year				
ON	$220,000	$229,000	$225,000	$674,000		2019	$220,000	$290,000	$130,000	$640,000
QC	$290,000	$303,000	$299,000	$892,000	⇔	2020	$229,000	$303,000	$143,000	$675,000
BC	$130,000	$143,000	$140,000	$413,000		2021	$225,000	$299,000	$140,000	$664,000
Total	$640,000	$675,000	$664,000	$1,979,000		Total	$674,000	$892,000	$413,000	$1,979,000

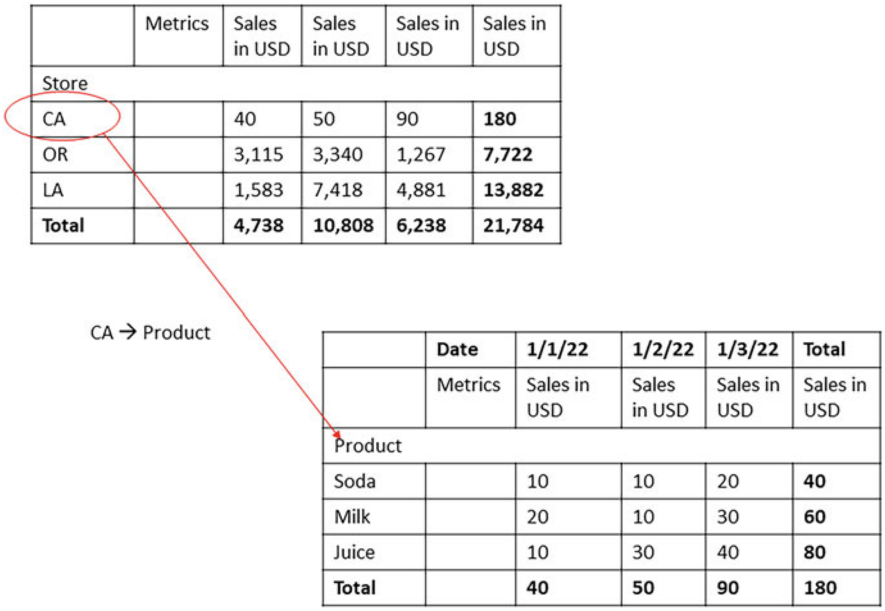

Beverage → Group
Beverage → Group → Pop

	Store	**CA**	**OR**	**LA**	**Total**
	Metrics	Sales in USD	Sales in USD	Sales in USD	Sales in USD
Group class					
Beverage		180	7,722	13,882	**21,784**
Total		**180**	**7,722**	**13,882**	**21,784**

	Store	**CA**	**OR**	**LA**	**Total**
	Metrics	Sales in USD	Sales in USD	Sales in USD	Sales in USD
Group					
Pop		140	7,662	11,452	**19,254**
Water		40	60	2,430	**2,530**
Total		**180**	**7,722**	**13,882**	**21,784**

	Store	**CA**	**OR**	**LA**	**Total**
	Metrics	Sales in USD	Sales in USD	Sales in USD	Sales in USD
Product					
Cola		60	1,452	2,346	**19,254**
Lime		80	6,210	9,106	**2,530**
Total		**140**	**7,662**	**11,452**	**19,254**

Fig. 1.8 Drilling down (adapted from Ballard et al. (2012) [24])



	Metrics	Sales in USD	Sales in USD	Sales in USD	Sales in USD
Store					
CA		40	50	90	**180**
OR		3,115	3,340	1,267	**7,722**
LA		1,583	7,418	4,881	**13,882**
Total		**4,738**	**10,808**	**6,238**	**21,784**

	Date	**1/1/22**	**1/2/22**	**1/3/22**	**Total**
	Metrics	Sales in USD	Sales in USD	Sales in USD	Sales in USD
Product					
Soda		10	10	20	**40**
Milk		20	10	30	**60**
Juice		10	30	40	**80**
Total		**40**	**50**	**90**	**180**

Fig. 1.9 Drilling across (adapted from Ballard et al. (2012) [24])

from disparate source systems to optimize business usage and understanding through a user-friendly interface" [29] and as "the concepts and methods to improve business decision-making by using fact-based support systems" [30]. BI is a conceptual framework for decision support that combines a system architecture, databases and data warehouses, analytical tools, and applications [22]. BI is a mature concept that applies to many fields, despite the presence of the word "business." While remaining a quite common term, BI is slowly being replaced by the term "analytics," sometimes referring to the same thing. The major objective of BI is to enable interactive access to data and data manipulation, and to provide end users (e.g., managers, professionals) with the capacity to perform analysis for decision-making. BI analyzes historical and current data and transforms it into information and valuable insights (and knowledge), which lead to more informed and evidence-based decision-making [10]. BI has been very valuable in applications such as customer segmentation in marketing, fraud detection in finance, demand forecasting in manufacturing, and risk factor identification and disease prevention and control in healthcare. BI uses a set of metrics to measure past performance and report a set of indicators that can guide decision-making; it involves a set of methods such as querying structured datasets and reporting the findings, using dashboards, automated monitoring of critical situations, online analytical processing (OLAP) using cubes, slice and dice, and drilling. BI is essentially reactive and performed with much human involvement [13].

Analytics, alternately, are more proactive and can be performed automatically by a set of algorithms (e.g., data mining and machine learning algorithms). Analytics access structured data (e.g., product code, quantity sold, and current inventory level) and unstructured data (e.g., free text describing the product or pictures of the product); they describe what happened in the past, such as how many units of a certain product were sold last year (descriptive analytics); predict what will (most likely) happen in the future, such as how many units we expect to sell next year (predictive analytics); or even prescribe what actions we should take to have certain outcomes in the future (prescriptive analytics), such as what quantity of the product we should order and when. Analytics analyze trends, recognize patterns, and possibly prescribe actions for better outcomes, and they use a multitude of methods, such as predictive modeling, data mining, text mining, statistical analysis, simulation, and optimization [13].

Some sources offer a distinction between BI and analytics using a spectrum of analytics capabilities. BI is traditional and mature and looks at the present and historical data to describe the current state of a business. It uses basic calculations to provide answers. This functionality is compatible with what is referred to as "descriptive analytics" and is at the lower end of the spectrum. Analytics, on the other hand, mines data to predict where the business is heading and prescribes actions to maximize beneficial outcomes. It uses mathematical models to determine attributes and offer predictions. These functionalities are referred to as "predictive" and "prescriptive analytics" and fall on the higher end of the analytics spectrum [13, 31]. Having clarified to a certain extent the difference between BI and analytics, we will refrain from using the term BI and rely instead on the analytics taxonomy:

descriptive, diagnostic, predictive, and prescriptive analytics, which will be described in detail below.

1.5.2.1 Types of Analytics (Descriptive, Diagnostic, Predictive, Prescriptive)

Analytics are of four types: descriptive, diagnostic, predictive, and prescriptive. These types have increasing difficulty and complexity levels and provide increasing value to the users (Fig. 1.10).

1.5.2.1.1 Descriptive Analytics

Descriptive analytics query past or current data and report on what happened (or is happening). Descriptive analytics display indicators of past performance to assist in understanding successes and failures and provide evidence for decision-making; for instance, decisions related to the delivery of quality care and optimization of performance need to be based on evidence [13].

Using descriptive analytics, such as reports and data visualization tools (e.g., dashboards), end users can look retrospectively into past events; draw insight across different units, departments, and, ultimately, the entire organization; and collect evidence that is useful for an informed decision-making process and evidence-based actions. At the initial stages of analysis, descriptive analytics provide an understanding of patterns in data to find answers to the "What happened?" questions,

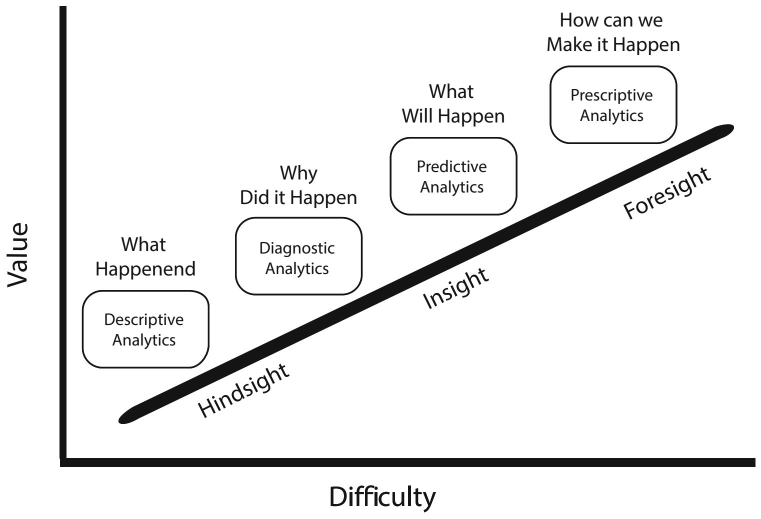


Fig. 1.10 Types of analytics, the value they provide, and their level of difficulty (adapted from Rose Business Technologies [32])

for example, “Who are our best customers in terms of sales volume?” and “What are our least selling products?” Descriptive statistics, such as measures of central tendency (mean, median, and mode) and measures of dispersion (minimum, maximum, range, quartiles, and standard deviations), as well as distribution of variables (e.g., histograms), are used in descriptive analytics [13].

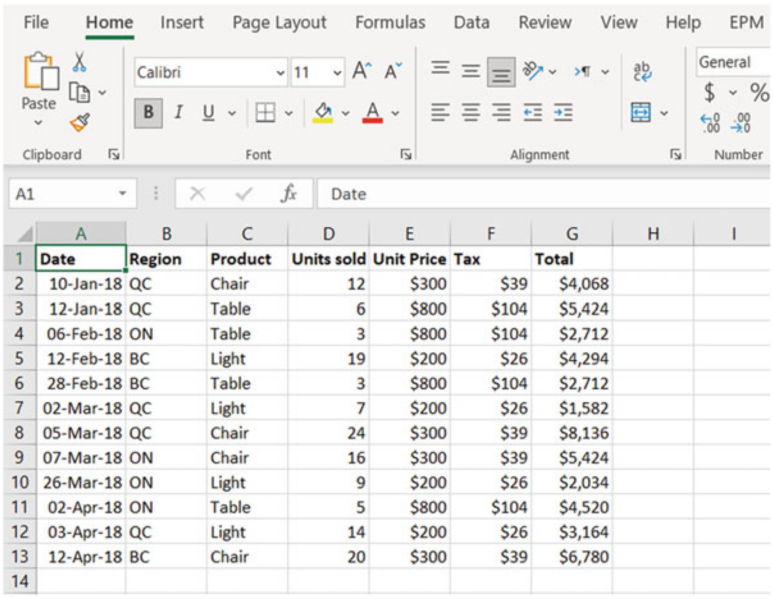
Descriptive analytics can quantify events and report on them and are a first step in turning data into actionable insights. Descriptive analytics, for example, can help with population health management tasks, such as identifying how many patients are living with diabetes, benchmarking outcomes against government expectations, or identifying areas for improvement in clinical quality measures or other aspects of care [33]. Descriptive analytics considers past data analysis to make decisions that help us achieve current and future goals. Statistical analysis is the main “tool” used to perform descriptive analytics; it includes descriptive statistics that provide simple summaries, including graphics analysis, measures of central tendencies (e.g., frequency graphs, average/mean, median, mode), or measures of data variation or dispersion (e.g., standard deviation) [13].

Surveys, interviews, focus groups, web metrics data (e.g., number of hits on a webpage, number of visitors to a page), app metrics data (e.g., number of minutes spent using a feature), and health data stored in electronic records can be the source for all analytics, including descriptive analytics. Media companies and social media platforms (e.g., Facebook) use descriptive analytics to measure customer engagement; managers in hospitals can use descriptive analytics to understand the average wait times in the emergency room (ER) or the number of available beds. Descriptive analytics allow us to access information needed to make *actionable decisions* in the workplace. They allow decision-makers to *explore* trends in data (why do we have long lines in the ER?), to *understand* the “business” environment (who are the patients coming to the ER?), and to possibly *infer an association* (i.e., a correlation) between an outcome and some other variables (patients with the chronic obstructive pulmonary disease tend to have more visits to the ER) [13].

Reports are the main output in descriptive analytics, where findings are presented in charts (e.g., a bar graph or pie chart), summary tables, and most interestingly, pivot tables. A pivot table is a table that summarizes data originating from another table and provides users with the functionality to sort, average, sum, and group data in a meaningful way [13] (Fig. 1.11, 1.12 and 1.13).

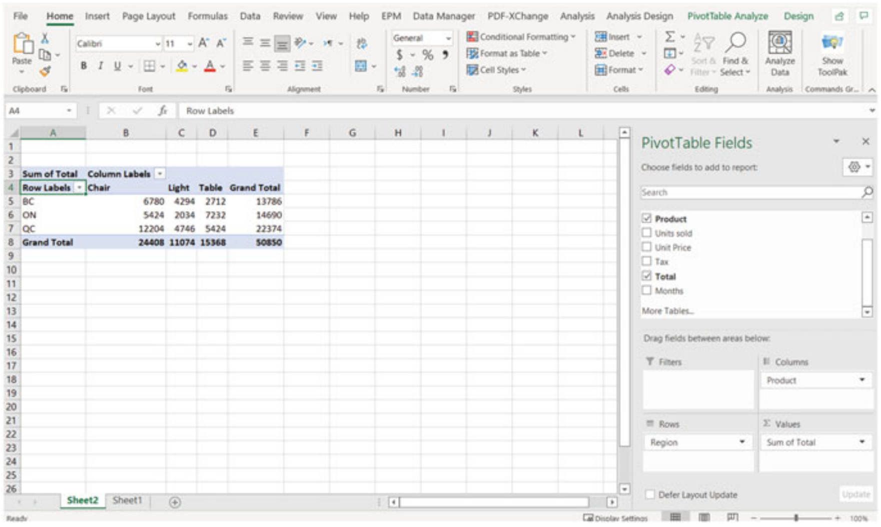
1.5.2.1.2 Diagnostic Analytics

Descriptive analytics give us insight into the past but do not answer the question, “Why did it happen?” Diagnostic analytics aims to answer that type of question. They focus on enhancing processes by identifying why something happened and what the relationships are between the event and other variables that could constitute its causes [34]. They involve trend analysis, root cause analysis [35], cause and effect analysis [36, 37], and cluster analysis [38]. They are exploratory and provide users with interactive data visualization tools [39]. An organization can monitor its performance indicators through diagnostic analysis.



	A	B	C	D	E	F	G
1	Date	Region	Product	Units sold	Unit Price	Tax	Total
2	10-Jan-18	QC	Chair	12	$300	$39	$4,068
3	12-Jan-18	QC	Table	6	$800	$104	$5,424
4	06-Feb-18	ON	Table	3	$800	$104	$2,712
5	12-Feb-18	BC	Light	19	$200	$26	$4,294
6	28-Feb-18	BC	Table	3	$800	$104	$2,712
7	02-Mar-18	QC	Light	7	$200	$26	$1,582
8	05-Mar-18	QC	Chair	24	$300	$39	$8,136
9	07-Mar-18	ON	Chair	16	$300	$39	$5,424
10	26-Mar-18	ON	Light	9	$200	$26	$2,034
11	02-Apr-18	ON	Table	5	$800	$104	$4,520
12	03-Apr-18	QC	Light	14	$200	$26	$3,164
13	12-Apr-18	BC	Chair	20	$300	$39	$6,780
14							

Fig. 1.11 Example of a data sheet in Microsoft Excel



Sum of Total	Column Labels			
Row Labels	Chair	Light	Table	Grand Total
BC	6780	4294	2712	13786
ON	5424	2034	7232	14690
QC	12204	4746	5424	22374
Grand Total	24408	11074	15368	50850

PivotTable Fields

Choose fields to add to report:

- [✓] Product
- [ ] Units sold
- [ ] Unit Price
- [ ] Tax
- [✓] Total
- [ ] Months

Drag fields between areas below:

Filters:
Columns: Product
Rows: Region
Values: Sum of Total

Fig. 1.12 Example of a pivot table in Microsoft Excel that summarizes data from Fig. 1.11. On the right, we can notice the pivot table fields that allow users to control which summaries are being computed and displayed

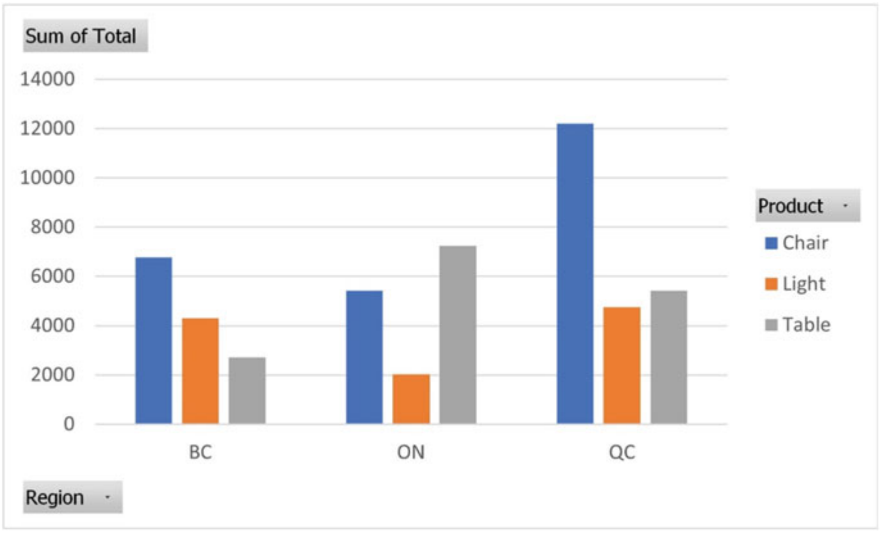


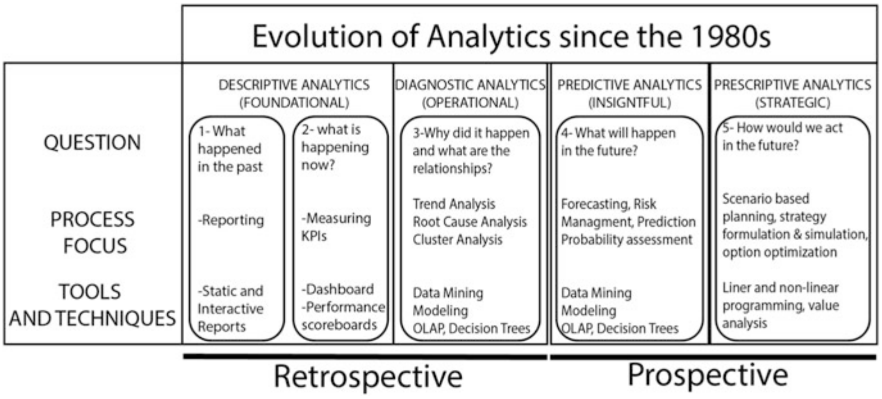
Fig. 1.13 Example of a column chart in Microsoft Excel that visualizes data from Fig. 1.12

1.5.2.1.3 Predictive Analytics

Predictive analysis uses past data to create a model that answers the question, “What will happen?”; it analyzes trends in historical data and identifies what is likely to happen in the future. Using predictive analytics, users can prepare plans and proactively implement corrective actions in advance of the occurrence of an event [39]. Some of the techniques used are what-if analysis, predictive modeling [40–42], machine learning algorithms [43–45], and neural network algorithms [46, 47]. Predictive analytics can be used for forecasting and resource planning. Predictive analytics share many basic concepts and techniques, like algorithms, with machine learning, which is covered in detail later in this textbook.

1.5.2.1.4 Prescriptive Analytics

While predictive analytics estimate what may happen in the future, prescriptive analytics goes a step further by prescribing a certain action plan to address the problems revealed by diagnostic analytics and increase the likelihood of the occurrence of the desired outcome (which may not have been forecasted by predictive analytics) [39, 48–50]. Prescriptive analytics encompasses *simulating*, evaluating several *what-if scenarios,* and advising how to maximize the likelihood of the occurrence of desired outcomes. Some of the techniques used in prescriptive analytics are graph analysis, simulation [51–53], stochastic optimization [54–56], and nonlinear programming [57–59]. Prescriptive analytics is beneficial for advising a course of action to reach a desirable goal.



Evolution of Analytics since the 1980s

	DESCRIPTIVE ANALYTICS (FOUNDATIONAL)		DIAGNOSTIC ANALYTICS (OPERATIONAL)	PREDICTIVE ANALYTICS (INSIGNTFUL)	PRESCRIPTIVE ANALYTICS (STRATEGIC)
QUESTION	1- What happened in the past	2- what is happening now?	3-Why did it happen and what are the relationships?	4- What will happen in the future?	5- How would we act in the future?
PROCESS FOCUS	-Reporting	-Measuring KPIs	Trend Analysis Root Cause Analysis Cluster Analysis	Forecasting, Risk Managment, Prediction Probability assessment	Scenario based planning, strategy formulation & simulation, option optimization
TOOLS AND TECHNIQUES	-Static and Interactive Reports	-Dashboard -Performance scoreboards	Data Mining Modeling OLAP, Decision Trees	Data Mining Modeling OLAP, Decision Trees	Liner and non-linear programming, value analysis
	Retrospective			Prospective	

Fig. 1.14 Analytics: questions, focus, and tools (adapted from Podolak [60])

Prescriptive analytics go beyond prediction to prescribe an optimal course of action to reach a certain goal based on predictions of future events. A simple example would be an app that predicts the duration of a journey from a current location to certain destinations; if the app is equipped with prescriptive analytics, then it can prescribe the shortest path to reach the destination after comparing several alternative routes [13] (Fig. 1.14).

1.6 Conclusion

Machine learning has proven itself to be a sustainable and useful technology in today's world, and its use is increasing every single day. Everything from smart devices to sophisticated automated systems such as self-driving cars uses machine learning in order to operate. Our progressively complex world is better understood with machine learning because we are currently exposed to more information than ever before and it will only continue growing [1]. In this chapter, we introduced the concept of machine learning and its origins, applications, and building blocks. In the following chapters, we elaborate more on the concept and explore in depth its different algorithms.

1.7 Key Terms

1. Machine learning
2. Artificial intelligence
3. Parallel computing
4. Distributed computing

5. Graphics processing units (GPU)
6. Big data
7. Transaction processing systems (TPS)
8. Online transaction processing systems (OLTP)
9. Online analytical processing (OLAP)
10. Data variety
11. Data velocity
12. Data veracity
13. Databases
14. Data warehouses
15. Data marts
16. Data slicing
17. Data dicing
18. Analytics
19. Descriptive analytics
20. Diagnostic analytics
21. Predictive analytics
22. Prescriptive analytics

1.8 Test Your Understanding

1. Write a definition of analytics.
2. How are descriptive analytics different than diagnostic analytics?
3. How are diagnostic analytics different than predictive analytics?
4. What is data slicing?
5. When do we use data dicing?
6. Which system is focused on daily business processes: OLTP or OLAP?
7. Enumerate five advantages of big data in healthcare.
8. Choose a sector of society and specify five advantages of the use of AI and machine learning in that sector.
9. Choose a sector of society and specify five disadvantages of the use of AI and machine learning in that sector.
10. Can AI be a source of bias? How? Search for examples in the literature.

1.9 Read More

1. Biswas, R. (2021). Outlining Big Data Analytics in Health Sector with Special Reference to Covid-19. Wirel Pers Commun, 1–12. https://doi.org/10.1007/s11277-021-09446-4
2. Clark, C. R., Wilkins, C. H., Rodriguez, J. A., Preininger, A. M., Harris, J., DesAutels, S., Karunakaram, H., Rhee, K., Bates, D. W., & Dankwa-Mullan, I. (2021). Health Care Equity in the Use of Advanced Analytics and Artificial

Intelligence Technologies in Primary Care. J Gen Intern Med, 36(10), 3188–3193. https://doi.org/10.1007/s11606-021-06846-x

3. El Morr, C., & Ali-Hassan, H. (2019). Analytics in Healthcare: A Practical Introduction. Springer.
4. IBM. (2022). What are healthcare analytics? IBM. Retrieved May, 10, 2022 from https://www.ibm.com/topics/healthcare-analyticsKhalid, S., Yang, C., Blacketer, C., Duarte-Salles, T., Fernández-Bertolín, S., Kim, C., Park, R. W., Park, J., Schuemie, M. J., Sena, A. G., Suchard, M. A., You, S. C., Rijnbeek, P. R., & Reps, J. M. (2021). A standardized analytics pipeline for reliable and rapid development and validation of prediction models using observational health data. Comput Methods Programs Biomed, 211, 106,394. https://doi.org/10.1016/j.cmpb.2021.106394
5. Lopez, L., Chen, K., Hart, L., & Johnson, A. K. (2021). Access and Analytics: What the Military Can Teach Us About Health Equity. Am J Public Health, 111(12), 2089–2090. https://doi.org/10.2105/ajph.2021.306535
6. Moreno-Fergusson, M. E., Guerrero Rueda, W. J., Ortiz Basto, G. A., Arevalo Sandoval, I. A. L., & Sanchez-Herrera, B. (2021). Analytics and Lean Health Care to Address Nurse Care Management Challenges for Inpatients in Emerging Economies. J Nurs Scholarsh, 53(6), 803–814. https://doi.org/10.1111/jnu.12711
7. Mukherjee, S., Frimpong Boamah, E., Ganguly, P., & Botchwey, N. (2021). A multilevel scenario based predictive analytics framework to model the community mental health and built environment nexus. Sci Rep, 11(1), 17,548. https://doi.org/10.1038/s41598-021-96801-x
8. Qiao, S., Li, X., Olatosi, B., & Young, S. D. (2021). Utilizing Big Data analytics and electronic health record data in HIV prevention, treatment, and care research: a literature review. AIDS Care, 1–21. https://doi.org/10.1080/09540121.2021.1948499

1.10 Lab

All instructions will be for Windows users; Mac users can follow overall the same instructions.

1.10.1 Introduction to R

R is an open-source integrated development environment (IDE) used for statistical analysis. This section describes step-by-step instructions to download and install R v4.1.0 and RStudio IDE v1.4.1717.

R can be downloaded and installed from the following location: https://www.r-project.org/. Below are instructions for R's download and installation.

1. Go to the following mirror location and download R (The Comprehensive R Archive Network (sfu.ca)). For Windows users, click on "Download R for Windows"; for other operating systems, click on the corresponding link (Fig. 1.15).

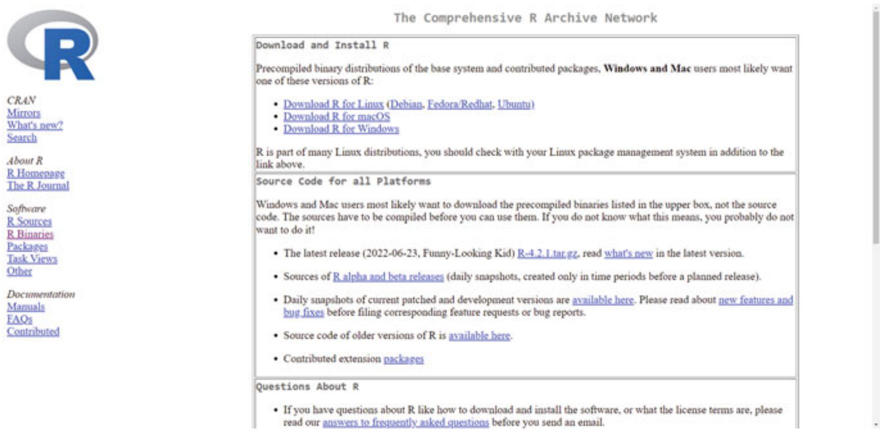
2. While we will demonstrate the installation for Windows, the installation for macOS is similar. For Windows, click on the "install R for the first time" link as shown in Fig. 1.16:

3. Click on the "Download R 4.1.0 Windows" link (Fig. 1.17):

4. R-4.1.0-win.exe will be installed into the Downloads folder. Click on the R-4.1.0-win.exe file to start the installation and continue by clicking the "Next" button to complete the installation (Fig. 1.18).

5. If a shortcut for R was not automatically created on your desktop, you can always create one. On Windows, go to the following location C:\Program Files \R-4.1.0 (on a Mac, go to the Applications folder) (Fig. 1.19), right click on R. exe, and choose "Create Shortcut" (or "Make Alias" for Mac users). A new shortcut will be created; move it to your desktop. This will enable you to launch R easily from the desktop.

6. Double-click on the "R" icon and launch the R software; a new command prompt appears (Fig. 1.20):



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

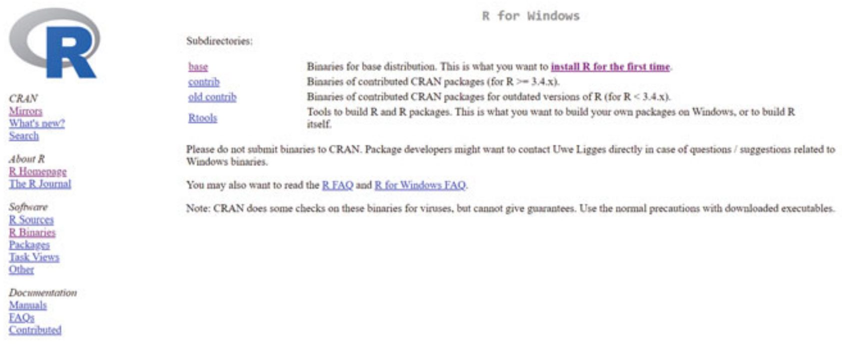
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2022-06-23, Funny-Looking Kid) R-4.2.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Fig. 1.15 R Installation for Windows users and Mac users



R for Windows

CRAN
Mirrors
What's new?
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Subdirectories:

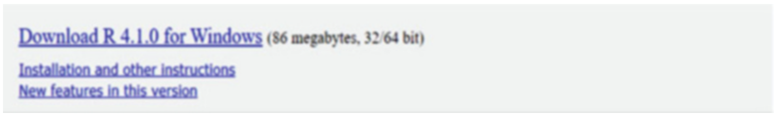
base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

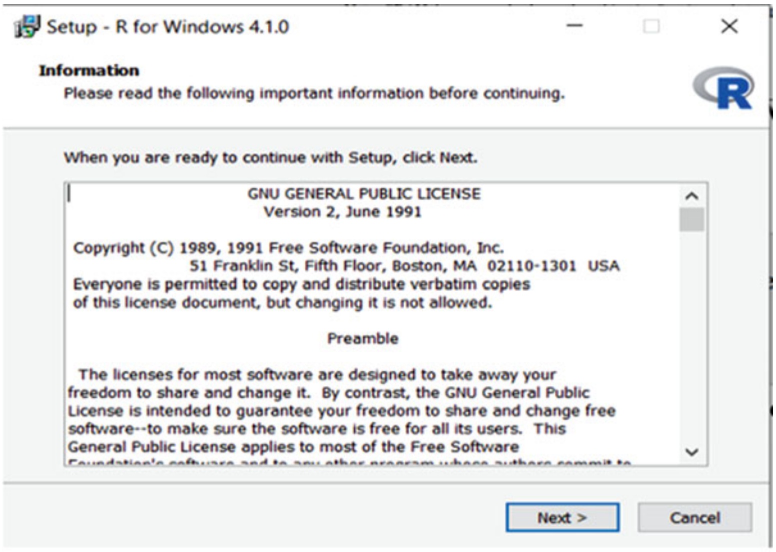
Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Fig. 1.16 First-time R installation



Download R 4.1.0 for Windows (86 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

Fig. 1.17 R v4.1.0 for Windows installation



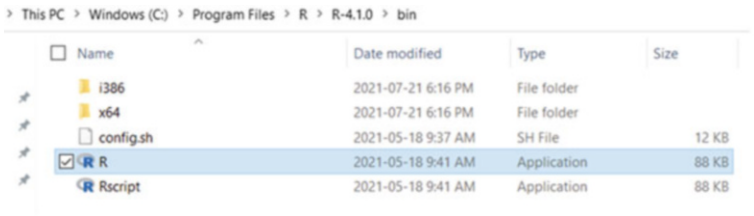
Setup - R for Windows 4.1.0

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
```

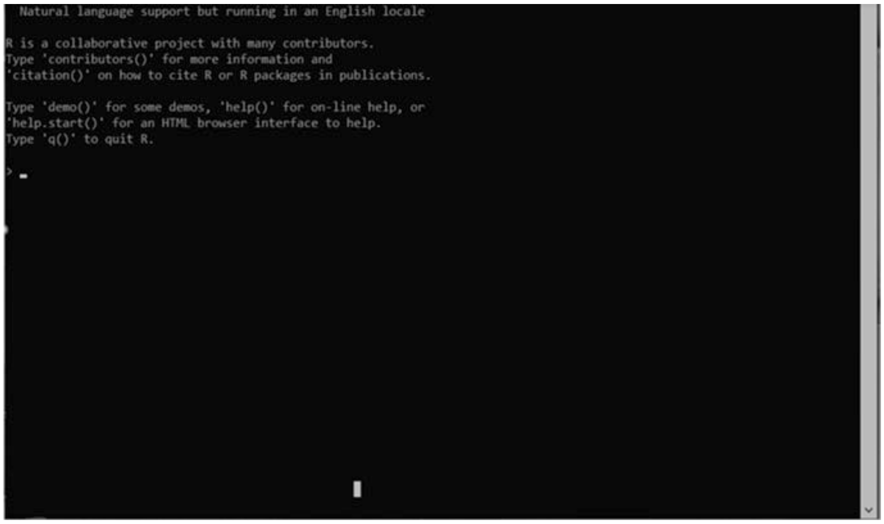
Fig. 1.18 R setup for Windows user



```
> This PC > Windows (C:) > Program Files > R > R-4.1.0 > bin
```

Name	Date modified	Type	Size
i386	2021-07-21 6:16 PM	File folder	
x64	2021-07-21 6:16 PM	File folder	
config.sh	2021-05-18 9:37 AM	SH File	12 KB
R	2021-05-18 9:41 AM	Application	88 KB
Rscript	2021-05-18 9:41 AM	Application	88 KB

Fig. 1.19 R installation local directory



```
 Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Fig. 1.20 Launching R application

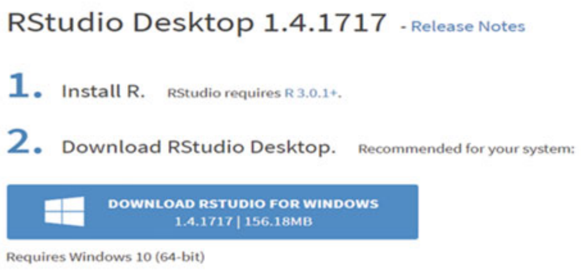
1.10.2 Introduction to RStudio

RStudio v1.4.1717 is the IDE for the R language. It includes a workspace for coding, debugging, plotting, etc. It can be installed from the following location: Download the RStudio IDE—RStudio. Below are instructions for RStudio's download and installation.

1.10.2.1 RStudio Download and Installation

1. Download RStudio: Click on "Download RStudio for Windows." RStudio is available for Mac users as well on the same webpage (Fig. 1.21).

2. Double-click on "RStudio-1.4.1717.exe" and click on "Setup RStudio v1.4.1717"; the installation will start (Fig. 1.22).

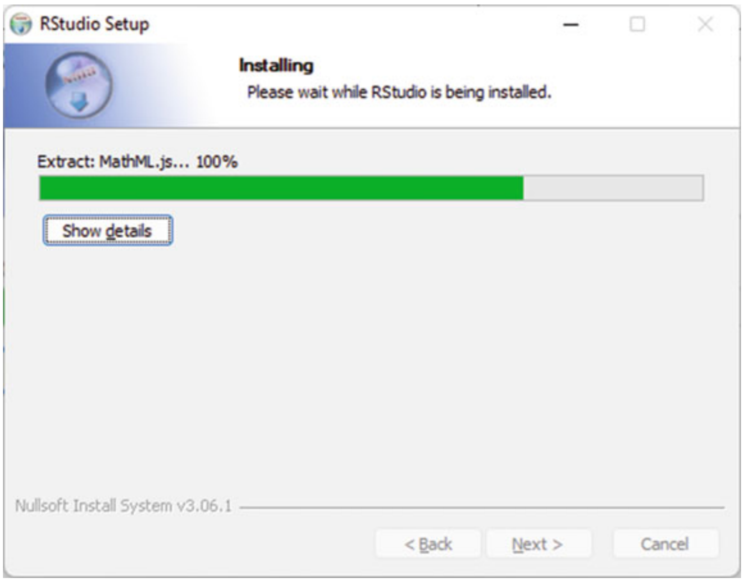

RStudio Desktop 1.4.1717 - Release Notes

1. Install R. RStudio requires R 3.0.1+.

2. Download RStudio Desktop. Recommended for your system:

Requires Windows 10 (64-bit)

Fig. 1.21 Installing RStudio Desktop 1.4.1717


RStudio Setup

Installing

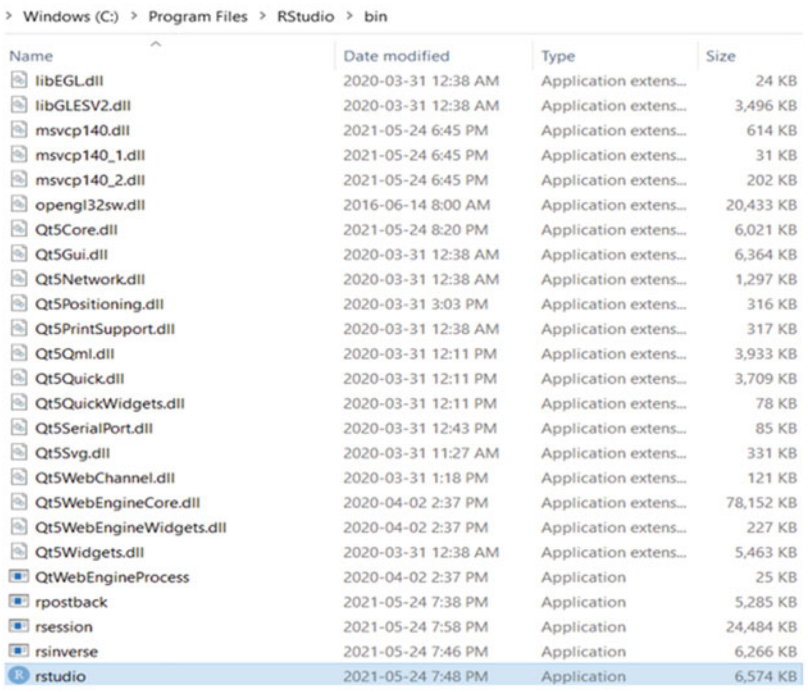
Please wait while RStudio is being installed.

Extract: MathML.js... 100%

Fig. 1.22 RStudio setup

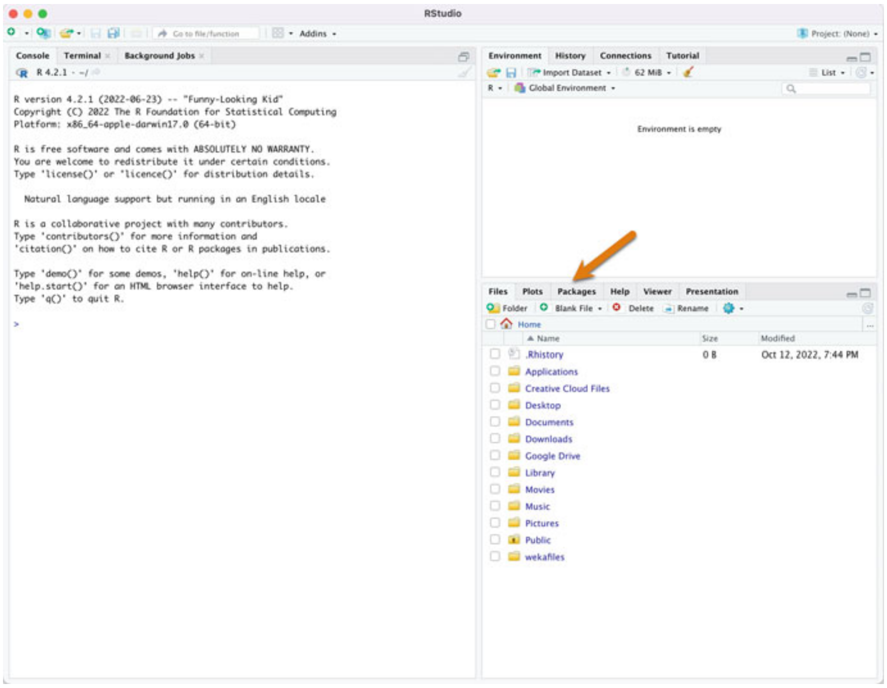
3. If a shortcut for RStudio was not automatically created on your desktop, you can always create one. On Windows, go to the following location c:\Program files \RStudio\Bin (go to the Applications folder if you are using macOS) (Fig. 1.23).

4. After launching the RStudio application, its IDE will appear as shown in Fig. 1.24:



Name	Date modified	Type	Size
libEGL.dll	2020-03-31 12:38 AM	Application extens...	24 KB
libGLESV2.dll	2020-03-31 12:38 AM	Application extens...	3,496 KB
msvcp140.dll	2021-05-24 6:45 PM	Application extens...	614 KB
msvcp140_1.dll	2021-05-24 6:45 PM	Application extens...	31 KB
msvcp140_2.dll	2021-05-24 6:45 PM	Application extens...	202 KB
opengl32sw.dll	2016-06-14 8:00 AM	Application extens...	20,433 KB
Qt5Core.dll	2021-05-24 8:20 PM	Application extens...	6,021 KB
Qt5Gui.dll	2020-03-31 12:38 AM	Application extens...	6,364 KB
Qt5Network.dll	2020-03-31 12:38 AM	Application extens...	1,297 KB
Qt5Positioning.dll	2020-03-31 3:03 PM	Application extens...	316 KB
Qt5PrintSupport.dll	2020-03-31 12:38 AM	Application extens...	317 KB
Qt5Qml.dll	2020-03-31 12:11 PM	Application extens...	3,933 KB
Qt5Quick.dll	2020-03-31 12:11 PM	Application extens...	3,709 KB
Qt5QuickWidgets.dll	2020-03-31 12:11 PM	Application extens...	78 KB
Qt5SerialPort.dll	2020-03-31 12:43 PM	Application extens...	85 KB
Qt5Svg.dll	2020-03-31 11:27 AM	Application extens...	331 KB
Qt5WebChannel.dll	2020-03-31 1:18 PM	Application extens...	121 KB
Qt5WebEngineCore.dll	2020-04-02 2:37 PM	Application extens...	78,152 KB
Qt5WebEngineWidgets.dll	2020-04-02 2:37 PM	Application extens...	227 KB
Qt5Widgets.dll	2020-03-31 12:38 AM	Application extens...	5,463 KB
QtWebEngineProcess	2020-04-02 2:37 PM	Application	25 KB
rpostback	2021-05-24 7:38 PM	Application	5,285 KB
rsession	2021-05-24 7:58 PM	Application	24,484 KB
rsinverse	2021-05-24 7:46 PM	Application	6,266 KB
rstudio	2021-05-24 7:48 PM	Application	6,574 KB

Fig. 1.23 Creating a shortcut for the RStudio application



```
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
.Rhistory	0 B	Oct 12, 2022, 7:44 PM
Applications		
Creative Cloud Files		
Desktop		
Documents		
Downloads		
Google Drive		
Library		
Movies		
Music		
Pictures		
Public		
wekafiles		

Fig. 1.24 Launching RStudio IDE

1.10.2.2 Install a Package

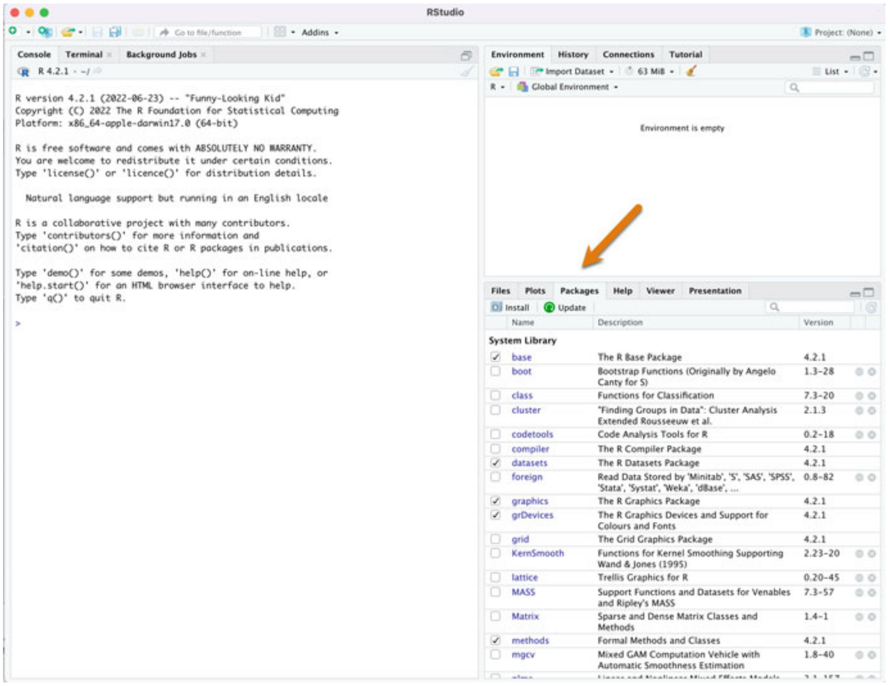
Packages are libraries that allow us to do specific tasks in RStudio (e.g., load a file, display a result, do an analysis). A package can be installed in the RStudio console using the following instructions:

1. Click on the Packages tab and click on the Install button (Fig. 1.25):

2. Install the readr package by typing “readr” and clicking on the Install button (Fig. 1.26).

1.10.2.3 Activate Package

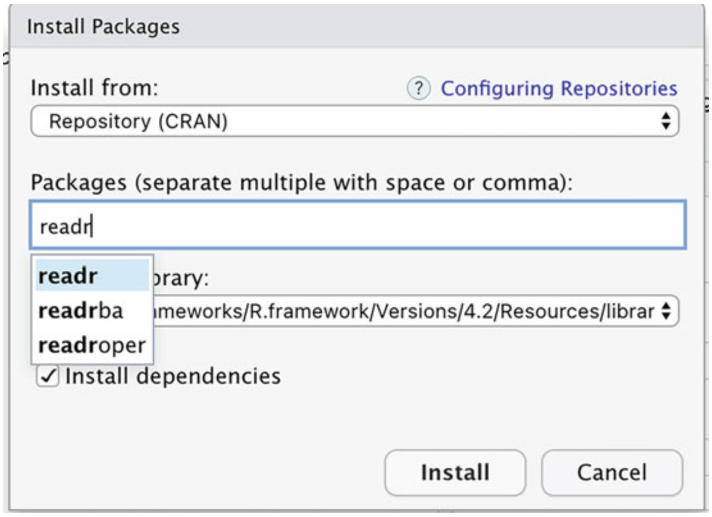
1. To activate the readr library used to read csv and txt files, type “library(readr)” (Fig. 1.27):



```
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

	Name	Description	Version
System Library			
✓	base	The R Base Package	4.2.1
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
☐	class	Functions for Classification	7.3-20
☐	cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.1.3
☐	codetools	Code Analysis Tools for R	0.2-18
☐	compiler	The R Compiler Package	4.2.1
✓	datasets	The R Datasets Package	4.2.1
☐	foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-82
✓	graphics	The R Graphics Package	4.2.1
✓	grDevices	The R Graphics Devices and Support for Colours and Fonts	4.2.1
☐	grid	The Grid Graphics Package	4.2.1
☐	KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-20
☐	lattice	Trellis Graphics for R	0.20-45
☐	MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-57
☐	Matrix	Sparse and Dense Matrix Classes and Methods	1.4-1
✓	methods	Formal Methods and Classes	4.2.1
☐	mgcv	Mixed GAM Computation Vehicle with Automatic Smoothness Estimation	1.8-40

Fig. 1.25 Navigating RStudio Packages tab



Install Packages

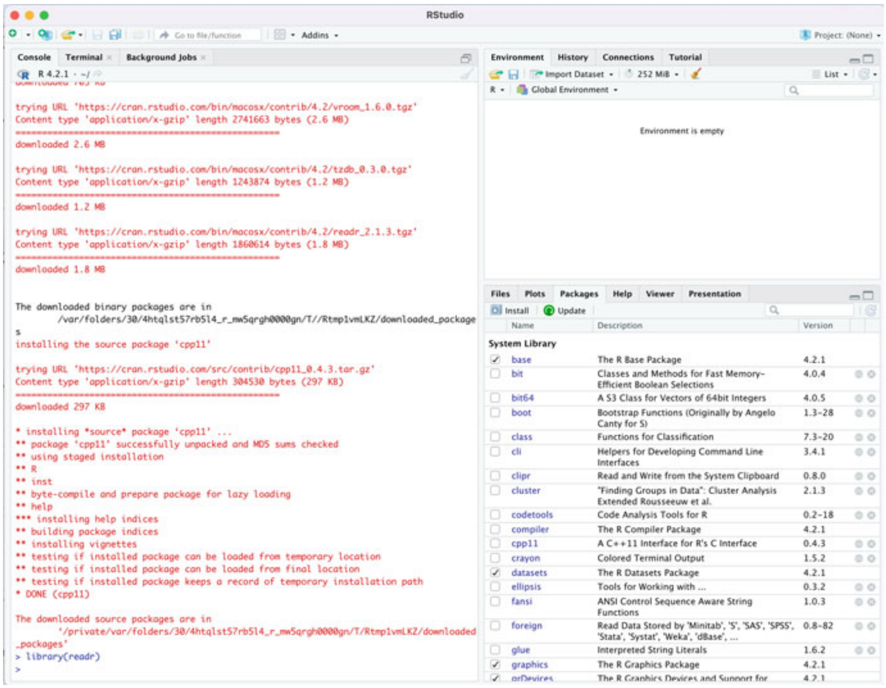
Install from: Repository (CRAN)

Packages (separate multiple with space or comma): readr

Install to library: …meworks/R.framework/Versions/4.2/Resources/librar

✓ Install dependencies

Fig. 1.26 Installing readr package for loading files



```
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.2/vroom_1.6.0.tgz'
Content type 'application/x-gzip' length 2741663 bytes (2.6 MB)
==================================================
downloaded 2.6 MB

trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.2/tzdb_0.3.0.tgz'
Content type 'application/x-gzip' length 1243874 bytes (1.2 MB)
==================================================
downloaded 1.2 MB

trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.2/readr_2.1.3.tgz'
Content type 'application/x-gzip' length 1860614 bytes (1.8 MB)
==================================================
downloaded 1.8 MB


The downloaded binary packages are in
        /var/folders/30/4htqlst57rb514_r_mw5qrgh0000gn/T//Rtmp1vmLKZ/downloaded_package
s
installing the source package 'cpp11'

trying URL 'https://cran.rstudio.com/src/contrib/cpp11_0.4.3.tar.gz'
Content type 'application/x-gzip' length 304530 bytes (297 KB)
==================================================
downloaded 297 KB

* installing *source* package 'cpp11' ...
** package 'cpp11' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (cpp11)

The downloaded source packages are in
        '/private/var/folders/30/4htqlst57rb514_r_mw5qrgh0000gn/T/Rtmp1vmLKZ/downloaded
_packages'
> library(readr)
>
```

	Name	Description	Version
	System Library		
✓	base	The R Base Package	4.2.1
☐	bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.4
☐	bit64	A S3 Class for Vectors of 64bit Integers	4.0.5
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
☐	class	Functions for Classification	7.3-20
☐	cli	Helpers for Developing Command Line Interfaces	3.4.1
☐	clipr	Read and Write from the System Clipboard	0.8.0
☐	cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.1.3
☐	codetools	Code Analysis Tools for R	0.2-18
☐	compiler	The R Compiler Package	4.2.1
☐	cpp11	A C++11 Interface for R's C Interface	0.4.3
☐	crayon	Colored Terminal Output	1.5.2
✓	datasets	The R Datasets Package	4.2.1
☐	ellipsis	Tools for Working with ...	0.3.2
☐	fansi	ANSI Control Sequence Aware String Functions	1.0.3
☐	foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-82
☐	glue	Interpreted String Literals	1.6.2
✓	graphics	The R Graphics Package	4.2.1
✓	grDevices	The R Graphics Devices and Support for	4.2.1

Fig. 1.27 Activating readr package for use

1.10.2.4 User Readr to Load Data

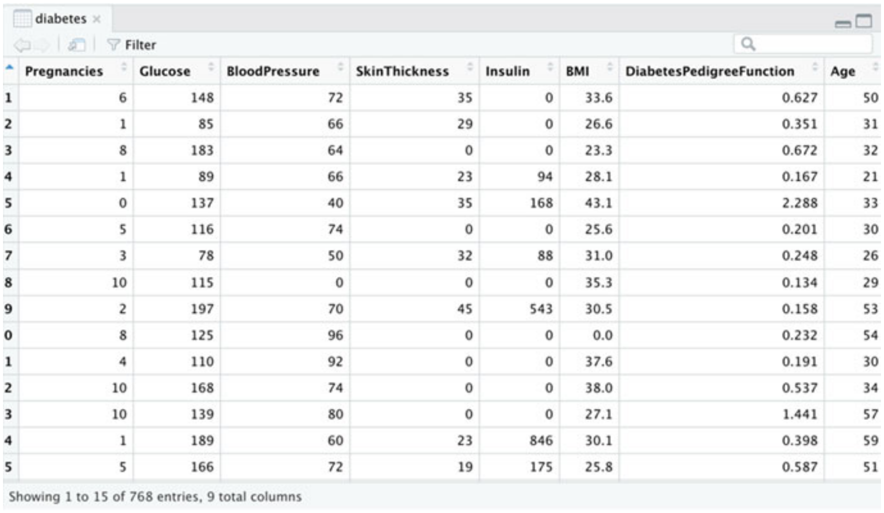
Different dataset types, such as txt, csv, and xlsx, can be imported into RStudio files (Fig. 1.28)

1. Download Diabetes.csv: Go to the following link https://www.kaggle.com/uciml/pima-indians-diabetes-database/version/1 (or go to kaggle.com and search for "Pima Indians Diabetes Database").
2. Next, you will load the diabetes.csv file and plot it as a histogram. Under the main menu, click on the File menu/Import dataset/From text readr. In case you are prompted to download a library, accept to download it. Choose the input file from the folder where you have saved it and click Import.

1.10.2.5 Run a Function

The hist function can be used to visualize the data in a histogram. The hist function can present the data imported earlier (blood pressure vs. age) in a histogram (Fig. 1.29). Type the following:

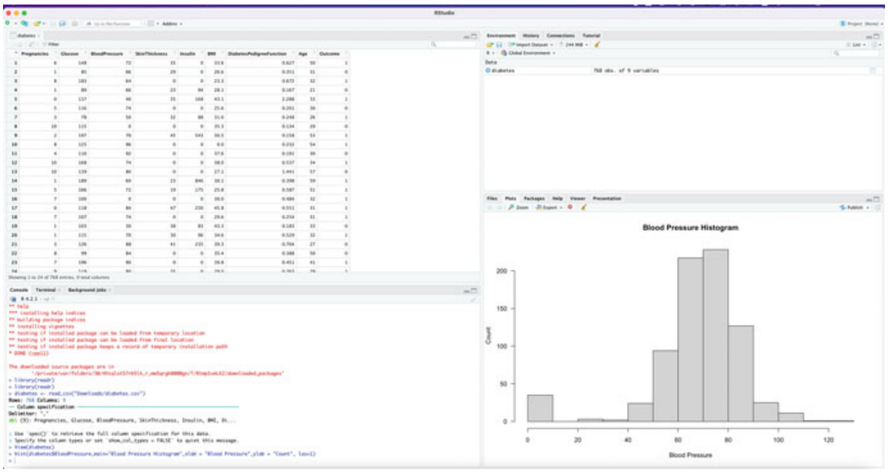
```
hist(diabetes$BloodPressure,main="Blood Pressure Histogram",xlab =
"Blood Pressure",ylab = "Count", las=1).
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
1	6	148	72	35	0	33.6	0.627	50
2	1	85	66	29	0	26.6	0.351	31
3	8	183	64	0	0	23.3	0.672	32
4	1	89	66	23	94	28.1	0.167	21
5	0	137	40	35	168	43.1	2.288	33
6	5	116	74	0	0	25.6	0.201	30
7	3	78	50	32	88	31.0	0.248	26
8	10	115	0	0	0	35.3	0.134	29
9	2	197	70	45	543	30.5	0.158	53
0	8	125	96	0	0	0.0	0.232	54
1	4	110	92	0	0	37.6	0.191	30
2	10	168	74	0	0	38.0	0.537	34
3	10	139	80	0	0	27.1	1.441	57
4	1	189	60	23	846	30.1	0.398	59
5	5	166	72	19	175	25.8	0.587	51

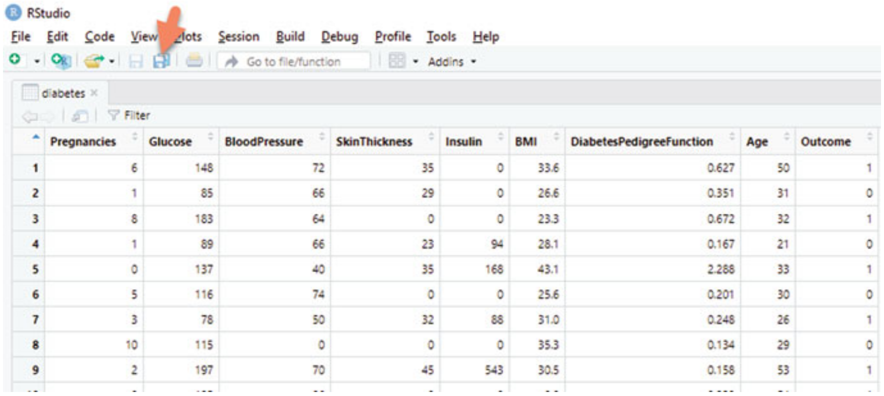
Showing 1 to 15 of 768 entries, 9 total columns

Fig. 1.28 Importing dataset using the readr package



```
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (cpp11)

The downloaded source packages are in
	'/private/var/folders/[illegible]/downloaded_packages'
> library(readr)
> library(readr)
> diabetes <- read_csv("Downloads/diabetes.csv")
Rows: 768 Columns: 9
── Column specification ──────────────────────────
Delimiter: ","
dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, Di...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> View(diabetes)
> hist(diabetes$BloodPressure,main="Blood Pressure Histogram",xlab = "Blood Pressure",ylab = "Count", las=1)
> 
```

Fig. 1.29 Visualize data in a histogram



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31.0	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	1

Fig. 1.30 Save progress status in RStudio

1.10.2.6 Save Status

To save your progress, use the ctrl + S shortcut or click the blue Save button in the main menu (Fig. 1.30):

1.10.3 Introduction to Python and Jupyter Notebook IDE

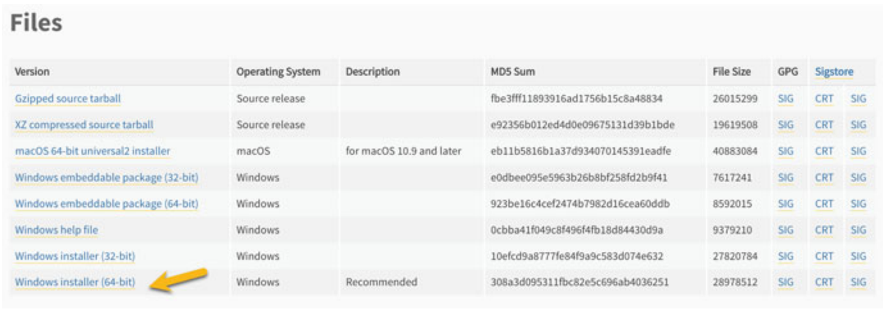
Python is an interpreted programming language. It is characterized by human readability; it is an important programming language in machine learning and artificial intelligence due to its flexible libraries and ease of use.

In this book, Jupyter Notebook IDE is used for Python labs and examples.

1.10.3.1 Python Download and Installation

Python programming language can be installed from the following location: Python.org. Below are instructions for python download and installation for Python v3.9.6.

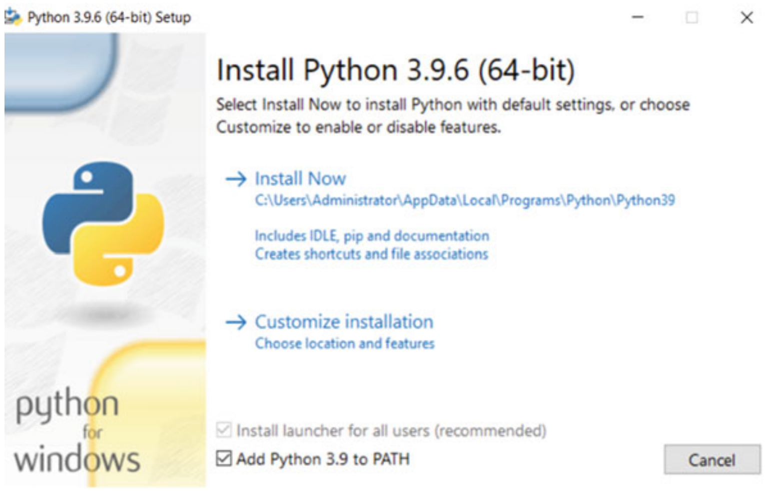
1. Download the Windows installer to your computer location (Fig. 1.31).
2. Install Python v3.9.6 (64-bit) using the “Customize installation” option; please follow the screenshots carefully by checking the right checkboxes as indicated (Figs. 1.32, 1.33, and 1.34).
3. To validate the installation, open a terminal (open “cmd” in Windows; open a terminal in macOS) and type “*python -v*.”



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore	
Gzipped source tarball	Source release		fbe3fff11893916ad1756b15c8a48834	26015299	SIG	CRT	SIG
XZ compressed source tarball	Source release		e92356b012ed4d0e09675131d39b1bde	19619508	SIG	CRT	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	eb11b5816b1a37d934070145391eadfe	40883084	SIG	CRT	SIG
Windows embeddable package (32-bit)	Windows		e0dbee095e5963b26b8bf258fd2b9f41	7617241	SIG	CRT	SIG
Windows embeddable package (64-bit)	Windows		923be16c4cef2474b7982d16cea60ddb	8592015	SIG	CRT	SIG
Windows help file	Windows		0cbba41f049c8f496f4fb18d84430d9a	9379210	SIG	CRT	SIG
Windows installer (32-bit)	Windows		10efcd9a8777fe84f9a9c583d074e632	27820784	SIG	CRT	SIG
Windows installer (64-bit)	Windows	Recommended	308a3d095311fbc82e5c696ab4036251	28978512	SIG	CRT	SIG

Fig. 1.31 Install Python v3.9.6 Windows installer



Python 3.9.6 (64-bit) Setup

Install Python 3.9.6 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

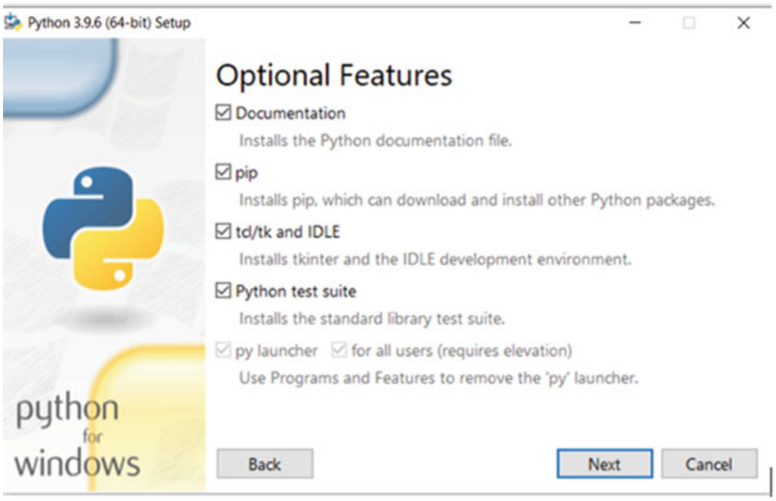
→ Install Now
C:\Users\Administrator\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

Fig. 1.32 Setting up Python 3.9.6


Python 3.9.6 (64-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

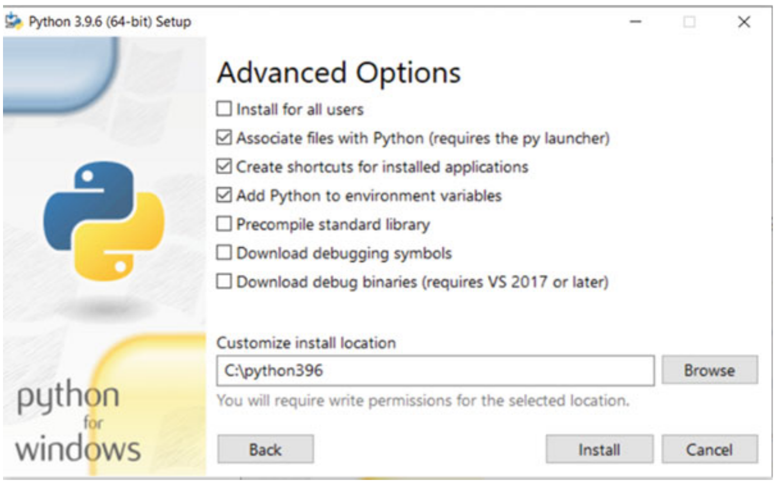
☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Use Programs and Features to remove the 'py' launcher.

Fig. 1.33 Configuring Python 3.9.6 features


Python 3.9.6 (64-bit) Setup

Advanced Options

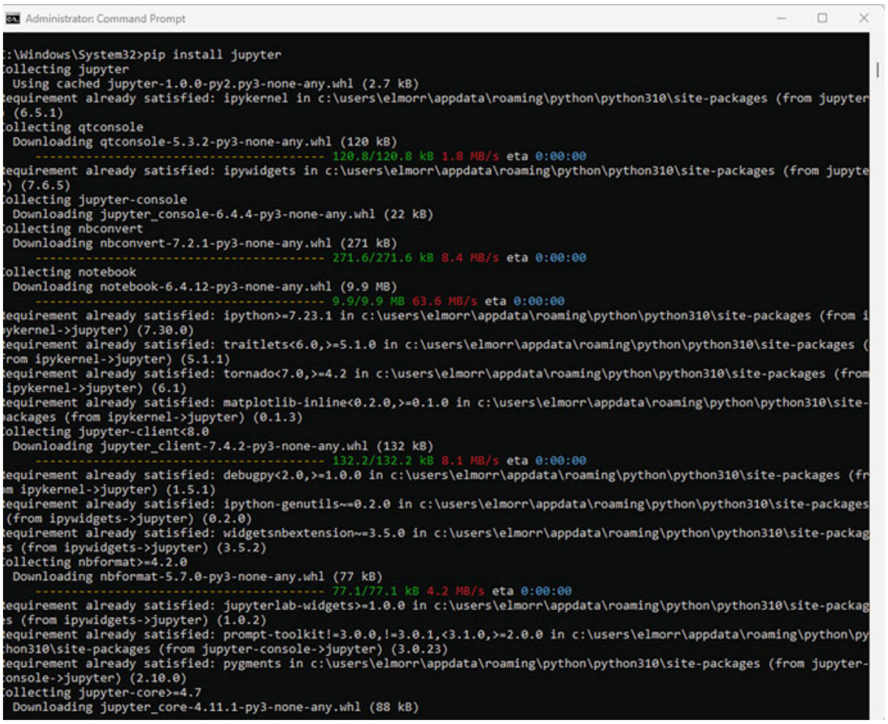
☐ Install for all users
☑ Associate files with Python (requires the py launcher)
☑ Create shortcuts for installed applications
☑ Add Python to environment variables
☐ Precompile standard library
☐ Download debugging symbols
☐ Download debug binaries (requires VS 2017 or later)

Customize install location
C:\python396
You will require write permissions for the selected location.

Fig. 1.34 Configuring more Python v3.9.6 installation options

1.10.3.2 Jupyter Download and Installation

1. To install "Jupyter Notebook" IDE, open a terminal and type "*pip install jupyter*"; if the command does not work then type "*pip3 install jupyter*" (Fig. 1.35).



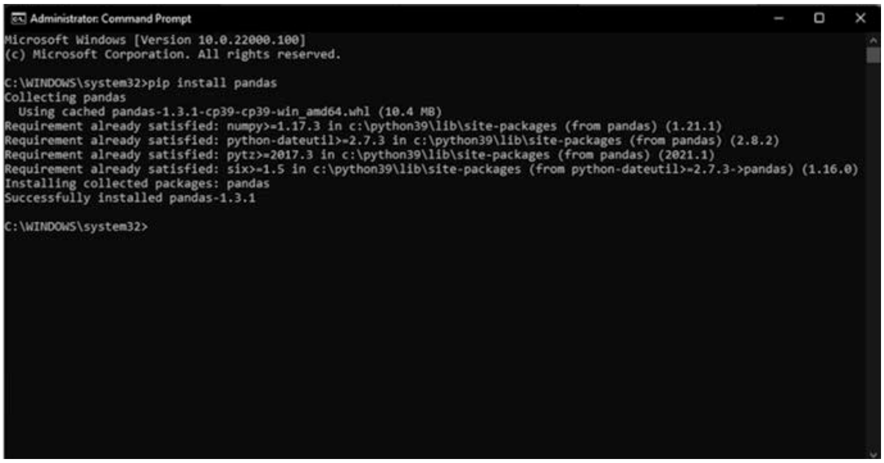
```
:\Windows\System32>pip install jupyter
ollecting jupyter
 Using cached jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
equirement already satisfied: ipykernel in c:\users\elmorr\appdata\roaming\python\python310\site-packages (from jupyter
 (6.5.1)
ollecting qtconsole
 Downloading qtconsole-5.3.2-py3-none-any.whl (120 kB)
    ---------------------------------------- 120.8/120.8 kB 1.8 MB/s eta 0:00:00
equirement already satisfied: ipywidgets in c:\users\elmorr\appdata\roaming\python\python310\site-packages (from jupyte
) (7.6.5)
ollecting jupyter-console
 Downloading jupyter_console-6.4.4-py3-none-any.whl (22 kB)
ollecting nbconvert
 Downloading nbconvert-7.2.1-py3-none-any.whl (271 kB)
    ---------------------------------------- 271.6/271.6 kB 8.4 MB/s eta 0:00:00
ollecting notebook
 Downloading notebook-6.4.12-py3-none-any.whl (9.9 MB)
    ---------------------------------------- 9.9/9.9 MB 63.6 MB/s eta 0:00:00
equirement already satisfied: ipython>=7.23.1 in c:\users\elmorr\appdata\roaming\python\python310\site-packages (from i
ykernel->jupyter) (7.30.0)
equirement already satisfied: traitlets<6.0,>=5.1.0 in c:\users\elmorr\appdata\roaming\python\python310\site-packages (
rom ipykernel->jupyter) (5.1.1)
equirement already satisfied: tornado<7.0,>=4.2 in c:\users\elmorr\appdata\roaming\python\python310\site-packages (from
 ipykernel->jupyter) (6.1)
equirement already satisfied: matplotlib-inline<0.2.0,>=0.1.0 in c:\users\elmorr\appdata\roaming\python\python310\site-
ackages (from ipykernel->jupyter) (0.1.3)
ollecting jupyter-client<8.0
 Downloading jupyter_client-7.4.2-py3-none-any.whl (132 kB)
    ---------------------------------------- 132.2/132.2 kB 8.1 MB/s eta 0:00:00
equirement already satisfied: debugpy<2.0,>=1.0.0 in c:\users\elmorr\appdata\roaming\python\python310\site-packages (fr
m ipykernel->jupyter) (1.5.1)
equirement already satisfied: ipython-genutils~=0.2.0 in c:\users\elmorr\appdata\roaming\python\python310\site-packages
 (from ipywidgets->jupyter) (0.2.0)
equirement already satisfied: widgetsnbextension~=3.5.0 in c:\users\elmorr\appdata\roaming\python\python310\site-packag
s (from ipywidgets->jupyter) (3.5.2)
ollecting nbformat>=4.2.0
 Downloading nbformat-5.7.0-py3-none-any.whl (77 kB)
    ---------------------------------------- 77.1/77.1 kB 4.2 MB/s eta 0:00:00
equirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\elmorr\appdata\roaming\python\python310\site-packag
s (from ipywidgets->jupyter) (1.0.2)
equirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\users\elmorr\appdata\roaming\python\py
hon310\site-packages (from jupyter-console->jupyter) (3.0.23)
equirement already satisfied: pygments in c:\users\elmorr\appdata\roaming\python\python310\site-packages (from jupyter-
onsole->jupyter) (2.10.0)
ollecting jupyter-core>=4.7
 Downloading jupyter_core-4.11.1-py3-none-any.whl (88 kB)
```

Fig. 1.35 Installing Jupyter Notebook IDE

2. We need a useful Python package called “pandas” to manipulate data. In order to use pandas, we need first to install them. To do so we will use the command pip install.
   On Windows, open a terminal (cmd) *as an administrator*; you can do so by right clicking on cmd and choosing Run As Administrator. In the terminal, type “pip install pandas” (Fig. 1.36).
   On macOS, open the terminal and write “sudo pip install pandas.”
   macOS will ask you for your password; the system assumes you have administrative powers. Enter the password and the library package will be installed (Fig. 1.37).
3. Using the same strategy, install matplot and openpyxl libraries (Fig. 1.38):

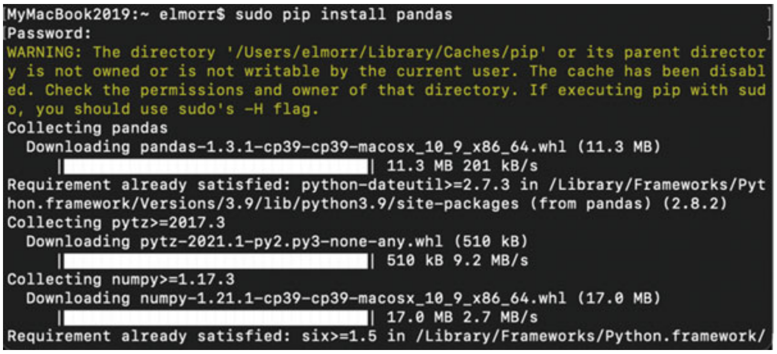
```
pip install matplotlib
pip install openpyxl
```

For MacOS, launch Jupyter Notebook by typing the following in the terminal: “Jupyter Notebook”. For windows type “python -m jupyter notebook”. Then, click “New” button and choose “Python 3” to create a notebook (Figs. 1.39 and 1.40).



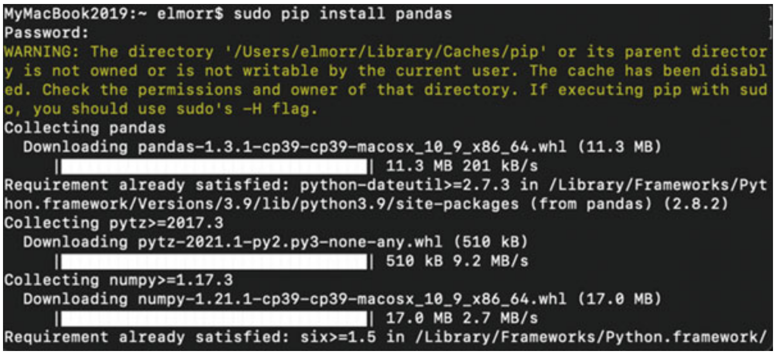
```
Microsoft Windows [Version 10.0.22000.100]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip install pandas
Collecting pandas
  Using cached pandas-1.3.1-cp39-cp39-win_amd64.whl (10.4 MB)
Requirement already satisfied: numpy>=1.17.3 in c:\python39\lib\site-packages (from pandas) (1.21.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\python39\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\python39\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: six>=1.5 in c:\python39\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Installing collected packages: pandas
Successfully installed pandas-1.3.1

C:\WINDOWS\system32>
```

Fig. 1.36 Installing pandas package to work with data frames for Windows users



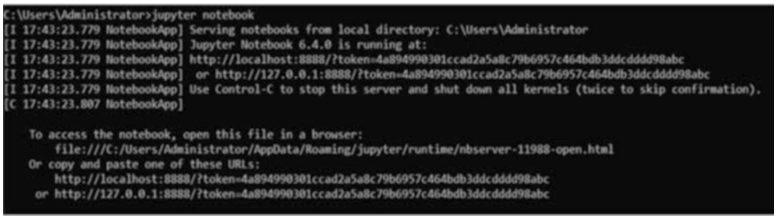
```
MyMacBook2019:~ elmorr$ sudo pip install pandas
Password:
WARNING: The directory '/Users/elmorr/Library/Caches/pip' or its parent director
y is not owned or is not writable by the current user. The cache has been disabl
ed. Check the permissions and owner of that directory. If executing pip with sud
o, you should use sudo's -H flag.
Collecting pandas
  Downloading pandas-1.3.1-cp39-cp39-macosx_10_9_x86_64.whl (11.3 MB)
     |████████████████████████████████| 11.3 MB 201 kB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from pandas) (2.8.2)
Collecting pytz>=2017.3
  Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
     |████████████████████████████████| 510 kB 9.2 MB/s
Collecting numpy>=1.17.3
  Downloading numpy-1.21.1-cp39-cp39-macosx_10_9_x86_64.whl (17.0 MB)
     |████████████████████████████████| 17.0 MB 2.7 MB/s
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/
```

Fig. 1.37 Installing pandas package to work with data frames for Mac users



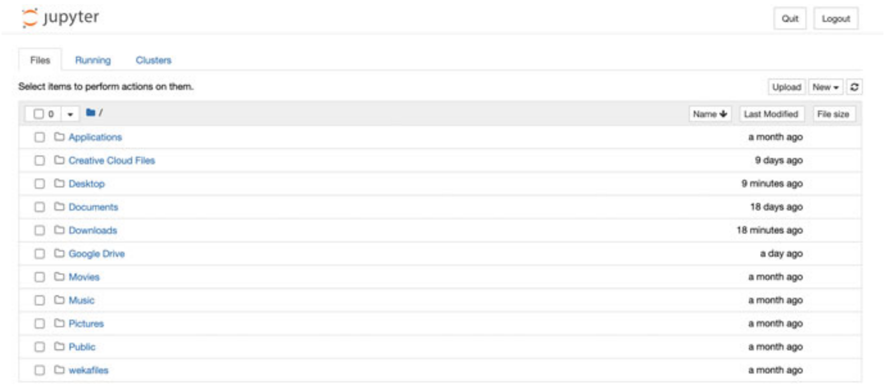
```
MyMacBook2019:~ elmorr$ sudo pip install pandas
Password:
WARNING: The directory '/Users/elmorr/Library/Caches/pip' or its parent director
y is not owned or is not writable by the current user. The cache has been disabl
ed. Check the permissions and owner of that directory. If executing pip with sud
o, you should use sudo's -H flag.
Collecting pandas
  Downloading pandas-1.3.1-cp39-cp39-macosx_10_9_x86_64.whl (11.3 MB)
     |████████████████████████████████| 11.3 MB 201 kB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from pandas) (2.8.2)
Collecting pytz>=2017.3
  Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
     |████████████████████████████████| 510 kB 9.2 MB/s
Collecting numpy>=1.17.3
  Downloading numpy-1.21.1-cp39-cp39-macosx_10_9_x86_64.whl (17.0 MB)
     |████████████████████████████████| 17.0 MB 2.7 MB/s
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/
```

Fig. 1.38 Installing matplotlib and openpyxl packages



```
C:\Users\Administrator>jupyter notebook
[I 17:43:23.779 NotebookApp] Serving notebooks from local directory: C:\Users\Administrator
[I 17:43:23.779 NotebookApp] Jupyter Notebook 6.4.0 is running at:
[I 17:43:23.779 NotebookApp] http://localhost:8888/?token=4a894990301ccad2a5a8c79b6957c464bdb3ddcdddd98abc
[I 17:43:23.779 NotebookApp]  or http://127.0.0.1:8888/?token=4a894990301ccad2a5a8c79b6957c464bdb3ddcdddd98abc
[I 17:43:23.779 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:43:23.807 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/Administrator/AppData/Roaming/jupyter/runtime/nbserver-11988-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=4a894990301ccad2a5a8c79b6957c464bdb3ddcdddd98abc
     or http://127.0.0.1:8888/?token=4a894990301ccad2a5a8c79b6957c464bdb3ddcdddd98abc
```

Fig. 1.39 Launching Jupyter Notebook IDE



Select items to perform actions on them.

Name	Last Modified
Applications	a month ago
Creative Cloud Files	9 days ago
Desktop	9 minutes ago
Documents	18 days ago
Downloads	18 minutes ago
Google Drive	a day ago
Movies	a month ago
Music	a month ago
Pictures	a month ago
Public	a month ago
wekafiles	a month ago

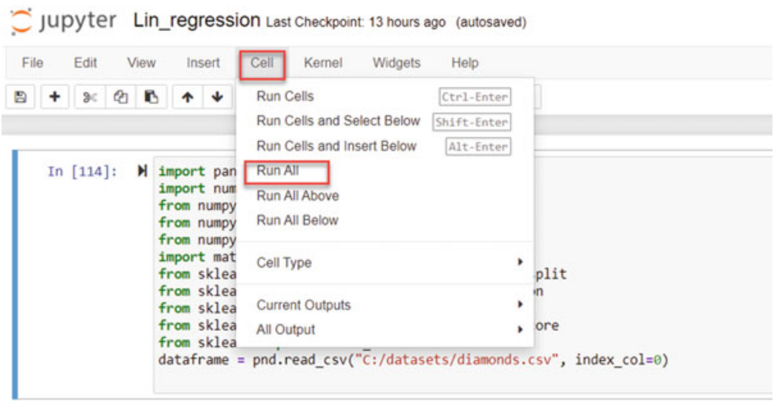
Fig. 1.40 Notebook webpage

1.10.3.3 Load Data and Plot It Visually

Python code can be added in the Jupyter Notebook IDE file, and every line can be executed using the “Run” button for every line. It is important to note that code on all lines can be run once by doing the following: Under the Cell menu, click “Run All.” This is shown below in Fig. 1.41.

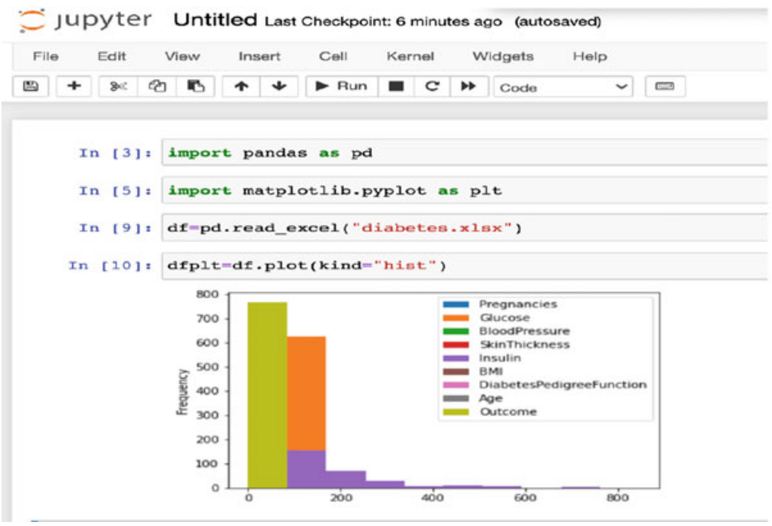
The code below allows us to read the blood pressure measurement from the diabetes.xlsx file and plot it in a histogram. Open diabetes.csv and save it as diabetes. xlsx, then follow the instructions below (Fig. 1.42):

```
Type: import pandas as pd
Type: import matplotlib.pyplot as plt
Type: df=pd.read_excel("diabetes.xlsx")
Type: dfplt=df.plot(kind="hist")
```


jupyter Lin_regression Last Checkpoint: 13 hours ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run Cells Ctrl-Enter
Run Cells and Select Below Shift-Enter
Run Cells and Insert Below Alt-Enter
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output

```
In [114]:   import pan
            import num
            from numpy
            from numpy
            from numpy
            import mat
            from sklea                  plit
            from sklea                  n
            from sklea
            from sklea                  ore
            from sklea
            dataframe = pnd.read_csv("C:/datasets/diamonds.csv", index_col=0)
```

Fig. 1.41 Load data in Jupyter Notebook and execute code



```
In [3]: import pandas as pd

In [5]: import matplotlib.pyplot as plt

In [9]: df=pd.read_excel("diabetes.xlsx")

In [10]: dfplt=df.plot(kind="hist")
```

Fig. 1.42 Visualizing blood pressure measurements in a graph

1.10.3.4 Save the Execution

All that we have done can be saved in a file with the extension “ipynb.” This file can be loaded later in Jupyter Notebook IDE for any updates or changes to continue working from where you left. Click on “File,” then choose “Save” or click on the save icon.

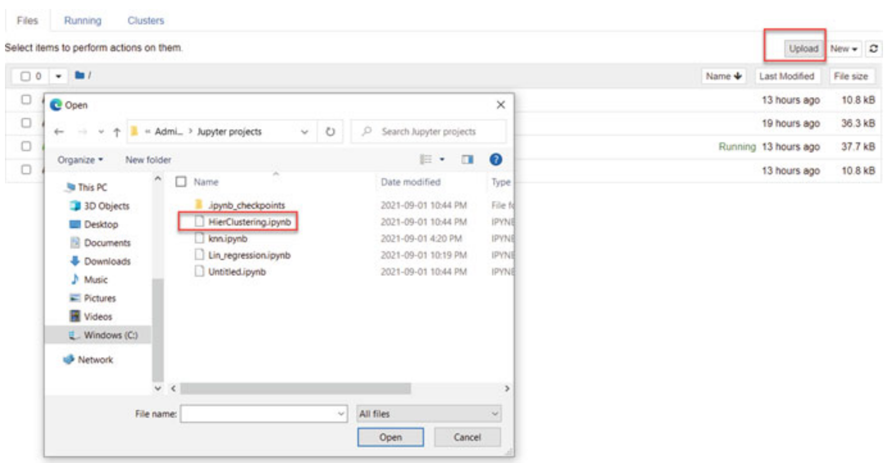


Fig. 1.43 Upload Jupyter Notebook files into the application

1.10.3.5 Load a Saved Execution

Go to your workspace folder in the command line and lunch Jupyter Notebook. Then, double-click on the file that you need to continue working on.

1.10.3.6 Upload a Jupyter Notebook File

You can also upload files into Jupyter Notebook through the application interface. After launching the program, click on the Upload button and upload all the files that you want to upload, as shown in Fig. 1.43.

1.10.4 Do It Yourself

The following problem is to try by yourself:

1. Weka is a machine learning software. Install Weka from the following link: https://waikato.github.io/weka-wiki/downloading_weka/
2. Jupyter lab is the next generation Jupyter Notebook interface. Install Jupyter lab from https://jupyter.org/. We suggest that you use Jupyter lab instead of Jupyter Notebook.
3. Download any dataset from Find Open Datasets and Machine Learning Projects | Kaggle and plot the data visually in R.
4. Download any dataset from Find Open Datasets and Machine Learning Projects | Kaggle and plot the data visually in Jupyter Notebook or Jupyter lab.
5. Try Colabroatory, also known as Colab, the online Python development environment provided by Google: https://colab.research.google.com/. We strongly advise you to use either Colab or JupyterLab for your projects.

References

1. E. Alpaydin, *Machine Learning: The New AI* (MIT Press, 2016)
2. H. Hassani, P. Amiri Andi, A. Ghodsi, K. Norouzi, N. Komendantova, S. Unger, Shaping the future of smart dentistry: From Artificial Intelligence (AI) to Intelligence Augmentation (IA). IoT **2**(3), 510–523 (2021)
3. R.E. Neapolitan, X. Jiang, *Artificial Intelligence: With an Introduction to Machine Learning* (CRC Press, 2018)
4. A.L. Fradkov, Early history of machine learning. IFAC-PapersOnLine **53**(2), 1385–1390 (2020)
5. J.A. Nichols, H.W.H. Chan, M.A. Baker, Machine learning: Applications of artificial intelligence to imaging and diagnosis. Biophys. Rev. **11**(1), 111–118 (2019)
6. Q. Bi, K.E. Goodman, J. Kaminsky, J. Lessler, What is machine learning? A primer for the epidemiologist. Am. J. Epidemiol. **188**(12), 2222–2239 (2019)
7. X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning. Synth. Lect. Artif. Intell. Mach. Learn. **3**(1), 1–130 (2009)
8. I.H. Sarker, Machine learning: Algorithms, real-world applications and research directions. SN Comput. Sci. **2**(3), 1–21 (2021)
9. N. Kalé, N. Jones, *Practical Analytics* (Epistemy Press, 2015)
10. R. Sharda, D. Delen, E. Turban, *Business Intelligence: A Managerial Perspective on Analytics: A Managerial Perspective on Analytics* (Pearson, 2015), pp. 416–416
11. K.C. Laudon, J.P. Laudon, *Management Information Systems: Managing the Digital Firm* (Pearson, 2017)
12. K.E.S.C.S. Pearlson, D.F. Galletta, *Managing and Using Information Systems a Strategic Approach* (John Wiley & Sons, 2016)
13. C. El Morr, H. Ali-Hassan, *Analytics in Healthcare: A Practical Introduction* (Springer, 2019)
14. P. Drucker, The coming of the new organization. Harv. Bus. Rev. **Jan–Feb**, 45–53 (1988)
15. T. Davenport, *Information Ecology* (Oxford University Press, New York, 1997)
16. SAS, Big Data - What it is and why it matters. https://www.sas.com/en_ca/insights/big-data/what-is-big-data.html. Accessed
17. D. Faggella, Where Healthcare's big data actually comes from. 11 Jan 2018. [Online]. Available: https://www.techemergence.com/where-healthcares-big-data-actually-comes-from/
18. D. Pogue, Exclusive: Fitbit's 150 billion hours of heart data reveal secrets about health. August 27, 2018. [Online]. Available: https://finance.yahoo.com/news/exclusive-fitbits-150-billion-hours-heart-data-reveals-secrets-human-health-133124215.html?linkId=56096180
19. J. Bresnick, Understanding the Many V's of Healthcare Big data analytics. 5 June 2017. [Online]. Available: https://healthitanalytics.com/news/understanding-the-many-vs-of-healthcare-big-data-analytics
20. R. Sharda, D. Delen, E. Turban, *Business Intelligence: A Managerial Perspective on Analytics* (Prentice Hall Press, 2015)
21. T. Economist, *Data Is Giving Rise to a New Economy* (The Economist, 6 May 2017)
22. R. Sharda, D. Delen, E. Turban, J. Aronson, T.P. Liang, *Businesss Intelligence and Analytics: Systems for Decision Support* (Prentice Hall Press, 2014)
23. K.C. Laudon, J.P. Laudon, *Essentials of Management Information Systems* (Pearson Upper Saddle River, 2011)
24. C. Ballard, D.M. Farrell, A. Gupta, C. Mazuela, S. Vohnik, *Dimensional Modeling: In a Business Intelligence Environment* (IBM Redbooks, 2012)
25. K.E. Pearlson, C.S. Saunders, D.F. Galletta, *Managing and Using Information Systems a Strategic Approach* (John Wiley & Sons, 2016)
26. R. Sharda, D. Delen, E. Turban, *Business Intelligence: A Managerial Perspective on Analytics* (Prentice Hall Press, 2013)
27. H. Mailvaganam, Introduction to OLAP. http://www.dwreview.com/OLAP/Introduction_OLAP.html. Accessed

28. R. Sharda, D. Delen, E. Turban, *Business Intelligence: A managerial Perspective on Analytics* (Prentice Hall Press, 2014)
29. L. Madsen, Business intelligence an introduction, in *Healthcare Business Intelligence: A Guide to Empowering Successful Data Reporting and Analytics*, (Wiley, 2012)
30. K.D. Lawrence, R. Klimberg, *Contemporary Perspectives in Data Mining*, vol 1 (Information Age Publishing, 2013)
31. M. K. Pratt, Business intelligence vs. business analytics: Where BI fits into your data strategy. *CIO Magazine,* 2017. Available: https://www.cio.com/article/2448992/business-intelligence/business-intelligence-vs-business-analytics-where-bi-fits-into-your-data-strategy.html
32. Rose Business Technologies, *Descriptive Diagnostic Predictive Prescriptive Analytics*. Rose Business Technologies. http://www.rosebt.com/blog/descriptive-diagnostic-predictive-prescriptive-analytics. Accessed 26 April 2018
33. J. Bresnick, Healthcare big data analytics: from description to prescription. https://healthitanalytics.com/news/healthcare-big-data-analytics-from-description-to-prescription. Accessed
34. S. Maloney, Making Sense of Analytics. Presented at the eHealth2018, Toronto ON. [Online]. Available: http://www.healthcareimc.com/main/making-sense-of-analytics/
35. R.S. Uberoi, U. Gupta, A. Sibal, Root cause analysis in healthcare. Apollo Med. **1**(1), 60–63 (9 Jan 2004). https://doi.org/10.1016/S0976-0016(12)60044-1
36. W.E. Fassett, Key performance outcomes of patient safety curricula: root cause analysis, failure mode and effects analysis, and structured communications skills. Am. J. Pharm. Educ. **75**(8), 164 (10 Oct 2011). https://doi.org/10.5688/ajpe758164
37. R. Ursprung, J. Gray, Random safety auditing, root cause analysis, failure mode and effects analysis. Clin. Perinatol. **37**(1), 141–165 (Mar 2010). https://doi.org/10.1016/j.clp.2010.01.008
38. M. Liao, Y. Li, F. Kianifard, E. Obi, S. Arcona, Cluster analysis and its application to healthcare claims data: A study of end-stage renal disease patients who initiated hemodialysis. BMC Nephrol. **17**, 25 (2016). https://doi.org/10.1186/s12882-016-0238-2
39. M. Chowdhury, A. Apon, K. Dey, *Data Analytics for Intelligent Transportation Systems* (Elsevier Science, 2017)
40. H.H. Hijazi, H.L. Harvey, M.S. Alyahya, H.A. Alshraideh, R.M. Al Abdi, S.K. Parahoo, The impact of applying quality management practices on patient centeredness in jordanian public hospitals: results of predictive modeling. Inquiry: J Medical Care Organization, Provision and Financing **55**, 46958018754739 (Jan–Dec 2018). https://doi.org/10.1177/0046958018754739
41. F. Noviyanti, Y. Hosotani, S. Koseki, Y. Inatsu, S. Kawasaki, Predictive modeling for the growth of salmonella enteritidis in chicken juice by real-time polymerase chain reaction. Foodborne Pathog. Dis. **15**(7), 406–412 (2 Apr 2018). https://doi.org/10.1089/fpd.2017.2392
42. M.M. Safaee et al., Predictive modeling of length of hospital stay following adult spinal deformity correction: Analysis of 653 patients with an accuracy of 75% within 2 days. World Neurosurg **115**, e422–e427 (17 Apr 2018). https://doi.org/10.1016/j.wneu.2018.04.064
43. B. Baessler, M. Mannil, D. Maintz, H. Alkadhi, R. Manka, Texture analysis and machine learning of non-contrast T1-weighted MR images in patients with hypertrophic cardiomyopathy-preliminary results. Eur. J. Radiol. **102**, 61–67 (May 2018). https://doi.org/10.1016/j.ejrad.2018.03.013
44. P. Karisani, Z.S. Qin, E. Agichtein, Probabilistic and machine learning-based retrieval approaches for biomedical dataset retrieval. Database: J. Biol. Databases Curation **2018**, bax104 (1 Jan 2018). https://doi.org/10.1093/database/bax104
45. M.R. Schadler, A. Warzybok, B. Kollmeier, Objective prediction of hearing aid benefit across listener groups using machine learning: Speech recognition performance with binaural noise-reduction algorithms. Trends Hear. **22**, 2331216518768954 (Jan–Dec 2018). https://doi.org/10.1177/2331216518768954
46. Y. Wu, K. Doi, C.E. Metz, N. Asada, M.L. Giger, Simulation studies of data classification by artificial neural networks: Potential applications in medical imaging and decision making. J. Digit. Imaging **6**(2), 117–125 (May 1993)

47. J. Zhang, M. Liu, D. Shen, Detecting anatomical landmarks from limited medical imaging data using two-stage task-oriented deep neural networks. IEEE Trans. Image Process. **26**(10), 4753–4764 (Oct 2017). https://doi.org/10.1109/tip.2017.2721106
48. E. Chalmers, D. Hill, V. Zhao, E. Lou, Prescriptive analytics applied to brace treatment for AIS: A pilot demonstration. Scoliosis **10**(Suppl 2), S13 (2015). https://doi.org/10.1186/1748-7161-10-s2-s13
49. F. Devriendt, D. Moldovan, W. Verbeke, A literature survey and experimental evaluation of the state-of-the-art in uplift modeling: A stepping stone toward the development of prescriptive analytics. Big Data **6**(1), 13–41 (Mar 2018). https://doi.org/10.1089/big.2017.0104
50. S. Van Poucke, M. Thomeer, J. Heath, M. Vukicevic, Are Randomized controlled trials the (G)old standard? From clinical intelligence to prescriptive analytics. *Journal of medical Internet Research* **18**(7), e185 (6 Jul 2016). https://doi.org/10.2196/jmir.5549
51. G.K. Alexander, S.B. Canclini, J. Fripp, W. Fripp, Waterborne disease case investigation: Public health nursing simulation. J. Nurs. Educ. **56**(1), 39–42 (1 Jan 2017). https://doi.org/10.3928/01484834-20161219-08
52. M. Lee, Y. Chun, D.A. Griffith, Error propagation in spatial modeling of public health data: A simulation approach using pediatric blood lead level data for Syracuse, New York. Environ. Geochem. Health **40**(2), 667–681 (Apr 2018). https://doi.org/10.1007/s10653-017-0014-7
53. M. Moessner, S. Bauer, Maximizing the public health impact of eating disorder services: A simulation study. Int. J. Eat. Disord. **50**(12), 1378–1384 (Dec 2017). https://doi.org/10.1002/eat.22792
54. O. El-Rifai, T. Garaix, V. Augusto, X. Xie, A stochastic optimization model for shift scheduling in emergency departments. Health Care Manag. Sci. **18**(3), 289–302 (Sep 2015). https://doi.org/10.1007/s10729-014-9300-4
55. A. Jeremic, E. Khoshrowshahli, Detecting breast cancer using microwave imaging and stochastic optimization. Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference **2015**, 89–92 (2015). https://doi.org/10.1109/embc.2015.7318307
56. A. Legrain, M.A. Fortin, N. Lahrichi, L.M. Rousseau, Online stochastic optimization of radiotherapy patient scheduling. Health Care Manag. Sci. **18**(2), 110–123 (Jun 2015). https://doi.org/10.1007/s10729-014-9270-6
57. M.A. Christodoulou, C. Kontogeorgou, Collision avoidance in commercial aircraft free flight via neural networks and non-linear programming. Int. J. Neural Syst. **18**(5), 371–387 (Oct 2008). https://doi.org/10.1142/s0129065708001658
58. S.I. Saffer, C.E. Mize, U.N. Bhat, S.A. Szygenda, Use of non-linear programming and stochastic modeling in the medical evaluation of normal-abnormal liver function. I.E.E.E. Trans. Biomed. Eng. **23**(3), 200–207 (May 1976)
59. G.H. Simmons, J.M. Christenson, J.G. Kereiakes, G.K. Bahr, A non-linear programming method for optimizing parallel-hole collimator design. Phys. Med. Biol. **20**(3), 771–788 (Sep 1975)
60. I. Podolak, Making sense of analytics. Presented at the eHealth 2017, Toronto ON, 2017. [Online]. Available: http://www.healthcareimc.com/main/making-sense-of-analytics/

Chapter 2
Statistics

2.1 Overview of the Chapter

Statistical analysis is an analytical tool used to describe, summarize, and draw conclusions about data. The main two aims for statistics are descriptive statistics and inferential statistics. Descriptive statistics involve collecting, organizing, and summarizing a given dataset, whereas inferential statistics relate to concluding a population from the results obtained in a sample. In the following chapter, we will give an overview of some common descriptive and inferential statistical tests, their uses, and how to interpret their results.

2.2 Definition of General Terms

Population: the pool of individuals in which we are interested at a particular time.

Sample: a group of individuals selected from the population.

Simple random sample: a sample that is drawn from a population in such a way that each individual in the population has an equal chance of being selected for the sample.

Parameter: a descriptive measure computed from the data of a population.

Statistic: a descriptive measure computed from the data of a sample.

Variables: characteristics of the individuals under study (such as age, gender, blood pressure, etc.).

Dependent variable: a variable of interest ("outcome variable" is another term used for a dependent variable).

Independent variable: a variable that influences the dependent variable ("exposure variable" is another term used for an independent variable).

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_2

2.3 Types of Variables

Broadly, variables can be categorized into numerical (quantitative) and categorical (qualitative) types. Numerical are measurable data and can be divided into continuous and discrete (number of steps, number of apples, etc.). For continuous data, it has an infinite number of evenly spaced values, such as age, blood pressure, height, and weight. Categorical are data classes that cannot be measured and can be divided into nominal and ordinal [1]. Categories for variables can be grouped into intervals; for example, education level can be categorized into primary school, high school, undergraduate degree, and graduate degree; income might be grouped into categories, such as less than $50,000, [50,000–69,999], [70,000–79,999], and so on. Such data are often referred to as *categorical* data [1].

Nominal data, such as gender and race, do not have an established order or rank and have a finite number of values. Nominal variables can be dichotomous, where the variable has only two levels (e.g., death), or polytomous, where the variable can have more than two levels (such as blood group).

Ordinal data have a limited number of options with an implied order, such as education. Descriptive Statistics

2.3.1 Measures of Central Tendency

Frequencies and percentages are usually used to summarize and describe information for categorical data. On the other hand, measures of central tendency and measures of dispersion, also referred to as measures of variability, are used to summarize information on continuous variables [2]. The most common measures of central tendency for continuous variables are the mean, median, and mode. The mean, also known as the average, is the sum of all values divided by the number of observations [3]. The mean is unique in value and simple to calculate; however, means are affected by extreme values [2]. The median is the value that divides the data such that half of the data points or observations are lower than it and half are higher [3]. A median of 35 years of age for a sample of participants means that half the participants are younger than 35, whereas the other half are older than 35. The median is unique in value and simple to calculate and is not affected by extreme values. The mode is the most common value in a dataset [3]. If the mode for an exam is 76/100, it means that the most common grade is 76. The mode is simple to calculate but not unique in value and is not affected by extreme values. The mean is the most widely used measure of central tendency, and many statistical tests are based on it [2]. The median is best used when the database is fairly small, the distribution is badly skewed, or we have missing values [4].

2.3.1.1 Measures of Dispersion

Measures of dispersion capture the spread and variability of the data points. The most common measures of dispersion include the variance, standard deviation, and range. The variance and standard deviation measure the dispersion of the individual values relative to the mean. Variance is the average of the squared deviations of the data points from the mean. The standard deviation is the square root of the variance. A smaller variance and standard deviation mean that data values are clustered close to the mean, while a higher value for both measures means that the data points are more spread [3]. The range is a simple measure of dispersion capturing the difference between the highest and lowest values for a certain variable [4].

2.4 Inferential Statistics

2.4.1 Data Distribution

Data distribution is a representation of the spread of continuous data across a range of values. Distribution charts can help better understand the data and the statistical analysis that needs to be performed. Such distributions are informative, as they provide the level of symmetry or skewness of the data, telling us whether there are roughly as many data points above the mean as there are below it (in the case of symmetry) or whether more observations are above the mean (positive skewness) or below it (negative skewness) [3].

A special case of data distribution is called the normal distribution, also known as the Gaussian distribution. The normal distribution is symmetrical and bell-shaped, with the **mean, median, and mode** all being identical. Approximately 68% of all the data values lie within plus or minus one standard deviation from the mean, 95% lie within plus or minus two standard deviations from the mean, and nearly all data values lie within plus or minus three standard deviations from the mean [3]. The normal distribution is a central component of inferential statistics [4].

2.4.2 Hypothesis Testing

Hypothesis testing is a method where researchers conclude a population based on the results of a sample. A hypothesis is a statement that something is correct. For hypothesis testing, tests of significance are conducted to conclude a population parameter. For hypothesis testing, the researcher first states the null hypothesis, usually of "no difference" (e.g., no difference between the population means of two groups), and an alternative hypothesis that disagrees with the null hypothesis and is a statement of "difference" (difference between the two population means).

The null hypothesis is usually denoted by H_0, whereas the alternative hypothesis is denoted by H_a. A test statistic is a statistic used for deciding whether the null hypothesis should be rejected or not. Examples of test statistics are chi-square tests, Student's t-tests, analysis of variance tests, simple linear regression, and correlation tests. A *p*-value, the probability of obtaining results at least as extreme as the observed results in the sample if the null hypothesis is true, is generated by default with different statistical tests. If the *p*-value is low, then this is taken as evidence that it is unlikely (although still possible) that the data are consistent with the null hypothesis, and hence we reject the null hypothesis. However, if the *p*-value is high, it indicates that most probably, the sample data are consistent with the null hypothesis, and thus we do not reject the H_0. The lower the *p*-value, the greater the significance of the findings [3].

2.4.3 Type I and II Errors

When we statistically test a hypothesis, we can accept a certain level of significance, known as α (alpha). When we say that a finding α is "statistically significant," it means that the finding is unlikely to have occurred by chance and that the level of significance is the maximum chance that we are willing to accept. A very common threshold for the level of significance, or α, is 0.05 [3].

Two types of errors may result from hypothesis testing: Type I and Type II. A Type I error occurs when we reject the null hypothesis (e.g., we conclude that there is a significant association between two variables or that there was a significant difference between the measurements of two or more different groups of patients) when in fact the null hypothesis is true (there is no significant association or difference between the variables). A Type II error occurs when we do not reject the null hypothesis when in fact it is false. If a Type I error is costly, meaning your belief that your theory is correct when it is not could be problematic, then you should choose a low value for α to avoid that error [3].

2.4.4 Steps for Performing Hypothesis Testing

The following steps summarize the steps for performing hypothesis testing for the association between two variables:

1. State the null and alternative hypotheses.
2. Perform the statistical test (chi-square test, t-test, one-way ANOVA test, correlation, or simple linear regression test).
3. Determine the level of significance of your test (e.g., $\alpha = 0.05$).

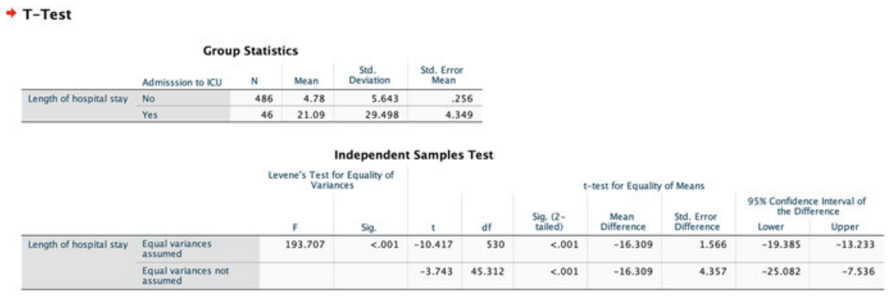
4. Compare the *p*-value associated with the test statistic to the α level; if the *p*-value is less than the α value, then reject the null hypothesis; otherwise, fail to reject the null hypothesis [4].

2.4.5 Test Statistics

To assess the bivariate relationship between two variables (one being the independent variable and the other being the dependent variable), a test statistic is performed, and a *p*-value will be generated for the test statistic. The choice of what test statistic to use depends on the type of variables. If the research question requires a comparison of two means, then a Student's t-test statistic is used, whereas a one-way analysis of variance is used to compare more than two means, and a chi-square test is used to compare two or more proportions. On the other hand, to assess the relationship between two continuous variables, correlation or simple linear regression analysis may be used [4]. In the section below, specific examples will be given to illustrate the use and interpretation of the test statistics.

2.4.5.1 Student's t-test

The student's t-test is used when one of the variables is continuous and the other is dichotomous (2-level), thus aiming to compare the two means [1]. As an example, assume an investigator is interested in comparing the mean length of hospital stay for patients admitted to the intensive care unit (ICU) compared to patients not admitted to the ICU. For this example, the null hypothesis is that there is no association between ICU admission and length of hospital stay (in other words, the mean of length of hospital stay for patients admitted to the ICU is equal to that for patients not admitted to the ICU). The alternative hypothesis is that there is no association (no difference in the means for the population). Figure 2.1 shows the output of the


T-Test

Group Statistics

	Admisssion to ICU	N	Mean	Std. Deviation	Std. Error Mean
Length of hospital stay	No	486	4.78	5.643	.256
	Yes	46	21.09	29.498	4.349

Independent Samples Test

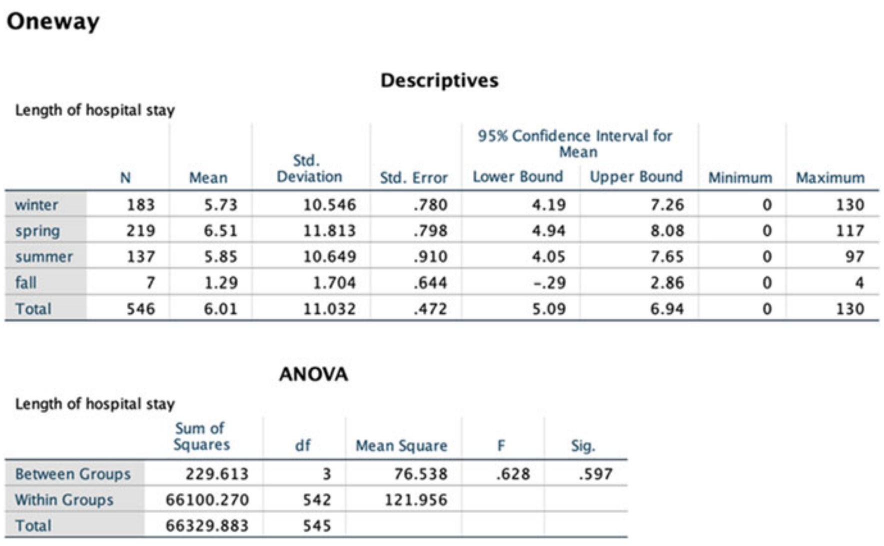
		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Length of hospital stay	Equal variances assumed	193.707	<.001	-10.417	530	<.001	-16.309	1.566	-19.385	-13.233
	Equal variances not assumed			-3.743	45.312	<.001	-16.309	4.357	-25.082	-7.536

Fig. 2.1 Example of Student's t-test (results based on SPSS output)

t-test analysis, thus showing that the mean of length of hospital stay for those admitted to the ICU is 21.09 days, compared to the mean of 4.78 days for patients who were not admitted to the ICU. The t-test shows a value of −3.743, and the *p*-value associated with the t-test is 0.001. Since the *p*-value is $0.001 < 0.05$ (the α level), we reject the null hypothesis and conclude that at an α level of 0.05, the results are significant (in other words, in the population, the mean length of hospital stay is different for patients admitted to the ICU vs. those who are not). Note that this is an example of an independent sample t-test because the two samples are not related. However, there are instances when the investigator may want to compare means for the same sample at different points in time (e.g., before vs. after a treatment). For such analysis, a dependent Student's t-test (or paired sample t-test) is used, since the individuals in the sample are the same and the measurements are taken at two different points in time [2].

2.4.5.2 One-Way Analysis of Variance

Analysis of variance, or ANOVA, is used when one wants to compare more than two means [2]. Figure 2.2 shows the SPSS output of the results for the analysis assessing whether the length of hospital stay is associated with the season of hospital admission (winter, spring, summer, and fall). One-way analysis of variance should be used here, since the dependent variable is continuous (length of hospital stay) and the independent variable (season of hospital of admission) is a nominal variable with four categories. The null hypothesis would be that the length of hospital stay is the same for those admitted in the winter, spring, summer, and fall, whereas the



Oneway

Descriptives

Length of hospital stay

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
winter	183	5.73	10.546	.780	4.19	7.26	0	130
spring	219	6.51	11.813	.798	4.94	8.08	0	117
summer	137	5.85	10.649	.910	4.05	7.65	0	97
fall	7	1.29	1.704	.644	-.29	2.86	0	4
Total	546	6.01	11.032	.472	5.09	6.94	0	130

ANOVA

Length of hospital stay

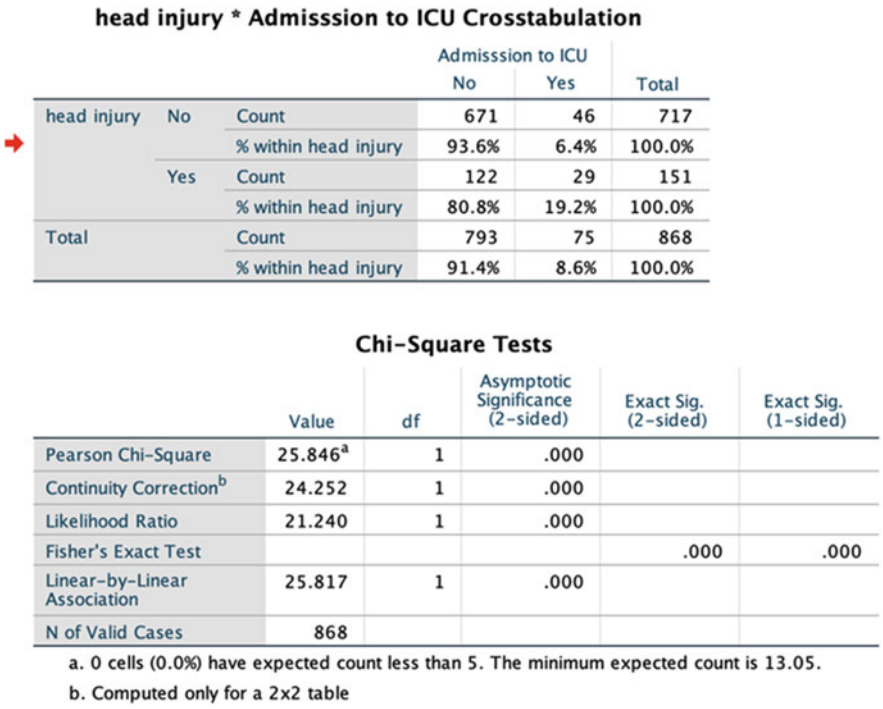
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	229.613	3	76.538	.628	.597
Within Groups	66100.270	542	121.956		
Total	66329.883	545			

Fig. 2.2 Example of analysis of variance test (results based on SPSS output)

alternative hypothesis would be that the length of hospital stay is different for hospital admission during at least two seasons. The F test calculated for this analysis as shown in the output below is 0.628, and the *p*-value is 0.597. Since the *p*-value of 0.597 is >0.05 (alpha level), the decision would be not to reject the null hypothesis. The conclusion for this analysis would be that there is no evidence to show that the length of hospital stay is associated with the season of admission.

2.4.5.3 Chi-Square Statistic

The chi-square test is used when both independent and dependent variables have categorical levels [4]. For example, an investigator is interested in assessing whether the proportion of patients admitted to the intensive care unit (ICU) is different if the patient had a head injury compared to not having a head injury. The null hypothesis would be that the proportion of patients with a head injury who are admitted to the ICU is equal to that for patients with no head injury, and the alternative hypothesis would be that the two proportions are different. Figure 2.3 shows the SPSS results of the test statistics for this relationship, where the proportion of patients admitted to the



head injury * Admisssion to ICU Crosstabulation

			Admisssion to ICU		
			No	Yes	Total
head injury	No	Count	671	46	717
		% within head injury	93.6%	6.4%	100.0%
	Yes	Count	122	29	151
		% within head injury	80.8%	19.2%	100.0%
Total		Count	793	75	868
		% within head injury	91.4%	8.6%	100.0%

Chi-Square Tests

	Value	df	Asymptotic Significance (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	25.846[a]	1	.000		
Continuity Correction[b]	24.252	1	.000		
Likelihood Ratio	21.240	1	.000		
Fisher's Exact Test				.000	.000
Linear-by-Linear Association	25.817	1	.000		
N of Valid Cases	868				

a. 0 cells (0.0%) have expected count less than 5. The minimum expected count is 13.05.

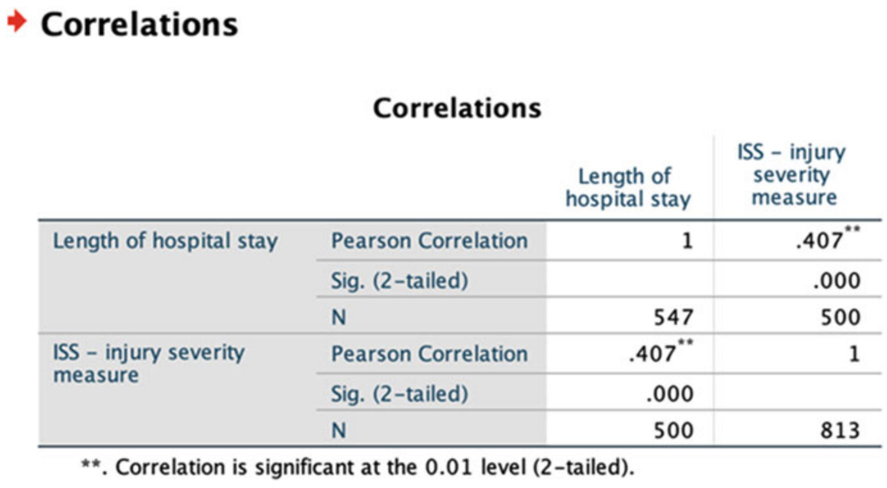
b. Computed only for a 2x2 table

Fig. 2.3 Example of chi-square test (results based on SPSS output)

ICU was 19.2% for patients with head injury compared to 6.4% for patients with no head injury. The chi-square test calculated was 25.846, with an associated *p*-value of <0.001. Since the *p*-value is <0.05, the decision would be to reject the null hypothesis and conclude that having a head injury is associated with ICU admission.

2.4.5.4 Correlation

Pearson correlation, measured by Pearson's correlation coefficient (*r*), is a test statistic used mainly when both independent and dependent variables are continuous [4]. For this analysis, we are testing whether there is a linear relationship between the two variables. The values for *r* range from −1 (perfect negative linear relationship between the two variables) to +1 (perfect positive linear relationship between the two variables). An *r* that is equal to 0 indicates no linear relationship between the two variables [2, 3]. Figure 2.4 shows the results of a Pearson correlation coefficient to answer the question of whether there is a linear association between injury severity score (ISS) (a measure of injury severity that ranges between 1 and 75) and length of hospital stay. The null hypothesis is that there is no linear relationship between the two variables (correlation at the population level = 0), whereas the alternative hypothesis is that there is a linear relationship (correlation at the population level is different from 0). Since the two variables are continuous, a correlation test could be performed. Based on the results shown below, *r* is 0.407 and the *p*-value is <0.001. Because the *p*-value is <0.05, the null hypothesis would be rejected, and the conclusion would be that, at the population level, we have enough evidence that there is a linear relationship between ISS and hospital stay.



Correlations

Correlations

		Length of hospital stay	ISS – injury severity measure
Length of hospital stay	Pearson Correlation	1	.407**
	Sig. (2-tailed)		.000
	N	547	500
ISS – injury severity measure	Pearson Correlation	.407**	1
	Sig. (2-tailed)	.000	
	N	500	813

**. Correlation is significant at the 0.01 level (2-tailed).

Fig. 2.4 Example of correlation test (results based on SPSS output)

2.4.5.5 Simple Linear Regression

Simple linear regression is performed when the investigators want to assess if the relationship between two variables is linear [4]. For this situation, the mathematical model to use is that of a straight line:

$$y = B_0 + B_1 x$$

y = dependent variable
x = independent variable
B_0 = y-intercept
B_1 = slope

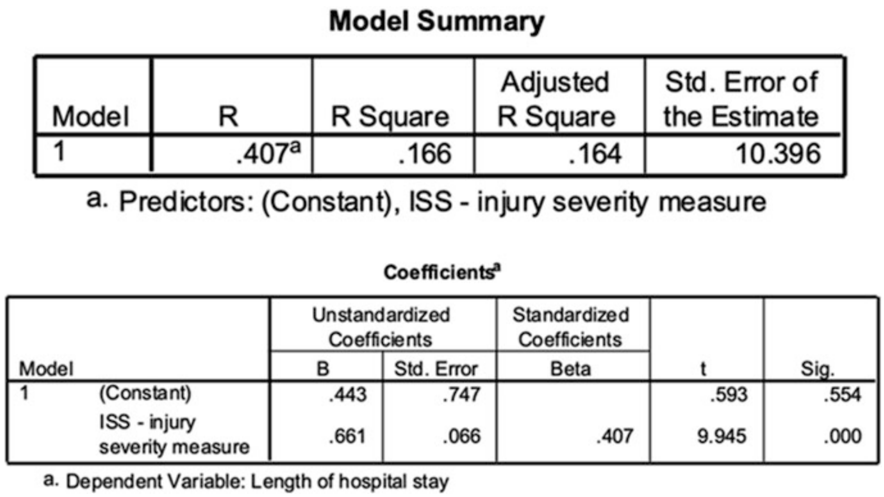
where the intercept B_0 is the value of y when $x = 0$. The slope B_1 is the amount of change in y for each 1-unit change in x.

To explain why patients have different blood pressure, we try to associate the differences in blood pressure with differences in other relevant patient characteristics (variables). For example, we may try to answer the question of whether variation in blood pressure could be explained by age. The formula of the straight line becomes:

$$\text{Blood pressure} = B_0 + B_1 \text{age}$$

When simple linear regression analysis is performed, one is trying to answer the following questions: Does age help to predict blood pressure using a straight-line model? Is there a linear relationship between age and blood pressure? Does age explain a significant portion of the variation in the values of blood pressure?

Figure 2.5 shows the results of a simple linear regression analysis that assesses the research question, where ISS is linearly associated with length of hospital stay. The



Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.407[a]	.166	.164	10.396

a. Predictors: (Constant), ISS - injury severity measure

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.443	.747		.593	.554
	ISS - injury severity measure	.661	.066	.407	9.945	.000

a. Dependent Variable: Length of hospital stay

Fig. 2.5 Example of simple linear regression test (results based on SPSS output)

dependent variable is length of hospital stay, and the independent variable is ISS. The null hypothesis is that the slope of the straight line in the population is equal to 0 (in other words, there is no linear relationship), and the alternative hypothesis is that the slope is different from 0. Figure 2.5 shows that the slope of the straight line based on the sample is 0.661, indicating that increasing ISS by 1 unit increases the length of hospital stay by 0.661 days. The p-value of the slope is <0.001; thus, we reject the null hypothesis and conclude that ISS provides significant information for predicting the length of hospital stay. R^2, one of the statistics computed when simple linear regression is performed, indicates the level of variation in the dependent variable that is explained by the independent variable. Based on Fig. 2.5, the R^2 value shows that ISS explains 16.6% of the variation observed in the length of hospital stay.

2.5 Conclusion

This chapter has provided an introduction to basic statistics on the key pillars in data analysis. Because knowledge of basic statistical concepts is necessary for understanding and appreciating the complexity of data analytics, key concepts, such as statistical tests and hypothesis verification, were covered.

2.6 Key Terms

1. Population
2. Sample
3. Simple random sample
4. Parameter
5. Statistic
6. Variables
7. Dependent variable
8. Independent variable
9. Descriptive statistics
10. Mean
11. Median
12. Mode
13. Inferential statistics
14. Hypothesis testing
15. Test statistics
16. Student's t-test
17. Chi-square
18. ANOVA

19. Correlation
20. Simple linear regression

2.7 Test Your Understanding

1. Define each of the key terms.
2. How would you interpret the relationship between two variables that are negatively correlated?
3. How would you interpret the relationship between two variables that are positively correlated?
4. How would you interpret a correlation where the correlation coefficient is 0?
5. How would you interpret a correlation where the correlation coefficient is 1?
6. Which statistics would you choose to test the association of two categorical variables?
7. Which statistics would you choose to test the association of two numeric variables?
8. Can a statistics result allow us to reject a hypothesis?
9. Can a statistics result allow us to confirm a hypothesis?

2.8 Read More

1. Faltin, F. W., Kenett, R. S., & Ruggeri, F. (2012). Statistical Methods in Healthcare. Wiley. https://books.google.ca/books?id=3cRPEGOn1hUC
2. Hahs-Vaughn, D., & Lomax, R. (2013). An introduction to statistical concepts. Routledge.
3. Lalanne, C., & Mesbah, M. (2017). Biostatistics and Computer-based Analysis of Health Data Using SAS. Elsevier Science. https://books.google.ca/books?id=HT3jCwAAQBAJ

2.9 Lab

2.9.1 Working Example in R

2.9.1.1 Statistical Measures Overview

The purpose of this lab is to apply different statistical measures in R and Python, such as measures of central tendency and dispersion. These concepts are important ones in applying machine learning algorithms. As described above, there are three central tendency measures: mean, median, and mode. Measures for central tendency are used to find the center or typical point in the data distribution. Meanwhile, dispersion measures, such as standard deviation, range, and interquartile range, are used to illustrate the variability in the data. Moreover, this lab will also use the *p*-value test statistics to approve or reject the null hypothesis (i.e., statistically significant or not).

2.9.1.2 Central Tendency Measures in R

Before calculating mean, median, and mode measures in R, it is necessary to note that the bank loan status dataset will be used across this lab. You can download the data file using the following link (https://www.kaggle.com/datasets/zaurbegiev/my-dataset).

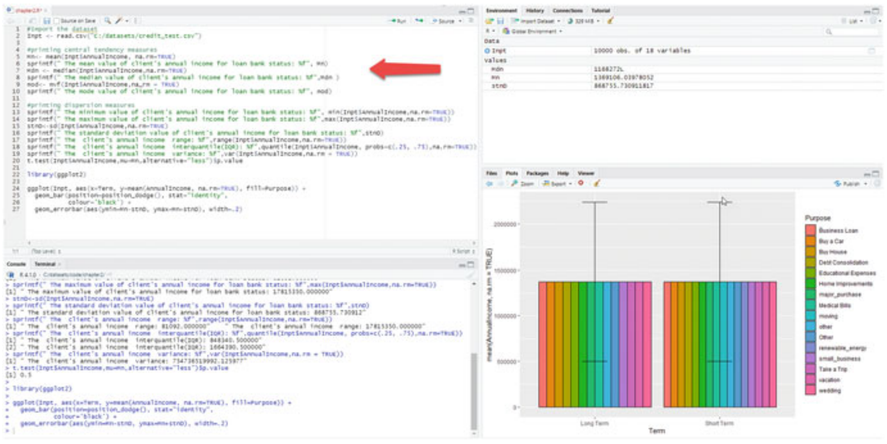
As shown in Fig. 2.6, the mean, median, and mode are calculated based on the Annual Income column. The plot in this figure shows the purpose of the loan vs. the mean of annual income based on short-long and long-term loans.

2.9.1.3 Dispersion in R

Standard deviation, interquartile range, and variance are used to calculate dispersion measures of the borrowers' annual income as shown in Fig. 2.7.

2.9.1.4 Statistical Test Using *p*-value in R

In order to approve or reject the null hypothesis, *p*-value is calculated using the t-statistic test as shown in Fig. 2.8.

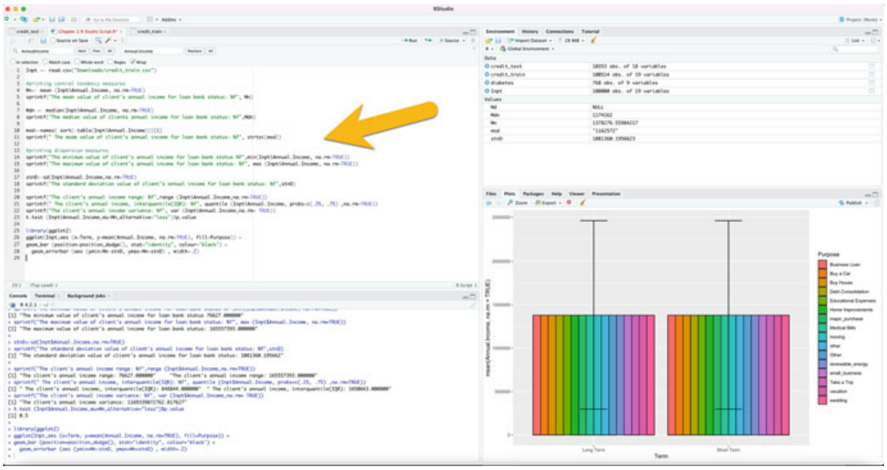


```
#Import the dataset
Inpt <- read.csv("C:/datasets/credit_test.csv")

#printing central tendency measures
mn<- mean(Inpt$AnnualIncome, na.rm=TRUE)
sprintf(" The mean value of client's annual income for loan bank status: %f", mn)
mdn <- median(Inpt$AnnualIncome,na.rm=TRUE)
sprintf(" The median value of client's annual income for loan bank status: %f",mdn )
mod<- mvf(Inpt$AnnualIncome,na_rm = TRUE)
sprintf(" The mode value of client's annual income for loan bank status: %f", mod)

#printing dispersion measures
sprintf(" The minimum value of client's annual income for loan bank status: %f", min(Inpt$AnnualIncome,na.rm=TRUE))
sprintf(" The maximum value of client's annual income for loan bank status: %f",max(Inpt$AnnualIncome,na.rm=TRUE))
stnD<-sd(Inpt$AnnualIncome,na.rm=TRUE)
sprintf(" The standard deviation value of client's annual income for loan bank status: %f",stnD)
sprintf(" The  client's annual income  range: %f",range(Inpt$AnnualIncome,na.rm=TRUE))
sprintf(" The  client's annual income  interquantile(IQR): %f",quantile(Inpt$AnnualIncome, probs=c(.25, .75),na.rm=TRUE))
sprintf(" The  client's annual income  variance: %f",var(Inpt$AnnualIncome,na.rm = TRUE))
t.test(Inpt$AnnualIncome,mu=mn,alternative="less")$p.value

library(ggplot2)

ggplot(Inpt, aes(x=Term, y=mean(AnnualIncome, na.rm=TRUE), fill=Purpose)) +
  geom_bar(position=position_dodge(), stat="identity",
           colour='black') +
  geom_errorbar(aes(ymin=mn-stnD, ymax=mn+stnD), width=.2)
```

```
> sprintf(" The maximum value of client's annual income for loan bank status: %f",max(Inpt$AnnualIncome,na.rm=TRUE))
[1] " The maximum value of client's annual income for loan bank status: 17815350.000000"
> stnD<-sd(Inpt$AnnualIncome,na.rm=TRUE)
> sprintf(" The standard deviation value of client's annual income for loan bank status: %f",stnD)
[1] " The standard deviation value of client's annual income for loan bank status: 868755.730912"
> sprintf(" The  client's annual income  range: %f",range(Inpt$AnnualIncome,na.rm=TRUE))
[1] " The  client's annual income  range: 81092.000000"    " The  client's annual income  range: 17815350.000000"
> sprintf(" The  client's annual income  interquantile(IQR): %f",quantile(Inpt$AnnualIncome, probs=c(.25, .75),na.rm=TRUE))
[1] " The  client's annual income  interquantile(IQR): 848340.500000"
[2] " The  client's annual income  interquantile(IQR): 1664390.500000"
> sprintf(" The  client's annual income  variance: %f",var(Inpt$AnnualIncome,na.rm = TRUE))
[1] " The  client's annual income  variance: 754736519992.125977"
> t.test(Inpt$AnnualIncome,mu=mn,alternative="less")$p.value
[1] 0.5
>
> library(ggplot2)
>
> ggplot(Inpt, aes(x=Term, y=mean(AnnualIncome, na.rm=TRUE), fill=Purpose)) +
+   geom_bar(position=position_dodge(), stat="identity",
+            colour='black') +
+   geom_errorbar(aes(ymin=mn-stnD, ymax=mn+stnD), width=.2)
> |
```

```
Data
Inpt      10000 obs. of 18 variables
Values
  mdn     1168272L
  mn      1369106.03978052
  stnD    868755.730911817
```

Fig. 2.6 Calculating central tendency measure in R

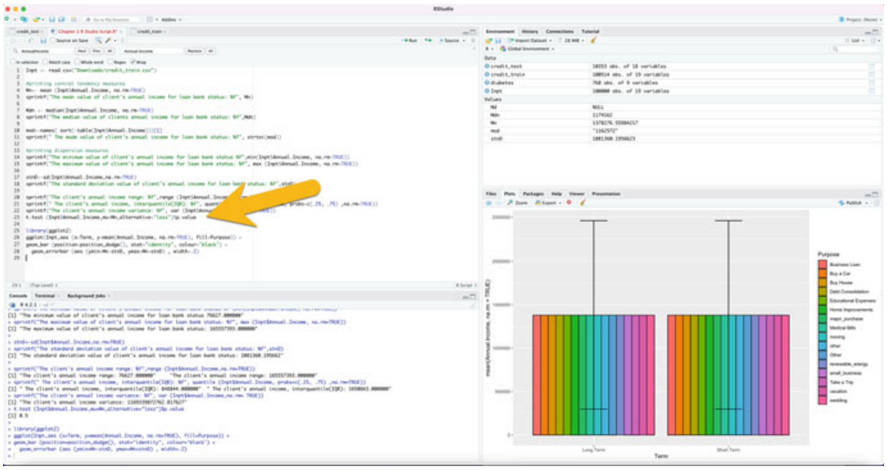


```
Inpt <- read.csv("Downloads/credit_train.csv")

#printing central tendency measures
Mn<- mean (Inpt$Annual.Income, na.rm=TRUE)
sprintf("The mean value of client's annual income for loan bank status: %f", Mn)

Mdn <- median(Inpt$Annual.Income, na.rm=TRUE)
sprintf("The median value of clients annual income for loan bank status: %f",Mdn)

mod<-names( sort(-table(Inpt$Annual.Income)))[1]
sprintf(" The mode value of client's annual income for loan bank status: %f", strtoi(mod))

#printing dispersion measures
sprintf("The minimum value of client's annual income for loan bank status %f",min(Inpt$Annual.Income, na.rm=TRUE))
sprintf("The maximum value of client's annual income for loan bank status: %f", max (Inpt$Annual.Income, na.rm=TRUE))

stnD<-sd(Inpt$Annual.Income,na.rm=TRUE)
sprintf("The standard deviation value of client's annual income for loan bank status: %f",stnD)

sprintf("The client's annual income range: %f",range (Inpt$Annual.Income,na.rm=TRUE))
sprintf(" The client's annual income, interquantile(IQR): %f", quantile (Inpt$Annual.Income, probs=c(.25, .75) ,na.rm=TRUE))
sprintf("The client's annual income variance: %f", var (Inpt$Annual.Income,na.rm= TRUE))
t.test (Inpt$Annual.Income,mu=Mn,alternative="less")$p.value

library(ggplot2)
ggplot(Inpt,aes (x=Term, y=mean(Annual.Income, na.rm=TRUE), fill=Purpose)) +
geom_bar (position=position_dodge(), stat="identity", colour="black") +
  geom_errorbar (aes (ymin=Mn-stnD, ymax=Mn+stnD) , width=.2)
```

```
[1] "The minimum value of client's annual income for loan bank status 76627.000000"
> sprintf("The maximum value of client's annual income for loan bank status: %f", max (Inpt$Annual.Income, na.rm=TRUE))
[1] "The maximum value of client's annual income for loan bank status: 165557393.000000"
>
> stnD<-sd(Inpt$Annual.Income,na.rm=TRUE)
> sprintf("The standard deviation value of client's annual income for loan bank status: %f",stnD)
[1] "The standard deviation value of client's annual income for loan bank status: 1081360.195662"
>
> sprintf("The client's annual income range: %f",range (Inpt$Annual.Income,na.rm=TRUE))
[1] "The client's annual income range: 76627.000000"     "The client's annual income range: 165557393.000000"
> sprintf(" The client's annual income, interquantile(IQR): %f", quantile (Inpt$Annual.Income, probs=c(.25, .75) ,na.rm=TRUE))
[1] " The client's annual income, interquantile(IQR): 848844.000000"  " The client's annual income, interquantile(IQR): 1650663.000000"
> sprintf("The client's annual income variance: %f", var (Inpt$Annual.Income,na.rm= TRUE))
[1] "The client's annual income variance: 1169339872762.81762"
> t.test (Inpt$Annual.Income,mu=Mn,alternative="less")$p.value
[1] 0.5
>
> library(ggplot2)
> ggplot(Inpt,aes (x=Term, y=mean(Annual.Income, na.rm=TRUE), fill=Purpose)) +
+ geom_bar (position=position_dodge(), stat="identity", colour="black") +
+   geom_errorbar (aes (ymin=Mn-stnD, ymax=Mn+stnD) , width=.2)
>
```

```
Data
credit_test     10353 obs. of 18 variables
credit_train    100514 obs. of 19 variables
diabetes        768 obs. of 9 variables
Inpt            100000 obs. of 19 variables
Values
Md              NULL
Mdn             1174162
Mn              1378276.55984217
mod             "1162572"
stnD            1081360.1956623
```

Fig. 2.7 Calculating dispersion measures for annual income in R



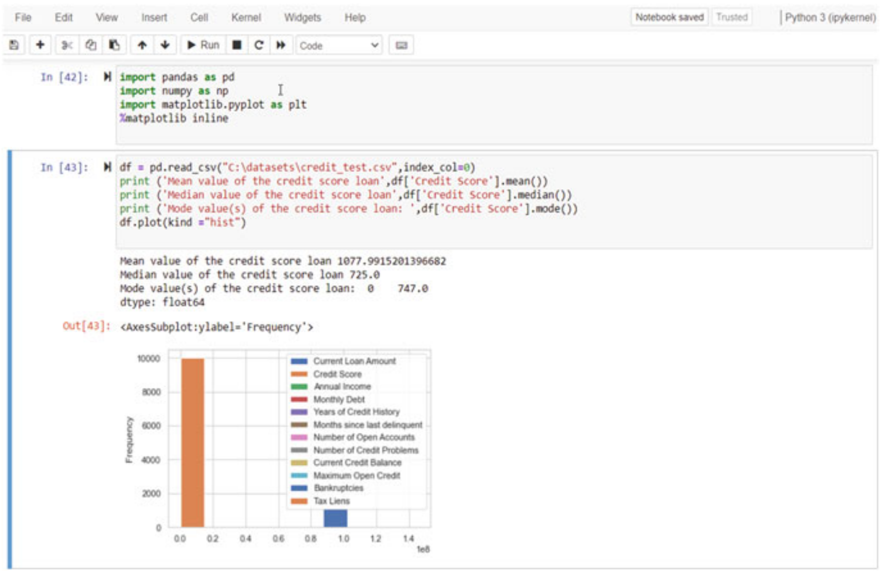
```
Inpt <- read.csv("Downloads/credit_train.csv")

#printing central tendency measures
Mn<- mean (Inpt$Annual.Income, na.rm=TRUE)
sprintf("The mean value of client's annual income for loan bank status: %f", Mn)

Mdn <- median(Inpt$Annual.Income, na.rm=TRUE)
sprintf("The median value of clients annual income for loan bank status: %f",Mdn)

mod<-names( sort(-table(Inpt$Annual.Income)))[1]
sprintf(" The mode value of client's annual income for loan bank status: %f", strtoi(mod))

#printing dispersion measures
sprintf("The minimum value of client's annual income for loan bank status %f",min(Inpt$Annual.Income, na.rm=TRUE))
sprintf("The maximum value of client's annual income for loan bank status: %f", max (Inpt$Annual.Income, na.rm=TRUE))

stnD<-sd(Inpt$Annual.Income,na.rm=TRUE)
sprintf("The standard deviation value of client's annual income for loan bank status: %f",stnD)

sprintf("The client's annual income range: %f",range (Inpt$Annual.Income,na.rm=TRUE))
sprintf(" The client's annual income, interquantile(IQR): %f", quantile (Inpt$Annual.Income, probs=c(.25, .75) ,na.rm=TRUE))
sprintf("The client's annual income variance: %f", var (Inpt$Annual.Income,na.rm= TRUE))
t.test (Inpt$Annual.Income,mu=Mn,alternative="less")$p.value

library(ggplot2)
ggplot(Inpt,aes (x=Term, y=mean(Annual.Income, na.rm=TRUE), fill=Purpose)) +
geom_bar (position=position_dodge(), stat="identity", colour="black") +
  geom_errorbar (aes (ymin=Mn-stnD, ymax=Mn+stnD) , width=.2)
```

Fig. 2.8 Applying statistical test by calculating *p*-value using t-test

2.9.2 *Working Example in Python*

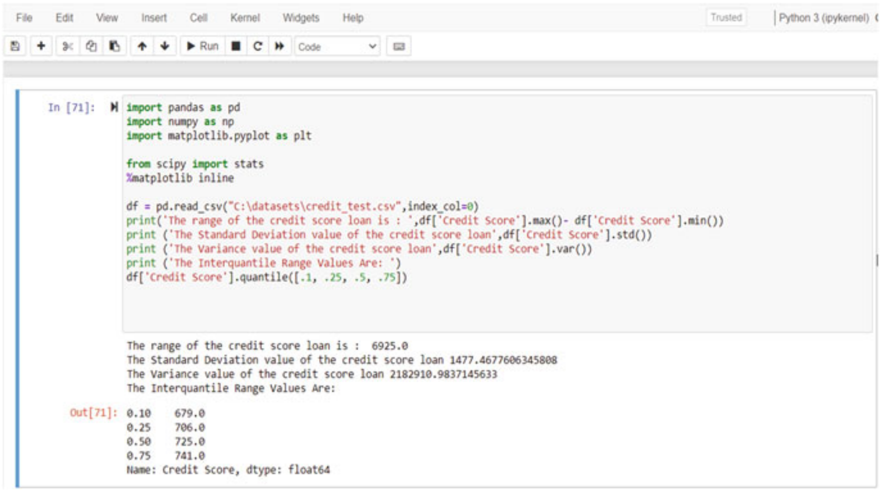
2.9.2.1 Central Tendency Measure in Python

In this section, the bank loan status dataset is used again to calculate the mean, median, and mode for the credit score and plot that in a graph as shown in Fig. 2.9 using Python.



```
In [42]:  import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          %matplotlib inline

In [43]:  df = pd.read_csv("C:\datasets\credit_test.csv",index_col=0)
          print ('Mean value of the credit score loan',df['Credit Score'].mean())
          print ('Median value of the credit score loan',df['Credit Score'].median())
          print ('Mode value(s) of the credit score loan: ',df['Credit Score'].mode())
          df.plot(kind ="hist")

          Mean value of the credit score loan 1077.9915201396682
          Median value of the credit score loan 725.0
          Mode value(s) of the credit score loan:  0    747.0
          dtype: float64

Out[43]:  <AxesSubplot:ylabel='Frequency'>
```

Fig. 2.9 Measures of central tendency in Python



```
In [71]:  import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt

          from scipy import stats
          %matplotlib inline

          df = pd.read_csv("C:\datasets\credit_test.csv",index_col=0)
          print('The range of the credit score loan is : ',df['Credit Score'].max()- df['Credit Score'].min())
          print ('The Standard Deviation value of the credit score loan',df['Credit Score'].std())
          print ('The Variance value of the credit score loan',df['Credit Score'].var())
          print ('The Interquantile Range Values Are: ')
          df['Credit Score'].quantile([.1, .25, .5, .75])

          The range of the credit score loan is :  6925.0
          The Standard Deviation value of the credit score loan 1477.4677606345808
          The Variance value of the credit score loan 2182910.9837145633
          The Interquantile Range Values Are:

Out[71]:  0.10    679.0
          0.25    706.0
          0.50    725.0
          0.75    741.0
          Name: Credit Score, dtype: float64
```

Fig. 2.10 Dispersion measures for the bank status loan dataset

2.9.2.2 Dispersion Measures in Python

The same dataset is used to calculate the standard deviation, variance, and interquartile range of dispersion measures as shown in Fig. 2.10.

2.9.2.3 Statistical Testing Using *p*-value in Python

Statistical testing (*p*-value) is used to find a relationship between two variables ("Current Credit Balance" and "Monthly Debt"). As you can see below in Fig. 2.11, the *p*-value is around 0, which means that the two groups of data do not have a statistically significant relationship.

2.9.3 Do It Yourself

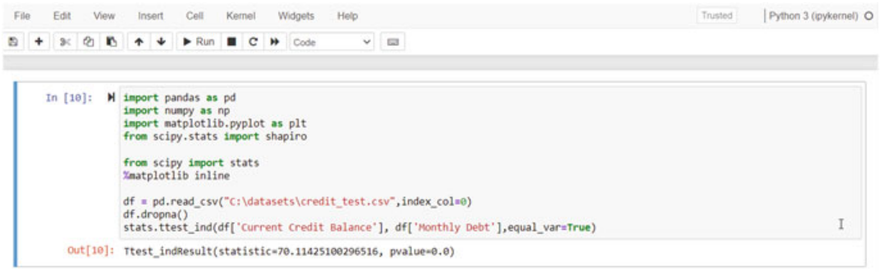
This section is for students to apply what they have learned in this chapter. Students need to solve the following problem in R and Python:

1. Load the medical cost personal dataset: Medical Cost Personal Datasets | Kaggle.
2. Calculate the mean, mode, and median central tendency measures and plot the mean against the Charges column based on the Gender column.
3. Calculate the standard deviation, variance, and interquartile range dispersion measures against the Charges column.
4. Set the null hypothesis, calculate the *p*-value, and deduce a conclusion from the result.

2.9.4 Do More Yourself (Links to Available Datasets for Use)

Below are a few datasets that you might use to do more exercises:

1. Students' Academic Performance Dataset | Kaggle
2. Iris Species | Kaggle
3. Climate Change: Earth Surface Temperature Data | Kaggle



```
In [10]:  import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          from scipy.stats import shapiro

          from scipy import stats
          %matplotlib inline

          df = pd.read_csv("C:\datasets\credit_test.csv",index_col=0)
          df.dropna()
          stats.ttest_ind(df['Current Credit Balance'], df['Monthly Debt'],equal_var=True)

Out[10]:  Ttest_indResult(statistic=70.11425100296516, pvalue=0.0)
```

Fig. 2.11 Statistical testing using *p*-value

References

1. L. K. Alexander, B. Lopes, K. Ricchetti-Masterson, and K. B. Yeatts, Common statistical tests and applications in epidemiological literature. Accessed
2. B.M. Thorne, J.M. Giesen, *Statistics for the Behavioral Sciences* (McGraw-Hill Humanities, Social Sciences & World Languages, 2003)
3. D. Nevo, *Making Sense of Data Through Statistics - An Introduction* (Legerity Digital Press, 2014)
4. H. Motulsky, *Intuitive Biostatistics: A Nonmathematical Guide to Statistical Thinking* (Oxford University Press, 2018), p. 568

Chapter 3
Overview of Machine Learning Algorithms

3.1 Introduction

Knowledge is an invaluable resource for almost all entities, be they firms, organizations, communities, or individuals. Knowledge needs to be captured, processed, and analyzed. Well-defined knowledge can be represented in an accurate manner, such as a mathematical formula or a certain set of rules [1]. Knowledge can also be modeled, where a model permits us to explain reality, classify objects, and predict a value (or if an event will occur) knowing its relationship to other known values. If our knowledge is not complete, then we can approximate reality by learning from previous experiences and predicting an outcome with a certain likelihood of accuracy. Alongside the representation of knowledge, we need to store on a computer a reasoning method, i.e., an algorithm (a series of steps to be followed) to process this knowledge to arrive at an outcome/output (e.g., a decision, classification, or diagnosis).

Data mining and machine learning are prominent ways to represent and process knowledge and will be introduced in the next paragraphs. We will overview the main machine learning algorithms, among other techniques, followed by examples of machine learning applications from different fields using different algorithms.

3.2 Data Mining

Data mining is a cross-disciplinary field that aims to discover novel and useful patterns within *large* datasets using multiple approaches, including machine learning, statistics, and database systems. The data mining process is automatic or semiautomatic (involves human interaction), and it must lead to patterns that are

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_3

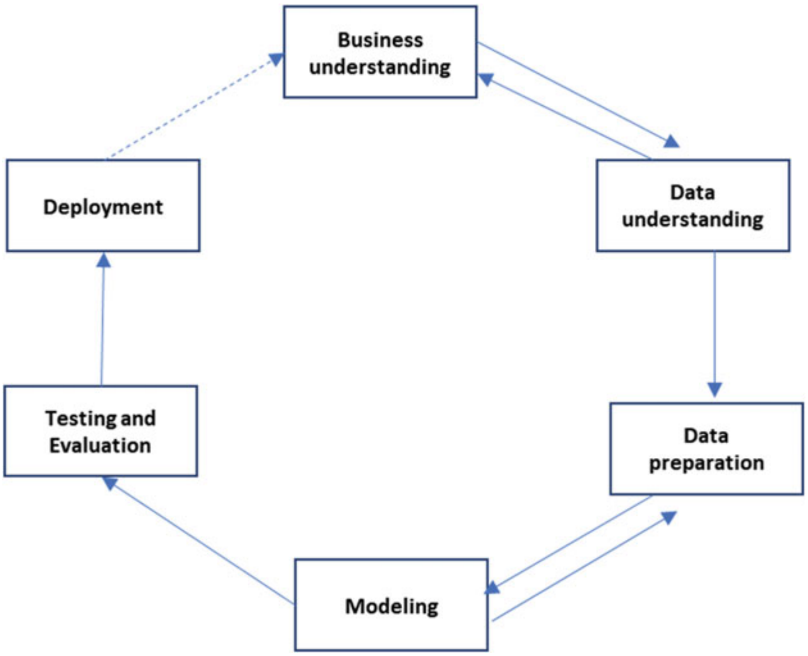


Fig. 3.1 CRISP-DM data mining life cycle (adapted from [3])

meaningful to the data stakeholders and provide some advantages (e.g., health or economic) [2].

Data mining is a process with a life cycle that can be represented by the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework (Fig. 3.1) [4, 5].

This life-cycle model consists of six phases. In the business understanding phase, the scope and objectives of the project are defined from the business/stakeholders' point of view, and then these objectives are transformed into a data mining problem with a defined plan to follow. The data understanding phase is second and entails data collection, exploratory data analysis, and evaluation of the data quality. The business and data understanding phases can be iterative, as one may be needed to understand the other. The data preparation phase involves cleaning the data and preparing it for analysis in later stages, selecting the variables and cases for analysis, and transforming data where needed. The modeling phase comprises a selection of modeling techniques and generating and fine-tuning the models. Models may require a return to the previous phase for further preparation. More details about modeling will be covered later in this chapter. During the evaluation phase, the generated models' quality and effectiveness in achieving the set objectives are evaluated, and a final decision on the adoption of a model is made. Finally, the model is deployed, which usually involves generating reports.

As we can notice, data mining automates the process of searching for patterns in a large amount of data, and modeling is a core task in data mining. Modeling can be achieved using machine learning methods.

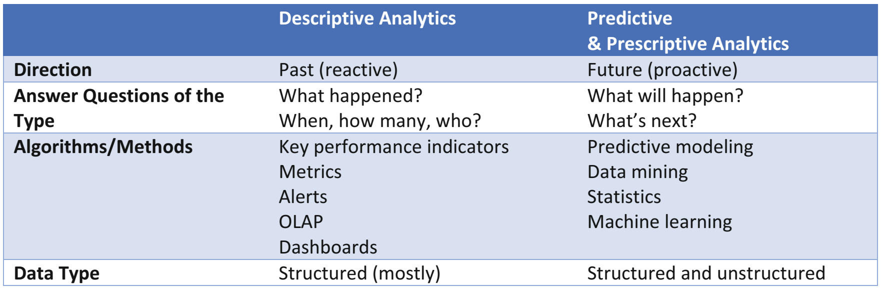
3.3 Analytics and Machine Learning

In the first chapter of this book, we introduced the concepts of descriptive, predictive, and prescriptive analytics. While descriptive analytics are reactive and focus on understanding the past, predictive and prescriptive analytics are proactive and oriented toward the future. Predictive and prescriptive analytics can be defined as the art of constructing models based on historical data and then using them to make predictions [6].

Instead of answering the "when," "who," and "how many" descriptive analytics questions, predictive and prescriptive analytics investigate "what will happen" and "what next." The former deals with *key performance indicators* (KPI) or metrics, dashboards, alerts, and OLAP (cubes, slice, dice, and drill), while the latter's analytics methods include statistical analysis, predictive modeling, and data mining. Descriptive analytics deal mainly with structured data acted upon by humans, while predictive/prescriptive analytics process structured and unstructured data that are acted upon automatically (or semi-automatically) by computer algorithms [7] (Fig. 3.2).

Machine learning is an automated process that detects patterns in data [6]; it aims to learn how to improve at tasks with experience and uses many types of techniques, such as neural networks and clustering algorithms [8].

The idea behind machine learning is that computers can "learn" to accomplish a task by applying a certain algorithm (i.e., a series of steps) to a set of examples (i.e., the training dataset). The training dataset is a partition of around 60–80% of the complete dataset. Once this training phase is performed and the model is built based on a selected machine learning algorithm, the model is tested using the testing dataset, consisting of the remaining 20–40% of the original data. In this phase, the selected model is tested for its accuracy and can be fine-tuned. Measures of



	Descriptive Analytics	Predictive & Prescriptive Analytics
Direction	Past (reactive)	Future (proactive)
Answer Questions of the Type	What happened? When, how many, who?	What will happen? What's next?
Algorithms/Methods	Key performance indicators Metrics Alerts OLAP Dashboards	Predictive modeling Data mining Statistics Machine learning
Data Type	Structured (mostly)	Structured and unstructured

Fig. 3.2 Comparison between descriptive and predictive/prescriptive analytics [7]

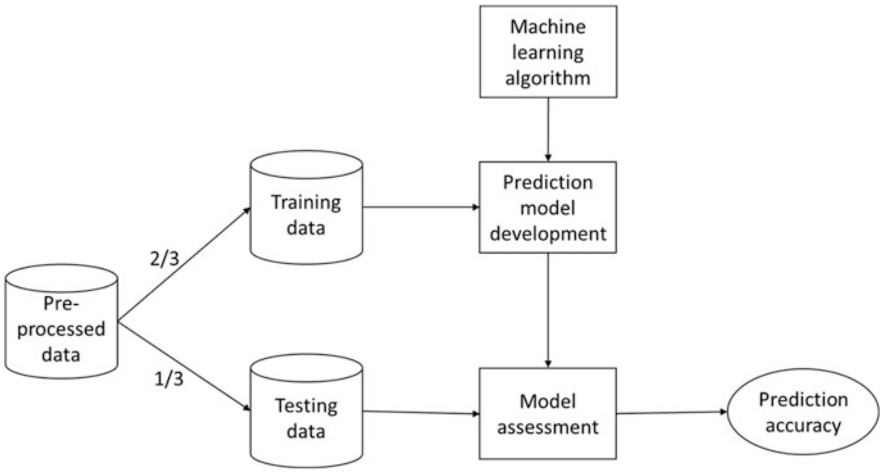


Fig. 3.3 Building a prediction model (adapted from [9])

prediction accuracy are generated and used to assess the model (Fig. 3.3) [9]. In some cases, like with artificial neural networks (ANNs), the model development consists of three phases: the model building (or training) with 60% of the dataset, the model validation phase with 20% of the data, and the model testing with the remaining data. The validation phase is used to fine-tune the model, while the testing phase provides an unbiased assessment of the model's accuracy [10]. In a nutshell, "learning" is about building a data model; the training set is the input to the machine learning algorithm, and the model is the output. Subsequently, the model is used to form predictions based on new datasets.

Prescriptive analytics models add to predictive analytics the ability not only to predict but also to explain why an event happened through a set of rules that are easy to interpret, which allows us to act based on the event using those rules. Some of the predictive analytics algorithms, such as ANNs, do not allow us to understand why a prediction was made; others will and thus allow us to create rules that are actionable (immediately usable), such as decision trees, fuzzy rule-based systems, switching neural networks (SNNs) and logic learning machines (LLMs), which are a well-known efficient implementation of SSNs. Most of these algorithms will not be covered in this book.

3.3.1 Terminology Used in Machine Learning

Machine learning aims to learn or estimate a model of the dataset we have and use that model either to predict the class of new data in the future (e.g., classification) or the value of an output (e.g., regression), or to detect patterns/groups in the data (e.g., clustering).

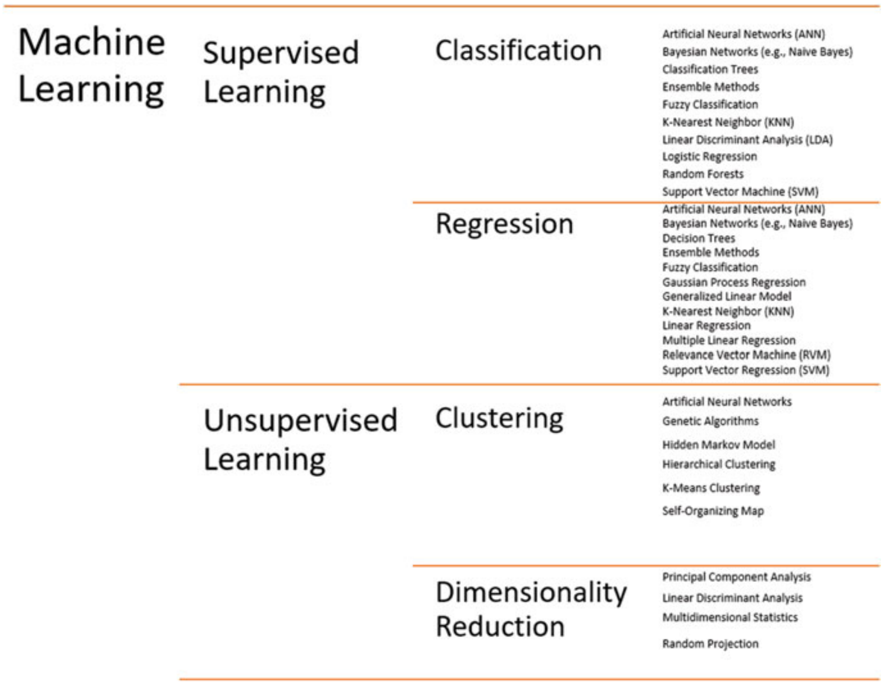
Each instance of the dataset is represented by attributes or features; for example, when an insurance company wants to estimate the risk of a driver having an accident in order to calculate a car insurance premium, an instance of a driver might be represented by (1) age, (2) gender, (3) years of driving experience, (4) number of car accidents, (5) number of driving violations, and (6) the type and model of the driven car. Each one of these data instances of a particular driver is called a feature vector; in the aforementioned example, each feature vector in the dataset is composed of six features (i.e., each feature vector has six dimensions); in other words, the dimensionality of this dataset is six.

As mentioned earlier, the dataset we use for learning is called the training dataset; learning is nothing but the process to generate a model based on the training data. It is important to remember that there is a well-known output for each vector in the training data. Suppose we are trying to predict the likelihood of a driver having an accident knowing their aforementioned six features; our training data should include each driver's previous accident features (e.g., either yes or no, or several accidents). Once the model is generated (i.e., the learning is performed), then we need to use another dataset with known output to validate the model, i.e., to assess the model's performance and fine-tune its parameters. Such a dataset is called the validation dataset. Once validation is performed, another dataset with known output is used to estimate the model error; such a dataset is called the test dataset, and the uncovered error is called the test error. A model generated for classification is called a classifier; if it is for regression, it is called a fitted regression model. Generally speaking, a model that makes predictions is called a predictor [11].

3.3.2 Machine Learning Algorithms: A Classification

The concept of machine learning is based on utilizing a wide range of algorithms to make intelligent predictions based on existing historical datasets. Many datasets are extremely large, consisting of millions of data that cannot be processed by the human mind alone [12]. Machine learning research has been very active in recent years, and it usually deals with large datasets and aims for the creation of statistical models without the need for hypothesis testing. Classifying the techniques into the four key categories (classification, regression, clustering, and dimensionality reduction) for supervised and unsupervised learning is not simple, because some techniques are used across these classes of machine learning styles; however, to organize our understanding of the field, we will rely on Fig. 3.4 [7] as a guide, noting that it is not an exhaustive list of techniques and that techniques will appear in more than one category.

The following figure gives some examples of the possible applications of supervised, unsupervised, and reinforced learning [13, 14] (Fig. 3.5).



Machine Learning	Supervised Learning	Classification	Artificial Neural Networks (ANN) Bayesian Networks (e.g., Naive Bayes) Classification Trees Ensemble Methods Fuzzy Classification K-Nearest Neighbor (KNN) Linear Discriminant Analysis (LDA) Logistic Regression Random Forests Support Vector Machine (SVM)
		Regression	Artificial Neural Networks (ANN) Bayesian Networks (e.g., Naive Bayes) Decision Trees Ensemble Methods Fuzzy Classification Gaussian Process Regression Generalized Linear Model K-Nearest Neighbor (KNN) Linear Regression Multiple Linear Regression Relevance Vector Machine (RVM) Support Vector Regression (SVM)
	Unsupervised Learning	Clustering	Artificial Neural Networks Genetic Algorithms Hidden Markov Model Hierarchical Clustering K-Means Clustering Self-Organizing Map
		Dimensionality Reduction	Principal Component Analysis Linear Discriminant Analysis Multidimensional Statistics Random Projection

Fig. 3.4 Machine learning types and applications with examples of corresponding algorithms; some algorithms can be used in many types of learning [7]

3.4 Supervised Learning

In supervised learning, the training dataset contains an input and an output/solution for each data point or record. The algorithm then "learns" how to process these data in a certain way such that it ends up with the provided solution as an output. As explained earlier, the learning process is about building a model that ultimately can predict a likely output in the absence of outputs/solutions (i.e., in the presence of uncertainty). Indeed, once learning is performed, the supervised learning software uses its learned model to provide a reasonable output/solution prediction for any new dataset input. Supervised learning algorithms can use *classification* and *regression* techniques [7].

A classification technique starts with a set of data and predicts if an output belongs to a certain category/class; for example, for a voice input or a handwriting image, it predicts the correct word; for a medical image, it predicts a certain diagnosis; for a given driver with known age, gender, years of driving experience, type and model of car, and other measures, it predicts the likelihood that they will have a car accident [7].

Some of the main techniques used for classification are:

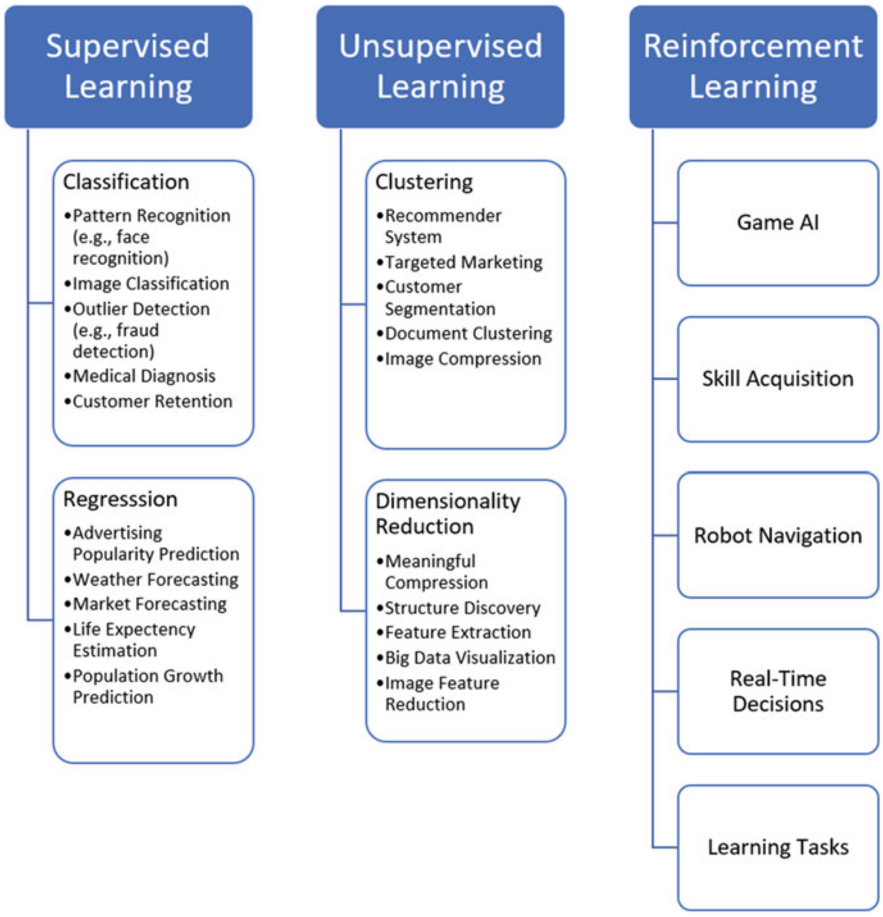


Fig. 3.5 Machine learning applications from a variety of fields (adapted from [13, 14])

1. Classification trees
2. Support vector machine
3. Random forests
4. Artificial neural networks
5. Discriminant analysis
6. Naïve Bayes
7. K-nearest neighbor
8. Logistic regression (despite its name, it is used in classification)
9. Support vector machine
10. Ensemble methods

Instead of classifying an output variable in categories, a regression technique predicts a value for that variable (i.e., predicts a solution) of a continuous nature (i.e., a number), such as the price of a house given its features, location, and market conditions, or predicting the amount of rain based on temperature, humidity, atmospheric pressure, and other predictors.

Simply understood, a regression technique approximates a function f of a data input x that produces an output $y = f(x)$; x and y are known, and f is approximated. Then, the function f is used to predict future values of y given measured values of x.

Some of the main techniques used in regression are:

1. Linear regression
2. Generalized linear model
3. Decision trees
4. Bayesian networks
5. Fuzzy classification
6. Neural networks
7. Gaussian process regression
8. Relevance vector machine
9. Support vector regression
10. Ensemble methods

A problem such as predicting the number of days a patient will be in good health after discharge from a hospital is a regression problem because we are trying to predict a number. If we are instead interested in predicting if the patient is at high or low risk of readmission to the hospital in the next 30 days, or if we are interested in predicting whether or not the patient will be readmitted to the hospital in the next 30 days, then this is a classification problem because we are trying to predict the class that the patient fits in (i.e., low risk vs. high risk, readmitted vs. not readmitted) given some of her characteristics (e.g., age, comorbidities) [7].

As you can notice, some techniques are used in both classification and regression. Below is a description of some supervised machine learning algorithms that will be covered in this book.

3.4.1 Multivariate Regression

Multivariate regression is one of the most common techniques used to create a model that links the dependent outcome variable to multiple independent variables (i.e., the predictors). Multivariate *linear* regression is used when the outcome in question is a continuous variable (i.e., a real number), such as blood pressure, cost, or weight, while multiple *logistic* regression is used when the outcome is categorical, such as blood type, or dichotomous (i.e., binary), such as readmission to the hospital (i.e., yes/no) or risk of readmission (i.e., high/low) [7].

3.4.1.1 Multiple Linear Regression

In multiple linear regression, the outcome variable (prediction) is expressed in terms of a linear function of the independent variables; for example, healthcare cost = a×(age) + b×(gender) + c×(Carlson comorbidity score) + d.

Using past data, a multiple linear regression algorithm can compute the coefficients (a, b, c, d, etc.) for the independent variables, which leads to an expression of their relationship to the dependent example; for instance, cost = 0.5×(age) + 3×(gender) + 0.2×(Carlson comorbidity score) + 4 [15]. The mathematical expression represents a *model* that ties the dependent variable (outcome) to the independent variables; the linear logistic regression model can then be used to predict the outcome (e.g., the cost) for any given values of the predictor variables. The score computed by the model can then be a multiplier for the mean (average) outcome variable in the population to predict the outcome for that particular instance/person. In the previous example, for a person whose age is 50 and gender is female (coded as 1) and who has a Carlson comorbidity score of 6, the computed cost score is 0.5×50 + 3×1 + 0.2×6 + 4 = 25 + 3 + 3 + 4 = 35; we would multiply this score (35) by the mean cost in the population for a certain period of time (e.g., a year) to predict the healthcare cost for this person in the future. In the previous example, if the average healthcare cost per year in the population of interest is $2000, then we can predict that the cost for that individual would be 35×$2000 = $75,000 [16].

3.4.1.2 Multiple Logistic Regression

Logistic regression is used to express a categorical or dichotomous variable as a function of a set of independent variables using one coefficient for each. A categorical variable is a variable that can have only a specific number of values; examples of such variables are blood type, gender, and province. Categorical variables with only two possible values are called dichotomous variables.

The model is expressed in a mathematical formula, but unlike a linear regression, the predicted value is a *probability* and hence has a value between 0 and 1 (Fig. 3.6). The logistic regression model predicts the probability that an observation falls into one of the categories of the dependent variable [15].

Multiple logistic regression is referred to as *polynomial* when it is used to predict a categorical variable, and it is referred to as *binomial* when it is used to predict a dichotomous variable. As in linear regression, a coefficient is created for each predictor variable and used in the regression model to predict individual probabilities of unknown observations' outcomes [16].

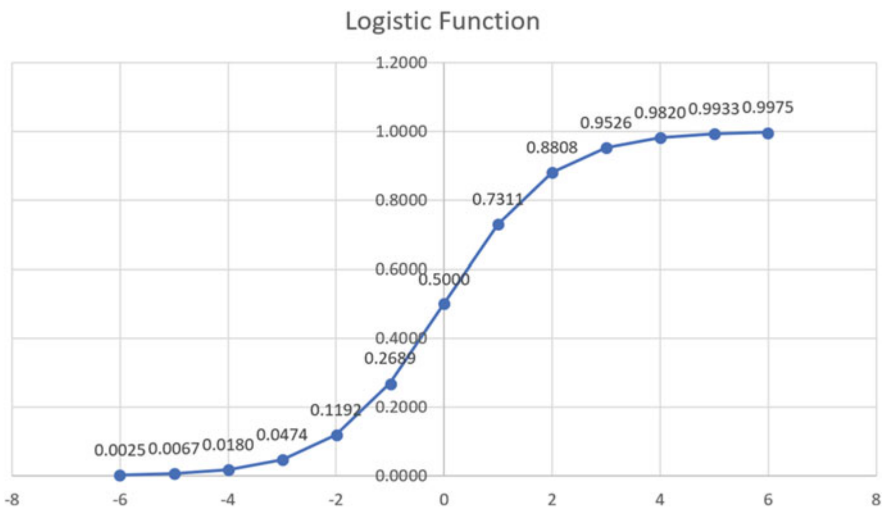


Fig. 3.6 Logistic regression function for data varying between −6 and + 6

3.4.2 Decision Trees

Decision trees have made it possible to find features and patterns in large databases that can be used for discrimination and predictive modeling. In addition to their intuitive interpretation, these characteristics have led decision trees to be widely used for both exploratory data analysis and predictive modeling for more than two decades [17].

Decision trees are simple representations of a finite number of classes. There are three parts to a tree: nodes (the name of the object), edges (the possible values for the object), and leaves (the different classes). Objects are classified by following a path down the tree, picking the edges that correspond to its values [18]. Decision trees are constructed by analyzing a set of training examples where the class labels are known. These trees are then used to classify previously unknown examples. The accuracy of their predictions depends on the quality of the training data [19].

In data mining, decision trees are commonly used for developing classification systems based on multiple covariates or for developing predictive algorithms for a target variable. By classifying a population into branch-like segments, an inverted tree is constructed with a root node, internal nodes, and leaf nodes. This algorithm is non-parametric and is capable of handling large datasets efficiently without imposing a complex parametric structure. Data from a study can be separated into training and validation datasets if the sample size is large enough. The training dataset is used to build a decision tree model, and the validation dataset is used to determine the appropriate tree size needed to achieve the ideal model [20].

Figure 3.7 depicts a very simple decision tree where the top node is called the root node, the nodes at the tip of the tree are called leaf nodes, and the rest are called

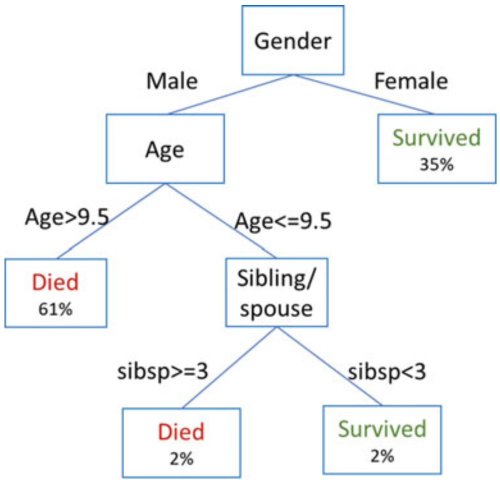


Fig. 3.7 Simple decision tree for *Titanic* survivability (adapted from [10])

interior nodes. The percentage in each leaf node represents the percentage of the data points, in this case, the *Titanic* passengers, in each node [10].

There are two main decision trees used in data mining:

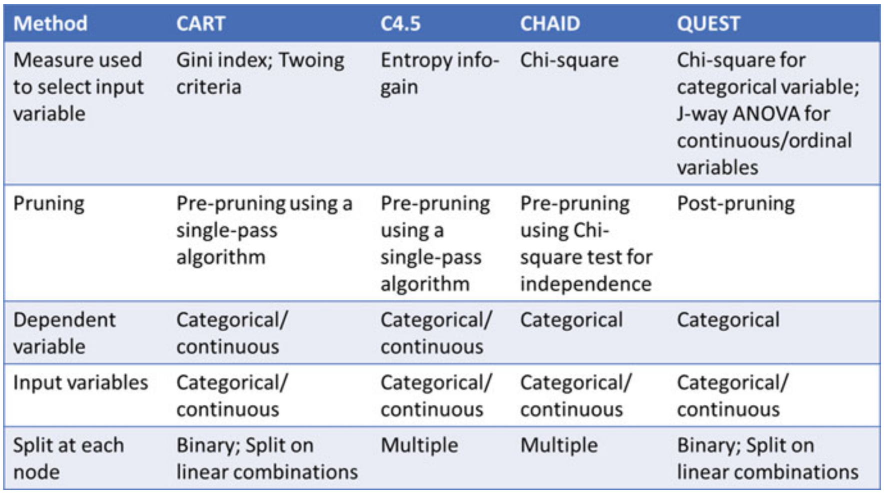
1. *Classification trees*, where the predicted outcome determines the class to which the data belongs. Examples include safe or risky outcomes for loans [21].
2. *Regression trees*, where the predicted outcome can be considered as a real number; for example, the population of a state [21].

There are several statistical algorithms for building decision trees, including CART (classification and regression trees), C4.5, CHAID (chi-squared automatic interaction detection), and QUEST (quick, unbiased, efficient, statistical tree). The CART algorithm builds both classification trees and regression trees. CART constructs the classification tree through the binary splitting of the feature. Additionally, CART is also used in regression analysis with the help of the regression tree. CART has an average processing speed and supports both nominal and continuous feature data [21].

C4.5 is an algorithm used to generate a decision tree that was developed by Ross Quinlan. It essentially generates decision trees which can then be used for classification, which is why C4.5 is frequently referred to as a statistical classifier. C4.5 performs a tree pruning process which leads to the formation of smaller trees and simpler rules and produces a significantly more intuitive analysis [21].

CHAID is an algorithm analysis of a decision tree model. This method generates a tree diagram that exhibits the relationship between split variables and their accompanying related factors [22]. The CHAID algorithm primarily regulates the results of individual defects and effectively classifies data to successfully determine the importance values of the defects [23].

QUEST is an algorithm aimed at classifying tree structures proposed by Loh and Shih in 1997. The rule in this algorithm is the notion that the target variable is



Method	CART	C4.5	CHAID	QUEST
Measure used to select input variable	Gini index; Twoing criteria	Entropy info-gain	Chi-square	Chi-square for categorical variable; J-way ANOVA for continuous/ordinal variables
Pruning	Pre-pruning using a single-pass algorithm	Pre-pruning using a single-pass algorithm	Pre-pruning using Chi-square test for independence	Post-pruning
Dependent variable	Categorical/ continuous	Categorical/ continuous	Categorical	Categorical
Input variables	Categorical/ continuous	Categorical/ continuous	Categorical/ continuous	Categorical/ continuous
Split at each node	Binary; Split on linear combinations	Multiple	Multiple	Binary; Split on linear combinations

Fig. 3.8 Decision tree methods: applications for classification and prediction (adapted from [20])

continuous. The computation speed of the QUEST algorithm is higher than those of other methods. To add on, this algorithm is also capable of avoiding the bias that may exist in other methods. Overall, the QUEST algorithm is more suitable for multiple category variables but can only process binary data [23] (Fig. 3.8).

3.4.3 Artificial Neural Networks

Artificial neural networks (ANNs) are computer technique that mimics or is inspired by the functioning of the brain's neurons. The ANN software can be trained to "learn" how to recognize patterns and classify data [11].

A neuron is a software element that uses an activation function (f), a set of *adaptive weights* ($w_0 \ldots w_n$), and data *input* ($x_0 \ldots x_n$) to generate an *output* $y = f\left(\sum_{i=1}^{n}(w_i \times x_i)\right) = f(w_0 \times x_0 + w_1 \times x_1 + w_1 \times x_1 \ldots)$. The input vector ($x_0$, x_1, $x_2 \ldots$) can be sent from another neuron or from other data sources (e.g., observations). The weights are the parameters of the data model (Fig. 3.9).

The artificial neural network is composed of one input layer of neurons, one or more hidden layers, and one output layer [15] (Fig. 3.10).

The aim of the neural network learning process, which can be either supervised or unsupervised, is to adjust the model's *weights* to arrive at the correct output, i.e., choose the correct class, arrive at the correct value, or recognize the correct patterns. In a supervised learning environment, the adjustment of the weights is performed by comparing the ANN output to a known output class for the input used; if the output is

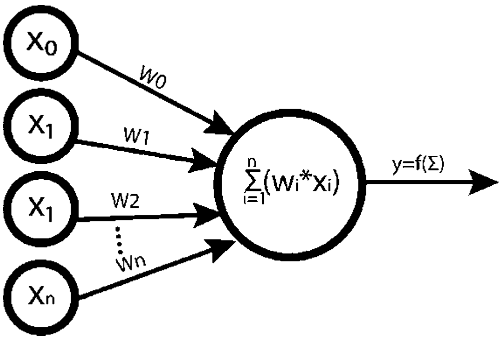


Fig. 3.9 A neuron [7]

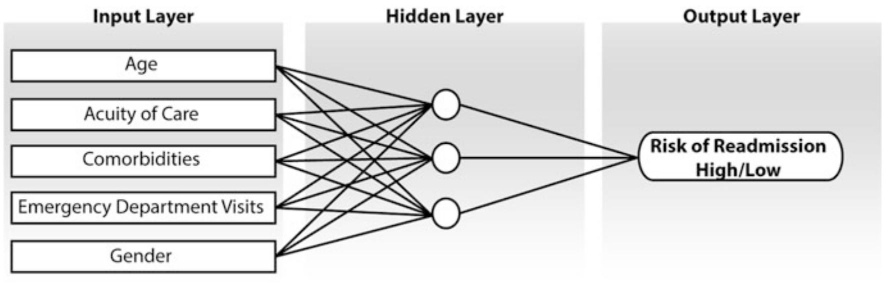


Fig. 3.10 An artificial neural network with an input layer with five nodes each for one variable of the input data, one hidden layer, and one node in the output layer representing two classes: High Risk of Hospital Readmission and Low Risk of Hospital Readmission (adapted from [24])

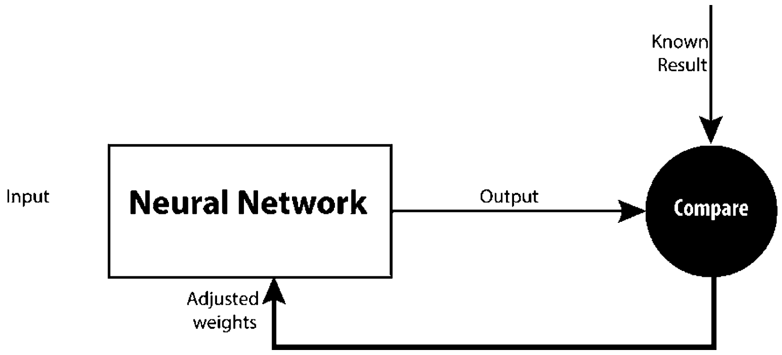


Fig. 3.11 Overall functioning of artificial neural network supervised learning [7]

not correct, then the weights are adjusted iteratively until a correct classification is found [7] (Fig. 3.11).

There are many types of ANNs; one of the most commonly used ones is the multilayer perceptron (MLP).

3.4.3.1 Perceptron

A perceptron is a simple ANN that has one input layer and one output layer (Fig. 3.12).

The perceptron model can be written as a formula $y = f$ (WX), where f is the activation function and W and X are two vectors. Indeed, the data input of m variables x_0, x_1, ... x_m can be expressed as a feature vector X:

$$X = \begin{matrix} x_0 \\ x_n \\ \vdots \\ x_m \end{matrix}$$

Similarly, the connections (i.e., weights of the model) can be expressed as a weight vector W:

$$W = w_0,\ w_1, \ldots\ w_m$$

Mathematically, the multiplication of two vectors W and X is written as WX and is equal to $w_0 \times x_0 \quad + w_1 \times x_1 \ldots \quad + w_n \times x_m$, and the perceptron output can be written as a mathematical equation $y = f\ (w_0 \times x_0 + w_1 \times x_1 \ldots + w_n \times x_m)$, where x_0 is a constant that represents the bias of the model and can be initialized to 1.

The perceptron algorithm can be iterative, where the weights are adjusted. Suppose that we have N examples from the training dataset, which we can denote as X_1, X_2, ... X_N. For each X_j ($j = 0$ to N), the desired output (class) d_j is already known (remember that, during the learning process, we use observations with known classes). For each X_j of known output d_j, we can compute the output at iteration or time t: $y_j(t) = f(\ w_0(t) \times x_{j,0} \quad + w_1(t) \times x_{j,1} \ldots \quad + w_n(t) \times x_{j,m}\ )$. When the process starts, the weights can be initialized to zero.

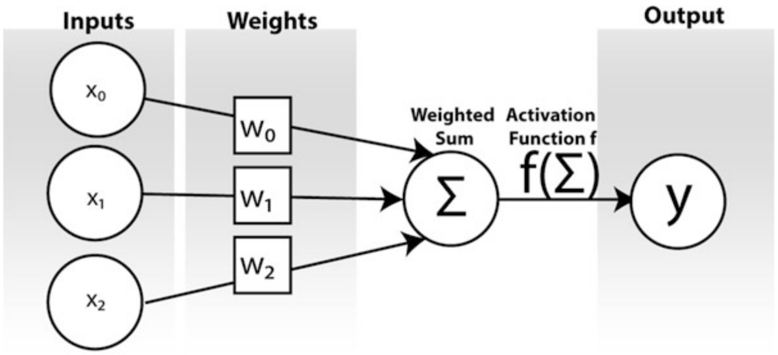


Fig. 3.12 An example of a perceptron [7]

If we are "satisfied" with the classification result (i.e., the output), then the algorithm stops. "Satisfaction" is reached when the perceptron output $y_j(t)$, at a certain iteration t, is "very close" to the known output d_j; to put it differently, the learning stops when the error (i.e., the difference between $y_j(t)$ and d_j) at a certain iteration t is less than a certain set threshold γ. After each iteration at time t, if we do not have a satisfactory solution (i.e., if the error $> \gamma$), then the weights are automatically updated using a certain "update rule" that we will not detail here, and the algorithm performs another iteration [7].

A well-established field of applications for neural networks is in medical imaging, where they are used to detect patterns of interest (e.g., cancer cells) and in image recognition (e.g., facial recognition).

A multilayer perceptron (MLP) has one or more hidden layers, similar to the ANN in Fig. 3.10; it functions like a simple perceptron (i.e., error calculation and weight adjustment). There are many activation functions (f) that can be used, as well as many update rules.

3.4.4 Naïve Bayes Classifier

Naïve Bayes is a classifier based on the Bayes conditional probability, which is an easy technique that assumes that the predictors (i.e., the attributes/features of the input variables) are statistically independent [15].

According to Bayes' theorem, with two events y and x, the posterior probability of y happening given that the event x has occurred is expressed as the product of the probability of the event x happening given y has occurred, multiplied by the probability of y and divided by the probability of x:

$$P(y|x) = \frac{P(x|y) \times P\ (y)}{P(x)}$$

(the "|" sign can be read as "given") [11].

For example, consider that the prevalence of a disease (event D) in a population is 5%; then, there is a 5% chance that any given individual from that population has the disease D. Suppose that a person conducted a blood test and that the test was positive (event P); Bayes' theorem allows us to calculate the probability that that person has the disease *given* that her test was positive: $P(D|P)$. Bayes' theorem suggests that the solution is equal to the probability of the person having a positive test given that she has the disease (i.e., a measure of what is called the test sensitivity) multiplied by the probability of that person having the disease as a member of the population (i.e., equal to the prevalence of the disease: 5%) and divided by the probability that a tested person shows a positive test in the population.

$$P(D|P) = \frac{P(P|D) \times P(D)}{P(P)}$$

Suppose we want to classify a patient using five of her features (age, blood pressure, weight, height) into one of three classes: diabetic, pre-diabetic, not diabetic. In the following, x represents one patient, C represents one of the three classes, and N represents the five patients' features.

To classify an input data x into a class C, we need a probabilistic model to estimate the posterior probability $P(C|x)$; that is, given that we have the input x, we need to estimate, for each known class C, the probability that x belongs to C; then, we need to choose the class with the highest posterior probability, i.e., the maximum a posteriori probability (MAP). We know from Bayes' theorem that:

$$P(C|x) = \frac{P(x|C) \times P(C)}{P(x)}$$

How to compute these three items $P(x|C)$, $P(C)$, and $P(x)$:

1. $P(C)$ is the probability of an output class C and can be estimated by counting the proportion of the occurrence of that class in the training dataset that we have.
2. We aim to compare different output classes for the *same* input x, so $P(x)$ is the same for all of the classes and hence can be ignored.
3. We still need to have an estimation for $P(x|C)$. To compute it, we will assume that the N features of each observation x (x_0, x_1, x_2, ... x_n) are independent of each other (which is not necessarily true/accurate), and then the laws of probabilities allow us to compute $P(x|C)$ as the product of all $P(x_i)$: $P(x|C) = \Pi_{i=1}^{n} P(x_i|C)$.

Hence, using the *training* dataset, the naïve Bayes classifier estimates the $P(C)$ for all possible classes (estimated as the proportion of that class in the training dataset), as well as the $P(\text{feature}_i|C)$ for all features in every class (estimated by the proportion of feature i in the class y).

Using the *test* dataset, a test input x will be predicted as part of class C_j if the probability of that class C_j, $P(C_j|x)$, is the highest among all classes' probabilities: $P(\mathrm{C}_j|x)$ j=1 to M, where M is the number of classes. Each $P(C_j|x)$ is computed as $P\left(C_j|x\right) = \mathrm{P}(C) \times \Pi_{i=1}^{n} P\left(x_i|C_j\right)$ (ignoring the denominator $P(x)$ because it is the same for all classes for a particular observation x) [7].

3.4.5 Random Forest

Random forests (RFs) are supervised machine learning algorithms used for regression and classification [25]. The concept of random forests gained traction after Breiman described them in 2001. In particular, he was influenced by Amit and German's "randomized trees" method as well as Ho's "random decision forests."

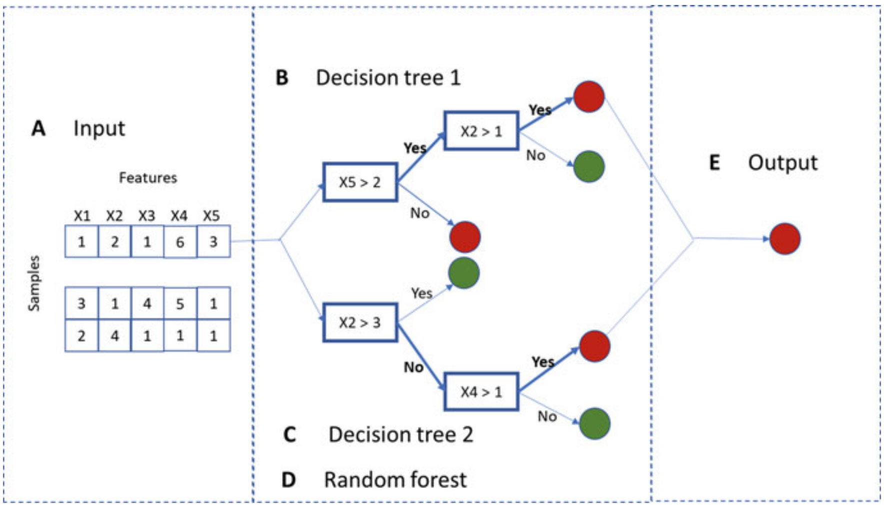


Fig. 3.13 Example of a random forest (adapted from [27])

Due to their high predictive accuracy, random forests have since proved useful in many different fields [26]. For example, through their use, we have been able to predict drug response in cancer cell lines, identify DNA-binding proteins, and localize cancer to specific tissues using liquid biopsies. Moreover, RFs have also proven to be capable of recognizing speech as well as handwritten digits with significantly high accuracy [26].

Random forests consist of trees, just as in real life. More specifically, random forests consist of decision trees. In 1963, Morgan and Sonquist developed the decision tree methodology, an intuitive approach that simplifies the analysis of multiple features during prediction tasks. A decision tree is used on an input dataset made up of samples, and each sample is described by features. These samples represent entities that we want to assign to one of several classes. Using a forking path of decision points, the decision tree classifies the samples. The branch to be taken at each decision point is determined by rules. The sample's features are analyzed at each decision point as we move down the tree. Finally, the end of the branch is reached where the leaf has a class label, and we assign the sample to that class at the end of our journey through the tree. Random forest classifiers have a high projection performance as well as the ability to reveal feature importance, showing us their importance for class prediction [26].

A random forest is a classifier containing a collection of tree-structured classifiers: $\{h(x, k), k = 1, \ldots\}$ where the $\{k\}$ are independent and identically dispersed random vectors [25]. A classifier can tell which parts are most important and how they relate or communicate with each other even when there is a lack of previous knowledge available [26].

As shown in Fig. 3.13, decision trees vote for class outcomes in an example random forest. There are five features ($x1$, $x2$, $x3$, $x4$, and $x5$) describing each sample in this input dataset (*A*). In *B*, decision trees consist of branches that fork at decision points. Depending on a feature's value, a sample is assigned to one of the decision points. The branches terminate in leaves that belong to either the red or green class. This decision tree then classifies sample 1 into the red class [26]. *C* is another decision tree that has different rules at each decision point and also classifies sample 1 into the red class. *D* is a random forest that combines the votes from its fundamental decision trees to make the final class prediction. Finally, E is the final output prediction [26].

Despite their ease of interpretation, decision trees frequently perform poorly independently. Their performance and accuracy can be improved by using a compilation of decision trees and combining the votes from each. A random forest is a compilation where we can select the best feature for splitting at each node from a random subset of the already available features. This random selection then causes the individual decision trees of a random forest to highlight different features. This then results in diverse trees that can capture more complicated feature patterns than a single decision tree could as well as being able to reduce the chances of overfitting to the available training data. This is essentially how random forests improve predictive accuracy as compared to independent decision trees. Specifically, as the number of trees increases, the error rate has been mathematically proven to always convene [26].

Key advantages of random forests as compared to AdaBoost, which is described later in this chapter, are sturdiness to noise and overfitting. Some other advantages of RFs as compared to AdaBoost include identical or better accuracy than AdaBoost, being faster, providing useful internal estimations, and simplicity [28]. When a model is constructed in a way that fits data more than is necessary, overfitting occurs. The models that have been overfitting will usually have poor predictive performance due to not generalizing well. Generalization is essentially how well the model would make predictions for cases not present in the training set. Having said that, overfitting adds complications to a model without adding any improvement and could potentially lead to poor performance. Classifiers that experience overfitting are more likely to have higher error rates for out-of-bag errors and low error rates for in-bag instances [28].

3.4.6 Support Vector Machines (SVM)

The support vector machine (SVM) is a machine learning method based on statistical learning theory and is categorized as one of the computational approaches developed by Vapnik [29]. It is essentially a supervised machine learning algorithm with the main goal of prediction. It is an abstract learning technique that learns from a set of training data and attempts to make correct predictions based on existing data [30]. According to the principle of structural risk minimization (SRM), SVM can obtain decision-making rules and achieve small errors for independent test sets, making it an efficient tool for solving learning problems. Compared to other

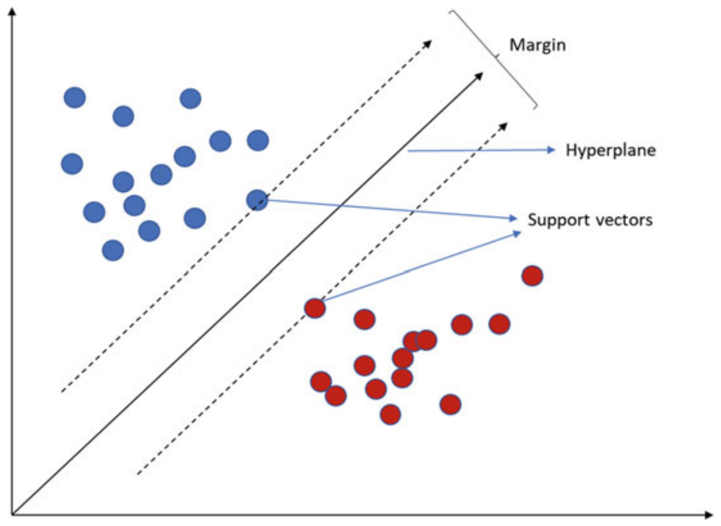


Fig. 3.14 Representation of a hyperplane in two dimensions (adapted from [32])

computational methods, SVM is generally more accurate for long-term predictions [29].

There are three main advantages of SVM theory and application [31]:

1. It only has two parameters to choose from: the upper bound and kernel parameter.
2. It is unique, optimal, and global in order to solve a linearly controlled quadratic problem.
3. It has good generalization performance because of the implementation of the SRM principle.

SVM aims to create a decision boundary between two classes, enabling the prediction of labels from one or more feature vectors. The decision boundary, also known as the hyperplane, is positioned in a way that makes it as far as possible from the closest data points to each of the classes. These closest data points are known as support vectors [31]. Decision planes, which indicate decision boundaries, are the basis of SVM. By using a linear model to implement nonlinear class boundaries, SVM generates a hyperplane at a high level with nonlinear mapping inputs [29] (Fig. 3.14).

A hyperplane in p dimensions refers to a p-1–dimensional “flat” subspace that resides inside the larger p-dimensional space. The hyperplane is only a line in two dimensions. The hyperplane is a regular 2D plane in three dimensions [33]. This is the mathematical equivalent of the equations defining a hyperplane:

Equation of a hyperplane in two dimensions: $\beta 0 + \beta 1X1 + \beta 2X2 = 0$.
Equation of a hyperplane in p dimensions: $\beta 0 + \beta 1X1 + \ldots + \beta pXp = 0$

With X as a feature vector defined by $X=[X1, \ldots., Xp]^T$, we can make a classification by considering one class defined by $\beta 0 + \beta 1X1 + \ldots + \beta pXp > 0$ with the other defined by $\beta 0 + \beta 1X1 + \ldots + \beta pX < 0$.

In practice, data is unlikely to be linearly separable, which is why we need to convert the data into a higher-dimensional space to fit a support vector classifier [33].

SVMs are more difficult to interpret in higher dimensions, since it is significantly harder to visualize how the data can be linearly separated, as well as the decision boundary [33]. The kernel method is an alternative use for SVM which allows us to model higher-dimensional, nonlinear models. The kernel trick is commonly used in order to avoid complex calculations that can also become significantly expensive. The kernel method is useful for adding dimensions to nonlinear problems, thus turning them into linear problems in the resulting higher-dimensional space [31].

The kernel trick provides the solution of modeling higher dimensions. The kernel trick involves only pairwise comparisons of the original data observations x rather than explicitly applying the transformations and representing the data in the higher-dimensional feature space. In kernel methods, dataset X is signified by an $n \times n$ kernel matrix of pairwise comparisons where the entries (i, j) are defined by the kernel function: $k(x_i, x_j)$. There is a special mathematical property associated with this kernel function. In essence, the kernel function acts as a modified dot product. The ultimate benefit of the kernel trick is that we are optimizing only the dot product of the transformed feature vectors in order to fit the higher-dimensional decision boundary. We can therefore use kernel functions to replace these dot product terms, resulting in a higher-dimensional space [33].

3.5 Unsupervised Learning

In unsupervised learning, there is no output or dependent variable for the data input. The algorithm tries to detect hidden patterns or structures and data groupings within the dataset without human intervention. Once learning is performed, the algorithm will then be able to predict the possible output or solution from a new dataset in the future. Two main techniques are used in unsupervised learning: *clustering* and *dimension reduction*.

Clustering aims to find hidden patterns and groupings within the data (i.e., input); it takes a dataset as an input and partitions it into clusters. Clustering is helpful in problems such as object recognition, gene sequencing, and market research. Clustering can, for example, identify groups of customers based on their potential profitability.

Some of the main algorithms used in regression are:

1. K-means clustering
2. Hierarchical clustering
3. Genetic algorithms

4. Artificial neural networks
5. Hidden Markov models

Dimension reduction is mainly interested in reducing the number of variables/features/attributes (number of dimensions) needed to represent the data input, thus projecting the dataset to fewer dimensions. The reduction of the data's dimensions (i.e., the number of variables) to the minimum necessary will allow a simpler representation of the data and faster processing time [7].

Some of the main algorithms used in regression are:

1. Principal component analysis
2. Linear discriminant analysis
3. Multidimensional statistics
4. Random projection

Presented with 100,000 health records for patients with congestive heart failure (CHF) with some readmission history, an algorithm that tries to group patients based on some common characteristics is a clustering algorithm that belongs to the unsupervised learning category. Alternately, if we have 50 characteristics/variables (e.g., age, weight, height, education level, and income level) for each CHF patient, it will be very complex to analyze these characteristics to detect which ones are most related to their readmission history, and we can instead use an algorithm, such as principal component analysis (PCA), to detect which (fewer) characteristics most explain the patients' readmission history. Once we have reduced the number of dimensions to a few, seven, for example, we can then proceed with further analysis of the problem [7].

Below is a description of some unsupervised machine learning algorithms that will be covered in this book.

3.5.1 K-*Means*

K-means is a common algorithm used for cluster analysis, which is an essential data mining method to classify items, concepts, or events into common groupings called clusters. *K* represents the predefined number of clusters to be generated by the algorithm. *K*-means is an exploratory data analysis tool for solving classification problems by sorting cases into *k* clusters or groups so that the degree of association in a cluster is strong among its members and weak with members of other clusters [34]. *K*-means is a common method in many disciplines including business with applications like market segmentation (classification) of customers and fraud detection [34]. In medicine, *k*-means has been used, for example, for the classification of healthcare claims of end-stage renal disease patients [35] and for examining the heterogeneity of a complex geriatric population [36]. *K*-means, where *k* represents a predetermined number of clusters and where each input belongs to the cluster with the nearest mean, is one of the most referenced clustering algorithms and is one of the best-known predictive analysis algorithms [34, 37].

K-means starts by selecting the optimal number of clusters k. There are multiple methods for selecting a value of *k*. The simplest is to compute $k = (n/2)^{1/2}$, where n is the number of data points (i.e., observations). Another method for selecting *k* is to try multiple different values, starting with 1 (where all the observations fall in the same cluster), and keep increasing the value of *k* until it reaches *n* (where each observation is in its own cluster). Obviously, the values of $k = 1$ or *n* are not useful and are not selected. At each iteration, a measure of fitness, like the average within-cluster sum of squares, is calculated. When the relative increase in fit becomes low, the corresponding value of k is adopted.

The statistical software used to apply *k*-means generates k random points as cluster centers, and each data point or record is assigned to the nearest cluster center. The mean of the data assigned for each cluster is computed and considered the new cluster center, and then the last two steps (assignment of points to the centers and computation of new centers) are repeated until convergence is achieved or the optimal centers are identified [34]. The centers are chosen to minimize the sum of the squares of the distances between a data point and the center point of its cluster and minimize the total intra-cluster variance [10]. *K*-means is simple and logical and thus is widely accepted for cluster analysis [37].

3.5.2 K-*Nearest Neighbors (KNN)*

K-nearest neighbors is a pattern recognition classifier [15]. The algorithm does not rely on a training process. Instead, it relies on a lazy learning approach for which the main principle is that similar inputs will produce the same output, i.e., will belong to the same class. Given an integer *k*, a set of already labeled examples with known classes, and a metric to measure "closeness" [15], for each input/observation in the *training* data test (i.e., a test instance), the KNN algorithm detects the k input data in the *training* dataset that is closest to it and that belong to known classes, and it classifies the observation in the majority class to which the largest number of k instances belong [11].

Figure 3.15 shows an example of a 3-NN algorithm that identified three instances closest (in terms of "distance") to the unknown instance "?"; the 3-NN would decide that the unknown instance should belong to the class "+," since it is the majority class in its three nearest neighbors (two instances belong to the class "+" vs. one instance that belongs to the class "–") [7].

KNN is also used in regression analysis to predict a value for an instance; in that case, the test instance is assigned the average values of the k instances, which is particularly useful in, for example, medical image processing, where the pixel in an image is given as a color value the average color values of its neighbors [7] (Fig. 3.16).

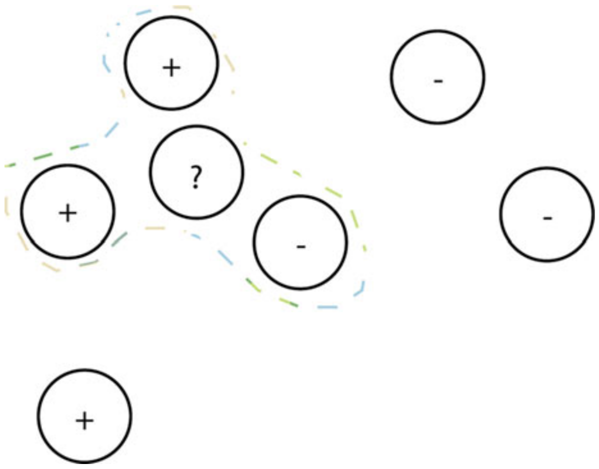


Fig. 3.15 NN algorithm making a choice to classify an unknown data instance "?" [7]

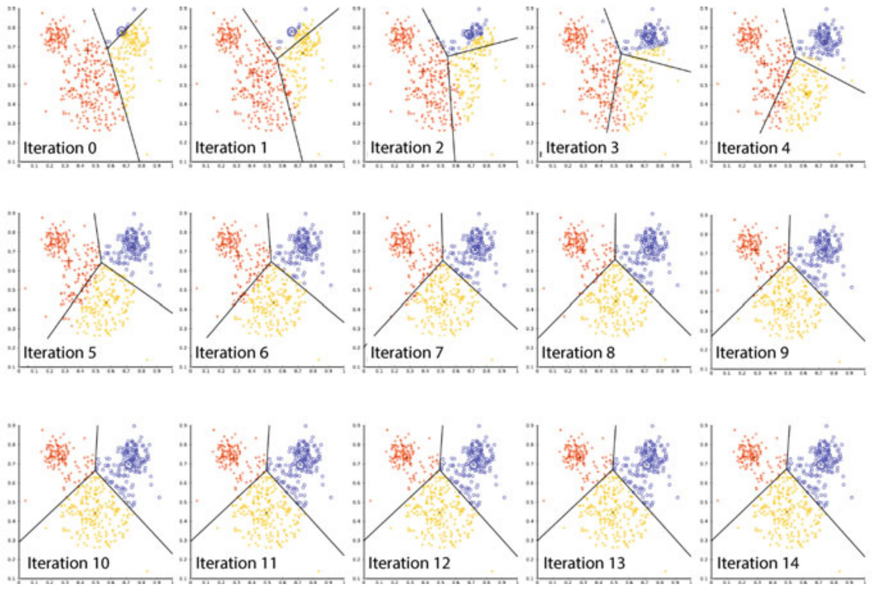


Fig. 3.16 Example of *k*-means convergence (by Chire CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0), from Wikimedia Commons)

3.5.3 AdaBoost

AdaBoost is one of the most promising machine learning algorithms due to its fast convergence and ease of implementation. It is easy to combine with other methods, such as support vector machines, to find weak hypotheses and does not require prior knowledge about weak learners. One of the main ideas of the AdaBoost algorithm is to maintain a distribution or set of weights over the training set. In the trading set, all weights are initially initialized equally, but as each round progresses, the weights of incorrectly classified examples are increased, forcing the weak learner to focus on the harder examples [38].

In 1995, Yoav Freund and Robert Schapire proposed the AdaBoost algorithm to generate a strong classifier from a set of weak ones. With the example of horse-racing gamblers, Freund and Schapire explained optimization and the solution space search. Before making his decision on which horse to bet on, the gambler would naturally seek the advice of some highly successful expert gamblers. Based on their own experiences, they would provide him with some insightful suggestions. These patterns were classifiable as a large pool of classifiers, despite their obvious roughness and inaccuracy. The question was to determine whether individuals' experiences could be used to make a better classifier for gamblers' betting. Thereafter, this issue attracted the attention of many researchers seeking valuable strategies to handle it. Kearns and Valiant were the first to ask whether a weak learning algorithm that performs only slightly better than random guessing in the PAC model could be "boosted" into a more accurate strong learning algorithm [38].

By maintaining a collection of weights over training data, the AdaBoost algorithm creates a set of poor learners and adaptively adjusts them after each weak learning cycle. As a result, the weights of the training samples that are misclassified by weak learners will be increased, while those that are correctly classified will be decreased [38].

Due to its robustness and efficiency, AdaBoost is widely used in data classification and object detection. By reweighting samples, AdaBoost combines weak classifiers to construct an optimal global classification model. Combining these two techniques improves classification performance significantly [39].

As AdaBoost becomes more popular, other variants have been suggested in order to improve its performance. Even so, there is not enough mathematical analysis of the generalization abilities for AdaBoost's variants. Real AdaBoost calculates weak hypotheses by optimizing upper bounds of training errors and convenes faster than AdaBoost in training. In 2000, Friedman et al. [40] utilized additive calculated models to explain AdaBoost and proposed Gentle AdaBoost, which processes weak hypotheses by limiting the errors. Friedman et al. likewise demonstrated that Gentle AdaBoost is more powerful than AdaBoost and Real AdaBoost. To decrease the generalization error of Gentle AdaBoost, A. Vezhnevets and V. Vezhnevets [41] recommended Modest AdaBoost, which features weak classifiers that function well on hard-to-classify occurrences. Modest AdaBoost no doubt achieves better generalization errors, but its performance is unstable due to its occasional accuracy dropping [39].

3.6 Applications of Machine Learning

In this section, four case studies of machine learning applications are explored. The first case involves the use of ML and neural networks for demand forecasting in supply chains [42]. In the second case, the potential presence of cervical pain in patients affected by whiplash is predicted for insurance-related investigations using several predictive models, including logistic regression, support vector machines, k-nearest neighbors, gradient boosting, decision trees, random forest, and neural network algorithms [43]. In the third case study, six ML predictive modeling techniques, including logistic regression (LR), linear discriminant analysis (LDA), random forests (RF), support vector machines (SVM), neural networks (NN), and random forests of conditional inference trees (CRF), were employed to predict bank insolvencies in a sample of US-based financial institutions [44]. In the final case study, a framework for skill assessment in robot-assisted surgical training based on deep learning and convolutional neural network is proposed and tested [45].

3.6.1 Machine Learning Demand Forecasting and Supply Chain Performance [42]

This case study focuses on using AI and machine learning in the field of commerce with an emphasis on demand forecasting, supply chain performance, and efficiency. A hybrid method was developed by combining time-series data with leading indicators based on machine-learning algorithms. Time-series data is basically data that is sequenced in time order, while leading indicators are measurable variables of interest that can predict data movement. The predictor variables in this case study are the machine learning forecasting approaches, while the supply chain is the outcome variable [42].

Companies in upstream stages of supply chains suffer from variance amplification, whereby there is an increase in inconsistent data as a result of demand information distortion in a multistage supply chain, which is detrimental to their performance. Recent research suggests that deploying advanced demand forecasting, such as machine learning, can mitigate the impact and improve performance. Demand forecasting refers to the process of estimating future demand over a period of time using past data. The goal of this study is to develop hybrid demand forecasting methods based on machine learning, including ARIMAX and neural networks. This research also highlights the value and importance of the ML forecasting approach to improve the supply chain efficiency performance of a supply chain. It is especially crucial since the conventional approach to forecasting demand has limited ability to take into account a variety of factors such as trends, seasonality, cyclicality, etc. ML-based forecasting methods, however, can combine learning algorithms with large datasets so that the sheer number of causal factors with nonlinear relationships can be analyzed and accounted for simultaneously. Modeling and managing supply chain operations in an uncertain environment can be simplified and facilitated by ML approaches, especially since they can handle much more complex tasks. Machine learning approaches can learn from data, make decisions based on the environment's parameters, and then continue to learn from these decisions making ML a useful tool [42].

Hypothesis: *"Compared with traditional time series–based forecasting approaches, ML-based forecasting approaches could lead to significant improvement in the efficiency of the supply chain."* [42]

Datasets from a multinational steelmaker's sales and supply chain performance were obtained for this study. The monthly sales volume panel data was obtained for the time frame of 2012–2017 and quarterly-basis data for inventory accounts receivable, accounts payable, the cost of goods sold, and revenue from Compustat Capital IQ. Compustat Capital IQ is a database containing financial and statistical information on North American companies. Various economic indicators are compiled in this database on a regional or country-by-country basis, including consumer prices, high-tech market indicators, industrial production, and goods and services trade. A total of 30 macroeconomic indicators were obtained between 2012 and 2017 [42].

To develop the ML-based forecasting model, two established ML models were used: autoregressive integrated moving average with exogenous variables (ARIMAX) and two-layer feedforward neural networks (NN) with backpropagation learning. The neural networks (NN) handle the nonlinear correlations between input variables. Autoregressive integrated moving average (ARIMA) models are ML algorithms that predict future values based on past values to perform time screening forecasting. Similarly, ARIMAX is the generic ARIMA model with the inclusion of exogenous variables, such as macroeconomic factors. Overall, ARIMA models have performed inferiorly in demand forecasting against exponential smoothing methods based on the premise that future demand is a function of the past. On the other hand, ARIMAX is based not only on past demand time series but also on leading indicators time series, making it a good fit for the ML-based forecasting model [42].

The main decision and duty of the system were essentially to predict the demand and then produce the steel based on the demand projection. Evidently, the ML-based models resulted in, on average, a 5% improvement in forecast accuracy. Given that steel manufacturing is a capital-intensive industry, a 5% improvement can have a substantial impact on supply chain efficiency. The research results reconfirmed the role of macroeconomic factors in predicting demand at the aggregate level, and the newly developed ML model was successful in capturing and integrating the complex and nonlinear relationship among many variables [42].

The results indicated that the higher level of demand forecast accuracy improved both operational and financial performance outcomes. The results also showed that the ARIMAX technique was better at predicting the peaks in demand, while neural networks generated a prediction with better accuracy. Additionally, the hybrid approach is beneficial, in which several ML techniques can be blended to create a more effective and efficient forecasting technique, largely because the blended forecast method handles the model, parameter, and data uncertainties more effectively [42].

A notable limitation of this study was the use of a single dataset to evaluate the forecasting methods rather than multiple. Even so, the main intention of this research was to contribute to the forecasting process rather than offering an ideal forecasting method [42].

3.6.2 A Case Study on Cervical Pain Assessment with Motion Capture [43]

This case study's research takes place in a healthcare setting. There is an increasing need to manage and analyze massive health data in an effective and efficient way. The technique to apply machine learning (ML) and data mining (DM) techniques in the field of healthcare will help technologists, researchers, and clinicians to create systems that provide support to represent the diagnosis, improve test treatment efficacy, as well as save resources. Having said that, DM and ML are effective tools for managing and storing health data. The sheer volume of data generated in the healthcare field is such that its processing and analysis through traditional methods is extremely complex, inefficient, time-consuming, and overwhelming. In such circumstances, data mining (DM) can play a useful role, since it can allow the discovery of patterns and trends in vast amounts of complex data. Even so, due to the complicated characteristics of the field of healthcare, an appropriate DM and ML approach adapted to these characteristics is essential. The goal of this case study of cervical assessment is to predict the potential presence of cervical pain in patients that are affected by whiplash. The presence of cervical pain is the outcome variable, while the ML process is the predictor variable [43].

Musculoskeletal disorders of the cervical spine have a high incidence and prevalence rate and are deemed a public health challenge. Similarly, cervical injuries are difficult to diagnose because high-trauma cervical spine injuries and their related symptoms are extremely diverse, which makes it hard to detect the presence of cervical pain. Predicting cervical pain presence is important not only in the healthcare field but especially in the forensic field, as the incidence and prognosis of injury from motor vehicles are relevant to insurance proceedings for pain and mental suffering. Such a tool would be able to detect the presence or absence of pain with enough accuracy in order to aid clinicians and healthcare professionals to detect further injury complications in the affected individuals helping to identify the pain and resulting in unbiased compensation for the patient. Furthermore, it can help predict pain in patients that have a high degree of anxiety and patients with hypochondriasis [43].

In this case study, the prediction model intended to predict the presence of cervical pain in patients who suffered from whiplash or cervical cancer. A real dataset was collected by assessing the movement of the cervical spine in 151 patients, of whom 60 were asymptomatic, 42 had cervical pain resulting from a traffic accident, and 49 had neck discomfort because of other causes. The medical test in the study was performed twice on each patient, giving a total of 302 samples. Additionally, all the participants were assessed with a clinical examination to verify that they met the inclusion criteria, which were that they had to be between 18 and 65 years old, not be involved in any judicial process, and have had no surgery and/or cervical fracture [43].

Using 302 samples, several supervised machine learning predictive models were generated, including logistic regression, *k*-nearest neighbors, support vector machines, decision trees, random forest, gradient boosting, and neural network algorithms. Since the aim of the study was to classify the presence or absence of cervical pain unsupervised learning algorithms were dismissed. Logistic regression basically uses logistic functions to model a binary dependent variable. Support vector machines are supervised learning models that analyzes information to classify, detect outliers, or for regression. *K*-nearest neighbors, gradient boosting and random forest algorithms are all supervised learning models used for classification and regression tasks. Decision trees can create prediction training models that use historical data for decisions. Neural network algorithms work similarly to neural networks within the human brain as computing systems for predictions [43].

Applying ML in the field of healthcare has been demonstrated through this case study of cervical pain assessment, where the prediction of the presence of cervical pain was made with accuracy, precision, and recall above 85% with methods based on ML algorithms, including SVM, random forest, MLP neural networks, and GBA. Four of the original seven models, including SVM, random forest, MLP neural network, and GBA, were able to obtain an accuracy and precision rate of more than 85% and a recall of more than 90%, meaning that the percentage of false negatives was less than 10%. The results showed that it is possible to consistently predict the presence of cervical pain with accuracy, precision, and recall above 90%. The process, which was applied to data taken from a cervical assessment study, is also appropriate for any healthcare field regardless of the origin of the data being used. It can be useful in many other medical areas such as cardiac failure, fertility tests, and other predictions in healthcare [43].

Despite the usefulness and importance of ML in the field of healthcare, limitations were detected in the field. A major limitation was how to achieve an appropriate stream of data from health organizations and hospitals, as well as the accessibility regarding privacy policies, security, authorizations, and integration of the data. Without a flow of data from health organizations, clinics, and hospitals, a lot of data is not utilized. Nevertheless, a global information collaboration between hospitals could solve this issue regarding the accessibility of information and data [43].

3.6.3 Predicting Bank Insolvencies Using Machine Learning Techniques [44]

Bank failures and insolvency have led to the monitoring and evaluation of the economic health of financial institutions by administrative authorities. In this case study, a series of machine learning (ML) modeling techniques and algorithms are used to predict bank insolvencies using a sample of US-based financial institutions. Insolvency refers to debtors being unable to pay back their debts, and similarly, bank insolvencies refer to when banks are unable to reimburse their customers or depositors. The setting for this case study concerns the field of finance, particularly, banks. Current research has produced inconclusive results with regard to whether certain capital assessment indicators are more likely to predict bank failures than others, which is why it is important to pursue this study. The predictor variable in this case study is the possibility or chance of bank insolvency, while the outcome variables are the ML modeling techniques [44].

A series of performance statistics are used in this case to assess the power of six ML predictive modeling techniques in predicting bank insolvencies, including logistic regression (LR), linear discriminant analysis (LDA), random forests (RF), support vector machines (SVM), neural networks (NN), and random forests of conditional inference trees (CRF). LR refers to a method that is used for creating corporate rating systems. LDA is basically a method of finding a linear combination of structures that characterizes and separates two or more classes of objects or events. RF is essentially a popular method of classifying problems. SVMs are a family of nonlinear, large-margin binary classifiers that estimate a hyperplane that achieves maximum separability between the data of the modeled cases. Neural networks are a widely known machine learning technique that is commonly used in credit rating classification problems. Random forests that include conditional Inference trees encompass the distributional properties of the measures when distinguishing between a significant and an insignificant improvement in the information measure [44].

A dataset covering the period 2008–2014, a 7-year period with quarterly data, was collected from the Federal Deposit Insurance Corporation (FDIC), which resulted in a dataset with more than 175,000 records. The FDIC is an independent US government agency created to maintain the stability of the financial system [44].

The model evaluation measures used in this analysis are customized to assess model performance on imbalanced samples. Model performance was assessed based on in-sample, out-of-sample, and out-of-time scenarios. A complete approach was taken to assess the survival likelihood of banks by identifying the most significant indicators that predict survival rates, and by selecting the appropriate machine learning technique that aggregates all critical data. While developing the model specifications, an extended set of variables that follow the classification groups of CAMELS (capital, asset quality, management, earnings, liquidity, and sensitivity to market risk) were examined. Particularly, capital adequacy, asset quality, management capability, earnings, liquidity, and market risk are the independent variables that were tested [44].

In addition to selecting the appropriate modeling technique, another important factor in predicting bank insolvencies is the number of explanatory variables to be considered. The financial condition of individual banks appears to be a crucial driver in distinguishing their performance during the recent financial crisis, despite the use of macroeconomic determinants to develop early warning systems for bank failures. Furthermore, supervisory authorities seek to identify bank-specific issues that may contribute to insolvency so that they can follow targeted corrective actions in each case. This study follows the same philosophy by utilizing an extended dataset of quantitative variables customized for financial institutions to differentiate and predict failing ones from non-failing ones [44].

The results indicated that RF has superior predictive performance in out-of-sample and out-of-time samples, while neural networks perform almost equally well in out-of-time samples. Based on all performance metrics, neural networks and random forests outperformed Logit and LDA. Analyzing the results across all samples, it is evident that the proposed RF rating system exhibits greater discriminatory power than all the benchmark models when the data skewness is considered. In addition, the performance obtained across all test samples is more stable, resulting in reduced performance variability [44].

The approach taken in this case has some limitations in that it uses a random forest model based only on data from US banks and exploits its capacity on European banks. To build a global rating system for banks, an enriched dataset composed of multiple jurisdictions can be analyzed in the future for better results. In the in-sample dataset, there is also a possibility that overfitting may have caused the overperformance of neural networks. Based on the out-of-sample and out-of-time data, the core conclusions of this analysis were based on the predictive performance of each model. Out-of-sample, RFs performed best across almost all performance measures. In terms of limitations, this study also does not consider whether adding macroeconomic variables to our model will improve its prediction capabilities. In order to capture the variability in the state of the entire banking system, a similar approach could be explored for multiple business cycle setups [44].

3.6.4 Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-Assisted Surgery [45]

With the invention of robot-assisted surgery, the role of data-driven methods to incorporate statistics and machine learning is growing rapidly. However, much of the existing work requires translating robot motion kinematics into intermediate features or gesture segments, which are expensive to extract, lack efficiency, and require significant domain-specific knowledge. In this case, surgical skill assessment is introduced and evaluated in order to evaluate its applicability to deep learning. Particularly, a deep surgical skill model with a novel analytical framework is proposed to directly process multivariate time series using automatic learning. This study focuses on the field of medical sciences and essentially highlights the abilities of deep architectures to create a proficient online skill assessment in modern surgical training. The predictor variables in this case study are the deep learning (DL) approaches, while the performance of skill assessment systems is the outcome variable [45].

Hypothesis: *"The learning-based approach could help to explore the intrinsic motion characteristics for decoding skills and promote optimal performance in online skill assessment systems."* [45]

Due to the growing demand for quality and safety in surgery, trainee surgeons need to attain the required expertise levels before operating on patients. An absence of proper training can considerably compromise clinical outcomes, which has been shown in several studies. Thus, efficient training and consistent methods to evaluate surgical skills are critical for supporting trainees in gaining the appropriate technical skills. Concurrently, surgical training is experiencing significant changes, with the rapid acceptance of minimally invasive robot-assisted surgery. Nevertheless, despite advances in surgical technology, most evaluations of trainee skills are still performed through outcome-based analysis, structured checklists, and rating scales. Since such evaluation requires large amounts of expert monitoring and manual ratings, it can be inconsistent due to biases in human interpretations and errors. Having said that, conventional and traditional methods are no longer adequate in advanced surgery settings, which is where machine learning (ML) and deep learning (DL) step in [45].

Deep learning, which is also referred to as deep structured learning, is essentially a set of learning methods that permit a machine to automatically process and learn from inputted data using classified layers from low to high levels. These algorithms fundamentally achieve feature self-learning to progressively discover abstract depictions during the training process. Presently, deep learning models have achieved success in fields such as strategic games, speech recognition, medical imaging, health informatics, and much more. Even so, little work has been done to study deep learning methods for surgical skill assessment [45].

The dataset used for this study contains recordings from eight surgeons with diverse robotic surgical experience. Each surgeon completed three different training tasks, namely, suturing (SU), knot-tying (KT), and needle-passing (NP), and each task was repeated five times. All three of these tasks are typically standard components in the surgical skill training curriculum [45]. The dataset comes from the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS), the only publicly available minimally invasive surgical database.

An analytical deep learning framework was proposed for skill assessment in surgical training. A deep convolutional neural network was implemented to map the multivariate time series data of the motion kinematics for individual skill levels. A deep convolutional neural network is a type of deep learning artificial neural network that deals with visual images. To implement the proposed framework, the deep learning skill model was trained from scratch, therefore, not requiring any pre-trained model. The network algorithm was applied using the Keras library with a TensorFlow backend based on Python 3.6. The Keras library is essentially an archive that provides Python interfaces for artificial neural networks. Similarly, the TensorFlow library is an artificial intelligence and machine learning archive with a focus on neural networks [45].

The proposed learning model attained a competitive accuracy of 92.5%, 95.4%, and 91.3% in the standard training tasks of suturing, needle-passing, and knot-tying, respectively. The model could successfully decode skill data from raw motion profiles via end-to-end learning without the need for engineered structures or carefully tuned gesture classification. Also, the proposed model was able to dependably understand skills within a window of 1–3 seconds without the need to observe the entire training trial. The proposed learning model successfully highlights the high potential of deep learning for a proficient online skill assessment in modern surgical training [45].

3.7 Conclusion

Paired with abundant data and advanced technology, data mining, analytics, and machine learning have gained increasing popularity due to their ability to enhance performance in any industry or field by extracting, manipulating, modeling, and analyzing data, transforming it into information that helps professionals make well-informed decisions.

In this chapter, we explored machine learning, which is the concept of utilizing a wide range of algorithms to make intelligent predictions based on existing historical datasets. We introduced the four variants of ML, known as supervised, unsupervised, semi-supervised, and reinforcement learning. We focused on the first two models of learning and introduced the key applications of classification, regression, clustering, and dimensionality reduction for supervised and unsupervised learning. For each application, several algorithms were briefly described. These algorithms will be explored in more detail in the following chapters of this book. This chapter ended by summarizing four case studies of using diverse ML algorithms and techniques for multiple purposes, in different applications, industries, and contexts.

3.8 Key Terms

1. Data mining
2. Feature
3. Attribute
4. Supervised learning

5. Multivariate regression
6. Multiple linear regression
7. Multiple logistic regression
8. Decision trees
9. Artificial neural networks
10. Perceptron
11. Naïve Bayes classifier
12. Random forest
13. Support vector machines (SVM)
14. Unsupervised learning
15. *K*-means
16. *K*-nearest neighbor (KNN)
17. AdaBoost
18. Applications of machine learning
19. Deep learning

3.9 Test Your Understanding

1. We have a customer dataset, and we want to create a model that recognizes the types of customers that spend a lot on luxury items. Is this a supervised or unsupervised learning problem?
2. We have a patient dataset, and we want to create a model that classifies the patients as at high or low risk of a heart attack. Is this a supervised or unsupervised learning problem?
3. What are some of the differences between linear and logistic regressions?
4. Give an example of a problem where a decision tree seems convenient to use.
5. Without knowing how the different machine learning algorithms function, which algorithm seems convenient to you for diagnosing a disease based on a lab test's results?
6. Medical images are composed of complex structures; which algorithm seems to you more aligned with the objective of finding complex patterns in medical images?

3.10 Read More

1. Ghassemi, M., & Mohamed, S. (2022). Machine learning and health need better values. NPJ Digit Med, 5(1), 51. https://doi.org/10.1038/s41746-022-00595-9
2. Habehh, H., & Gohel, S. (2021). Machine Learning in Healthcare. Curr Genomics, 22(4), 291–300. https://doi.org/10.2174/1389202922666210705124359
3. Hahm, K. S., Chase, A. S., Dwyer, B., & Anthony, B. W. (2021). Indoor Human Localization and Gait Analysis using Machine Learning for In-home Health

Monitoring. Annu Int Conf IEEE Eng Med Biol Soc, 2021, 6859–6862. https://doi.org/10.1109/embc46164.2021.9630761

4. Harrison, J. H., Gilbertson, J. R., Hanna, M. G., Olson, N. H., Seheult, J. N., Sorace, J. M., & Stram, M. N. (2021). Introduction to Artificial Intelligence and Machine Learning for Pathology. Arch Pathol Lab Med, 145(10), 1228–1254. https://doi.org/10.5858/arpa.2020-0541-CP
5. Khan, R., Kumar, S., Srivastava, A. K., Dhingra, N., Gupta, M., Bhati, N., & Kumari, P. (2021). Machine Learning and IoT-Based Waste Management Model. Comput Intell Neurosci, 2021, 5,942,574. https://doi.org/10.1155/2021/5942574
6. Khera, R., Haimovich, J., Hurley, N. C., McNamara, R., Spertus, J. A., Desai, N., Rumsfeld, J. S., Masoudi, F. A., Huang, C., Normand, S. L., Mortazavi, B. J., & Krumholz, H. M. (2021). Use of Machine Learning Models to Predict Death After Acute Myocardial Infarction. JAMA Cardiol, 6(6), 633–641. https://doi.org/10.1001/jamacardio.2021.0122
7. Nabi, W., Bansal, A., & Xu, B. (2021). Applications of artificial intelligence and machine learning approaches in echocardiography. Echocardiography, 38(6), 982–992. https://doi.org/10.1111/echo.15048
8. Oala, L., Murchison, A. G., Balachandran, P., Choudhary, S., Fehr, J., Leite, A. W., Goldschmidt, P. G., Johner, C., Schörverth, E. D. M., Nakasi, R., Meyer, M., Cabitza, F., Baird, P., Prabhu, C., Weicken, E., Liu, X., Wenzel, M., Vogler, S., Akogo, D., . . . Wiegand, T. (2021). Machine Learning for Health: Algorithm Auditing & Quality Control. J Med Syst, 45(12), 105. https://doi.org/10.1007/s10916-021-01783-y
9. Xie, W., Ji, M., Zhao, M., Lam, K. Y., Chow, C. Y., & Hao, T. (2021). Developing Machine Learning and Statistical Tools to Evaluate the Accessibility of Public Health Advice on Infectious Diseases among Vulnerable People. Comput Intell Neurosci, 2021, 1,916,690. https://doi.org/10.1155/2021/1916690
10. Zeng, Y., & Cheng, F. (2021). Medical and Health Data Classification Method Based on Machine Learning. J Healthc Eng, 2021, 2,722,854. https://doi.org/10.1155/2021/2722854

3.11 Lab

3.11.1 Machine Learning Overview in R

Machine learning is a set of algorithms that train data to create a model to make predictions and decisions. R is one of the programming languages used to create machine learning models; it includes machine language packages that allow the development of different models. In the following, we will cover some essential machine language packages in R.

3.11.1.1 Caret Package

Caret stands for classification, regression, and training. This package helps you to classify data by splitting it into testing and training sets. After that, an algorithm needs to be chosen in order to train the model. After training the model, it is time to predict the model and evaluate how it is performing with data. Caret package snapshot code is shown in Fig. 3.17.

3.11.1.2 ggplot2 Package

This package creates data visualizations using the basic units of graphics grammar. These basic units are divided into three features:

1. A dataset that needs to be visualized and plotted in a graph
2. Geometries that describe shapes to visualize data such as dots, bar charts, etc.
3. Visual features that need to be used in graph, such as color, fill etc.

Below in Figs. 3.18 and 3.19 is an example of how to use ggplot2 in R.

3.11.1.3 mlBench Package

Short for Machine Learning Benchmark Problems, this package includes datasets of real artificial intelligence benchmark problems, such as breast cancer, Pima Indian diabetes, etc.

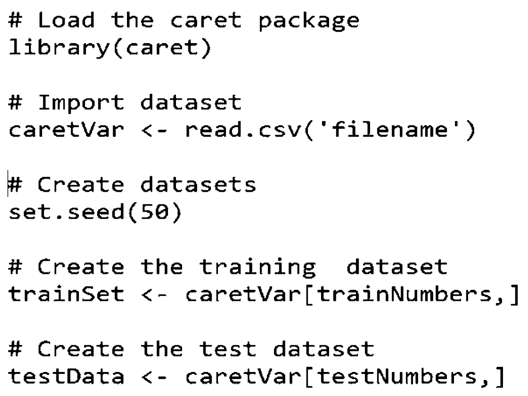
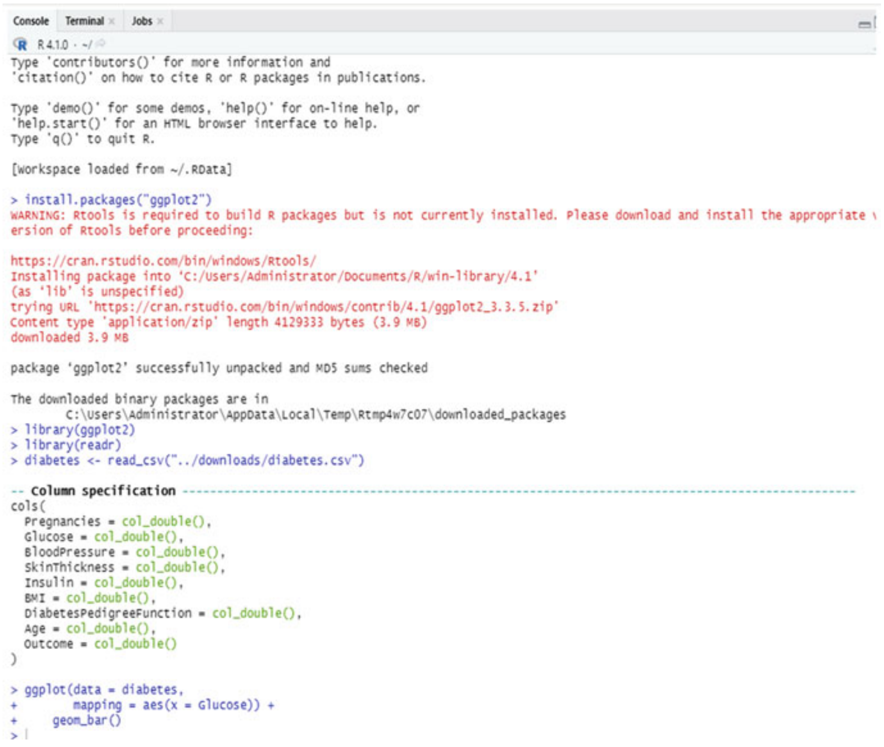


Fig. 3.17 Snapshot code of Caret package usage in R

```
# Load the caret package
library(caret)

# Import dataset
caretVar <- read.csv('filename')

# Create datasets
set.seed(50)

# Create the training  dataset
trainSet <- caretVar[trainNumbers,]

# Create the test dataset
testData <- caretVar[testNumbers,]
```



```
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> install.packages("ggplot2")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate v
ersion of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/Administrator/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/ggplot2_3.3.5.zip'
Content type 'application/zip' length 4129333 bytes (3.9 MB)
downloaded 3.9 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Administrator\AppData\Local\Temp\Rtmp4w7c07\downloaded_packages
> library(ggplot2)
> library(readr)
> diabetes <- read_csv("../downloads/diabetes.csv")

-- Column specification --------------------------------------------------------------------------------
cols(
  Pregnancies = col_double(),
  Glucose = col_double(),
  BloodPressure = col_double(),
  SkinThickness = col_double(),
  Insulin = col_double(),
  BMI = col_double(),
  DiabetesPedigreeFunction = col_double(),
  Age = col_double(),
  Outcome = col_double()
)

> ggplot(data = diabetes,
+        mapping = aes(x = Glucose)) +
+     geom_bar()
> |
```

Fig. 3.18 Function to plot glucose values in a bar chart

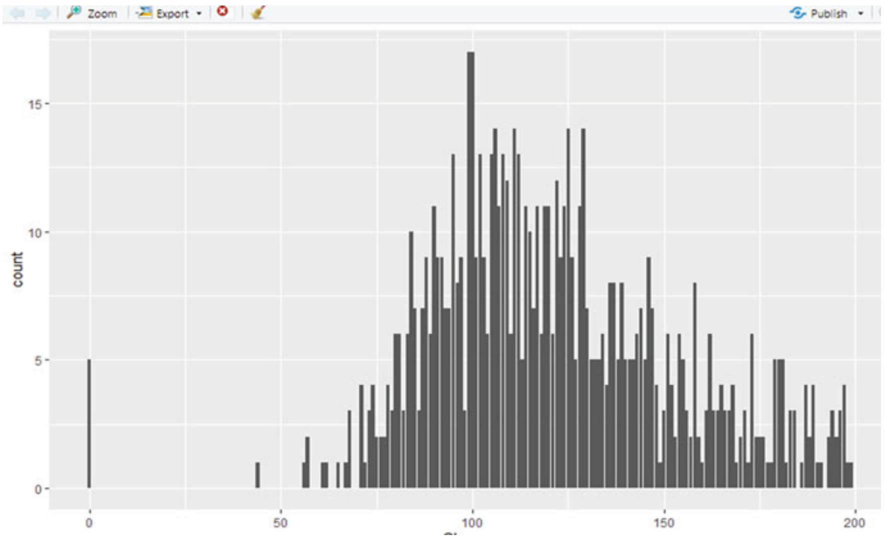


Fig. 3.19 Visualizing glucose values in a bar chart

3.11.1.4 Class Package

The Classification package contains functions that classify input data into output categories. There are different classifiers, such as *k*-nearest neighbors, neural networks, logistic regression, etc.

3.11.1.5 DataExplorer Package

The DataExplorer package includes functions that automate the handling and visualization of data. This would be the first step that allows analysts to create hypotheses to predict models. Figure 3.20 is an example where New York City flights dataset can be analyzed and visualized quickly using the DataExplorer package.

3.11.1.6 Dplyr Package

This package includes a set of functions in terms of grammar verbs to solve challenges of data manipulation. Below are a few of them:

- select(): to pick up specific variables based on names
- filter(): to filter cases based on specific variables
- arrange(): to order rows of data

3.11.1.7 KernLab Package

The kernel-based machine learning lab package includes methods for classification, regression, clustering, support vector machines, novelty detection, etc.

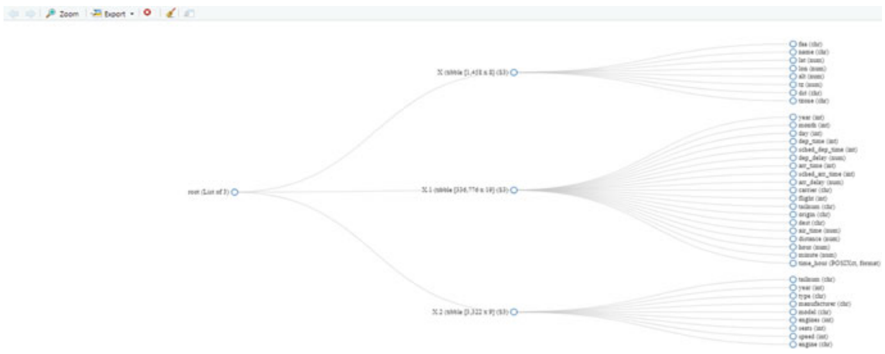


Fig. 3.20 Using DataExplorer package to visualize and analyze flights data

3.11.1.8 Mlr3 Package

This package provides building blocks for machine learning workflows. The workflow is initiated by load data. Then, these data are trained to predict the model. MLR has different algorithms, such as classification, regression, clustering, etc.

3.11.1.9 Plotly Package

The Plotly package allows you to interface with ggplot2 to create interactive web graphics for that package. Some examples include line plots, chart bars, and scatterplots. This package provides best practices to visualize data for statistical data, high-dimensional data, etc.

3.11.1.10 Rpart Package

Rpart stands for recursive partitioning and regression trees. It applies a tree decision model to classification and regression problems. Decision tree problems can be modeled using the Rpart package. The resulting model would be represented as binary trees.

3.11.2 Supervised Learning Overview

Supervised machine learning (SML) is an approach when an algorithm is trained with never-seen labeled inputs in order to predict a labeled output. SML is a great approach for classification and regression problems. For example, labeling email as spam or not spam is a binary classification. Another type of SML is regression, where the output label is quantitative. There are different algorithms for SML under classification and regression algorithms, such as *k*-nearest neighbors (KNN), decision trees, linear regression, logistic regression, etc. In the section below, the KNN algorithm will be implemented in an example step by step in order to predict the price of diamonds.

3.11.2.1 KNN Diamonds Example

The KNN Diamonds example is divided into different tasks.

3.11.2.1.1 Loading KNN Algorithm Package

In order to use the KNN algorithm in your code, the Class package needs to be installed, as shown in Fig. 3.21.

3.11.2.1.2 Loading Dataset for KNN

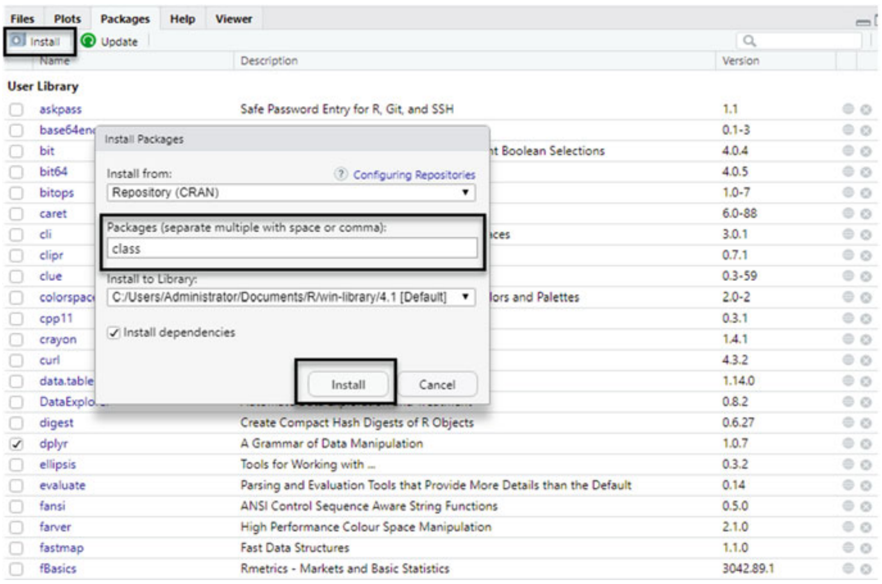
In this example, the diamonds dataset is used with the KNN algorithm. This dataset can be downloaded from the following URL: *Diamonds | Kaggle*.

This dataset consists of the following columns: index, carat, cut, color, clarity, depth, table, price, *x*, *y*, and *z*. To load this dataset to a variable, below is the code: #Import the dataset

```
dmndInput <- read.csv("C:/datasets/diamonds.csv")
```

3.11.2.1.3 Preprocessing Data

After uploading the diamonds data, these data need to be normalized in order to apply the KNN algorithm to unbiased data. It is important to note that the KNN algorithm works with numeric data only. However, the following columns are strings: cut, color, and clarity. So, it is essential to convert these columns' values



User Library

	Name	Description	Version
☐	askpass	Safe Password Entry for R, Git, and SSH	1.1
☐	base64en		0.1-3
☐	bit	t Boolean Selections	4.0.4
☐	bit64		4.0.5
☐	bitops		1.0-7
☐	caret		6.0-88
☐	cli	ces	3.0.1
☐	clipr		0.7.1
☐	clue		0.3-59
☐	colorspac	lors and Palettes	2.0-2
☐	cpp11		0.3.1
☐	crayon		1.4.1
☐	curl		4.3.2
☐	data.table		1.14.0
☐	DataExplo		0.8.2
☐	digest	Create Compact Hash Digests of R Objects	0.6.27
✓	dplyr	A Grammar of Data Manipulation	1.0.7
☐	ellipsis	Tools for Working with ...	0.3.2
☐	evaluate	Parsing and Evaluation Tools that Provide More Details than the Default	0.14
☐	fansi	ANSI Control Sequence Aware String Functions	0.5.0
☐	farver	High Performance Colour Space Manipulation	2.1.0
☐	fastmap	Fast Data Structures	1.1.0
☐	fBasics	Rmetrics - Markets and Basic Statistics	3042.89.1

Install from: Repository (CRAN) ? Configuring Repositories

Packages (separate multiple with space or comma): class

Install to Library: C:/Users/Administrator/Documents/R/win-library/4.1 [Default]

✓ Install dependencies

Fig. 3.21 Loading class package in RStudio program

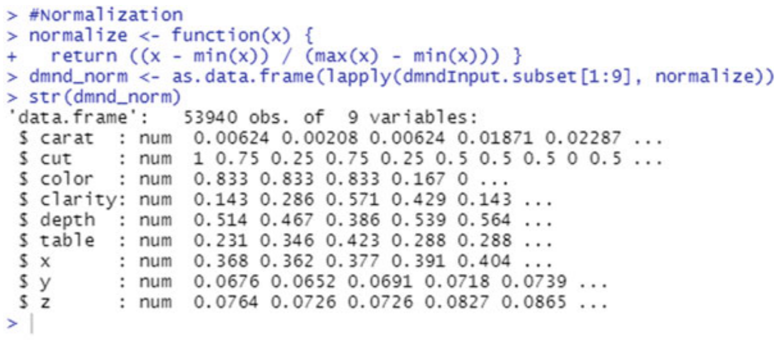
to numeric values. It is also important to note that every value for these columns is categorized. So, the values need to reflect that. The code below does the mapping based on the definition of every column at *Diamonds | Kaggle*. The dplyr and plyr packages are required to use helper functions in this mapping, such as the "mappingvalues method."

```
#Mapping string values to numeric values
require(dplyr)
require(plyr)
require(gmodels)
cut_class_dict = c('Fair', 'Good', 'Very Good', 'Premium', 'Ideal')
cut_mapped = c(1,2,3,4,5)

clarity_dict = c('I3', 'I2', 'I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2',
'VVS1', 'IF', 'FL')
clarity_mapped = c(1,2,3,4,5,6,7,8,9,10,11)

color_dict = c('J','I','H','G','F','E','D')
color_mapped = c(1,2,3,4,5,6,7)

dmndInput$cut <- mapvalues(dmndInput$cut, cut_class_dict,
cut_mapped)
dmndInput$cut <- as.integer(dmndInput$cut)

dmndInput$clarity <- mapvalues(dmndInput$clarity, clarity_dict,
clarity_mapped)
dmndInput$clarity <- as.integer(dmndInput$clarity)

dmndInput$color <- mapvalues(dmndInput$color, color_dict,
color_mapped)
dmndInput$color <- as.integer(dmndInput$color)
```

3.11.2.1.4 Scaling Data

As per the KNN algorithm definition, the next step is to label inputs and the single output. Since the index (first column) is irrelevant, it can be removed from the training set. Our target is to predict the accuracy of price. As such, the price column needs to be removed from the labeled inputs as well. So, the input data will be composed of nine columns: carat, cut, color, clarity, depth, table, *x*, *y*, and *z*. The output label will be the price column. After labeling data, it is required to normalize them, as there is a wide range of values between some columns, such as the "table" column values and "*x*" column values. This will help to train the KNN algorithm on unbiased data. Below is the R code to achieve that:

```
#Normalization
normalize <- function(x) {
 return ((x - min(x)) / (max(x) - min(x))) }
```



```
> #Normalization
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x))) }
> dmnd_norm <- as.data.frame(lapply(dmndInput.subset[1:9], normalize))
> str(dmnd_norm)
'data.frame':   53940 obs. of  9 variables:
 $ carat  : num  0.00624 0.00208 0.00624 0.01871 0.02287 ...
 $ cut    : num  1 0.75 0.25 0.75 0.25 0.5 0.5 0.5 0 0.5 ...
 $ color  : num  0.833 0.833 0.833 0.167 0 ...
 $ clarity: num  0.143 0.286 0.571 0.429 0.143 ...
 $ depth  : num  0.514 0.467 0.386 0.539 0.564 ...
 $ table  : num  0.231 0.346 0.423 0.288 0.288 ...
 $ x      : num  0.368 0.362 0.377 0.391 0.404 ...
 $ y      : num  0.0676 0.0652 0.0691 0.0718 0.0739 ...
 $ z      : num  0.0764 0.0726 0.0726 0.0827 0.0865 ...
> |
```

Fig. 3.22 Normalized data after preprocessing

```
dmnd_norm <- as.data.frame(lapply(dmndInput.subset[1:9],
normalize))
str(dmnd_norm)
```

As you can see in Fig. 3.22, the normalized data is within a close range.

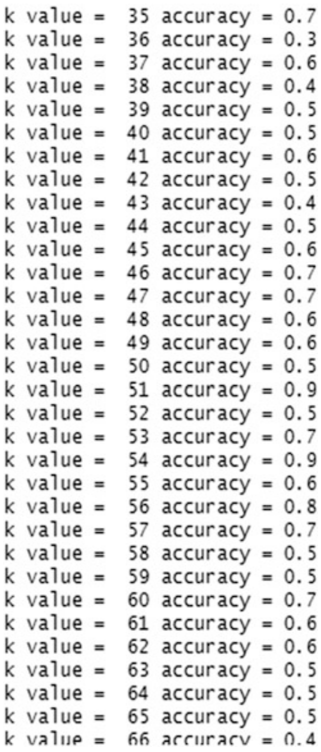
3.11.2.1.5 Splitting Data and Applying KNN Algorithm

As per the code below, a random sample is taken of 2000 observations from the dataset. This sample is split equally into test and train tests. A simple approach to choose the *k*-value is to apply this formula: *k* = sqrt(size of sample observations). In our case here, the *k*-value is around 45.

```
#split data between test and train sets
set.seed(2000)
dmnd_train <- dmnd_norm[1:1000, ]
dmnd_test <- dmnd_norm[1001:2000, ]
dmnd_target_train <- dmndInput.dmnd_target [1:1000, 1]
dmnd_target_test <- dmndInput.dmnd_target [1001:2000, 1]
```

After that, the KNN algorithm is run in a loop where the *k*-value is between 35 and 70, and price accuracy is calculated as per below:

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
require(class)
i=1
k.optm=1
for (i in 35:70){
 knn.232 <- knn(train=dmnd_train, test=dmnd_test,
cl=dmnd_target_train, k=i)
 k.optm[i] <- accuracy(table(knn.232,dmnd_target_test))
 k=i
```



```
k value =  35 accuracy = 0.7
k value =  36 accuracy = 0.3
k value =  37 accuracy = 0.6
k value =  38 accuracy = 0.4
k value =  39 accuracy = 0.5
k value =  40 accuracy = 0.5
k value =  41 accuracy = 0.6
k value =  42 accuracy = 0.5
k value =  43 accuracy = 0.4
k value =  44 accuracy = 0.5
k value =  45 accuracy = 0.6
k value =  46 accuracy = 0.7
k value =  47 accuracy = 0.7
k value =  48 accuracy = 0.6
k value =  49 accuracy = 0.6
k value =  50 accuracy = 0.5
k value =  51 accuracy = 0.9
k value =  52 accuracy = 0.5
k value =  53 accuracy = 0.7
k value =  54 accuracy = 0.9
k value =  55 accuracy = 0.6
k value =  56 accuracy = 0.8
k value =  57 accuracy = 0.7
k value =  58 accuracy = 0.5
k value =  59 accuracy = 0.5
k value =  60 accuracy = 0.7
k value =  61 accuracy = 0.6
k value =  62 accuracy = 0.6
k value =  63 accuracy = 0.5
k value =  64 accuracy = 0.5
k value =  65 accuracy = 0.5
k value =  66 accuracy = 0.4
```

Fig. 3.23 *K*-values against accuracy of price target

```
 cat('k value = ',k,'accuracy =',k.optm[i],'\n')
}
```

The result is shown in Fig. 3.23. It is noticeable that price accuracy is best when the *k*-value is between 50 and 60.

To see the data visually, the plot function may be used. This is shown in Fig. 3.24. It is important to note that the gmodels package needs to be included in the code in order to use the function.

```
 require(gmodels)
plot(k.optm, type="b", xlab="K-Value",ylab="Accuracy level")
```

3.11.2.1.6 Model Performance

Model performance or cross-validation is a mechanism to study the performance of the model when the dataset is divided into k groups. It uses one group against the test

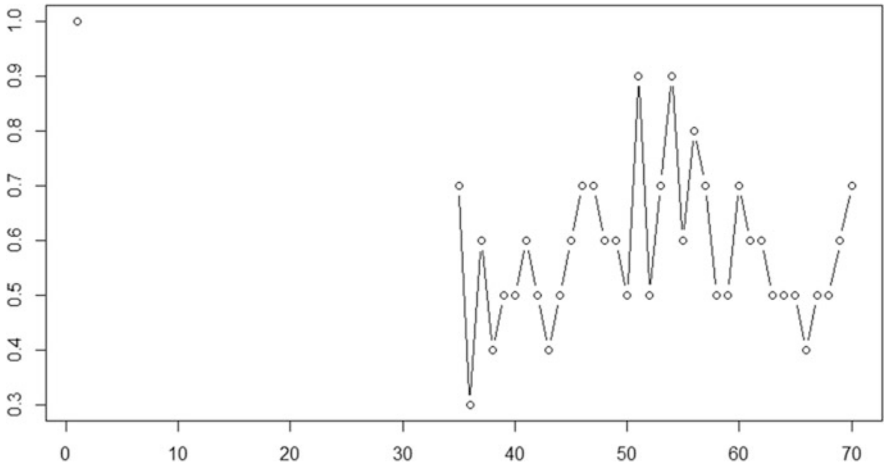
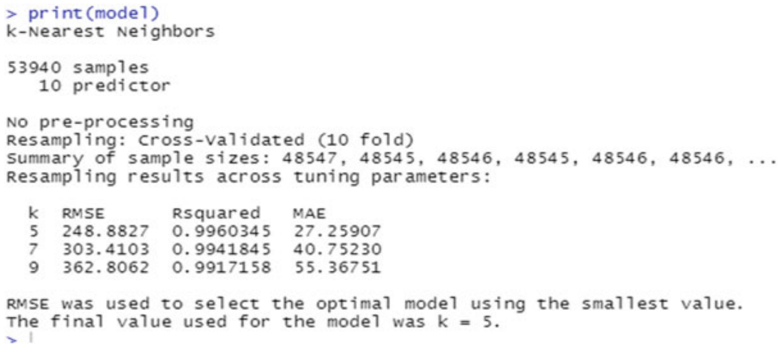


Fig. 3.24 Diamond price accuracy vs. *k*-value of KNN algorithm



```
> print(model)
k-Nearest Neighbors

53940 samples
   10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 48547, 48545, 48546, 48545, 48546, 48546, ...
Resampling results across tuning parameters:

  k  RMSE      Rsquared   MAE
  5  248.8827  0.9960345  27.25907
  7  303.4103  0.9941845  40.75230
  9  362.8062  0.9917158  55.36751

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.
>
```

Fig. 3.25 Showing cross-validation statistics for the KNN model

set and the rest against the training set. The accuracy score is applied to the test set as in the code below (see Fig. 3.25):

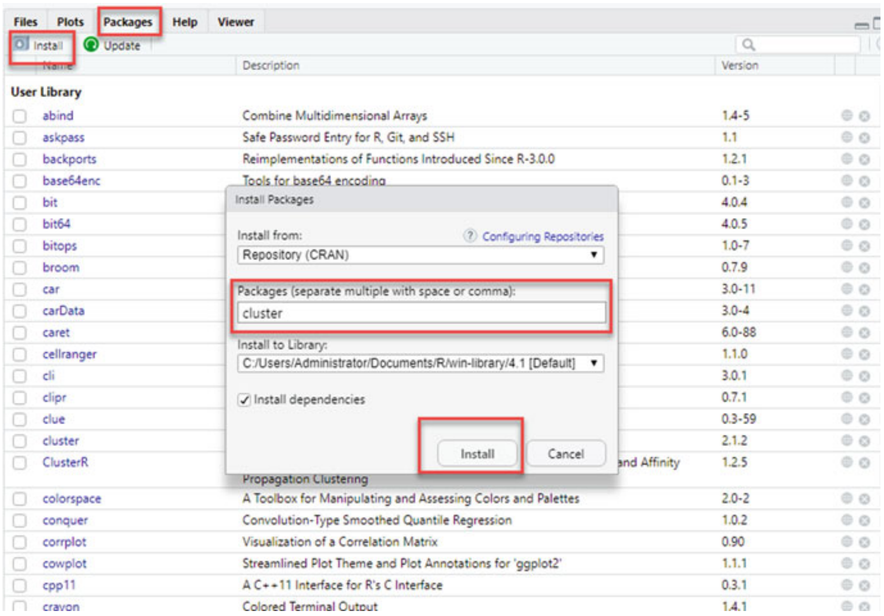
```
set.seed(42)
model <- train(price ~ ., dmndInput,
   method = "knn",
   trControl = trainControl(method = "cv",
  number = 10,
  verboseIter = FALSE))
print(model)
```

3.11.3 Unsupervised Learning Overview

Unsupervised machine learning (UML) is a set of algorithms that are run on data without any labels and predict the hidden pattern in data information. The main challenge of UML is that there is no output labeled goal. There are two types of UML: clustering and dimensionality reduction. The clustering algorithm is an approach to find homogeneous groups within a dataset. On the other hand, dimensionality reduction is an approach to reduce the dimension if it is too large. This is usually used in the preprocessing stage of supervised machine learning to work with a manageable dataset. In the next section, we will work on the UML example step by step. The UML algorithm used below is the *k*-means clustering one.

3.11.3.1 Loading *K*-Means Clustering Package

In order to use *k*-means helper functions, the following packages need to be installed: Cluster and ClusterR, as shown in Figs. 3.26 and 3.27.



Name	Description	Version
User Library		
abind	Combine Multidimensional Arrays	1.4-5
askpass	Safe Password Entry for R, Git, and SSH	1.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
base64enc	Tools for base64 encoding	0.1-3
bit		4.0.4
bit64		4.0.5
bitops		1.0-7
broom		0.7.9
car		3.0-11
carData		3.0-4
caret		6.0-88
cellranger		1.1.0
cli		3.0.1
clipr		0.7.1
clue		0.3-59
cluster		2.1.2
ClusterR	and Affinity Propagation Clustering	1.2.5
colorspace	A Toolbox for Manipulating and Assessing Colors and Palettes	2.0-2
conquer	Convolution-Type Smoothed Quantile Regression	1.0.2
corrplot	Visualization of a Correlation Matrix	0.90
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.1
cpp11	A C++11 Interface for R's C Interface	0.3.1
crayon	Colored Terminal Output	1.4.1

Install Packages

Install from: Configuring Repositories

Repository (CRAN)

Packages (separate multiple with space or comma):

cluster

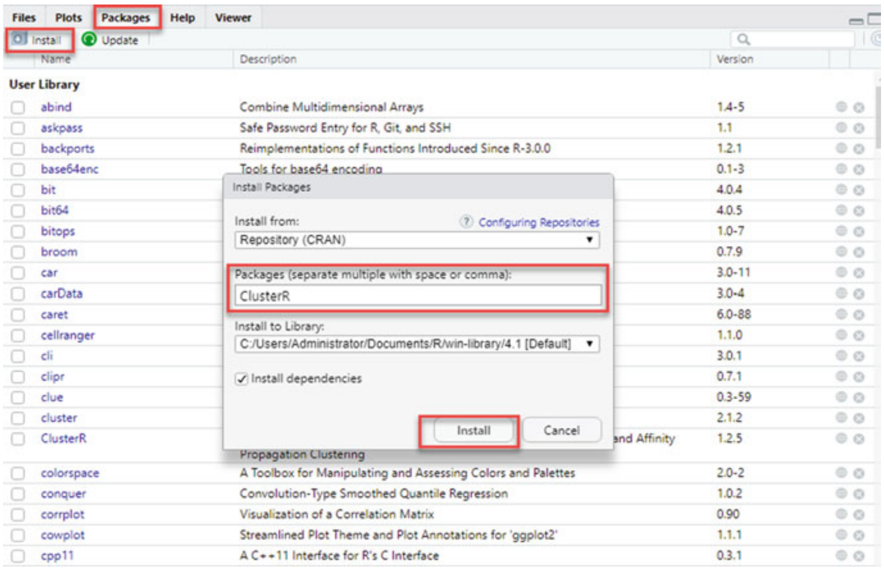
Install to Library:

C:/Users/Administrator/Documents/R/win-library/4.1 [Default]

[✓] Install dependencies

Install Cancel

Fig. 3.26 Installing Cluster package in RStudio


Files Plots Packages Help Viewer

Install Update

Name	Description	Version
User Library		
abind	Combine Multidimensional Arrays	1.4-5
askpass	Safe Password Entry for R, Git, and SSH	1.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
base64enc	Tools for base64 encoding	0.1-3
bit		4.0.4
bit64		4.0.5
bitops		1.0-7
broom		0.7.9
car		3.0-11
carData		3.0-4
caret		6.0-88
cellranger		1.1.0
cli		3.0.1
clipr		0.7.1
clue		0.3-59
cluster		2.1.2
ClusterR	and Affinity Propagation Clustering	1.2.5
colorspace	A Toolbox for Manipulating and Assessing Colors and Palettes	2.0-2
conquer	Convolution-Type Smoothed Quantile Regression	1.0.2
corrplot	Visualization of a Correlation Matrix	0.90
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.1
cpp11	A C++11 Interface for R's C Interface	0.3.1

Install Packages

Install from: Configuring Repositories

Repository (CRAN)

Packages (separate multiple with space or comma):

ClusterR

Install to Library:

C:/Users/Administrator/Documents/R/win-library/4.1 [Default]

✓ Install dependencies

Install Cancel

Fig. 3.27 Installing ClusterR package in RStudio

3.11.3.2 Loading Dataset for *K*-Means Clustering Algorithm

Edgar Anderson's iris dataset is used in this example to apply the *k*-means algorithm. This dataset can be downloaded from the following URL location: *Iris Species | Kaggle*. The iris dataset consists of five columns: sepal length, sepal width, petal length, petal width, and species. To load this dataset to a variable, use the below code:

```
#Import the dataset
Input <- read.csv("C:/datasets/iris.csv")
```

3.11.3.3 Preprocessing Data

The second step in the process is to clean the data by omitting missing data with null values and removing columns that are irrelevant to the *k*-means clustering algorithm's computing:

```
#Remove species column from training set
Inpt.subset <- Inpt[, -5]

#Remove data with missing values
Inpt.subset <- na.omit(Inpt.subset)
```

```
#Scaling data
Inpt.subset <- scale(Inpt.subset)
```

3.11.3.4 Executing *K*-Means Clustering Algorithm

As per the code below, a random sample of 100 observations is taken initially. Please note that the *kmeans* function below is taking the data frame, the number of clusters, and 20 different initial samples as parameters. A confusion matrix is calculated below to learn the performance of the algorithm. These data are plotted visually as well.

```
set.seed(100) # Setting seed
kmeans.result <- kmeans(Inpt.subset, centers = 3, nstart = 20)

# Calculating Confusion Matrix
cMatrix <- table(Inpt$species, kmeans.result$cluster)
cMatrix

## Visualizing clusters
y_kmeans <- kmeans.result$cluster
clusplot(Inpt.subset[, c("sepal_length", "sepal_width")],
  y_kmeans,
  lines = 0,
  shade = TRUE,
  color = TRUE,
  labels = 2,
  plotchar = FALSE,
  span = TRUE,
  main = paste("Kmeans Iris Clusters"),
  xlab = 'sepal_length',
  ylab = 'sepal_width')
```

3.11.3.5 Results Discussion

As per Fig. 3.28, data are clustered into three groups. The groups are related to each other in terms of sepal length and sepal width to predict the species: *Iris setosa*, *Iris versicolor*, or *Iris virginica*. As you can see, the data analysis would predict the species category without any labeling.

3.11.4 Python Scikit-Learn Package Overview

The Scikit-Learn package is the most important package in Python to implement machine learning algorithms. It is an open-source library for machine learning. It provides tools and selection of different algorithms such as classification, regression,

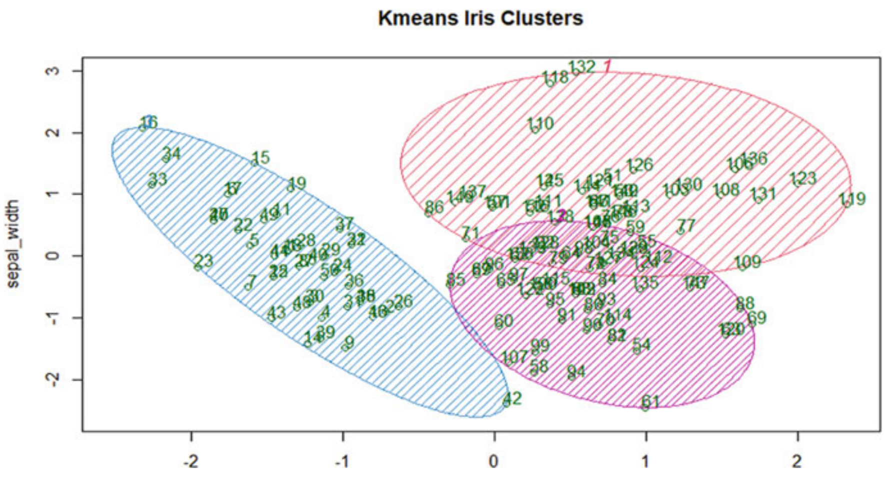


Fig. 3.28 Iris plot for *k*-means algorithm

clustering, and dimensionality reduction. The following URL *Getting Started—scikit-learn 0.24.2 documentation* contains all information, installation instructions, and examples of this package's usage.

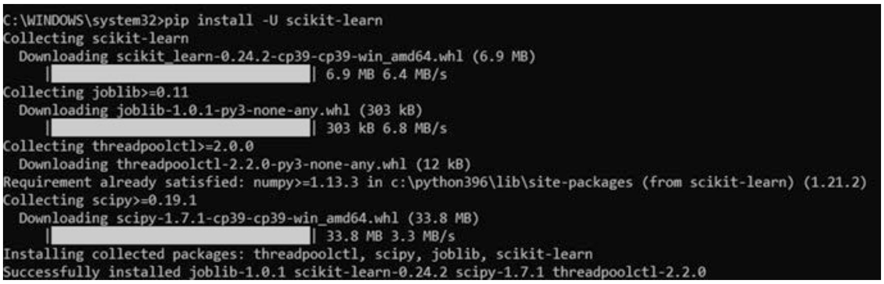
3.11.5 Python Supervised Learning Machine (SML)

In this section, we will work on a linear regression algorithm example step by step using Python and the Scikit-Learn package. This package contains many machine learning algorithms as well as their function helpers to create the model. As mentioned earlier, SML is about labeling inputs and an output to predict the goal in dataset. We will use diamonds dataset in the Python linear regression example.

3.11.5.1 Using Scikit-Learn Package

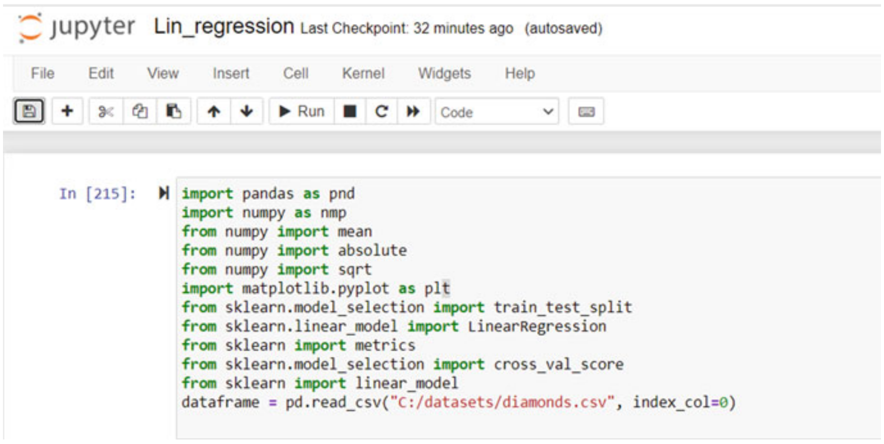
The first thing to do in this example is to ensure all required packages are installed for use, especially the Scikit-Learn package. In order to install this package, a Windows or Mac user needs to execute the following command as shown in Fig. 3.29:

```
pip install -U scikit-learn
```



```
C:\WINDOWS\system32>pip install -U scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-0.24.2-cp39-cp39-win_amd64.whl (6.9 MB)
     |████████████████████████████████| 6.9 MB 6.4 MB/s
Collecting joblib>=0.11
  Downloading joblib-1.0.1-py3-none-any.whl (303 kB)
     |████████████████████████████████| 303 kB 6.8 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.2.0-py3-none-any.whl (12 kB)
Requirement already satisfied: numpy>=1.13.3 in c:\python396\lib\site-packages (from scikit-learn) (1.21.2)
Collecting scipy>=0.19.1
  Downloading scipy-1.7.1-cp39-cp39-win_amd64.whl (33.8 MB)
     |████████████████████████████████| 33.8 MB 3.3 MB/s
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.0.1 scikit-learn-0.24.2 scipy-1.7.1 threadpoolctl-2.2.0
```

Fig. 3.29 Installing Scikit-Learn package for machine learning examples


jupyter Lin_regression Last Checkpoint: 32 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [215]:  import pandas as pnd
           import numpy as nmp
           from numpy import mean
           from numpy import absolute
           from numpy import sqrt
           import matplotlib.pyplot as plt
           from sklearn.model_selection import train_test_split
           from sklearn.linear_model import LinearRegression
           from sklearn import metrics
           from sklearn.model_selection import cross_val_score
           from sklearn import linear_model
           dataframe = pd.read_csv("C:/datasets/diamonds.csv", index_col=0)
```

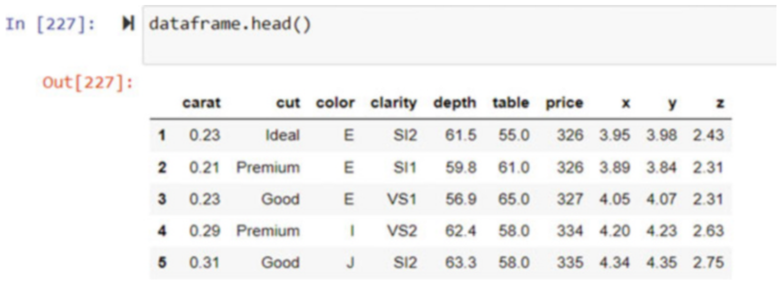
Fig. 3.30 Importing required packages and loading diamonds dataset

For further details, you might check the Scikit-Learn website that was mentioned above. After the packages' installation, the Jupyter notebook application needs to be launched to create the KNN algorithm model.

3.11.5.2 Loading Diamonds Dataset Using Python

After installing all required packages for this example, the first step is to load the dataset into a variable, as shown in Fig. 3.30.

Running the "head" function will show the first few rows of the dataset, as shown in Fig. 3.31.

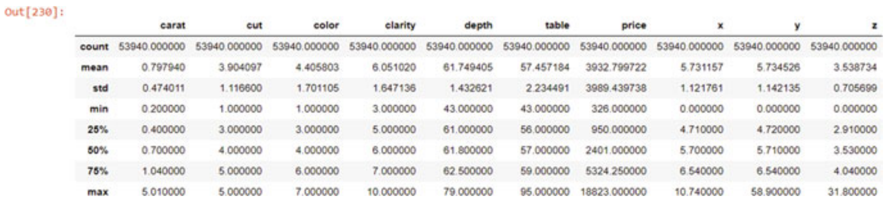


```
In [227]:  dataframe.head()
```

Out[227]:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

Fig. 3.31 Showing details about diamonds dataset



Out[230]:

	carat	cut	color	clarity	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	3.904097	4.405803	6.051020	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.116600	1.701105	1.647136	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	1.000000	1.000000	3.000000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	3.000000	3.000000	5.000000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	4.000000	4.000000	6.000000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	5.000000	6.000000	7.000000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	5.000000	7.000000	10.000000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Fig. 3.32 Describe function showing details about dataset

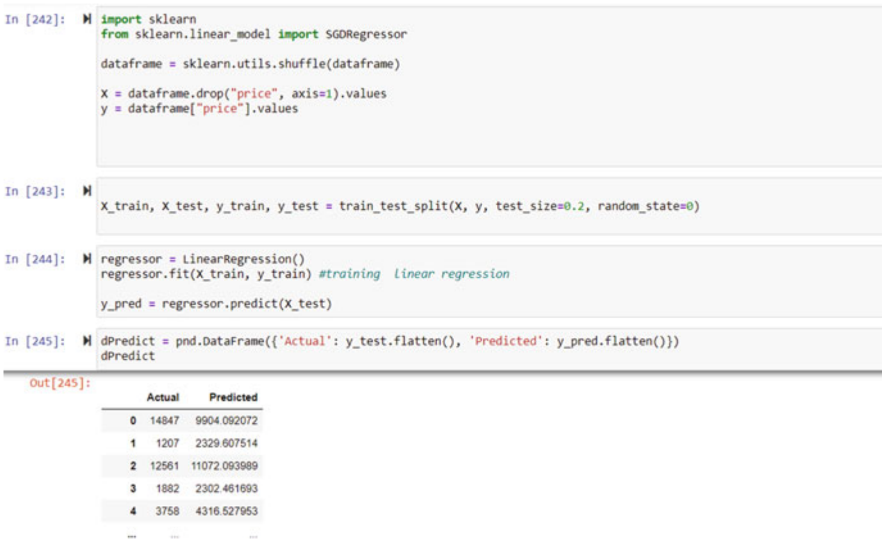
3.11.5.3 Preprocessing Data

Before executing the linear regression algorithm on the diamonds dataset, it is important to remove noise and map string values into weighted numeric values, as shown below in the code. The describe function shows details about the dataset in Fig. 3.32.

```
 clarity_dict = {"I3": 1, "I2": 2, "I1": 3, "SI2": 4, "SI1": 5, "VS2":
6, "VS1": 7, "VVS2": 8, "VVS1": 9, "IF": 10, "FL": 11}
color_dict = {"J": 1,"I": 2,"H": 3,"G": 4,"F": 5,"E": 6,"D": 7}
cut_class_dict = {"Fair": 1, "Good": 2, "Very Good": 3, "Premium":
4, "Ideal": 5}
dataframe['cut'] = dataframe['cut'].map(cut_class_dict)
dataframe['clarity'] = dataframe['clarity'].map(clarity_dict)
dataframe['color'] = dataframe['color'].map(color_dict)
```

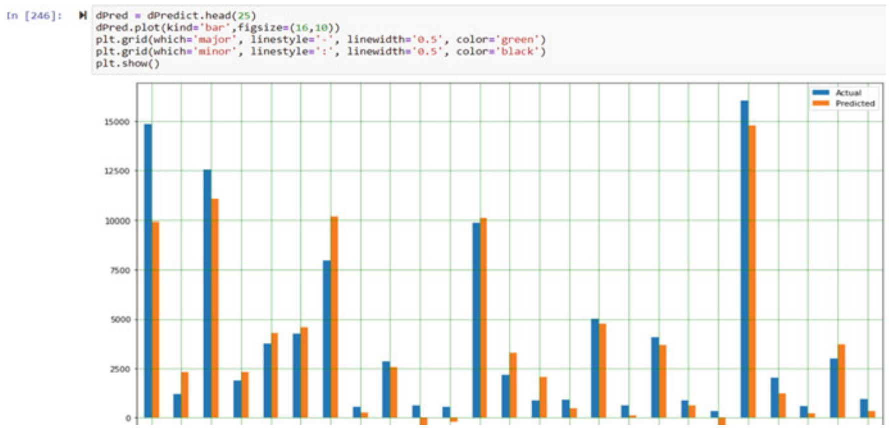
3.11.5.4 Splitting Data and Executing Linear Regression Algorithm

The first task to do before splitting the data is labeling inputs and an output. As you can see below in Fig. 3.16, the price column is removed from the training set. Also, this column is labeled as the output or goal for prediction. The next step is to split data between the train and test sets and apply the linear regression algorithm to the



```
In [242]:  import sklearn
           from sklearn.linear_model import SGDRegressor

           dataframe = sklearn.utils.shuffle(dataframe)

           X = dataframe.drop("price", axis=1).values
           y = dataframe["price"].values

In [243]:
           X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

In [244]:  regressor = LinearRegression()
           regressor.fit(X_train, y_train) #training  linear regression

           y_pred = regressor.predict(X_test)

In [245]:  dPredict = pnd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
           dPredict

Out[245]:
                Actual      Predicted
           0    14847     9904.092072
           1     1207     2329.607514
           2    12561    11072.093989
           3     1882     2302.461693
           4     3758     4316.527953
         ...      ...             ...
```

Fig. 3.33 Splitting data and executing linear regression algorithm



```
In [246]:  dPred = dPredict.head(25)
           dPred.plot(kind='bar',figsize=(16,10))
           plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
           plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
           plt.show()
```

Fig. 3.34 Bar diagram for predicted vs. actual diamond prices

train set. You might note below in Fig. 3.33 that the test set is 20% of the dataset to leave 80% for the train set.

The data can then be visualized in a bar diagram that will help to analyze the performance of the model against this dataset. The bar diagram for predicted price vs. actual price is shown in Fig. 3.34.

3.11.5.5 Model Performance Explanation

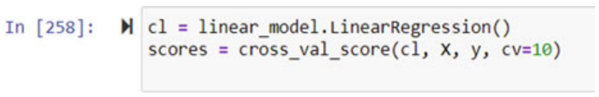
Cross-validation (CV) is a tool used in SML to measure the performance of the model, as mentioned earlier, in order to decide if that model is overfitting or underfitting using the dataset. As shown in Fig. 3.35, a k-fold of 10 is used for cross-validation to get different scores and evaluate the performance of the model.

3.11.5.6 Classification Performance

The classification performance measure is used to improve the model and make it fit the dataset. This is done by calculating the mean absolute error, mean squared error, root mean squared error, and cross-validation score. As per the values in Fig. 3.36, root mean squared error is high compared to the mean of the actual price. As you can also see, the CV score is around 0.9 for the k-ten fold, which means the model is overfitting and not working well on a new previously unseen dataset. As such, it appears that the diamonds dataset is not applicable to real-world data.

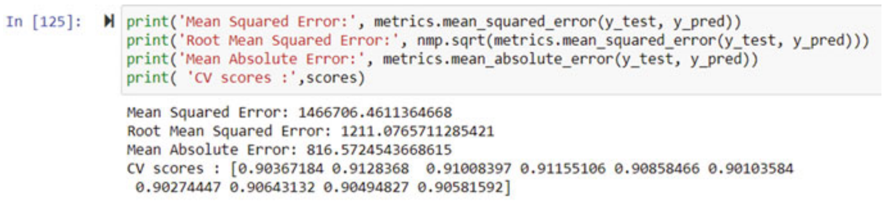
3.11.6 Unsupervised Machine Learning (UML)

As discussed earlier, UML is a set of algorithms that find patterns or trends within data without any labeled input or output. In this section, we will work on executing the hierarchical clustering algorithm example step by step on the iris data.



```
In [258]:  cl = linear_model.LinearRegression()
           scores = cross_val_score(cl, X, y, cv=10)
```

Fig. 3.35 Calculating cross-validation score



```
In [125]:  print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
           print('Root Mean Squared Error:', nmp.sqrt(metrics.mean_squared_error(y_test, y_pred)))
           print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
           print( 'CV scores :',scores)

           Mean Squared Error: 1466706.4611364668
           Root Mean Squared Error: 1211.0765711285421
           Mean Absolute Error: 816.5724543668615
           CV scores : [0.90367184 0.9128368  0.91008397 0.91155106 0.90858466 0.90103584
            0.90274447 0.90643132 0.90494827 0.90581592]
```

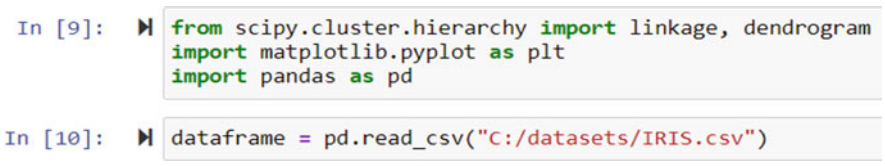
Fig. 3.36 Calculating performance features for the model

3.11.6.1 Loading Dataset for Hierarchical Clustering Algorithm

Edgar Anderson's iris dataset is used for executing the hierarchical clustering algorithm. The first step is to import the required packages and the dataset, as shown in Fig. 3.37.

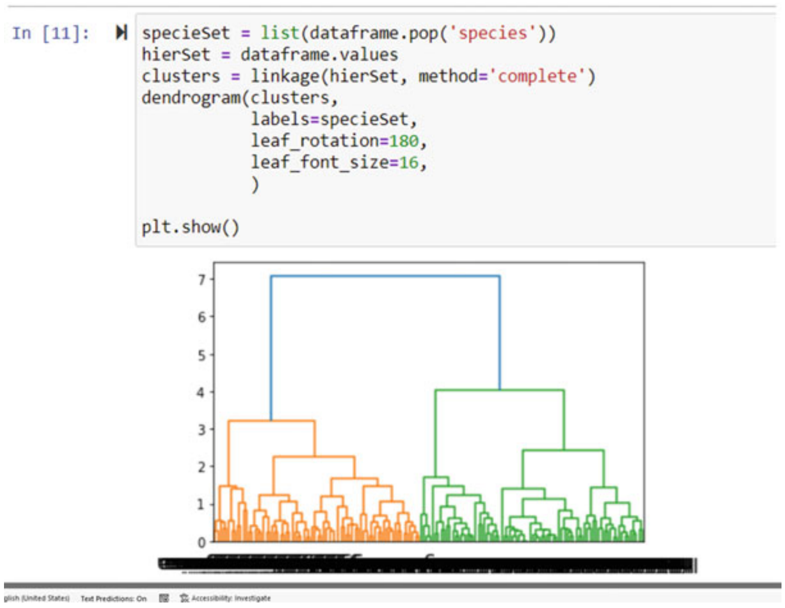
3.11.6.2 Running Hierarchical Algorithm and Plotting Data

After loading the data into a variable, the hierarchical algorithm is executed on the data to create clusters. As you know, there are three species in the iris dataset. However, the two closest species are merged to form one cluster *t*, and the model ends up with two clusters, as shown in Fig. 3.38.



```
In [9]:   from scipy.cluster.hierarchy import linkage, dendrogram
          import matplotlib.pyplot as plt
          import pandas as pd

In [10]:  dataframe = pd.read_csv("C:/datasets/IRIS.csv")
```

Fig. 3.37 Importing required packages and loading iris dataset



```
In [11]:  specieSet = list(dataframe.pop('species'))
          hierSet = dataframe.values
          clusters = linkage(hierSet, method='complete')
          dendrogram(clusters,
                     labels=specieSet,
                     leaf_rotation=180,
                     leaf_font_size=16,
                     )

          plt.show()
```

Fig. 3.38 Hierarchical clustering algorithm plot using IRIS dataset

3.11.7 Do It Yourself

This section is for students to apply what they learned in this chapter. Students need to solve the following problem in R and Python:

1. Load the stroke prediction dataset from this location: *Stroke Prediction Dataset | Kaggle*.
2. Preprocess data and apply any required mapping from string to numeric values.
3. Apply linear regression and KNN algorithms for SML, *k*-means and hierarchical clustering for UML.
4. Study the model's performance for SML using the cross-validation concept.
5. Visualize your data in three different plots or graphs for SML and UML algorithms.
6. Discuss your model and its performance with the class in terms of applying the same model to a new unseen dataset (generalization).

3.11.8 Do More Yourself

Below are a few datasets that you might use to do more exercises:

1. *Pima Indians Diabetes Database | Kaggle.*
2. *FIFA World Cup | Kaggle.*
3. *Flu Shot Prediction | Kaggle.*

References

1. E. Coiera, Computational reasoning methods, in *Guide to Health Informatics*, 3rd edn., (CRC Press, 2015) ch. 26
2. I.H. Witten, E. Frank, M.A. Hall, C.J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques* (Elsevier Science, 2016)
3. IBM, *CRISP-DM Help Overview*. https://www.ibm.com/docs/en/spss-modeler/SaaS?topic=dm-crisp-help-overview. Accessed
4. Smart Vision Europe, *What Is the CRISP-DM Methodology?* Smart Vision Europe. https://www.sv-europe.com/crisp-dm-methodology/. Accessed 27 April 2018
5. D.T. Larose, C.D. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining* (Wiley, 2014)
6. J.D. Kelleher, B.M. Namee, A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies* (MIT Press, 2015)
7. C. El Morr, H. Ali-Hassan, *Analytics in healthcare: A practical introduction* (Springer, 2019)
8. B. Marr, *What Is the Difference Between Artificial Intelligence and Machine Learning?* Forbes. Accessed 30 April 2018
9. R. Sharda, D. Delen, E. Turban, *Business Intelligence: A Managerial Perspective on Analytics* (Prentice Hall Press, 2014)
10. N. Kalé, N. Jones, *Practical Analytics* (Epistemy Press, 2015)

11. Z.H. Zhou, Introduction, in *Ensemble Methods: Foundations and Algorithms*, ((Chapman & Hall/CRC Machine Learning & Pattern Recognition Series: CRC Press), 2012)
12. J.A. Nichols, H.W.H. Chan, M.A. Baker, Machine learning: Applications of artificial intelligence to imaging and diagnosis. Biophys. Rev. **11**(1), 111–118 (2019)
13. V. Jha, Machine Learning Algorithm - Backbone of Emerging Technologies. https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/. Accessed April 30 2018
14. E. Alpaydin, *Introduction to Machine Learning* (MIT Press, 2014)
15. J. Fahl, *Data Analytics: A Practical Guide to Data Analytics for Business, Beginner to Expert* (CreateSpace Independent Publishing Platform, 2017)
16. M.L. Sylvia, M.F. Terhaar, *Clinical analytics and data management for the DNP*, 2nd edn. (Springer Publishing Company, 2018)
17. A.J. Myles, R.N. Feudale, Y. Liu, N.A. Woody, S.D. Brown, An introduction to decision tree modeling. J. Chemom. **18**(6), 275–285 (2004). https://doi.org/10.1002/cem.873
18. K.R. Detlev Ganten, W. Birchmeier, J.T. Epplen, K. Genser, M. Gossen, B. Kersten, H. Lehrach, H. Oschkinat, P. Ruiz, P. Schmieder, E. Wanker, C. Nolte, Decision Tree, in *Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine*, (Springer, Berlin, Heidelberg, 2006), pp. 380–380
19. C. Kingsford, S.L. Salzberg, What are decision trees? Nat. Biotechnol. **26**(9), 1011–1013 (1 Sep 2008). https://doi.org/10.1038/nbt0908-1011
20. Y.-Y. Song, Y. Lu, Decision tree methods: Applications for classification and prediction. Shanghai Arch. Psychiatry **27**(2), 130–135 (2015). https://doi.org/10.11919/j.issn.1002-0829.215044
21. B. Gupta, A. Rawat, A. Jain, A. Arora, N. Dhami, Analysis of various decision tree algorithms for classification in data mining. Int. J. Comput. Appl. **163**, 15–19 (04/17 2017). https://doi.org/10.5120/ijca2017913660
22. F. Ye et al., Chi-squared automatic interaction detection decision tree analysis of risk factors for infant anemia in Beijing, China. Chin. Med. J. **129**(10), 1193–1199 (2016)
23. C.-L. Lin, C.-L. Fan, Evaluation of CART, CHAID, and QUEST algorithms: A case study of construction defects in Taiwan. J. Asian Archit. Build. Eng. **18**(6), 539–553 (2019)
24. C. El Morr, *Introduction to Health Informatics: A Canadian Perspective* (Canadian Scholars' Press, 2018)
25. L. Breiman, Random forests. Mach. Learn. **45**(1), 5–32 (2001)
26. D. Denisko, M.M. Hoffman, Classification and interaction in random forests. Proc. Natl. Acad. Sci. U. S. A. **115**(8), 1690–1692 (2018). https://doi.org/10.1073/pnas.1800256115
27. D. Denisko, M.M. Hoffman, Classification and interaction in random forests. Proc. Natl. Acad. Sci. **115**(8), 1690–1692 (2018)
28. K. Fawagreh, M.M. Gaber, E. Elyan, Random forests: From early developments to recent advancements. Syst. Sci. Control Eng. **2**(1), 602–609 (1 Dec 2014). https://doi.org/10.1080/21642583.2014.956265
29. A.M. Deris, A.M. Zain, R. Sallehuddin, Overview of support vector machine in modeling machining performances. Procedia Eng. **24**(Complete), 308–312 (2011). https://doi.org/10.1016/j.proeng.2011.11.2647
30. C. Campbell, Y. Ying, Learning with support vector machines. Synth. Lect. Artif. Intell. Mach. Learn. **5**(1), 1–95 (2011)
31. S. Huang, N. Cai, P.P. Pacheco, S. Narrandes, Y. Wang, W. Xu, Applications of Support Vector Machine (SVM) learning in cancer genomics. Cancer Genomics Proteomics **15**(1), 41–51 (2018)
32. S. Premanand, The A-Z guide to support vector machine. https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/. Accessed 18 March 2022
33. D. Willimitis, The Kernel Trick in support vector classification. 12 Dec 2018 (2018). [Online]. Available: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

34. R. Sharda, D. Delen, E. Turban, J. Aronson, T.P. Liang, *Businesss Intelligence and Analytics: Systems for Decision Support* (Pearson Edition Limited, 2014)
35. M. Liao, Y. Li, F. Kianifard, E. Obi, S. Arcona, Cluster analysis and its application to healthcare claims data: A study of end-stage renal disease patients who initiated hemodialysis. BMC Nephrol. **17**, 25 (2016). https://doi.org/10.1186/s12882-016-0238-2
36. J.J. Armstrong, M. Zhu, J.P. Hirdes, P. Stolee, K-means cluster analysis of rehabilitation service users in the home health care system of Ontario: Examining the heterogeneity of a complex geriatric population. Arch. Phys. Med. Rehabil. **93**(12), 2198–2205 (2012)
37. J. MacGregor, *Predictive Analysis with SAP* (Galileo Press, Bonn, 2013)
38. R. Wang, AdaBoost for feature selection, classification and its relation with SVM, a review. Phys. Procedia **25**, 800–807 (2012)
39. S. Wu, H. Nagahashi, Analysis of generalization ability for different AdaBoost variants based on classification and regression trees. J. Electr. Comput. Eng. **2015**, 835357 (2015, Feburary 10). https://doi.org/10.1155/2015/835357
40. J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Stat. **28**(2), 337–407 (2000)
41. A. Vezhnevets, V. Vezhnevets, Modest AdaBoost-teaching AdaBoost to generalize better. Graphicon **12**(5), 987–997 (2005)
42. J. Feizabadi, Machine learning demand forecasting and supply chain performance. Int J Log Res Appl **25**, 1–24 (2020). https://doi.org/10.1080/13675567.2020.1803246
43. J. de la Torre, J. Marin, S. Ilarri, J.J. Marin, Applying machine learning for healthcare: A case study on cervical pain assessment with motion capture. Appl. Sci. **10**(17), 5942 (2020)
44. A. Petropoulos, V. Siakoulis, E. Stavroulakis, N.E. Vlachogiannakis, Predicting bank insolvencies using machine learning techniques. Int. J. Forecast. **36**(3), 1092–1113 (2020)
45. Z. Wang, A. Majewicz Fey, Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery. Int. J. Comput. Assist. Radiol. Surg. **13**(12), 1959–1970 (2018)

Chapter 4
Data Preprocessing

4.1 The Problem

Preprocessing is the practice of cleaning, altering, and reorganizing raw data prior to processing and analysis, which is also known as data preparation [1]. It is an important step before processing and usually entails reformatting, adjusting, and integrating datasets to improve the information contained within them. Even though data preprocessing can be an onerous task, it is necessary as a precondition for putting data into context and reducing the possibility of bias [2]. An Aberdeen Group study states that data preprocessing refers to any activity taken in order to improve the quality, usability, accessibility, and portability of data [3]. In a poll published in Forbes, data scientists reported that they spend 60% of their time on data preprocessing (Fig. 4.1).

The data preparation process often consists of standardizing data formats, enhancing data, and eliminating outliers. Data preparation consists of collecting, cleaning, and merging information into one file for analysis. When it comes to machine learning applications, proper data preparation is critical. However, machine learning techniques are also being used for automated data preparation purposes [5]. Correct data preparation prepares both the miner and the data in advance of the mining operation [6]. Preparing the data ensures that the model is constructed correctly. Preparing the miner entails creating the appropriate model. Data preparation leads to a better knowledge of the data, which in turn allows the proper model to be generated the first time and built correctly. Indeed, the preparation may bring the data miner to an understanding of the knowledge contained within the data, which may be sufficient in and of itself.

In the past, data preprocessing was considered a specialized skill. It is usually business managers who are most in need of the findings, insights, and intuitions contained inside recorded data. Twenty years ago, spreadsheets were thought of as accounting-specific tools with limited relevance to the rest of the company. In

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_4

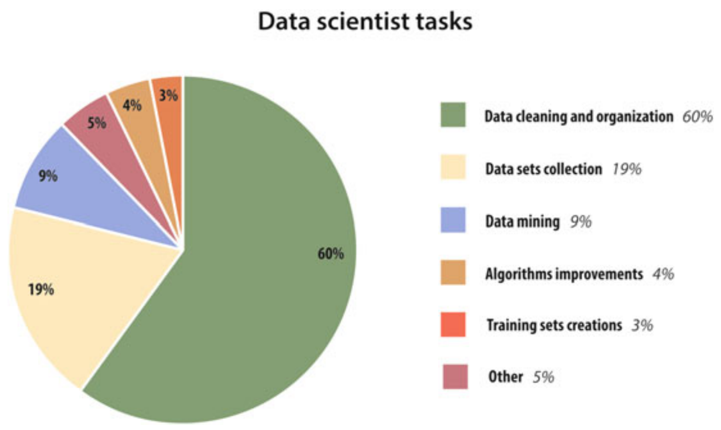


Fig. 4.1 Proportion of data scientists time spent by task [4]

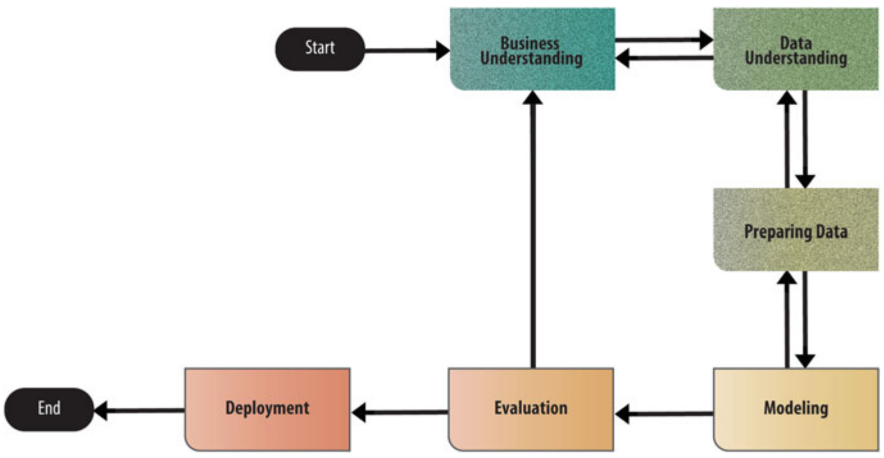


Fig. 4.2 Phases before and after the data prep phase [7]

today's corporate world, spreadsheets are viewed as an essential tool. Data preparation tools will soon be used by company managers in the same way that spreadsheets were used by accountants in the 1970s and 1980s. In the future, many critical business activities will be run by automated systems, which will be monitored, guided, and controlled by business managers and analysts via control panels. Structures of this type are already in use. In order to build and manage these systems, highly skilled data modelers and data preparations will be required [6] (Fig. 4.2).

4.2 Data Preprocessing Steps

Predictive modeling projects require the preparation of data. Some of these concepts may be better suited as sub-activities inside the overall data preparation process rather than as separate tasks. Data preparation may also be defined as the process of transforming raw data into a form that is more suitable for use in modeling applications [8].

What you should do here is very dependent on both your data and the objectives of your project, as well as the algorithms that will be utilized to represent your data. There is some difference in the data preparation stages data specialists and software providers recommend. These steps also fluctuate depending on the industry and company. Nonetheless, there are many procedures that you may utilize during data preparation and that we will investigate below [5, 8, 9].

4.2.1 Data Collection

Finding the appropriate data is the first step. Depending on the situation, this information can be obtained from a pre-existing data catalogue as well as operating systems, data warehouses, data lakes, and other data sources, or it can be added on the fly. A suitable fit for the objectives of the intended analytics applications should be confirmed during this stage by information professionals [5, 9].

4.2.2 Data Profiling, Discovery, and Access

It is necessary to examine the data that has been gathered in order to have a better understanding of what each dataset includes and what needs to be done in order to prepare it for the intended usage [9]. To complete this phase, you must first become familiar with the data and understand what needs to be done for the data to be relevant in a certain context. Data profiling can aid in this process by identifying correlations, inconsistencies, anomalies, missing values, allowing them to be addressed more effectively [5].

4.2.3 Data Cleansing and Validation

The data errors and issues that have been detected are addressed to produce complete and accurate datasets [5, 9]. During the cleansing process, for example, incorrect information is deleted or corrected, missing values are filled in, and inconsistencies between entries are reconciled. Cleaning up the data is generally the most onerous step. The data inaccuracies and problems that have been detected are rectified to produce full and accurate datasets. Among the most important responsibilities in this area are:

- Removing and/or fixing outliers
- Imputing (filling in) missing values
- Harmonizing inconsistent entries
- Hiding sensitive data entries

4.2.4 Data Structuring

At this phase, the data is modeled and arranged to match the criteria of the analytics system. For example, we might need to export data saved in a certain format into other formats before it can be accessed by business intelligence and analytics applications such as Tableau, Microsoft Power BI, and Apache [5].

4.2.5 Feature Selection

This is also known as variable selection, attribute selection, or variable subset selection. Relevant characteristics are identified and/or noisy data is filtered out. Finding the input variables that are most important to the project is what this approach is all about [8]. Improved prediction accuracy and comprehensibility are achieved with this method. A feature is considered irrelevant if it does not give any valuable information and redundant if it does not contribute any more information [10].

4.2.6 Data Transformation and Enrichment

To transform data means to change its format or value entries in order to achieve a predetermined result or to make the data more easily understandable to an even greater number of people, as defined by the data's intended audience. Examples of data transformation include the creation of features that aggregate values from previously existing features [9]. Meanwhile, enriching data refers to the process of integrating data with other related information in order to give deeper insights; it also refers to the process of further improving and optimizing datasets as needed, using techniques such as augmenting and adding data [5].

4.2.7 Data Validation, Storage, and Publishing

At the end of the process, automated algorithms are performed on the data to ensure that it is consistent, comprehensive, and accurate. It is then saved in a database, a data lake, or another repository and may either be utilized directly by the person who created it or is made available for other users to access and use as needed [5].

4.3 Feature Engineering

"Feature engineering" is a term often used in the data science field to refer to data preparation [11]. While the terms "data prep" and "feature engineering" are used interchangeably, as compared to the typical data prep method, feature engineering is more dependent on domain-specific expertise. Feature engineering is used to develop "features" for certain machine learning algorithms, whereas data prep is used to prepare data for distribution to many people. Data preparation and feature engineering are two of the most time-consuming and critical activities in data mining, and they are both essential. The ability to provide accurate results is enhanced by having data that has been properly prepared. Data preparation operations, on the other hand, tend to be repetitive, boring, and time-consuming. When it comes to data preparation for machine learning, feature engineering is a critical step. Building relevant features from existing features is a strategy that leads to increased predictive performance when done correctly. Feature engineering is the process of applying transformation functions to existing features, such as arithmetic and aggregation operators, to develop new ones. Transformations aid in the scaling of a feature or the conversion of a nonlinear relationship between a feature and a target class into a linear relationship that is easier to learn [12]. A feature vector is fed into a machine learning model such as a neural network, decision tree, random forest, or gradient boosting machine, which then predicts the outcome of the experiment. These models are trained in a supervised manner, which means that they are given a set of feature vectors with predicted output. It is a typical practice to create new features from the existing feature set that has been offered. Depending on how they are constructed, they will either supplement or completely replace elements of the present feature vector. Engineering features are essentially computed fields that are determined by the values of other features in a given feature set [13]. The notion that there are various ways to represent predictors in a model, and that some of these representations are superior to others, gives rise to the concept of feature engineering, which is the process of developing data representations that improve the effectiveness of a model (also known as predictive modeling) [14].

Features, also known as signals, are information encoded from raw data that allows machine learning algorithms to classify an unknown item or estimate an unknown value based on the information encoded. During the learning phase, a trained system determines the relative importance of this feature for future tasks by

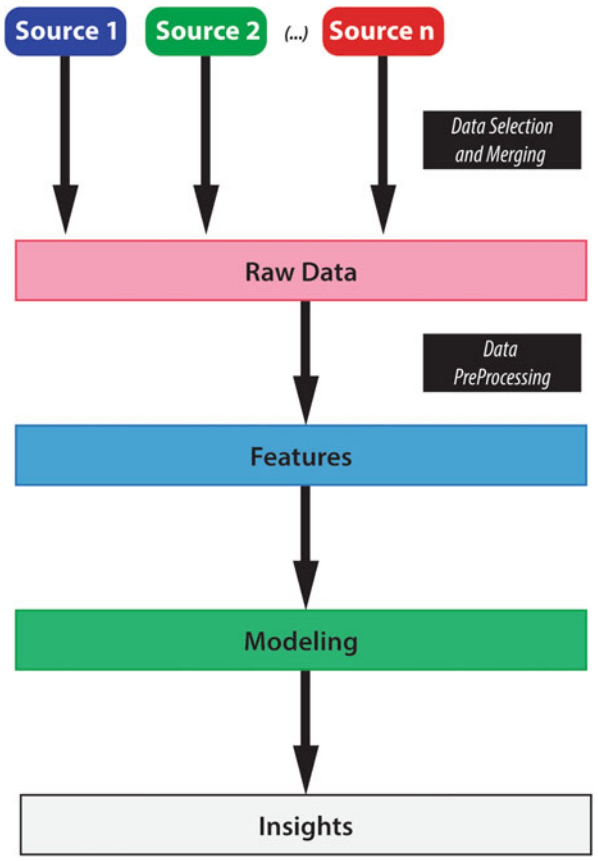


Fig. 4.3 Feature Engineering as a step toward data modeling

comparing it to other features. Features may appear to be inconsequential, as if they were only one of the many components that go into a large-scale machine-learning endeavor (along with raw data, cluster software, human-provided labels, and scalable statistical algorithms). They are, however, extremely exceptional, according to a few lessons from the expanding body of wisdom concerning trained systems [15] (Fig. 4.3).

In the science of feature engineering, there are three primary steps: feature creation, transformation, and feature extraction [16].

4.3.1 Feature Creation

Feature creation involves determining the most helpful variables to utilize in a prediction model. It is a process that needs human input and innovation. Addition, subtraction, and ratios are used to combine existing characteristics to produce the new ones, which offers a lot of versatility. For example, in a problem involving purchases, one could create a feature for the day of the week the purchases happen, and the time of the day (e.g., morning, afternoon). In a problem involving people's age, one can discretize a numeric salary into new features by binning the salary (e.g., "less than 40,000," "40,000–60,000"). In a problem involving a category related to mental health (e.g., "poor," "fair," "good," "excellent") one can create new features by binarizing the mental health (e.g., creating four new binary features called Poor, Fair, Good, and Excellent).

4.3.2 Transformation

The transformation process of feature engineering includes modifying a predictor variable somehow (e.g., using standardization) to improve the model's performance. For example, it guarantees that the model is able to accept diverse data; it makes certain that all the features are on an equal scale; and it increases the model's correctness and guarantees that all the characteristics are within the permissible range to prevent any errors.

4.3.3 Feature Extraction

Transformations entail producing a new variable by modifying one variable in some manner or another. Feature extraction includes constructing variables by extracting them from some other data. The major purpose of this stage is to minimize the size of the data so that it can be conveniently utilized and handled for data modeling. Feature extraction approaches include procedures such as principal components analysis (PCA).

4.4 Feature Engineering Techniques

You may utilize these best practices in feature engineering. Some of the techniques in this list may be more effective when used with specific algorithms or datasets, while others may be applicable in any case [17].

4.4.1 Imputation

Missing values are a common problem when preparing data for machine learning. Human error, disruptions in data flow, privacy issues, and other circumstances can all result in missing values in a model's output. Machine learning models are affected by missing values, no matter the reason. An imputation strategy is designed to deal with missing values. Imputations can be categorized into two distinct types: numerical and categorical.

4.4.1.1 Numerical Imputation

We normally utilize data from completed surveys or censuses to figure out what numbers should be allocated to persons who are currently in the population. Among the information contained in these datasets is information on how many people consume different types of food, whether they reside in a city or postcode with good infrastructure, and how much tax they pay on a yearly basis. The reason for this is that numerical imputation is used to fill gaps in data when certain pieces of information are lacking [17].

4.4.1.2 Categorical Imputation

When working with categorical values, it can be good to use the column's greatest value as a replacement for any missing data. The imputation will be more likely to converge to a random selection if you feel the values in the column are widely distributed and there is no dominant value. Imputing a category like "Other" in this case would be better [18].

4.4.2 Discretizing Numerical Features

Decision trees and rule-based algorithms perform better when using discrete features (e.g., categorical), as they might have to sort the values repeatedly which becomes time-consuming with continuous values (i.e., real numbers). Some other algorithms like Naïve Bayes cannot handle well continuous data. Hence the need for discretization [19].

Discretization is the process of placing the values into containers, bins, or buckets. The result is a limited number of containers that are treated as discrete values.

4.4.3 Converting Categorical Discrete Features to Numeric (Binarization)

Some algorithms, like KNN and regression algorithms, cannot work with strings (categorical values) and instead need numeric values. We can convert a categorical feature to many numeric features by assigning for each of its categorical values a new binary numeric feature that can hold a value of 0 or 1. For example, in a student dataset, a categorical feature “major,” the values of which are “engineering,” “health informatics,” and “information technology,” can be discretized by creating three new numeric binary features: “engineering,” “health informatics,” and “information technology.” Depending on its “major” value, each instance in the dataset can have a value of 1 in one of these new features and a 0 in the others. A student with a value of “engineering” in the “major” column would have 1 in the new feature “engineering” and 0 in “health informatics” and “information technology.” This strategy is better than converting the major into one numeric feature in which each major category is replaced with a number; for example, adding a new feature major_numeric that equals 1 for health informatics, 2 for information technology, 3 for business, and 4 for engineering is not advisable because the algorithm will suppose that there is an order among the values of (i.e., the majors they represent), which is not the case, and it will try to use that order in the prediction, which is misleading and will confuse the algorithm.

4.4.4 Log Transformation

One of the most often utilized mathematical transformations in feature engineering is the logarithm transformation (also known as the log transform) [17]. It is usually used to make a normal or less-skewed distribution out of a skewed one. When we do this conversion, we use the log of the column’s values. As a result of this method’s adoption, data that was before considered difficult to interpret is now easier to interpret [18].

4.4.5 One-Hot Encoding

It is possible to encode elements of finite sets by giving a code for each possible value. A feature that can have n values will be transformed into n features. Each feature may have two values 0 or 1. For example, if we have a feature called Specialization with the following possible values: “Informatics,” “IT,” and “Business”; one-hot encoding will create three new features informatics, IT, and business; if an instance has Specialization- “informatics” then there will a value of 1 in the

Informatics features and two values of 0 in the IT and Business features, and so on and so forth [18].

4.4.6 Scaling

Numerical features' scale might differ in range; for example, Age and Income features will not have the same range and the large values of Income might affect model training. Scaling overcomes this problem. The numeric features become equivalent in terms of the range after scaling. Scaling can be accomplished in one of two ways [17, 18]:

4.4.6.1 Normalization (Min-Max Normalization)

Normalization scales all values in a range between 0 and 1 using normalization. This modification does not affect the distribution of the feature, although due to the lower standard deviations, it does amplify the impact of outliers. As a result, it is recommended to deal with the outliers before normalization. It can be mathematically defined as follows:

$$X_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Data normalization [20, 21] is concerned with rescaling of the data of a numeric feature to a small range, such as [0, 1]. Normalization is needed whenever we have large differences in the scales of numeric values for different features. For example, if your dataset contains a numeric feature with values ranging from 10 to 20 and another one with values ranging from 1000 to 1000,000, and we are trying to use both features to train our model, the difference in scales between the features might influence the model, as variables with variance in their data are more likely to contribute more to the model and distort the impact of the features with narrow ranges.

Feature selection (using principal component analysis, for example) is another reason to normalize the data. Often, feature selection involves algorithms that rely on the variance value; hence, the need for normalization (i.e., reducing the scales of all features to [0, 1]) to make the features appropriately comparable. Finally, regularization can be affected by the data values, and normalization will reduce such distortion.

Normalization allows us to avoid such problems by rescaling the values in each of the two features to the common scale [0, 1] while maintaining the general distribution of data within each feature. It is also possible to scale data to a range other than to 1, such as -1 to 1.

Normalization can help reduce the learning time; for example, during gradient descent (see chapter on linear regression), when we update the parameters of the model, we compute the derivatives of the error function with respect to the parameters (e.g., w1, w2), so if a feature has large values, it will dominate the updates [22]. Since normalization depends on minimum and maximum values it is sensitive to outliers, unlike standardization.

4.4.6.2 Standardization (*Z*-Score Normalization)

When we have data stored as a feature in a dataset, we like to compare it to draw meaningful knowledge. For example, students' grades could be compared between different sessions of the same course, different courses, or universities; likewise, the number of patients who were readmitted to a hospital after discharge can be compared across hospitals, etc.

However important these comparisons are, they might not be meaningful unless we understand their contexts (e.g., the units/scale they were measured with). For instance, comparing grades is not appropriate if the grades were not measured on the same scale; we cannot compare a grade of 15 for a student in course A with a grade of 60 for a student in course B unless we know in advance that both grades were computed using the same scale (e.g., 0 to 100). If the grade of 15 was computed using a scale from 0 to 20, then we need to scale both grades to the same interval to obtain meaningful information. For instance, we can multiply 15 by 5 to obtain the equivalent grade over 100 (i.e., 75), and we can then deduce that in fact the grade for the student in course A is higher than the grade for the student in course B. The process of transforming data to the same scale for comparison is called standardization [20].

Another way to standardize data is by computing their *Z*-score or standard score. The standard score lets us know how far from the mean a score is; more precisely, it tells how many standard deviations below or above the mean a score is. For a data score x_i in a dataset that has an average $\bar{X}$ and a standard deviation σ, *Z*-score = $(x_i\text{-}\bar{X})/\sigma$. For the example above, suppose that the grade of 15 was in a dataset where the average is 12 and the standard deviation is 2; then the *Z*-score corresponding to 15 is 1.5 (i.e., the student's grade is 1.5 standard deviations above the average of class A); if we suppose that the grade of 60 was in a dataset of average 50 and standard deviation 10, then the Z-score corresponding to 15 is 1 (i.e., the student's grade is 1 standard deviation above the average of class B); obviously, the grade of 15 is a higher score than 60. This example is simple and straightforward; other examples might be much more complicated to guess: is the readmission rate at hospital *X* more substantial than that at hospital *Y*?

Standardization is the process of scaling values while accounting for the standard deviation. It is necessary to divide the variance of the distribution by the number of data points to arrive at a distribution with a 0 mean and 1 variance. It can be mathematically defined as follows [18]:

$$Z - \text{score} = \frac{(x - \bar{X})}{\sigma}$$

The mean is denoted by the symbol μ, whereas the standard deviation is denoted by the symbol σ. When we standardize all features in a dataset, they will all have 0 as the mean and 1 as the variance; each data point's value describes how many standard deviations away from the mean it is.

However, if the characteristics' standard deviations vary, the range of those features will likewise differ [17]. Because of this, outliers in the data have more of an impact. To account for the presence of outliers, one might remove outliers before standardization or apply standardization techniques that account for outliers, such as robust scaling.

Naïve Bayes (probabilistic) and decision trees look at features independently and hence are not sensitive to difference in feature scales. While algorithms such as support vector machines, *K*-Nearest Neighbor (KNN), *K*-means, and *K*-Means rely on a form of distance and hence are sensitive to difference in features. Also, algorithms (e.g., linear regression, logistics regression) that use stochastic gradient descent for optimization (i.e., hyperparameter tuning) will benefit from standardization. Artificial Neural Networks (ANNs) often rely on values between 0 and 1, so they would benefit from normalization instead of standardization.

4.4.7 Reduce the Features Dimensionality

Reducing the feature dimensionality (how many features we originally have) can be done through the Principal Component Analysis (PCA). PCA is a technique that projects the features into a smaller dimension (less than the original), it creates new features and let you know how much each feature explains of the variance in the data (i.e., how much each feature is important to model your dataset); hence you can choose the most important features that explain together the most of your dataset and obtain a smaller space (i.e., less features than what we had originally). If we choose the first 4 components in Fig. 4.4, we will have a feature space of 4 (4 new features) instead of the original 8. The first 4 components explain 80% of variance in the data and hence represents it well; however, there is no guarantee that algorithms will perform better after dimensionality reduction.

4.5 Overfitting

Overfitting occurs when the model performs well in the training dataset but does not perform well in the testing dataset. Overfitting might occur because of noise in the dataset, the small size of the training set, or the complexity of the machine learning model [23]. When it comes to the integrity and trustworthiness of a statistical model,

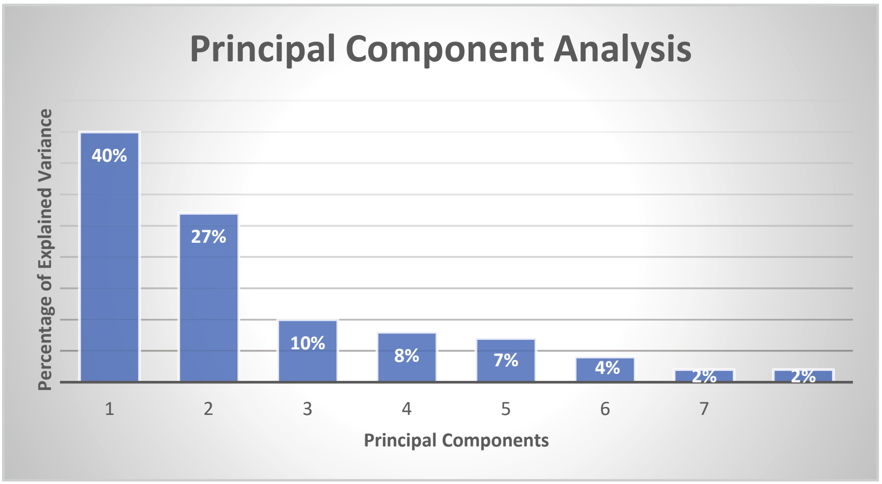


Fig. 4.4 Principal components with their corresponding percentage of explained variance

overfitting is a major concern [24]. More so, the notion of parsimony, often known as Occam's razor, urges the use of models and procedures that contain only the information required for the modeling process and nothing else [25]. For example, if a regression model with two predictors is sufficient to describe *y*, then no more predictions should be utilized than these two. To take things even further, if the relationship can be captured by a linear function of these two predictors (which is described by three numbers—the intercept and two slopes—then utilizing a quadratic function violates the principle of parsimony).

Overfitting is the employment of models or procedures that violate the principle of parsimony; that is, models or methods that include more terms than are essential or use more intricate approaches than are required. Overfitting can be minimized using strategies like preselecting components, centering, and cross-validation, but the only definite approach to address the issue is to replicate findings in another dataset [24]. Overfitting a model often takes the form of developing an excessively complicated model to explain quirks in the data under investigation. In fact, the data that is analyzed often has a certain degree of error or random noise. For example, striving to make the model adhere too closely to somewhat erroneous data might cause the model to get infected with significant mistakes, reducing its predictive value significantly [26]. When overfitting occurs, they reflect low bias and high variance [27]. Overfitting is more likely to occur with nonparametric and nonlinear models, which have greater flexibility while learning a target function than linear models. As a result, many nonparametric machine learning methods incorporate parameters or approaches that restrict and constrain the amount of detail that the model learns as it progresses [28]. Figure 4.5 shows an example of overfitting while Fig. 4.6 compares overfitting with underfitting, the subject of our next paragraph.

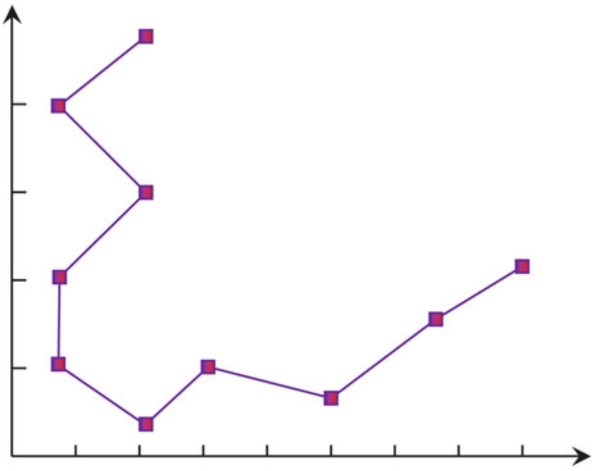


Fig. 4.5 A graphical illustration of overfitting

4.6 Underfitting

Underfitting occurs when a data model fails to correctly represent the relationship between input features and target. This would result in high error rates during both training and testing phases. Underfitting is the opposite of overfitting; it is a situation in which a model is incapable of capturing the variability of the data [29].

Underfitting risks oversimplifying a model; it occurs when an algorithm lacks sufficient training time or features or needs less regularization. When a model is underfitted, it is unable to create a dominating trend within the data, resulting in increased training mistakes and poor performance of the model, like overfitting. It is not possible to use a model for classification or prediction tasks if it does not generalize effectively to fresh data. The ability to generalize a model to new data is, in the end, what allows us to utilize machine learning algorithms daily to make predictions and categorize information. To determine the fundamental reason for low model accuracy, it is necessary to understand how models are fitted. This understanding will assist you in taking the necessary remedial action. Underfitting can be spotted if the prediction error on the training and the evaluation data are high. Underfitting typically occurs when we do not have enough data to develop an appropriate model, or when we use a simple model (e.g., linear) on data where the underlying relationship between the features and the target is non-linear. Since the rules of such machine learning model are simple the model is likely to generate many incorrect predictions (Fig. 4.7). To avoid underfitting, it is necessary to collect additional data, to reduce the number of features by employing feature selection, or to use more complex models.

In a nutshell, overfitting is characterized by high variance and low bias, whereas underfitting is characterized by high bias and low variance (Figs. 4.8 and 4.9).

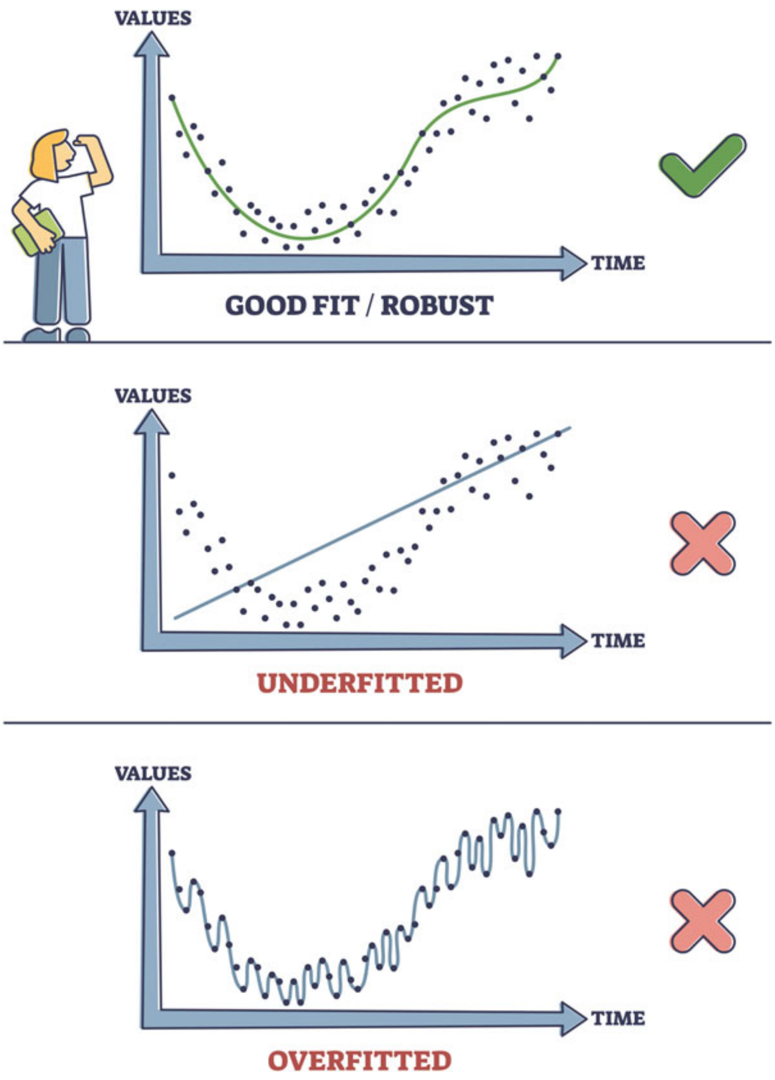
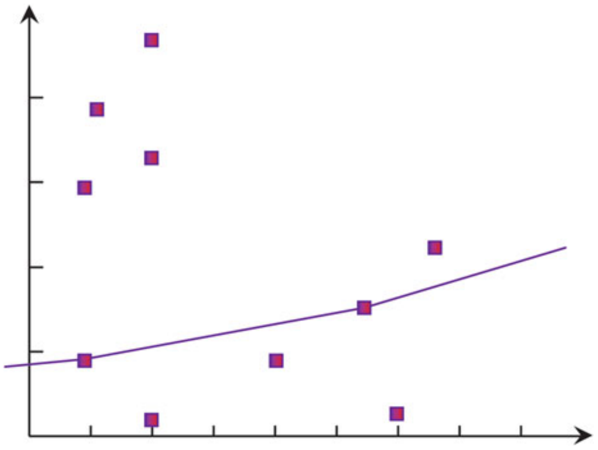


Fig. 4.6 Overfitting compared to underfitting and a good/robust fit

4.7 Model Selection: Selecting the Best Performing Model of an Algorithm

There are several algorithms for ML. In a project, we might choose one or more of these algorithms. When an algorithm is chosen, we use it to develop a model of the dataset we have. There are three steps involved in developing an ML model:

Fig. 4.7 A graphical illustration of underfitting



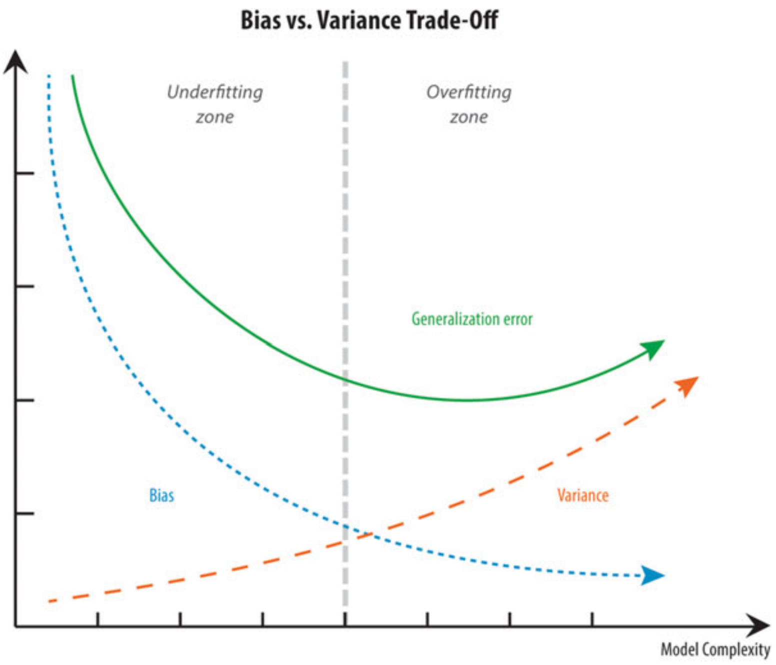


Fig. 4.8 A graphical illustration of overfitting and underfitting in terms of the bias vs. variance tradeoff

1. Training (i.e., fitting/developing) the model
2. Performing a **model selection**, i.e., finetuning the model's hyperparameters to find the best performing model with a good balance between bias and variance (i.e., between underfitting and overfitting). For each set of chosen hyperparameters' values, we train a model on an evaluation dataset and measure

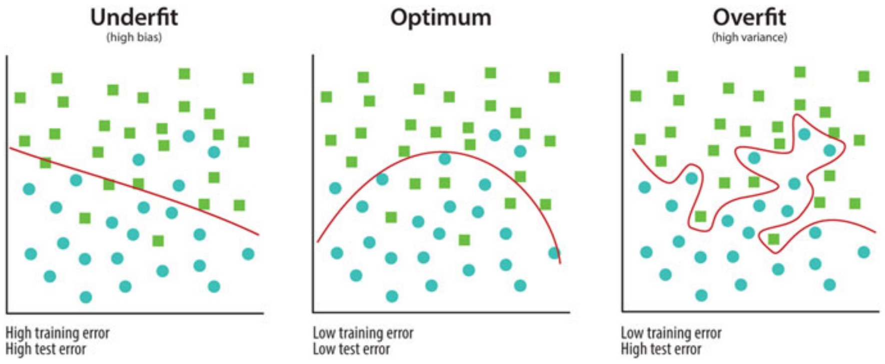


Fig. 4.9 Training and testing errors in the cases of underfitting, overfitting, and an optimum fit

its performance. Finally, we *select* the best-performing model. The performance of the selected model would be a good estimate of its generalizability.

3. Testing the performance of the model on an *unseen* dataset (i.e., the testing dataset) and reporting it.

4.7.1 Model Selection Using the Holdout Method

In its simplest form, the holdout method consists of holding out one dataset for testing. We split the dataset into training and testing datasets; we train the model on the training dataset and evaluate its generalizability by measuring its performance on the testing dataset.

We can repeat the process several times, each time we change the hyperparameters values and test the new model on the testing dataset until we find a model that is best performing on the testing dataset. However, this process risks to overfit the model to the testing dataset; the obtained performance would not represent the generalizability of the model.

A better way consists of holding out two subsets, one for testing and one for validation. We use the validation dataset to perform model selection: train different models with different hyperparameters values, measure their performances, and finally choose the best performing one (Fig. 4.10).

However, most of the time we do not have a large dataset and we can only split the dataset into training and testing dataset. Also, with a validation dataset, the performance of the model is based on one dataset and is dependent on how the split was done on the initial dataset: different split may lead to a substantially different performance. Cross-validation solves both holdout method's limitations.

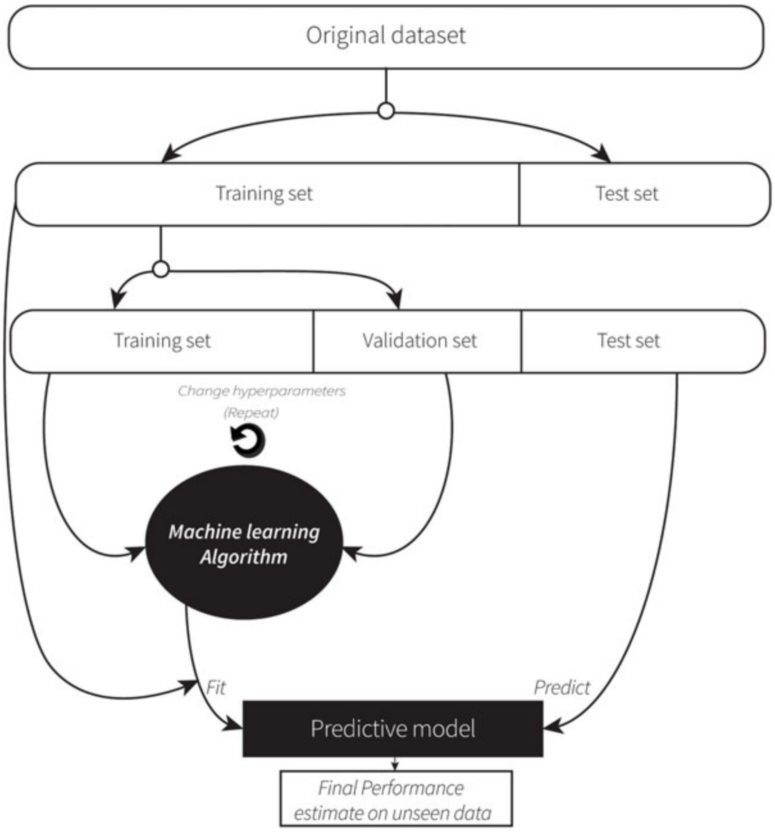


Fig. 4.10 Model selection by hyperparameter tuning

4.7.2 Model Selection Using Cross-Validation

Cross-validation can be performed in several ways; we will cover the *K*-fold cross-validation, the stratified *K*-fold cross-validation, and the Leave-one-out cross-validation.

In *K*-fold cross-validation, we split the *training* dataset into *K* equal-sized parts or folds. We keep *k*-1 folds for training the model and one for evaluating it. The process is repeated for each of the *K* parts. Each time the cross-validation algorithm saves the performance measures of the trained model. In the end, the cross-validation algorithm displays the average and the standard deviation of all performances. That average would be a good estimate of the generalizability of the model. Research has shown that $K = 10$ yields the best estimates.

Hence, for the same algorithm (e.g., logistic regression, decision tree), we can use cross-validation to estimate the performance of different models trained using different hyperparameters. In the end, cross-validation can deduce which set of

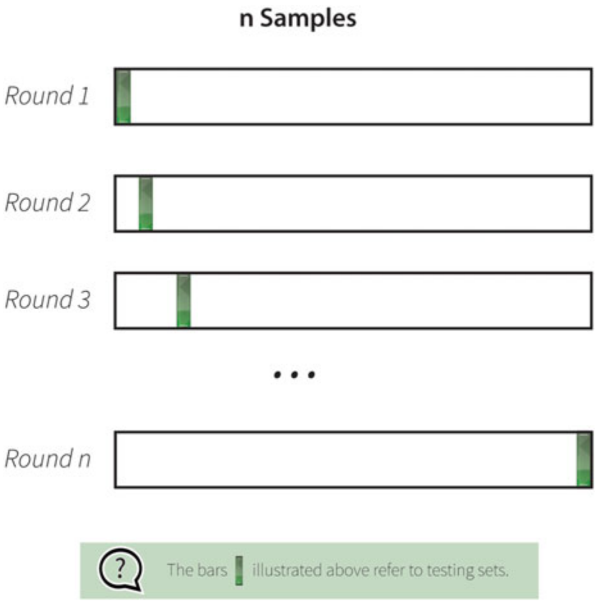


Fig. 4.11 *K*-fold cross-validation

hyperparameters' values provided the best performing model. Cross-validation provides us with the best hyperparameters' values, so that we use them to train the model on the whole training dataset and test it on the testing dataset to get a measure of its performance on unseen data (Fig. 4.11).

A variation of the *K*-fold cross-validation is a stratified *K*-fold cross-validation that guarantees that each fold maintains the same class proportion that exists in the training dataset. It is particularly useful in the case of imbalanced classes.

The Leave-one-out cross-validation (LOOCV) algorithm is similar to the *K*-fold cross-validation but each fold will hold out one instance (instead of a group of instances) of the training dataset; you can think of it as a *K*-fold cross-validation where *K* is equal to the number of training instances. Hence, it is much more onerous in terms of time and computing power than the *K*-fold cross-validation.

Finally, there is a nested cross-validation approach that we recommend exploring it yourself. It consists of a cross-validation within a cross-validation. In Python, it involves using a cross-val method and providing it with a grid (or randomized) search cross-validation as a parameter instead of providing it with an ML algorithm.

4.7.3 Evaluating Model Performance in Python

In Python, the *K*-fold cross-validation (i.e., cross_val_score) uses a stratified *k*-fold strategy by default for binary and multiclass classification problems. We can use two strategies to conduct model selection through hyperparameter tuning: Grid Search

(GridSearchCV in Python) and Randomized Search (RandomizedSearchCV in python).

Grid search needs a list of hyperparameters and their corresponding values. It uses cross-validation to try all the possible combinations of hyperparameters' values and find the set of hyperparameter values with the optimal performance. Grid search can display the best-performing hyperparameters and the best cores for the optimal model. Note that after obtaining the optimal model you need to train the model on the full testing dataset and then test it using the testing dataset.

Instead of going through the provided list of values exhaustively, the randomized search algorithm randomly chooses hyperparameter values from the list. Hence, its performance is much better than the grid search algorithm. The hyperparameter values obtained by randomized search cross-validation might not be the optimal ones; however, the model found performs as well as the optimal model obtained through grid search cross-validation.

4.8 Data Quality

Data is a valuable resource in all businesses, and its quality is crucial, as high-quality data make it easier for managers and operational processes to detect and resolve performance issues that might arise. Furthermore, high-quality data might improve an organization's chances of providing top-notch services to its customers. However, it is necessary to recognize numerous areas of data quality, ranging from definition to dimensions to kinds to strategies and procedures, in order to equip methodologies and processes for data improvement [30]. The study of data quality began in earnest in the 1990s in Europe, when a slew of scientists presented a variety of definitions of data quality as well as several division techniques for quality dimensions. Data quality can be defined as "suitability for usage" [31].

The term "data quality" already refers to a multidimensional notion that is difficult to quantify with specific definitions, especially when dealing with well-structured data. There are several conceptions of quality, each of which should be applied to certain forms of big data, and which should be carefully addressed when working with large datasets and doing analytics on them [32]. The traditional way of describing or characterizing data quality is to use several features or factors that help to rank the data delivered to users (e.g., amount of time it has been there, its accuracy, its completeness) or data processes (e.g., response time, reliability, and security). Quality factors include accuracy, completeness, currency, and consistency [33, 34].

On the other hand, by definition, the concept of data freshness brings up the question of how old the data is: Do you think it is up to date in terms of what users expect? Which data source contains the most up-to-date information? Is the information that has been extracted outdated? When was the information gathered and compiled? However, there is no consensus on what constitutes "fresh data" in the scientific community. In order to account for the different types of data integration systems, several definitions of freshness have been proposed [34]. Data freshness has been highlighted as one of the most significant qualities of data quality for data consumers, yet it is one of the most difficult to measure [31, 35]. High-quality data are required for the analysis and use of big data, as well as for ensuring that the data's value is not diminished. As of the time of this writing, a detailed study of quality standards and quality evaluation methodologies for big data is being carried out [36].

4.9 Key Terms

1. Imputation
2. Discretization
3. Log transformation
4. One-hot encoding
5. Normalization
6. Standardization
7. Feature creation
8. Feature extraction
9. Underfitting
10. Overfitting

4.10 Test Your Understanding

1. Why should we check for missing values in a dataset?
2. Are missing values always a problem?
3. Should we always delete missing values? Why or why not?
4. Explain discretization by giving a practical example.
5. Explain normalization.
6. Mention two reasons in support of using normalization.
7. Using an example, demonstrate the need for using standardization in some situations.
8. Do we always need to remove outliers? Why or why not?

4.11 Read More

1. Deshmukh, D. H., Ghorpade, T., & Padiya, P. (2015, 15–17 Jan. 2015). Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset. 2015 International Conference on Communication, Information & Computing Technology (ICCICT),
2. Moreira, L., Dantas, C., Oliveira, L., Soares, J., & Ogasawara, E. (2018, 8–13 July 2018). On Evaluating Data Preprocessing Methods for Machine

Learning Models for Flight Delays. 2018 International Joint Conference on Neural Networks (IJCNN),
3. Outrata, J. (2010, 12–14 Dec. 2010). Boolean Factor Analysis for Data Preprocessing in Machine Learning. 2010 Ninth International Conference on Machine Learning and Applications,
4. Esnaola, L., Tessore, J. P., Ramón, H., & Russo, C. (2019, 30 Sept.-4 Oct. 2019). Effectiveness of preprocessing techniques over social media texts for the improvement of machine learning based classifiers. 2019 XLV Latin American Computing Conference (CLEI),
5. Long, J., Wang, X., Zhou, W., Zhang, J., Dai, D., & Zhu, G. (2021). A Comprehensive Review of Signal Processing and Machine Learning Technologies for UHF PD Detection and Diagnosis (I): Preprocessing and Localization Approaches. IEEE Access, 9, 69,876–69,904. https://doi.org/10.1109/ACCESS.2021.3077483
6. Mannan, A., Javed, K., & Noon, S. K. (2020, 5–7 Nov. 2020). Statistical Boosting: A Preprocessing Technique to Enhance Performance of Machine Learning and Deep Learning Models on Partially Occluded Traffic Signs. 2020 IEEE 23rd International Multitopic Conference (INMIC),
7. Nagashima, Y., Araki, K., & Tochinai, K. (2001, 7–10 Oct. 2001). Evaluation of generality of inductive learning for preprocessing in machine translation. 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236),
8. Sharma, P., Hans, P., & Gupta, S. C. (2020, 29–31 Jan. 2020). Classification Of Plant Leaf Diseases Using Machine Learning And Image Preprocessing Techniques. 2020 tenth International Conference on Cloud Computing, Data Science & Engineering (Confluence),
9. Wu, Y. C., Chow, C. W., Liu, Y., Lin, Y. S., Hong, C. Y., Lin, D. C., Song, S. H., & Yeh, C. H. (2020). Received-Signal-Strength (RSS) Based 3D Visible-Light-Positioning (VLP) System Using Kernel Ridge Regression Machine Learning Algorithm With Sigmoid Function Data Preprocessing Method. IEEE Access, 8, 214,269–214,281. https://doi.org/10.1109/ACCESS.2020.3041192
10. Zelaya, C. V. G. (2019, 8–11 April 2019). Towards Explaining the Effects of Data Preprocessing on Machine Learning. 2019 IEEE 35th International Conference on Data Engineering (ICDE),

4.12 Lab

4.12.1 Working Example in Python

Download the California housing market dataset from the following: https://www.kaggle.com/datasets/camnugent/california-housing-prices

The features of this dataset are described as follows:

1. longitude: a higher value is farther west
2. latitude: a higher value is farther north
3. housingMedianAge: the median age of a house within a block
4. totalRooms: the total number of rooms within a block
5. totalBedrooms: the total number of bedrooms within a block
6. population: the total number of people residing within a block
7. households: the total number of households, a home unit, within a block
8. medianIncome: the median income for households within a block of houses (in tens of thousands of US Dollars)
9. medianHouseValue: the median house value within a block (measured in US Dollars)
10. oceanProximity: the location of the house with respect to the ocean

4.12.1.1 Read the Dataset

Import few libraries and read the file (Fig. 4.12).

You can display some information about the dataset by typing df.info() (Fig. 4.13).

We can already notice that total_bedrooms have missing values (i.e., have less non-null values than the other features).

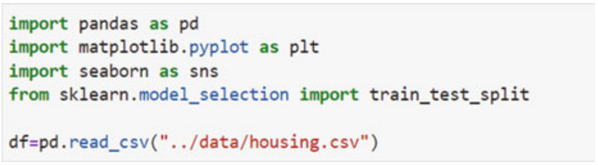


Fig. 4.12 reading the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

df=pd.read_csv("../data/housing.csv")
```

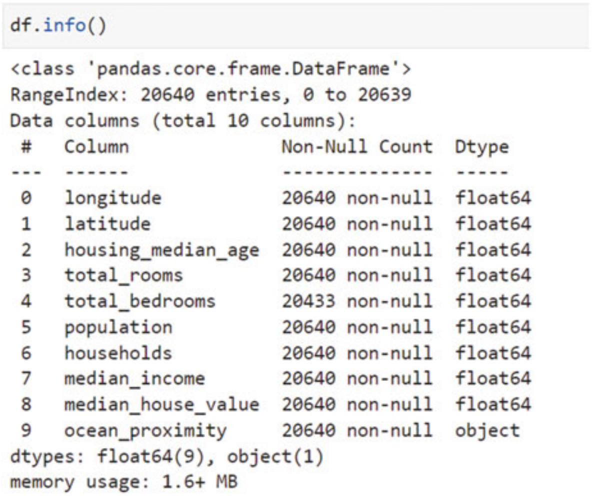
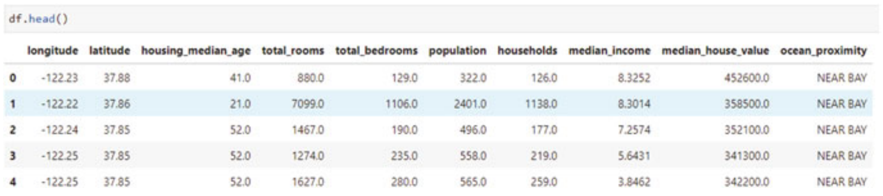


Fig. 4.13 information about the dataset displayed by python

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```



```
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Fig. 4.14 Information displayed by the data frame .head() method

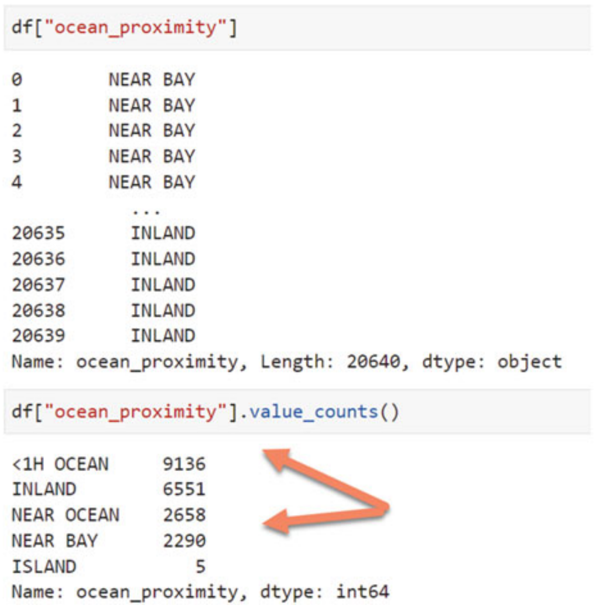


Fig. 4.15 Five distinct values are stored in feature ocean_proximity

```
df["ocean_proximity"]

0          NEAR BAY
1          NEAR BAY
2          NEAR BAY
3          NEAR BAY
4          NEAR BAY
             ...
20635       INLAND
20636       INLAND
20637       INLAND
20638       INLAND
20639       INLAND
Name: ocean_proximity, Length: 20640, dtype: object

df["ocean_proximity"].value_counts()

<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

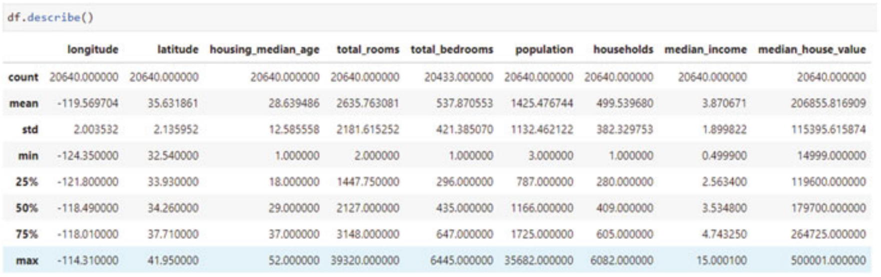
df.head() displays another kind of information. Try it. Only the first 5 rows of data care displayed by default; however, you can specify another number, if you wish to, by passing a value to in the .head() (e.g., df.head(10)) (Fig. 4.14).

We can notice in Fig. 4.13 that all variables are float but ocean proximity which is displayed as of type "object." Figure 4.14 displays values for ocean_proximity and we can notice that the features hold String values.

We can display the different stored values (Fig. 4.15). There are five distinct values are stored in feature ocean_proximity. This is a categorical feature that can benefit from One-Hot-Encoding.

Let us explore the data frame df more, type df.describe() (figure). Python will display few statistics for the numeric data only (Fig. 4.16).

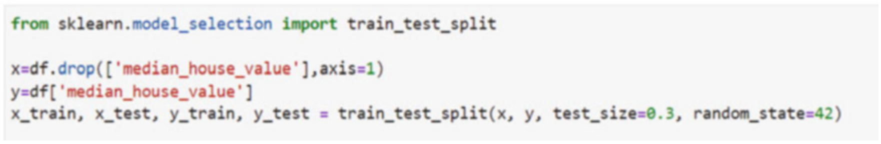
We can notice that these numeric variables are measured on different scales, they would benefit from standardization.



```
df.describe()
```

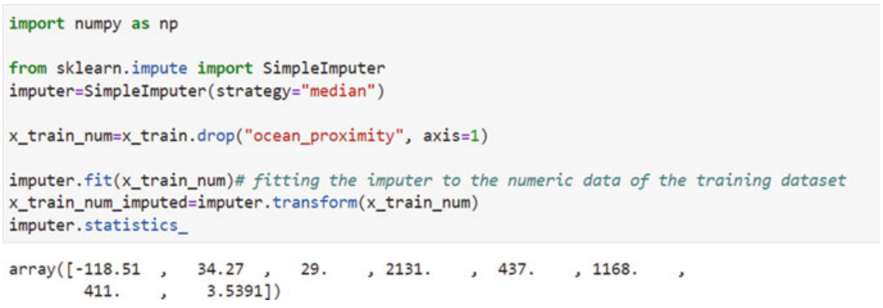
```
          longitude      latitude  housing_median_age   total_rooms  total_bedrooms    population    households  median_income  median_house_value
count  20640.000000  20640.000000        20640.000000  20640.000000    20433.000000  20640.000000  20640.000000   20640.000000        20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081      537.870553   1425.476744    499.539680       3.870671       206855.816909
std        2.003532      2.135952           12.585558   2181.615252      421.385070   1132.462122    382.329753       1.899822       115395.615874
min     -124.350000     32.540000            1.000000      2.000000        1.000000      3.000000      1.000000       0.499900        14999.000000
25%     -121.800000     33.930000           18.000000   1447.750000      296.000000    787.000000    280.000000       2.563400       119600.000000
50%     -118.490000     34.260000           29.000000   2127.000000      435.000000   1166.000000    409.000000       3.534800       179700.000000
75%     -118.010000     37.710000           37.000000   3148.000000      647.000000   1725.000000    605.000000       4.743250       264725.000000
max     -114.310000     41.950000           52.000000  39320.000000     6445.000000  35682.000000   6082.000000      15.000100       500001.000000
```

Fig. 4.16 Statistics for the numeric data



```
from sklearn.model_selection import train_test_split

x=df.drop(['median_house_value'],axis=1)
y=df['median_house_value']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

Fig. 4.17 Data frame splitting



```
import numpy as np

from sklearn.impute import SimpleImputer
imputer=SimpleImputer(strategy="median")

x_train_num=x_train.drop("ocean_proximity", axis=1)

imputer.fit(x_train_num)# fitting the imputer to the numeric data of the training dataset
x_train_num_imputed=imputer.transform(x_train_num)
imputer.statistics_
```

```
array([-118.51  ,   34.27  ,   29.    , 2131.    ,  437.    , 1168.    ,
        411.    ,    3.5391])
```

Fig. 4.18 Imputing data in

4.12.1.2 Split the Dataset

Our aim is to predict the median house value. Let us start by splitting the df data frame into training and testing datasets (Fig. 4.17). x_train, y_train will contain the training features and target/outcome and x_test and y_train the training features and target/outcome. The split was 70% for training and 40% for testing. The random_state can be assigned any number and it will serve to make the same split every time we run the program.

4.12.1.3 Impute Data

Let us start with imputation; to impute the missing values follow the instructions in Fig. 4.18. We are imputing the missing values in total_bedrooms by replacing the missing values with the median value for bedrooms (i.e., strategy="median'"). In the following, we would apply the imputation to all numeric features regardless if they had or not missing values. But first we would copy the numeric values in a special data frame x_train_num (you can call it anything) after dropping the string feature ocean_proximity.

We can have other strategies in other problems, such as the mode or the mean. Axis = 1 informs python that we are targeting a feature (column in the frame), axis = 0 is reserved for rows.

The *.fit ()* method of the imputer computes the median for each feature. While imputer_statistics displays the median computed for each numeric feature (e.g., − 118.51, 34.26).

Now if you type x_train_num_imputed = imputer.transform(x_train_num) then you will be making imputation on the x_train_num and python returns the result to x_train_num_imputed. The *.transform()* method applies the operation on the data.

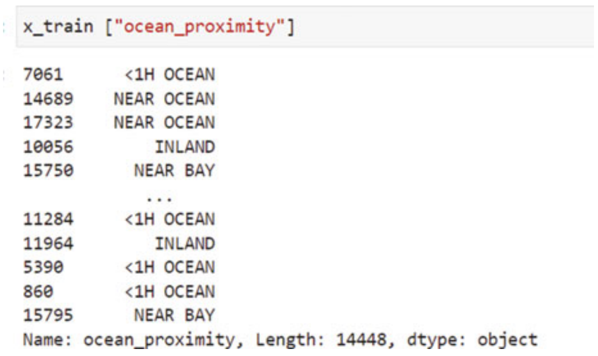
Try to display both x_train_num and x_train_num_imputed and you will see the result.

4.12.1.4 One-Hot-Encode Data

Now let us one-hot-encode the proximity to the ocean feature. Remember that df [“ocean_proximity”].value_counts() displayed 5 different categorical values for ocean_proximity (Fig. 4.15). Try to display the ocean_proximity values by writing x_train [“ocean_proximity”] (Fig. 4.19); we can notice that it is an array.

We need to store the data in a data frame to apply one-hot-encoder. Write proximity = pd.DataFrame(x_train[“ocean_proximity”]) to create a new variable called “proximity” that holds the data frame, then display the data frame to check the result (Fig. 4.20).

Follow the instructions in Fig. 4.21 to one-hot-encode proximity. Notice the . fit_transform() function, it fits the transformer to *proximity* and then it transforms data in *proximity*. When we tried to display the encoded variable proximity_one_hot_encoded the array was large and contained a lot of zeros in

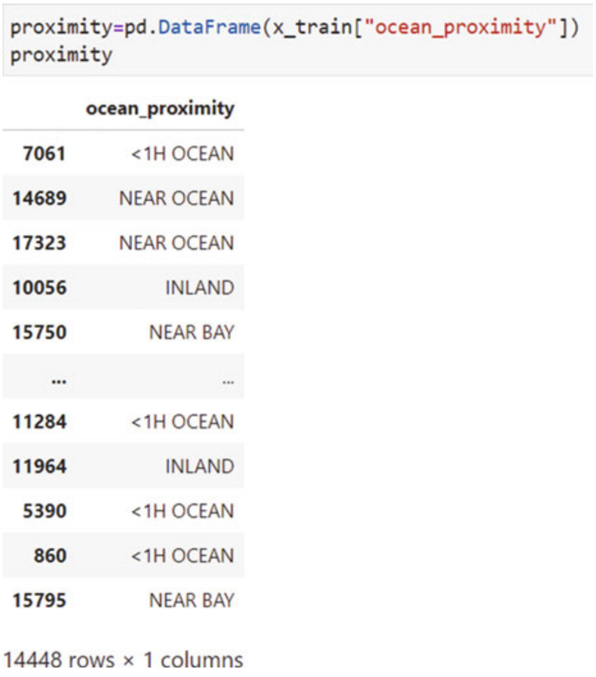


```
x_train ["ocean_proximity"]
```

```
7061      <1H OCEAN
14689    NEAR OCEAN
17323    NEAR OCEAN
10056        INLAND
15750      NEAR BAY
            ...
11284     <1H OCEAN
11964        INLAND
5390      <1H OCEAN
860       <1H OCEAN
15795      NEAR BAY
Name: ocean_proximity, Length: 14448, dtype: object
```

Fig. 4.19 Displaying proximity as an array

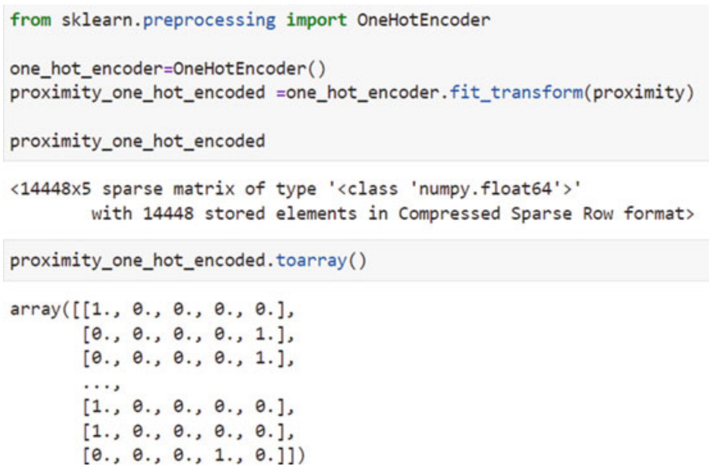
Fig. 4.20 Transform the array proximity to a data frame and save it in proximity



```
proximity=pd.DataFrame(x_train["ocean_proximity"])
proximity
```

	ocean_proximity
7061	<1H OCEAN
14689	NEAR OCEAN
17323	NEAR OCEAN
10056	INLAND
15750	NEAR BAY
...	...
11284	<1H OCEAN
11964	INLAND
5390	<1H OCEAN
860	<1H OCEAN
15795	NEAR BAY

14448 rows × 1 columns



```
from sklearn.preprocessing import OneHotEncoder

one_hot_encoder=OneHotEncoder()
proximity_one_hot_encoded =one_hot_encoder.fit_transform(proximity)

proximity_one_hot_encoded
```

```
<14448x5 sparse matrix of type '<class 'numpy.float64'>'
	with 14448 stored elements in Compressed Sparse Row format>
```

```
proximity_one_hot_encoded.toarray()
```

```
array([[1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       ...,
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

Fig. 4.21 One-Hot-Encoding in python of the *proximity* data frame

each column; that is called a sparse array in python and cannot be displayed, to display it we have used the .toarray() method.

4.12.1.5 Scale Numeric Data: Standardization

To scale the numeric data that we have imputed, follow the instruction in Fig. 4.22. The instructions are self-explanatory by now. If you need to normalize data, you can use the MinMaxScaler in the same manner you have used the StandardScaler. Not that we scaled x_train_num_imputed not x_train_num.

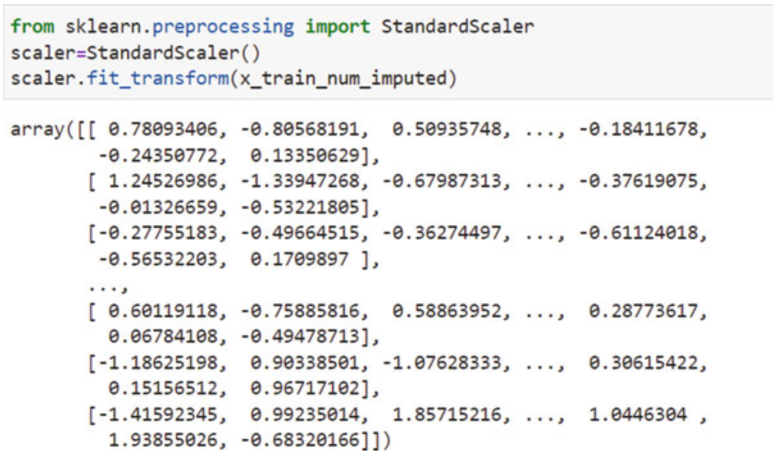
4.12.1.6 Create Pipelines

More to learn in the next chapters. For the time being, try can explore more how to put back together in one data frame, the *x_tain_num* data frame containing all the numeric imputed and standardized features and proximity_one_hot_encoded containing the one-hot-encoded ocean proximity feature (check Pipeline and ColumnTransformer in Fig. 4.23). You can read more about python data frames and more on this website: https://docs.python.org/3/tutorial/

4.12.1.7 Creating Models

To complete our exploration of how to proceed in a machine learning project, let us move a bit further and use a few algorithms that will be covered in detail later. In the following, we will use a linear regression, random forest, and decision trees [37].

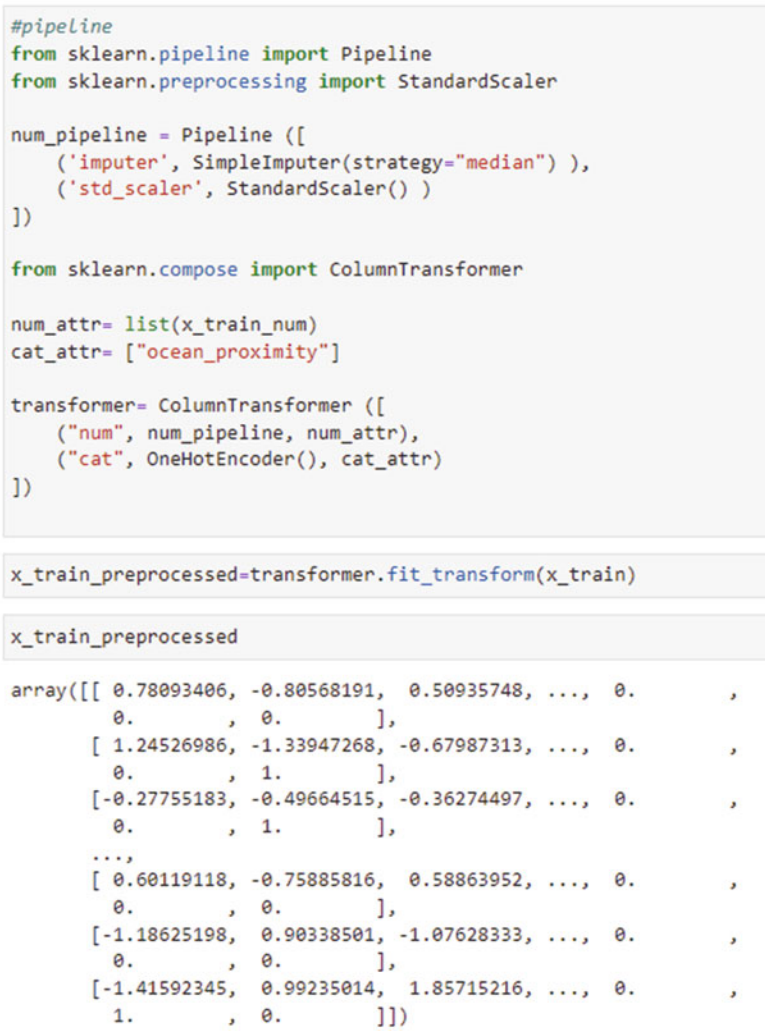
Linear regression is one algorithm that we can use to estimate the price of a house based on our dataset, we can use the root-mean-square error (RMSE) as a measure of the prediction error the algorithm is making. Figure 4.24 shows that the RMSE (68763.97) for the linear regression model fitted on the x_train_preprocessed dataset.



```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit_transform(x_train_num_imputed)
```

```
array([[ 0.78093406, -0.80568191,  0.50935748, ..., -0.18411678,
        -0.24350772,  0.13350629],
       [ 1.24526986, -1.33947268, -0.67987313, ..., -0.37619075,
        -0.01326659, -0.53221805],
       [-0.27755183, -0.49664515, -0.36274497, ..., -0.61124018,
        -0.56532203,  0.1709897 ],
       ...,
       [ 0.60119118, -0.75885816,  0.58863952, ...,  0.28773617,
         0.06784108, -0.49478713],
       [-1.18625198,  0.90338501, -1.07628333, ...,  0.30615422,
         0.15156512,  0.96717102],
       [-1.41592345,  0.99235014,  1.85715216, ...,  1.0446304 ,
         1.93855026, -0.68320166]])
```

Fig. 4.22 Standardization of the numeric data frame



```
#pipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline ([
    ('imputer', SimpleImputer(strategy="median") ),
    ('std_scaler', StandardScaler() )
])

from sklearn.compose import ColumnTransformer

num_attr= list(x_train_num)
cat_attr= ["ocean_proximity"]

transformer= ColumnTransformer ([
    ("num", num_pipeline, num_attr),
    ("cat", OneHotEncoder(), cat_attr)
])
```

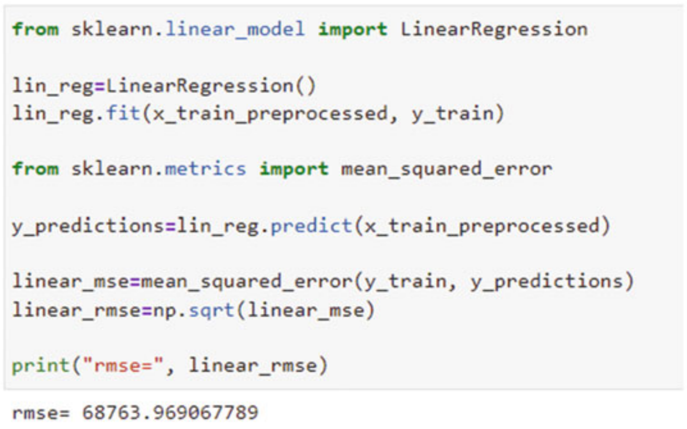
```
x_train_preprocessed=transformer.fit_transform(x_train)
```

```
x_train_preprocessed
```

```
array([[ 0.78093406, -0.80568191,  0.50935748, ...,  0.        ,
         0.        ,  0.        ],
       [ 1.24526986, -1.33947268, -0.67987313, ...,  0.        ,
         0.        ,  1.        ],
       [-0.27755183, -0.49664515, -0.36274497, ...,  0.        ,
         0.        ,  1.        ],
       ...,
       [ 0.60119118, -0.75885816,  0.58863952, ...,  0.        ,
         0.        ,  0.        ],
       [-1.18625198,  0.90338501, -1.07628333, ...,  0.        ,
         0.        ,  0.        ],
       [-1.41592345,  0.99235014,  1.85715216, ...,  0.        ,
         1.        ,  0.        ]])
```

Fig. 4.23 Preprocessing the training data set using a Pipeline and a ColumnTransformer

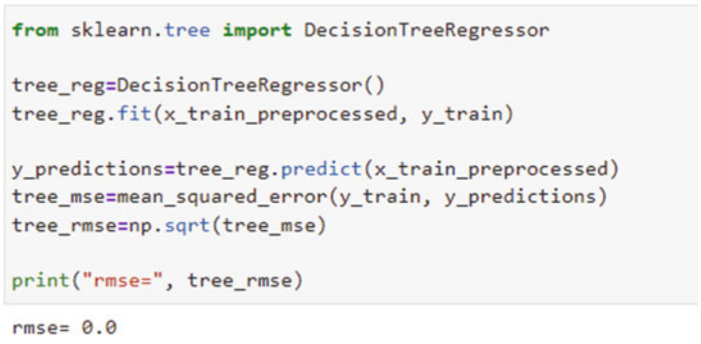
While the RMSE for a decision tree regressor algorithm shows a zero as error (RMSE = 0) (Fig. 4.25). Whenever it is too good to be true then it is not true. The model has fitted the training dataset so well that it did not do any error in predicting the houses' prices; obviously, the model overfitted the training data.



```
from sklearn.linear_model import LinearRegression

lin_reg=LinearRegression()
lin_reg.fit(x_train_preprocessed, y_train)

from sklearn.metrics import mean_squared_error

y_predictions=lin_reg.predict(x_train_preprocessed)

linear_mse=mean_squared_error(y_train, y_predictions)
linear_rmse=np.sqrt(linear_mse)

print("rmse=", linear_rmse)
```

```
rmse= 68763.969067789
```

Fig. 4.24 Applying linear regression and checking the algorithm performance (RMSE)



```
from sklearn.tree import DecisionTreeRegressor

tree_reg=DecisionTreeRegressor()
tree_reg.fit(x_train_preprocessed, y_train)

y_predictions=tree_reg.predict(x_train_preprocessed)
tree_mse=mean_squared_error(y_train, y_predictions)
tree_rmse=np.sqrt(tree_mse)

print("rmse=", tree_rmse)
```

```
rmse= 0.0
```

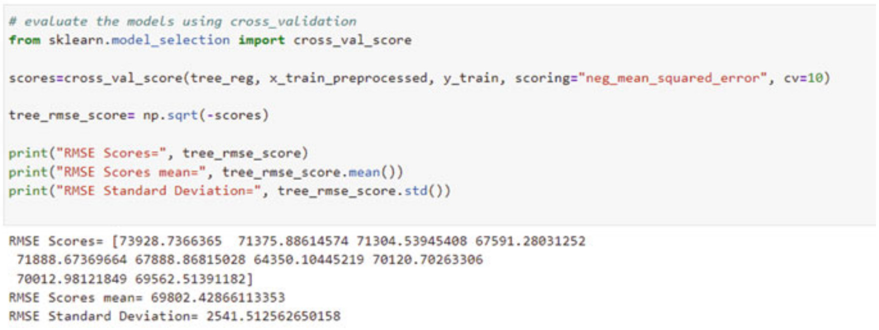
Fig. 4.25 Applying the decision tree regressor algorithm and checking the algorithm performance (RMSE)

4.12.1.8 Cross-Validation

In Figs. 4.26 and 4.27 we can see the performance of the decision tree and the linear regression trained models evaluated using ten-fold cross-validation. $K = 10$ has proven to be leading to good estimates.

We can notice, that the mean RMSE score for the decision tree model is much higher than zero (which confirms that the learned model overfitted the training dataset). Cross-validation will display the scores for each iteration, and the mean and standard deviation of the scores.

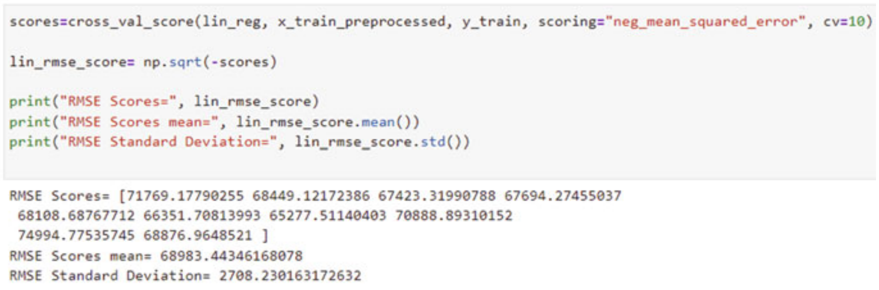
Let us try a third algorithm called random forest. In Fig. 4.28, we can notice the random forest model training, the RMSE score after training (18496) and the mean RMSE score after k-fold cross-validation (49710), the latter is lower than the corresponding performance for both the linear regression and the decision tree



```
# evaluate the models using cross_validation
from sklearn.model_selection import cross_val_score

scores=cross_val_score(tree_reg, x_train_preprocessed, y_train, scoring="neg_mean_squared_error", cv=10)

tree_rmse_score= np.sqrt(-scores)

print("RMSE Scores=", tree_rmse_score)
print("RMSE Scores mean=", tree_rmse_score.mean())
print("RMSE Standard Deviation=", tree_rmse_score.std())
```

```
RMSE Scores= [73928.7366365  71375.88614574 71304.53945408 67591.28031252
 71888.67369664 67888.86815028 64350.10445219 70120.70263306
 70012.98121849 69562.51391182]
RMSE Scores mean= 69802.42866113353
RMSE Standard Deviation= 2541.512562650158
```

Fig. 4.26 Applying cross-validation to evaluate the performance of the decision tree on an evaluation dataset



```
scores=cross_val_score(lin_reg, x_train_preprocessed, y_train, scoring="neg_mean_squared_error", cv=10)

lin_rmse_score= np.sqrt(-scores)

print("RMSE Scores=", lin_rmse_score)
print("RMSE Scores mean=", lin_rmse_score.mean())
print("RMSE Standard Deviation=", lin_rmse_score.std())
```

```
RMSE Scores= [71769.17790255 68449.12172386 67423.31990788 67694.27455037
 68108.68767712 66351.70813993 65277.51140403 70888.89310152
 74994.77535745 68876.9648521 ]
RMSE Scores mean= 68983.44346168078
RMSE Standard Deviation= 2708.230163172632
```

Fig. 4.27 Applying cross-validation to evaluate the performance of the linear regression on an evaluation dataset

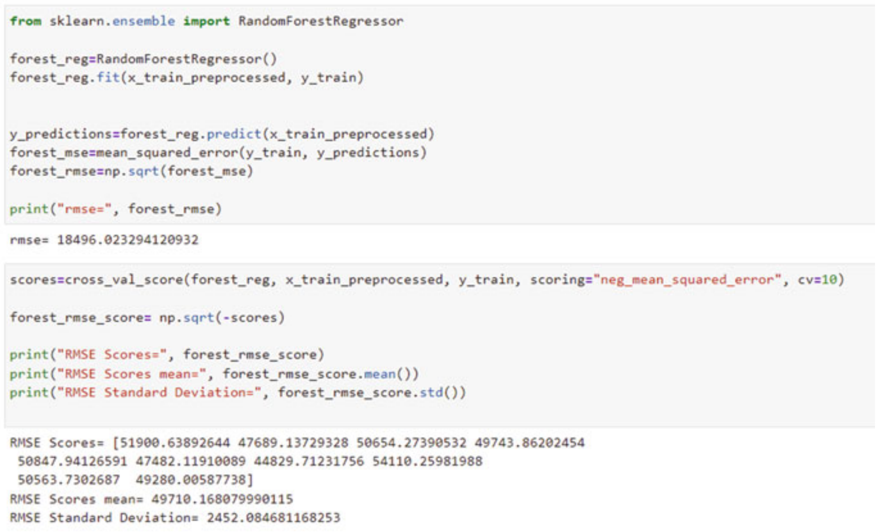
regressor models. The random forest model seems a good candidate for our dataset and we can adopt it as the model of choice.

4.12.1.9 Hyperparameter Finetuning

Two final steps are needed: fine tune the model of choice and testing it on the testing dataset. Each algorithm uses a set of hyperparameters; these are parameters that are set before the learning starts; they are different than the model's parameters that are found through the learning process (e.g., the coefficients of a linear model). Each algorithm has default values for its hyperparameters.

So, the random forest model found using the default hyperparameters is only one combination of values among many others; if we tweak the hyperparameters of the random forest (or any other algorithm) we might end up with a model that performs better (or worse) than the one we found using the default hyperparameters' values.

So, we can try to tweak the hyperparameters to find the optimal model for our dataset. In Fig. 4.29 we can find the instructions in python to finetune two random



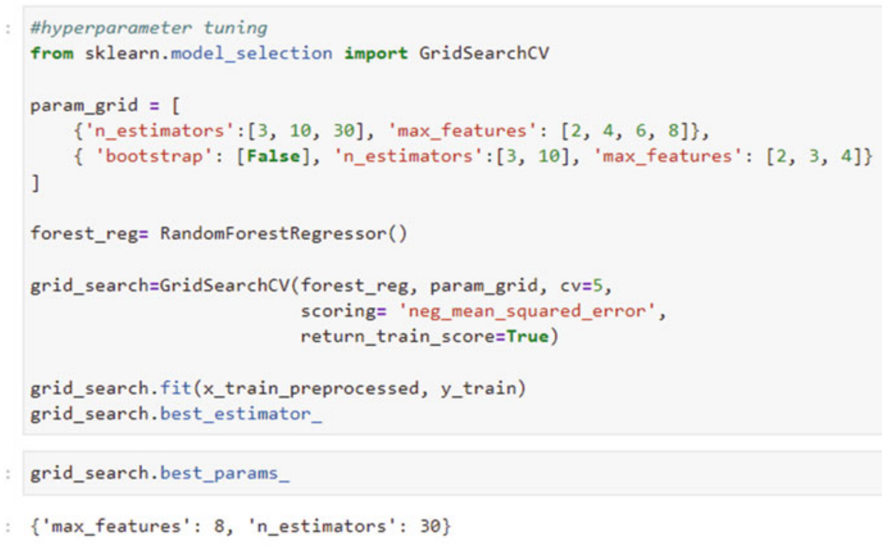
```
from sklearn.ensemble import RandomForestRegressor

forest_reg=RandomForestRegressor()
forest_reg.fit(x_train_preprocessed, y_train)


y_predictions=forest_reg.predict(x_train_preprocessed)
forest_mse=mean_squared_error(y_train, y_predictions)
forest_rmse=np.sqrt(forest_mse)

print("rmse=", forest_rmse)
```

```
rmse= 18496.023294120932
```

```
scores=cross_val_score(forest_reg, x_train_preprocessed, y_train, scoring="neg_mean_squared_error", cv=10)

forest_rmse_score= np.sqrt(-scores)

print("RMSE Scores=", forest_rmse_score)
print("RMSE Scores mean=", forest_rmse_score.mean())
print("RMSE Standard Deviation=", forest_rmse_score.std())
```

```
RMSE Scores= [51900.63892644 47689.13729328 50654.27390532 49743.86202454
 50847.94126591 47482.11910089 44829.71231756 54110.25981988
 50563.7302687  49280.00587738]
RMSE Scores mean= 49710.168079990115
RMSE Standard Deviation= 2452.084681168253
```

Fig. 4.28 Applying the random forest algorithm and checking the algorithm performance (RMSE)



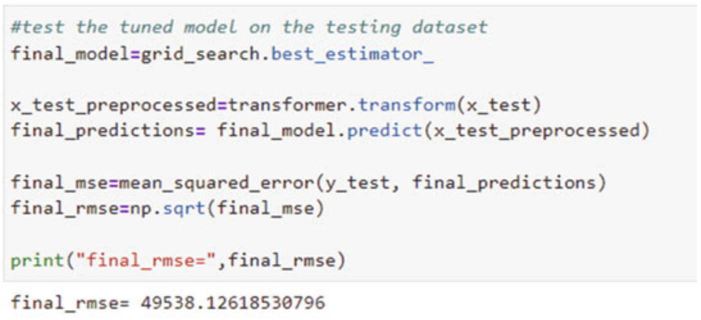
```
#hyperparameter tuning
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators':[3, 10, 30], 'max_features': [2, 4, 6, 8]},
    { 'bootstrap': [False], 'n_estimators':[3, 10], 'max_features': [2, 3, 4]}
]

forest_reg= RandomForestRegressor()

grid_search=GridSearchCV(forest_reg, param_grid, cv=5,
                         scoring= 'neg_mean_squared_error',
                         return_train_score=True)

grid_search.fit(x_train_preprocessed, y_train)
grid_search.best_estimator_
```

```
grid_search.best_params_
```

```
{'max_features': 8, 'n_estimators': 30}
```

Fig. 4.29 Fine tuning the hyperparameters of our algorithm of choice (e.g., the random forest) and checking the best hyperparameters

forest hyperparameters (n_estiamtors and max_features). We are using the GridSearchCV algorithm in python to find the best estimates for those parameters among a set of values that we provide. In our example, we provide for n_estimators



```
#test the tuned model on the testing dataset
final_model=grid_search.best_estimator_

x_test_preprocessed=transformer.transform(x_test)
final_predictions= final_model.predict(x_test_preprocessed)

final_mse=mean_squared_error(y_test, final_predictions)
final_rmse=np.sqrt(final_mse)

print("final_rmse=",final_rmse)
```

```
final_rmse= 49538.12618530796
```

Fig. 4.30 Testing the performance of the fine-tuned random forest model on the testing dataset

three values to try 3, 10, and 30. GridSearchCV go through all the provided hyperparameters' values and create a model for each combination of values and assess its performance, then chooses the best performing model., RandomizedSearchCV is another way of searching for the best model; we provide for RandomizedSearchCV a range of valid values and RandomizedSearchCV selected randomly from these sets of values to generate the best model.

Notice that after finding the optimal model we have trained it on the training dataset (Fig. 4.29). The last step is to finally measure the optimal model's performance on the testing dataset (Fig. 4.30). Frist we apply the preprocessing steps to the testing dataset using the transformer that was fitted on the training dataset (i.e., we never use transformer.fit() on the testing dataset).

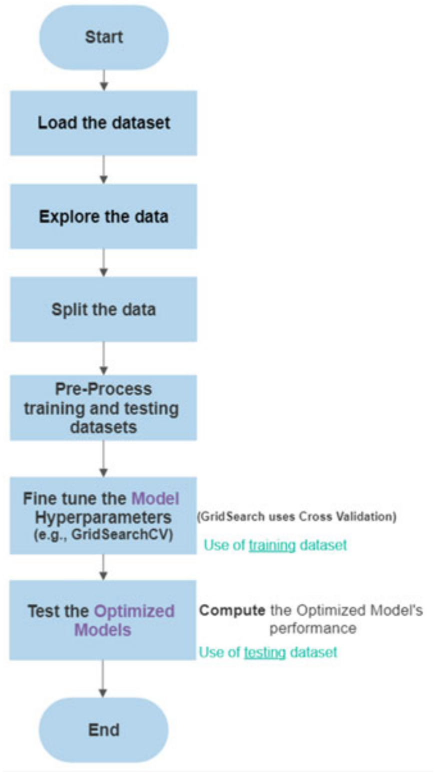
Using GridSearch, we have demonstrated the hyperparameter tuning for the random forest model only. The whole process can be seen in Fig. 4.31. In a project, you might want to tune several candidate models, measure their performances using the testing dataset, and choose the best-performing one.

NOTE: The machine learning labs, starting at Chap. 6, demonstrate the use of one specific algorithm. Hence, those labs are structured generally in the following way: split the data into training and testing datasets, pre-process the datasets, display graphs, finetune the model's hyperparameters (e.g., perform GridSearchCV), fit the optimal model on the testing dataset, and finally test it on the testing dataset and display its performance (Fig. 4.31). However, there is little commenting on the displayed graphs.

Hence, your first mission in every machine learning lab will be to:

1. Check the graphs and comment on interesting observations.
2. Once you have learned a few algorithms, you can compare the performance of few and choose the best-performing one.

Fig. 4.31 Steps in a Machine learning project


Start
Load the dataset
Explore the data
Split the data
Pre-Process training and testing datasets
Fine tune the Model Hyperparameters (e.g., GridSearchCV)
(GridSearch uses Cross Validation)
Use of training dataset
Test the Optimized Models
Compute the Optimized Model's performance
Use of testing dataset
End

4.12.2 Working Example in Weka

4.12.2.1 Missing Values

We will use the Pima Indians diabetes dataset that can be found with other well-known datasets in the UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/ or https://archive-beta.ics.uci.edu/) as well as on Kaggle.com (https://www.kaggle.com/). You can download the dataset from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. However, the diabetes.arff file, formatted for Weka, can be found in the data folder of the installed software.

The diabetes dataset contains a list of predictor variables to the outcome of interest (i.e., diabetes) for women 21 years of age or older. Predictors include the following variables:

– Pregnancies: the number of pregnancies the woman has had
– Glucose: plasma glucose concentration (mmol/L)
– BloodPressure: diastolic blood pressure (mm Hg)

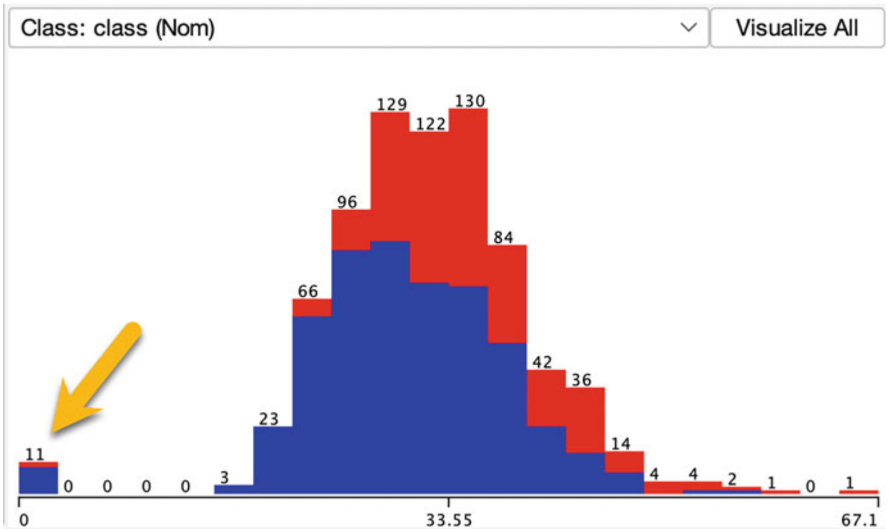
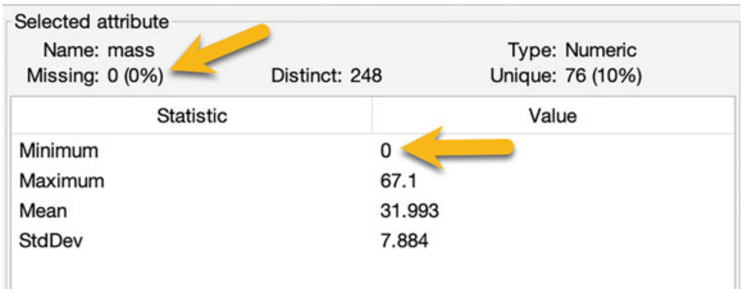


Fig. 4.32 Histogram of the BMI variable in the diabetes dataset displayed in Weka



Selected attribute

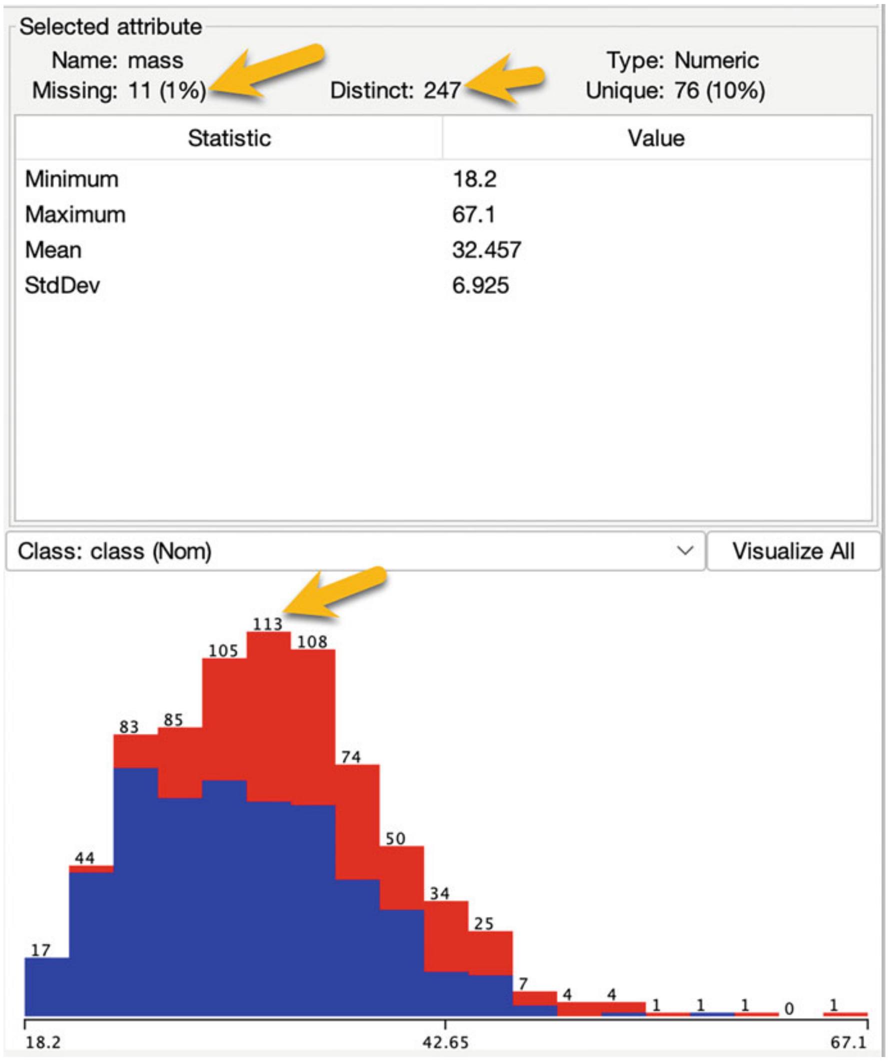
Name: mass Type: Numeric
Missing: 0 (0%) Distinct: 248 Unique: 76 (10%)

Statistic	Value
Minimum	0
Maximum	67.1
Mean	31.993
StdDev	7.884

Fig. 4.33 BMI attribute descriptive statistics

– SkinThickness: triceps skinfold thickness (mm)
– Insulin: insulin level (mu U/ml)
– BMI: body mass index (kg/m^2)
– DiabetesPedigreeFunction: diabetes pedigree function (a function that scores the probability of diabetes based on family history)
– Age: the age in years
– Outcome: 0 for not diabetic, and 1 for diabetic

To understand the distribution of values of predictors, a histogram is beneficial. The histogram of the mass variable in the diabetes dataset shows us that 11 data instances have missing BMI measurements, i.e., mass = 0, which is not a possible value for BMI (Figs. 4.32 and 4.33).



Selected attribute

Name: mass
Missing: 11 (1%)
Distinct: 247
Type: Numeric
Unique: 76 (10%)

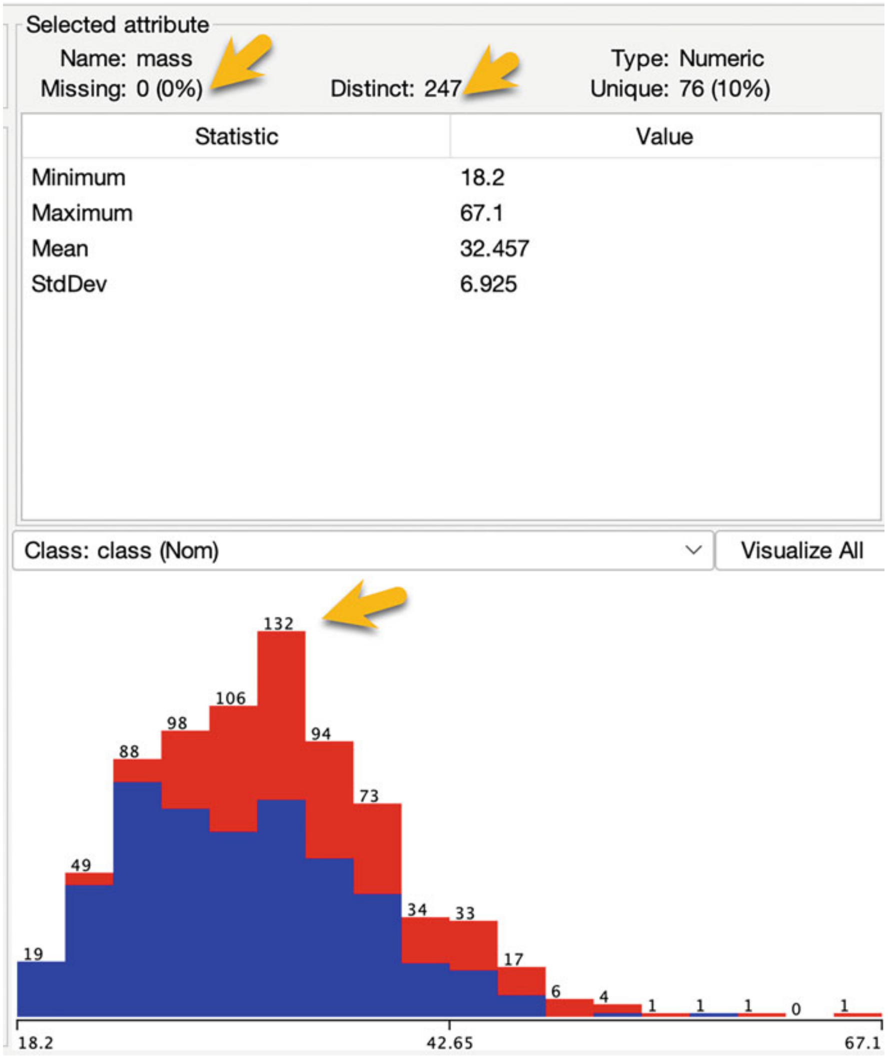
Statistic	Value
Minimum	18.2
Maximum	67.1
Mean	32.457
StdDev	6.925

Class: class (Nom)

Fig. 4.34 Marking missing values using NumericCleaner

We can decide to delete these 11 instances or to impute the missing values. Let's delete them. We can always verify the result by plotting the histogram again.

We can mark the missing values using the NumericCleaner filter (Fig. 4.34); then we can either delete all instances containing 0 as a BMI value using the RemoveWithValues (Fig. 4.35) or replace the 0 with the mean BMI using the ReplaceMissingValues filter (Fig. 4.36). To apply NumericCleaner, hoose the NumericCleaner filter from Filter/Usupervised/Attribute. Click on its name. A list of parameters appear. Set the following parameters: (1) *attributeIndex* to 6 (i.e., the

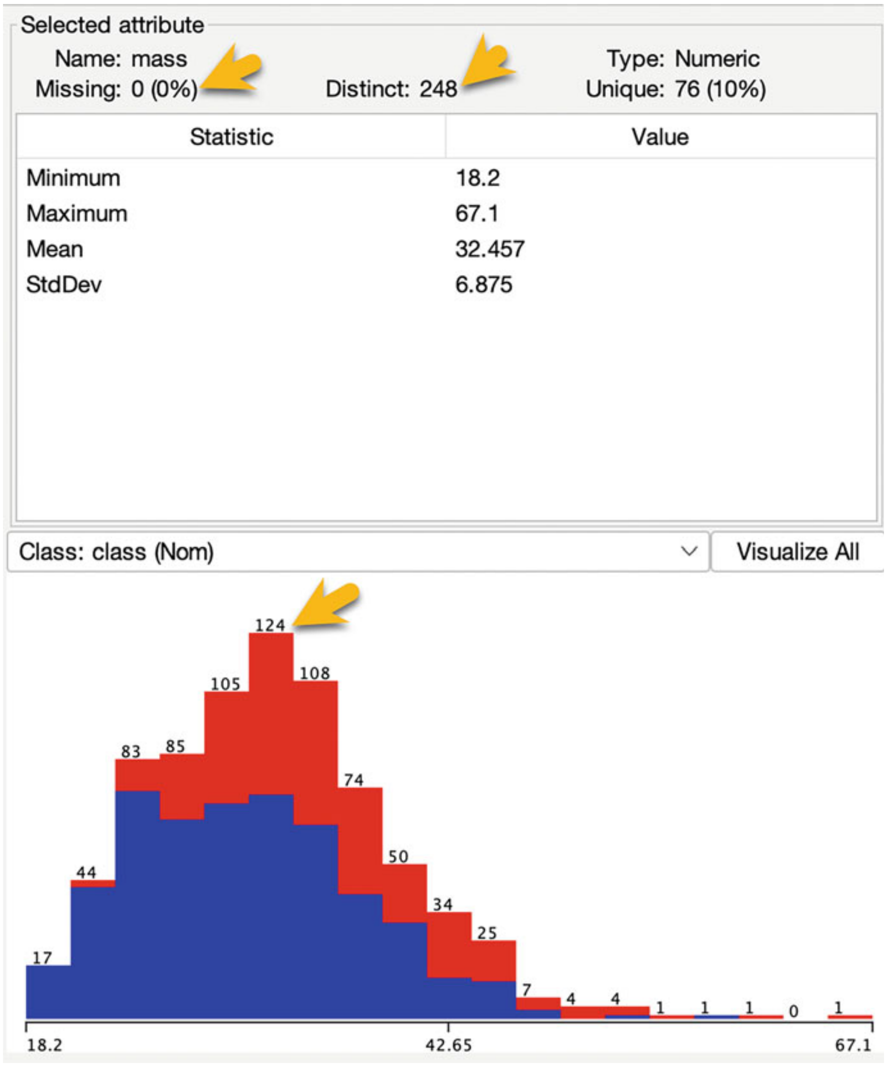

Selected attribute
Name: mass
Missing: 0 (0%)
Distinct: 247
Type: Numeric
Unique: 76 (10%)

Statistic	Value
Minimum	18.2
Maximum	67.1
Mean	32.457
StdDev	6.925

Class: class (Nom)

Fig. 4.35 Histogram of the BMI variable in the diabetes dataset after deleting instances of missing BMI

mass), (2) the minimum allowed for the attribute *minThreshold* to a value close to zero (e.g., 0.000001), and the *minDefault* to the value NaN, which represents the value "unknown" and will be used to replace that values that are below the threshold (in our case the value zero). Click on Apply. For RemoveWithValues filter that can be found under Filter/Usupervised/Instance, set the *attributeIndices* to 6 and *matchMissingValues* to True, then click on Apply. For the ReplaceMissingValue filter that can be found under Filter/Usupervised/Attribute, there are no parameters to set.


Selected attribute
Name: mass
Missing: 0 (0%)
Distinct: 248
Type: Numeric
Unique: 76 (10%)

Statistic	Value
Minimum	18.2
Maximum	67.1
Mean	32.457
StdDev	6.875

Class: class (Nom)

Fig. 4.36 Histogram of the BMI variable in the diabetes dataset after replacing instances of missing BMI with the mean BMI. Notice that the number of distinct values has increased by one (from 247 to 248) and the number of instances around the mean value mean has increased by 11 (from 113 to 124)

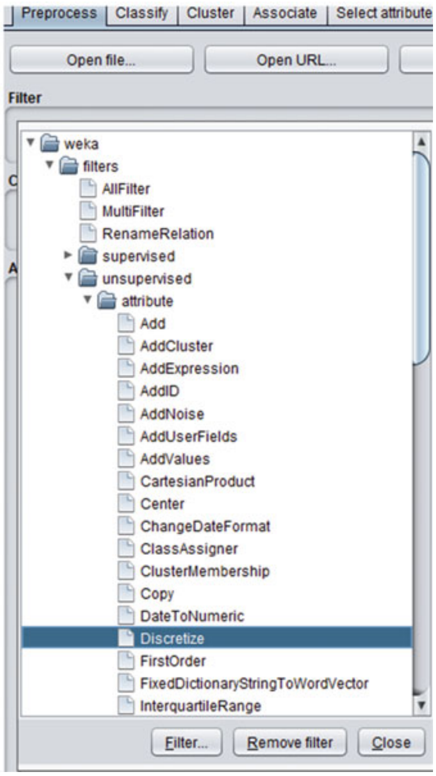


Fig. 4.37 Choosing Discretize filter in Weka

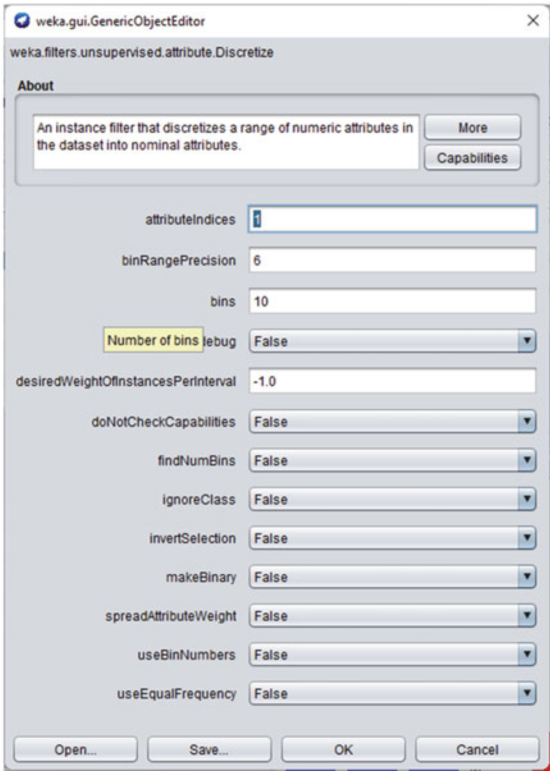
4.12.2.2 Discretization (or Binning)

To discretize one or more attributes in Weka, we can use the filter unsupervised/attribute/Discretize (Fig. 4.37). Type "1" in the Attribute Indices to discretize the Pregnancies attribute (or leave it first-last if you want to attempt to discretize all attributes (Fig. 4.38); other parameters that you can explore can also be set. The result displayed shows 10 discrete intervals (Fig. 4.39). As mentioned above, some algorithms such as naïve Bayes perform better using discrete values.

4.12.2.3 Data Normalization and Standardization

In Weka, we can use the unsupervised/attribute/Normalize filter to normalize an attribute and the unsupervised/attribute/Standardize filter to standardize

Fig. 4.38 Changing filter parameters


weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.Discretize

About

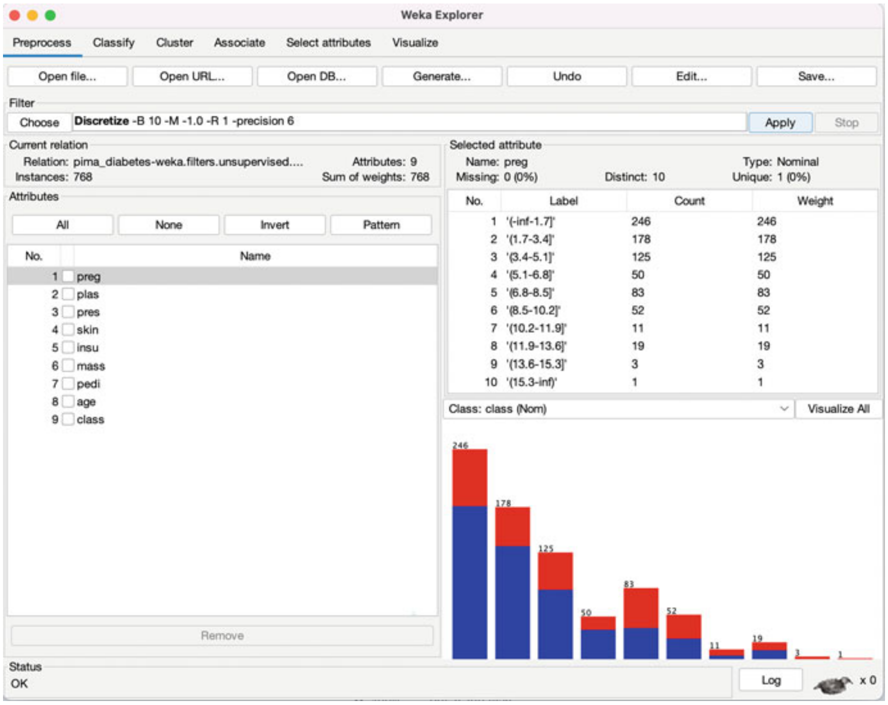
An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

attributeIndices: 1
binRangePrecision: 6
bins: 10
debug: False
desiredWeightOfInstancesPerInterval: -1.0
doNotCheckCapabilities: False
findNumBins: False
ignoreClass: False
invertSelection: False
makeBinary: False
spreadAttributeWeight: False
useBinNumbers: False
useEqualFrequency: False

it. Figure 4.40 shows the results of normalizing the whole dataset, while Fig. 4.41 shows the results of standardizing it.

4.12.2.4 One-Hot-Encoding (Nominal to Numeric)

Figures 4.42 and 4.43 show the result of converting the discrete feature Age in the contact lenses dataset to a numeric value using unsupervised/attribute/NominalToBinary. The dataset can be found in the Weka data folder, or in the UCI repository https://archive.ics.uci.edu/ml/datasets/Lenses. It would be much easier to use the Weka file “contact-lenses.arff”.



Filter: **Discretize** -B 10 -M -1.0 -R 1 -precision 6

Current relation
Relation: pima_diabetes-weka.filters.unsupervised.... Attributes: 9
Instances: 768 Sum of weights: 768

Attributes

No.	Name
1	preg
2	plas
3	pres
4	skin
5	insu
6	mass
7	pedi
8	age
9	class

Selected attribute
Name: preg Type: Nominal
Missing: 0 (0%) Distinct: 10 Unique: 1 (0%)

No.	Label	Count	Weight
1	'(-inf-1.7]'	246	246
2	'(1.7-3.4]'	178	178
3	'(3.4-5.1]'	125	125
4	'(5.1-6.8]'	50	50
5	'(6.8-8.5]'	83	83
6	'(8.5-10.2]'	52	52
7	'(10.2-11.9]'	11	11
8	'(11.9-13.6]'	19	19
9	'(13.6-15.3]'	3	3
10	'(15.3-inf)'	1	1

Class: class (Nom)

Status
OK

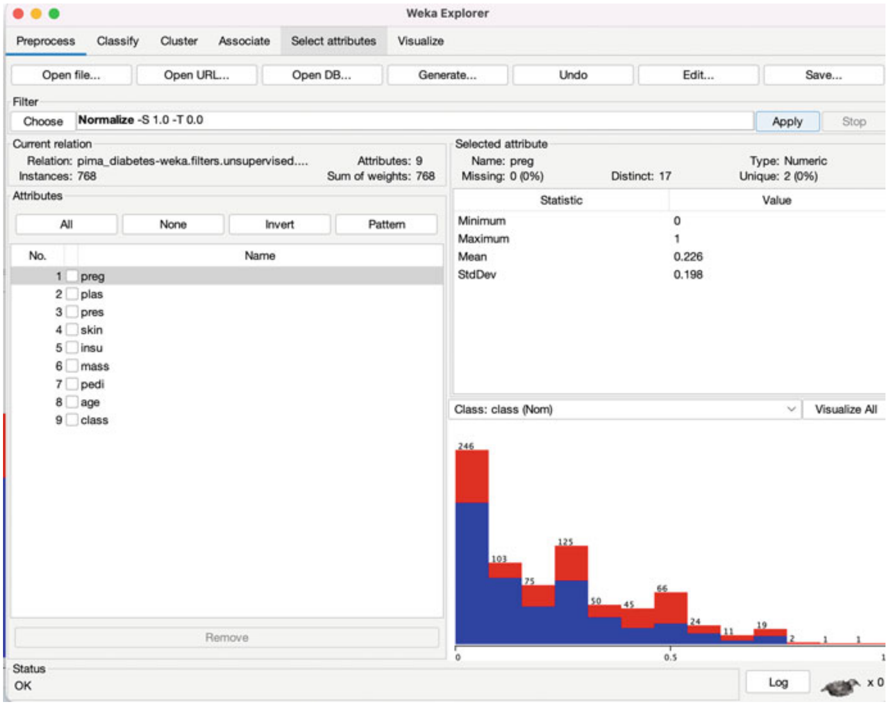
Fig. 4.39 Changing filter parameters

4.12.3 Do It Yourself

4.12.3.1 Lenses Dataset

The following exercise is for you to execute using RStudio and Python.

1. Using the lenses dataset:
   (a) Perform discretization of the Spectacle-Prescrip attribute.
   (b) What other attributes can be discretized?
2. Using the diabetes dataset:
   (a) Normalize all attributes.
   (b) How would you check if the normalization filter was executed?
   (c) Standardize all attributes.



Filter: Normalize -S 1.0 -T 0.0

Current relation
Relation: pima_diabetes-weka.filters.unsupervised.... Attributes: 9
Instances: 768 Sum of weights: 768

Attributes

No.	Name
1	preg
2	plas
3	pres
4	skin
5	insu
6	mass
7	pedi
8	age
9	class

Selected attribute
Name: preg Type: Numeric
Missing: 0 (0%) Distinct: 17 Unique: 2 (0%)

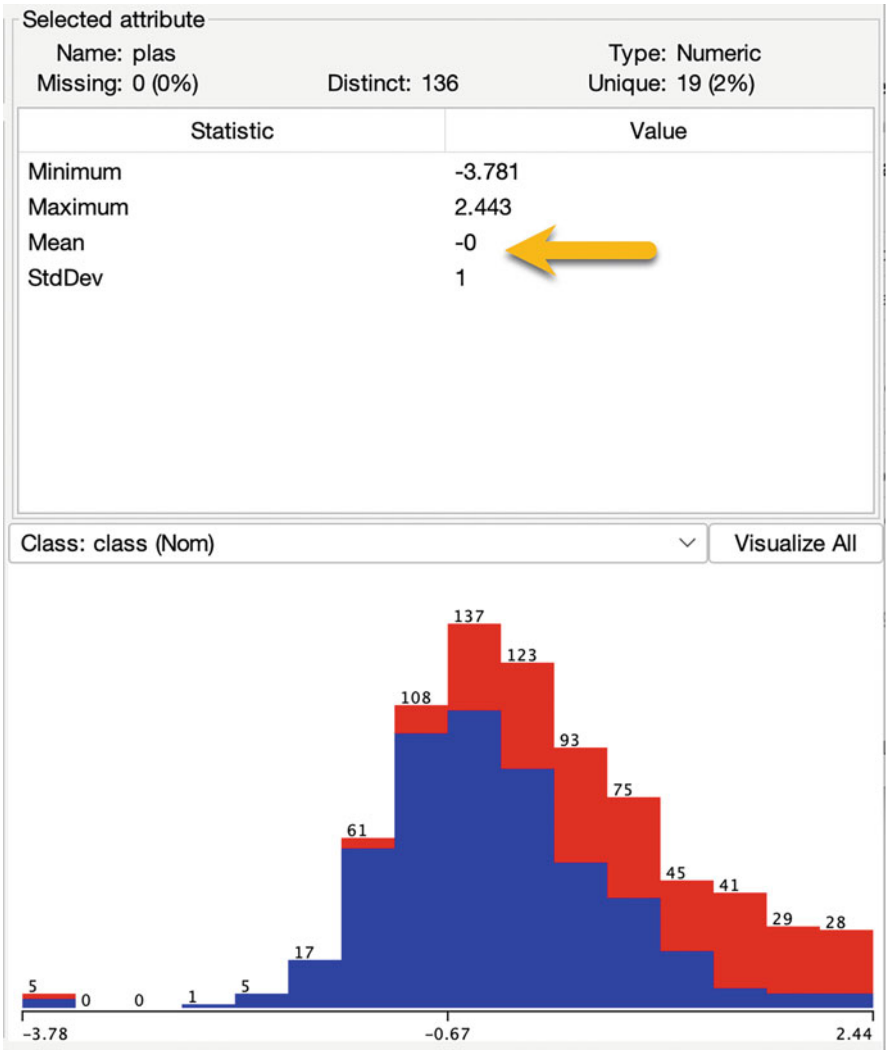
Statistic	Value
Minimum	0
Maximum	1
Mean	0.226
StdDev	0.198

Class: class (Nom)

Status
OK

Fig. 4.40 The result of normalizing the diabetes dataset; the minimum and maximum of every attribute are 0 and 1

(d) How would you check if the normalization filter was executed?
(e) Create a new dataset where missing values in the attribute Pregnancies are deleted.
(f) Create a new dataset where missing values in the attribute Pregnancies are imputed with the mean of the attribute.
(g) What other imputation methods can be used besides the mean?
(h) Create a new dataset where missing values in attribute Pregnancies are imputed with a method other than the mean?
(i) Are some imputation methods better than others? How?

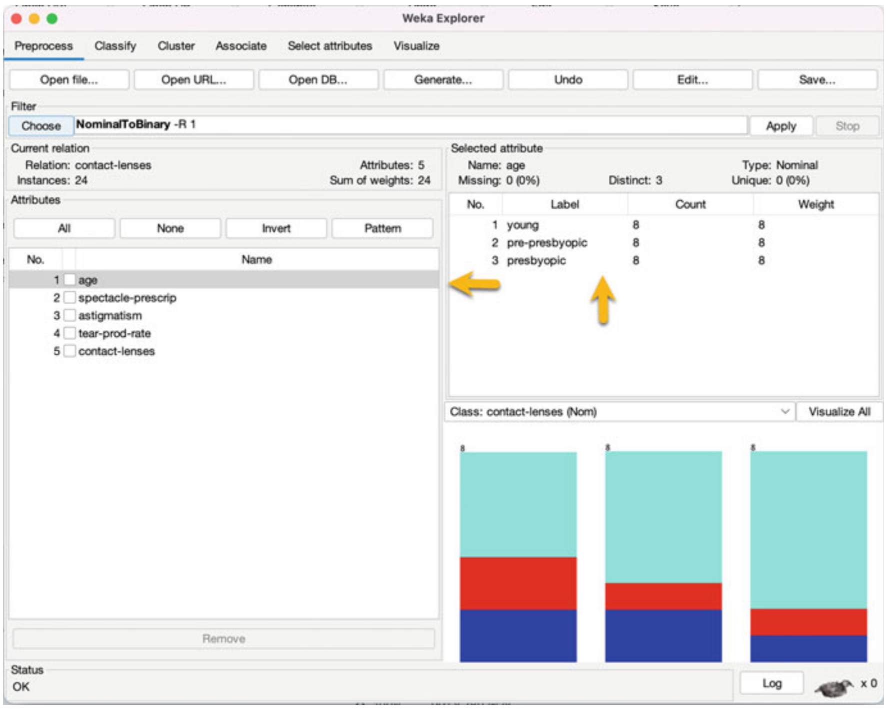


Selected attribute

Name: plas
Missing: 0 (0%)
Distinct: 136
Type: Numeric
Unique: 19 (2%)

Statistic	Value
Minimum	-3.781
Maximum	2.443
Mean	-0
StdDev	1

Fig. 4.41 The result of standardizing the diabetes dataset; the mean and standard of every attribute are 0 and 1



Filter: NominalToBinary -R 1

Current relation
Relation: contact-lenses Attributes: 5
Instances: 24 Sum of weights: 24

Attributes

No.	Name
1	age
2	spectacle-prescrip
3	astigmatism
4	tear-prod-rate
5	contact-lenses

Selected attribute
Name: age Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	young	8	8
2	pre-presbyopic	8	8
3	presbyopic	8	8

Class: contact-lenses (Nom)

Status
OK

Fig. 4.42 Age attribute set to three categories before discretization

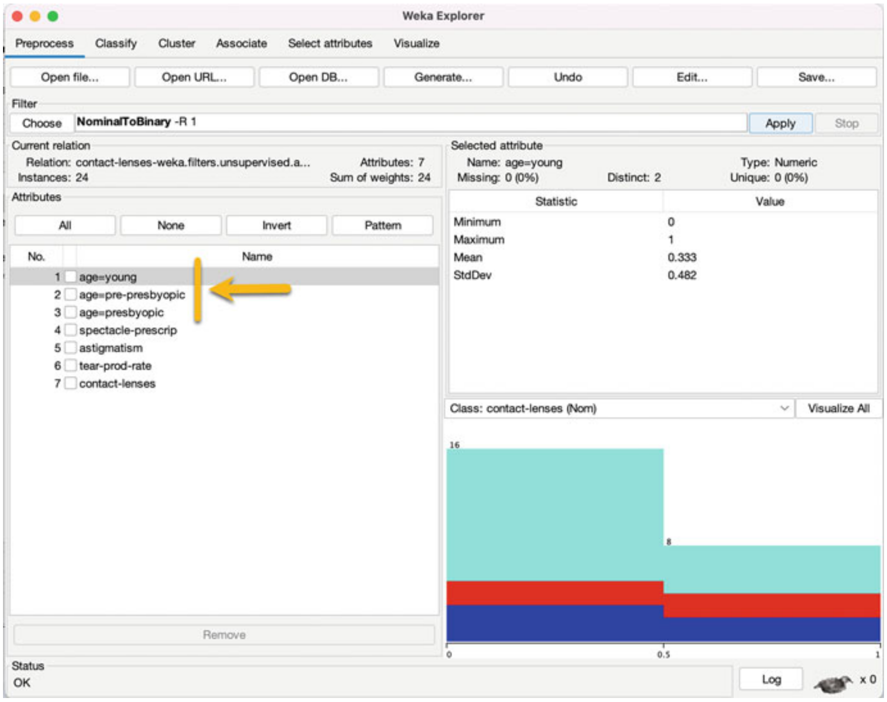
4.12.3.2 Nested Cross-Validation

For the California housing prices problem presented above, implement nested cross-validation in Python for the Random Forest, Logistic Regression, and Decision trees. Use the hyperparameters used above.

4.12.4 Do More Yourself

You can apply the strategies explained in his chapter on other datasets, such as:

1. Glass identification dataset, which can be downloaded using the following link: https://archive-beta.ics.uci.edu/ml/datasets/42
2. Labor relation dataset, which can be downloaded using the following link: https://archive-beta.ics.uci.edu/ml/datasets/56


Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter

Choose NominalToBinary -R 1 Apply Stop

Current relation

Relation: contact-lenses-weka.filters.unsupervised.a... Attributes: 7
Instances: 24 Sum of weights: 24

Selected attribute

Name: age=young Type: Numeric
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

Statistic	Value
Minimum	0
Maximum	1
Mean	0.333
StdDev	0.482

Attributes

All None Invert Pattern

No.	Name
1	age=young
2	age=pre-presbyopic
3	age=presbyopic
4	spectacle-prescrip
5	astigmatism
6	tear-prod-rate
7	contact-lenses

Class: contact-lenses (Nom) Visualize All

Remove

Status
OK

Fig. 4.43 Effect of discretization on the Age attribute

References

1. S. Bandgar, *Data Preparation in Data Science*. Analytics Vidhya. https://medium.com/analytics-vidhya/data-preparation-in-data-science-16f9311760. Accessed 17 March 2022
2. B.C. Boehmke, *Data Wrangling with R (Use R!)* (Springer International Publishing, 2016)
3. H. Jamshed, M.S.A. Khan, M. Khurram, S. Inayatullah, S. Athar, Data preprocessing: A preliminary step for web data mining. *3C Tecnología_Glosas de innovación aplicadas a la pyme* **8**(1), 206–221 (2019)
4. G. Press, *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says*. Forbes. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#1594bda36f63. Accessed 22 June 2022
5. C. Stedman, E. Burns, M. K. Pratt, *What Is Data Preparation? An In-Depth Guide to Data Prep*. Teach Target. https://www.techtarget.com/searchbusinessanalytics/definition/data-preparation. Accessed 18 March 2022
6. D. Pyle, *Data Preparation for Data Mining (ITPro Collection)* (Elsevier Science, 1999)
7. M. Spruit, T. Dedding, D. Vijlbrief, Self-service data science for healthcare professionals: A data preparation approach, in *Proceedings of the 13th International Joint conference on Biomedical Engineering systems and Technologies (BICSTEC 2020) – Volume 5: HEALTHINF*, (ScitePress, Valetta, 2020), pp. 724–734
8. J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python* (Machine Learning Mastery, 2020)

9. Talend.com. *What Is Data Preparation?* Talend.com. https://www.talend.com/resources/what-is-data-preparation/. Accessed 8 May 2022
10. V. Kumar, S. Minz, Feature selection: A literature review. Smart Comput. Rev. **4**, 211–229 (2014)
11. Data Meer, *Data Preparation*. Data Meer. https://www.datameer.com/data-preparation/. Accessed 16 March 2022
12. F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, D. Turaga, Learning feature engineering for classification. 2017, pp. 2529–2535, [Online]. Available: https://doi.org/10.24963/ijcai.2017/352
13. J. Heaton, An empirical analysis of feature engineering for predictive modeling, in *SoutheastCon 2016*, 30 March–3 April 2016, pp. 1–6, https://doi.org/10.1109/SECON.2016.7506650
14. M. Kuhn, K. Johnson, *Feature Engineering and Selection: A Practical Approach for Predictive Models* (CRC Press, 2019)
15. M. Anderson, et al., Brainwash: A Data System for Feature Engineering, 21 Nov 2012
16. T. Bock, What is feature Engineering? Displayr. https://www.displayr.com/what-is-feature-engineering/. Accessed 18 March 2022
17. H. Patel, *What is Feature Engineering — Importance, Tools and Techniques for Machine Learning*. Medium. https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10. Accessed 16 March 2022
18. E. Rencberoglu, *Fundamental Techniques of Feature Engineering for Machine Learning*. Medium. https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114. Accessed 18 March 2022
19. R. Chopra, A. England, M.N. Alaudeen, *Data Science with Python: Combine Python with Machine Learning Principles to Discover Hidden Patterns in Raw Data* (Packt Publishing, 2019)
20. S. Raschka, *Python Machine Learning* (Packt Publishing, 2015)
21. J. Brownlee, *Machine Learning Mastery with Weka: Analyze Data, Develop Models, and Work Through Projects* (Machine Learning Mastery, 2016)
22. A. Burkov, *The Hundred-Page Machine Learning Book* (Andriy Burkov, 2019)
23. X. Ying, An Overview of Overfitting and its Solutions. J. Phys. Conf. Ser. **1168**, 022022 (Feb 2019). https://doi.org/10.1088/1742-6596/1168/2/022022
24. J.A. Cook, J. Ranstam, Overfitting. Br. J. Surg. **103**(13), 1814–1814 (2016). https://doi.org/10.1002/bjs.10244
25. D.M. Hawkins, The problem of overfitting. J. Chem. Inf. Comput. Sci. **44**(1), 1–12 (1 Jan 2004). https://doi.org/10.1021/ci0342472
26. A. Twin, *How Overfitting Works*. Investopedia. https://www.investopedia.com/terms/o/overfitting.asp#:~:text=Overfitting%20is%20a%20modeling%20error. Accessed 18 March 2022
27. IBM Cloud Education. *What Is Overfitting?* www.ibm.com. https://www.ibm.com/cloud/learn/overfitting. Accessed 18 March 2022
28. J. Brownlee, *Overfitting and Underfitting with Machine Learning Algorithms*. Machine Learning Mastery. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. Accessed 17 March 2022
29. H. K. Jabbar, R. Z. Khan, Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). (2014)
30. F. Sidi, P.H.S. Panah, L.S. Affendey, M.A. Jabar, H. Ibrahim, A. Mustapha, Data quality: A survey of data quality dimensions, in *2012 International Conference on Information Retrieval & Knowledge Management*, (2012), pp. 300–304
31. R.Y. Wang, D.M. Strong, Beyond accuracy: What data quality means to data consumers. J. Manag. Inf. Syst. **12**(4), 5–33 (1 March 1996). https://doi.org/10.1080/07421222.1996.11518099

32. D. Firmani, M. Mecella, M. Scannapieco, C. Batini, On the meaningfulness of "big data quality" (invited paper). Data Sci. Eng. **1**(1), 6–20 (2016). https://doi.org/10.1007/s41019-015-0004-7
33. T.C. Redman, *Data Quality for the Information Age* (Artech House, 1996)
34. P. Costabel, V. d. Carmen, Data freshness and data accuracy: A state of the art. (2006)
35. B. Shin, An exploratory investigation of system success factors in data warehousing. J. AIS **4**, 0 (1 Jan 2003). https://doi.org/10.17705/1jais.00033
36. L. Cai, Y. Zhu, The challenges of data quality and data quality assessment in the big data era. Data Sci. J. **14**, 2 (22 May 2015). https://doi.org/10.5334/dsj-2015-002
37. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly Media, 2019)

Chapter 5
Data Visualization

5.1 Introduction

Visualization via graphics like charts, graphs, and images is an effective and efficient way to interpret and understand data and help spot valuable information such as patterns, trends, and anomalies [1]. The reason is that, unlike tables and written text, graphs are primarily visual in nature, and approximately 70% of our sense receptors are dedicated to vision [2]. Moreover, our eyes are drawn to patterns and colors, can easily differentiate red from blue and a circle from a square, and can quickly see trends and outliers [3].

While the invention of data visualization may not be easily attributed to one individual, William Playfair (1759–1823) is generally viewed as the inventor of many common graphical forms, such as bar and pie charts [4]. One of his well-known visualizations is his balance of trade and chart of the national debt of England, one of the earliest line charts used to represent time series (Fig. 5.1).

Playfair was also one of the first people to use charts not only to educate but also to persuade and convince, for example, by comparing the "weekly wages of a good mechanic" and the "price of a quarter of wheat" from 1565 to 1821 (Fig. 5.2) [7].

Another classic example of an old visualization is the illustration of Napoleon's failed Russian campaign of 1812 by Charles Minard (Fig. 5.3). The graph displays the number of French soldiers marching toward and then retreating from Moscow, overlaid on top of a map. The thickness of the band is representative of the number of soldiers, which decreased as the army moved from France on the right to Russia on the left. Underneath the map is a line chart displaying the temperature that soldiers faced as they moved during the campaign.

Florence Nightingale visualized in 1858 the factors affecting the lives and death rates of the British army in a graphic known as "Nightingale's Rose" or "Nightingale's Coxcomb" (Fig. 5.4). She showed Queen Victoria in her visual graphic that it was infections (in blue) killing the highest number of soldiers and not wounds [7].

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_5

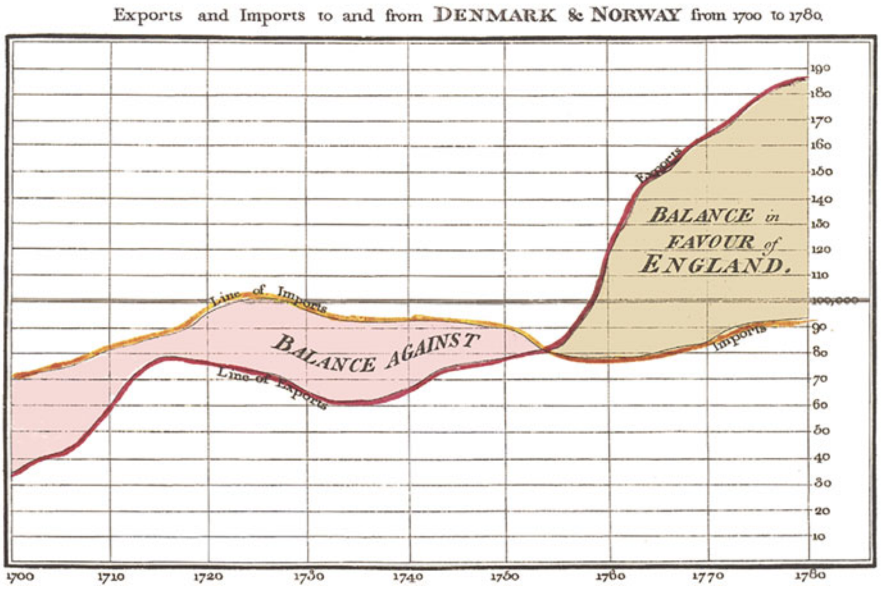


Fig. 5.1 William Playfair's balance of trade and chart of national debt of England (Source: Wikimedia [5])

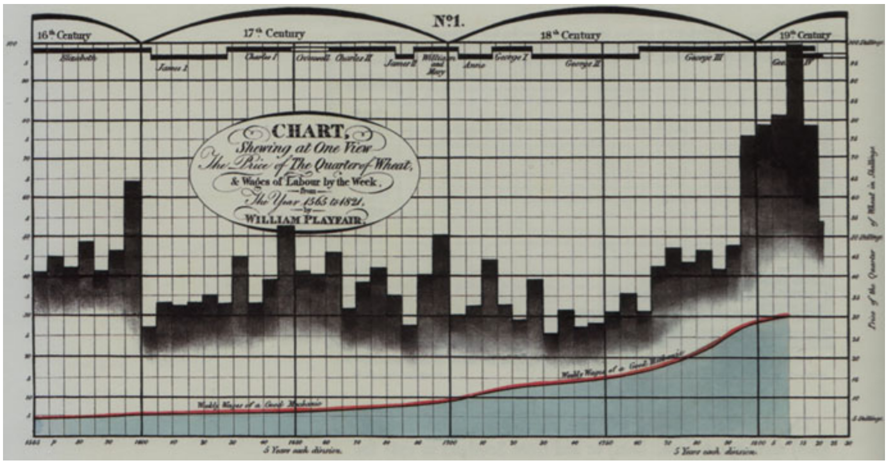


Fig. 5.2 William Playfair's 1821 chart (Source: Wikimedia [6])

In the remainder of this chapter, we introduce the basics of data visualizations, including a taxonomy of basic graphical objects and charts and their uses. We include several visualizations from different fields generated with different software

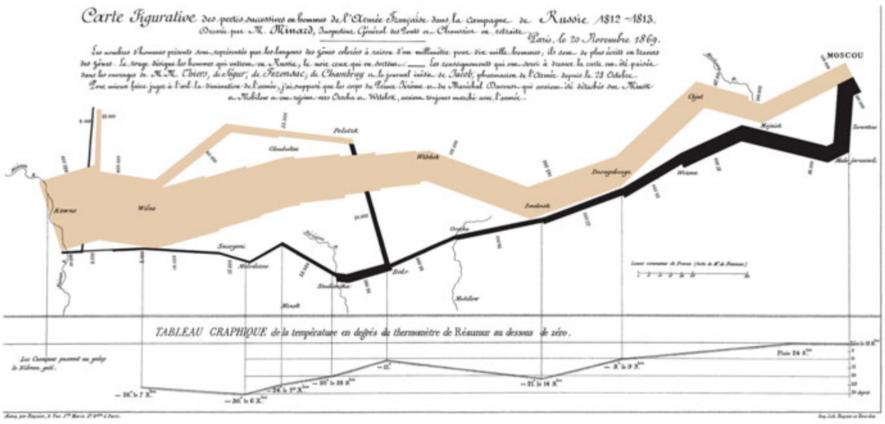
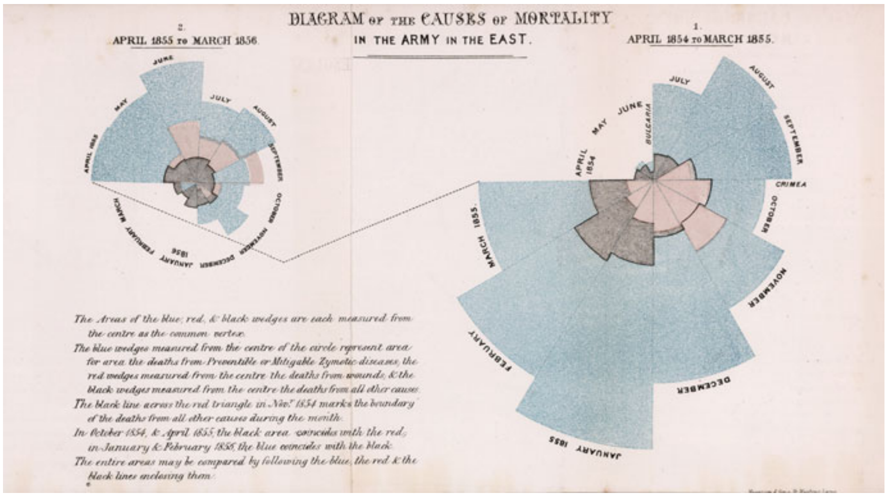


Fig. 5.3 Napoleon's failed Russian campaign of 1812 by Charles Minard (Source: Wikimedia [8])



The Areas of the blue, red, & black wedges are each measured from the centre as the common vertex.
The blue wedges measured from the centre of the circle represent area for area the deaths from Preventible or Mitigable Zymotic diseases, the red wedges measured from the centre the deaths from wounds, & the black wedges measured from the centre the deaths from all other causes.
The black line across the red triangle in Nov.r 1854 marks the boundary of the deaths from all other causes during the month.
In October 1854, & April 1855, the black area coincides with the red, in January & February 1856, the blue coincides with the black.
The entire areas may be compared by following the blue, the red & the black lines enclosing them.

Fig. 5.4 Florence Nightingale's 1858 diagram of the causes of mortality in the army in the East (Source: Wikimedia [9])

packages using different sources of open data. We also cover infographics and dashboards, which are visualization-rich tools that are increasingly being used in many industries. We finish with guidelines for building good visualizations.

5.2 Presentation and Visualization of Information

The type of data sometimes dictates the type of graph that can or cannot be used. As a quick reminder, data are broken into two types: quantitative and categorical. Quantitative values measure things and consist of a quantity and unit of measure (e.g., 300 km). Categorical data divide information into useful groups and are nominal (e.g., fall, winter, spring, summer), ordinal (e.g., low, medium, high), interval (e.g., 0–9, 10–19, . . .), and hierarchical (e.g., year, quarter, month, week, day). In many cases, the type of data dictates the type of graph and visualization to be used.

5.2.1 A Taxonomy of Graphs

Of the different available taxonomies of graphs, we will follow the one proposed by Stephen Few [2], a well-known expert in data visualization. According to Few, quantitative data can be basically represented in graphs by the following six basic objects: points, lines, bars, boxes, shapes with varying 2D areas, and shapes with varying color intensity.

A *point* is a simple dot on a graph representing two values, one on each axis, and a graph consisting of such points is referred to as a scatterplot (Figs. 5.5 and 5.6). Scatterplots, which are representations of many distinct data points on a single chart, give a general idea about the distribution of the data and are useful in highlighting relationships between different variables, showing if the two variables tend to vary independently or not. Scatterplots are also useful in showing correlations and in detecting data outliers [1, 2, 11].

A *line* connects a series of values or distinct points in a graph and is a good representation of how values change or evolve over time (called a time series) [1]. Line charts are used to view trends and cycles in data, usually over time or other ordinal data (Figs. 5.7 and 5.8) [11].

A *bar*, one of the most common types of data visualizations, is a rectangle that encodes quantitative information by its length (Fig. 5.9). Bars are easy to see and compare and should always begin at the value of 0 [2]. Bar charts are used to quickly compare data across categories, show trends and outliers, and highlight differences at a glance. Bar charts are especially effective when data can be split into multiple categories [11]. Graphs composed of vertical bars are referred to as column charts.

A *box* is also rectangular but encodes a wide range of values, such as the minimum, maximum, and median values (Fig. 5.10). Graphs of such boxes are referred to as box-and-whisker plots or boxplots and are used to show distributions of data. Typically, the box contains the median of the data, the first quartile (25% less than the median), the third quartile (25% greater than the median), and the whiskers represent data within 1.5 times the interquartile range (i.e., the range between the first and third quartiles). The whiskers can also be used to show the maximum and minimum points within the data [11]. Boxplots are often used to compare the distribution of different datasets [2].

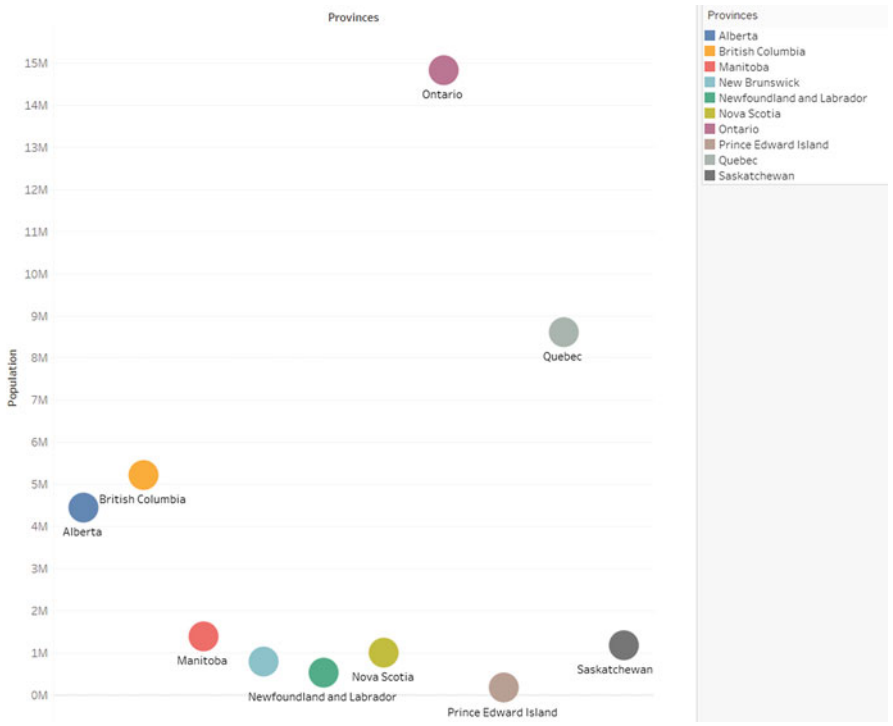


Fig. 5.5 This simple example of a *scatterplot* shows the 2021 population estimate of Canadian provinces. With this scatterplot, the significantly large size of the population in Ontario is immediately evident, as is the very low population of Prince Edward Island. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

Shapes with 2D areas represent values in proportion to their area rather than their location on the graph. A popular example is the *pie chart* (Fig. 5.11), where each sector of the pie represents a percentage of the whole. However, despite its frequent use, a pie chart is not recommended when the compared values are close or when there are many categories or sectors to compare [1, 2].

Another example of shapes with 2D areas is the *bubble,* which is a scatterplot that quantifies three values, two by their relative location on each axis and the third by the size of the bubble. A fourth variable can be quantified by applying *variable intensities of the same color* to the bubbles [2] or simply by using different colors (Fig. 5.12).

5.2.2 *Relationships and Graphs*

Graphs are used to display relationships in data by giving them shapes. There are eight main types of relationship graphs that are typically used: time series, ranking, part-to-whole, deviation, distribution, correlation, geospatial, and nominal comparison [2]. *Time series* graphs show how something changed (increased, fluctuated,

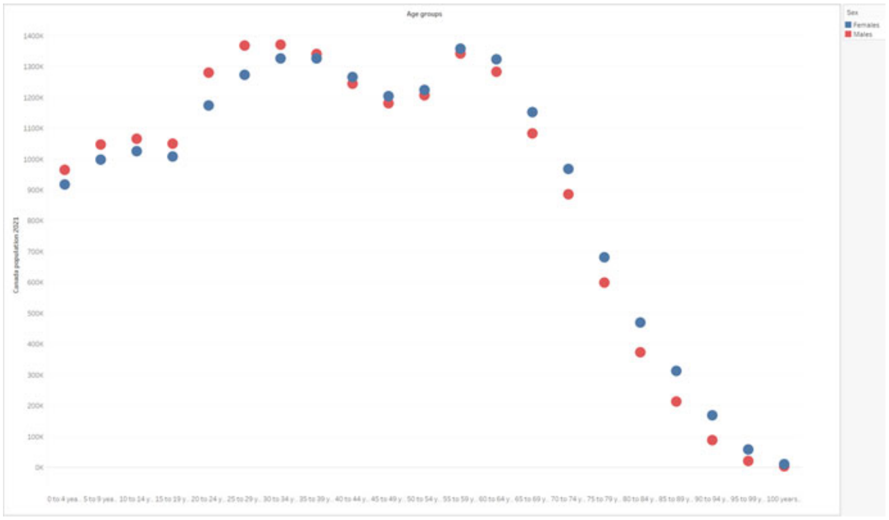


Fig. 5.6 This example of a *scatterplot* shows the population of Canada in 2021 by age group and by sex. We notice that until the age of 40, the number of males exceeds the number of females and that this is reversed afterward. We also see a dip in the population aged around 40–60 years. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

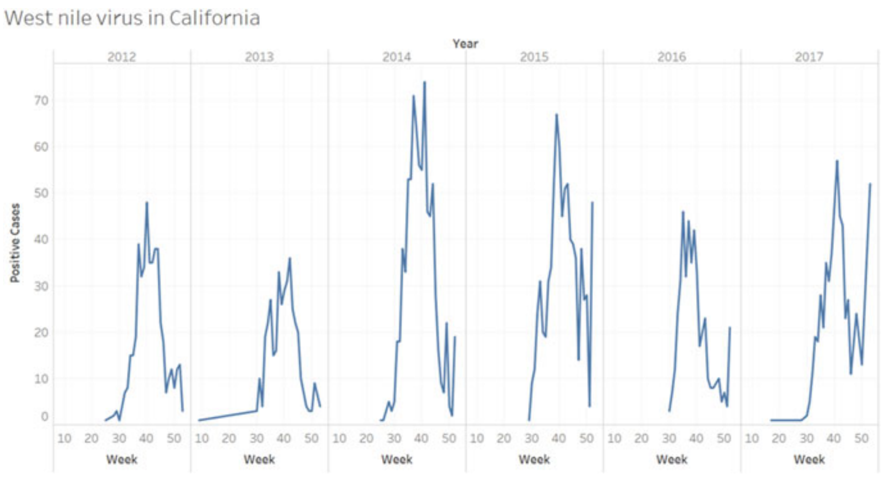


Fig. 5.7 This example of a *line chart* or time series displays the incidence of the West Nile virus in California on a weekly basis from 2012 to 2015. It clearly shows a yearly cycle where the incidence peaks between weeks 39 and 42. This graph was generated using Tableau Desktop software and the California Department of Public Health Open Data [12]

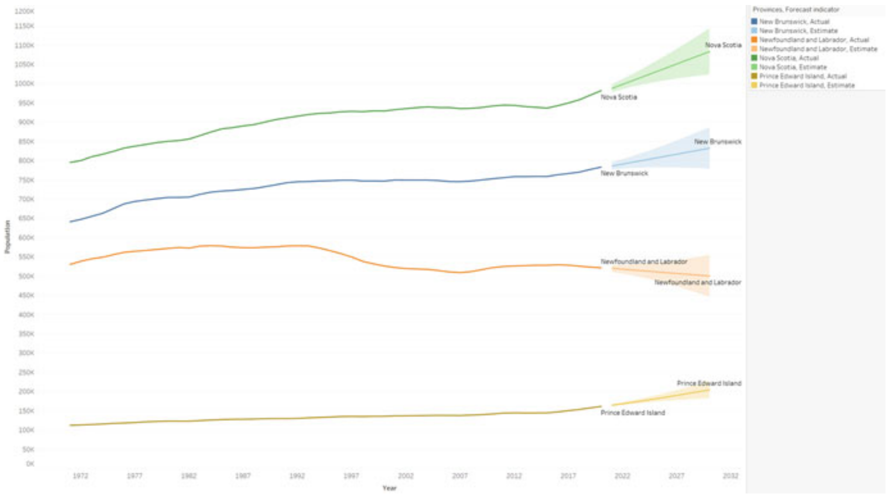


Fig. 5.8 This example of a *line chart* highlights the trend of slow population increase in three Canadian Maritime Provinces, and a decreasing trend in the fourth, from 1972 to 2021. The software generated a forecast statistic until 2032. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

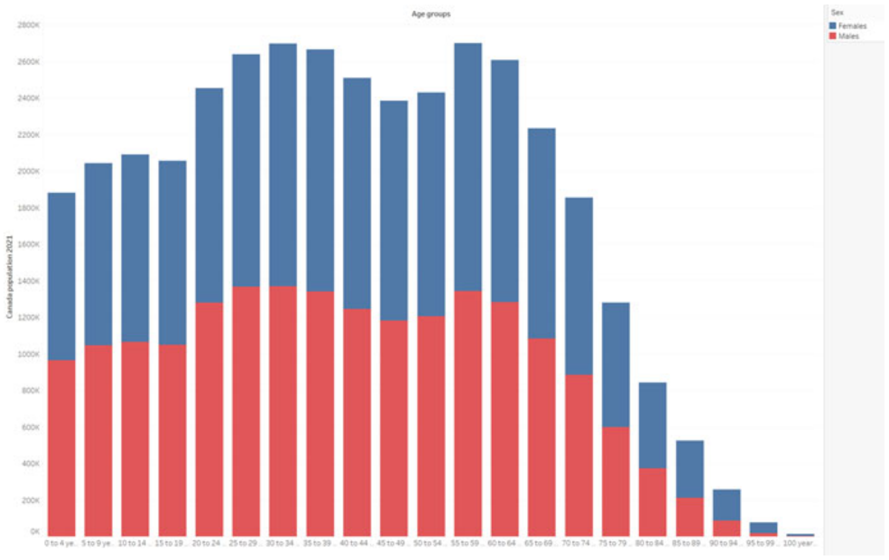


Fig. 5.9 This example of a *bar graph*, also called a column chart, or a stacked bar chart, shows the population of Canada in 2021 by age group and by sex. It is a different visualization of the same data in Fig. 5.6. This graph was generated using Tableau Desktop software Statistics Canada population open data [10]

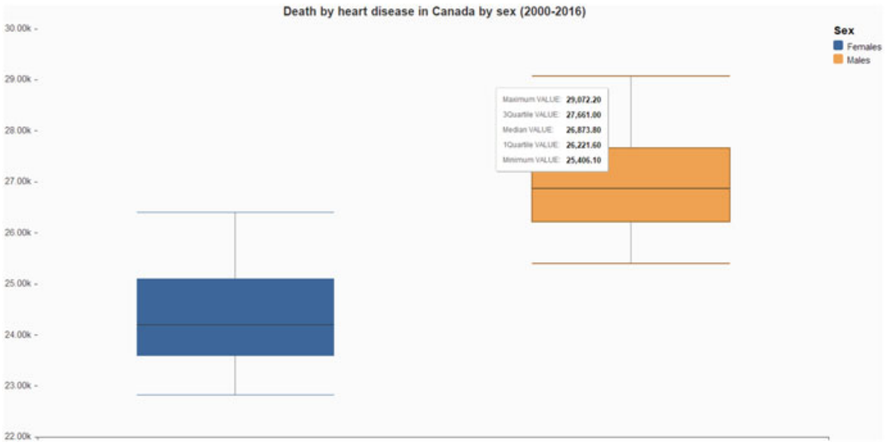


Fig. 5.10 This example of a box graph, also known as a *box plot*, represents death by heart disease by sex in Canada (2000–2016). Each box displays the minimum, first quartile, median, third quartile, and maximum values in the dataset. This graph was generated using SAP Lumira software and Statistics Canada leading cause of death open data [13]

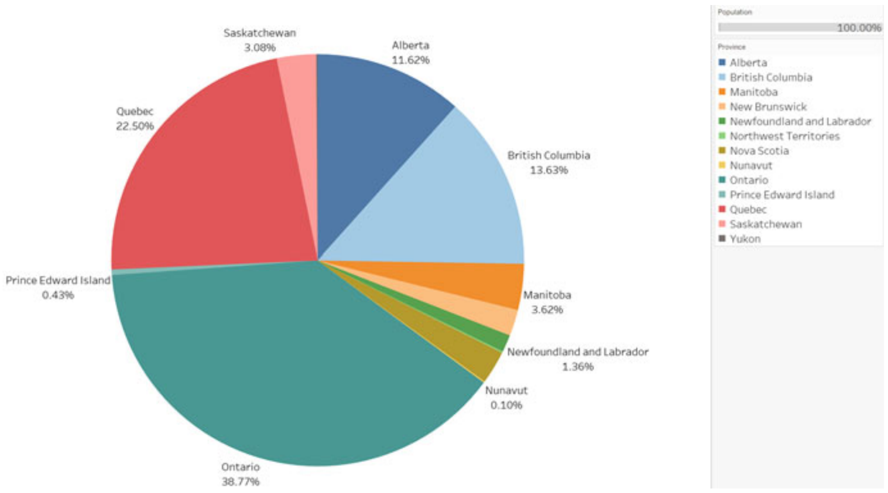


Fig. 5.11 This example of a *pie chart* depicts the distribution of the Canadian population by province, clearly highlighting the relative population size in each. It is a different visualization of the same data in Fig. 5.5. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

declined, etc.) over time (e.g., Figs. 5.7 and 5.8). Graphs display *ranking* relationships such as larger than, smaller than, and equal to, sorted in increasing or decreasing order (Fig. 5.9, though, is not sorted). Graphs display *part-to-whole*

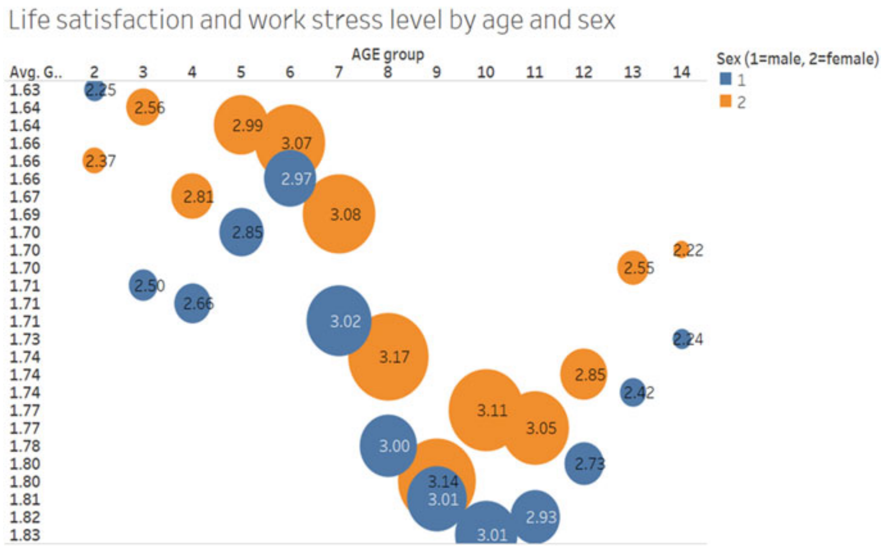


Fig. 5.12 This example of a *bubble chart* displays the average life satisfaction level (*y*-axis) and work stress level (bubble size), by age group (*x*-axis) and sex (color) in Canada (2012). Among the visual observations are that after a certain age, females have a lower life satisfaction and higher stress level, but that there is no clear relationship between life satisfaction and work stress in general. This graph was generated using Tableau Desktop software and Statistics Canada Community Health open data [14]

relationships by showing how individual values make up the whole of something (for example, by percentage or rate of total) and how they compare to each other (Figs. 5.9 and 5.11). *Deviations* represent how one or more sets of values differ from a reference set of values (Fig. 5.13) [2].

A *distribution* represents how values are distributed across an entire range, from the lowest to the highest, and is called a *frequency distribution* when it shows the number of times something occurs. When bars are used, it is referred to as a histogram (Fig. 5.14) [2]. Histograms group the data into specific categories known as bins and assign a bar size that is proportional to the number of records in each bin [11].

A graph displays a *correlation* when it shows whether two sets of values vary (increase, decrease, follow) in relation to each other, positively or negatively, and to what degree (e.g., Figs. 5.6, 5.9, and 5.12). G*eospatial* relationships between values are displayed by plotting them on a map (Fig. 5.15). Finally, a *nominal comparison* is the simple display of a set of discrete quantitative values so that they can be easily read and compared (e.g., Fig. 5.13) [2].

To display a specific relationship graphically, different objects and types of graphs can be used, with some being more adequate for the task than others, while others should be avoided. Table 5.1 is a summary of the recommended graphical objects used to display each type of relationship described above.

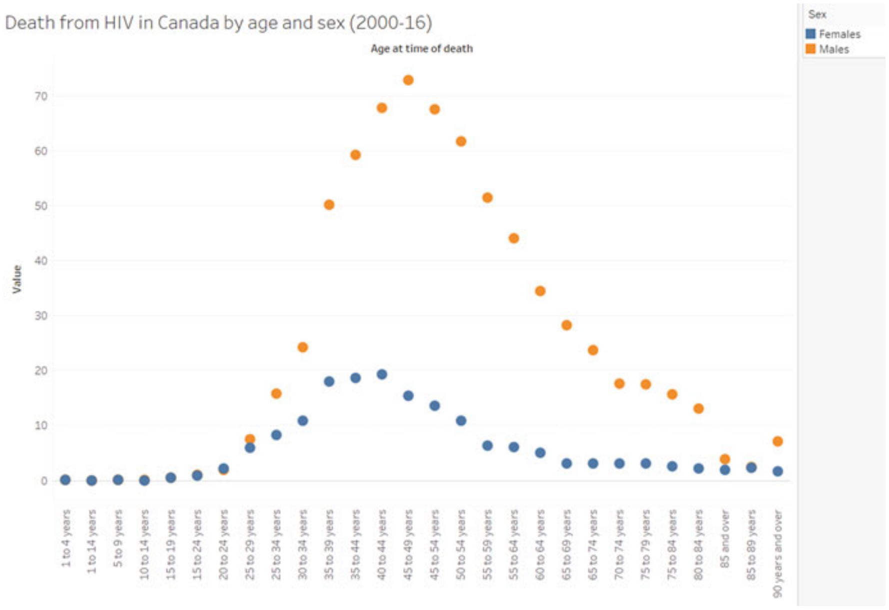


Fig. 5.13 This graph represents the number of deaths per 100,000 citizens from HIV in Canada by age group and sex (2000–2016). It shows a significant and very clear *deviation* for males, compared to females, after the age of 25. This graph was generated using Tableau Desktop software and Statistics Canada leading cause of death open data [13]

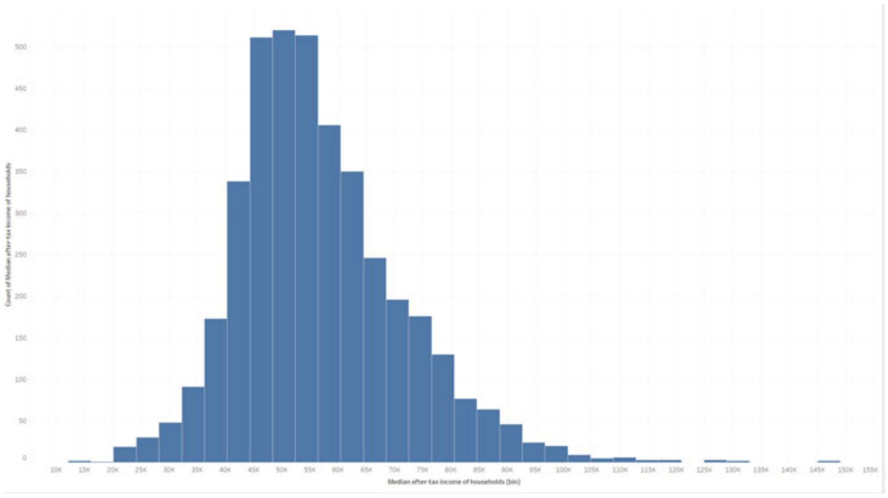


Fig. 5.14 This example of a *histogram* displays the *distribution* of households' median after-tax income (2016). This graph was generated using Tableau Desktop software and Statistics Canada 2016 Census open data [15]

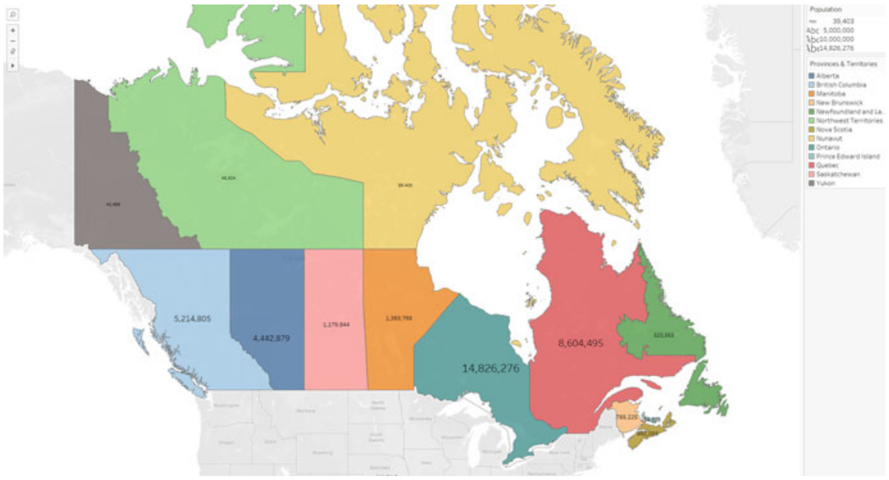


Fig. 5.15 This example of a *geospatial map* displays the population by province in Canada, where a larger font indicates a larger population in the different Canadian provinces and territories in 2016. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

In addition to the visualizations shown so far, there is a large number of possible visualizations, many of which are very popular and useful. Below are some additional examples of popular or interesting advanced visualizations (Figs. 5.16, 5.17, 5.18, 5.19, 5.20 and 5.21).

There are infinite additional ways to visualize data and information. What has been covered so far in this chapter is an introduction to the basic and most popular visualizations. To view additional examples of interesting and rich visualizations, you can explore numerous sources such as the Information Is Beautiful website (https://informationisbeautiful.net/), the Data Visualization Catalogue (https://datavizcatalogue.com/), and Tableau's public gallery (https://public.tableau.com/en-us/s/gallery).

A popular guide for selecting a chart based on the number of variables, the kind of comparison needed, and the time frame can be found at the Extreme Presentation website (https://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html).

A number of excellent interactive visualizations can be found at different websites, such as Statistics Canada's Interact with Data site (https://www.statcan.gc.ca/en/interact). Two examples can be found in Figs. 5.22 and 5.23. Another good example is the website of the Institute for Health Metrics and Evaluation (IHME), which is an independent global health research center at the University of Washington (http://www.healthdata.org/results/data-visualizations). Most visualizations are interactive with filters that allow the viewer to select from a variety of graph

Table 5.1 Graphical object to use for each type of relationship (adapted from Few (2012) [2])

	Graphical Objects			
Relationship	Points	Lines	Bars	Boxes
Time series (categorical data on the *x*-axis and quantitative values on the *y*-axis)	Dot plot only when values were not collected at consistent intervals of time	For emphasis on the overall pattern (e.g., Figs. 5.7 and 5.8)	For emphasis on individual values	Only when showing distributions that change over time
Ranking	Dot plot when the quantitative scale does not start at 0; otherwise, use bar	Avoid	Horizontal or vertical (e.g., Fig. 5.9, but preferably sorted)	Only when ranking multiple distributions; horizontal or vertical
Part-to-whole	Avoid	To display how parts of a whole change over time	Horizontal or vertical (e.g., Fig. 5.9)	Avoid
Deviation	Dot plot when the quantitative scale does not start at 0	Useful when combined with time series	Horizontal or vertical; always vertical if combined with time series	Avoid
Distribution (single)	Known as a strip plot; emphasis on individual values	Known as a frequency polygon; emphasis on overall pattern	Known as a histogram; emphasis on individual intervals (e.g., Fig. 5.14)	Avoid
Distribution (multiple)		Known as a frequency polygon. Limit to a few lines	Avoid	Known as a box plot (e.g., Fig. 5.10)
Correlation	Known as a scatterplot (e.g., Fig. 5.6)	Avoid	Horizontal or vertical	Avoid
Geospatial	Different point sizes encode values (e.g., Fig. 5.16)	Used to mark routes (e.g., Fig. 5.16)	Avoid	Avoid
Nominal comparison	Use a dot plot if scales do not start at 0	Avoid	Horizontal or vertical; use when the scale starts at 0 (e.g., Fig. 5.9)	Avoid

types, regions, dates, measures, indicators, etc. Some are also dynamic and display data evolving over a period of time. Another very common type of interactive visualization is the dashboard, which is introduced next.

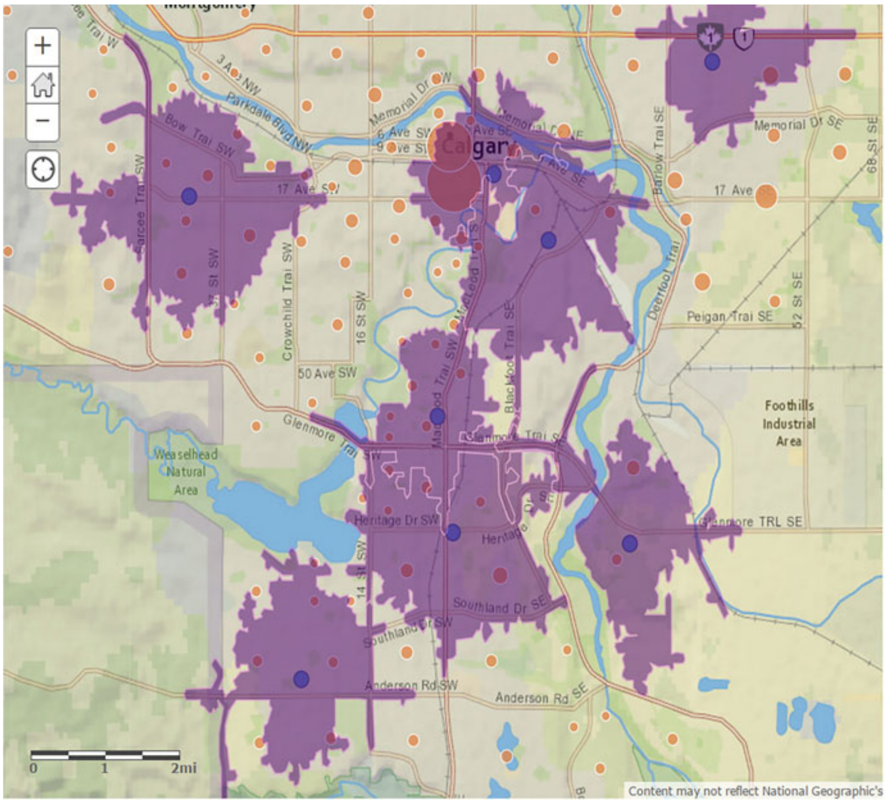


Fig. 5.16 This example of a *geospatial graph* shows the typical distance that can be reached in 5 minutes from the police stations in the city of Calgary in Canada (blue dots). The orange circles in different sizes represent the relative crime rate in the different municipalities of the city. This graph was generated using the ESRI ArcGIS Online tool [16] and the city of Calgary open data [17]

5.2.3 Dashboards

The dashboard is "a visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so the information can be monitored at a glance" [20]. It is a collection of related visualizations that are tied together through interactivity and are displayed on a single page and can combine multiple different types of data in a single location [21]. Dashboards can be used, for example, to monitor marketing campaigns' landing pages, conversion rates, visitors by location, lead by campaign source, and other key performance indicators (KPIs) (Fig. 5.24). This interactive analytical dashboard allows the user to select dates and regions for analysis. Another example is an e-commerce dashboard (Fig. 5.25), which can be used to analyze and monitor sales and revenue from different perspectives. More examples of dashboards from different industries and

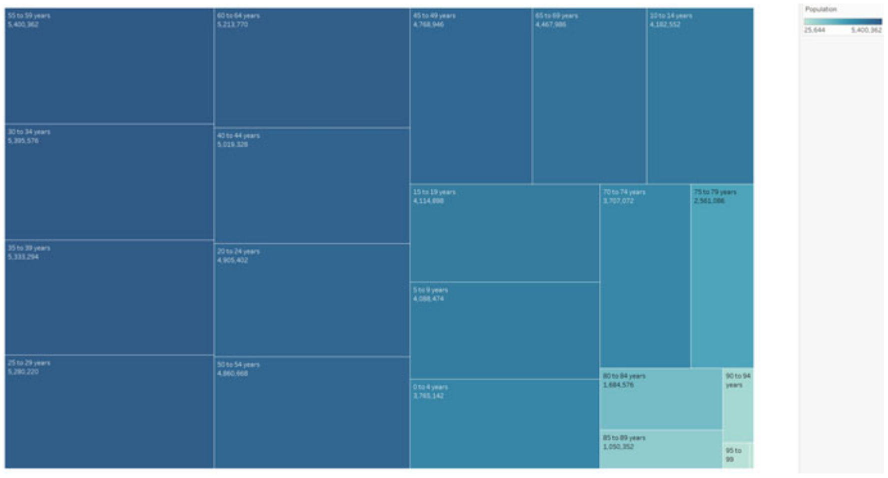


Fig. 5.17 This is an example of a *heat map*, displaying the distribution of the Canadian population by age group (2021). The larger the rectangle and the darker (or "hotter") the color, the higher the population. We see here that the 55–59 years old is the largest group in Canada. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

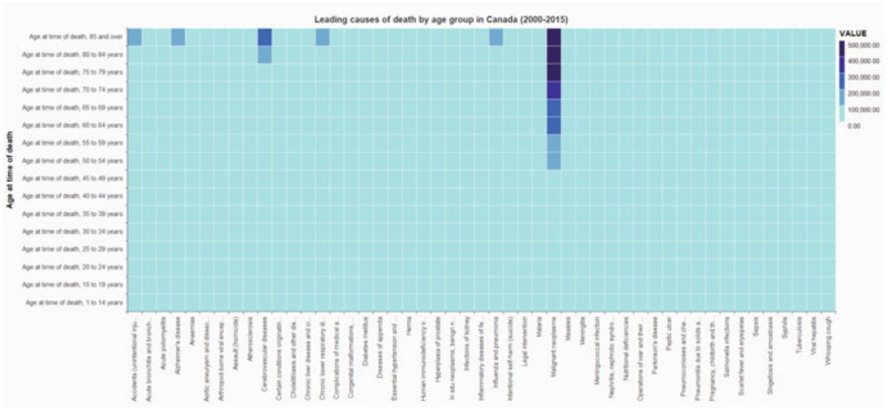


Fig. 5.18 This is an example of a *heat map* displaying the leading causes of death by age in Canada (2000–2015). The darker the color, the higher the total number of deaths. The main difference between a heat map and a treemap is that the latter can enable a hierarchical presentation of additional variables [1]. This graph was generated using SAP Lumira software and Statistics Canada leading cause of death open data [13]

job functions can be found on the websites of business intelligence companies such as Qlik (https://www.qlik.com/us/dashboard-examples) and Sisense (https://www.sisense.com/dashboard-examples/).

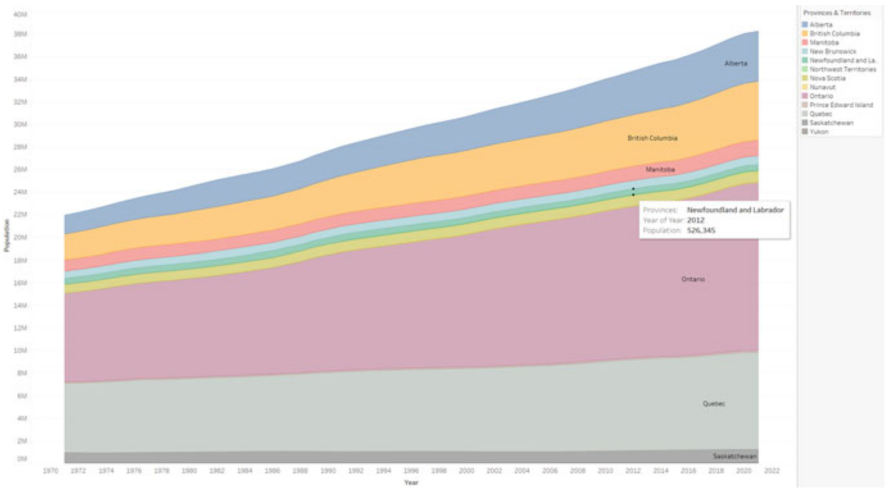


Fig. 5.19 This example of an *area chart* displays the growth of the population by province in Canada between 1971 and 2021. This graph was generated using Tableau Desktop software and Statistics Canada population open data [10]

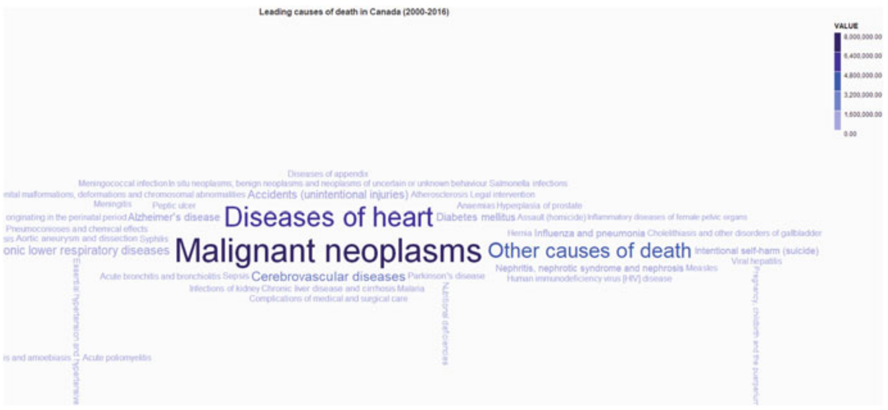


Fig. 5.20 This is an example of a *tag cloud* used to display the leading causes of death in Canada (2000–2015). The larger the text and the darker its color, the higher the total number of incidents. Tag clouds are useful for displaying words or phrases based on their frequency and hence importance [1]. This graph was generated using SAP Lumira software and Statistics Canada leading cause of death open data [13]

Dashboards can be broken down into three roles: strategic, analytical, and operational. At the executive level of an organization, dashboards support long-term strategic decisions and focus on high-level measures of performance, including forecasts. They tend to be simple and not interactive and do not require real-time data updates. Dashboards that support data analysis demand rich comparisons, more extensive history, and interaction with data, such as drilling down for more details.

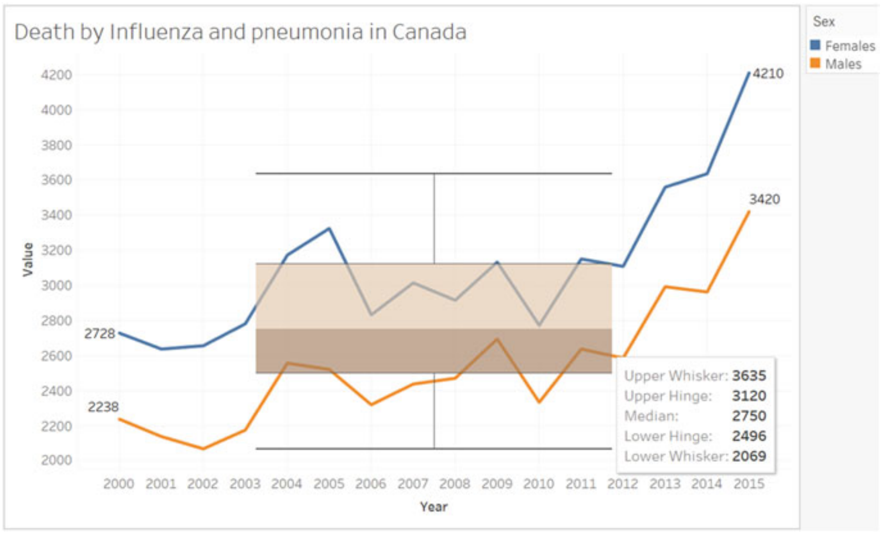


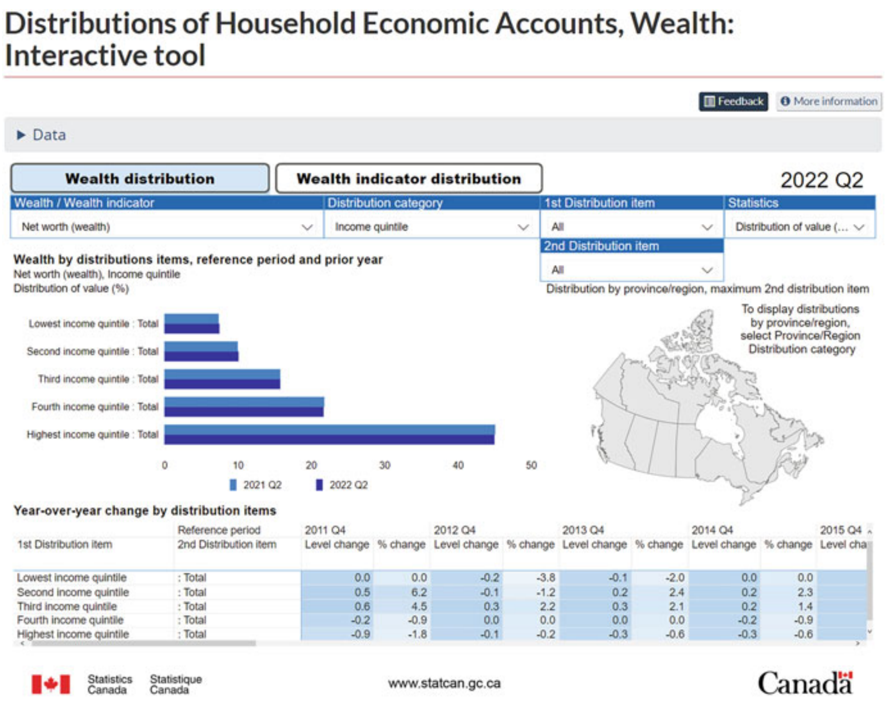
Fig. 5.21 This is an example of a chart combining different graphical objects. It displays via lines the number of deaths by influenza and pneumonia by sex from 2000 to 2015. It also includes a box plot displaying the upper and lower whiskers, median, and upper and lower hinges. This graph was generated using Tableau Desktop software and Statistics Canada leading cause of death open data [13]

They can help detect patterns in the data to identify the causes of problems, for example. Similar to strategic dashboards, analytical dashboards work with static, not real-time, data. Finally, operational dashboards are dynamic and immediate in their nature. They present real-time data in a simple way but also have the means to attract attention in cases when an operation falls outside the range of the acceptable threshold of performance [20].

A good dashboard should be designed with the most important view in the top left corner, not include too many views, use a consistent color scheme or compatible ones, and have all the filters grouped together [21].

5.2.4 *Infographics*

The term "infographics" is an abbreviation of "information graphics." They are a combination of data visualizations, text, and images, presented in a logical manner similar to storytelling, and are used to convey information and messages in an attractive and easy-to-understand format [1, 24]. Infographics use many different visual cues to convey information. With the overwhelming amount of data and content generated and shared online, infographics have become very important due to their ability to present information to an audience in a way that can capture



Distributions of Household Economic Accounts, Wealth: Interactive tool

Feedback | More information

▶ Data

Wealth distribution | Wealth indicator distribution | 2022 Q2

Wealth / Wealth indicator	Distribution category	1st Distribution item	Statistics
Net worth (wealth)	Income quintile	All	Distribution of value (...

2nd Distribution item: All

Wealth by distributions items, reference period and prior year
Net worth (wealth), Income quintile
Distribution of value (%)

Lowest income quintile : Total
Second income quintile : Total
Third income quintile : Total
Fourth income quintile : Total
Highest income quintile : Total
0
10
20
30
40
50
2021 Q2
2022 Q2
Distribution by province/region, maximum 2nd distribution item
To display distributions by province/region, select Province/Region Distribution category

Year-over-year change by distribution items

	Reference period	2011 Q4		2012 Q4		2013 Q4		2014 Q4		2015 Q4
1st Distribution item	2nd Distribution item	Level change	% change	Level change	% change	Level change	% change	Level change	% change	Level cha
Lowest income quintile	: Total	0.0	0.0	-0.2	-3.8	-0.1	-2.0	0.0	0.0	
Second income quintile	: Total	0.5	6.2	-0.1	-1.2	0.2	2.4	0.2	2.3	
Third income quintile	: Total	0.6	4.5	0.3	2.2	0.3	2.1	0.2	1.4	
Fourth income quintile	: Total	-0.2	-0.9	0.0	0.0	0.0	0.0	-0.2	-0.9	
Highest income quintile	: Total	-0.9	-1.8	-0.1	-0.2	-0.3	-0.6	-0.3	-0.6	

Statistics Canada Statistique Canada www.statcan.gc.ca Canada

Fig. 5.22 This is an example of an *interactive visualization* consisting of a table, bar chart, and map representing the distribution of household wealth in Canada. The site allows users to select wealth indicator, distribution, and statistics. The graph was generated on Statistics Canada's Interact with Data—Data Visualization site [18]

and keep the audience's attention, engage them, and aid in their comprehension and retention of the material [25]. Infographics are used in multiple disciplines, such as public policy, journalism, business, and politics. In the healthcare field, infographics are used for health communication and engagement, particularly to support comprehension among individuals with low health literacy [26]. They are helpful tools for communicating key messages clearly, challenging people's thinking, and changing behaviors and attitudes [24].

Figure 5.26 shows two examples of infographics by Statistics Canada. The first one was created to inform the public of its new 24-hour movement guidelines for children and youth [27]. It includes general information, basic statistics about the current physical activity levels in the country, and the factors that can increase them, all in a simple, clear, easy-to-understand, and visually stimulating fashion for both parents and young people. The second example is an infographic on rail transportation in Canada in 2020 [28]. Its role is to inform the public of the expenses and revenues, the origin and destination of shipments, and the volume and type of products in a simple and clear way that is easy to understand.

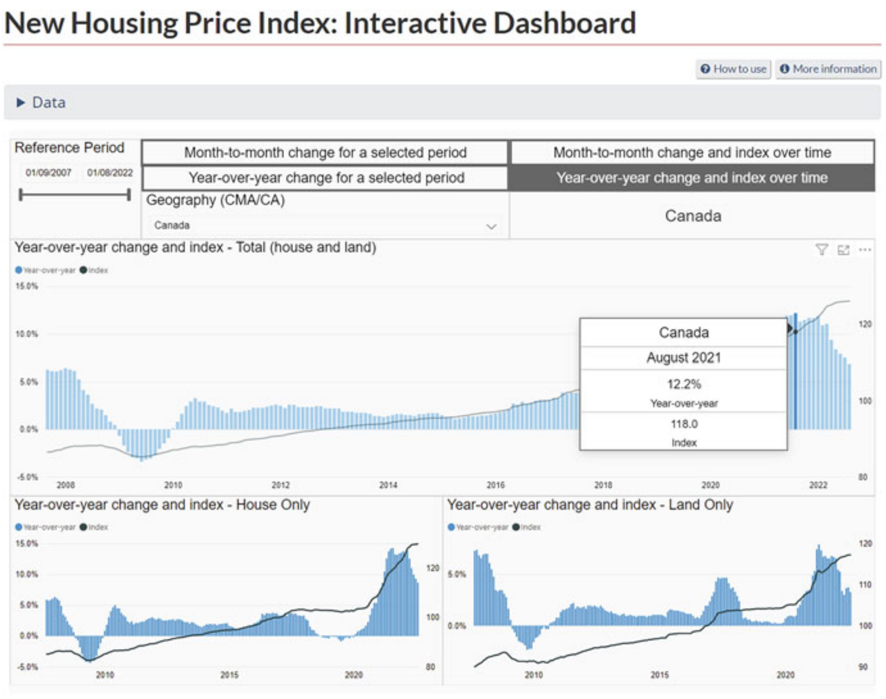
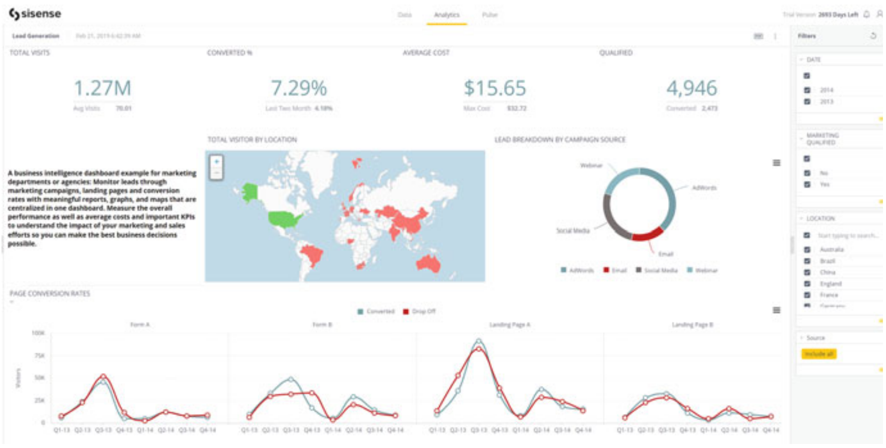

New Housing Price Index: Interactive Dashboard
How to use
More information
Data
Reference Period
01/09/2007
01/08/2022
Month-to-month change for a selected period
Month-to-month change and index over time
Year-over-year change for a selected period
Year-over-year change and index over time
Geography (CMA/CA)
Canada
Canada
Year-over-year change and index - Total (house and land)
Year-over-year
Index
Canada
August 2021
12.2%
Year-over-year
118.0
Index
Year-over-year change and index - House Only
Year-over-year change and index - Land Only

Fig. 5.23 This is an example of *interactive visualizations* consisting of combined bar (change) and line (index) charts representing the new housing price index for houses and land in Canada. The site allows users to select the reference period, region, and type of change. The graph was generated on Statistics Canada's Interact with Data—Data Visualization site [19]



A business intelligence dashboard example for marketing departments or agencies: Monitor leads through marketing campaigns, landing pages and conversion rates with meaningful reports, graphs, and maps that are centralized in one dashboard. Measure the overall performance as well as average costs and important KPIs to understand the impact of your marketing and sales efforts so you can make the best business decisions possible.

Fig. 5.24 Example of a lead generation (marketing) dashboard (Source: Sisense.com with permission [22])

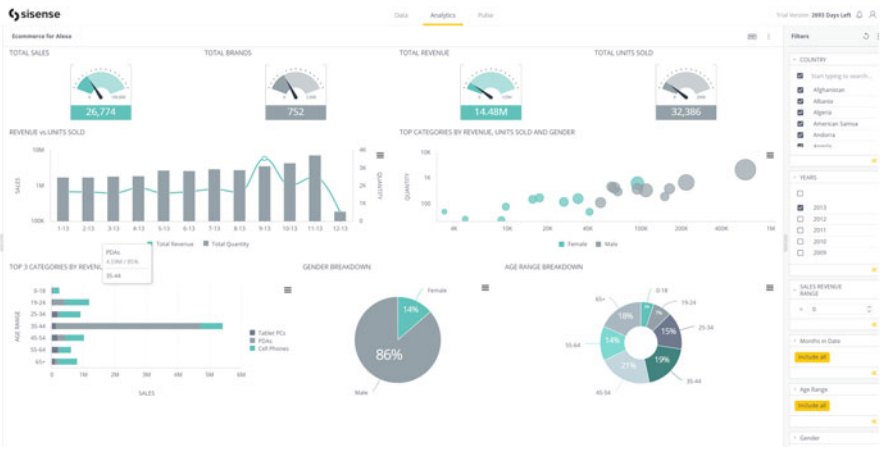


Fig. 5.25 Example of an e-commerce dashboard (Source: Sisense.com with permission [23])

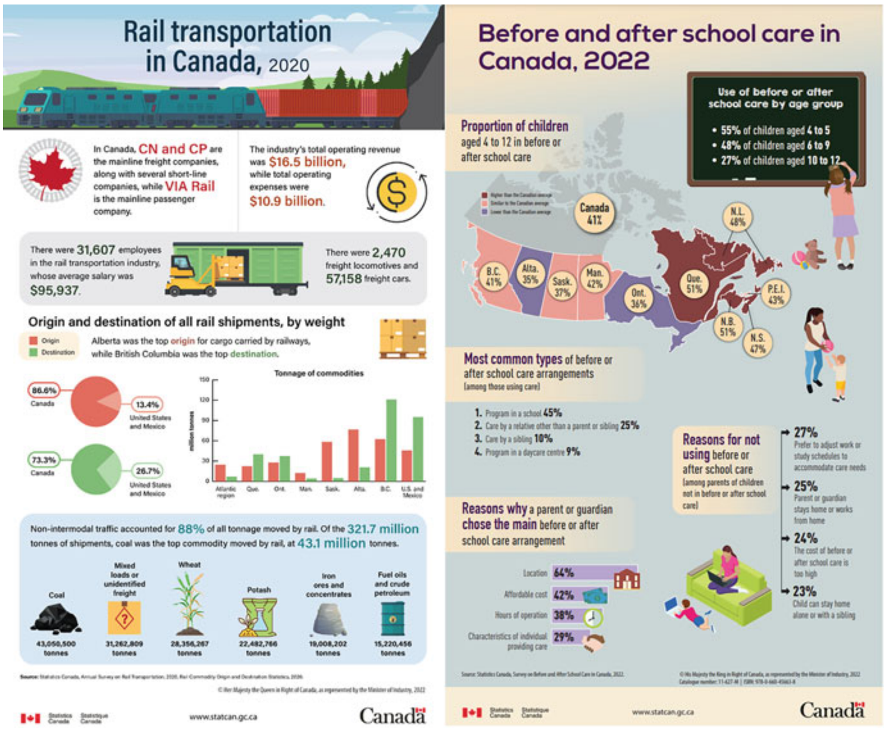


Fig. 5.26 Examples of infographics (Source: Statistics Canada [27, 28])

In general, infographics are designed without complex terminology, allowing the public to understand the message without explanation from professionals [24]. A study on the design of health-related infographics for engaging community members with varying levels of health literacy found that successful designs are rich in information but without distracting details. They support comparison, between treatments, for example, with a clear recommendation. They provide valuable contextual information and use familiar colors and symbolic analogies such as the battery charging level to represent a patient's sleep and energy levels [26].

5.3 Building Effective Visualizations

Clear communication of quantitative information is the essence of a graph. Six principles, known as ACCENT, are at the basis of effective visual display of data [4]:

- Apprehension: the ability to correctly recognize relationships between variables
- Clarity: the ability to visually differentiate the different elements of a graph
- Consistency: the ability to build your understanding of a graph on similarities with previous ones
- Efficiency: the ability to identify a complex relationship in a simple manner
- Necessity: the need for the graph
- Truthfulness: the ability to determine the true value represented by the graph

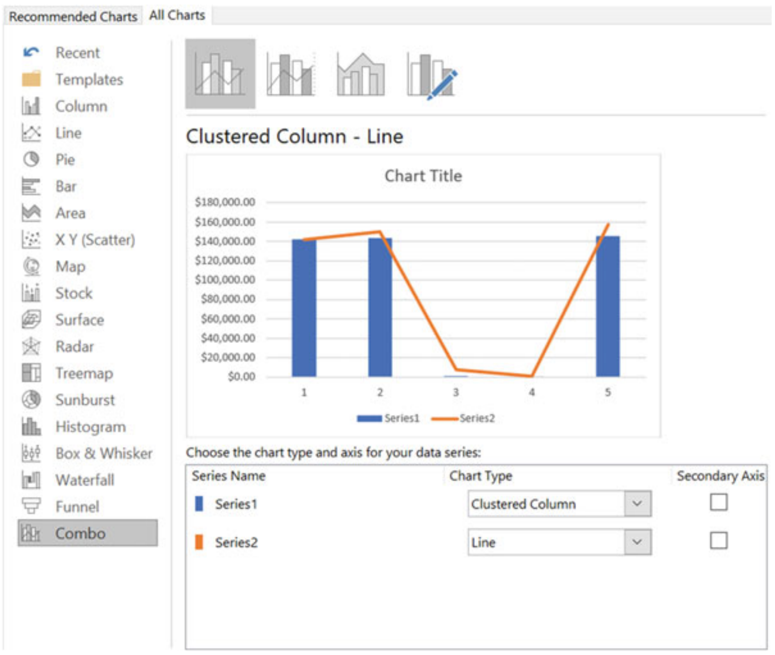
It is important to remember that despite the richness of graphs, sometimes the use of a simple table is more efficient and effective in achieving the goal of data interpretation. Tables are recommended when you want to look up individual values, compare individual values, use precise values, or use summary and detailed information in a single display [2]. Graphs are best used when the message is contained in the shape of the values, such as trends and patterns, and to reveal relationships between sets of values [2].

While graphs and charts are excellent tools for conveying information and telling stories, you should be aware of many bad visualizations that are encountered regularly. Among the sources of deficiency is the use of certain types of popular graphs that should be avoided and the bad design of appropriate graphs. Among the charts to avoid are donut charts, radar charts, circle charts, funnel charts, 3D charts, and in some cases, pie charts [2, 29]. These somewhat popular charts are visually appealing and are available in many software packages, but they fail to present information accurately, clearly, accessibly, and efficiently [2]. In terms of design, one of the most important guidelines is to avoid clutter, which includes visual elements that use space and do not increase our understanding, which increases our cognitive load or the mental effort required to learn new information [29]. Examples of clutter are too many elements, such as lines and bars, axes, data labels, colors, and text. In addition to simplicity, the data need to be put in context, so the reader can understand its meaning. The numbers that give a more faithful representation, be it percentage change or absolute value, should be used. Color and fonts should convey information and not be used for decoration. Natural increments for the *y*-axis scale are required, as is a zero baseline in all bar charts. Finally, it is best to use as few graphical elements as possible to keep the visualization crisp and clean [30].

5.4 Data Visualization Software

The creation of appealing and beautiful visualizations can be achieved with a very large number of software tools, starting with the common Microsoft Excel. Today, Excel can create many different basic charts and graphs, such as columns and lines, and complex charts and graphs, such as treemaps and waterfalls. It can also create combinations of charts, such as clustered columns and lines (Fig. 5.27). For the novice user, Excel recommends the most appropriate charts to use based on the data selected in the spreadsheet.

Business analytics tools such as Tableau Desktop, Microsoft Power BI, and Lumira by SAP provide a very large number of visualizations that are easy to use and do not require advanced technical knowledge. Most of the work done is via simple pointing, clicking, and dragging with the mouse. These advanced analytics tools can also interpret the data to identify dimensions and measures, which are comparable to the categorical and quantitative data discussed earlier in the book. The



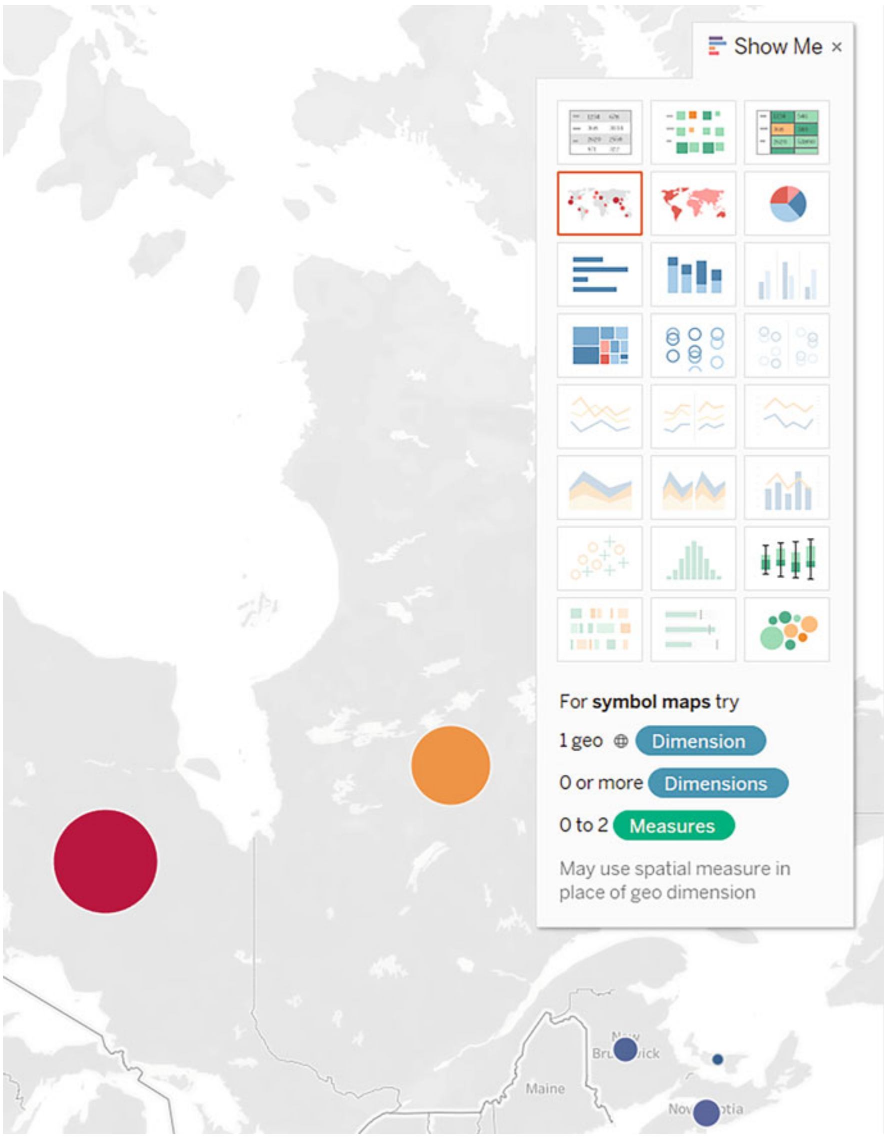
Clustered Column - Line

Choose the chart type and axis for your data series:

Series Name	Chart Type	Secondary Axis
Series1	Clustered Column	☐
Series2	Line	☐

Fig. 5.27 Example of a visualization with Microsoft Excel

charts to use are recommended based on the data available. Tableau Desktop, for example, has a Show Me button (Fig. 5.28) that highlights the available charts that can be used based on the data and makes suggestions for using them. Such tools also allow you to easily create presentations, infographics combining different charts, and dashboards connected to dynamic data sources.



For **symbol maps** try

1 geo ⊕ Dimension

0 or more Dimensions

0 to 2 Measures

May use spatial measure in place of geo dimension

Fig. 5.28 Tableau analytics tool interface with the Show Me button

While we have discussed Tableau, SAP Lumira, and Microsoft Excel in this chapter and used them to generate the visualizations above, it is important to note that there are many large software companies, such as SAS and IBM, and relatively smaller niche players, such as QlikView and Sisense, that have software with remarkable visualization capabilities. According to Gartner's 2022 Magic Quadrant, the leading platforms in analytics and business intelligence are Microsoft, Salesforce (Tableau), and Qlik [31]. An article by *PC Magazine* lists Microsoft Power BI, Tableau Desktop, Sisense, Domo, Google Analytics, Salesforce Einstein, Zoho, SAP Analytics Cloud, and Chartio as the nine best data visualization tools [32].

In addition to the visualization tools or applications mentioned above, there are many open-source libraries that allow analysts to present data in an interactive way and engage a broad audience with new data [33]. An example of open-source visualization libraries is D3.js (https://d3js.org/), where D3 stands for "Data Driven Documents." It is a JavaScript library for producing dynamic, interactive data visualizations in web browsers, with features for interactions and animations. D3.js uses HTML, CSS, and SVG to create data visualizations to be viewed on any browser [33]. Another example is Google Charts (https://developers.google.com/chart), which provides interactive charts for browsers and mobile devices. It uses JavaScript and has a rich gallery of charts, is customizable, connects to dynamic data, and provides interactivity and dashboards.

5.5 Conclusion

Data visualization is a critical capability for understanding and interpreting complex data and relationships. Graphs and charts can tell a story, highlight trends, identify outliers and deviations, make comparisons, and more in a simple and effective way. There are many types of graphs and charts available, and selecting the one that best matches the data and the questions you are trying to answer is crucial. Bad visualizations are difficult to understand and can distort what the data are trying to tell us. Today, it is easy to create very rich visualizations using modern analytics tools with simple pointing and clicking; however, it remains critical to have a good understanding of the data to select the best visualization and be able to interpret it.

5.6 Key Terms

1. Scatterplot
2. Line charts
3. Bar charts
4. Box-and-whisker plots
5. Boxplot
6. Pie chart

7. Bubble chart
8. Time series graphs
9. Ranking relationships
10. Part-to-whole relationships
11. Deviations
12. Distribution
13. Frequency distribution
14. Histograms
15. Correlation
16. Geospatial relationships
17. Nominal comparison
18. Dashboard
19. Infographics
20. Analytics tools
21. Tableau Desktop
22. Power BI
23. Lumira

5.7 Test Your Understanding

1. Cite an example where would you use a scatterplot.
2. When would a line chart be a better fit to the objectives than a bar chart?
3. What is the benefit of a boxplot? Draw two boxplot examples to make your point clear.
4. Your city is looking for the best way to plot poverty levels on a map. What type of visualization tools would you suggest?
5. What is the difference between a bar chart and a histogram? Give a few examples that illustrate the difference.

5.8 Read More

1. Birnbaum, D. (2021). Regarding data visualization. Infect Control Hosp Epidemiol, 42(9), 1154–1155. https://doi.org/10.1017/ice.2020.457
2. Byrd, V., & Dwenger, N. (2021). Activity Worksheets for Teaching and Learning Data Visualization. IEEE Comput Graph Appl, 41(6), 25–36. https://doi.org/10.1109/mcg.2021.3115396
3. Min, S. H., & Zhou, J. (2021). smplot: An R Package for Easy and Elegant Data Visualization. Front Genet, 12, 802,894. https://doi.org/10.3389/fgene.2021.802894
4. Nguyen, V. T., Jung, K., & Gupta, V. (2021). Examining data visualization pitfalls in scientific publications. Vis Comput Ind Biomed Art, 4(1), 27. https://doi.org/10.1186/s42492-021-00092-y

5. Pakarinen, T., & Ojala, J. (2021). Profeel-An open source dosimetry data visualization and analysis software. Comput Methods Programs Biomed, 212, 106,457. https://doi.org/10.1016/j.cmpb.2021.106457
6. Park, S., Bekemeier, B., Flaxman, A., & Schultz, M. (2021). Impact of data visualization on decision-making and its implications for public health practice: a systematic literature review. Inform Health Soc Care, 1–19. https://doi.org/10.1080/17538157.2021.1982949
7. Park, S., Bekemeier, B., & Flaxman, A. D. (2021). Understanding data use and preference of data visualization for public health professionals: A qualitative study. Public Health Nurs, 38(4), 531–541. https://doi.org/10.1111/phn.12863
8. Senanayake, D. A., Wang, W., Naik, S. H., & Halgamuge, S. (2021). Self-Organizing Nebulous Growths for Robust and Incremental Data Visualization. IEEE Trans Neural Netw Learn Syst, 32(10), 4588–4602. https://doi.org/10.1109/tnnls.2020.3023941
9. Shee, K., Pal, S. K., Wells, J. C., Ruiz-Morales, J. M., Russell, K., Dudani, S., Choueiri, T. K., Heng, D. Y., Gore, J. L., & Odisho, A. Y. (2021). Interactive Data Visualization Tool for Patient-Centered Decision Making in Kidney Cancer. JCO Clin Cancer Inform, 5, 912–920. https://doi.org/10.1200/cci.21.00050
10. Wang, Q., Chen, Z., Wang, Y., & Qu, H. (2021). A Survey on ML4VIS: Applying MachineLearning Advances to Data Visualization. IEEE Trans Vis Comput Graph, Pp. https://doi.org/10.1109/tvcg.2021.3106142
11. Wu, A., Wang, Y., Shu, X., Moritz, D., Cui, W., Zhang, H., Zhang, D., & Qu, H. (2021). AI4VIS: Survey on Artificial Intelligence Approaches for Data Visualization. IEEE Trans Vis Comput Graph, Pp. https://doi.org/10.1109/tvcg.2021.3099002

5.9 Lab

5.9.1 Working Example in Tableau

In this chapter, you will learn how to perform basic visualizations using Tableau Desktop. Tableau Desktop is one of the few major data visualization applications that work on both Windows and Mac computers. You will be provided the instructions to get a student copy of Tableau Desktop and be guided to a number of tutorials where you will follow up with the demonstration videos and practice with the provided data files.

5.9.1.1 Getting a Student Copy of Tableau Desktop

Go to Tableau for students (https://www.tableau.com/academic/students) and click on the "Get Tableau for Free" button. You will be asked to provide your student information in order to get a one-year free Tableau Desktop license. Until your credentials are verified, you can use a free 14-day license. You will be provided instructions to download Tableau Desktop for Windows or Mac.

Once your one-year student license expires, if you are still a student, you may request an extension by resubmitting a request at www.tableau.com/studentlicense.

5.9.1.2 Learning with Tableau's how-to Videos and Resources

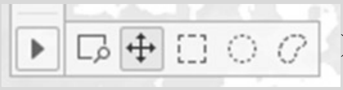
The simplest way to learn how to create visualizations is to use the how-to videos provided by Tableau at https://public.tableau.com/en-us/s/resources. While the videos refer to Tableau Public, the cloud-based visualization application, you can follow the instruction and use the provided data files with Tableau Desktop, which you have downloaded and installed. The videos, which mostly range between 3 and 7 minutes, include step-by-step instructions that you can follow using the respective data file.

The how-to-videos teach you how to connect to data in Excel and CSV formats, Google Sheets, Web Data Connectors, spatial files (for maps), and PDFs. They will teach you how to work with the data and prepare it by cleaning, structuring, pivoting, and merging. You will learn to understand the logic of charts, how to create them, and how to use the "Show Me" feature in Tableau. You will learn how to create and format dashboards and stories. Finally, you will learn how to make visualizations for multiple devices and for sharing on the web.

5.9.2 Do It Yourself

5.9.2.1 Assignment 1: Introduction to Tableau

Go to https://public.tableau.com/en-us/s/resources and watch the first video, entitled "Tableau Public Overview." Follow the instructions and apply them Tableau Desktop. There's no need to create a Tableau Public account and upload your file. Insert the two required screenshots below and submit this answer sheet and your saved Tableau file (yourname.twb) to your instructor. Use a different color or format than the one in the video demo.

1. Insert a screenshot of the map of **Europe** showing the CO2 emission per capita (*Hint: use these buttons:* ).
2. Insert a screenshot of your dashboard (two charts) showing **Canada's** emission trend. *(Hint: click on Canada on the map.)*

5.9.2.2 Assignment 2: Data Manipulation and Basic Charts with Tableau

Go to https://public.tableau.com/en-us/s/resources and watch videos 7–12 inclusive. Follow the instructions and apply them in Tableau Desktop. Insert the required screenshots below and describe in a couple of sentences what you learned from each video. Use a different color or format than the video demo. Submit this answer sheet and your saved Tableau file (yourname.twb) to your instructor.

1. Insert a screenshot and summary from video #7: Data Preparation—The Data Interpreter.
2. Insert a screenshot and summary from video #8: Data Preparation—Pivoting your Data.
3. Insert a screenshot and summary from video #9: Data Preparation—Splitting your Data.
4. Insert a screenshot and summary from video #10: Data Preparation—Joins and Unions.
5. Insert a screenshot and summary from video #11: Creating Your First Chart.
6. Insert a screenshot and summary from video #12: Using the Show Me Tool Bar.

5.9.3 Do More Yourself

5.9.3.1 Assignment 3: Charts and Dashboards with Tableau

Go to https://public.tableau.com/en-us/s/resources and watch videos 13–16 inclusive. Follow the instructions and apply them in Tableau Desktop. Insert the required screenshots below and describe in a couple of sentences what you learned from each video. Use a different color or format than the video demo. Submit this answer sheet and your saved Tableau file (yourname.twb) to your instructor.

1. Insert a screenshot and summary from video #13: Understanding the Logic of Charts.
2. Insert a screenshot and summary from video #14: Combining Sheets on a Dashboard.
3. Insert a screenshot and summary from video #15: Combining Sheets on a Dashboard.
4. Insert a screenshot and summary from video #16: Dashboard Formatting.

5.9.3.2 Assignment 4: Analytics with Tableau

Watch the four Tableau videos mentioned below and follow along with the provided workbook. For each video, you follow along with on your computer using Tableau, provide two screenshots showing that you did the work. Use a different color or format than the video demo. Submit this answer sheet to your instructor.

Note: the first time you access the videos, you may need to create a free online Tableau account if you do not have one.

1. Watch the Tableau Trend Lines video, follow along, and insert two screenshots below.
2. Watch the Tableau Reference Lines video, follow along, and insert two screenshots below.
3. Watch the Tableau Forecasting video, follow along, and insert two screenshots below.
4. Watch the Tableau Clustering video, follow along, and insert two screenshots below.

References

1. N. Kalé, N. Jones, *Practical Analytics* (Epistemy Press, 2015)
2. S. Few, *Show me the Numbers: Designing Tables and Graphs to Enlighten* (Analytics Press, 2012)
3. Tableau, *What Is Data Visualization? Definition, Examples, and Learning Resources*. https://www.tableau.com/learn/articles/data-visualization#:~:text=Data%20visualization%20is%20the%20graphical,outliers%2C%20and%20patterns%20in%20data. Accessed 1 April 2022
4. M. Friendly, *Gallery of Data Visualization*. [Online]. Available: www.datavis.ca/gallery/
5. W. Playfair, Exprorts and Imports to and from Denmark & Norway from 1700 to 1780., P. TimeSeries.png, Ed., ed. (Wikimedia, 1786)
6. W. Playfair, Chart showing at one view the price of the quarter of wheat, and wages of labour by the week, from 1565 to 1821. a. W. o. L. b. t. W. Chart showing at one view the price of the quarter of wheat, from 1565 to 1821.png, Ed., ed. (Wikimedia Commons, 1821)
7. E. Fink, *Three of History's Best Charts Ever According to the Economist*, vol. 2008, ed. (Tableau, 2008)
8. C. Minard, *Carte Figurative des pertes successives en hommes de l'armée française dans la campagne de Russie 1812–1813*. Minard.png, Ed., ed. (Wkimedia Commons, 1869)
9. F. Nightingale, *Diagram of the Causes of Mortality in the Army in the East*. Nightingale-mortality.jpg, Ed., ed. (Wikimedia Commons, 1858)
10. Statictics Canada, *Table 17-10-0005-01 Population Estimates on July 1st, by Age and Sex*. (2021)
11. Tableau, *Which Chart or Graph Is Right for You?* (Tableau, 2021). [Online]. Available: https://www.tableau.com/asset/which-chart-or-graph-is-right-for-you
12. CHHS, *West Nile Virus Cases, 2006-Present*. California Department of Public Health. https://data.chhs.ca.gov/dataset/west-nile-virus-cases-2006-present. Accessed 8 August 2018

13. S. Canada, *Statistics Canada. Table 13-10-0394-01 Leading Causes of Death, Total Population, by Age Group*. https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310039401. Accessed 28 June 2018
14. S. Canada, *Canadian Community Health Survey, 2012: Annual Component*. [Online]. Available: http://www23.statcan.gc.ca/imdb/p2SV.pl?Function=getSurvey&Id=135927
15. Statitics Canada, Table 11-10-0239-01 Income of Individuals by Age Group, Sex and Income Source, Canada, Provinces and Selected Census Metropolitan Areas. (2016)
16. ESRI, *ESRI ArcGIS Online*. https://www.esri.com/en-us/arcgis/products/arcgis-online/overview. Accessed
17. C. o. Calgary, *Open Calgary*. https://data.calgary.ca/. Accessed
18. Statistics Canada, *Distributions of Household Economic Accounts, Wealth: Interactive Tool*. https://www150.statcan.gc.ca/n1/pub/71-607-x/71-607-x2020006-eng.htm. Accessed 8 April 2022
19. Statistics Canada, New housing price index: Interactive dashboard. https://www150.statcan.gc.ca/n1/pub/71-607-x/71-607-x2019013-eng.htm. Accessed 8 April 2022
20. S. Few, *Information Dashboard Design* (O'Reilly, 2006)
21. Tableau, Visual analysis best practices - simple techniques for making every data visualization useful and beautiful. (2021)
22. Sisense, *Lead Generation Sample Dashboard*. https://trial.sisense.com/app/main#/dashboards/62518c886d94960036e1cb59. Accessed 9 April 2022.
23. Sisense, *E-commerce Sample Dashboard*. https://trial.sisense.com/app/main#/dashboards/625191e36d94960036e1cbeb. Accessed 9 April 2022
24. H. Scott, S. Fawkner, C. Oliver, A. Murray, *Why Healthcare Professionals Should Know a Little about Infographics*, vol 50 (Ed: BMJ Publishing Group Ltd and British Association of Sport and Exercise Medicine, 2016), p. 1104
25. J. Lankow, J. Ritchie, R. Crooks, *Infographics: The Power of Visual Storytelling* (John Wiley & Sons, 2012)
26. A. Arcia et al., Sometimes more is more: Iterative participatory design of infographics for engagement of community members with varying levels of health literacy. J. Am. Med. Inform. Assoc. **23**(1), 174–183 (2015)
27. S. Canada, Physical activity of Canadian children and youth, Ed. (2017)
28. Statistics Canada, *Rail Transportation in Canada*. (2020). https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2022024-eng.htm. Accessed
29. C.N. Knaflic, *Storytelling with Data: A Data Visualization Guide for Business Professionals* (John Wiley & Sons, 2015)
30. D. Wong, *Guide to Information Graphics (The Wall Street Journal)* (W. W. Norton & Company, New York, 2010)
31. A. Kronz, K. Schlegel, J. Sun, D. Pidsley, A. Ganeshan, Magic quadrant for analytics and business intelligence platforms. (2022). [Online]. Available: https://www.gartner.com/document/4012759?ref=solrAll&refval=322141776
32. P. Baker, The best data visualization tools. https://www.pcmag.com/picks/the-best-data-visualization-tools?test_uuid=06r4MYCu5PZzCkufjQSV3po&test_variant=a. Accessed 9 April 2022
33. IBM, *Data Visualization*. https://www.ibm.com/cloud/learn/data-visualization#toc-open-sourc-b_jcRWNy. Accessed 20 July 2022

Chapter 6
Linear Regression

6.1 The Problem

Regression aims at predicting a future *value*, so the outcome we are trying to predict is a number, not a class. Many problems can be reduced to predicting a number; for example, predicting the median house value, or predicting the number of people who will be infected by a virus, or predicting the rate of readmission to a hospital in a certain season, etc. In all these examples, our outcome is a number, so regression can be used to predict the outcome. In other words, we can use regression to build a model that can predict (with a certain likelihood of success) the outcome based on the existing features (i.e., dataset attributes). The situation resembles estimating a function f that takes many variables as an input and computes a number that estimates (with a certain likelihood of success) what the future outcome will be. The statement "with a certain likelihood of success" refers to the probability that the model (i.e., function) is correct; that model's probability of success can be computed when we build the model.

There are two types of regression algorithms: linear regression, which is the subject of this chapter, and logistic regression, which is the subject of the next chapter. Linear regression is used when we have reasons to believe that the model (i.e., function) that predicts the outcome based on the input features is a linear model; for example, a model such as outcome $= 2x + 1.5y$ is a linear model. Note that when the outcome is to be predicted from a single input variable (x), the regression is described as simple linear regression, and whenever multiple input variables are involved, we use multiple linear regression. Logistic regression is used when the outcome is binary; for instance, true or false, success or failure, admitted or not admitted.

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_6

6.2 A Practical Example

Let us take a simple example to understand the process of regression and the method called gradient descent.

Suppose we have the following dataset plotted in Fig. 6.1 and Table 6.1:

Our dataset seems to suggest that the relation between x and y is linear, which means that it is reasonable to approximate the graph that passes through all the points with a line (Fig. 6.2).

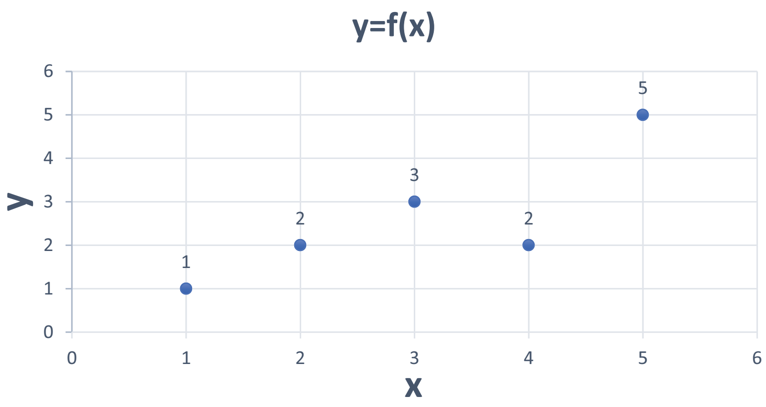


Fig. 6.1 Graph for the dataset

Table 6.1 Dataset

X	1	2	3	4	5
Y	1	2	3	2	5

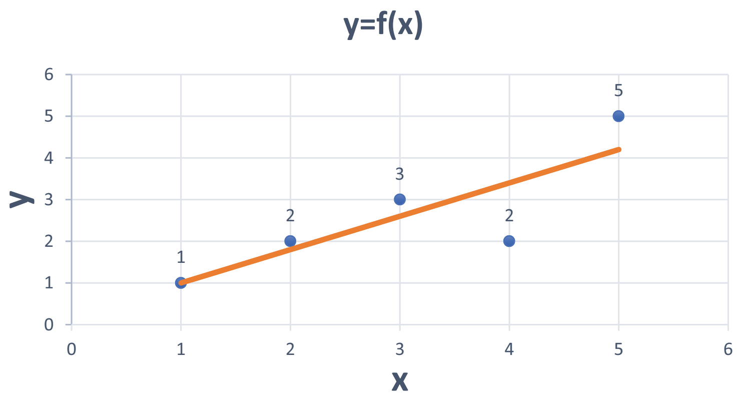


Fig. 6.2 A line representing the relationship between y and x

Of course, the line that can be represented by $f_{\mathrm{model}}(x) = w_1x + w_2$ cannot cross all the data points; hence, at each data point, an error will occur, which is equal to the difference between the real value y and the estimated or predicted value $\widehat{y}$. We can compute these errors, square them (to remove the sign effect), and sum them up: $E = \sum_{i=1}^{N}\left(y^i - \widehat{y}^i\right)^2$; this is the overall error introduced by the model. Our objective is to find (i.e., compute) the coefficients w_1 and w_2 that allow us to minimize the error E of the model. We can then use that line (the model) with minimal error to predict the y (the outcome) for any new point x (the input) that is not in our dataset.

Now, let us perform a gradient descent to calculate the w_1 and w_2 of the model with the minimum error.

We can start with random values for w_1 and w_2: $w_1 = 0$ and $w_2 = 0$; our linear model can be represented by $f_{\mathrm{model}}(x) = 0x + 0$.

We start with the first input in our dataset, $x = 1$ and we compute the predicted value for its outcome, $f_{\mathrm{model}}(1) = 0 \times 1 + 0 = 0$; the model's error for $x = 1$ is $e_1 = f_{\mathrm{model}}(x) - f(x) = 0 - 1 = -1$.

For the sake of simplicity, we will use the stochastic gradient descent that allows us to compute new coefficients after using each input instance of the dataset. We have passed through one instance (1, 1) so let us update the coefficients w_1 and w_2 based on the following formula: for all coefficients w_j, $j = 1$ to n; $w_j = w_j - \eta \times d$, where $d = \nabla f(\mathrm{w_j})$, where η is a learning rate that we will explore below and d is the derivative of $f(x)$ for w_j; we will consider $\eta = 0.01$; the *error* e^i is the error induced by the model at a particular instance i.

$$w_2 = w_2 - \eta \times e^i \times \frac{d(w_1x + w_2)}{dw_2} = w_2 - \eta \times e^i \tag{6.1}$$

$$w_1 = w_1 - \eta \times e^i \times \frac{d(w_1x + w_2)}{dw_1} = w_1 - \eta \times e^i \times x \tag{6.2}$$

$$w_2 = 0 - 0.01 \times (-1) = 0.01$$

$$w_1 = 0 - 0.01 \times (-1) \times 1 = 0.01$$

We will use these new coefficients for the second instance (2, 2), and so on and so forth. Each time we finish all the instances, we call this an epoch. After 30 epochs, we will find the following values for w_1 and w_2; $w_1 = 0.81626$ and $w_2 = 0.20506796$; we can use them to predict an $f(x)$ for a new value of x.

If we compute the average error for each epoch and we display an error graph, also called a cost graph (Fig. 6.3), we will have an idea about the convergence of the model. For our example, we can notice that the model converged after seven or eight epochs.

Now that we have seen a specific example, we can explore the theory behind it.

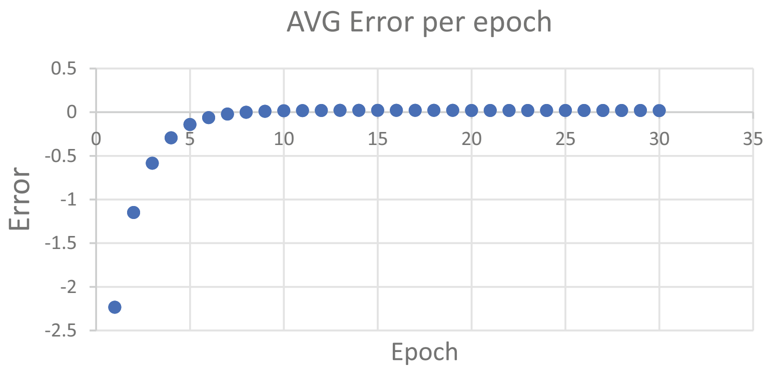


Fig. 6.3 Cost graph

6.3 The Algorithm

6.3.1 Modeling the Linear Regression

The prediction (y) for an input variable (x) in a simple linear regression can be represented as y = coefficient * x + constant (like the function that represents a line), also noted as

$$y = w \times x + w_0.$$

In a multiple regression, we need to predict y using n input variables $x_1, x_2, x_3, \ldots$ x_n; hence, for every instance i in the dataset, we can represent the prediction $\widehat{y}^i$ of the real value y by

$$\widehat{y}^i = w_1 \times x_1^i, w_2 \times x_2^i, \ldots w_n \times x_n^i$$
$$+ w_0,\text{,where w0,w1,} \ldots \text{wn are the parameters of the model.}$$

Since for an instance i (i.e., a data point i), $\widehat{y}^i$ is only a predicted (i.e., estimated) value of y^i, then the difference $y^i - \widehat{y}^i$ represents the error e^i of the model for the instance i. What we would like to construct is a model that minimizes the sum of the errors in relation to all the instances in the dataset; that is, to minimize the sum of error squares E:

$$E = \sum_{i=1}^{N} \left(e^i\right)^2 = \sum_{i=1}^{N} \left(y^i - \widehat{y}^i\right)^2 = \sum_{i=1}^{N} \left(y^i - \sum_{j=0}^{n} w_j \times x_j^i\right)^2$$

where $x_0^i = 1$; indeed, w_0 is a constant, so $w_0 = w_0 \times 1 = w_0 \times x_0^i$,where $x_0^i = 1$. Note that we have added the errors' squares instead of the errors because we are interested in the magnitude of the errors, not in their signs.

Solving this equation is computationally difficult; instead, a gradient descent will be used.

6.3.2 Gradient Descent

If the error E can be written as a function, the problem of finding the minimum E becomes a problem of finding the minimum of a function, which solution is relatively easy.

To simplify, we will take a function with one input variable x. To find the minimum of a function f at a single variable x, we need first to find the derivative $f^{'}(x)$, which measures how much the function $f(x)$ changes with respect to a change in the input (x).

$$f'(x) = \frac{\text{change in} \, f(x)}{\text{change in} \, x}$$

If the derivative $f^{'}(x)$ is negative, we know that the function f at that point is going downhill (i.e., f is decreasing); if the derivative is positive, then the original function is going uphill (i.e., f is increasing).

Knowing the direction of the function f will allow us to take a step in the required direction, i.e., downhill, so that we can find the minimum of f. The process is called gradient descent.

To illustrate the process, we will use a function $y = f(x) = x^2$. The derivative of $f(x)$, notated as $f^{'}(x)$, is equal to $2x$. The following figure (Fig. 6.4) plots $f(x)$ as a curve, and its corresponding derivative $f^{'}(x)$ as a line. Starting from the negative end of the x-axis, we can note that $f(x)$ is decreasing until it reaches a minimum and starts increasing again; while $f(x)$ is decreasing toward the minimum, $f^{'}(x)$ is always negative until $f(x)$ reaches its minimum (i.e., zero). Past the minimum, $f(x)$ increases and $f^{'}(x)$ becomes positive. Note that **$\boldsymbol{f^{'}(x)}$ is 0 when $\boldsymbol{f(x)}$ is at its minimum**.

$f(x) = x^2$ is a function with one global minimum; we call such functions convex functions. Non-convex functions have more than one local minimum and one global minimum (Fig. 6.5).

If we can demonstrate that the error E can be expressed as a function, then we can use the gradient descent to find its minimum. Given that $\widehat{y}^i = w_1 \times x_1^i$, $w_{2\times} x_2^i$, … $wn \times x_n^i + w_0$, using a vector notation, we can express all predicted $\widehat{y}^i$ as a matrix $\widehat{Y}$ that is a function of a two-dimensional matrix X and a weight vector W, $\widehat{Y} = WX$:

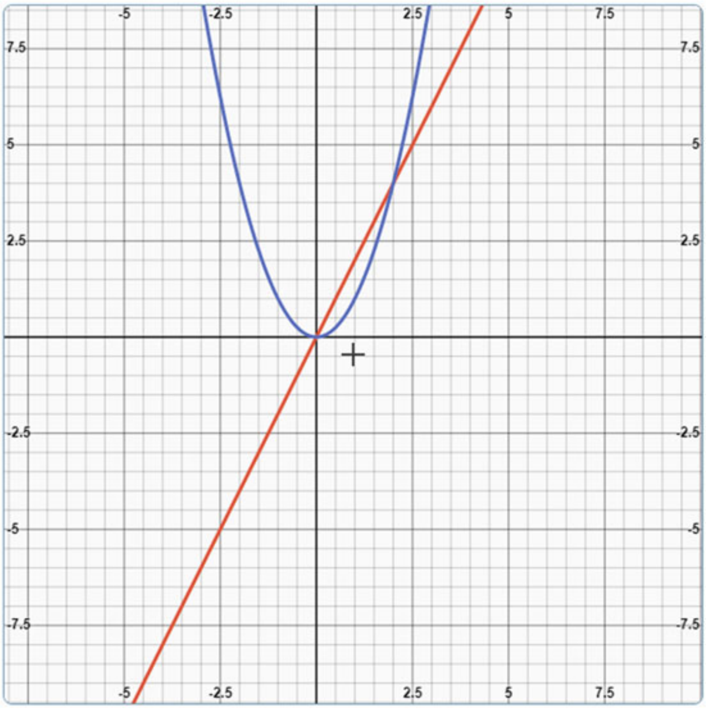


Fig. 6.4 The plot of $f(x) = x^2$ as a curve, and its derivative $f'(x) = 2x$ as a line. Note that the derivative = 0 when $x = 0$

$$\widehat{Y} = \left[\widehat{y}^1\ \widehat{y}^2 \dots\ \widehat{y}^N\right] = [\mathrm{w}_0, \mathrm{w}_1, \dots \mathrm{w}_n] \times \begin{bmatrix} x_0^1 & x_0^2 & \dots & x_0^N \\ x_1^1 & x_1^2 & \dots & x_1^N \\ \dots & \dots & \dots & \dots \\ x_n^1 & x_n^2 & \dots & x_n^N \end{bmatrix} = WX$$

Since $\widehat{Y}$ is a function, then E, which is equal to $\sum_{i=1}^{N}\left(y^i - \widehat{y}^i\right)^2$, is a function too, and its minimum can be found using a gradient descent approach, which requires finding the coefficients w_i for which the derivative of the function E is equal to 0, i.e., the gradient of E with respect to W, is equal to 0. For those values of w_i for which the gradient of E (noted ∇E) is zero, the function E is at its minimum. The graph in Fig. 6.6 represents the error function E for one variable w; in a real-life problem, we will have many features and hence any weights w_i.

The word "gradient" is borrowed from calculus, where the gradient ∇ of a function $f(v_0, v_1, \dots v_n)$ at a certain point p is the vector of partial derivatives of f at p, notated as $\nabla f(p)$:

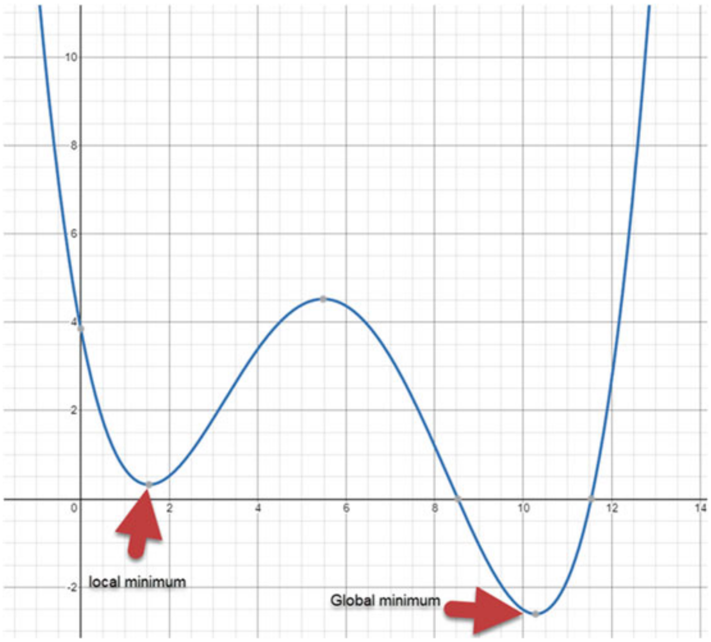


Fig. 6.5 Non-convex function with one local minimum and one global minimum

$$\nabla f(p) = \begin{bmatrix} \dfrac{\partial f}{\partial v_1}(p) \\ \cdots \\ \dfrac{\partial f}{\partial v_n}(p) \end{bmatrix}.$$

In our case, we are computing the partial derivatives of the error function E with respect to W, which express how much E varies with the variation of each w_i.

$$\nabla E = \begin{bmatrix} \dfrac{\partial f}{\partial w_0}(p) \\ \cdots \\ \dfrac{\partial f}{\partial w_n}(p) \end{bmatrix}.$$

To find the minimum, we start at a random point on the function E. We compute the derivative of E at that point; then we slightly change W by adding a value noted η to each coefficient; η denotes the learning rate, $w_j = w_j - \eta \times d$, where $d = \nabla f(\mathrm{w_j})$, and represents the model's error attributed to the weight [1]. The learning rate η should be small and is used to avoid oscillation [2], i.e., avoid learning by jumping back and forth over a minimum, as we will see in the next paragraph.

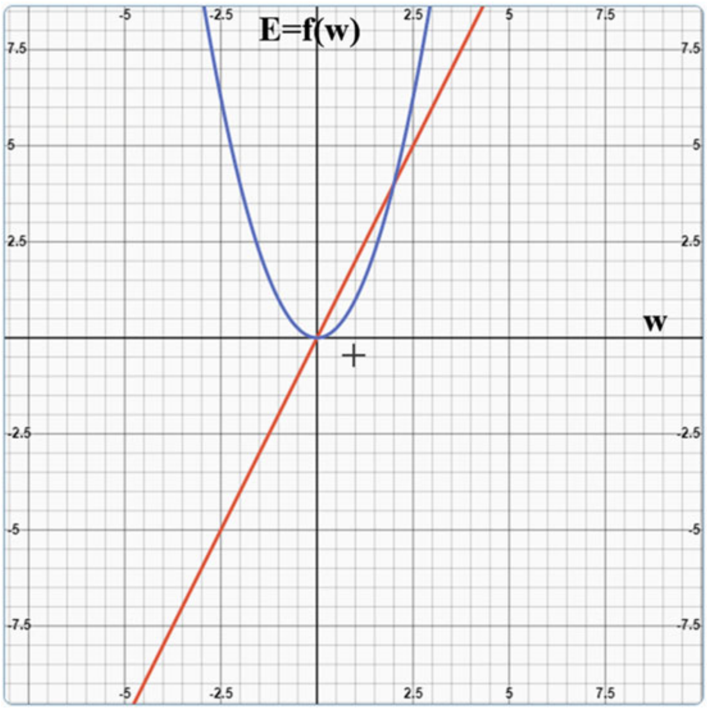


Fig. 6.6 The error function E in terms of one variable w

6.3.3 Gradient Descent Example

To clarify the above theoretical background, let us take the following practical example to illustrate how gradient descent works.

We will start with the simple function $f(x) = x^2$ (Fig. 6.1), which we will take as an example of a function whose minimum we want to find.

Let us start with some point on the graph and try to reach the minimum; any point is OK. We will take $x = 4$, for which $y = x^2 = 16$, so our starting point P is at $x = 4$, $y = 16$.

The gradient of $f(x)$ at P is the derivative, computed as follows:

$$f'(x) = \frac{\mathrm{d}y}{\mathrm{d}x} = 2x = 2 \times 4 = 8$$

since the derivative is positive, then the curve is ascending at P, so we need to go in the other direction to reach the minimum; that is why we have the minus in the formula that updates the coefficient $x_j = x_j - d$, where d is a derivative of $f(x)$; the minus will force x to decrease if the derivative is positive (i.e., the graph is ascending). As mentioned above, the correct formula is $x_j = x_j - \eta \times d$; however, we will not use η in order to illustrate its need in the next paragraphs.

Let us update the position of P to make it move toward the minimum, the next position for P will be $x = x - 8 = 4 - 8 = -4$; the corresponding y is $(-4)^2 = 16$. So, our next position is $P(-4, 16)$. We can notice that we have moved too far and passed the minimum. Let us continue the gradient descent from the new position $P(-4, 16)$.

The gradient of $f(x)$ at $P(-4, 16)$ is computed as follows: $f'(x) = \frac{dy}{dx} = 2x = 2 \times (-4) = -8$; since the derivative is negative, the curve is descending at $P(-4, 16)$, so we need to go in the same direction (i.e., downward) to reach the minimum, and that is why we have the minus in the formula that updates the coefficient $x_j = x_j - \eta \times d$ (always considering $\eta = 1$), as it will allow us to increase x, which is what we need at this point.

The next position for P will be at $x = x - d = -4 - (-8) = -4 + 8 = 4$; we are returning to the original point $P(4, 16)$. If we continue, we will oscillate around the minimum without ever reaching it. That is the importance of having the learning rate η and having it less than 1. In the following, we will redo our calculation for $P(4, 16)$ using $\eta = 0.1$:

$$f'(x) = \frac{dy}{dx} = 2x = 2 \times 4 = 8$$

$$x_j = x_j - \eta \times d = 4 - 0.1 \times 8 = 4 - 0.8 = 3.2$$

For $x = 3.2, f(x) = (3.2)^2 = 10.24$.

The next point would be $P(3.2, 10.24)$. It is a step in the right direction, but we are not at the minimum yet. We can continue our descent until we attain the minimum, or practically until we are very close to it. Basically, we do not have to reach the minimum of 0; it is enough to be very close to it. With more and more iterations, the change in x becomes insignificant and the model becomes stable; we say that the model *converges* to a good solution, and there is no gain from moving toward the minimum anymore. We say that we have attained convergence.

This example considered a function $f(x) = x^2$, but the gradient descent can be followed with any function, including the error function E, also known as the cost function. In that case, we will compute the derivatives of E over all the function weights, since we are trying to compute in which direction (and how much) the function changes with respect to changes in the weights.

6.3.4 Batch Versus Stochastic Gradient Descent

There are two approaches for iteration: the batch and stochastic methods. The batch gradient descent uses all the datasets for every iteration, calculating the sum of all errors over all instances in the training dataset. For every iteration, the gradient is calculated using the entire training dataset.

On the other hand, stochastic gradient descent is instance-based; for every iteration, one single instance of the training dataset is used to calculate the gradient of the error function. This is much less complex and more efficient than the batch method. In stochastic gradient descent, we update the model weights incrementally, calculating the error for each individual instance in the dataset:

$$w_j = w_j - \eta \times d.$$

6.3.5 *Examples of Error Functions*

There are many ways to calculate the errors.

1. Least Mean Squares (LMS)

One well-known error function E called the least mean squares is represented as

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(y^i - \sum_{j=0}^{n} w_j \times x_j^i\right)^2$$

The $\frac{1}{2}$ is added for computational convenience [3].

2. Mean Squared Error (MSE)

Another well-known error function is the mean squared error (MSE) function, represented as follows:

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \sum_{j=0}^{n} w_j \times x_j^i\right)^2$$

3. Root Mean Squared Error (RMSE)

A variation of the MSE is the root mean squared error (RMSE) function, represented as follows:

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y^i - \sum_{j=0}^{n} w_j \times x_j^i\right)^2}$$

6.3.6 Gradient Descent Types

In this paragraph, we will show a brief description on how to proceed in scholastic and batch gradient descents.

6.3.6.1 Stochastic Gradient Descent

Consider a least mean squared error function $E=\frac{1}{2}\sum_{i=1}^{N}\left(y^i-\widehat{y}^i\right)^2$; to show the process, we will consider a function with one variable x.

In a stochastic gradient descent method, the error for an instance i is computed as follows:

$$e^i=\frac{1}{2}\left(y^i-\left(w_1x^i+w_2\right)\right)^2=y^{i2}-2yw_1x-2yw_2+w_1x+2w_1w_2x+{w_2}^2$$

$$\frac{d(e^i)}{dw_1}=\frac{d\left(y^{i2}-2yw_1x-2yw_2+w_1x+2w_1w_2x+{w_2}^2\right)}{dw_1}=-x\left(y^i-\widehat{y}^i\right)=-x\times e^i$$

$$\frac{d(e^i)}{dw_2}=\frac{d\left(y^{i2}-2yw_1x-2yw_2+w_1x+2w_1w_2x+{w_2}^2\right)}{dw_2}=-\left(y^i-\widehat{y}^i\right)=-e^i$$

Hence, the coefficients of the error function E in the next iteration will be updated according to the formula:

$$w_2=w_2-\eta\times\frac{d(e^i)}{dw_2}=w_2-\eta\times\text{error}$$

$$w_1=w_1-\eta\times\frac{d(e^i)}{dw_1}=w_1-\eta\times\text{error}\times x$$

Those are the formulas we used in the practical example at the beginning of the chapter. After each iteration, the weights are updated, and the next prediction is made until the end of the instance (i.e., an epoch elapses). We continue with new epochs until convergence.

6.3.6.2 Batch Gradient

Considering

1. An MSE error function, $E = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^{2.}$
2. A two-dimensional data array called "data" of N rows and two columns, 0 and 1, which stores the instances' x values in column 0 and the instances' y values in column 1 (x^0 is in data[0,0], y^0 is in data[0,1], x^1 is in data[1,0], y^0 is in data [1], . . . x^N is in data[0,0], y^N is in data[0,1])
3. A simple linear regression that consists of $\hat{y} = ax + b$

We can compute the batch gradient descent in the following way:

1. A function called *compute_E* that computes the error function result

```
BEGIN function compute_E
error=0
For i=0 to N-1 do
BEGIN
 x= data [i, 0]
 y= data [i,1]
 y_predicted= a*x + b
 error=error+(y- y_predicted)$^2$
END
Return error / N
END function compute_E
```

2. A function called *compute_gradient* that uses the MSE function to compute the gradients

   Given that $E = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2$, we can demonstrate that $\frac{dE}{da} = \frac{2}{N}\sum_{i=1}^{N}(-x_i(y^i - (ax_i + b)))$ and $\frac{dE}{db} = \frac{2}{N}\sum_{i=1}^{N}(-y_i(y^i - (ax_i + b)))$.

   We will use a two-dimensional array with one row called "gradient" to store all gradients for all dataset instances.

```
BEGIN function compute_gradient
gradient [0,0] = 0
gradient [0,1] = 0
For i=0 to N
BEGIN
x= data [i, 0]
y= data [i,1]
gradient [0,0] = gradient [0,0] - (2/N) * x* (y- (m*x + b))
gradient [0,1] = gradient [0,1] - (2/N) * (y- (m*x + b))
END
Return the array gradient
END function compute_gradient
```

3. A function *linear_regression* that uses the *compute_gradient* and *compute_E* functions to find the minimum error

   We will use a two-dimensional array with one row called "computed_gradients" to store all gradients for all dataset instances.

```
BEGIN function linear_regression
maximum_iterations = 60000
learning_rate=0.01
Erro_of_the_model=0
For i=0 to maximum_iterations
BEGIN
//Call the function compute_gradients and return gradient array
computed_gradients = compute_gradients
//Update the parameters of the model
a=a- learning_rate * computed_gradients [0,0]
b=b- learning_rate * computed_gradients [0,1]

//Call compute_E and get the error of the model
Error_of_the_model= compute_E
//Plot the error of the model on the screen
plot Error_of_the_model
END
//Print on the screen the final values of a and b for the final model
Print "the model is y=", a, "x +" b
BEGIN function linear_regression
```

6.4 Final Notes: Advantages, Disadvantages, and Best Practices

Linear regression works under the assumption that the underlying model can predict if the data is linear (i.e., the curve that passes through all or most of the data points is a line). If the assumption is different, a nonlinear model such as decision trees, naïve Bayes, k-nearest neighbors, or support vector machines should be employed.

The main advantages of the algorithm are its simplicity and efficiency. When using the algorithm, use the following tips from [1]:

1. Display a cost graph: Plot the cost graph for each iteration. If your algorithm does not converge, reduce the learning rate.
2. Learning rate: Try different values for the learning rate.
3. Rescale inputs: Standardize your input variables to the same range; it will enhance the algorithm's performance.
4. Use stochastic gradient descent: It performs much better than batch gradient descent. The stochastic method converges within many fewer epochs (e.g., 10).
5. Plot mean cost: In stochastic gradient descent, to avoid a jittery graph, plot the average of the error over many updates (e.g., 10, 100).
6. Linear relationship: You may need to transform your data to make the relationship linear; for example, in an exponential relation, you can use a log transform.
7. Remove noise: Preprocess your data to remove outliers if needed.
8. Avoid collinearity: Avoid highly correlated input variables, as linear regression will overfit your data in this case. Remove the most correlated variables.

6.5 Key Terms

1. Regression
2. Logistic regression
3. Simple linear regression
4. Multiple linear regression
5. Gradient descent
6. Global minimum
7. Convex functions
8. Non-convex functions
9. Local minimum
10. Learning rate
11. Oscillation
12. Algorithm convergence
13. Cost function
14. Error function
15. Least mean squares
16. Mean squared error (MSE)
17. Root mean squared error (RMSE)
18. Batch gradient descent
19. Stochastic gradient descent

6.6 Test Your Understanding

1. Describe a regression in your own words.
2. Describe a linear regression in your own words.
3. What is the difference between a simple linear regression and a multiple linear regression?
4. What is the difference between a logistic regression and a simple linear regression?
5. What is the benefit of a learning rate?
6. What is the difference between the mean squared error (MSE) and the least mean squares?
7. What is the use of the mean squared error (MSE) and the least mean squares?
8. What do we mean when we say that the stochastic gradient descent is "instance-based"?
9. Is stochastic gradient descent more efficient than batch gradient descent or not? Explain your answer.
10. Given an error function $E = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \widehat{y}^i\right)^2$, demonstrate that

$$\frac{dE}{da}=\frac{2}{N}\sum_{i=1}^{N}\left(-x_i\left(y^i-(ax_i+b)\right)\right)\text{and }\frac{dE}{db}=\frac{2}{N}\sum_{i=1}^{N}\left(-y_i\left(y^i-(ax_i+b)\right)\right).$$

6.7 Read More

1. Amini Pishro, A., Zhang, S., Huang, D., Xiong, F., Li, W., & Yang, Q. (2021). Application of artificial neural networks and multiple linear regression on local bond stress equation of UHPC and reinforcing steel bars. Sci Rep, 11(1), 15,061. https://doi.org/10.1038/s41598-021-94480-2
2. Barmpalexis, P., Partheniadis, I., Mitra, K. S., Toskas, M., Papadopoulou, L., & Nikolakakis, I. (2020). Application of Multiple Linear Regression and Artificial Neural Networks for the Prediction of the Packing and Capsule Filling Performance of Coated and Plain Pellets Differing in Density and Size. Pharmaceutics, 12(3). https://doi.org/10.3390/pharmaceutics12030244
3. Christodoulou, E., Ma, J., Collins, G. S., Steyerberg, E. W., Verbakel, J. Y., & Van Calster, B. (2019). A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. Journal of Clinical Epidemiology, 110, 12–22. https://doi.org/10.1016/j.jclinepi.2019.02.004
4. Curtis, D. (2020). Multiple Linear Regression Allows Weighted Burden Analysis of Rare Coding Variants in an Ethnically Heterogeneous Population. Hum Hered, 85(1), 1–10. https://doi.org/10.1159/000512576
5. Ghosal, S., Sengupta, S., Majumder, M., & Sinha, B. (2020). Linear Regression Analysis to predict the number of deaths in India due to SARS-CoV-2 at 6 weeks from day 0 (100 cases - March 14th 2020). Diabetes Metab Syndr, 14(4), 311–315. https://doi.org/10.1016/j.dsx.2020.03.017
6. Hossain, S., Biswas, R. K., & Hossain, M. A. (2021). Body mass index of women in Bangladesh: comparing Multiple Linear Regression and Quantile Regression. J Biosoc Sci, 53(2), 247–265. https://doi.org/10.1017/s0021932020000176
7. Jiao, S., Gao, Y., Feng, J., Lei, T., & Yuan, X. (2020). Does deep learning always outperform simple linear regression in optical imaging? Opt Express, 28(3), 3717–3731. https://doi.org/10.1364/oe.382319
8. Jung, W. S., Park, H. Y., Kim, S. W., Kim, J., Hwang, H., & Lim, K. (2021). Estimating excess post-exercise oxygen consumption using multiple linear regression in healthy Korean adults: a pilot study. Phys Act Nutr, 25(1), 35–41. https://doi.org/10.20463/pan.2021.0006
9. Kern, C., Stefan, T., & Hinrichs, J. (2019). Multiple linear regression modeling: Prediction of cheese curd dry matter during curd treatment. Food Res Int, 121, 471–478. https://doi.org/10.1016/j.foodres.2016.11.061
10. Liem, Y., Judge, A., Kirwan, J., Ourradi, K., Li, Y., & Sharif, M. (2020). Multivariable logistic and linear regression models for identification of

clinically useful biomarkers for osteoarthritis. Sci Rep, 10(1), 11,328. https://doi.org/10.1038/s41598-020-68077-0
11. Meulenbroek, N. E., & Pichardo, S. (2021). Multiple Linear Regression Estimation of Onset Time Delay for Experimental Transcranial Narrowband Ultrasound Signals. IEEE Trans Ultrason Ferroelectr Freq Control, 68(4), 1032–1039. https://doi.org/10.1109/tuffc.2020.3030196
12. Mokhort, H. (2020). Multiple Linear Regression Model of Meningococcal Disease in Ukraine: 1992–2015. Comput Math Methods Med, 2020, 5105120. https://doi.org/10.1155/2020/5105120
13. Norouzian, M. A., Bayatani, H., & Vakili Alavijeh, M. (2021). Comparison of artificial neural networks and multiple linear regression for prediction of dairy cow locomotion score. Vet Res Forum, 12(1), 33–37. https://doi.org/10.30466/vrf.2019.98275.2346
14. Paulo Lopes-Silva, J., Panissa, V. L. G., Julio, U. F., & Franchini, E. (2021). Influence of Physical Fitness on Special Judo Fitness Test Performance: A Multiple Linear Regression Analysis. J Strength Cond Res, 35(6), 1732–1738. https://doi.org/10.1519/jsc.0000000000002948
15. Schober, P., & Vetter, T. R. (2021). Linear Regression in Medical Research. Anesth Analg, 132(1), 108–109. https://doi.org/10.1213/ane.0000000000005206
16. Siavash, N. K., Ghobadian, B., Najafi, G., Rohani, A., Tavakoli, T., Mahmoodi, E., Mamat, R., & Mazlan, M. (2021). Prediction of power generation and rotor angular speed of a small wind turbine equipped to a controllable duct using artificial neural network and multiple linear regression. Environ Res, 196, 110,434. https://doi.org/10.1016/j.envres.2020.110434
17. Yap, D. F., Nasir, N., Tan, K. S. M., & Lau, L. H. S. (2019). Variables which predict maternal self-efficacy: A hierarchical linear regression analysis. J Appl Res Intellect Disabil, 32(4), 841–848. https://doi.org/10.1111/jar.12575

6.8 Lab

The following represent typical steps in a machine learning lab exercise [4]:

1. Load your dataset.
2. Explore the data.
3. Split your data into two datasets: one for training and one for testing.
4. Preprocess your data (missing values, erroneous values, conversions of categorical to numeric and vice versa, standardization, normalization, etc.).
5. Create and fine-tune a model (e.g., linear regression).
6. Test the model and evaluate its performance (e.g., accuracy).

The focus of this lab is to create and optimize a linear regression model (LRM). The first part of the lab (in R) uses a diabetes dataset and applies LRM, while the second part (in python) uses a USA housing prices dataset and applies LRM. In both parts, we will be using gradient descent to optimize the linear regression model. Below are typical tasks to create and optimize a linear regression model.

6.8.1 Working Example in R

Here is where you can download the diabetes dataset: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.tab.txt.

This dataset has the following information:

- Age: patient's age
- BMI: body mass index
- BP: average blood pressure reading
- S1: total serum cholesterol
- S2: low-density cholesterol
- S3: high-density cholesterol
- S4: total cholesterol
- S5: serum triglycerides level
- S6: blood sugar level
- Y: quantitative measure of diabetes progression one year after baseline

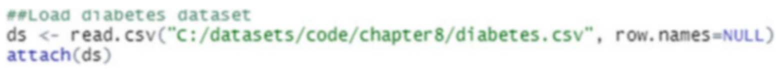
Note that while executing the code below some packages in *R* need to be installed. *R*-Studio will detect them and automatically install them for you; it is instructed to do so by the presence of the function require(). An example would be: require(caTools) which instruct *R*-Sudio to install the caTools library.

6.8.1.1 Load Diabetes Dataset

Load the diabetes dataset as shown in Fig. 6.7.

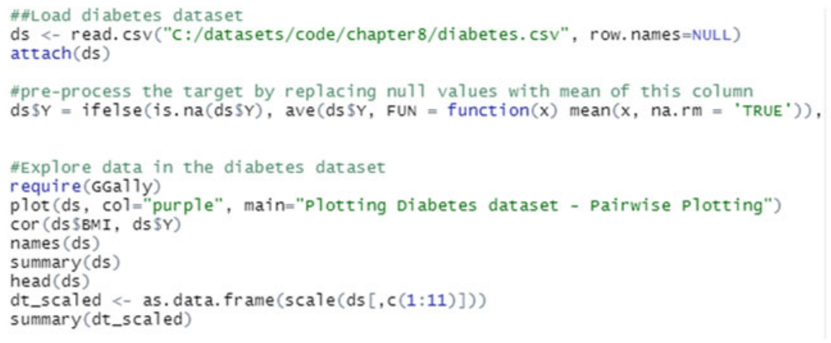
6.8.1.2 Preprocess Diabetes Dataset

Preprocess the data in the diabetes dataset by removing null values and noise. In the snapshot below in Fig. 6.8, all null values are removed from the target field (Y) and replaced with the mean of this column. Replacing null values with the mean value for this column is just one method to process null values, we could also replace them with zeros, the last valid values. There are other methods to preprocess data, such as standardization and normalization, one-hot-encoding of categorical data, etc.



```
##Load diabetes dataset
ds <- read.csv("C:/datasets/code/chapter8/diabetes.csv", row.names=NULL)
attach(ds)
```

Fig. 6.7 Load diabetes dataset



```
##Load diabetes dataset
ds <- read.csv("C:/datasets/code/chapter8/diabetes.csv", row.names=NULL)
attach(ds)

#pre-process the target by replacing null values with mean of this column
ds$Y = ifelse(is.na(ds$Y), ave(ds$Y, FUN = function(x) mean(x, na.rm = 'TRUE')), ds$Y)


#Explore data in the diabetes dataset
require(GGally)
plot(ds, col="purple", main="Plotting Diabetes dataset - Pairwise Plotting")
cor(ds$BMI, ds$Y)
names(ds)
summary(ds)
head(ds)
dt_scaled <- as.data.frame(scale(ds[,c(1:11)]))
summary(dt_scaled)
```

Fig. 6.8 Preprocess data in diabetes dataset

6.8.1.3 Choose Dependent and Independent Variables

In a supervised learning algorithm such as linear regression, we should specify the outcome/target and the features. We are trying to predict the quantitative measure of diabetes progression one year after baseline, *Y* is our target feature (i.e., dependent variable) (Fig. 6.11). The other columns are the features (i.e., factors or independent variables). It is important to note that using more than one feature in the model would make it a multiple linear regression model. A simple linear regression model uses only one factor to predict the target. Moreover, choosing a few factors instead of all available factors is another approach too. To choose a subset of factors for a multiple linear regression model, the correlation between the outcome *Y* and the rest of the factors can be computed; only the factors with the highest correlation scores with the outcome would be retained for use to generate the model. A threshold (e.g., 0.25) can be chosen to rule out factors that have low correlation scores. Figure 6.9 shows the visual correlation between *Y* and the factors.

6.8.1.4 Visualize Your Dataset

The next step is visualizing the dataset, which will allow you to analyze and explore the data for preprocessing before creating the LRM. GGALY is an R library that can be used to visualize data in pairwise plotting. It also helps to understand pair correlation in a graph. This is shown in Fig. 6.9. However, there are also other R libraries for visualizing data.

6.8.1.5 Split Data into Test and Train Datasets

The next step is to split the dataset into training and testing sets. Usually, the training set includes 70% of the dataset, and the remaining 30% is for the testing set (Fig. 6.10).

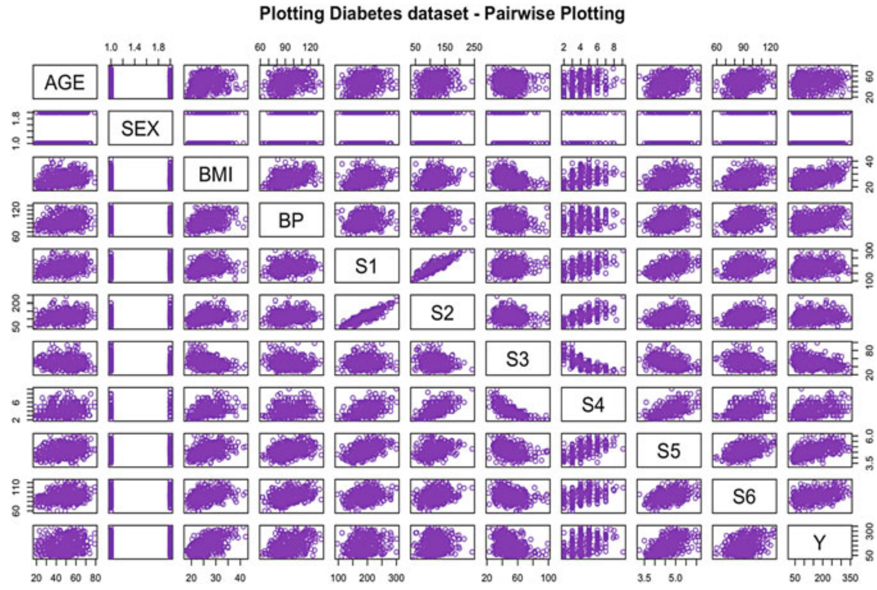


Fig. 6.9 Explore diabetes dataset using visualization

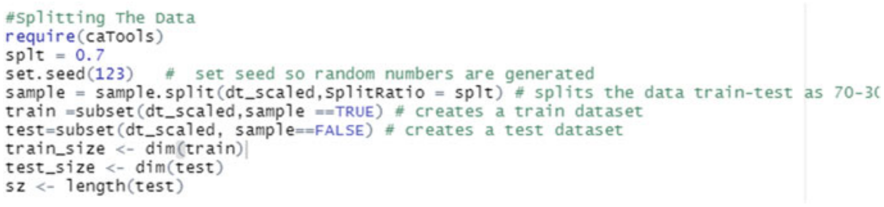
6.8.1.6 Create Linear Regression Model and Visualize it

As shown in Fig. 6.11, the following factors are the independent variables: Age, BMI, BP, S1, S2, S3, S4, S5, and S6. Column *Y* is the outcome that we want to predict. The model is trained and then its performance measure using the testing datasets. The coefficients of the model are displayed; as such, the linear regression formula is:

$$Y = 0.31736 \times \mathrm{BMI} + -0.01647 \times \mathrm{Age} + -0.15220 \times \mathrm{Sex} + 0.22752 \times \mathrm{BP} + -0.18401 \times \mathrm{S1} + 0.06103 \times \mathrm{S2} + -0.04008 \times \mathrm{S3} + 0.11324 \times \mathrm{S4} + 0.32710 \times \mathrm{S5} + 0.05315 \times \mathrm{S6} + (-0.04936).$$

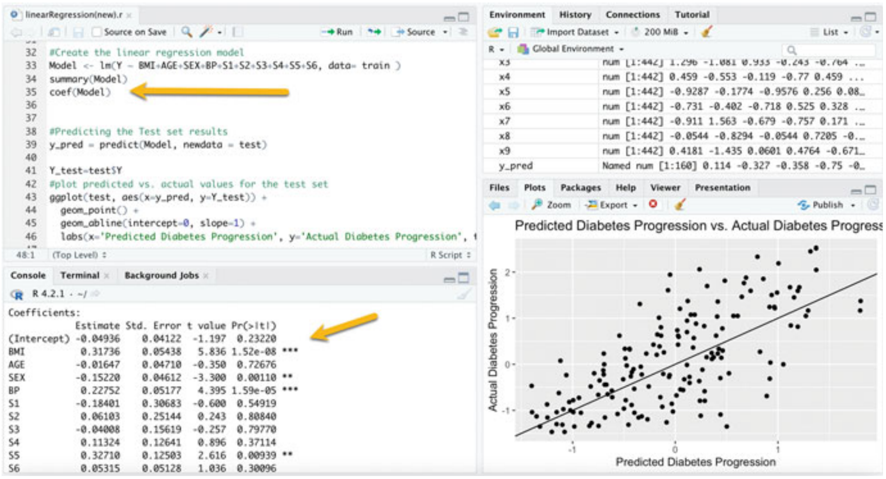
6.8.1.7 Calculate Confusion Matrix

A confusion matrix is a performance measurement for the machine learning algorithm. It summarizes in a table four combinations of the actual and predicted values: true-positives (TP), true-negatives (TN), false-positives (FP), and false-negatives (FN). It is important to note that a confusion matrix works with categorical data. As



```
#Splitting The Data
require(caTools)
splt = 0.7
set.seed(123)   #  set seed so random numbers are generated
sample = sample.split(dt_scaled,SplitRatio = splt) # splits the data train-test as 70-3(
train =subset(dt_scaled,sample ==TRUE) # creates a train dataset
test=subset(dt_scaled, sample==FALSE) # creates a test dataset
train_size <- dim(train)
test_size <- dim(test)
sz <- length(test)
```

Fig. 6.10 Split dataset into train and test size sets



```
#Create the linear regression model
Model <- lm(Y ~ BMI+AGE+SEX+BP+S1+S2+S3+S4+S5+S6, data= train )
summary(Model)
coef(Model)


#Predicting the Test set results
y_pred = predict(Model, newdata = test)

Y_test=test$Y
#plot predicted vs. actual values for the test set
ggplot(test, aes(x=y_pred, y=Y_test)) +
  geom_point() +
  geom_abline(intercept=0, slope=1) +
  labs(x='Predicted Diabetes Progression', y='Actual Diabetes Progression', 
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.04936    0.04122  -1.197  0.23220
BMI          0.31736    0.05438   5.836 1.52e-08 ***
AGE         -0.01647    0.04710  -0.350  0.72676
SEX         -0.15220    0.04612  -3.300  0.00110 **
BP           0.22752    0.05177   4.395 1.59e-05 ***
S1          -0.18401    0.30683  -0.600  0.54919
S2           0.06103    0.25144   0.243  0.80840
S3          -0.04008    0.15619  -0.257  0.79770
S4           0.11324    0.12641   0.896  0.37114
S5           0.32710    0.12503   2.616  0.00939 **
S6           0.05315    0.05128   1.036  0.30096
```

```
x3      num [1:442] 1.296 -1.081 0.933 -0.243 -0.764 ...
x4      num [1:442] 0.459 -0.553 -0.119 -0.77 0.459 ...
x5      num [1:442] -0.9287 -0.1774 -0.9576 0.256 0.08...
x6      num [1:442] -0.731 -0.402 -0.718 0.525 0.328 ...
x7      num [1:442] -0.911 1.563 -0.679 -0.757 0.171 ...
x8      num [1:442] -0.0544 -0.8294 -0.0544 0.7205 -0...
x9      num [1:442] 0.4181 -1.435 0.0601 0.4764 -0.671...
y_pred  Named num [1:160] 0.114 -0.327 -0.358 -0.75 -0...
```

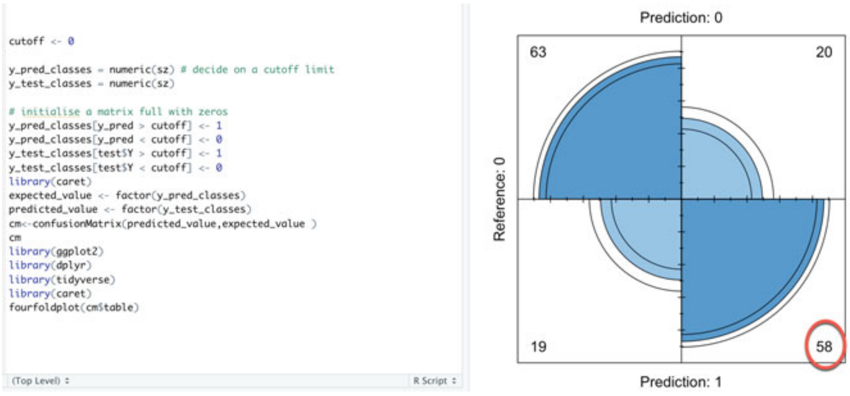
Fig. 6.11 Generate multiple linear regression model and visualize it

such, it is necessary to map out continuous data to labeled data, as shown in Fig. 6.12. Figure 6.12 shows that TP = 58, TN = 63, FP = 19 and FN = 20.

Now we need to compute the error of the model using some statistical metrics (Fig. 6.13). *R*-squared is 0.23 which indicates that the performance of the model is low. Since we have the confusion matrix, we could also compute the sensitivity and the specificity of the model, both of which are important performance measures especially in the health domain. We suggest that you try to compute the sensitivity and specificity of the model; to do so, you might find the "caret" package easy to use.

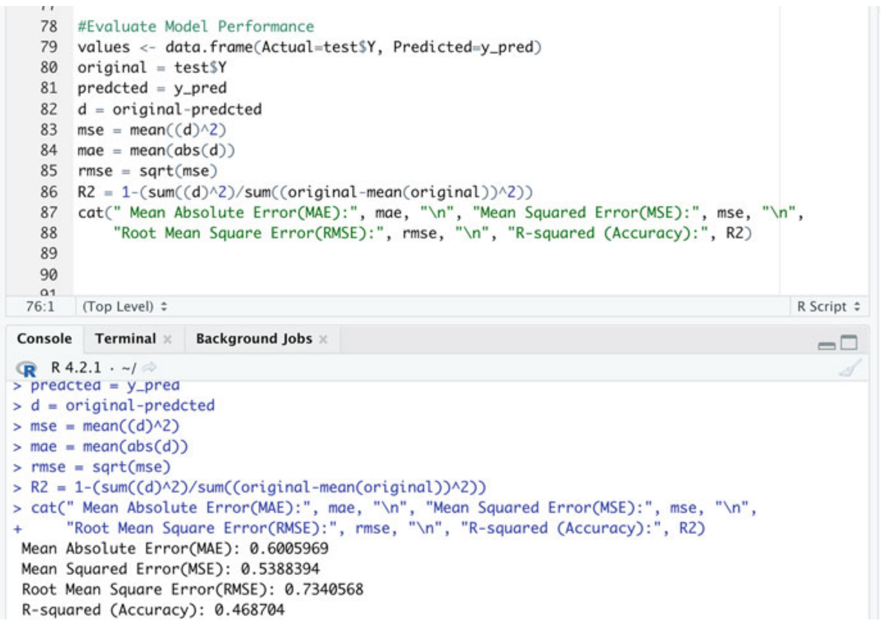
6.8.1.8 Gradient Descent

Since the model is a linear regression, we will implement the gradient descent to optimize convergence and we will print the cost function vs. the number of iterations to notice the relationship between both. As shown in Fig. 6.14, the model's error rate decreases with the number of iterations until around 400 iterations before increasing again. You can try larger learning rates and see the results; it will be interesting to reflect on the changes in the cost function vs. the number of iterations plot.



```
cutoff <- 0

y_pred_classes = numeric(sz) # decide on a cutoff limit
y_test_classes = numeric(sz)

# initialise a matrix full with zeros
y_pred_classes[y_pred > cutoff] <- 1
y_pred_classes[y_pred < cutoff] <- 0
y_test_classes[test$Y > cutoff] <- 1
y_test_classes[test$Y < cutoff] <- 0
library(caret)
expected_value <- factor(y_pred_classes)
predicted_value <- factor(y_test_classes)
cm<-confusionMatrix(predicted_value,expected_value )
cm
library(ggplot2)
library(dplyr)
library(tidyverse)
library(caret)
fourfoldplot(cm$table)
```

Fig. 6.12 Calculate configuration matrix and visualize it



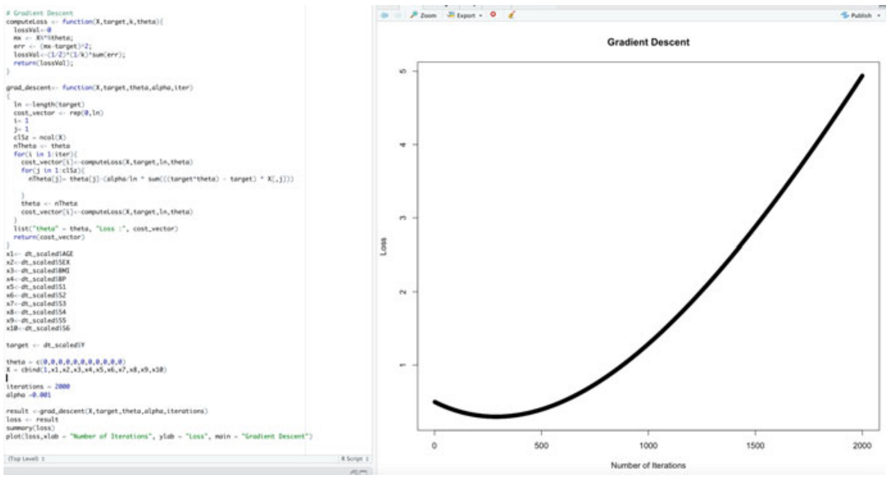
```
#Evaluate Model Performance
values <- data.frame(Actual=test$Y, Predicted=y_pred)
original = test$Y
predcted = y_pred
d = original-predcted
mse = mean((d)^2)
mae = mean(abs(d))
rmse = sqrt(mse)
R2 = 1-(sum((d)^2)/sum((original-mean(original))^2))
cat(" Mean Absolute Error(MAE):", mae, "\n", "Mean Squared Error(MSE):", mse, "\n",
    "Root Mean Square Error(RMSE):", rmse, "\n", "R-squared (Accuracy):", R2)
```

```
> predcted = y_pred
> d = original-predcted
> mse = mean((d)^2)
> mae = mean(abs(d))
> rmse = sqrt(mse)
> R2 = 1-(sum((d)^2)/sum((original-mean(original))^2))
> cat(" Mean Absolute Error(MAE):", mae, "\n", "Mean Squared Error(MSE):", mse, "\n",
+     "Root Mean Square Error(RMSE):", rmse, "\n", "R-squared (Accuracy):", R2)
 Mean Absolute Error(MAE): 0.6005969
 Mean Squared Error(MSE): 0.5388394
 Root Mean Square Error(RMSE): 0.7340568
 R-squared (Accuracy): 0.468704
```

Fig. 6.13 Using statistical metrics to evaluate model performance

6.8.2 *Working Example in Python*

In this example, a linear regression model will be created using the USA housing prices dataset to predict house prices in the United States. The model will be trained and tested on an unseen dataset and optimized to fine-tune the model. The first task in this process is to download the USA housing prices dataset. This dataset can be



```
# Gradient Descent
computeLoss <- function(X,target,k,theta){
  lossVal<-0
  mx <- X%*%theta;
  err <- (mx-target)^2;
  lossVal<-(1/2)*(1/k)*sum(err);
  return(lossVal);
}

grad_descent<- function(X,target,theta,alpha,iter)
{
  ln <-length(target)
  cost_vector <- rep(0,ln)
  i= 1
  j= 1
  cl5z = ncol(X)
  nTheta <- theta
  for(i in 1:iter){
    cost_vector[i]<-computeLoss(X,target,ln,theta)
    for(j in 1:cl5z){
      nTheta[j]= theta[j]-(alpha/ln * sum(((target*theta) - target) * X[,j]))

    }
    theta <- nTheta
    cost_vector[i]<-computeLoss(X,target,ln,theta)
  }
  list("theta" = theta, "Loss :", cost_vector)
  return(cost_vector)
}
x1<- dt_scaled$AGE
x2<-dt_scaled$SEX
x3<-dt_scaled$BMI
x4<-dt_scaled$BP
x5<-dt_scaled$S1
x6<-dt_scaled$S2
x7<-dt_scaled$S3
x8<-dt_scaled$S4
x9<-dt_scaled$S5
x10<-dt_scaled$S6

target <- dt_scaled$Y

theta = c(0,0,0,0,0,0,0,0,0,0,0)
X = cbind(1,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)

iterations = 2000
alpha =0.001

result <-grad_descent(X,target,theta,alpha,iterations)
loss <- result
summary(loss)
plot(loss,xlab = "Number of Iterations", ylab = "Loss", main = "Gradient Descent")
```

Fig. 6.14 Gradient descent for LRM

downloaded from the following location: https://www.kaggle.com/faressayah/linear-regression-house-price-prediction/data. The next step is to load the dataset.

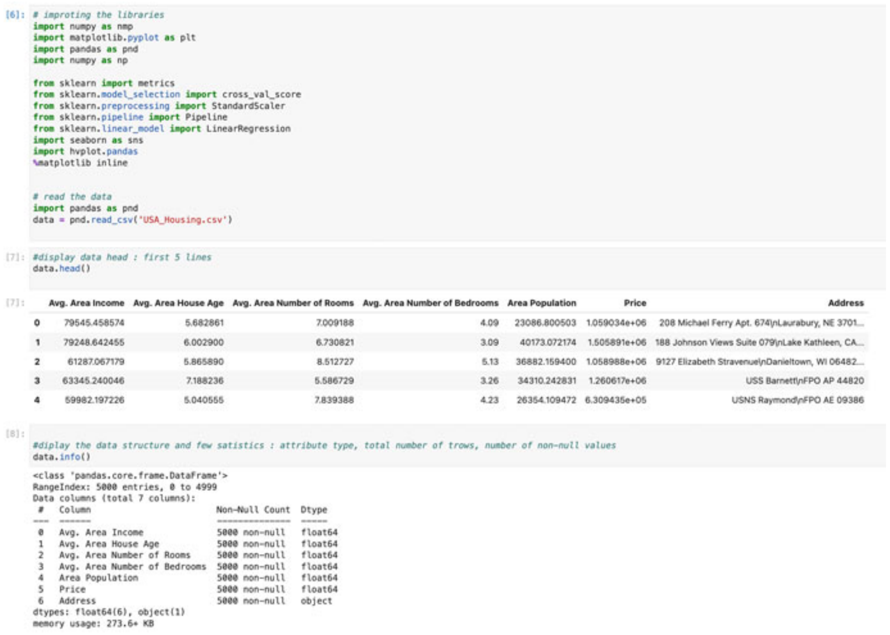
6.8.2.1 Load USA House Prices Dataset

Import the required libraries and install any library that does not exist, such as the hvplot library, using the pip install command. Then, load the dataset, display the head of the data frame as well as the data structure of the dataset (Fig. 6.15).

6.8.2.2 Explore Housing Prices Visually

Use pandas and the seaborn libraries to visualize the data, as shown in Figs. 6.16 and 6.17. It is important to note that Python libraries have different methods to create different types of visualizations, such as bar charts, scatter charts, histograms, etc. We have used pandas to display histograms (Fig. 6.16) and seaborn to visualize the correlation between a pair of columns (Fig. 6.17). This helps to identify factors to predict the house price in the United States. Figure 6.16 shows a histogram of that dataset based on price.

What about trying the following code and explore another way to display correlations?



```
[6]: # improting the libraries
     import numpy as nmp
     import matplotlib.pyplot as plt
     import pandas as pnd
     import numpy as np

     from sklearn import metrics
     from sklearn.model_selection import cross_val_score
     from sklearn.preprocessing import StandardScaler
     from sklearn.pipeline import Pipeline
     from sklearn.linear_model import LinearRegression
     import seaborn as sns
     import hvplot.pandas
     %matplotlib inline


     # read the data
     import pandas as pnd
     data = pnd.read_csv('USA_Housing.csv')
```

```
[7]: #display data head : first 5 lines
     data.head()
```

[7]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

```
[8]:
     #diplay the data structure and few satistics : attribute type, total number of trows, number of non-null values
     data.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 5000 entries, 0 to 4999
     Data columns (total 7 columns):
      #   Column                        Non-Null Count  Dtype
     ---  ------                        --------------  -----
      0   Avg. Area Income              5000 non-null   float64
      1   Avg. Area House Age           5000 non-null   float64
      2   Avg. Area Number of Rooms     5000 non-null   float64
      3   Avg. Area Number of Bedrooms  5000 non-null   float64
      4   Area Population               5000 non-null   float64
      5   Price                         5000 non-null   float64
      6   Address                       5000 non-null   object
     dtypes: float64(6), object(1)
     memory usage: 273.6+ KB
```

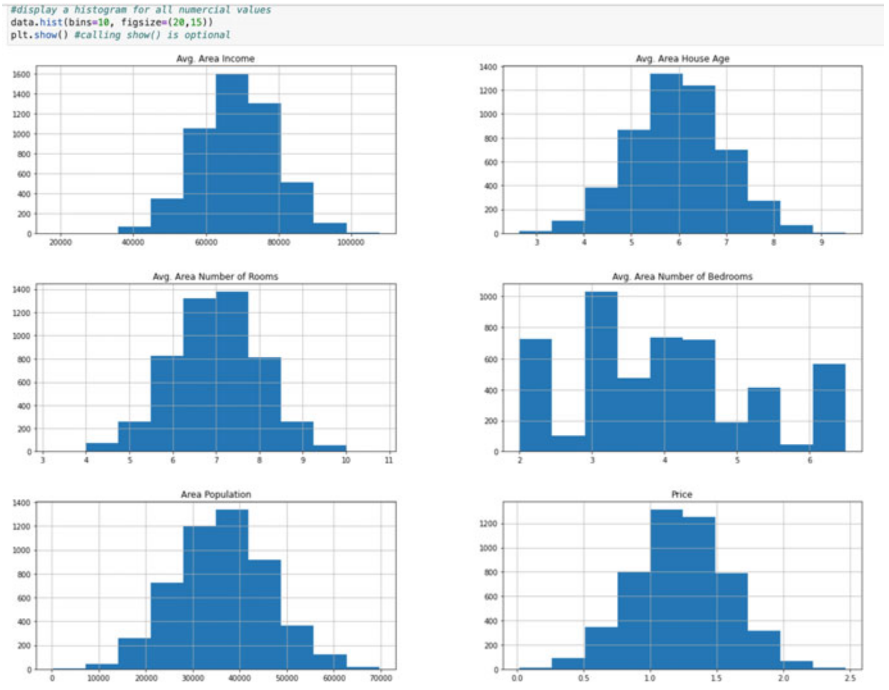
Fig. 6.15 Import libraries and load the USA housing prices dataset and display few information

```
correlation = data.corr()
plt.figure(figsize=(10,10))
sns.heatmap(correlation, vmax=1, square=True,annot=True,
cmap='cubehelix')
plt.title('Correlation between different fearures')
```

6.8.2.3 Preprocess Data

After loading the data, it is necessary to remove null values, if any (Fig. 6.18); we are demonstrating its use here but we know from data.info() in Fig. 6.16 that we have 5000 entries and none of the columns have any null value (all of them have 5000 non-null values).

Our aim is to predict the price of houses (i.e., the target). The features used for prediction are Avg. Area Income, Avg. Area House Age, Avg. Area Number of Rooms, Avg. Area Number of Bedrooms, Area Population; they are all stored in x. while the Price column is placed in y. The “reshape” method is required to transform *x* and *y* from 2D to 1D data.



```
#display a histogram for all numercial values
data.hist(bins=10, figsize=(20,15))
plt.show() #calling show() is optional
```

Fig. 6.16 Explore USA housing prices using a histogram

6.8.2.4 Split Data and Scale Features

We will start by splitting the data into two parts, 70% for training and 30% for testing. Using StandardScaler, we will choose to standardize the data as the scales of the numeric data values varied immensely across features. Instead of scaling and then applying the algorithm, we can combine both steps into what we call a pipeline (Fig. 6.19). We can of course combine more than 2 steps (e.g., many pr-processing steps in addition to the machine learning one, or even many algorithms).

6.8.2.5 Create and Visualize Model Using the LinearRegression Algorithm

A linear regression model is created, and its predictions are visualized in a plot (Fig. 6.20).

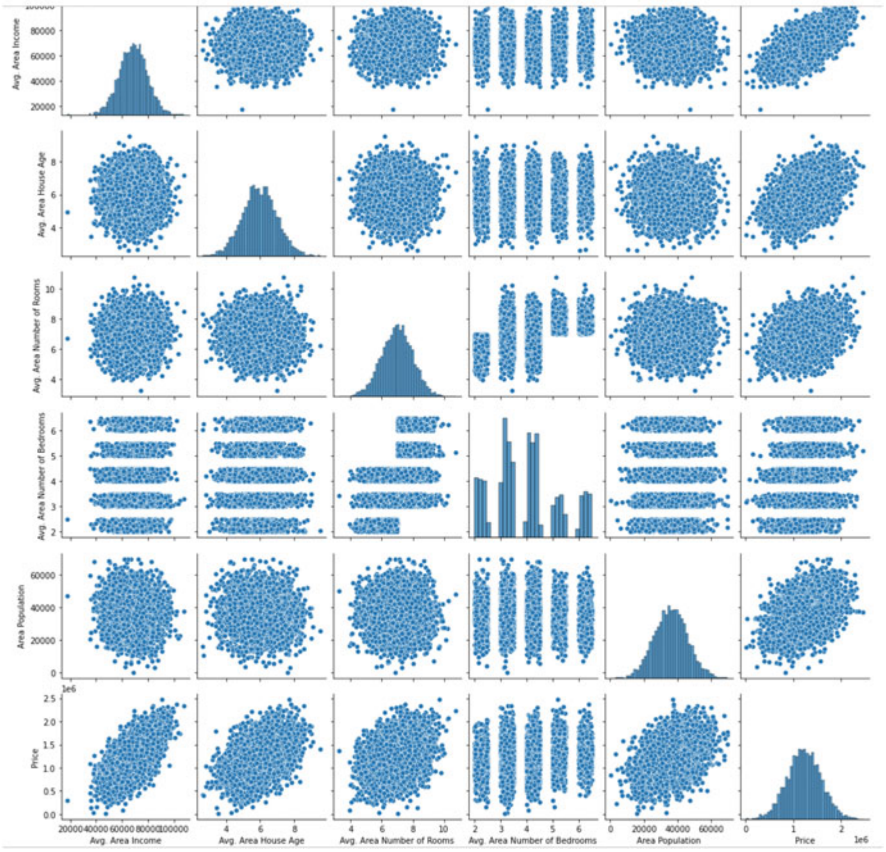
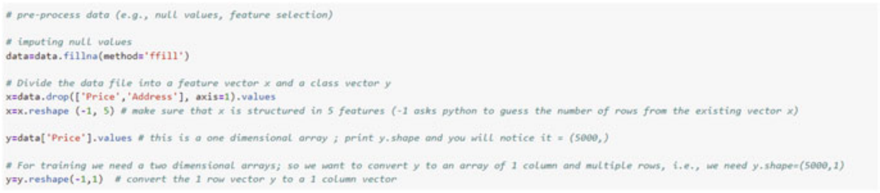
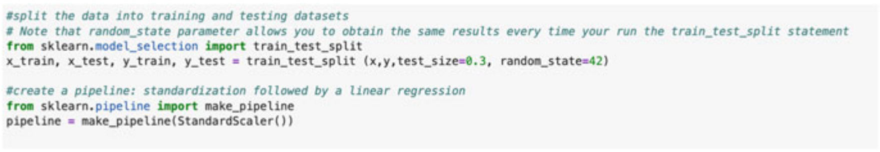


Fig. 6.17 Visualizing data graphically



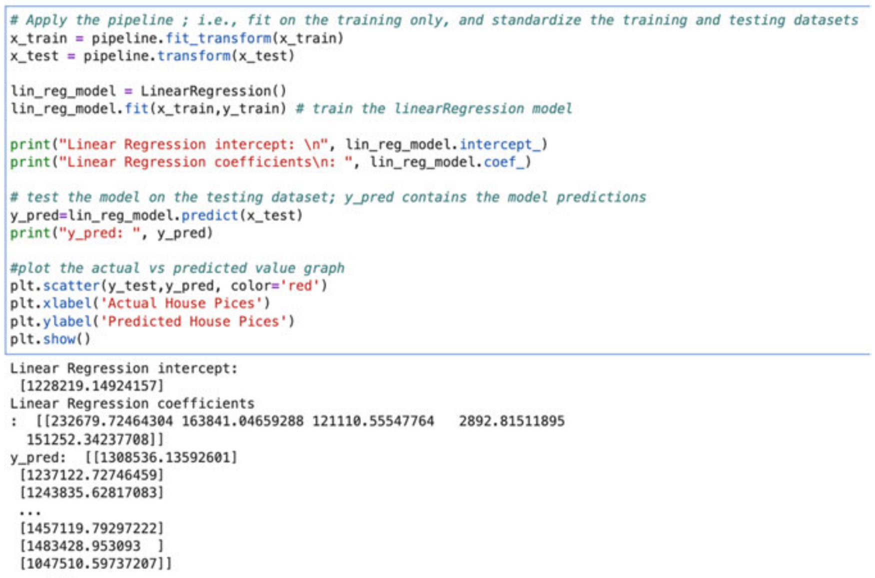
```
# pre-process data (e.g., null values, feature selection)

# imputing null values
data=data.fillna(method='ffill')

# Divide the data file into a feature vector x and a class vector y
x=data.drop(['Price','Address'], axis=1).values
x=x.reshape (-1, 5) # make sure that x is structured in 5 features (-1 asks python to guess the number of rows from the existing vector x)

y=data['Price'].values # this is a one dimensional array ; print y.shape and you will notice it = (5000,)

# For training we need a two dimensional arrays; so we want to convert y to an array of 1 column and multiple rows, i.e., we need y.shape=(5000,1)
y=y.reshape(-1,1)  # convert the 1 row vector y to a 1 column vector
```

Fig. 6.18 Preprocess housing data



```
#split the data into training and testing datasets
# Note that random_state parameter allows you to obtain the same results every time your run the train_test_split statement
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (x,y,test_size=0.3, random_state=42)

#create a pipeline: standardization followed by a linear regression
from sklearn.pipeline import make_pipeline
pipeline = make_pipeline(StandardScaler())
```

Fig. 6.19 Split and scale data



```
# Apply the pipeline ; i.e., fit on the training only, and standardize the training and testing datasets
x_train = pipeline.fit_transform(x_train)
x_test = pipeline.transform(x_test)

lin_reg_model = LinearRegression()
lin_reg_model.fit(x_train,y_train) # train the linearRegression model

print("Linear Regression intercept: \n", lin_reg_model.intercept_)
print("Linear Regression coefficients\n: ", lin_reg_model.coef_)

# test the model on the testing dataset; y_pred contains the model predictions
y_pred=lin_reg_model.predict(x_test)
print("y_pred: ", y_pred)

#plot the actual vs predicted value graph
plt.scatter(y_test,y_pred, color='red')
plt.xlabel('Actual House Pices')
plt.ylabel('Predicted House Pices')
plt.show()
```

```
Linear Regression intercept:
 [1228219.14924157]
Linear Regression coefficients
:  [[232679.72464304 163841.04659288 121110.55547764   2892.81511895
  151252.34237708]]
y_pred:  [[1308536.13592601]
 [1237122.72746459]
 [1243835.62817083]
 ...
 [1457119.79297222]
 [1483428.953093  ]
 [1047510.59737207]]
```

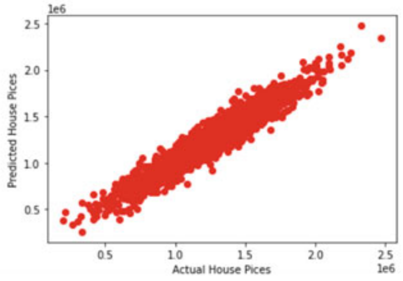


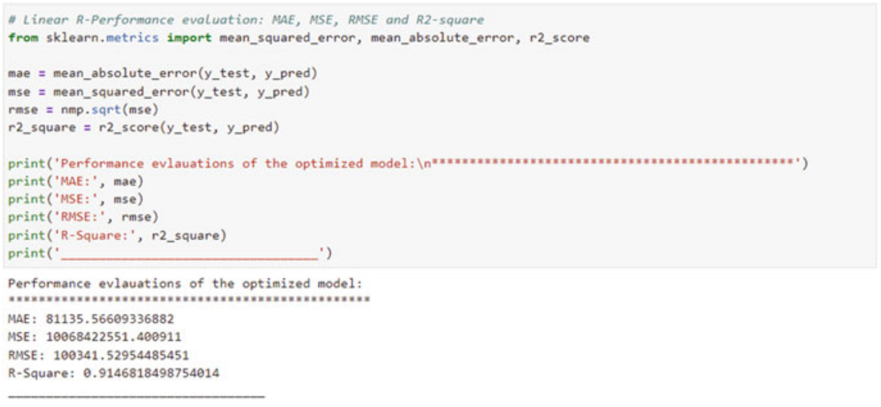
Fig. 6.20 Create and visualize a linear regression model

6.8.2.6 Evaluate Performance of LRM

In this section, we calculate the following metrics to evaluate the performance of the model using the training and testing model: MAE (mean absolute error), MSE (mean squared error), RMSE (root mean squared error), and R-square (accuracy). This is shown in Fig. 6.21.

6.8.2.7 Optimize LRM Manually with Gradient Descent

Linear regression in python has no hyperparameters to change and that can optimize a model. It has a few hyperparameters that you can check here. So, there will be no hyperparameter finetuning for this chapter, we will instead implement the gradient descent manually and in the next heading, we will implement it using Stochastic Gradient Descent (SGD) which is much faster (Fig. 6.25).



```
# Linear R-Performance evaluation: MAE, MSE, RMSE and R2-square
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = nmp.sqrt(mse)
r2_square = r2_score(y_test, y_pred)

print('Performance evlauations of the optimized model:\n************************************************')
print('MAE:', mae)
print('MSE:', mse)
print('RMSE:', rmse)
print('R-Square:', r2_square)
print('__________________________________')
```

```
Performance evlauations of the optimized model:
************************************************
MAE: 81135.56609336882
MSE: 10068422551.400911
RMSE: 100341.52954485451
R-Square: 0.9146818498754014
__________________________________
```

Fig. 6.21 Evaluating performance for LRM

We will start by modeling the linear regression using a gradient descent strategy (Fig. 6.22). We calculate the gradient (Fig. 6.23) and plot the number of iterations vs. the loss values and display predicted values against actual values (Fig. 6.24). We can notice that the error is very low and stabilized after 1000 iterations. This is a good opportunity to learn more about Python, you will need some time to understand the different functions defined to calculate the gradient descent (Figs. 6.22 and 6.23).

6.8.2.8 Create and Visualize a Model Using the Stochastic Gradient Descent (SGD)

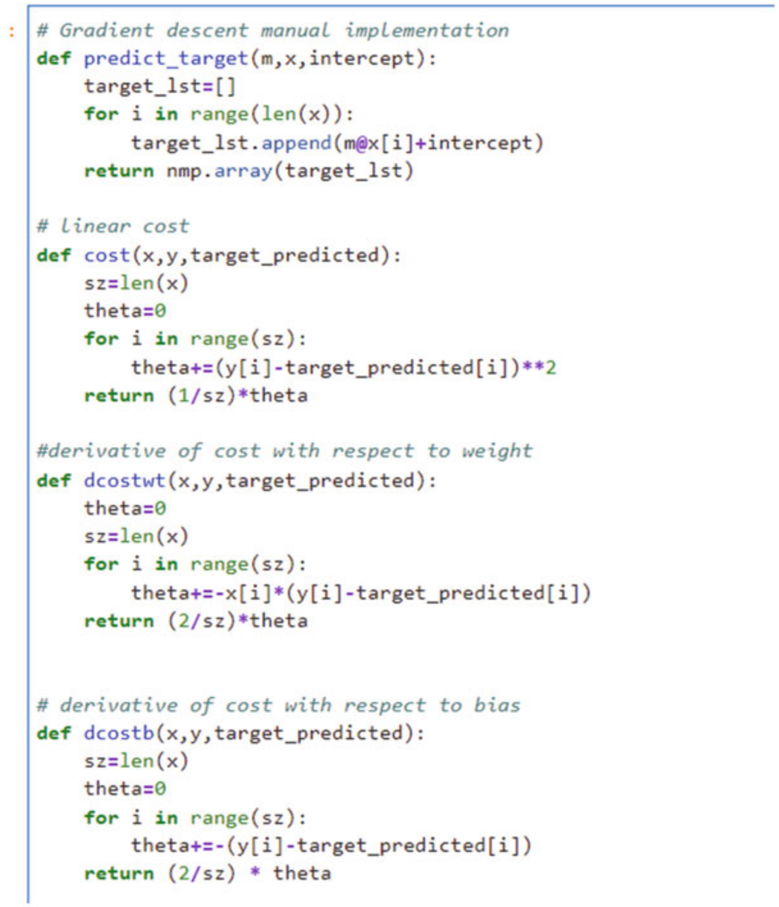
Alternatively to computing the gradient descent manually, we can use a gradient descent algorithm called Stochastic Gradient Descent (SGD) in Python (Fig. 6.25).

Finally, we measure the performance of the SDG model on the testing dataset (Fig. 6.26).

You can notice how fast SDG converges in comparison with the manual implementation. Why? Instead of computing the gradients for all instances of the training dataset as we did manually (i.e., something called batch gradient descent), SGD chooses a random instance of the dataset at every step and computes the gradients based on it! Hence the convergence speed of SGD, which is very useful with large training datasets.

Another method that could be used is the mini-batch gradient descent, which computes the gradients based on small sets (batches) of the dataset (something in between one instance and all the instances). we invite you to read about it.

Note: What about finetuning the SDGRegression hyperparameters? It will not be necessary for this chapter, but we invite you to explore the hyperparameters for both algorithms and try to finetune them.



```
# Gradient descent manual implementation
def predict_target(m,x,intercept):
    target_lst=[]
    for i in range(len(x)):
        target_lst.append(m@x[i]+intercept)
    return nmp.array(target_lst)

# linear cost
def cost(x,y,target_predicted):
    sz=len(x)
    theta=0
    for i in range(sz):
        theta+=(y[i]-target_predicted[i])**2
    return (1/sz)*theta

#derivative of cost with respect to weight
def dcostwt(x,y,target_predicted):
    theta=0
    sz=len(x)
    for i in range(sz):
        theta+=-x[i]*(y[i]-target_predicted[i])
    return (2/sz)*theta


# derivative of cost with respect to bias
def dcostb(x,y,target_predicted):
    sz=len(x)
    theta=0
    for i in range(sz):
        theta+=-(y[i]-target_predicted[i])
    return (2/sz) * theta
```

Fig. 6.22 Helper functions for gradient descent

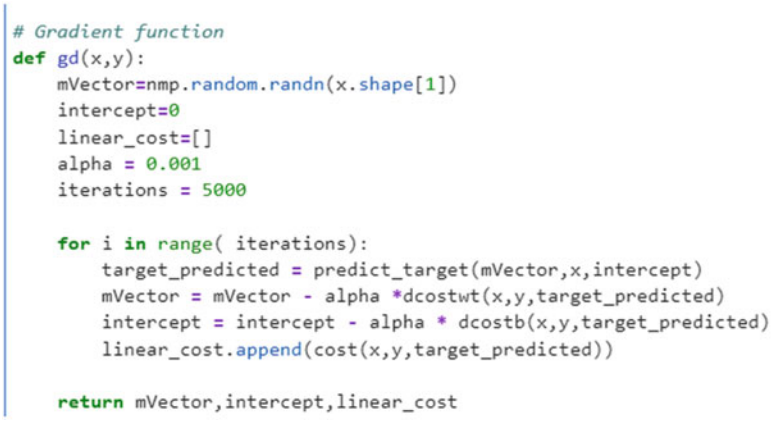
6.8.3 Working Example in Weka

Download the daily temperature of major cities of the world dataset from the following link: https://www.kaggle.com/sudalairajkumar/daily-temperature-of-major-cities.

The following example will analyze Cairo data. Open the csv file for Cairo (Fig. 6.27).

Go to the Classifier tab, click on the Choose button, and choose Linear Regression under functions (Fig. 6.28).

Click on the Linear Regression function, and the list of parameters will be displayed in a new window (Fig. 6.29); you can keep the parameter settings on the defaults. Notice that the dataset can be split into two parts: one to build the model and another to test it. We have chosen the cross-validation method instead. Cross-



```
# Gradient function
def gd(x,y):
    mVector=nmp.random.randn(x.shape[1])
    intercept=0
    linear_cost=[]
    alpha = 0.001
    iterations = 5000

    for i in range( iterations):
        target_predicted = predict_target(mVector,x,intercept)
        mVector = mVector - alpha *dcostwt(x,y,target_predicted)
        intercept = intercept - alpha * dcostb(x,y,target_predicted)
        linear_cost.append(cost(x,y,target_predicted))

    return mVector,intercept,linear_cost
```

Fig. 6.23 Gradient descent for LRM

validation splits the dataset into *k* parts or folds (e.g., 10), then it trains the model on all the folds except for one that it will use for testing. The process is repeated k times, and the average performance of all *k*-models is calculated.

The eliminateColinearAttributes parameters are set to True by default to detect and remove highly correlated input features. attributeSelectionMethod is enabled to perform feature selection so that only the attributes relevant to the outcome are selected, as unrelated attributes can negatively impact performance [1].

The result of the linear regression is shown in Fig. 6.30; we can note that the performance of the model is low.

6.8.4 Do It Yourself

6.8.4.1 Methods, Arguments, and Regularization

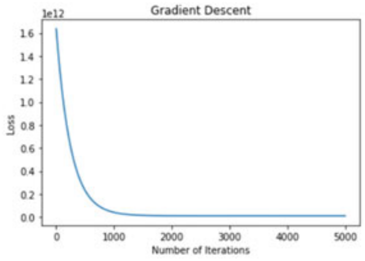
6.8.4.1.1 Methods and Arguments

Check the following code, and change your gd function accordingly.

```
# Gradient function
def gd(x,y, alpha = 0.001, iterations = 5000):
 mVector=nmp.random.randn(x.shape[1])
 intercept=0
 linear_cost=[]

 for i in range(iterations):
 target_predicted = predict_target(mVector,x,intercept)
 mVector = mVector - alpha *dsctdwt(x,y,target_predicted)
 intercept = intercept - alpha * dsctb(x,y,target_predicted)
 linear_cost.append(cost(x,y,target_predicted))
```

```
m,b,c=gd(x_train,y_train)

print("Weight:",m)
print("Bias:",b)
plt.title("Gradient Descent")
plt.plot(c)
plt.xlabel("Number of Iterations")
plt.ylabel("Loss")
```

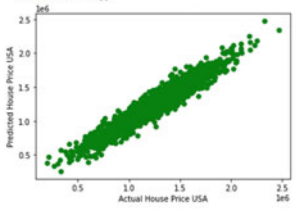
```
Weight: [232645.24688572 163822.93645244 120779.51235202   3224.8066755
 151249.38131995]
Bias: [1228163.94374749]
Text(0, 0.5, 'Loss')
```



```
def predict(inpt):
    target_list=[]
    sz= len(inpt)
    for i in range(sz):
        target_list.append(m@inpt[i]+b)
    return nmp.array(target_list)
```

```
y_pred=predict(X_test)
y_pred

print('Intercept: \n', lin_reg.intercept_)
print('Coefficients: \n', lin_reg.coef_)
plt.scatter(y_test, y_pred,  color='green')
plt.xlabel('Actual House Price USA')
plt.ylabel('Predicted House Price USA')
#plt.plot(y_test, y_pred, color='red', linewidth=2)
plt.show()
```

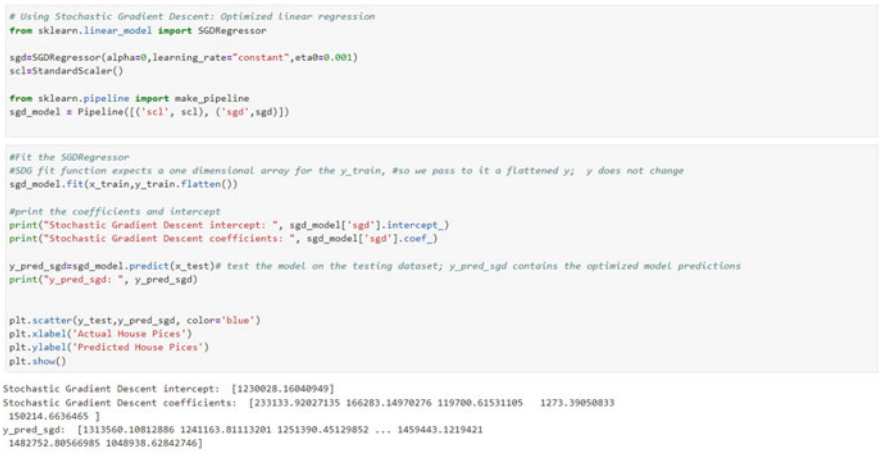
```
Intercept:
 [1228219.14924157]
Coefficients:
 [[232679.72464304 163841.04659288 121110.55547764   2892.81511895
  151252.34237708]]
```



```
from sklearn.metrics import  mean_squared_error, mean_absolute_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = nmp.sqrt(mse)
r2_square = r2_score(y_test, y_pred)
print('Performance evlauations of the LRM with gradient descent:\n************************************************')
print('MAE:', mae)
print('MSE:', mse)
print('RMSE:', rmse)
print('R-Square:', r2_square)
print('__________________________________')
```

```
Performance evlauations of the LRM with gradient descent:
************************************************
MAE: 81145.26861202244
MSE: 10070790240.217175
RMSE: 100353.32700123686
R-Square: 0.9146617864713434
__________________________________
```

Fig. 6.24 Gradient descent graph for LRM and corresponding



```
# Using Stochastic Gradient Descent: Optimized linear regression
from sklearn.linear_model import SGDRegressor

sgd=SGDRegressor(alpha=0,learning_rate="constant",eta0=0.001)
scl=StandardScaler()

from sklearn.pipeline import make_pipeline
sgd_model = Pipeline([('scl', scl), ('sgd',sgd)])
```

```
#Fit the SGDRegressor
#SDG fit function expects a one dimensional array for the y_train, #so we pass to it a flattened y;  y does not change
sgd_model.fit(x_train,y_train.flatten())

#print the coefficients and intercept
print("Stochastic Gradient Descent intercept: ", sgd_model['sgd'].intercept_)
print("Stochastic Gradient Descent coefficients: ", sgd_model['sgd'].coef_)

y_pred_sgd=sgd_model.predict(x_test)# test the model on the testing dataset; y_pred_sgd contains the optimized model predictions
print("y_pred_sgd: ", y_pred_sgd)


plt.scatter(y_test,y_pred_sgd, color='blue')
plt.xlabel('Actual House Pices')
plt.ylabel('Predicted House Pices')
plt.show()
```

```
Stochastic Gradient Descent intercept:  [1230028.16040949]
Stochastic Gradient Descent coefficients:  [233133.92027135 166283.14970276 119700.61531105   1273.39050833
 150214.6636465 ]
y_pred_sgd:  [1313560.10812886 1241163.81113201 1251390.45129852 ... 1459443.1219421
 1482752.80566985 1048938.62842746]
```

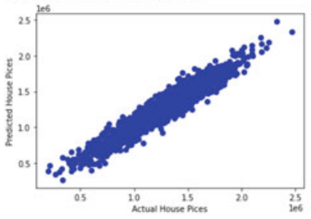
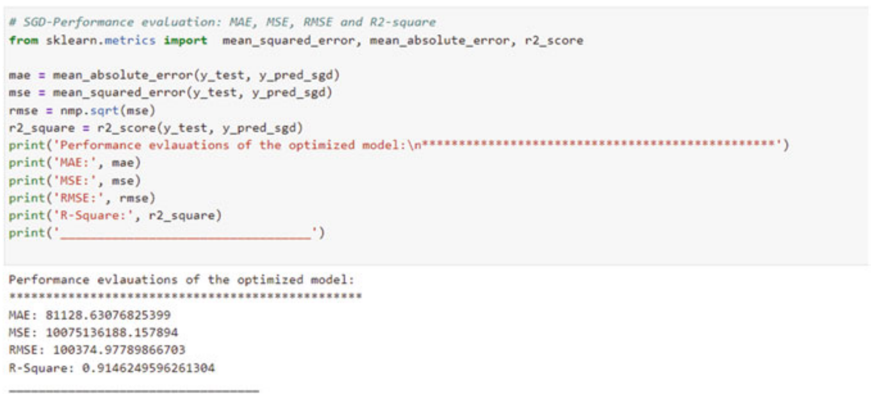


Fig. 6.25 Helper functions for gradient descent



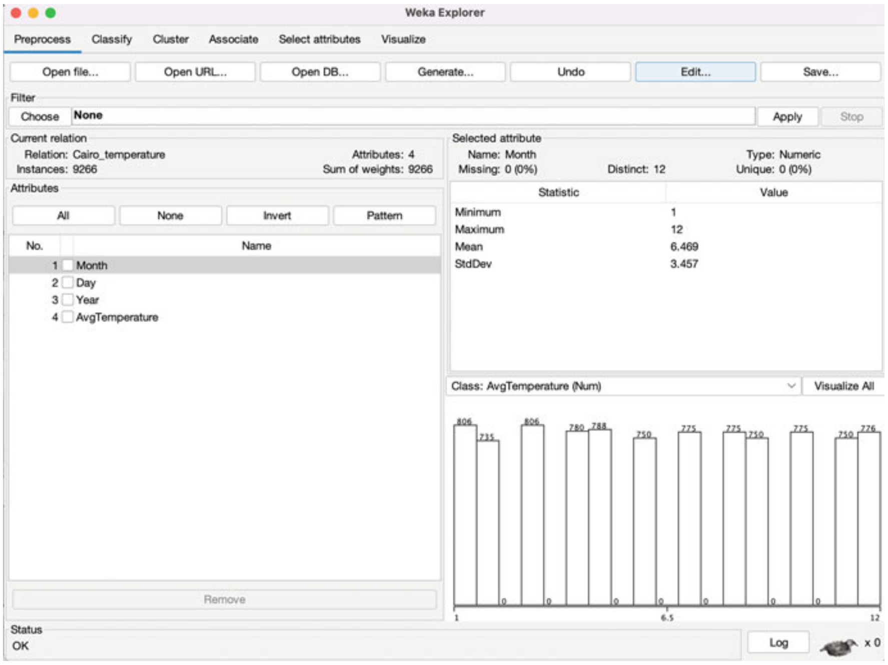
```
# SGD-Performance evaluation: MAE, MSE, RMSE and R2-square
from sklearn.metrics import  mean_squared_error, mean_absolute_error, r2_score

mae = mean_absolute_error(y_test, y_pred_sgd)
mse = mean_squared_error(y_test, y_pred_sgd)
rmse = nmp.sqrt(mse)
r2_square = r2_score(y_test, y_pred_sgd)
print('Performance evlauations of the optimized model:\n************************************************')
print('MAE:', mae)
print('MSE:', mse)
print('RMSE:', rmse)
print('R-Square:', r2_square)
print('__________________________________')
```

```
Performance evlauations of the optimized model:
************************************************
MAE: 81128.63076825399
MSE: 10075136188.157894
RMSE: 100374.97789866703
R-Square: 0.9146249596261304
__________________________________
```

Fig. 6.26 performance of the optimized model measured using the training dataset

```
return mVector,intercept,linear_cost
```

(a) What is exactly the difference with the previous gd code?
(b) Write the python code to call this function?
(c) Try this code to call the gd method.
    m,b,c = gd(x_train,y_train, alpha = 0.01, iterations = 200)



No.	Name
1	Month
2	Day
3	Year
4	AvgTemperature

Statistic	Value
Minimum	1
Maximum	12
Mean	6.469
StdDev	3.457

Fig. 6.27 Weka preprocessing screen for Cairo

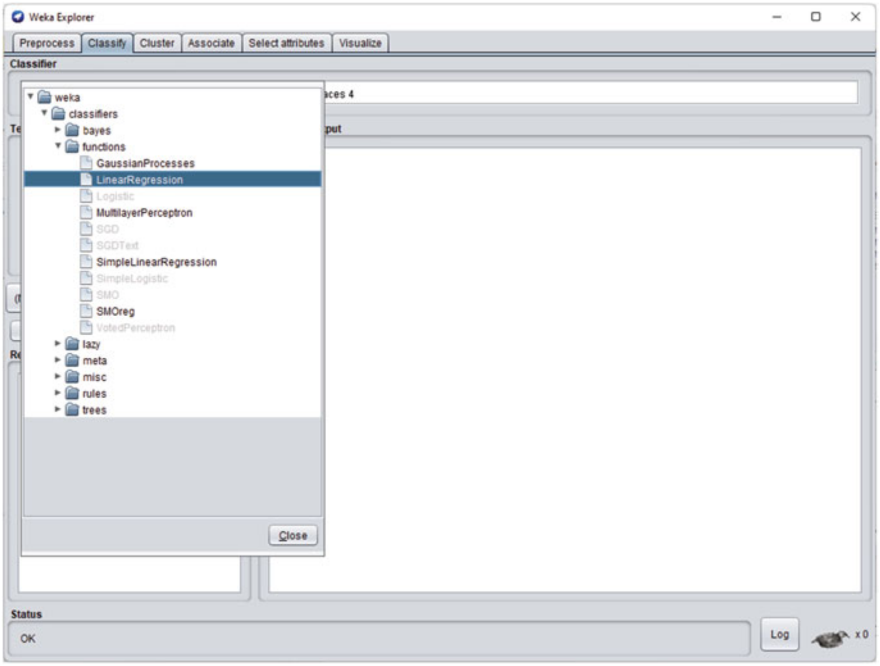
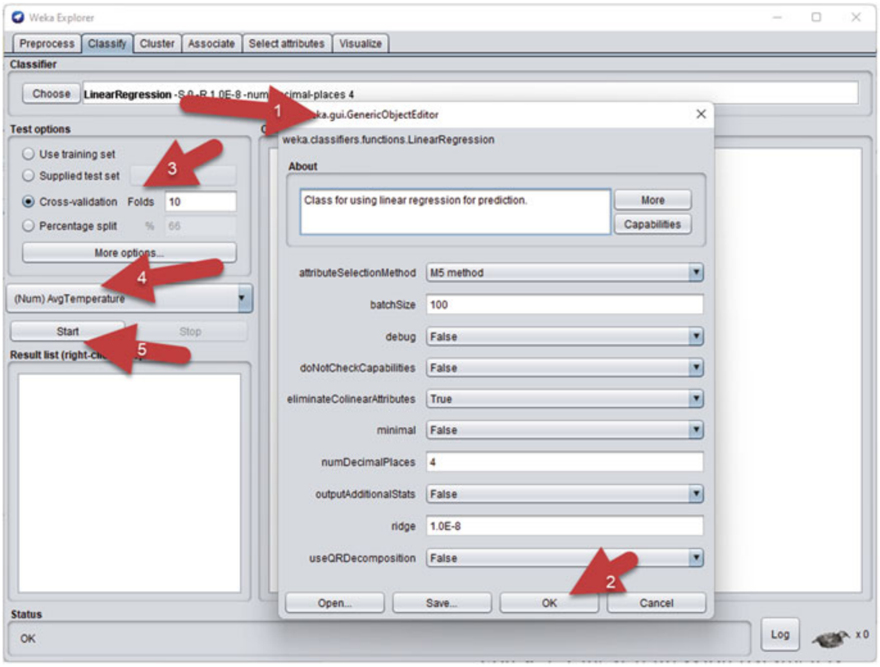


Fig. 6.28 Choosing Linear Regression in Weka



LinearRegression -S 0 -R 1.0E-8 -num imal-places 4

ka.gui.GenericObjectEditor

weka.classifiers.functions.LinearRegression

Class for using linear regression for prediction.

attributeSelectionMethod	M5 method
batchSize	100
debug	False
doNotCheckCapabilities	False
eliminateColinearAttributes	True
minimal	False
numDecimalPlaces	4
outputAdditionalStats	False
ridge	1.0E-8
useQRDecomposition	False

Cross-validation Folds 10

Percentage split % 66

(Num) AvgTemperature

Status: OK

Fig. 6.29 Parameter settings for the linear regression

(d) Comment on the convergence of the gradient descent. Did it converge earlier or later than above? Why?

(e) How would you change the argument alpha to make it converge at a slower pace? Try few values.

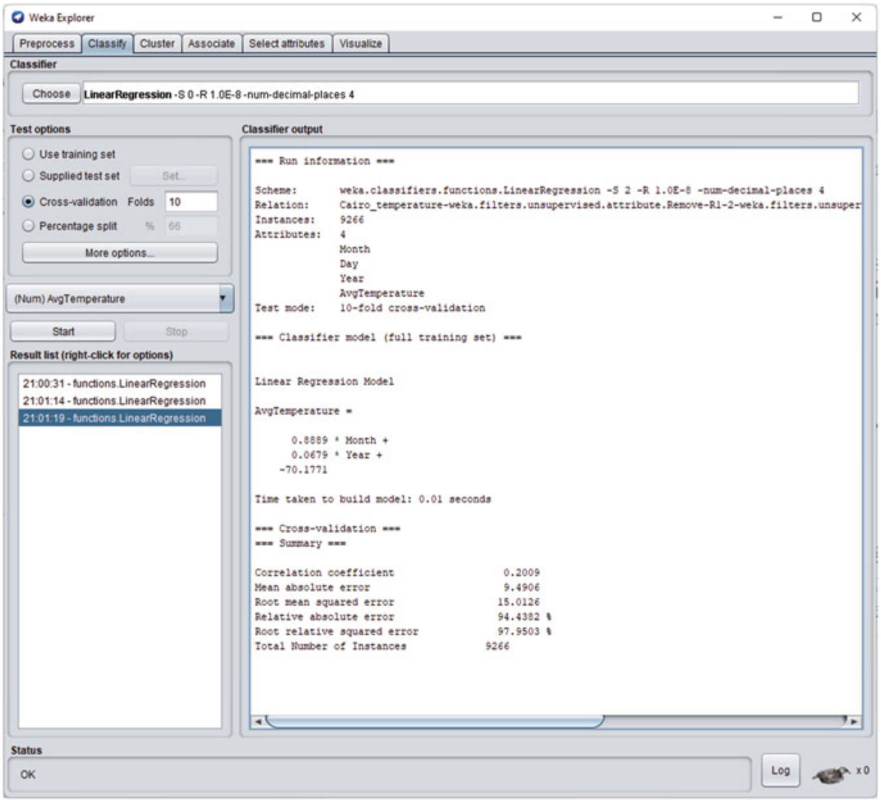
(f) If you changed alpha to be larger, would you need to change the number of iterations too? In which direction (higher or lower)? Give a rationale for your choice. Of course, you can try few values and check the results.

6.8.4.1.2 Regularization

To avoid overfitting, we can regularize a linear model using ridge regression, lasso regression, elastic net, as well as early stopping.

In python you can use:

1. Ridge algorithm or use SDGRegressor with penalty = "l2" to implement ridge regularization.
2. Lasso algorithm or use SDGRegressor with penalty = "l1" to implement lasso regularization.
3. ElasticNet algorithm or use SDGRegressor with penalty = "elasticnet" to implement elastic net regularization.



LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Cross-validation Folds 10

Percentage split % 66

(Num) AvgTemperature

21:00:31 - functions.LinearRegression
21:01:14 - functions.LinearRegression
21:01:19 - functions.LinearRegression

```
=== Run information ===

Scheme:       weka.classifiers.functions.LinearRegression -S 2 -R 1.0E-8 -num-decimal-places 4
Relation:     Cairo_temperature-weka.filters.unsupervised.attribute.Remove-R1-2-weka.filters.unsuper
Instances:    9266
Attributes:   4
              Month
              Day
              Year
              AvgTemperature
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===


Linear Regression Model

AvgTemperature =

      0.8889 * Month +
      0.0679 * Year +
    -70.1771

Time taken to build model: 0.01 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.2009
Mean absolute error                      9.4906
Root mean squared error                 15.0126
Relative absolute error                 94.4382 %
Root relative squared error             97.9503 %
Total Number of Instances             9266
```

Fig. 6.30 Result of the linear regression for the Cairo dataset

4. Early stopping involves stopping the iterations/epochs as soon as the validation error (RMSE) on the validation dataset reaches a minimum and starts increasing which indicates overfitting. You need to stop at the minimum error.

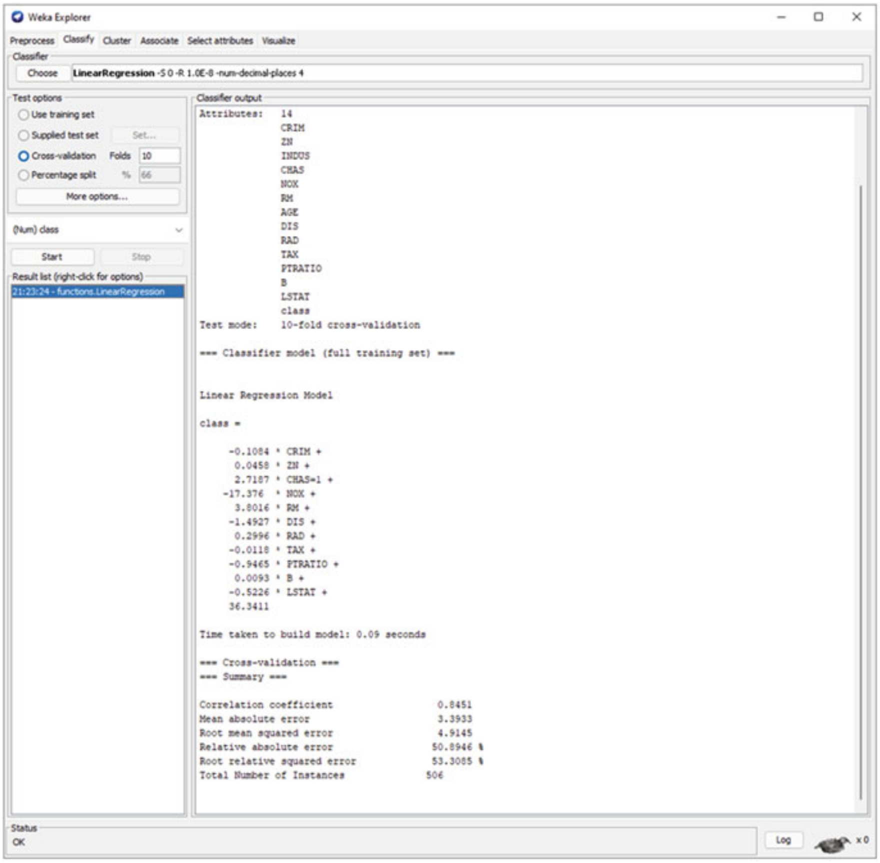
We invite you to try the three regularizations methods on the datasets encountered above as well as on different ones (check Do more Yourself below).

6.8.4.2 Predicting House Prices

You can also download 37 regression problems compressed into one file called *datasets-numeric.jar* from https://waikato.github.io/weka-wiki/datasets/.

You need to install the Java Development Kit (JDK) to unpack it. You can download JDK from https://www.oracle.com/java/technologies/javase-downloads.html. Once it is installed, you can unpack the .jar file by writing on the command line: jar -xvf datasets-numeric.jar

You can also use software like WinZip or WinRAR to unpack the jar file.



LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Folds: 10

21:23:24 - functions.LinearRegression

```
Attributes:   14
              CRIM
              ZN
              INDUS
              CHAS
              NOX
              RM
              AGE
              DIS
              RAD
              TAX
              PTRATIO
              B
              LSTAT
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===


Linear Regression Model

class =

     -0.1084 * CRIM +
      0.0458 * ZN +
      2.7187 * CHAS=1 +
    -17.376  * NOX +
      3.8016 * RM +
     -1.4927 * DIS +
      0.2996 * RAD +
     -0.0118 * TAX +
     -0.9465 * PTRATIO +
      0.0093 * B +
     -0.5226 * LSTAT +
     36.3411

Time taken to build model: 0.09 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8451
Mean absolute error                      3.3933
Root mean squared error                  4.9145
Relative absolute error                 50.8946 %
Root relative squared error             53.3085 %
Total Number of Instances              506
```

Fig. 6.31 Boston housing market linear regression result

Get the Boston housing market dataset from the following datasets-numeric.jar file (Housing.arff). Each instance of the dataset has 13 numerical input features that depict the properties of a Boston suburb; your task is to predict the house prices. Generate a linear regression model of the housing market; your model should generate the figure in Fig. 6.31.

6.8.5 *Do More Yourself*

Create simple and multiple linear regression models in Python and R using the datasets below. Ensure that you preprocess the data: remove null values, transform categorical data, and scale the data if necessary. Visualize the model. Train and test the model. Optimize the model using gradient descent and conclude whether it can be generalized to an unseen dataset.

1. From the temperature dataset above, choose a city and compute a simple/multiple linear regression model for that city. Predict the temperature in that city in 2030, 2050, 2070, and 2100.
2. California housing market dataset: https://www.kaggle.com/camnugent/california-housing-prices
3. Real estate price prediction https://www.kaggle.com/quantbruce/real-estate-price-prediction
4. Ecommerce: https://www.kaggle.com/kolawale/focusing-on-mobile-app-or-website
5. Medical cost personal datasets: https://www.kaggle.com/mirichoi0218/insurance
6. Fish market: https://www.kaggle.com/aungpyaeap/fish-market

References

1. J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement Them from Scratch* (Jason Brownlee, 2016)
2. S. Ravichandiran, *Hands-On Deep Learning Algorithms with Python: Master Deep Learning Algorithms with Extensive Math by Implementing Them Using TensorFlow* (Packt Publishing, 2019)
3. M. Gopal, *Applied Machine Learning* (McGraw-Hill Education, 2018)
4. P.J. Deitel, H. Deitel, *Python for Programmers: With intodoctory AI Case Studies* (Pearson Education, 2019)

Chapter 7
Logistic Regression

7.1 The Problem

Linear regression modeling is well suited to predicting continuous data where the outcome y is a real number (i.e., $y \in \mathbb{R}$). Logistic regression is a modeling technique for binary outcomes (i.e., yes/no, true/false, 1/0). Such outcomes are needed in many domains: public health officials might want to know the likelihood that a person will contract COVID-19 if she is a doctor in Ontario; a hospital would like to know if a discharged patient is more likely to be readmitted or not; a company would like to know if a customer visiting its website is more likely to order; a bank would like to know if a customer is more likely to default on a loan or not. Logistic regression has been much used in the medical field and yielded impressive results [1–10].

Logistic regression helps us make that type of prediction. We can think of it as a classification problem, the outcome being a choice between two classes, the question asked is: given x, what is the probability that the outcome belongs to class 1 (i.e., $P(\text{class} = 1|x)$)? And given that we have only two classes, the probability that the outcome belongs to class 0 is $P(\text{class} = 0|x) = 1 - P(\text{class} = 1|x)$.

7.2 A Practical Example

The outcome we are trying to predict is a probability (the likelihood that an outcome belongs to class 1 or class 0). Linear regression cannot solve this type of problem because its outcome is a number that can take any value and not only two values, 0 or 1.

Suppose we have administered a survey and collected the number of steps people walk in a day and if they had a heart attack (coded 0) or not (coded 1) during their

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_7

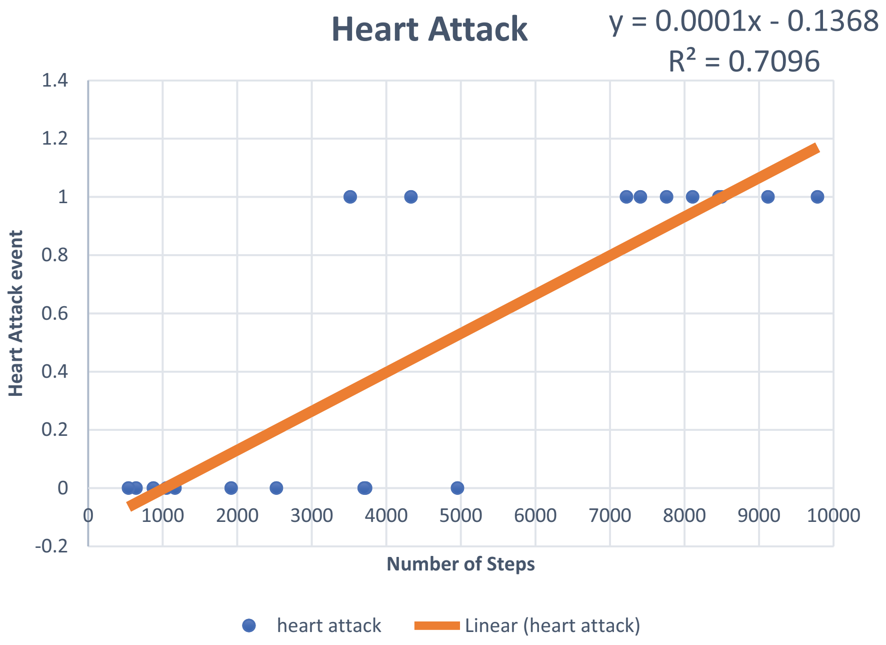


Fig. 7.1 A linear regression line fitted to the data points

lifetime; we want to predict if the number of steps taken in a day is related to the risk of a heart attack. We plot our fictitious data in Fig. 7.1.

We can notice that fewer steps are related to a heart attack but that the linear regression does not fit the data well. Also, the linear regression predicts probabilities of less than 0 below 100 steps and more than 1 for 9000 steps.

We instead use another function to approximate a good prediction; the logistic function provides an output that falls between 0 and 1: $0 \leq y \leq 1$. An example of a logistic function is $y = \frac{1}{1-e^{-x}}$ (Fig. 7.2), where e is the base of the natural logarithm ($e = 2.71828$, $\mathrm{Log}(e) = 1$). Generally, we can write the logistic function for a feature vector X and outcome vector Y with the coefficient W: $Y = \frac{1}{1-e^{-WX}}$.

7.3 The Algorithm

The S shape of the logistic function and the variation of its output between 0 and 1 makes it perfect for classification into two classes, 0 and 1. It is enough to take a certain threshold, for example, 0.5, above which we classify the outcome as belonging to class 1; otherwise, it belongs to class 0.

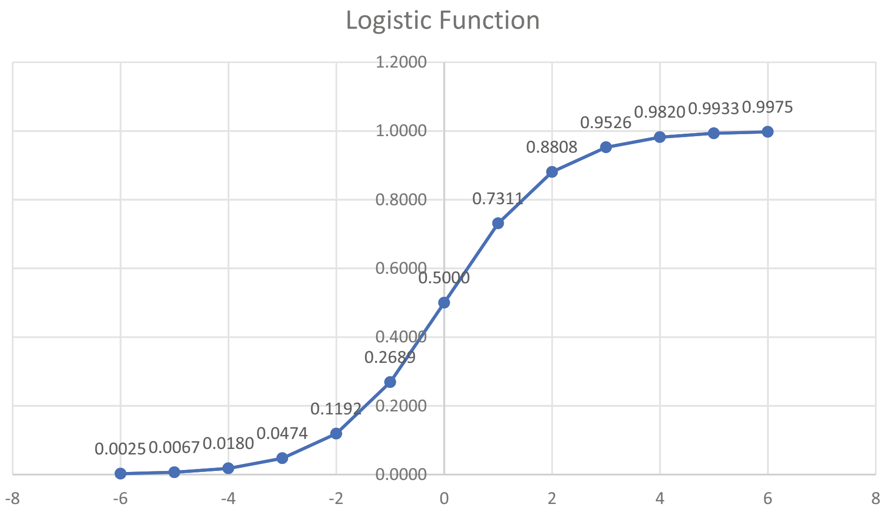


Fig. 7.2 The logistic function for data varying between −6 and + 6

The only problem is that the logistic function, also called the sigmoid function, is not linear, and we are interested in having a linear relationship between the outcome Y and the input X because it is simple to interpret.

To convert the logistic function to a linear function, we can compute the odds of the function and then take the Napierian logarithm of the odds.

The odds of an outcome are calculated as the probability of an outcome occurring divided by the probability that the outcome does not occur. For the logistic function, $\text{odds}(y) = \frac{P(y=1|x)}{1-P(y=1|x)}$ and is a value between 0 and infinity.

The $\log_n$ of the odds, also known as *logit*, is computed as $\log_n(\text{odds}(y)) = \log_n\left(\frac{P(y=1|x)}{1-P(y=1|x)}\right)$ is a function of X with an outcome that belongs to $\mathbb{R}$ that can be represented as a linear function of X. Indeed:

$$\log_n\left(\frac{P(y=1|x)}{1-P(y=1|x)}\right) = \log_n\left(\frac{\frac{1}{1-e^{-WX}}}{1-\frac{1}{1-e^{-WX}}}\right) = \log_n\left(\frac{\frac{1}{1-e^{-WX}}}{\frac{e^{WX}}{1-e^{-WX}}}\right)$$

$$= \log_n \log\left(\frac{1}{e^{-WX}}\right) = \log_n\left(e^{-1*(-WX)}\right) = \log_n\left(e^{WX}\right) = WX$$

$\text{Logit}(y) = \log_n(\text{odds}(y)) = WX = w_0x_0 + w_1x_1 + \ldots + w_nx_n$, which is a linear equation! Having a linear function, it is easy now to find the weights w_i for which the logit function best fits our dataset (i.e., with a minimal cost). Once those weights are found, we can compute $y = P(\text{class} = 1|x)$ as follows:

$$y = P(\text{class} = 1|x) = \frac{1}{1 + e^{-\left(w_0x_0 + w_1x_1 \ldots. + w_nx_n\right)}}$$

$$y = P(\text{class} = 0|x) = 1 - P(\text{class} = 1|x) = \frac{e^{-\left(w_0x_0 + w_1x_1 \ldots. + w_nx_n\right)}}{1 + e^{-\left(w_0x_0 + w_1x_1 \ldots. + w_nx_n\right)}}$$

7.4 Final Notes: Advantages, Disadvantages, and Best Practices

The interpretation of the weights W_i in the logistic regression is different than in linear regression.

$$\text{odds}(y) = \log_n\left(\frac{P(y = 1|x)}{1 - P(y = 1|x)}\right) = WX = w_0x_0 + w_1x_1 \ldots. + w_nx_n$$

The formula shows a linear model for the log odds, but that is not very useful. To make it useful, we can check how the prediction changes with every one unit change in a feature X. If we apply the exponential function to the previous formula, we obtain the following

$$\text{odds}(y) = \frac{P(y = 1|x)}{1 - P(y = 1|x)} = e^{(w_0x_0 + w_1x_1 \ldots. + w_nx_n)}$$

Let compute the odds (y) when a feature $\mathrm{x_j}$ changes by 1 unit and becomes $x_j + 1$. Given that $\frac{e^a}{e^b} = e^{(a-b)})$ we can compute the ratio of the new odds (related to $x_j + 1$) divided by the previous odds (y) (related to x_j):

$$\frac{\text{odds}\left(y_{\mathrm{xj}+1}\right)}{\text{odds}\left(y_{\mathrm{xj}}\right)} = \frac{e^{\left(w_0x_0 + w_1x_1 \ldots w_1\left(x_j+1\right)\ldots + w_nx_n\right)}}{e^{\left(w_0x_0 + w_1x_1 \ldots w_1\left(x_j\right)\ldots + w_nx_n\right)}}$$

$$= e^{\left(w_0x_0 + w_1x_1 \ldots w_j\left(x_j+1\right)\ldots + w_nx_n\right) - \left(w_0x_0 + w_1x_1 \ldots w_jx_j \ldots + w_nx_n\right)}$$

$$\frac{\text{odds}\left(y_{x_j+1}\right)}{\text{odds}\left(y_{x_j}\right)} = e^{\left(w_j\left(x_j+1\right) - w_1x_j)\right)} = e^{w_j}$$

This is simple to interpret; for every increase in one unit in one feature x_j the odds ratio of the outcome will be multiplied by a factor of e^{w_j}. So, we have a simple function—the exponential function—that allows us to compute for any change in one feature by one unit how much there is a change in the odds ratio.

For a quick example. If odds(y) = 2; that means that the probability of $y = 1$ (the outcome is 1) is twice as high as the probability of $y = 0$. If a change in a feature by one unit leads to a change in the odds ratio by 5, that means a change in a feature by one unit leads to having the probability of $y = 1$ (the outcome is 1) multiplied by fivefolds.

Finally, note that preprocessing is always important before building your model. You can

- **Check for outliers and remove them** when necessary.
- **Remove correlated variables:** If the input features are highly correlated, you might face model overfitting. A good practice is to compute the pairwise correlations between all input features.
- **Consider imputing missing values:** A lot of missing values might lead to non-convergence of your model [11].
- **One-hot-encode your categorical variables**

7.5 Key Terms

1. Binary outcomes
2. Logistic function
3. Sigmoid function
4. Odds
5. Logit
6. Napierian logarithm

7.6 Test Your Understanding

1. What is the major difference between logistic regression and linear regression? How do you know which one to choose for a certain problem?
2. What similarities do you find when comparing linear regression and logistic regression?
3. What is the type of outcome for logistic regression?
4. Can we use a logistic regression model to predict a value (i.e., a number) in the future, or is it only used for classification?

7.7 Read More

1. Arkin, F. S., Aras, G., & Dogu, E. (2020). Comparison of Artificial Neural Networks and Logistic Regression for 30-days Survival Prediction of Cancer Patients. Acta Inform Med, 28(2), 108–113. https://doi.org/10.5455/aim.2020.28.108-113
2. Chen, C. C. M., Schwender, H., Keith, J., Nunkesser, R., Mengersen, K., & Macrossan, P. (2011). Methods for Identifying SNP Interactions: A Review on Variations of Logic Regression, Random Forest and Bayesian Logistic Regression. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8(6), 1580–1591. https://doi.org/10.1109/TCBB.2011.46
3. Cheng, Q., Varshney, P. K., & Arora, M. K. (2006). Logistic Regression for Feature Selection and Soft Classification of Remote Sensing Data. IEEE Geoscience and Remote Sensing Letters, 3(4), 491–494. https://doi.org/10.1109/LGRS.2006.877949
4. Cowling, T. E., Cromwell, D. A., Bellot, A., Sharples, L. D., & van der Meulen, J. (2021). Logistic regression and machine learning predicted patient mortality from large sets of diagnosis codes comparably. J Clin Epidemiol, 133, 43–52. https://doi.org/10.1016/j.jclinepi.2020.12.018
5. Elkadiri, R., Sultan, M., Youssef, A. M., Elbayoumi, T., Chase, R., Bulkhi, A. B., & Al-Katheeri, M. M. (2014). A Remote Sensing-Based Approach for Debris-Flow Susceptibility Assessment Using Artificial Neural Networks and Logistic Regression Modeling. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7(12), 4818–4835. https://doi.org/10.1109/JSTARS.2014.2337273
6. Hajipour, F., Jozani, M. J., & Moussavi, Z. (2020). A comparison of regularized logistic regression and random forest machine learning models for daytime diagnosis of obstructive sleep apnea. Med Biol Eng Comput, 58(10), 2517–2527. https://doi.org/10.1007/s11517-020-02206-9
7. Khurshid, H., & Khan, M. F. (2015). Segmentation and Classification Using Logistic Regression in Remote Sensing Imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 8(1), 224–232. https://doi.org/10.1109/JSTARS.2014.2362769
8. Ksantini, R., Ziou, D., Colin, B., & Dubeau, F. (2008). Weighted Pseudometric Discriminatory Power Improvement Using a Bayesian Logistic Regression Model Based on a Variational Method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2), 253–266. https://doi.org/10.1109/TPAMI.2007.1165
9. Subagia, R., Saleh, J. H., Churchwell, J. S., & Zhang, K. S. (2020). Statistical learning for turboshaft helicopter accidents using logistic regression. PLoS One, 15(1), e0227334. https://doi.org/10.1371/journal.pone.0227334
10. Yang, J. H., Park, J. H., Jang, S. H., & Cho, J. (2020). Novel Method of Classification in Knee Osteoarthritis: Machine Learning Application Versus

Logistic Regression Model. Ann Rehabil Med, 44(6), 415–427. https://doi.org/10.5535/arm.20071

7.8 Lab

7.8.1 *Working Example in Python*

We will be using the Pima Indian Diabetes dataset that you can download from: https://www.kaggle.com/kumargh/pimaindiansdiabetescsv

This dataset relates to the risk of diabetes within 5 years and has the following information:

- Preg: number of pregnancies
- Plas: plasma glucose concentration
- Pres: diastolic blood pressure measurement
- Skin: triceps skinfold thickness (mm)
- Test: 2-hour serum insulin
- Mass: body mass index (BMI)
- Pedi: diabetes pedigree function
- Age: age
- Class: tested positive for diabetes or not (1 or 0)

7.8.1.1 Load Pima Indians Diabetes Dataset

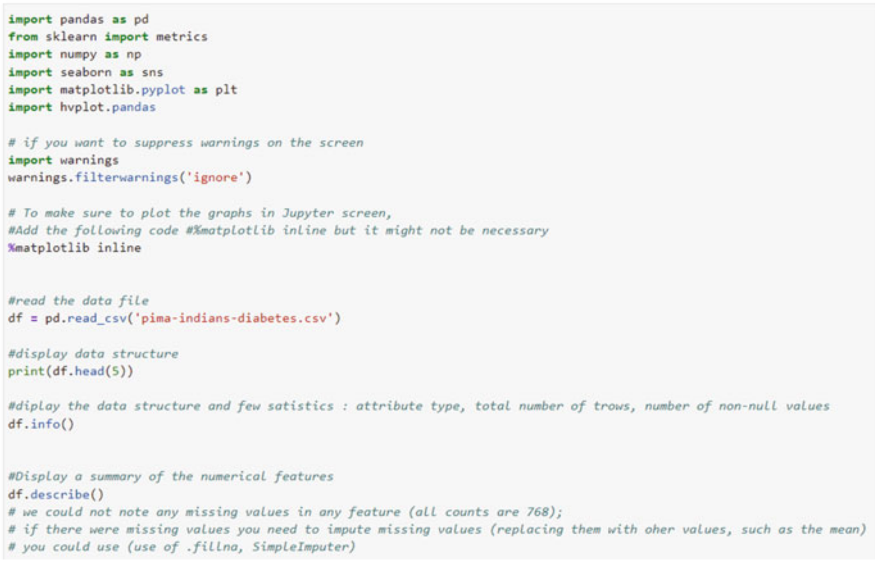
Start by importing the required libraries. Install any library that was not installed yet (e.g., hvplot), using the pip install command; then load the dataset into pandas (Fig. 7.3).

7.8.1.2 Visualize Pima Indians Dataset

Visualize the data using hist() from pandas and the pairplot() from the seaborn (Fig. 7.4). It is important to note that Python libraries have different methods to create different types of visualizations, such as bar charts, scatter charts, histograms. We can notice that the two classes are not balanced, there are much more people that are diabetes-free than those diagnosed with diabetes.

7.8.1.3 Preprocess Data

It is necessary to check for null values (if any). When we used df.info() we could notice that there were no missing values in any feature (all the counts were at 768). If there were missing values, we could impute missing values using of .fillna() (using



```
import pandas as pd
from sklearn import metrics
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas

# if you want to suppress warnings on the screen
import warnings
warnings.filterwarnings('ignore')

# To make sure to plot the graphs in Jupyter screen,
#Add the following code #%matplotlib inline but it might not be necessary
%matplotlib inline


#read the data file
df = pd.read_csv('pima-indians-diabetes.csv')

#display data structure
print(df.head(5))

#diplay the data structure and few satistics : attribute type, total number of trows, number of non-null values
df.info()


#Display a summary of the numerical features
df.describe()
# we could not note any missing values in any feature (all counts are 768);
# if there were missing values you need to impute missing values (replacing them with oher values, such as the mean)
# you could use (use of .fillna, SimpleImputer)
```

Fig. 7.3 Load Pima Indians diabetes dataset

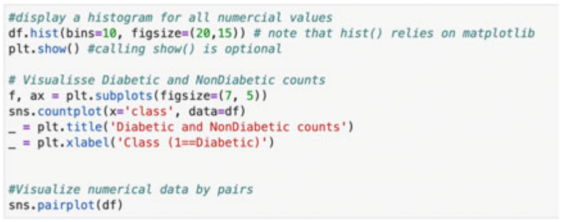
method = ‘backfill’ or ‘ffill’) or the SimpleImputer class (with the parameter strategy = ‘mean’).

For illustration purposes only, we opted to use “fill forward” missing values which propagate the last valid value forward to the next missing value. Then we separated the features used for prediction and the target class. And split the dataset into training and testing datasets (Fig. 7.5).

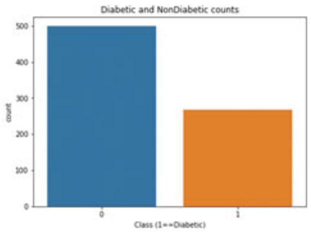
Then we will conduct standardization on the dataset. First, we will fit the Standardization on the training dataset so that it learns the averages and standard deviations for the features in the training dataset. Then we will transform (i.e., standardize) both training and testing datasets based on those learned training averages and standard deviations. We will never fit standardization on the testing dataset as it will bias the testing step, the testing dataset should be used to test the model based on information available only in the training phase. In the end, the logistic regression algorithm is prepared for use.

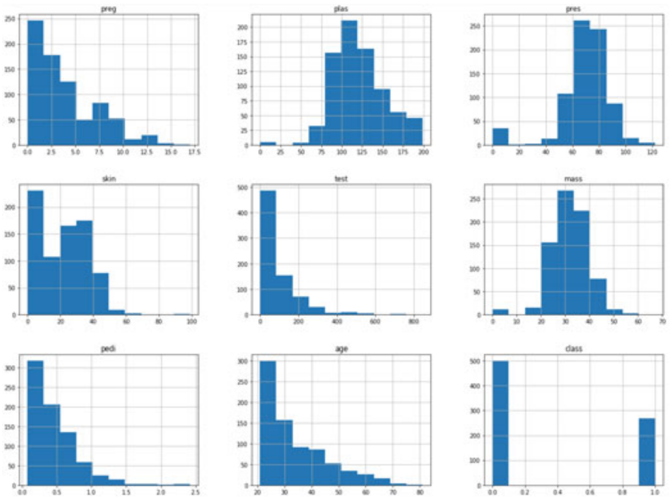
7.8.1.4 Optimize Logistic Regression Model

It is time to find the optimal model that fits the training dataset. The list of parameters for the logistic regression includes the regularization technique used (l1, l2, or elasticnet), the C value, the solver, and the number of iterations. The Gridsearch provides a few values for each hyperparameter. A cross-validation of five folds is specified. The variable optimalmodel holds the model found by the Gridsearch once it runs the .fit() method using the prepared training data (x_train_prepared and



```
#display a histogram for all numercial values
df.hist(bins=10, figsize=(20,15)) # note that hist() relies on matplotlib
plt.show() #calling show() is optional

# Visualisse Diabetic and NonDiabetic counts
f, ax = plt.subplots(figsize=(7, 5))
sns.countplot(x='class', data=df)
_ = plt.title('Diabetic and NonDiabetic counts')
_ = plt.xlabel('Class (1==Diabetic)')


#Visualize numerical data by pairs
sns.pairplot(df)
```





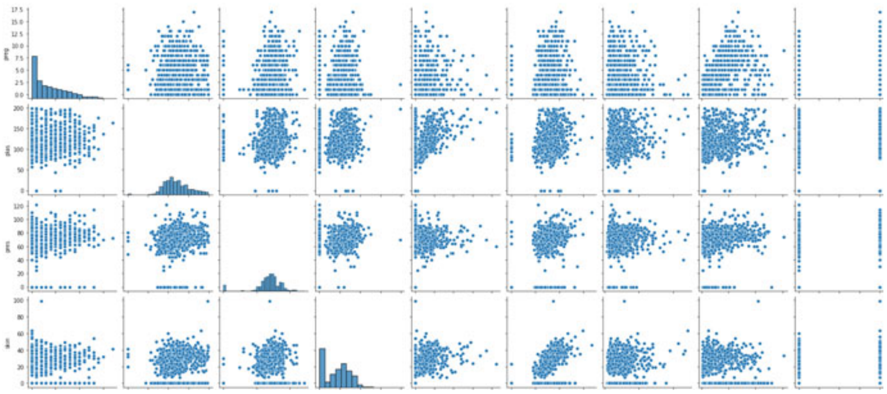
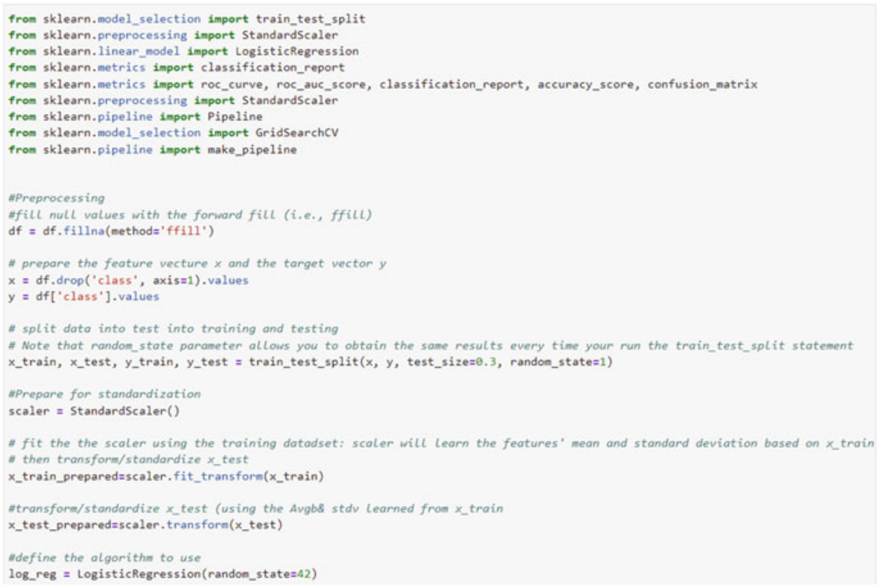
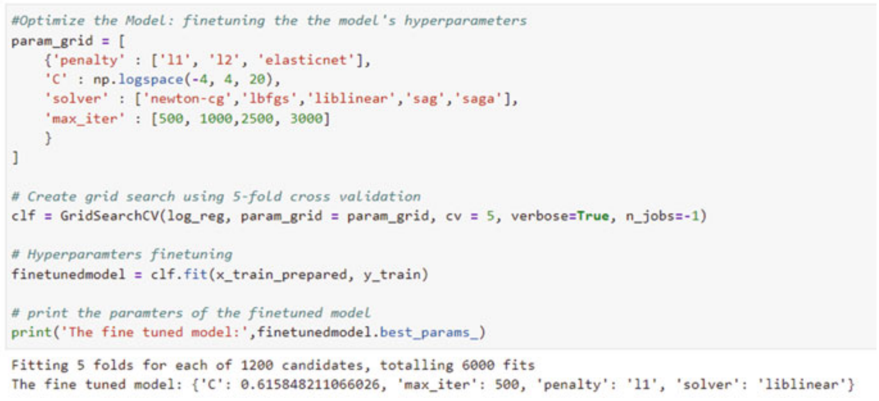


Fig. 7.4 Visualize Pima Indians diabetes dataset



```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, roc_auc_score, classification_report, accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline


#Preprocessing
#fill null values with the forward fill (i.e., ffill)
df = df.fillna(method='ffill')

# prepare the feature vecture x and the target vector y
x = df.drop('class', axis=1).values
y = df['class'].values

# split data into test into training and testing
# Note that random_state parameter allows you to obtain the same results every time your run the train_test_split statement
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

#Prepare for standardization
scaler = StandardScaler()

# fit the the scaler using the training datadset: scaler will learn the features' mean and standard deviation based on x_train
# then transform/standardize x_test
x_train_prepared=scaler.fit_transform(x_train)

#transform/standardize x_test (using the Avgb& stdv learned from x_train
x_test_prepared=scaler.transform(x_test)

#define the algorithm to use
log_reg = LogisticRegression(random_state=42)
```

Fig. 7.5 Preprocessing and scaling data



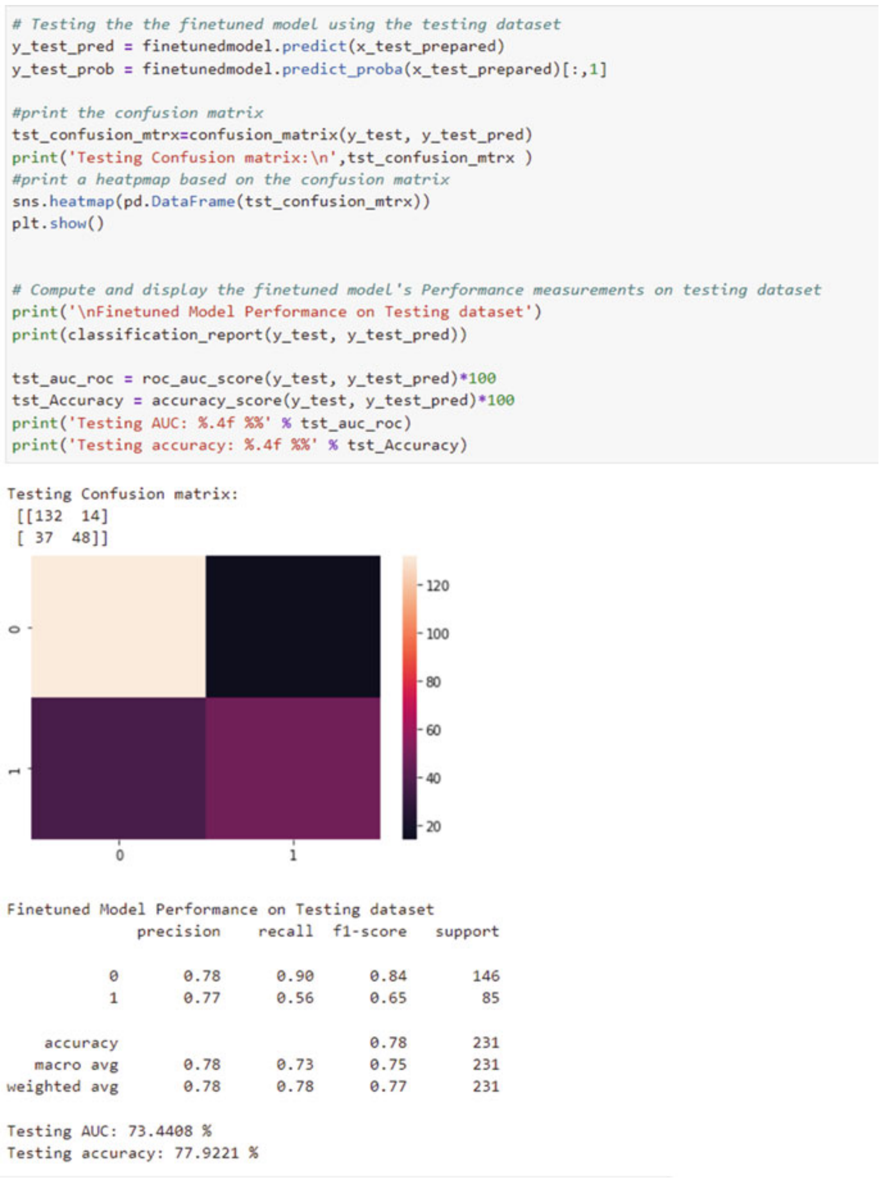
```
#Optimize the Model: finetuning the the model's hyperparameters
param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['newton-cg','lbfgs','liblinear','sag','saga'],
    'max_iter' : [500, 1000,2500, 3000]
    }
]

# Create grid search using 5-fold cross validation
clf = GridSearchCV(log_reg, param_grid = param_grid, cv = 5, verbose=True, n_jobs=-1)

# Hyperparamters finetuning
finetunedmodel = clf.fit(x_train_prepared, y_train)

# print the paramters of the finetuned model
print('The fine tuned model:',finetunedmodel.best_params_)
```

```
Fitting 5 folds for each of 1200 candidates, totalling 6000 fits
The fine tuned model: {'C': 0.615848211066026, 'max_iter': 500, 'penalty': 'l1', 'solver': 'liblinear'}
```

Fig. 7.6 Implementing grid search to optimize the model using training dataset

y_tain). The finetune model is stored in the variable finetunedmodel; once printed we can read the hyperparameters that provide the best performance (Fig. 7.6).

We can now use the fine-tuned model to make predictions on the testing dataset and print the performance measurement including the confusion matrix and a classification report that reports on precision, recall, f-score, and accuracy. Micro and macro averages are reported too (investigate what are those measurements and how they differ). Finally, the AUC is computed (Fig. 7.7). We can now plot the ROC curve (Fig. 7.8).



```
# Testing the the finetuned model using the testing dataset
y_test_pred = finetunedmodel.predict(x_test_prepared)
y_test_prob = finetunedmodel.predict_proba(x_test_prepared)[:,1]

#print the confusion matrix
tst_confusion_mtrx=confusion_matrix(y_test, y_test_pred)
print('Testing Confusion matrix:\n',tst_confusion_mtrx )
#print a heatpmap based on the confusion matrix
sns.heatmap(pd.DataFrame(tst_confusion_mtrx))
plt.show()


# Compute and display the finetuned model's Performance measurements on testing dataset
print('\nFinetuned Model Performance on Testing dataset')
print(classification_report(y_test, y_test_pred))

tst_auc_roc = roc_auc_score(y_test, y_test_pred)*100
tst_Accuracy = accuracy_score(y_test, y_test_pred)*100
print('Testing AUC: %.4f %%' % tst_auc_roc)
print('Testing accuracy: %.4f %%' % tst_Accuracy)
```

```
Testing Confusion matrix:
 [[132  14]
 [ 37  48]]
```

```
Finetuned Model Performance on Testing dataset
              precision    recall  f1-score   support

           0       0.78      0.90      0.84       146
           1       0.77      0.56      0.65        85

    accuracy                           0.78       231
   macro avg       0.78      0.73      0.75       231
weighted avg       0.78      0.78      0.77       231

Testing AUC: 73.4408 %
Testing accuracy: 77.9221 %
```

Fig. 7.7 Test the finetune model, display the confusion matric and the performance measurements

REMINDER It would be good to discuss in class about the interpretation of the graphs in the visualization phase.

```
# print the AUC value
print('Optimal model\'s AUC: %.4f %%' % tst_auc_roc)

# Plot the ROC curve of the finetuned model on the testing dataset
fRate, tRate, thresh = roc_curve(y_test, y_test_prob)
plt.plot(fRate,tRate)
plt.plot([0,1],[0,1],color="black",linestyle="--")

plt.xlabel("False Positive Rate - FPR")
plt.ylabel("True Positive Rate - TPR")
plt.title("Diabetes ROC Curve")
```

```
Optimal model's AUC: 73.4408 %
Text(0.5, 1.0, 'Diabetes ROC Curve')
```

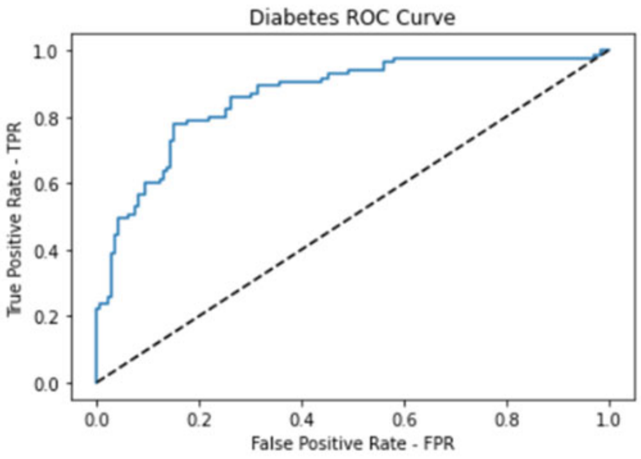
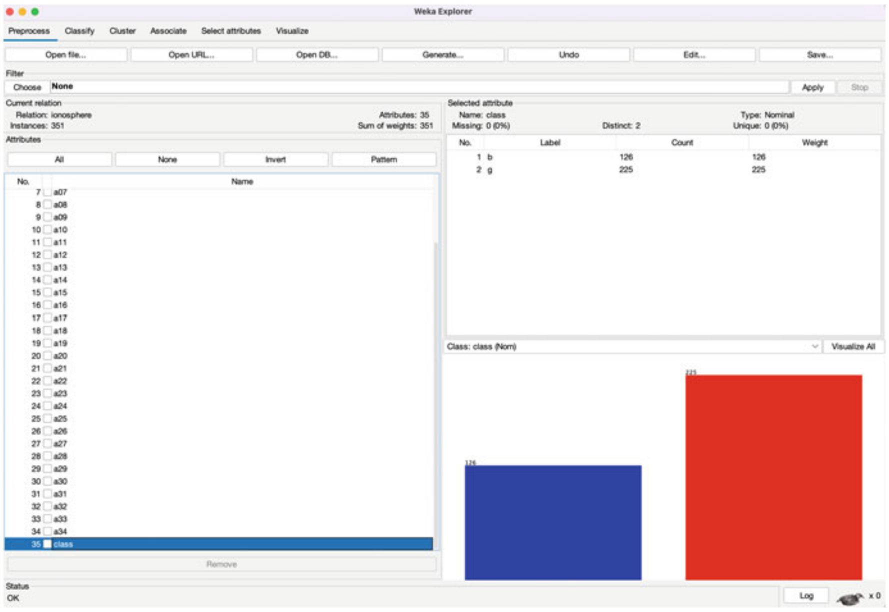


Fig. 7.8 Plot finetuned model ROC and AUC value

7.8.2 Working Example in Weka

NOTE on the use of Weka: Given the introductory nature of the book, and our focus on Python, we will not do GridSearch, instead we will use cross-validation to it obtain an estimate of the performance of an algorithm on the testing data set. We will always use the train-test split (70%, 30%) with cross-validation. However, we invite you to install a package called GridSearch and explore working with it. Then you can re-do the Weka exercises in this book but first splitting the dataset into training and testing, using Gridsearch to finetune your model hyperparameters, then using the testing the finetuned model on the testing dataset and displaying its performance measurements. We will provide online resources on how to use GridSearch in Weka for those who would like to work with this cool software.



Filter: None

Current relation
Relation: ionosphere
Instances: 351
Attributes: 35
Sum of weights: 351

Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	b	126	126
2	g	225	225

Attributes

No.	Name
7	a07
8	a08
9	a09
10	a10
11	a11
12	a12
13	a13
14	a14
15	a15
16	a16
17	a17
18	a18
19	a19
20	a20
21	a21
22	a22
23	a23
24	a24
25	a25
26	a26
27	a27
28	a28
29	a29
30	a30
31	a31
32	a32
33	a33
34	a34
35	class

Class: class (Nom)

Status
OK

Fig. 7.9 Weka preprocessing screen showing the binary outcome variable

For now, open the file “ionosphere” from the Weka datasets or download it from the Kaggle website using the following link: https://www.kaggle.com/prashant111/ionosphere.

This dataset was collected by antennas in Goose Bay, Labrador. The antennas targeted the detection of free electrons in the ionosphere. When radar returns showed evidence of a structure in the ionosphere, that was deemed a “good” radar detection; otherwise, it was deemed a “bad” detection. Received radar returns were processed using 17 “pulse” numbers, each having two attributes. All of the first 34 features are continuous, and the 35th attribute is binary (either “good” or “bad”) [2].

Open the file in Weka and explore the dataset (Fig. 7.9).

We note no missing outcome data in 351 instances, so there is no need for imputation.

Except for variable a02, the values of which are all zeros, all variables are standardized, as their values are between 0 and 1 (i.e., a01) or -1 and 1 (a03 to a034). So, we do not need to standardize our input features. Finally, we do not note any outliers.

Go to the Classify tab and click the Choose button. Select the Logistic function from the set of functions (Fig. 7.10).

Keep the default logistic parameters and click OK (Fig. 7.11).

In the Classifier window, choose 10 folds for the cross-validation (you can also try to split the data 75% for training and 25% for testing). Make sure the last column is chosen as the outcome. Figure 7.12 shows the feature “class,” which is the last column in the ionosphere.arff Weka file; if you have used the ionosphere_data.csv from the Kaggle.com website, your last column will be “column_ai”.

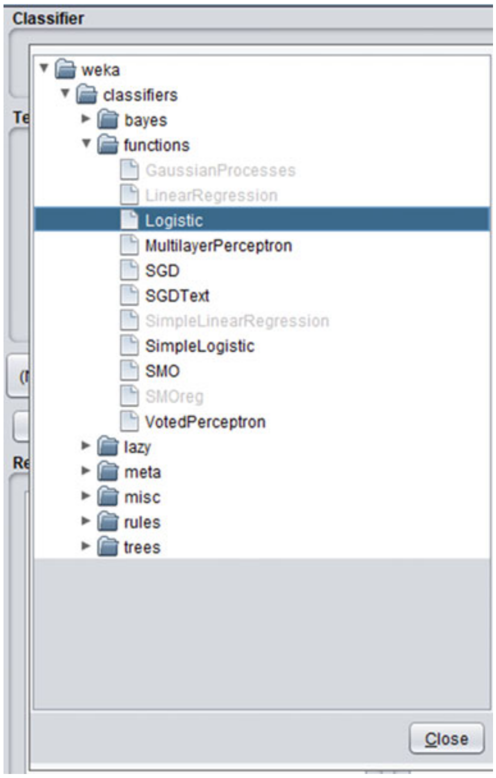


Fig. 7.10 Choosing the Logistic function

Click on the Start button to execute the logistic regression algorithm. The result is shown in Fig. 7.13.

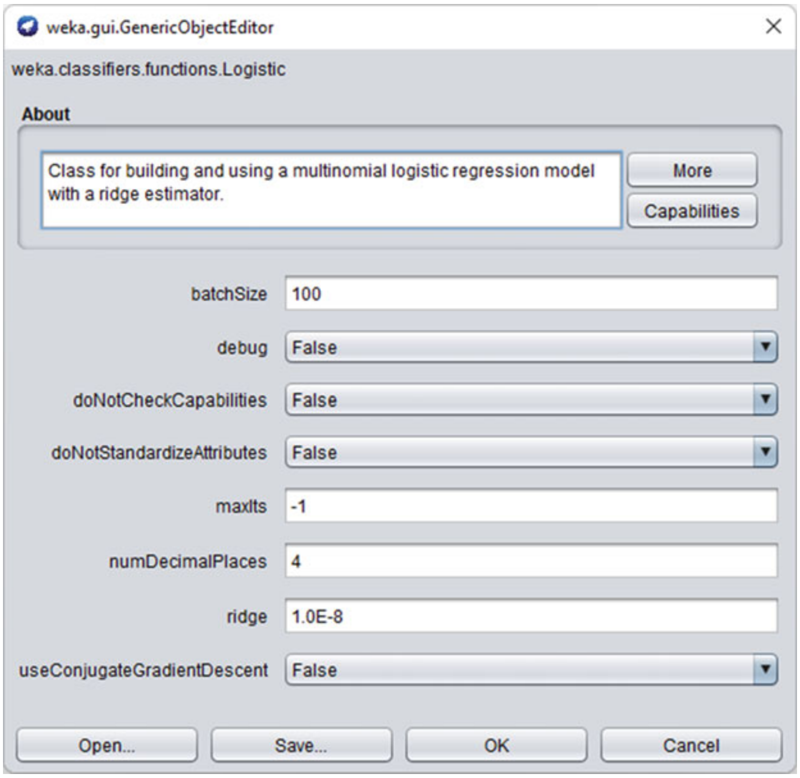
The model is nearly 89% accurate. Based on the confusion matrix (Fig. 7.13), 100 of 113 instances of class a (i.e., value = *b*) are correctly classified, and 212 of 238 instances of class b (i.e., value = *g*) are correctly classified.

Scroll up the screen and you will note the model's coefficients (Fig. 7.14a) and odds ratios (Fig. 7.14b). Given that the coefficients of the logit are the log of the odds ratios, the odds ratios of the model are the exponentials of the coefficients (e.g., the odds ratio of feature a03 = 0.1595 = $e^{-1.8357}$).

7.8.3 Do It Yourself

7.8.3.1 Predicting Online Purchases

Download the datafile social_network_Ads.csv from the following link: https://www.kaggle.com/ravinash218/social-network-ad-purchase.



weka.classifiers.functions.Logistic

Class for building and using a multinomial logistic regression model with a ridge estimator.

batchSize 100
debug False
doNotCheckCapabilities False
doNotStandardizeAttributes False
maxIts -1
numDecimalPlaces 4
ridge 1.0E-8
useConjugateGradientDescent False

Fig. 7.11 Logistic regression screen with default parameters

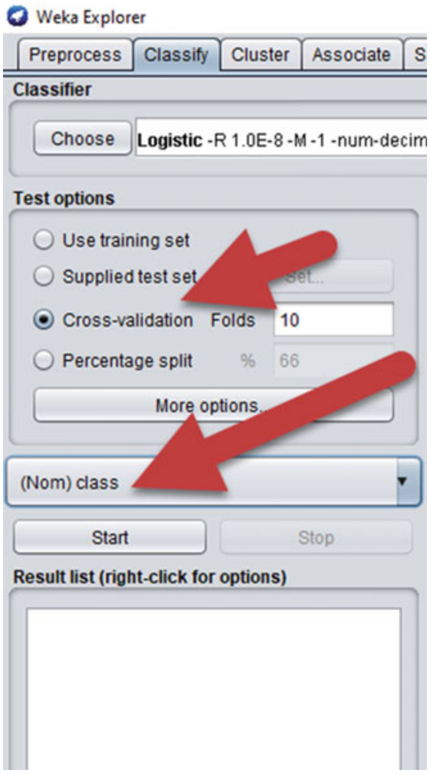
The csv file contains a dataset about people who made or did not make, online purchases: user id, gender, age, estimated salary, and the fact that they purchased or not. Build a logistic regression model that allows us to predict if a person will or will not purchase online based on their age, gender, and estimated salary.

7.8.3.2 Predicting Click-Through Advertisements

Download the three datafiles related to Click-Through Rate Prediction https://www.kaggle.com/c/avazu-ctr-prediction/data. One file (train) is for training and contains 10 days of click-through data ordered chronologically, and another file (test) is for testing and stores 1 day of ads click-through for testing the model predictions.

Use logistic regression to train and test a model that predicts click-throughs. The data fields are explained as follows on the Kaggle website:

1. id: ad identifier
2. click: 0 for “non-click” and for “click”



Logistic -R 1.0E-8 -M -1 -num-decim

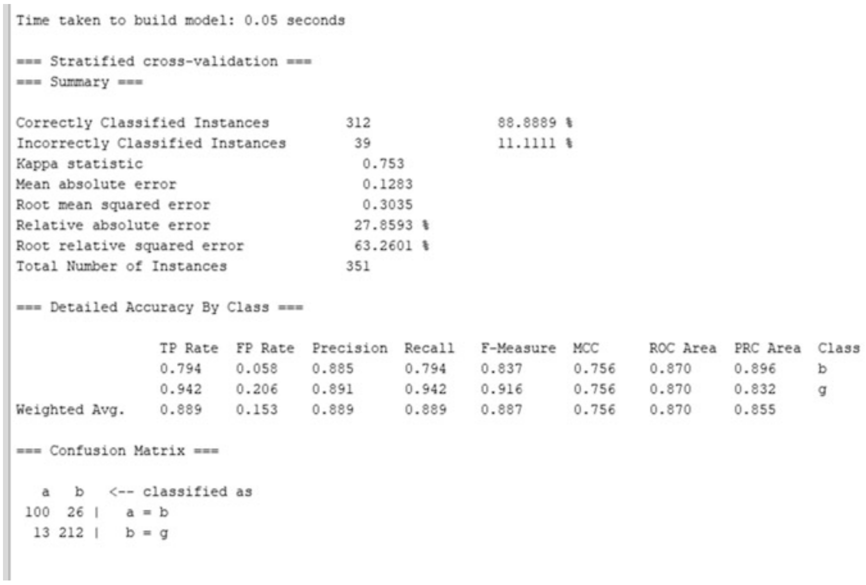
[✓ Cross-validation] Folds 10

Percentage split % 66

(Nom) class

Fig. 7.12 Parameters in the Classifier window

3. hour: format is YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014 UTC.
4. C1: anonymized categorical variable
5. banner_pos: the position of the banner
6. site_id: the website id
7. site_domain: the website domain
8. site_category: website category (this is a hashed value not readable)
9. app_id: mobile App identifier
10. app_domain: mobile App domain
11. app_category: mobile App category
12. device_id: mobile device identifier
13. device_ip: mobile IP address
14. device_model: the mobile device model (e.g., Samsung, iPhone, etc.) (also hashed value)
15. device_type: hashed value indicating the mobile device type (e.g., tablet, smartphone)
16. device_conn_type: such as wifi, 4G.
17. C14-C21: anonymized categorical variables



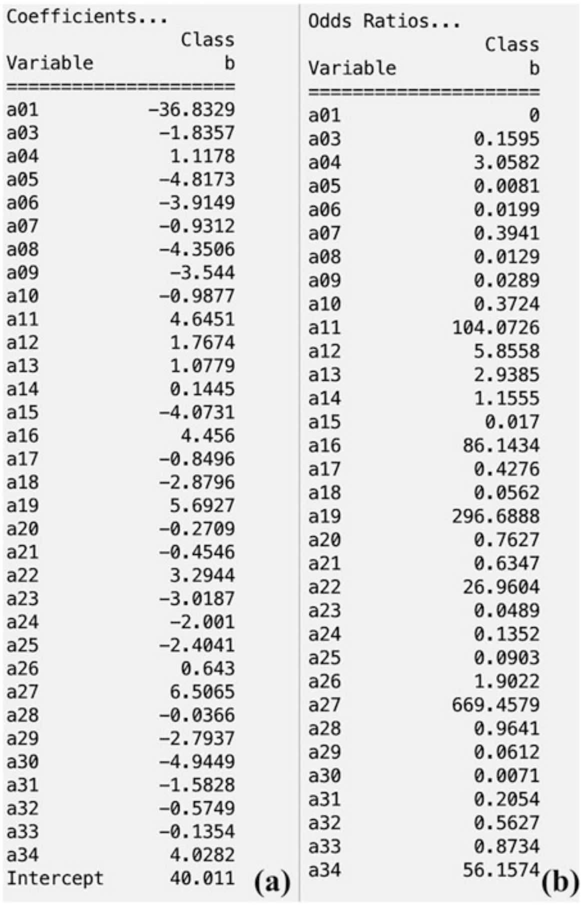
```
Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         312               88.8889 %
Incorrectly Classified Instances        39               11.1111 %
Kappa statistic                          0.753
Mean absolute error                      0.1283
Root mean squared error                  0.3035
Relative absolute error                 27.8593 %
Root relative squared error             63.2601 %
Total Number of Instances              351

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.794    0.058    0.885      0.794    0.837      0.756    0.870     0.896     b
                 0.942    0.206    0.891      0.942    0.916      0.756    0.870     0.832     g
Weighted Avg.    0.889    0.153    0.889      0.889    0.887      0.756    0.870     0.855

=== Confusion Matrix ===

   a   b   <-- classified as
 100  26 |   a = b
  13 212 |   b = g
```

Fig. 7.13 Result of the logistic regression. The model precision is nearly 89%

7.8.4 *Do More Yourself*

Create logistic regression models using the Python datasets below. Ensure that you preprocess the data.

1. Titanic: https://www.kaggle.com/heptapod/titanic
2. HR Analytics Case Study: https://www.kaggle.com/vjchoudhary7/hr-analytics-case-study
3. Logistic regression to predict heart disease: https://www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression



```
Coefficients...
                     Class
Variable                 b
=====================
a01               -36.8329
a03                -1.8357
a04                 1.1178
a05                -4.8173
a06                -3.9149
a07                -0.9312
a08                -4.3506
a09                 -3.544
a10                -0.9877
a11                 4.6451
a12                 1.7674
a13                 1.0779
a14                 0.1445
a15                -4.0731
a16                  4.456
a17                -0.8496
a18                -2.8796
a19                 5.6927
a20                -0.2709
a21                -0.4546
a22                 3.2944
a23                -3.0187
a24                 -2.001
a25                -2.4041
a26                  0.643
a27                 6.5065
a28                -0.0366
a29                -2.7937
a30                -4.9449
a31                -1.5828
a32                -0.5749
a33                -0.1354
a34                 4.0282
Intercept           40.011
```

(a)

```
Odds Ratios...
                     Class
Variable                 b
=====================
a01                      0
a03                 0.1595
a04                 3.0582
a05                 0.0081
a06                 0.0199
a07                 0.3941
a08                 0.0129
a09                 0.0289
a10                 0.3724
a11               104.0726
a12                 5.8558
a13                 2.9385
a14                 1.1555
a15                  0.017
a16                86.1434
a17                 0.4276
a18                 0.0562
a19               296.6888
a20                 0.7627
a21                 0.6347
a22                26.9604
a23                 0.0489
a24                 0.1352
a25                 0.0903
a26                 1.9022
a27               669.4579
a28                 0.9641
a29                 0.0612
a30                 0.0071
a31                 0.2054
a32                 0.5627
a33                 0.8734
a34                56.1574
```

(b)

Fig. 7.14 (**a**) Coefficients and (**b**) odds ratios of the logistic regression model

References

1. A.L. Lynam et al., Logistic regression has similar performance to optimised machine learning algorithms in a clinical setting: Application to the discrimination between type 1 and type 2 diabetes in young adults. Diagn Progn. Res. **4**, 6 (2020). https://doi.org/10.1186/s41512-020-00075-2
2. F. Khurshid et al., Comparison of multivariable logistic regression and machine learning models for predicting bronchopulmonary dysplasia or death in very preterm infants. Front. Pediatr. **9**, 759776 (2021). https://doi.org/10.3389/fped.2021.759776
3. R.D. Joshi, C.K. Dhakal, Predicting type 2 diabetes using logistic regression and machine learning approaches. Int. J. Environ Res. Public Health **18**(14), 7346 (9 Jul 2021). https://doi.org/10.3390/ijerph18147346
4. X. Song, X. Liu, F. Liu, C. Wang, Comparison of machine learning and logistic regression models in predicting acute kidney injury: A systematic review and meta-analysis. Int. J. Med. Inform. **151**, 104484 (Jul 2021). https://doi.org/10.1016/j.ijmedinf.2021.104484

5. A. Björkelund et al., Machine learning compared with rule-in/rule-out algorithms and logistic regression to predict acute myocardial infarction based on troponin T concentrations. J. Am. Coll. Emerg. Physicians Open **2**(2), e12363 (Apr 2021). https://doi.org/10.1002/emp2.12363
6. E. Christodoulou, J. Ma, G.S. Collins, E.W. Steyerberg, J.Y. Verbakel, B. Van Calster, A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. J. Clin. Epidemiol. **110**, 12–22 (Jun 2019). https://doi.org/10.1016/j.jclinepi.2019.02.004
7. T.E. Cowling, D.A. Cromwell, A. Bellot, L.D. Sharples, J. van der Meulen, Logistic regression and machine learning predicted patient mortality from large sets of diagnosis codes comparably. J. Clin. Epidemiol. **133**, 43–52 (May 2021). https://doi.org/10.1016/j.jclinepi.2020.12.018
8. H.R. Gosselt et al., Complex machine-learning algorithms and multivariable logistic regression on par in the prediction of insufficient clinical response to methotrexate in rheumatoid arthritis. J. Pers. Med. **11**(1), 44 (14 Jan 2021). https://doi.org/10.3390/jpm11010044
9. H. Lian et al., Integrative analysis of gene expression and DNA methylation through one-class logistic regression machine learning identifies stemness features in medulloblastoma. Mol. Oncol. **13**(10), 2227–2245 (Oct 2019). https://doi.org/10.1002/1878-0261.12557
10. R. Arabi Belaghi, J. Beyene, S.D. McDonald, Prediction of preterm birth in nulliparous women using logistic regression and machine learning. PLoS One **16**(6), e0252025 (2021). https://doi.org/10.1371/journal.pone.0252025
11. J. Brownlee, *Master Machine Learning Algorithms: Discover how They Work and Implement Them from Scratch* (Jason Brownlee, 2016)

Chapter 8
Decision Trees

8.1 The Problem

Linear and logistic regressions make predictions about numbers, but we also need algorithms to classify instances of data in a certain class, i.e., to label the instance as belonging to a class. The decision tree is our first approach to solve classification problems. However, decision trees can perform regression too, hence their name classification and regression trees (CART). The random forests that we will encounter in a later chapter are powerful variations of CART.

A CART is represented by a binary tree whose root is on top, and at each level, each node (including the root node) receives a data input that is examined. If the value of the feature is below a certain value, the left branch of the binary tree is followed; otherwise, the right branch is followed [1]. At the bottom level, we find the leaf nodes, or terminal nodes, which represent outcome values.

When we take an instance of data, we use the tree to compare the instance's attribute values to the root and decide whether the instance belongs to one subbranch or the other. The process is continued until we reach a leaf that represents a class.

Figure 8.1 represents a decision tree that mimics the underwriting process of a mortgage application. Each mortgage application contains the number of dependents, loan-to-value ratio, marital status, payment-to-income ratio, interest rate, years at the current address, and years in a current job.

8.2 A Practical Example

The Ionosphere dataset that was introduced in the previous chapter was collected in Goose Bay, Labrador, and represents 34 input features of continuous values and one binary output that classifies the measurement as either "good" (i.e., value *g*) or "bad"

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_8

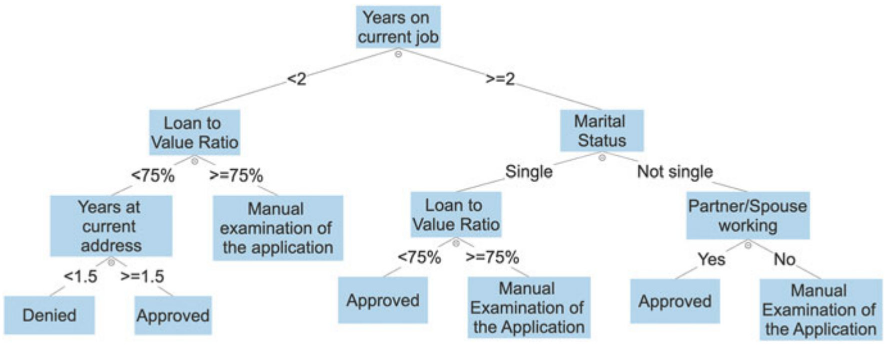


Fig. 8.1 An example of a decision tree to represent decision-making for a mortgage application (adapted from Maimon and Rokach [2])

(i.e., value *b*) [3]. In the previous chapter, we could build a logistic regression model with nearly 89% accuracy.

However, given the continuous nature of the input features, we can apply a decision tree to the same dataset to make a classification decision.

Open the dataset and choose the REPTree algorithm from the Trees classifiers (Fig. 8.2).

REPTree is the name of the CART algorithm in Weka. Keep the defaults for all REPTree parameters and run the algorithm (Fig. 8.3).

In the detailed accuracy table, we can notice the precision of almost 89.5%, which constitutes a slight improvement compared to the logistic regression model. Right-click on the Result List and click on Visualize Tree (Fig. 8.4).

Weka displays the decision tree for the Ionosphere dataset (Fig. 8.5). The feature a05 is in the root node, where a decision is made based on the threshold 0.02. The leaf nodes are radar detection outcomes: *b* (bad) or *g* (good). Only four features contributed to the decision tree: a03, a05, a22, and a27.

Using this decision tree, we can predict for any measurements made in the future if the radar detection will be bad or good based on the values of only four features.

8.3 The Algorithm

8.3.1 Tree Basics

In Fig. 8.6, we can recognize many elements of a decision tree.

The *root node* is the top (first) node of a decision tree, and a *leaf node* is an end node. At the root, all the dataset is present and is divided into homogeneous subsets based on certain decision rules. The leaves are nodes that do not split; they represent the outcome variable.

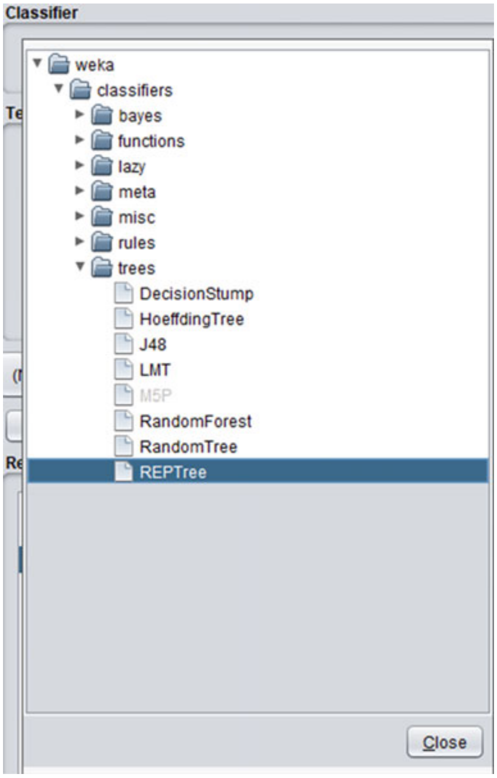
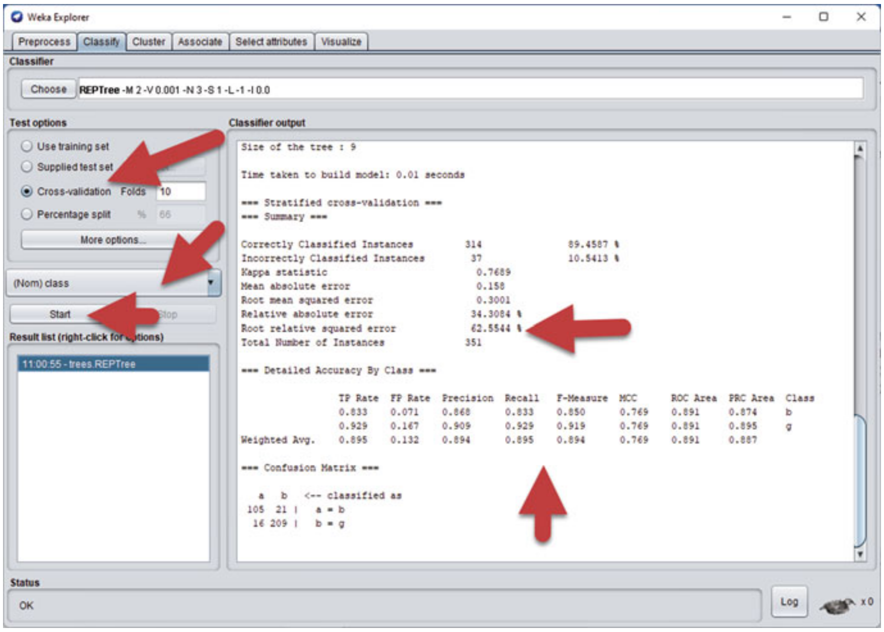


Fig. 8.2 REPTree chosen from the Trees classifiers in Weka

Every internal node between the root and the leaves is a *decision node* that splits the data based on splitting rules. Every node in the tree is a *parent node* for any node directly below it, which is called a *child node* to the parent. A subset of nodes and associated leaves is called a *subtree* or *branch*. While splitting is the process of dividing the data at a certain node into two or more sub-nodes, *pruning* is the process of deleting sub-nodes of a decision node and redistributing data associated with it; we will see more about pruning and its necessity below.

Trees can apply to classification problems when the outcome is a categorical variable, as we have seen in the ionosphere example; other examples include predicting if a patient will be subject to readmission after discharge, or if a person will get vaccinated for COVID-19, or will get her loan approved, or will make it to the Olympic finals. Also, trees can apply to regression problems where the outcome is a numeric (i.e., continuous) variable; for example, based on several features, we can try to predict a future house price, the infection rate in a population, the area of land that will be subject to desertification, or the number of migrants crossing the Mediterranean Sea.

The decision to split the data at a certain node, including the root, is not straightforward; the final tree and associated decisions (i.e., regression, classification) change drastically if we split based on one feature or another. We need a



REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

```
Size of the tree : 9

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         314               89.4587 %
Incorrectly Classified Instances        37               10.5413 %
Kappa statistic                          0.7689
Mean absolute error                      0.158
Root mean squared error                  0.3001
Relative absolute error                 34.3084 %
Root relative squared error             62.5544 %
Total Number of Instances              351

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.833    0.071    0.868      0.833    0.850      0.769    0.891     0.874     b
                 0.929    0.167    0.909      0.929    0.919      0.769    0.891     0.895     g
Weighted Avg.    0.895    0.132    0.894      0.895    0.894      0.769    0.891     0.887

=== Confusion Matrix ===

   a   b   <-- classified as
 105  21 |   a = b
  16 209 |   b = g
```

Fig. 8.3 REPTree execution in Weka

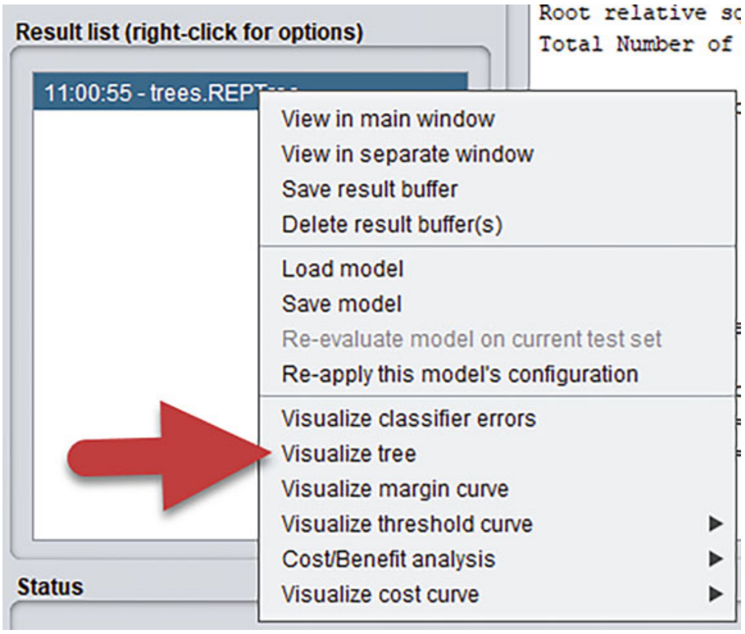


Fig. 8.4 Visualize tree option

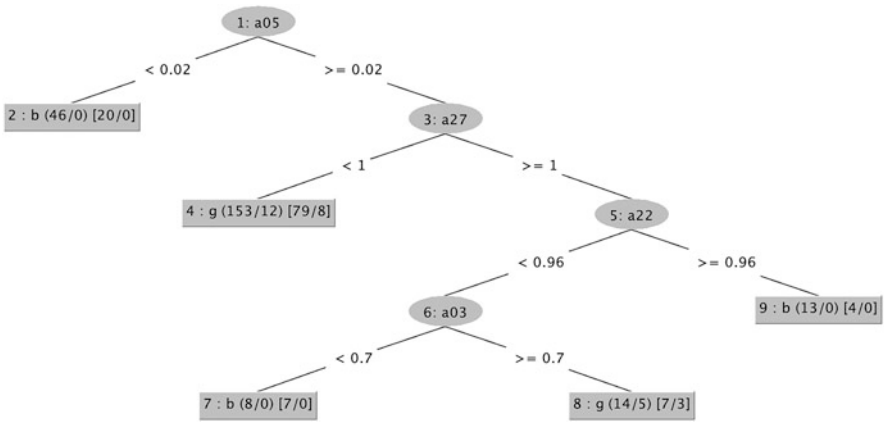


Fig. 8.5 A decision tree for the ionosphere problem

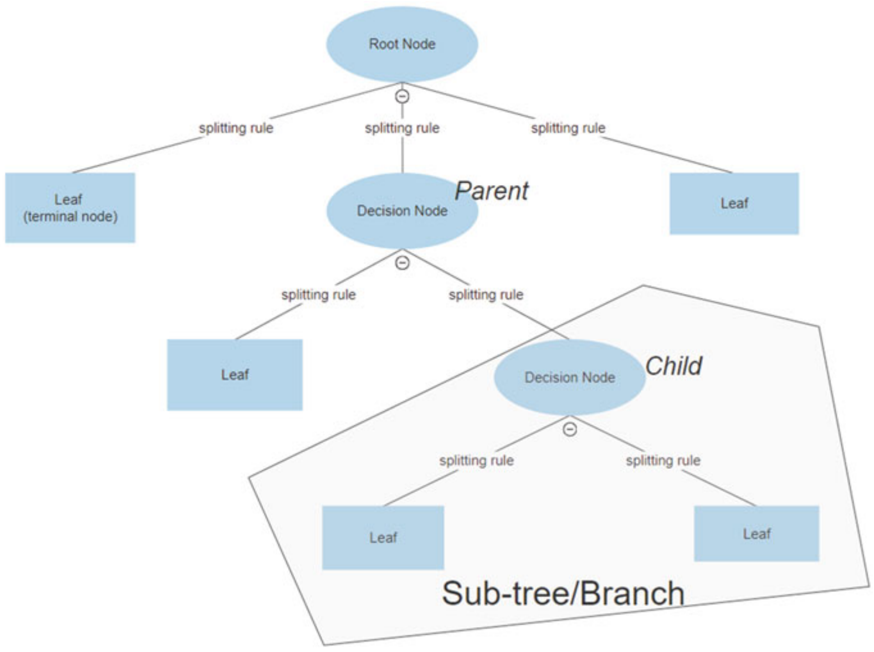


Fig. 8.6 Decision tree elements

decision criterion to decide which feature to choose at a certain node to base our splitting upon. How to choose a decision criterion? Each time we split the data into two or more subsets, we are aiming at homogenizing the subsets; therefore, researchers have invented functions to measure *homogeneity* or *purity* at a node with respect to the outcome variable and based the decision criteria of those purity measurements: at each node, we choose to split the data based on the feature that maximizes data homogeneity.

Researchers have invented many algorithms to create trees and select the feature for splitting, such as:

1. ID3: Extension of a previous version, D3 (ID stands for “iterative dichotomizer”)
2. C4.5: Successor to ID3
3. CART: Classification and regression tree
4. CHAID: Chi-square automatic interaction detection. It executes multilevel splits for classification trees.
5. MARS: Multivariate adaptive regression splines

A *greedy* algorithm like ID3 acts as follows: at each node, the decision tree algorithm will use the available dataset at that node, calculate all possible splits using every possible feature, and choose the feature that maximizes data homogeneity to do the split. The split is done, and the dataset is distributed among the sub-nodes. The same process continues at each sub-node using the remaining features until no more splitting can be done.

There are many data homogeneity functions, including the following:

1. Entropy
2. Information gain (IG)
3. Gini index
4. Information gain ratio

Entropy measures randomness (i.e., think of it as the opposite of homogeneity), so at each node, we choose to split the feature that minimizes entropy. Information gain is a measure of homogeneity; hence, at each node, we choose to split the feature that minimizes information gain. Like information gain, the Gini index is another measure of homogeneity. The information gain ratio is a correction included for the information gain; it is a measurement that allows us to avoid splitting based on an attribute with high information gain but a large number of distinct values [4], such as a credit card number. A feature like customers’ credit card numbers presents high information gain, but we should not split based on this feature, as it will not be helpful to predict anything about a future customer, who necessarily will have a different credit card number (i.e., the variation of distinct values of credit card numbers is extremely high). The information gain ratio for such features will be low, and the splitting criterion will not make the split based on ratios below the average IG. Information gain is a criterion used for categorical features, while the Gini index is used for continuous attributes. Below, we will see formulas and examples for entropy and information gain.

Entropy: We seek to minimize entropy because it is a measurement of non-homogeneity; the higher the entropy, the worse the solution for splitting the data will be. Entropy is calculated in the following way:

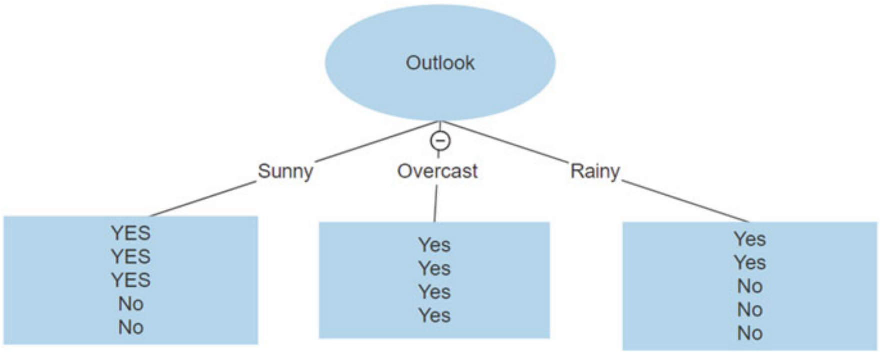


Fig. 8.7 A decision tree example showing the decision to play outdoors in relation to weather outlook

$$E(\mathrm{S}) = \sum_{n=1}^{c} \left(-p_i \log_2(p_i)\right)$$

S is the current state (e.g., current node), and p_i is the percentage of class *i* in a node of state *S*, or the probability of an event *i* of state *S*. Suppose we have a set of 16 instances at a node in relation to a feature called Humidity with two values, "High" and "Normal," where 10 instances have a value of "High" for Humidity and five have a value of "Normal." The entropy at this node in this status is calculated as

$$\text{Entropy (Humidity)} = \text{Entropy (5, 10)} = p_{\text{Normal}} \times \log_2(p_{\text{Normal}}) + p_{\text{High}} \times \log_2\left(p_{\text{High}}\right)$$

$$\text{Entropy} = -\frac{5}{15} \times \log_2\left(\frac{5}{15}\right) - \frac{10}{15} \times \log_2\left(\frac{10}{15}\right)$$

$$\text{Entropy} = -0.34 \times (-1.585) - 0.67 \times (-0.585)$$

$$\text{Entropy} = 0.931$$

For two features, we will illustrate the use of entropy with more than one feature through an example (Fig. 8.7).

Consider the tree shown in Fig. 8.7.

The entropy for playing outdoors given the different weather outlooks is computed as follows:

$$E(\text{Playing, Outlook}) = P(\text{sunny}) \times E(3\text{ Yes, }2\text{ No}) + P(\text{overcast}) \times E(4\text{ Yes, }0\text{ No}) + P(\text{rainy}) \times E(2\text{ Yes, }3\text{ No}) = \frac{5}{14} \times \left(-\frac{2}{5} \times \log_2\left(\frac{2}{5}\right) - \frac{3}{5} \times \log_2\left(\frac{3}{5}\right)\right) + \frac{4}{14} \times \left(-\frac{4}{4} \times \log_2\left(\frac{4}{4}\right) - \frac{0}{5} \times \log_2\left(\frac{0}{4}\right)\right) + \frac{5}{14} \times \left(-\frac{3}{5} \times \log_2\left(\frac{3}{5}\right) - \frac{2}{5} \times \log_2\left(\frac{2}{5}\right)\right) = \frac{5}{14} \times 0.971 + 0 + \frac{5}{14} \times 0.971 = 0.693$$

Information gain: As a measure of homogeneity, information gain (IG) is opposite to entropy; the higher the information gain, the lower the entropy. Information gain computes the difference between entropy before a split and average entropy after the split. Information gain is used by ID3.

The information gain resulting from splitting the 14 instances of the dataset in Fig. 8.7 into “Sunny” is calculated by

$$E(\text{Playing outdoors}) - E(\text{Playing, Outlook}) = E\left(\frac{9}{14}, \frac{5}{14}\right) - 0.693 = -\frac{9}{14} \times \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \times \log_2\left(\frac{5}{14}\right) - 0.693 = 0.940 - 0.693 = 0.247$$

We will see more about entropy and information gained below.

Gini index: The Gini index provides an indication of purity and is computed as follows:

$$G = \sum_{k=1}^{n} p_k \times (1 - p_k)$$

p_k is the proportion of training instances with class k in the leaf of interest, or the rectangle of interest when we look at scatter graphs (see below).

Sum squared error: When using trees for regression, we choose the split that will minimize the sum squared error across all training samples. The sum squared error is computed as follows:

$$S = \sum_{i=1}^{n} (\text{outcome}_i - \text{prediction}_i)$$

where n is the number of instances in question.

8.3.2 Training Decision Trees

Consider that the training dataset is *S*, the input features are I, and the outcome is *O*. The *split criterion* is the method used to decide if an instance should go to the left or the right of a node. The *stop criterion* is a condition that, if met, will stop the development of the tree.

The algorithm to create the tree can be illustrated using the following example [2]. Suppose we want to develop a smart model to filter spam emails. In real life, we will need many features to create such a model; however, for our illustration, we will suppose that we will build the model based only on two features: the length of the message and the number of new recipients of the email. Below is a scatter graph representing the dataset that we will use to train the decision tree (Fig. 8.8).

We start with one feature and try to divide the dataset in a manner to minimize the cost (the number of classification errors). Figure 8.9 is the result of using the New Recipients feature for classification (Fig. 8.9). Figure 8.10 is the result of using the Email Length feature (Fig. 8.10). Both figures show one single-node decision tree, called a decision stump.

If we use the New Recipients feature, we will end up with nine classification errors, while if we use the Email Length feature, we will end up with nine errors. Obviously, using the Email Length feature will incur less cost (fewer errors in classification); hence, we will use it in the next steps.

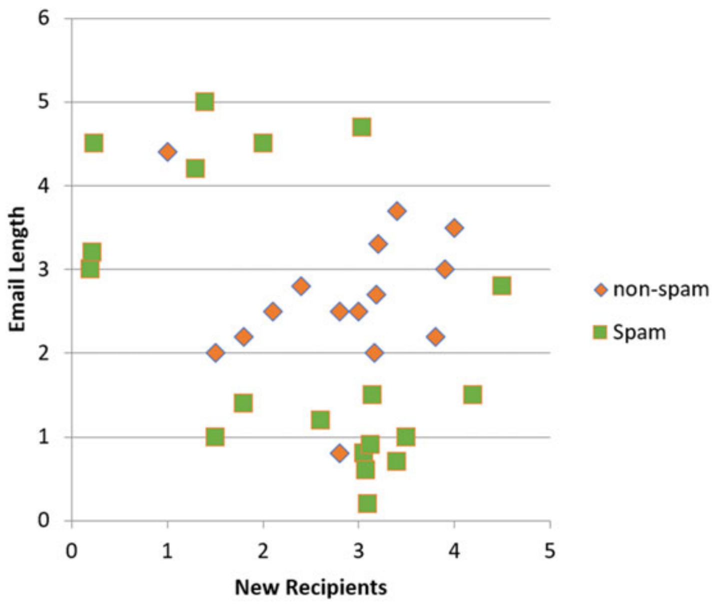


Fig. 8.8 Email spam training dataset scatter graph

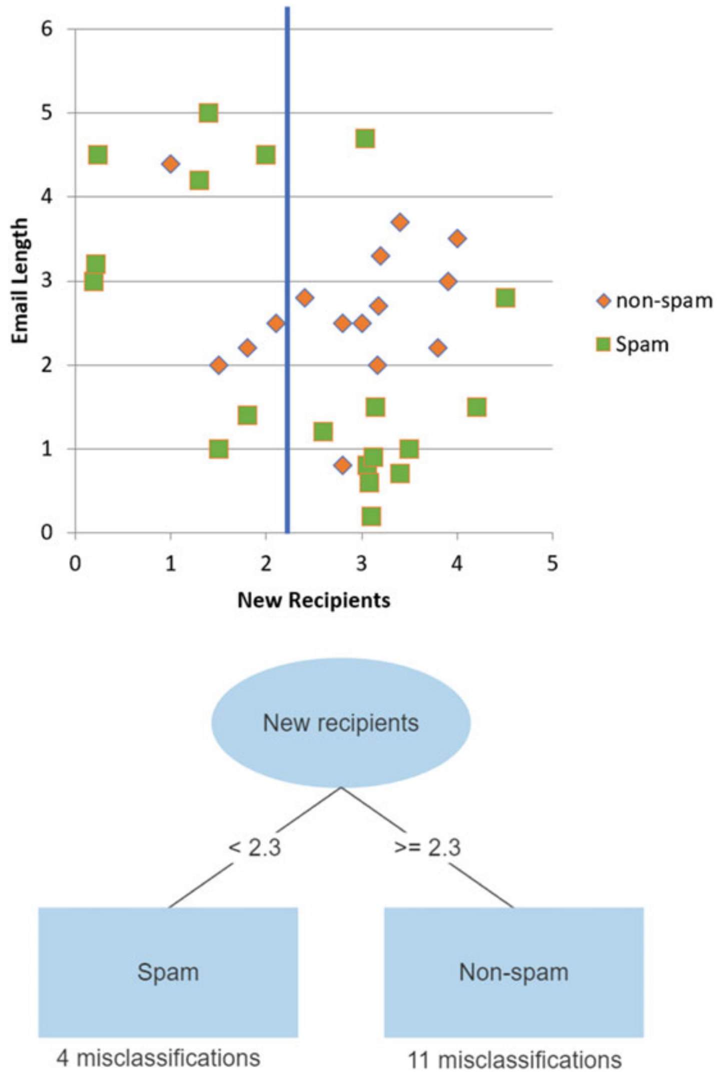


Fig. 8.9 A single-node decision tree using the New Recipients feature

In the next step, we will split the email subset with Email Length ≥ 1.8 into two new subsets: less than 4 and greater than or equal to 4; each area has a few classification errors (Fig. 8.11).

The process will continue until we reach convergence, i.e., until each region contains a sample from one class only (Fig. 8.12). Figure 8.12 shows nine different regions, each consisting of instances of the same class; the corresponding tree is shown in Fig. 8.13. This solution suffers from a lack of generalizability, as it has

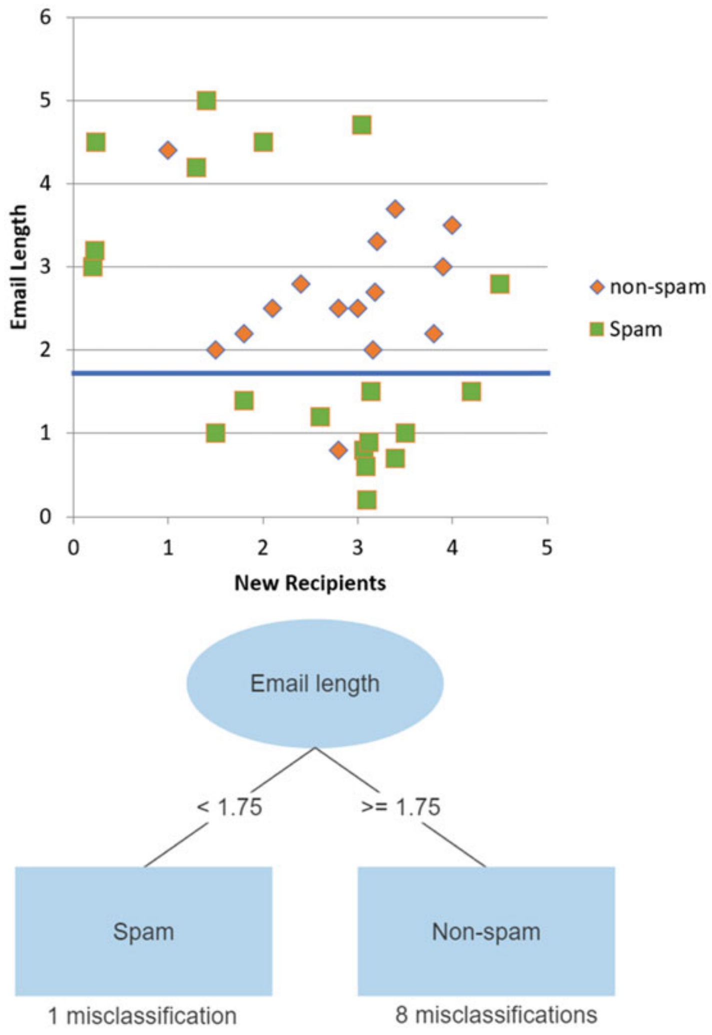


Fig. 8.10 A single-node decision tree using the Email Length feature

learned to classify the training dataset with 100% accuracy, but it risks not faring well with a new dataset.

8.3.3 *A Generic Algorithm*

Consider the dataset shown in Table 8.1, which represents a decision table to play outdoors based on four features related to the weather (Outlook, Temperature, Humidity, and Windy) [5].

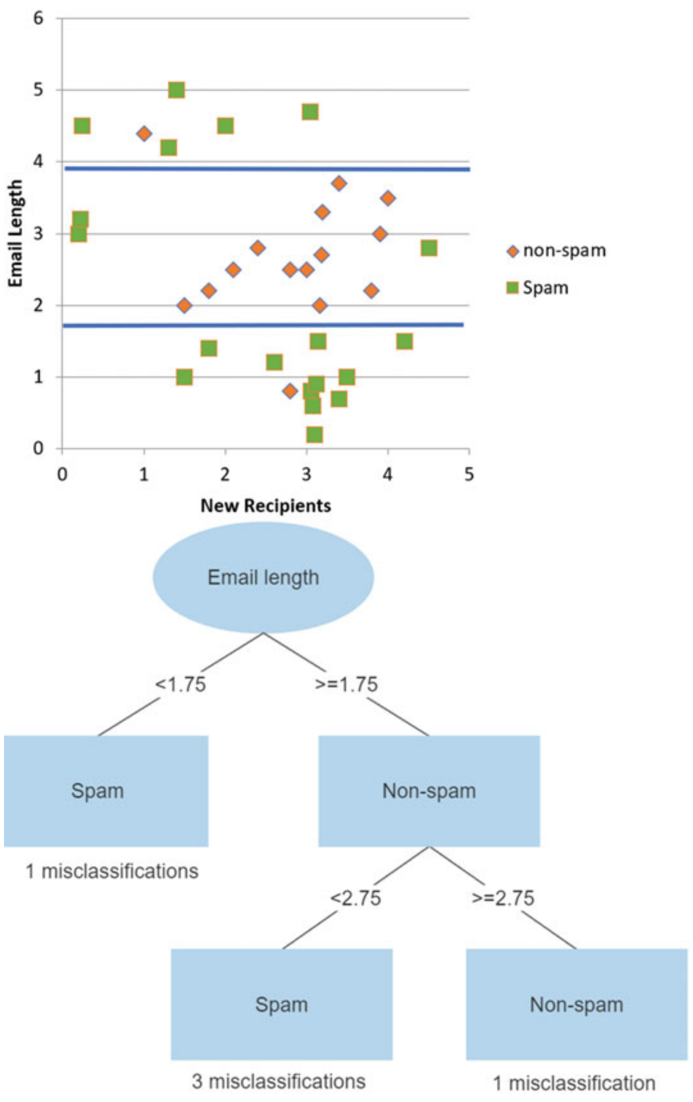


Fig. 8.11 The decision tree after two splits based on the Email Length feature

A decision tree can be created to decide whether to play outdoors or not based on the four weather conditions. The problem of creating the tree can be formulated as follows: choose an attribute to place at the tree's root, split the data to the left and right based on the values of the attribute, repeat the process for the left subset, then repeat the process for the right subset. For any left or right subset, stop when all instances at a node are of the same class. This is a recursive process.

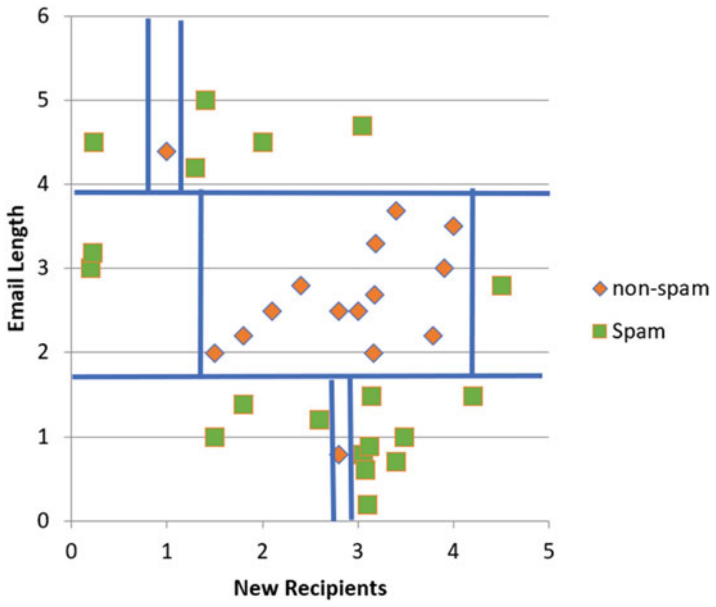


Fig. 8.12 The final graph split into distinct regions

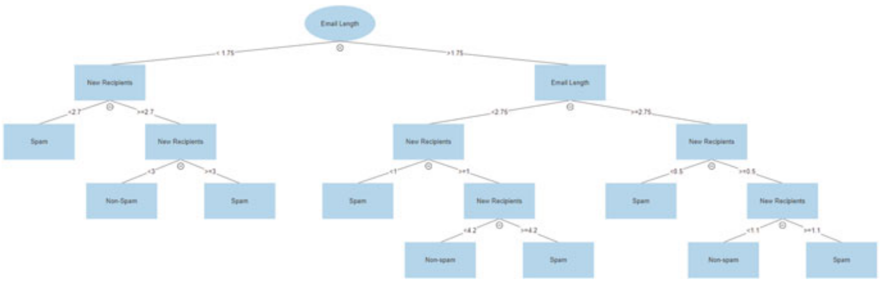


Fig. 8.13 The final tree solution corresponding to the graph in Fig. 8.8

Note that in this example, the data is binary nominal, while in the previous one, the data was numeric. The problem is that of classification and not of regression; however, trees can be used for regression.

Which attribute should we choose for the root? In the previous example about spam, we chose the attribute that generated the fewest classification errors. Here, we will aim at producing the fewest branches; we will do so by using a function known as the information value or entropy:

$$\text{entropy}\,(p_1, p_2 \ldots .p_n) = -p_1 \log\,(p_1) - p_2 \log\,(p_2) - \ \ldots \ - p_n \log\,(p_n)$$

where p_1, p_2, … p_n are fractions, and $p_1 + p_2 + \ldots + p_n = 1$.

Table 8.1 Weather dataset

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

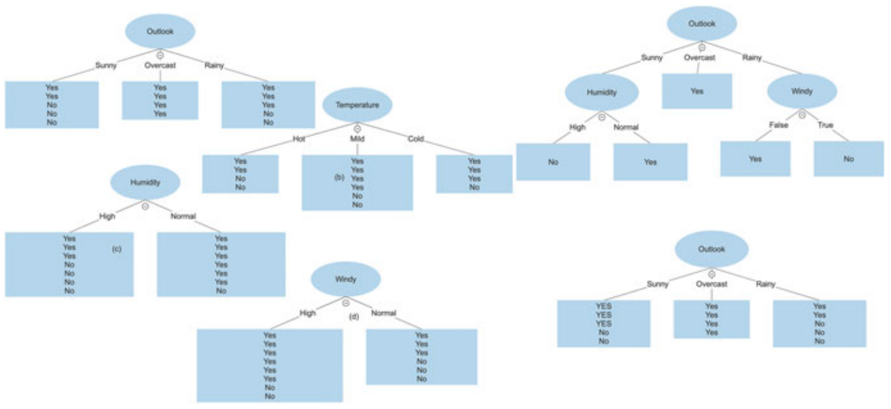


Fig. 8.14 Tree stumps for the weather dataset using each of the four features

The entropy is a measure of each node's purity, and we want to choose the feature that generates the purest daughter node, i.e., has as many instances of the same class as possible. If we take the example above and try to divide the instances based on each feature, we obtain the possibilities shown in Fig. 8.14.

The information value for the tree generated using the Outlook feature (Fig. 11.14) is computed as:

Information value [Sunny] = Information value [2 Yes and 3 No] = Entropy (2/5, 3/5) = −2/5×log(2/5)-3/5×log(3/5) = 0.971.

Information value [Overcast] = Information value [4 Yes and 0 No] = Entropy (4/4, 0/4) = 4/4×log(4/4) + 0/4×log(0/4) = 0.

Information value [Rainy] = Information value [3 Yes and 1 No] = Entropy (3/5, 2/5) = −3/5×log(3/5)−2/5×log(2/5) = 0.971.

The information value for the feature Outlook is computed as an average considering the number of instances in each subtree =5/14×0.971 + 4/14×0 + 5/14×0.971 = 0.694.

The root before any branching had 9 Yeses and 5 Nos, so the information value at the root was entropy(9/14, 5/14) = 0.940.

The *information gain* made by branching the tree using Outlook = 0.940–0.694 = 0.246.

Gain(Outlook) = 0.246 bits. The unit used for measurement is called "bits" but is not the same as computer bits.

We can do the same computation of the information gain resulting from using the Temperature, Humidity, and Windy features; we compare the results and choose the feature that provided the highest information gain. In our example,

Gain(Outlook) = 0.246 bits
Gain(Temperature) = 0.029 bits
Gain(Humidity) = 0.152 bits
Gain(Windy) = 0.048 bits

Hence, the best choice is to use the Outlook feature to split the tree at the root.

We continue using the same process and logic with each of the subtrees produced by Outlook, using the remaining features (i.e., Humidity and Windy). The result is shown in Fig. 8.15.

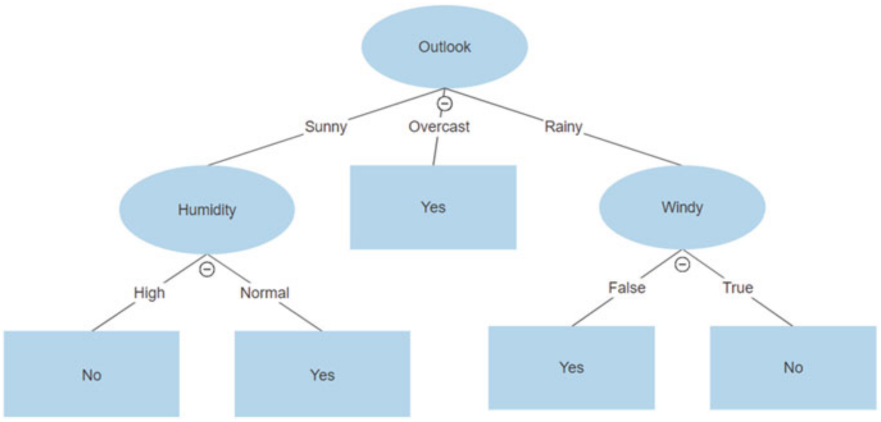


Fig. 8.15 The final decision tree for the weather dataset

8.3.4 Tree Pruning

Fully developed decision trees are complex in structure and risk overfitting as they learn to perfectly classify the training data and become less able to correctly classify new independent datasets. Pruning is a method that simplifies a decision tree; there are two methods for tree pruning: post-pruning or backward pruning and pre-pruning or forward pruning.

Pre-pruning involves a decision to stop developing subtrees while working on the development of a decision tree. Post-pruning seems more onerous, but it has the advantage of taking into account the combined effect of features on the decision instead of looking into the effect of each feature individually and deciding not to use it.

There are two methods for pruning: subtree replacement and subtree raising. In subtree replacement, we investigate the possibility of replacing a subtree with a leaf; it will make the tree less accurate on the training data but more generalizable (for unseen data). Subtree replacement works from the leaves upward in a tree. Subtree raising is more complex and time-consuming but more useful and is used in the well-known C.45 algorithm. In subtree raising, a whole subtree is removed, and its daughters are included in other subtrees. It is common to raise the subtree of the most popular branch.

If we take the example of a fully developed tree before pruning (Fig. 8.16a), subtree replacement of branch B can result in moving 4 and 5 to subtree C and deleting B (Fig. 8.16b); 1, 9, and 10 are the new instances resulting from the addition of instances 4 and 5 to 1, 2, and 3.

Subtree raising can result in the same replacement only if the total instances under 4 and 5 are fewer than those under C; otherwise, we replace B with node 4 or 5, whichever has more instances, and we reclassify all other instances 1, 2, 3 as well as 4 or 5 under the new node. Figure 8.16c shows a subtree raising result when node

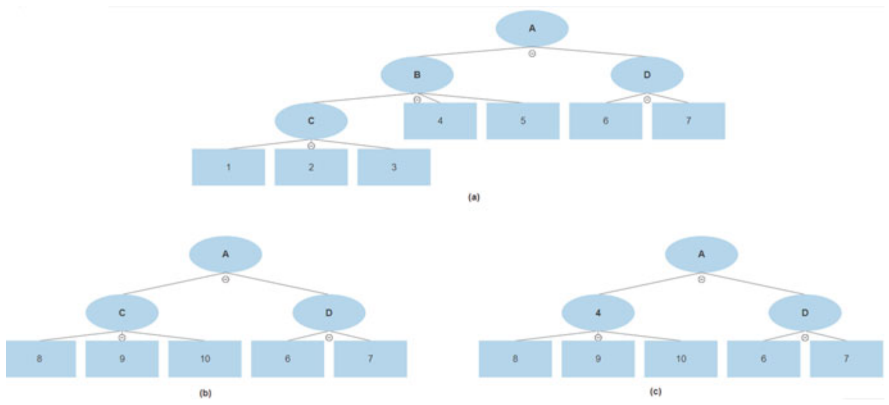


Fig. 8.16 Example of a fully developed tree before (**a**) and after (**b**, **c**) pruning

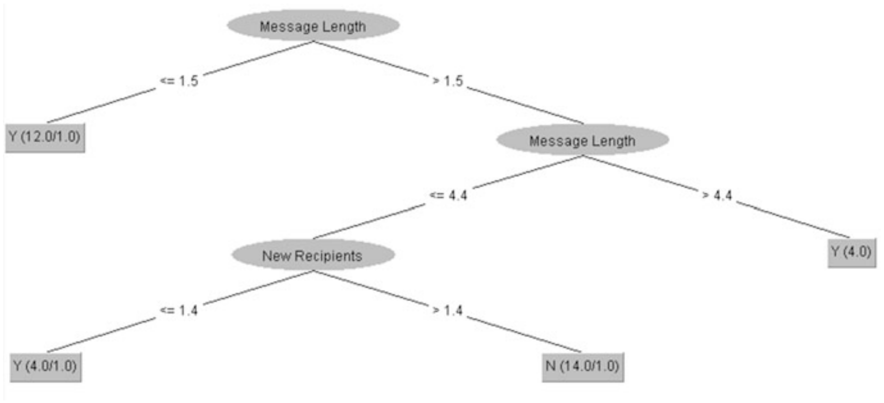


Fig. 8.17 Decision tree for the weather dataset using C.45 in Weka

4 has the most training instances; here, 8, 9, and 10 result from the reclassification of instances 1, 2, 3, 4, and 5.

In practice, if we apply the C.45 decision tree algorithm (called J48 in Weka) using pruning (done by default) to the weather data above, we will obtain the tree illustrated in Fig. 8.17.

In the leaf nodes, the first value in the parentheses is the number of instances from the training set in that leaf, while the second value is the number of instances incorrectly classified in that leaf.

8.4 Final Notes: Advantages, Disadvantages, and Best Practices

Decision trees are nonlinear algorithms, as opposed to the two linear algorithms we have introduced so far: linear regression and logistic regression. Linear discriminant analysis (LDA) is another traditional machine learning linear algorithm that we have not covered. Decision trees do not require specific data preparation steps and can be used for classification as well as for regression. However, in python, the tree-based algorithms in Python require numeric features only; hence, we need to transform categorical features to numeric ones using One-Hot Encoding.

8.5 Key Terms

1. Root
2. Leaf
3. decision node

4. Parent node
5. Child node
6. Subtree
7. Branch
8. Classification and regression trees
9. CART
10. Random forest
11. Binary tree
12. Entropy
13. Information gain
14. Gini index
15. Information gain Ratio
16. REPTree algorithm
17. Split criterion
18. Stop criterion
19. Decision stump
20. Entropy
21. Purity
22. Tree pruning
23. Overfitting
24. Pruning
25. Post-pruning
26. Backward pruning
27. Pre-pruning
28. Forward pruning
29. Subtree replacement
30. Subtree raising
31. C.45 algorithm
32. ID3

8.6 Test Your Understanding

1. Define the information gain ratio.
2. How do you compute the information gain ratio?
3. What does purity measure?
4. Give an example of a purity function.
5. What does CART stand for?
6. Which performs better, the REPTree algorithm or the J45 algorithm?
7. What does entropy measure exactly?
8. Define pre-pruning.
9. Define post-pruning.
10. Which is preferable, subtree raising or subtree replacement?

11. Can we use decision trees for regression analysis? Search for an example in the literature and summarize it.

8.7 Read More

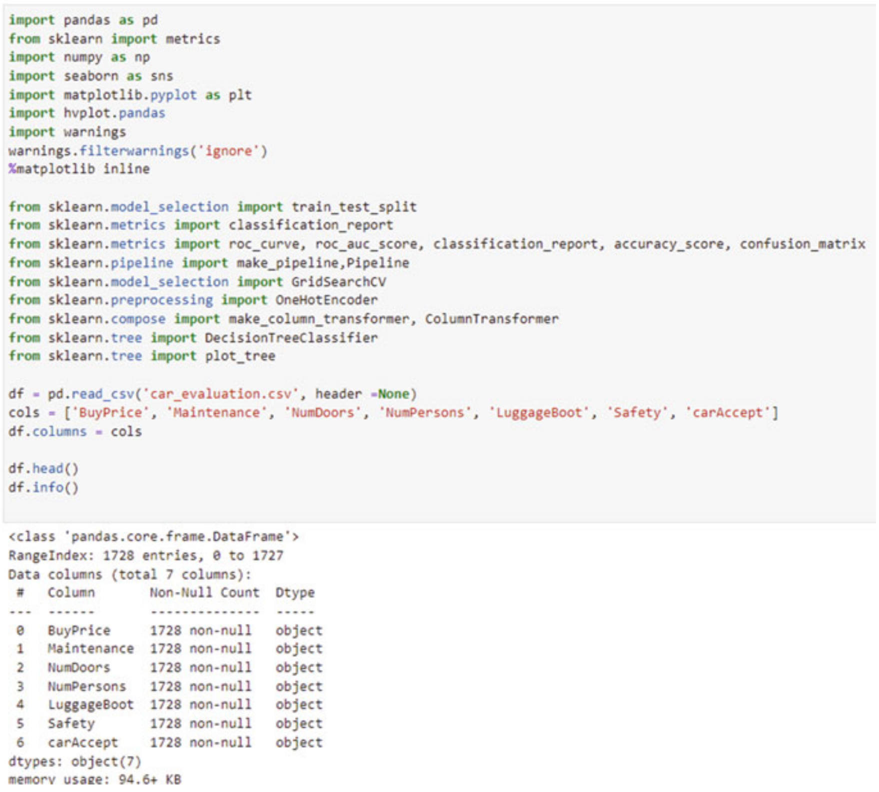
1. Arjoune, Y., & Faruque, S. (2020, 10–13 Dec. 2020). Real-time Machine Learning Based on Hoeffding Decision Trees for Jamming Detection in 5G New Radio. 2020 IEEE International Conference on Big Data (Big Data),
2. Bashar, S. S., Miah, M. S., Karim, A. H. M. Z., & Mahmud, M. A. A. (2019, 20–22 Dec. 2019). Extraction of Heart Rate from PPG Signal: A Machine Learning Approach using Decision Tree Regression Algorithm. 2019 fourth International Conference on Electrical Information and Communication Technology (EICT),
3. Bochkarev, B., & Rakitskiy, A. (2019, 21–27 Oct. 2019). The Study of the Applicability of Machine Learning Methods Based on Decision Trees for Holter Monitoring. 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON),
4. Chandrasegar, T., & Vutukuri, S. B. N. (2019, 22–23 March 2019). Optimized machine learning model using Decision Tree for cancer prediction. 2019 Innovations in Power and Advanced Computing Technologies (i-PACT),
5. Deen, A. J., & Gyanchandani, M. (2020, 8–10 Jan. 2020). Machine Learning Classifiers based on Predicting Membrane Protein using Decision Tree and Random Forest. 2020 Fourth International Conference on Inventive Systems and Control (ICISC),
6. Fontoura, L. C. M. M., Lins, H. W. D. C., Bertuleza, A. S., D'assunção, A. G., & Neto, A. G. (2021). Synthesis of Multiband Frequency Selective Surfaces Using Machine Learning With the Decision Tree Algorithm. IEEE Access, 9, 85,785–85,794. https://doi.org/8.1109/ACCESS.2021.3086777
7. Gupta, S. (2020, 10–13 Dec. 2020). A Hybrid Machine Learning Framework of Gradient Boosting Decision Tree and Sequence Model for Predicting Escalation in Customer Support. 2020 IEEE International Conference on Big Data (Big Data),
8. Hazra, R., Banerjee, M., & Badia, L. (2020, 4–7 Nov. 2020). Machine Learning for Breast Cancer Classification With ANN and Decision Tree. 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON),
9. Jimoh, I. A., Ismaila, I., & Olalere, M. (2019, 10–12 Dec. 2019). Enhanced Decision Tree-J48 With SMOTE Machine Learning Algorithm for Effective Botnet Detection in Imbalance Dataset. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO),
10. Kuang, W., Chan, Y., Tsang, S., & Siu, W. (2020). Machine Learning-Based Fast Intra Mode Decision for HEVC Screen Content Coding via Decision Trees.

IEEE Transactions on Circuits and Systems for Video Technology, 30(5), 1481–1496. https://doi.org/8.1109/TCSVT.2019.2903547
11. Li, Y., Song, X., Zhao, S., & Gao, F. (2020, 4–7 June 2020). A Line-Fault Cause Analysis Method for Distribution Network Based on Decision-Making Tree and Machine Learning. 2020 fifth Asia Conference on Power and Electrical Engineering (ACPEE),
12. Linty, N., Farasin, A., Favenza, A., & Dovis, F. (2019). Detection of GNSS Ionospheric Scintillations Based on Machine Learning Decision Tree. IEEE Transactions on Aerospace and Electronic Systems, 55(1), 303–317. https://doi.org/8.1109/TAES.2018.2850385
13. Mohagaonkar, S., Rawlani, A., & Saxena, A. (2019, 13–15 March 2019). Efficient Decision Tree using Machine Learning Tools for Acute Ailments. 2019 sixth International Conference on Computing for Sustainable Global Development (INDIACom),
14. Patil, S., & Kulkarni, U. (2019, 23–25 April 2019). Accuracy Prediction for Distributed Decision Tree using Machine Learning approach. 2019 third International Conference on Trends in Electronics and Informatics (ICOEI),
15. Posonia, A. M., Vigneshwari, S., & Rani, D. J. (2020, 3–5 Dec. 2020). Machine Learning based Diabetes Prediction using Decision Tree J48. 2020 third International Conference on Intelligent Sustainable Systems (ICISS),
16. Prapty, A. S., & Shitu, T. T. (2020, 19–21 Dec. 2020). An Efficient Decision Tree Establishment and Performance Analysis with Different Machine Learning Approaches on Polycystic Ovary Syndrome. 2020 23rd International Conference on Computer and Information Technology (ICCIT),
17. Romero, M. P., Chang, Y. M., Brunton, L. A., Parry, J., Prosser, A., Upton, P., Rees, E., Tearne, O., Arnold, M., Stevens, K., & Drewe, J. A. (2020). Decision tree machine learning applied to bovine tuberculosis risk factors to aid disease control decision making. Prev Vet Med, 175, 104,860. https://doi.org/8.1016/j.prevetmed.2019.104860
18. Shemu, N. A., Hossain, M. Z., Saleh, S. M., & Pavel, K. A. A. (2020, 9–10 Jan. 2020). A Machine Learning View for Health Data Mining Emphasizes on the Decision Trees. 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM),
19. Zhang, Y., Liu, J., Zhang, Z., & Huang, J. (2019, 12–14 July 2019). Prediction of Daily Smoking Behavior Based on Decision Tree Machine Learning Algorithm. 2019 IEEE ninth International Conference on Electronics Information and Emergency Communication (ICEIEC),

8.8 Lab

8.8.1 Working Example in Python

We will work with a car evaluation dataset that you can download from the following link:



```
import pandas as pd
from sklearn import metrics
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, roc_auc_score, classification_report, accuracy_score, confusion_matrix
from sklearn.pipeline import make_pipeline,Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer, ColumnTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

df = pd.read_csv('car_evaluation.csv', header =None)
cols = ['BuyPrice', 'Maintenance', 'NumDoors', 'NumPersons', 'LuggageBoot', 'Safety', 'carAccept']
df.columns = cols

df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   BuyPrice     1728 non-null   object
 1   Maintenance  1728 non-null   object
 2   NumDoors     1728 non-null   object
 3   NumPersons   1728 non-null   object
 4   LuggageBoot  1728 non-null   object
 5   Safety       1728 non-null   object
 6   carAccept    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

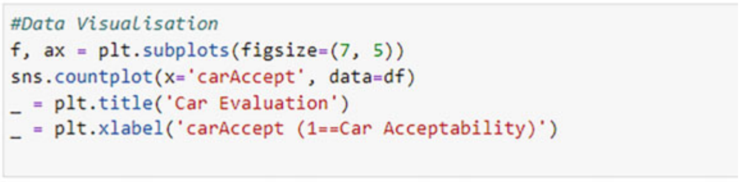
Fig. 8.18 Load car evaluation dataset

https://www.kaggle.com/elikplim/car-evaluation-data-set
This dataset evaluates cars (accept/reject) based on the following structure:

- Buying: buying car price
- Maintenance: maintenance car cost
- NumDoors: number of doors
- NumPersons: number of persons fitting in the car
- LuggageBoot: the size of the trunk (“luggage boot” in the UK)
- Safety: estimated safety of the car
- carAccept: car acceptability

8.8.1.1 Load Car Evaluation Dataset

Import the required libraries and install any library that you have not installed previously, then load the dataset into pandas in Fig. 8.18. Notice that we have read the csv file without the header and then added the column/features names according



```
#Data Visualisation
f, ax = plt.subplots(figsize=(7, 5))
sns.countplot(x='carAccept', data=df)
_ = plt.title('Car Evaluation')
_ = plt.xlabel('carAccept (1==Car Acceptability)')
```

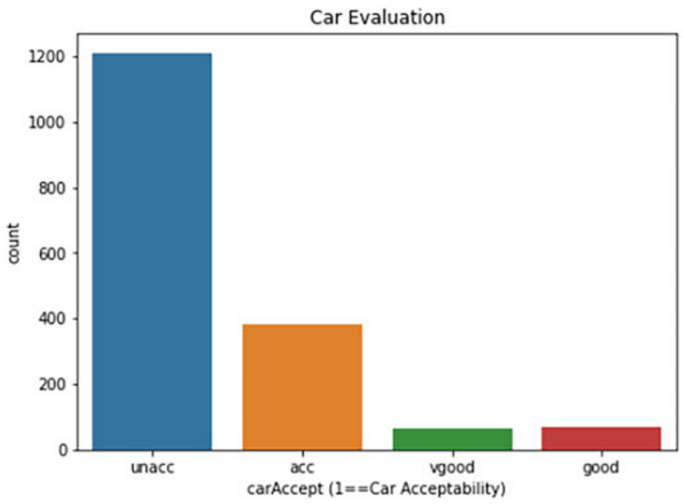


Fig. 8.19 Visualize car evaluation dataset and preprocess data

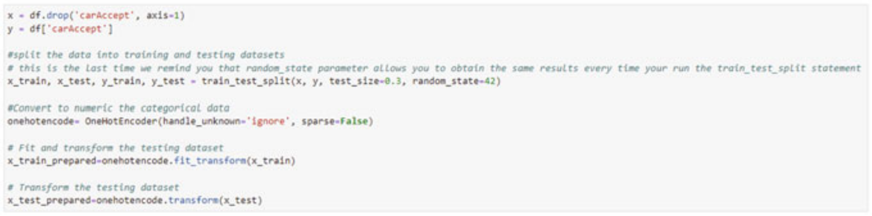
to our taste. None of the features present null values, so we will not process null value at this moment.

8.8.1.2 Visualize Car Evaluation

Visualize the data using the required libraries, we present in Fig. 8.19 a plot of the class (i.e., car evaluation).

8.8.1.3 Split and Scale Data

The next task is to split data into features vector x and class vector y, and then split x and y into training and testing datasets. Since trees in Python cannot process categorical features we need to one-hot encode all categorical features present in our datasets using OnEHotEncode. The result is two variables onehotencoded x_train_prepared and x-test_prepared (Fig. 8.20). You can display those variables to see the result.



```
x = df.drop('carAccept', axis=1)
y = df['carAccept']

#split the data into training and testing datasets
# this is the last time we remind you that random_state parameter allows you to obtain the same results every time your run the train_test_split statement
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

#Convert to numeric the categorical data
onehotencode= OneHotEncoder(handle_unknown='ignore', sparse=False)

# Fit and transform the testing dataset
x_train_prepared=onehotencode.fit_transform(x_train)

# Transform the testing dataset
x_test_prepared=onehotencode.transform(x_test)
```

Fig. 8.20 Converting and scaling data

8.8.1.4 Optimize Decision Tree Model

Grid search is used to tune the decision tree's hyperparamteers and find the optimal decision tree for the dataset. We also print the best parameters found and the best model, as well as its accuracy, and we plot the optimal decision tree (Fig. 8.21). Note that GridSearch will need several minutes to find the optimal tree, be patient.

Finally, we can test the best model by making predictions using the testing dataset. The performance (i.e., accuracy) is also computed and displayed (Fig. 8.22).

8.8.2 Working Example in Weka

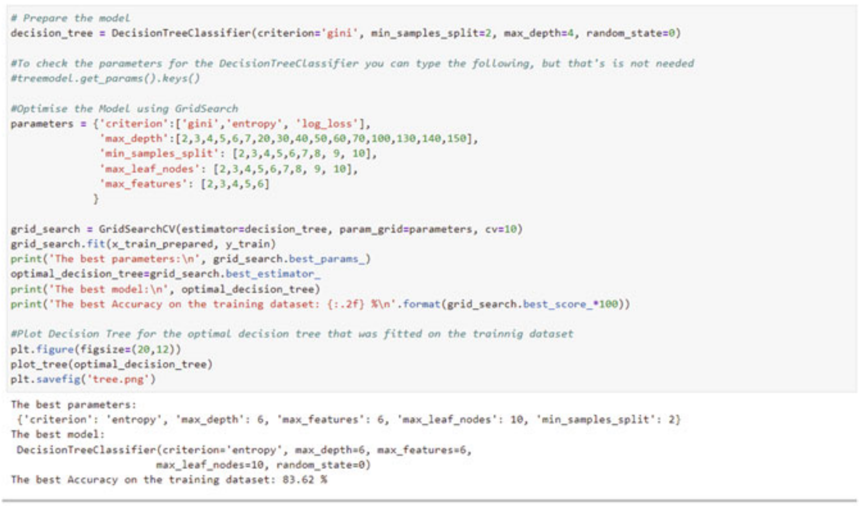
Download and open the iris dataset available from one of the following links:

1. https://archive-beta.ics.uci.edu/ml/datasets/53
2. https://archive.ics.uci.edu/ml/datasets/iris
3. https://www.kaggle.com/uciml/iris

The variables provided are as follows:

1. Id: identification
2. SepalLengthCm: sepal length in cm
3. SepalWidthCm: sepal width in cm
4. PetalLengthCm: petal length in cm
5. PetalWidthCm: petal width in cm
6. Species: the species (Iris-setosa, Iris-versicolor, or Iris-virginica)

1. Open the file in Weka and display the histograms for the four features.
2. Do you have an idea of which features are interesting for our classification problem? Probably not but take a few minutes to study the histograms and write your remarks.
3. Use the dataset to build a decision tree model to classify the irises into one of the three species based on the four features provided. You should be able to display the following output (Fig. 8.23) and tree (Fig. 8.24).



```
# Prepare the model
decision_tree = DecisionTreeClassifier(criterion='gini', min_samples_split=2, max_depth=4, random_state=0)

#To check the parameters for the DecisionTreeClassifier you can type the following, but that's is not needed
#treemodel.get_params().keys()

#Optimise the Model using GridSearch
parameters = {'criterion':['gini','entropy', 'log_loss'],
              'max_depth':[2,3,4,5,6,7,20,30,40,50,60,70,100,130,140,150],
              'min_samples_split': [2,3,4,5,6,7,8, 9, 10],
              'max_leaf_nodes': [2,3,4,5,6,7,8, 9, 10],
              'max_features': [2,3,4,5,6]
             }

grid_search = GridSearchCV(estimator=decision_tree, param_grid=parameters, cv=10)
grid_search.fit(x_train_prepared, y_train)
print('The best parameters:\n', grid_search.best_params_)
optimal_decision_tree=grid_search.best_estimator_
print('The best model:\n', optimal_decision_tree)
print('The best Accuracy on the training dataset: {:.2f} %\n'.format(grid_search.best_score_*100))

#Plot Decision Tree for the optimal decision tree that was fitted on the trainnig dataset
plt.figure(figsize=(20,12))
plot_tree(optimal_decision_tree)
plt.savefig('tree.png')
```

```
The best parameters:
 {'criterion': 'entropy', 'max_depth': 6, 'max_features': 6, 'max_leaf_nodes': 10, 'min_samples_split': 2}
The best model:
 DecisionTreeClassifier(criterion='entropy', max_depth=6, max_features=6,
                        max_leaf_nodes=10, random_state=0)
The best Accuracy on the training dataset: 83.62 %
```

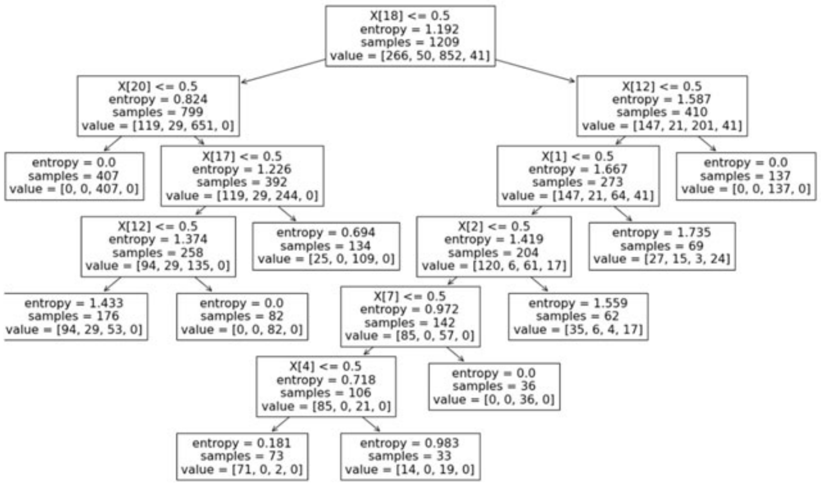
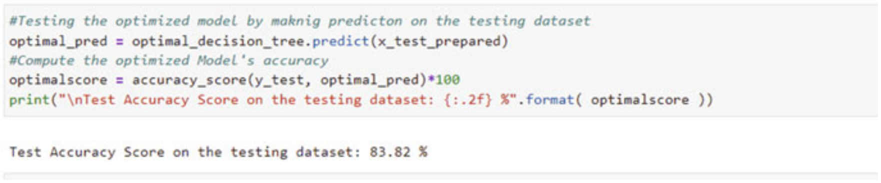


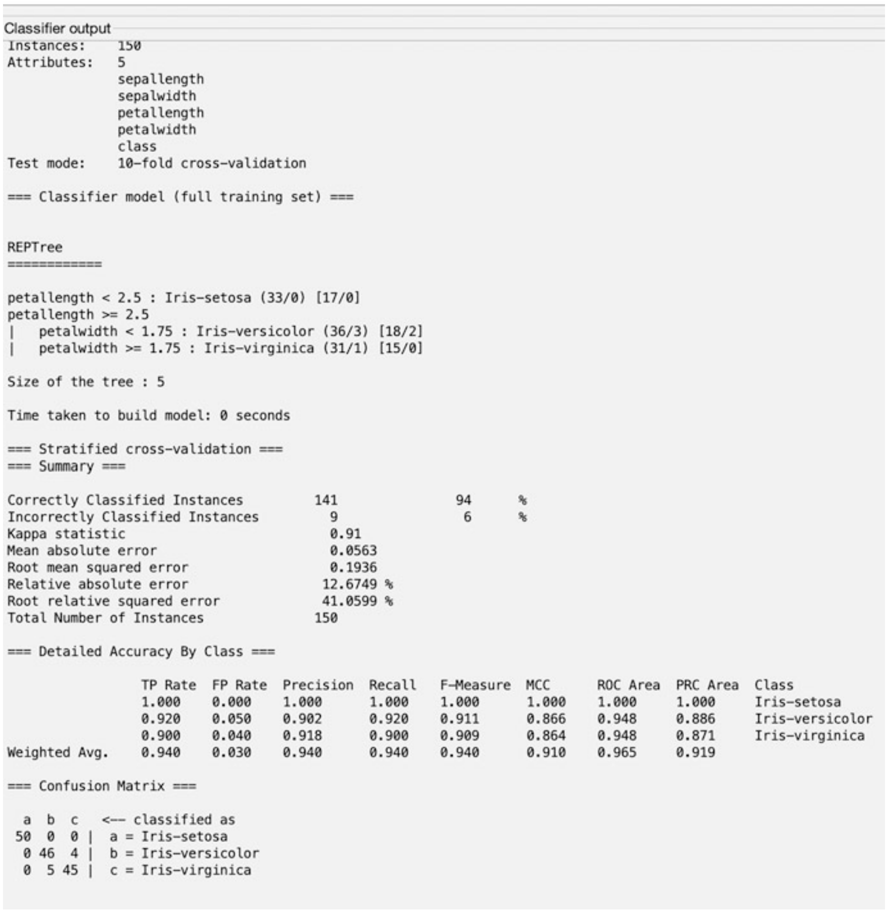
Fig. 8.21 Optimizing decision tree model using grid search cross-validation



```
#Testing the optimized model by maknig predicton on the testing dataset
optimal_pred = optimal_decision_tree.predict(x_test_prepared)
#Compute the optimized Model's accuracy
optimalscore = accuracy_score(y_test, optimal_pred)*100
print("\nTest Accuracy Score on the testing dataset: {:.2f} %".format( optimalscore ))
```

```
Test Accuracy Score on the testing dataset: 83.82 %
```

Fig. 8.22 Testing the best model and printing its accuracy on the testing dataset



```
Classifier output
Instances:     150
Attributes:    5
               sepallength
               sepalwidth
               petallength
               petalwidth
               class
Test mode:     10-fold cross-validation

=== Classifier model (full training set) ===


REPTree
============

petallength < 2.5 : Iris-setosa (33/0) [17/0]
petallength >= 2.5
|   petalwidth < 1.75 : Iris-versicolor (36/3) [18/2]
|   petalwidth >= 1.75 : Iris-virginica (31/1) [15/0]

Size of the tree : 5

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         141               94      %
Incorrectly Classified Instances         9                6      %
Kappa statistic                          0.91
Mean absolute error                      0.0563
Root mean squared error                  0.1936
Relative absolute error                 12.6749 %
Root relative squared error             41.0599 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.920    0.050    0.902      0.920    0.911      0.866    0.948     0.886     Iris-versicolor
                 0.900    0.040    0.918      0.900    0.909      0.864    0.948     0.871     Iris-virginica
Weighted Avg.    0.940    0.030    0.940      0.940    0.940      0.910    0.965     0.919

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 46  4 |  b = Iris-versicolor
  0  5 45 |  c = Iris-virginica
```

Fig. 8.23 Classifier output

8.8.3 *Do It Yourself*

8.8.3.1 Decision Tree: Reflections on the Car Evaluation Dataset

You can notice that the optimal decision tree in Fig. 8.21 does not provide feature names in the leaves, instead we have x[1], x[2], etc. This is because x_train_prepared is an array that has no titles.

1. Is it possible for you to convert it to a data frame (x_train_prepared_df) and provide appropriate feature names for the data frame's columns? Hint: use pandas.DataFrame for conversion and .columns to provide name for the columns like we did in Fig. 8.18.

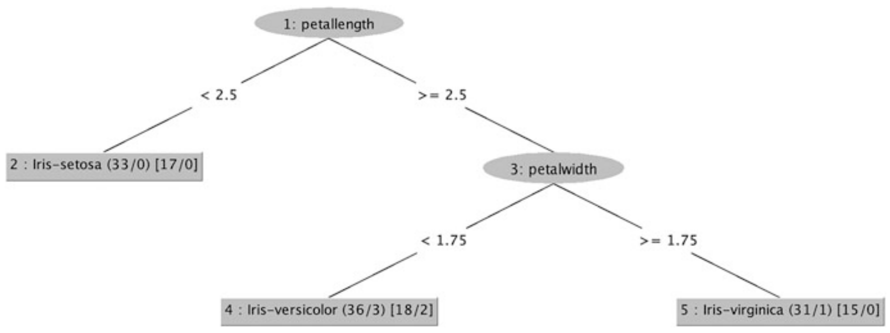

1: petallength
< 2.5
>= 2.5
2 : Iris-setosa (33/0) [17/0]
3: petalwidth
< 1.75
>= 1.75
4 : Iris-versicolor (36/3) [18/2]
5 : Iris-virginica (31/1) [15/0]

Fig. 8.24 Decision tree for the iris classification problem

2. If this is doable then can you use x_ train _prepared_df to fit a new similar model to the one presented in Fig. 8.20 with appropriate column names.
3. Trees are prone to overfitting. The minimum number of samples a node can have before it is a candidate for splitting (min_samples_split), the minimum number of samples a leaf must have (min_samples_leaf), the maximum number of leaf nodes (max_leaf_nodes) and the maximum number of features evaluated for splitting at each node (max_features) can regularize the tree: increasing the min_ hyperparameters and decreasing the max_ hyperparameters values. Try to reduce the change in the parameters provided to gridsearch; for example, remove some of the min and max hyperparameters or all, or add min_samples_leaf and see how the optimal tree and its accuracy change.
4. Can you notice overfitting when you remove parameters related to regularization? Explain.

8.8.3.2 Decision Trees for Regression

We have seen decision trees for classification. In this exercise, we will overview decision trees' use for regression. Download the Boston housing dataset from the following link: https://www.kaggle.com/prasadperera/the-boston-housing-dataset. The dataset is also available from the numeric dataset you downloaded previously.

The dataset is composed of the following features:

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq. ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxide concentration (parts per ten million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centers
9. RAD: index of accessibility to radial highways

10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: $1000(Bk - 0.63)^2$,where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: median value of owner-occupied homes in $1000s

1. Build a decision tree model to predict the median value of a Boston house (MEDV) based on the available features. Below is a sample output from Weka.
2. Do you want to do any preprocessing? Which processing? For which feature?
3. Compare your results before and after preprocessing.
4. Any notes about the data? Do you think that this data relates to questions of bias and racism? Explain your answer.
5. Do you know of any machine learning applications that have previously raised ethical concerns?
6. Do you have any ethical concerns regarding future applications for machine learning?

8.8.3.3 Decision Trees for Classification

Download the train and test datasets of the *Titanic* from https://www.kaggle.com/c/titanic/data?select=train.csv.

The variables provided are as follows:

1. Survival: 0 = No, 1 = Yes
2. Pclass is the ticket class: 1 = first, 2 = second, 3 = 3rd
3. Sex: M or F
4. Age: age in years
5. Sibsp: number of siblings or spouses aboard the ship
6. Parch: number of parents or children aboard the ship
7. Ticket: ticket number
8. Fare: passenger fare
9. Cabin: cabin number
10. Embarked: the port of embarkation, C=Cherbourg, Q = Queenstown, S=Southampton

1. Use decision trees to build a model that predicts the survival of a passenger based on the available features. Note the accuracy of the algorithm and other measurements and display the decision tree.
2. Which other algorithm can you use to tackle this classification problem? Suggest one and execute the necessary instructions to build a new classification model.
3. Compare the two models (the decision tree model and the other suggested model). Which one makes better decisions? On which data have you based your decision?

8.8.4 Do More Yourself

1. Mushroom dataset (classification): https://www.kaggle.com/uciml/mushroom-classification
2. London bike-sharing dataset (regression): https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset

References

1. A. Burkov, *The Hundred-Page Machine Learning Book* (Andriy Burkov, 2019)
2. O.Z. Maimon, L. Rokach, *Data Mining with Decision Trees: Theory and Applications*, 2nd edn. (World Scientific Publishing Company, 2014)
3. V.G. Sigillito, S.P. Wing, L.V. Hutton, K.B. Baker, Classification of radar returns from the ionosphere using neural networks. J. Hopkins APL Tech. Dig. **10**, 262–266 (1989)
4. J.R. Quinlan, Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1 March 1986). https://doi.org/10.1007/BF00116251
5. I.H. Witten, E. Frank, M.A. Hall, C. Pal, *Data Mining: Practical Machine Learning Tools and Techniques* (Elsevier Science, 2016)

Chapter 9
Naïve Bayes

9.1 The Problem

So far, during classification, we have been interested in finding a model that decides if an instance belongs to a class or not; the model's answer would be a yes or no with certainty. The situation with Bayesian modeling for decision-making is different—it estimates the probability that an instance belongs to a certain class, which is more nuanced [1].

While Bayesian classification models are more sophisticated and provide more fine-tuned optimal solutions for classification than other classifiers, in practice, they make assumptions about the probability distribution of the existing variables, and they estimate these probabilities based on the available data and assumptions about the underlying probability distribution, as we will see below. That is why other classifiers are needed despite being less sophisticated.

Nevertheless, classifiers based on Bayesian reasoning, like the naïve Bayes classifier, are among the most effective for learning tasks such as text document classification, and for very large datasets. Google employs a naïve Bayes classifier to autocorrect text typed by users [2]; other typical applications include information retrieval [3], medical diagnosis [4–9], spam filtering [10], classifying documents (e.g., spam or non-spam), sentiment prediction, and recommendation systems [11]. The naïve Bayes classifier assumes that all the features (i.e., input variables, predictors) are conditionally independent given a class.

Linear regression, logistic regression, and decision tree models were formulated to guess the class based on the instance values; they guess the conditional probability of a class is true given an instance (i.e., values of feature vectors); we can denote this probability as P(class|instance) [12]. Another way of proceeding is to try to understand a class first and then to infer the classification prediction when we meet an instance; this resembles in a way a person learning about different characteristics of some item (e.g., a tree, a house) and then, when seeing two pictures, deciding which one is the picture of a tree and which one is of a house. Here, we try to guess the instance's class given the classes' characteristics: the probability of an instance being in a class gives our knowledge about class P(instance|class).

9.2 The Algorithm

9.2.1 Bayes Theorem

Thomas Bayes, from the eighteenth century, created Bayes' theorem, which describes how to calculate the probability of a variable y given an observation x, also known as *posterior probability* of y: $P(y|x)$. For our machine learning classification problem, we can express the posterior probability as $P(\text{class} = C|\text{instance} = x)$: given an instance x, what is the probability that the class is C?

Bayes' theorem provides a way to compute posterior probabilities as follows:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

In a machine learning classification problem, we have a set of training data D with N observations/instances (x_i, y_i), $i = 1$ to N; x_i being the feature vector (i.e., input variable) and y_i the outcome variable/class. D:$\{x^{(i)},y^{(i)}\}$; $i = 1, 2, \ldots, N$.

In the training dataset D, each input variable has n attributes (i.e., features); hence, each instance x can be expressed as a vector of values x_j; $j = 1, \ldots, n$. Also, the possible values for y_i are the M classes; hence, we have M different values y_q; $q = 1, \ldots, M$.

The probability that x belongs to a class y_q before (a priori of) looking at any data instance x_i is expressed by the prior probability $P(y_q)$. The sum of all probabilities of the M classes is 1; $P(y_1) + \cdots + P(y_M) = 1$.

Based on Bayes' theorem, the posterior probability $P(y_q|x)$; $q \in \{1, \ldots, M\}$ is calculated based on the known a priori probability $P(y_q)$ and the known conditional probabilities $P(x|y_q)$; $q = 1, \ldots, M$.

$$P(y_q|x) = \frac{P(x|y_q)P(y_q)}{P(x)}$$

where

$$P(x) = \sum_{q=1}^{M} P(x|y_q)P(y_q)$$

Therefore:

$$P(y_q|x) = \frac{P(x|y_q)P(y_q)}{\sum_{q=1}^{M} P(x|y_q)P(y_q)}$$

Since the posterior probability $P(x|\, y_q)$ as well as the a priori probabilities $P(y_q)$ for each class y_q, $q = 1 \ldots M$, are known, we can deduce the probability of each class y_q given an observation x, $P(y_q|\, x)$, and choose the class with the highest probability $P(y_q|\, x)$ to classify instance x.

The decision can be made based on the largest posterior probability P(class| instance) for all classes. The Bayesian decision principle of predicting the class with the largest posterior probability can be written as an "argument of the maxima," or arg max, which is the point of a domain at which a function is maximized. In our case, it is the class y_k for which the posterior probability of an instance x, $P(y_k|x)$, is maximized; y_k is called the maximum a posteriori (MAP) class.

$$y_{\text{MAP}} = \arg_q^{\max} \; P(y_q|x) = \arg_q^{\max} \; \frac{P(x|y_q)P(y_q)}{P(x)}$$

Given that the denominator is independent of y:

$$y_{\text{MAP}} = \arg_q^{\max} \; P(y_q)P(x|y_{q.})$$

When we have only two classes, 1 and 0, the result becomes:

$$y_{\text{MAP}} = \arg_q^{\max} \; (P(x|y=0)P(y=0),\, P(x|y=1)P(y=1))$$

If we were in a case where all the classes were equally probable, then we could simplify and remove $P(y_q)$:

$$y_{\text{MAP}} = \arg_q^{\max} \; P(x|y_q)$$

In this case, since $P(x|\, y_q)$ measures the likelihood that x fits in y_q, the class that maximizes $P(x|\, y_q)$ is called the maximum likelihood (ML) class because we will be searching for the likelihood [2].

9.2.2 *The Naïve Bayes Classifier (NBC): Dealing with Categorical Variables*

In the real world, we might not know in advance $P(y^q)$ and $P(x|y_q)$, which are both necessary to compute $P(y_q|x)$. Naïve Bayes tries to overcome this difficulty.

To classify a feature vector x into one of the M classes, we can proceed with a simple rule that does not rely on any need for knowledge derived from x: x belongs to the majority class. As an example, suppose that you have a dataset for buses running throughout a city and that you have seven attributes for each bus: manufacturer, manufacturing year, years in service, number of times the bus has experienced mechanical failure, origin, destination, and current weather. Suppose you want to build a model to predict if a bus x will be delayed or not, but you know that 90% of the time the buses in the city are on time. Based on this knowledge, you can ignore all the seven attributes' values and rely on a *naïve* rule that bus x will be on time. Most probably, your rule will correctly work most of the time; however, a model built based on the seven attributes must perform better.

Another unrealistic approach consists of using the available attributes and considering them as equally important and *independent* of each other. The approach is unrealistic because, in real life, the attributes might depend on each other and most probably are not equally important in relation to our outcome. However, this unrealistic approach works surprisingly well in real life, and it is the approach taken by the naïve Bayes classifier (NBC). It is important to note that in this paragraph, we are considering categorical features; in the case of continuous features (e.g., age), we can always transform them to categorical ones through discretization, also known as binning (e.g., [1–17], [18–20], [21–30], [31–40], > 40).

To compute $P(y_q| x)$,we need $P(x| y_q)$ and $P(y_q)$.

$$P(y_q|x) = \frac{P(x|y_q)P(y_q)}{\sum_{q=1}^{M} P(x|y_q)P(y_q)}$$

we can estimate $P(y_q)$ as follows:

$$P(y_q) = \frac{\text{Number of instances in class } y_q}{\text{Total number of instances}}$$

while $P(x| y_q)$is estimated as follows:

$$P(x|y_q) = \frac{\text{Number of times } x \text{ appears in the class } y_q}{\text{Number of times } y_q \text{ appears in the dataset}}$$

To compute $P(\mathrm{x}| y_q)$, we need the feature vector x to appear many times in many classes, which is only possible in large datasets; however, there is a way around it:

$$P(x|y_q) = P(x_1, x_2 \ldots x_n)|y_q)$$

Since naïve Bayes assumes that the attributes are independent of each other, then

$$P(x_1, x_2 \ldots x_n)|y_q) = P(x_1|y_q) \times P(x_2|y_q) \ldots \times P(x_n|y_q)$$

This allows us to estimate $P(x|y_q)$ as follows:

$$P(x|y_q) = P(x_1|y_q) \times P(x_2|y_q) \ldots \times P(x_n|y_q)$$

$$P(x|y_q) = \prod_{i=1}^{n} P(x_i|y_q)$$

Finally, the naïve Bayes classifier class (y_{NB}) can be deduced from the following:

$$y_{\text{NB}} = \arg_q^{\max} P(y_q)P(x|y_q) = \arg_q^{\max} P(y_q)\prod_{i=1}^{n} P(x_i|y_q)$$

9.2.3 *Gaussian Naïve Bayes (GNB): Dealing with Continuous Variables*

Till now, we have mentioned that the naïve Bayes classifier works with categorical variables and that we can discretize continuous variables to handle continuous variables [13–15]. However, NBC also works with continuous variables using two methods:

- The *normal method* approximates the distribution of the continuous variable using parameterized distribution such as the normal distribution, also called Gaussian distribution. The word "parametric" means to rely on assumptions about the form of the distribution in the population and its parameters (e.g., mean, standard deviation). Non-parametric procedures do not rely on such assumptions.
- The *kernel method* uses a non-parameterized approximation of the distribution of the continuous variable [16].

A naïve Bayes classifier that uses a Gaussian distribution is called a Gaussian naïve Bayes classifier. For each class y_q and for each attribute, we compute the mean and the standard deviation (SD) as follows:

$$\text{mean}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\text{SD}(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \text{mean}(x))^2}$$

where n is the number of instances in the class y_q.

For each attribute, an estimate of the probability of x belonging to a class y_q is given by the probability density function (PDF):

$$P(x|y_q) = \text{pdf}(x, \text{mean}, \text{SD}) = \frac{1}{\sqrt{2\pi} \times \text{SD}} e^{-\left(\frac{(x-\text{mean})^2}{2\ \text{SD}^2}\right)}$$

where mean and SD are computed for each attribute in the class y_q.

Then, to make predictions with continuous variables, we replace each $P(x_i| y_q)$ with its estimate $\text{pdf}(x, \text{mean}_C, \text{SD}_C)$ as follows:

$$y_{\text{NB}} = \arg_q^{\max} P(y_q) \prod_{i=1}^{n} p(x_i|y_q) = \arg_q^{\max} p(y_q) \prod_{i=1}^{n} \text{pdf}(x_i)$$

9.3 A Practical Example

9.3.1 Naïve Bayes Classifier with Categorical Variables Example

Consider the following dataset formed of 15 instances provided by Gopal [2] (Table 9.1).

Height is a continuous variable that we can discretize by using the following six categories: [0–1.6],]1.6–1.7],]1.7–1.8],]1.8–1.9],]1.9–2.0],]2.0–∞[.

The dataset size is $N = 15$, and the number of classes is $M = 3$.

We can now generate the following useful table for our calculations (Table 9.2):

Let us consider an instance of the dataset (or a new instance) $x = \{x_1, x_2\} = \{M, 1.95\}$. To classify this instance x, we need to compute $P(y_q)$ and $P(x| y_q)$ for every value of the outcome (q) and choose the class of the maximum value $P(y_q) \times P(x| y_q)$.

Table 9.1 Dataset describing gender, height, and outcome of three classes

Gender (x_1)	Height (x_2)	Class (y)	y
F	1.6 m	Short	S
M	2 m	Tall	T
F	1.9 m	Medium	M
F	1.88 m	Medium	M
F	1.7 m	Short	S
M	1.85 m	Medium	M
F	1.6 m	Short	S
M	1.7 m	Short	S
M	2.2 m	Tall	T
M	2.1 m	Tall	T
F	1.8 m	Medium	M
M	1.95 m	Medium	M
F	1.9 m	Medium	M
F	1.8 m	Medium	M
F	1.75 m	Medium	M

Table 9.2 The number of training samples by class for each possible value of the input variables

Input features	Feature values	y_1=Short $q = 1$	y_2=Medium $q = 2$	y_3=Tall $q = 3$
x_1 = gender	M	1	2	3
	F	3	6	0
x_2 = height	]0, 1.6] bin	2	0	0
	]1.6, 1.7] bin	2	0	0
	]1.7, 1.8] bin	0	3	0
	]1.8,1.9] bin	0	4	0
	]1.9, 2.0] bin	0	1	1
	]2.0, ∞] bin	0	0	2

$$y_{\mathrm{NB}} = \arg_q^{\max} P(y_q)P(x|y_q) = \arg_q^{\max} P(y_q)\prod_{i=1}^{n} P(x_i|y_q)$$

Since x:{M, 1.95}, then the calculations are as follows:

$P(x_1|y_1) = 1/4$; $P(x_1|y_2) = 2/8$; $P(x_1|y_3) = 3/3$
$P(x_2|y_1) = 0/4$; $P(x_2|y_2) = 1/8$; $P(x_2|y_3) = 1/3$

The next step is to compute the factors $P(x|y_q)$ equal to $\prod_{i=1}^{n} P(x_i|y_q)$:

$P(x|y_1) = P(x_1|y_1)\times P(x_2|y_1) = 1/4\times 0/4 = 0$
$P(x|y_2) = P(x_1|y_2)\times P(x_2|y_2) = 2/8\times 1/8 = 1/32$
$P(x|y_3) = P(x_1|y_3)\times P(x_2|y_3) = 3/3\times 1/3 = 1/3$

Then we can compute all $P(y_q)$:

$P(y_1) = 4/15 = 0.267$
$P(y_2) = 8/15 = 0.533$
$P(y_3) = 3/15 = 0.2$

Finally, we compute $P(y_q)P(x|y_q)$ for all the q values:

$P(x|y_1)P(y_1) = 0\times 0.267 = 0$
$P(x|y_1)P(y_1) = 1/32\times 0.533 = 0.017$
$P(x|y_1)P(y_1) = 1/32\times 0.2 = 0.066$

The maximum value (0.066) is in relation to class 3 (Tall); hence, the decision is to classify the instance x = {M, 1.95} in the class Tall.

9.3.2 Gaussian Naïve Bayes Example

Consider the following dataset provided by Brownlee [17], with two continuous input variables, $X1$ and $X2$, and one output variable, Y, with two classes, 0 and 1 (Table 9.3).

The class probabilities are $P(Y = 0) = P(Y = 1) = 4/8 = 0.5$

Table 9.3 A sample dataset to describe dummy input variables, $X1$ and $X2$, classified into two classes, $Y = 0$ and $Y = 1$

$X1$	$X2$	Y
3.39353321	2.33127338	0
3.11007348	1.78153964	0
1.34380883	3.36836095	0
3.58229404	4.67917911	0
2.28036244	2.86699026	0
7.42343694	4.69652288	1
5.745052	3.5339898	1
9.17216862	2.51110105	1
7.79278348	3.42408894	1
7.93982082	0.79163723	1

Table 9.4 Means and standard deviations of each input variable by class

	$X1 \mid Y = 0$	$X1 \mid Y = 1$	$X2 \mid Y = 0$	$X2 \mid Y = 1$
Mean	2.7420144	7.614652372	3.005468669	2.991467979
SD	0.92656833	1.234432155	1.107329589	1.454193138

Table 9.5 pdf($X1$), pdf($X2$), and $P(Y = 0)$×pdf($X1$)×pdf($X2$) calculations for each instance of the dataset for the class $Y = 0$

Prediction: $Y = 0$				
$X1$	$X2$	pdf($X1$)	pdf($X2$)	$P(Y = 0) \times \text{pdf}(X_1) \times \text{pdf}(X_2)$
3.393533211	2.331273381	0.336255919	0.299321284	0.050324277
3.110073483	1.781539638	0.397895380	0.195590604	0.038912299
1.343808831	3.368360954	0.137898926	0.341437869	0.023541958
3.582294042	4.679179110	0.285395252	0.114958842	0.016404354
2.280362439	2.866990263	0.380301158	0.357468012	0.067972749
7.423436942	4.696522875	0.000001233	0.112255509	0.000000069
5.745051997	3.533989803	0.002254527	0.321488218	0.000362402
9.172168622	2.511101045	0.000000000	0.326100779	0.000000000
7.792783481	3.424088941	0.000000152	0.335427703	0.000000025
7.939820817	0.791637231	0.000000063	0.048830746	0.000000002

Applying the naïve Bayes classifier using continuous variables described above, we will calculate the mean and standard deviation for $X1$ and $X2$, for the class $Y = 0$ and the class $Y = 1$ (Table 9.4).

The calculation of PDF for each of the classes is done using the PDF function, and the result is shown in Tables 9.5 and 9.6.

$$P(x|y_q) = \text{pdf}(x, \text{mean}_C, \text{SD}_C) = \frac{1}{\sqrt{2\pi} \times \text{SD}_C} e^{-\left(\frac{(x-\text{mean}_C)^2}{2\ \text{SD}_C^2}\right)}$$

Table 9.6 pdf($X1$), pdf($X2$), and $P(Y = 0)$×pdf($X1$)×pdf($X2$) calculations for each instance of the dataset for the class $Y = 1$

Prediction: $Y = 1$				
$X1$	$X2$	pdf($X1$)	pdf($X2$)	$P(Y = 1) \times \text{pdf}(X_1) \times \text{pdf}(X_2)$
3.393533211	2.331273381	0.000934051	0.247475201	0.000115577
3.110073483	1.781539638	0.000414867	0.194072317	0.000040257
1.343808831	3.368360954	0.000000805	0.265278247	0.000000107
3.582294042	4.679179110	0.001557322	0.139894652	0.000108931
2.280362439	2.866990263	0.000028485	0.273336035	0.000003893
7.423436942	4.696522875	0.319324693	0.137961765	0.022027299
5.745051997	3.533989803	0.102645949	0.255896584	0.013133374
9.172168622	2.511101045	0.145798903	0.259772393	0.018937265
7.792783481	3.424088941	0.319831448	0.262463684	0.041972070
7.939820817	0.791637231	0.312158739	0.087370630	0.013636753

Table 9.7 Classification result

Final Predictions		
$X1$	$X2$	Prediction
3.39353321	2.33127338	0
3.11007348	1.78153964	0
1.34380883	3.36836095	0
3.58229404	4.67917911	0
2.28036244	2.86699026	0
7.42343694	4.69652288	1
5.745052	3.5339898	1
9.17216862	2.51110105	1
7.79278348	3.42408894	1
7.93982082	0.79163723	1

To decide on the classification of ($X1_i$, $X2_i$), we compare the two resulting numbers $P(Y = 0) \times \text{pdf}(X1) \times \text{pdf}(X2)$ and $P(Y = 1) \times \text{pdf}(X1) \times \text{pdf}(X2)$ for each ($X1_i$, $X2_i$); $i = 1.10$.

The classification result is shown in Table 9.7.

9.4 Final Notes: Advantages, Disadvantages, and Best Practices

The naïve Bayes classifier performs well in multiclass prediction and is fast to execute. With continuous variables, either they can be discretized and transformed into categorical data, or a Gaussian Naïve Bayes classifier is used.

While the naïve Bayes classifier is not sophisticated, evidence shows that it performs well (and even better in the case of large datasets [18]) in comparison to other algorithms (e.g., decision trees and neural networks).

One of the drawbacks of naïve Bayes is the assumption of independence among the predictors, which is not true in real life. However, in machine learning, we always try the simplest strategies first; trying logistic or linear regressions is always a good idea, as the model they produce is quite easy to understand: an outcome as a function of the input. The naïve Bayes classifier is also a good strategy for large datasets and document classification.

9.5 Key Terms

1. Naïve Bayes classifier
2. Posterior probability
3. Argument of the maxima
4. arg max
5. Maximum a posteriori class
6. MAP class
7. Maximum likelihood class
8. ML class
9. Normal method
10. Kernel method
11. Gaussian naïve Bayes classifier

9.6 Test Your Understanding

1. How does the Bayesian approach differ from decision trees or regression approaches?
2. How is Bayes' theorem used for classification?
3. What is the aim of a naïve Bayes classifier?
4. What is the aim of a Gaussian naïve Bayes (GNB) classifier?
5. We have not defined the kernel method for classification of continuous variables; search for information and explain how it works.

9.7 Read More

1. Adikara, P. P., Adinugroho, S., & Insani, S. (2020). Detection of cyber harassment (cyberbullying) on Instagram using naïve bayes classifier with bag of words and lexicon based features Proceedings of the fifth International Conference on Sustainable Information Engineering and Technology, Malang, Indonesia. https://doi.org/10.1145/3427423.3427436
2. Aguinta, D. J., Adikara, P. P., & Wihandika, R. C. (2020). Sentiment analysis of mass rapid transit jakarta using naïve bayes classifier and rule-based opinion

target detection on Twitter Proceedings of the fifth International Conference on Sustainable Information Engineering and Technology, Malang, Indonesia. https://doi.org/10.1145/3427423.3427437

3. Aldossari, B. S., Alqahtani, F. M., Alshahrani, N. S., Alhammam, M. M., Alzamanan, R. M., & Irfanullah, N. A. (2020). A Comparative Study of Decision Tree and Naive Bayes Machine Learning Model for Crime Category Prediction in Chicago Proceedings of 2020 the sixth International Conference on Computing and Data Engineering, Sanya, China. https://doi.org/10.1145/3379247.3379279
4. Azeraf, E., Monfrini, E., & Pieczynski, W. (2021). Using the Naive Bayes as a discriminative model 2021 13th International Conference on Machine Learning and Computing, Shenzhen, China. https://doi.org/10.1145/3457682.3457697
5. Guo, B., & Zheng, Q. (2018). Using Naïve Bayes Algorithm to Estimate the Response to Drug in Lung Cancer Patients. Comb Chem High Throughput Screen, 21(10), 734–748. https://doi.org/10.2174/1386207322666190125151624
6. Gutiérrez, A. M., Pacheco, P. A., Gutiérrez, J. C., & Bressan, G. (2019). Development of a naive bayes classifier for image quality assessment in biometric face images Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, Rio de Janeiro, Brazil. https://doi.org/10.1145/3323503.3360622
7. He, W., He, Y., Li, B., & Zhang, C. (2020). A Naive-Bayes-Based Fault Diagnosis Approach for Analog Circuit by Using Image-Oriented Feature Extraction and Selection Technique. IEEE Access, 8, 5065–5079. https://doi.org/10.1109/ACCESS.2018.2888950
8. Jiang, W., Shen, Y., Ding, Y., Ye, C., Zheng, Y., Zhao, P., Liu, L., Tong, Z., Zhou, L., Sun, S., Zhang, X., Teng, L., Timko, M. P., Fan, L., & Fang, W. (2018). A naive Bayes algorithm for tissue origin diagnosis (TOD-Bayes) of synchronous multifocal tumors in the hepatobiliary and pancreatic system. Int J Cancer, 142(2), 357–368. https://doi.org/10.1002/ijc.31054
9. Lagman, A. C., Calleja, J. Q., Fernando, C. G., Gonzales, J. G., Legaspi, J. B., Ortega, J. H. J. C., Ramos, R. F., Solomo, M. V. S., & Santos, R. C. (2019). Embedding naïve Bayes algorithm data model in predicting student graduation Proceedings of the third International Conference on Telecommunications and Communication Engineering, Tokyo, Japan. https://doi.org/10.1145/3369555.3369570
10. Li, D., Sun, J., Yang, H., & Wang, X. (2020). An Enhanced Naive Bayes Model for Dissolved Oxygen Forecasting in Shellfish Aquaculture. IEEE Access, 8, 217,917–217,927. https://doi.org/10.1109/ACCESS.2020.3042180
11. Li, L. X., & Abdul Rahman, S. S. (2018). Students' learning style detection using tree augmented naive Bayes. R Soc Open Sci, 5(7), 172,108. https://doi.org/10.1098/rsos.172108
12. Maheswari, S., & Pitchai, R. (2019). Heart Disease Prediction System Using Decision Tree and Naive Bayes Algorithm. Curr Med Imaging Rev., 15(8), 712–717. https://doi.org/10.2174/1573405614666180322141259

13. Miranda, E., Irwansyah, E., Amelga, A. Y., Maribondang, M. M., & Salim, M. (2016). Detection of Cardiovascular Disease Risk's Level for Adults Using Naive Bayes Classifier. Healthc Inform Res, 22(3), 196–205. https://doi.org/10.4258/hir.2016.22.3.196
14. Mughal, M. O., & Kim, S. (2018). Signal Classification and Jamming Detection in Wide-Band Radios Using Naïve Bayes Classifier. IEEE Communications Letters, 22(7), 1398–1401. https://doi.org/10.1109/LCOMM.2018.2830769
15. Remu, S. R. H., Faruque, M. O., Ferdous, R., Arifeen, M. M., Sakib, S., & Reza, S. M. S. (2020). Naive Bayes based Trust Management Model for Wireless Body Area Networks Proceedings of the International Conference on Computing Advancements, Dhaka, Bangladesh. https://doi.org/10.1145/3377049.3377084
16. Shirakawa, M., Nakayama, K., Hara, T., & Nishio, S. (2015). Wikipedia-Based Semantic Similarity Measurements for Noisy Short Texts Using Extended Naive Bayes. IEEE Transactions on Emerging Topics in Computing, 3(2), 205–219. https://doi.org/10.1109/TETC.2015.2418716
17. Sinha, A. K., Singh, P., Prakash, A., Pal, D., Dube, A., & Kumar, A. (2017). Putative Drug and Vaccine Target Identification in Leishmania donovani Membrane Proteins Using Naïve Bayes Probabilistic Classifier. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14(1), 204–29 https://doi.org/10.1109/TCBB.2016.2570217
18. Valdiviezo-Diaz, P., Ortega, F., Cobos, E., & Lara-Cabrera, R. (2019). A Collaborative Filtering Approach Based on Naïve Bayes Classifier. IEEE Access, 7, 108,581–108,592. https://doi.org/10.1109/ACCESS.2019.2933048
19. van der Heide, E. M. M., Veerkamp, R. F., van Pelt, M. L., Kamphuis, C., Athanasiadis, I., & Ducro, B. J. (2019). Comparing regression, naive Bayes, and random forest methods in the prediction of individual survival to second lactation in Holstein cattle. J Dairy Sci, 102(10), 9409–9421. https://doi.org/10.3168/jds.2019-16295
20. Vaseeharan, T., & Aponso, A. (2020). Review On Sentiment Analysis of Twitter Posts About News Headlines Using Machine Learning Approaches and Naïve Bayes Classifier Proceedings of the 2020 12th International Conference on Computer and Automation Engineering, Sydney, NSW, Australia. https://doi.org/10.1145/3384613.3384650
21. Widiyaningtyas, T., Zaeni, I. A. E., & Jamilah, N. (2020). Diagnosis of fever symptoms using naive bayes algorithm Proceedings of the fifth International Conference on Sustainable Information Engineering and Technology, Malang, Indonesia. https://doi.org/10.1145/3427423.3427426
22. Yang, B., Lei, Y., & Yan, B. (2016). Distributed Multi-Human Location Algorithm Using Naive Bayes Classifier for a Binary Pyroelectric Infrared Sensor Tracking System. IEEE Sensors Journal, 16(1), 216–223. https://doi.org/10.1109/JSEN.2015.2477540
23. Yasumura, Y., Ishimaki, Y., & Yamana, H. (2019). Secure Naïve Bayes Classification Protocol over Encrypted Data Using Fully Homomorphic Encryption Proceedings of the 21st International Conference on Information Integration and

Web-based Applications & Services, Munich, Germany. https://doi.org/10.1145/3366030.3366056
24. Zhang, H., Cao, Z. X., Li, M., Li, Y. Z., & Peng, C. (2016). Novel naïve Bayes classification models for predicting the carcinogenicity of chemicals. Food Chem Toxicol, 97, 141–149. https://doi.org/10.1016/j.fct.2016.09.005
25. Zhang, N., Wu, L., Yang, J., & Guan, Y. (2018). Naive Bayes Bearing Fault Diagnosis Based on Enhanced Independence of Data. Sensors (Basel), 18(2). https://doi.org/10.3390/s18020463
26. Zhang, Y., Fan, Y., & Guo, Y. (2020). A Geomagnetic Positioning Model Based on Naive Bayes Classifier Proceedings of the fourth International Conference on Computer Science and Application Engineering, Sanya, China. https://doi.org/10.1145/3424978.3425123
27. Zhang, Z. (2016). Naïve Bayes classification in R. Ann Transl Med, 4(12), 241. https://doi.org/10.21037/atm.2016.03.38
28. Zhirui, Y., & Chunyan, L. (2020). Analysis of Sentiment Classification of Hotel Reviews Based on Multinomial Naive Bayes 2020 The 11th International Conference on E-business, Management and Economics, Beijing, China. https://doi.org/10.1145/3414752.3414796

9.8 Lab

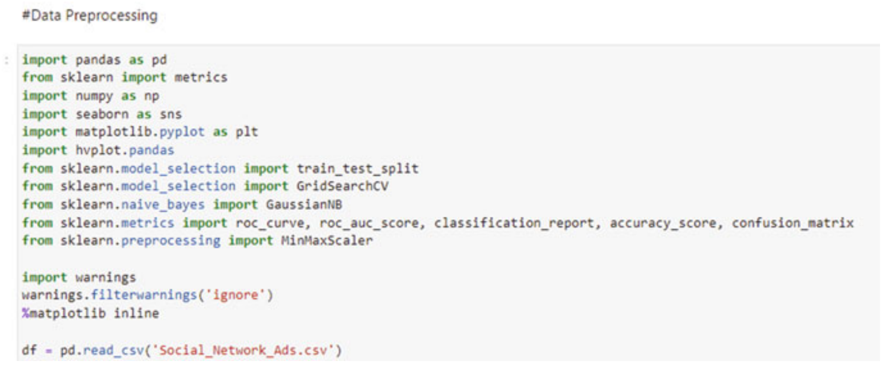
9.8.1 Working Example in Python

The social network ads dataset can be downloaded from the link below:
https://www.kaggle.com/datasets/mdwasimakhtar03/social-network-adscsv
This dataset contains the following fields:

- Age: the age of the purchaser.
- EstimatedSalary: the estimated salary of the purchaser
- Purchased: a flag of values 0 and 1 (1 for items purchased due to the ad and 0 for items not purchased)

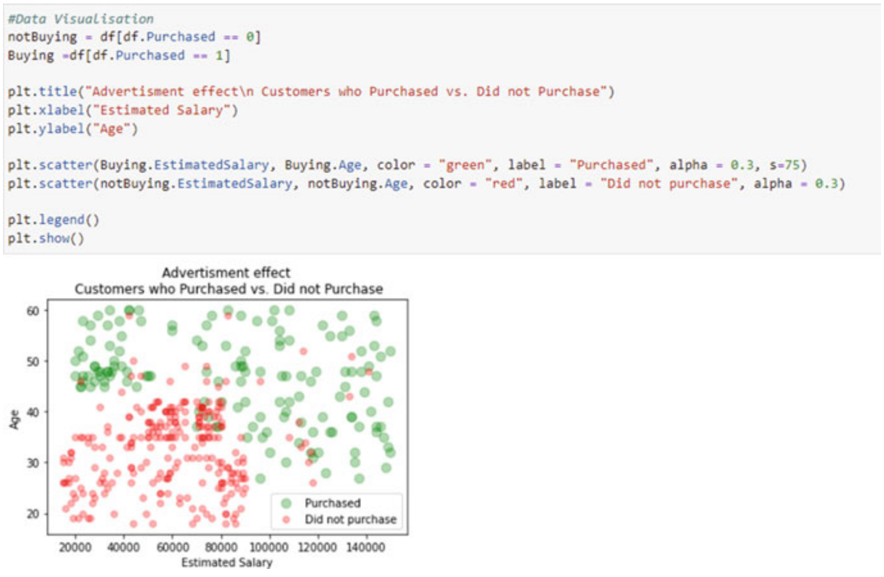
9.8.1.1 Load Social Network Ads Dataset

The first task in this lab is to load this dataset. As a reminder, the pip install command will allow you to install any missing libraries. Figure 9.1 shows how the social network ads dataset can be loaded into pandas.



```
#Data Preprocessing

import pandas as pd
from sklearn import metrics
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import roc_curve, roc_auc_score, classification_report, accuracy_score, confusion_matrix
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

df = pd.read_csv('Social_Network_Ads.csv')
```

Fig. 9.1 Load social network ads into pandas for data manipulation



```
#Data Visualisation
notBuying = df[df.Purchased == 0]
Buying =df[df.Purchased == 1]

plt.title("Advertisment effect\n Customers who Purchased vs. Did not Purchase")
plt.xlabel("Estimated Salary")
plt.ylabel("Age")

plt.scatter(Buying.EstimatedSalary, Buying.Age, color = "green", label = "Purchased", alpha = 0.3, s=75)
plt.scatter(notBuying.EstimatedSalary, notBuying.Age, color = "red", label = "Did not purchase", alpha = 0.3)

plt.legend()
plt.show()
```

Fig. 9.2 Visualizing data in scatterplot

9.8.1.2 Visualize Social Network Ads Dataset

The next step is to visualize this data. Figure 9.2 shows age vs. estimated salary based on two categories those who purchased following an advertisement and those who did not.

9.8.1.3 Choose Features and Normalize Data

The next step is to choose the model's features and target; the age and estimated salary are the features, and the purchased field are our target (Fig. 9.3). Since the actual data vary widely, we perform normalization using the min-max scaler to scale the data between 0 and 1.

9.8.1.4 Optimize GNB Model Using Hyperparameter

To produce an optimized model, we will tune the hyperparameters using the Grid search cross-validation (Fig. 9.4). var_smoothing s a hyper parameter used for calculation stability (read more here https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html). After optimizing the model is fitted to the training dataset. Then, the fitted model is used to predict probabilities on the testing dataset. The performance (e.g., ROC, AUC) curve of the optimal model on the testing dataset is then displayed (Figs. 9.4 and 9.5). The AUC score reached 90.80%, which indicates that the optimal model generalizes well on an unseen dataset.

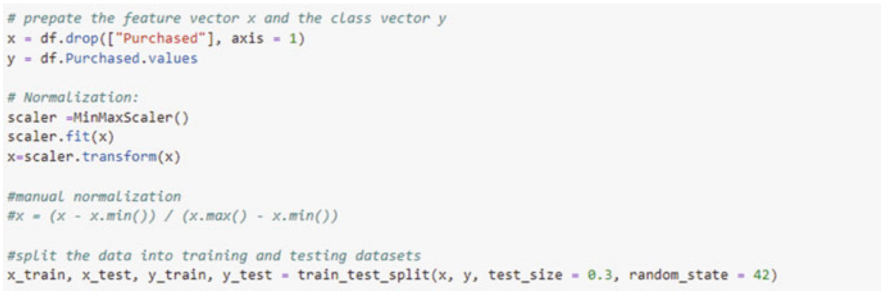
9.8.2 Working Example in Weka

Use the ionosphere dataset from Weka, or download it from the following link: https://www.kaggle.com/prashant111/ionosphere.

Open the file in Weka. The last column could have the name Class if you have used the file provided by Weka; if not, the last column that represents the outcome is named column_ai (Fig. 9.6).

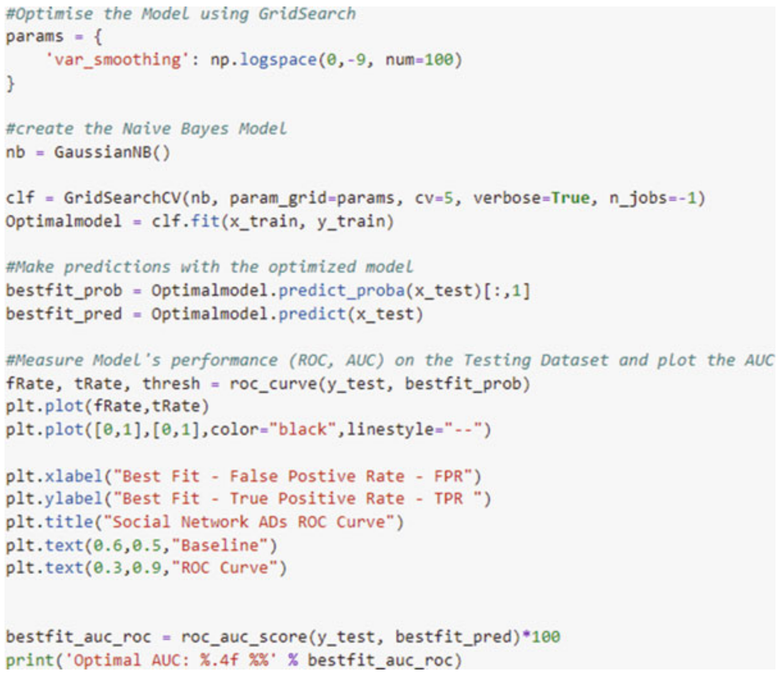
Choose the Classify tab and the naïve Bayes classifier from the Bayes set of classifiers (Fig. 9.7).

Make sure the cross-fold test option is chosen; choose the last column and click on the Start button (Fig. 9.8).



```
# prepate the feature vector x and the class vector y
x = df.drop(["Purchased"], axis = 1)
y = df.Purchased.values

# Normalization:
scaler =MinMaxScaler()
scaler.fit(x)
x=scaler.transform(x)

#manual normalization
#x = (x - x.min()) / (x.max() - x.min())

#split the data into training and testing datasets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)
```

Fig. 9.3 Choosing naïve Bayes features and applying normalization to them



```python
#Optimise the Model using GridSearch
params = {
    'var_smoothing': np.logspace(0,-9, num=100)
}

#create the Naive Bayes Model
nb = GaussianNB()

clf = GridSearchCV(nb, param_grid=params, cv=5, verbose=True, n_jobs=-1)
Optimalmodel = clf.fit(x_train, y_train)

#Make predictions with the optimized model
bestfit_prob = Optimalmodel.predict_proba(x_test)[:,1]
bestfit_pred = Optimalmodel.predict(x_test)

#Measure Model's performance (ROC, AUC) on the Testing Dataset and plot the AUC
fRate, tRate, thresh = roc_curve(y_test, bestfit_prob)
plt.plot(fRate,tRate)
plt.plot([0,1],[0,1],color="black",linestyle="--")

plt.xlabel("Best Fit - False Postive Rate - FPR")
plt.ylabel("Best Fit - True Positive Rate - TPR ")
plt.title("Social Network ADs ROC Curve")
plt.text(0.6,0.5,"Baseline")
plt.text(0.3,0.9,"ROC Curve")


bestfit_auc_roc = roc_auc_score(y_test, bestfit_pred)*100
print('Optimal AUC: %.4f %%' % bestfit_auc_roc)
```

Fig. 9.4 Applying grid search cross-validation to GNB model to find optimal model

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Optimal AUC: 90.8044 %
```

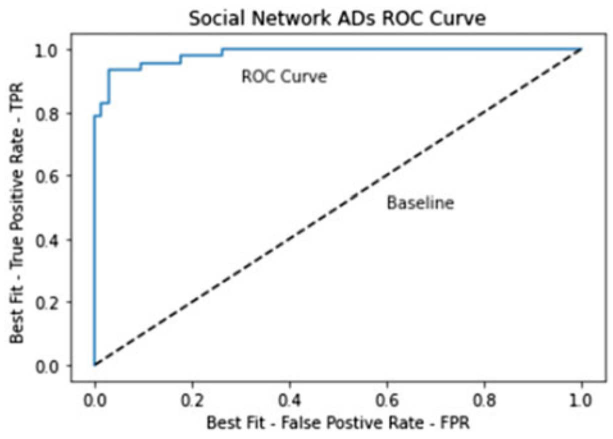
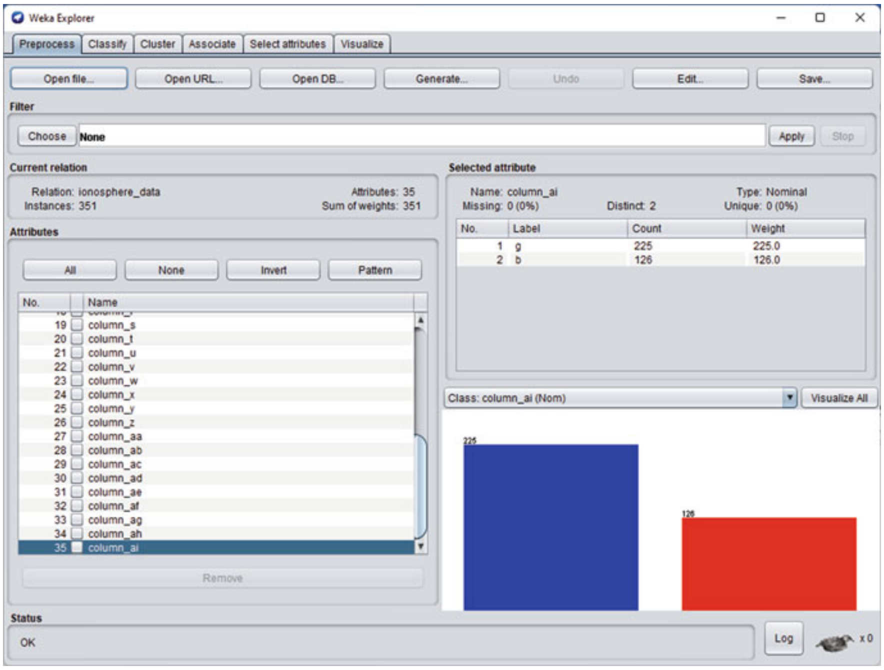


Fig. 9.5 Plotting ROC curve for the optimal GNB model

The classifier will execute and provide us with the results (Fig. 9.9). We notice that it was able to correctly classify 82% of the dataset instances (not bad for a “lazy” classifier!). The area under the curve (AUC)—under “ROC area”—is 0.935 for both



Filter: None

Relation: ionosphere_data
Instances: 351
Attributes: 35
Sum of weights: 351

Name: column_ai
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	g	225	225.0
2	b	126	126.0

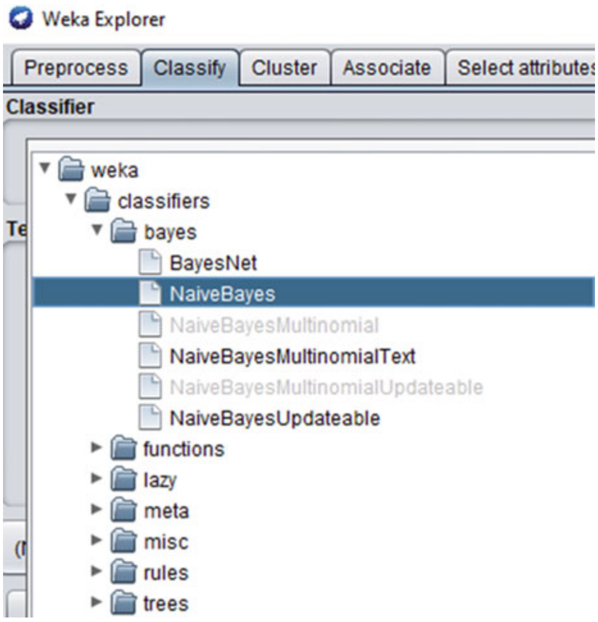
No.	Name
19	column_s
20	column_t
21	column_u
22	column_v
23	column_w
24	column_x
25	column_y
26	column_z
27	column_aa
28	column_ab
29	column_ac
30	column_ad
31	column_ae
32	column_af
33	column_ag
34	column_ah
35	column_ai

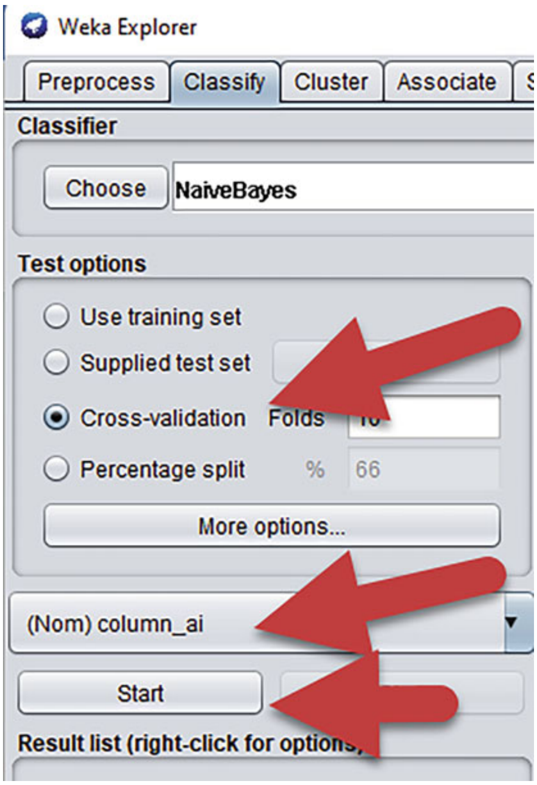
Class: column_ai (Nom)

OK

Fig. 9.6 Weka with the ionosphere dataset open in the last column

Fig. 9.7 IBK Weka classifier





NaiveBayes

- ( ) Use training set
- ( ) Supplied test set
- (●) Cross-validation Folds 10
- ( ) Percentage split % 66

(Nom) column_ai

Fig. 9.8 Execute the naïve Bayes classifier in Weka

outcome classes, which indicates excellent sensitivity and specificity. The confusion matrix shows us how many instances were assigned to the wrong class by the classifier for each class (45 for class "g" and 17 for class "b").

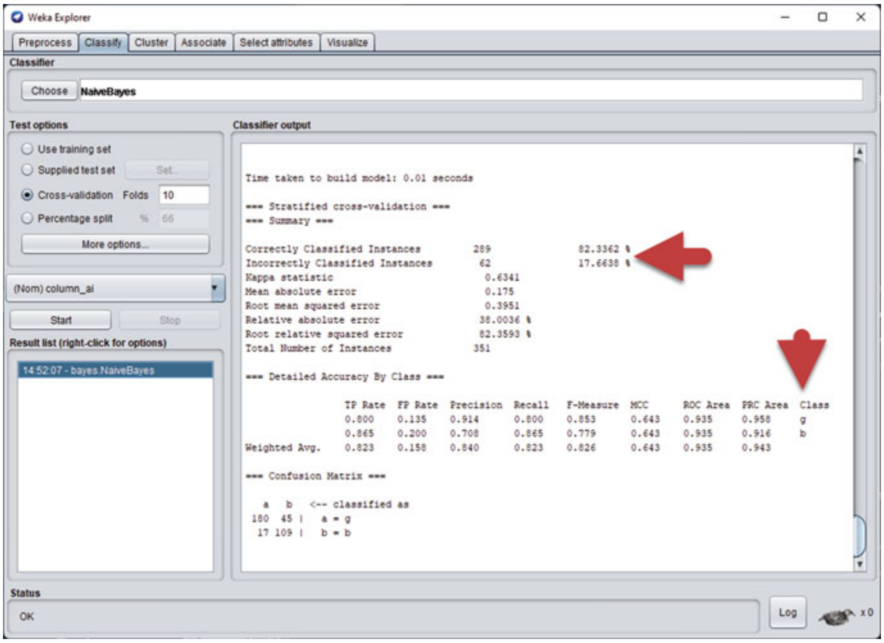
9.8.3 Do It Yourself

9.8.3.1 Building a Movie Recommender System

Download a dataset from the following site https://grouplens.org/datasets/movielens/. Choose the datafile recommended for education and development.

The zipped data file contains three files. You will need data from ratings.dat: that stores user movie ratings formatted as UserID::MovieID::Rating::Timestamp.

This problem requires a lot of time to solve and consider it as a mini project. Reading the datafile would require some effort, once that is done the rest is straightforward.



NaiveBayes

Cross-validation Folds 10

(Nom) column_ai

14:52:07 - bayes.NaiveBayes

OK

Classifier output

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         289               82.3362 %
Incorrectly Classified Instances        62               17.6638 %
Kappa statistic                          0.6341
Mean absolute error                      0.175
Root mean squared error                  0.3951
Relative absolute error                 38.0036 %
Root relative squared error             82.3593 %
Total Number of Instances              351

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.800    0.135    0.914      0.800    0.853      0.643    0.935     0.958     g
                 0.865    0.200    0.708      0.865    0.779      0.643    0.935     0.916     b
Weighted Avg.    0.823    0.158    0.840      0.823    0.826      0.643    0.935     0.943

=== Confusion Matrix ===

   a   b   <-- classified as
 180  45 |   a = g
  17 109 |   b = b
```

Fig. 9.9 The naïve Bayes classification result using Weka

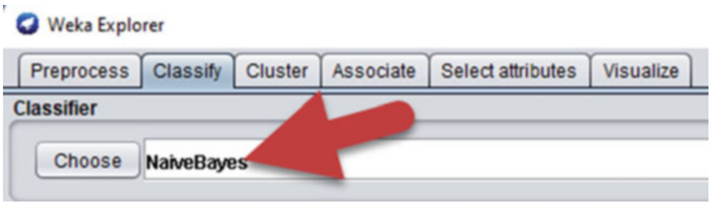


Fig. 9.10 Naïve Bayes classifier textbox

9.8.3.2 Predicting Flower Types

1. Open the Iris Dataset.
2. Click on the naïve Bayes classifier textbox to display its parameters (Fig. 9.10). Two parameters are of interest in this case: *UseKernelEstimator* and *useSupervisedDiscretization*. By default, the naïve Bayes algorithm assumes a Gaussian distribution for numeric variables; alternatively, you can choose the parameter UseKernelEstimator to be true, which will force the algorithm to use a kernel estimator instead of a Gaussian one, or choose useSupervisedDiscretization to be true to ask the algorithm to automatically discretize continuous attributes.

Do you have any continuous attributes in the ionosphere dataset? Use the above attributes and rerun the naïve Bayes classifier again and check if the results differ. Write your conclusions.

3. Use the NBC to solve the classification problem of the iris dataset.

9.8.4 Do More Yourself

Download the train and test datasets of the *Titanic* from https://www.kaggle.com/c/titanic/data?select=train.csv.

The aim is to use the *Titanic* dataset to predict who will survive and who will die.

1. Data exploration and visualization
    (a) Explore the dataset if you have not already done so.
    (b) Choose important features and visualize them according to survival/non-survival.
2. Feature engineering
    (a) Treat null values.
    (b) Encode categorical data.
    (c) Transform features if needed.
3. Classification
    (a) Test logistic regression (LR).
    (b) Test naïve Bayes.
    (c) Which one performs better?

References

1. P.D. Hoff, *A First Course in Bayesian Statistical Methods* (Springer, New York, 2009)
2. M. Gopal, *Applied Machine Learning* (McGraw-Hill Education, 2018)
3. D.D. Lewis, Naive (Bayes) at forty: The independence assumption in information retrieval, in *Machine Learning: ECML-98*, ed. by C. Nédellec, C. Rouveirol, (Springer, Berlin, Heidelberg, 1998), pp. 4–15
4. I. Kononenko, Semi-naive bayesian classifier, in *Machine Learning — EWSL-91*, ed. by Y. Kodratoff, (Springer, Berlin, Heidelberg, 1991), pp. 206–219
5. P. Golpour et al., Comparison of support vector machine, Naïve Bayes and logistic regression for assessing the necessity for coronary angiography. Int. J. Environ Res. Public Health **17**(18), 6449 (4 Sep 2020). https://doi.org/10.3390/ijerph17186449
6. D. Harada, H. Asanoi, T. Noto, J. Takagawa, Naive bayes prediction of the development of cardiac events in heart failure with preserved ejection fraction in an outpatient clinic - beyond B-type natriuretic peptide. Circ. J. **86**(1), 37–46 (30 Jul 2021). https://doi.org/10.1253/circj.CJ-21-0131

7. N.A. Mansour, A.I. Saleh, M. Badawy, H.A. Ali, Accurate detection of Covid-19 patients based on Feature Correlated Naïve Bayes (FCNB) classification strategy. J. Ambient Intell. Humaniz. Comput. **13**(1), 41–73 (15 Jan 2021). https://doi.org/10.1007/s12652-020-02883-2
8. L. Yang et al., Prediction model of the response to neoadjuvant chemotherapy in breast cancers by a naive Bayes algorithm. Comput. Methods Prog. Biomed. **192**, 105458 (Aug 2020). https://doi.org/10.1016/j.cmpb.2020.105458
9. H. Zhang et al., Developing novel computational prediction models for assessing chemical-induced neurotoxicity using naïve Bayes classifier technique. Food Chem. Toxicol. **143**, 111513 (Sep 2020). https://doi.org/10.1016/j.fct.2020.111513
10. E. Crawford, J. Kay, E. McCreath, IEMS - the intelligent email sorter, in *Presented at the Proceedings of the Nineteenth International Conference on Machine Learning* (2002)
11. R. J. Mooney, L. Roy, Content-based book recommending using learning for text categorization. Presented at the Proceedings of the fifth ACM conference on Digital libraries, San Antonio, Texas, USA, (2000). [Online]. Available: https://doi.org/10.1145/336597.336662
12. T. Trappenberg, *Fundamentals of Machine Learning* (OUP, Oxford, 2019)
13. R.R. Bouckaert, Naive Bayes classifiers that perform well with continuous variables, in *AI 2004: Advances in Artificial Intelligence*, ed. by G. I. Webb, X. Yu, (Springer, Berlin, Heidelberg, 2005), pp. 1089–1094
14. C. N. Hsu, H. J. Huang, T. T. Wong, Why discretization works for Naïve Bayesian classiers, in *17th International Conference on Machine Learning (ICML-2000)*, (2000)
15. Y. Yang, G.I. Webb, On why discretization works for naive-bayes classifiers, in *AI 2003: Advances in Artificial Intelligence*, ed. by T. D. Gedeon, L. C. C. Fung, (Springer, Berlin, Heidelberg, 2003), pp. 440–452
16. G. H. John, P. Langley, Estimating continuous distributions in BAYESIAN classifiers. Presented at the Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Montréal, Qué, Canada, (1995)
17. J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement them from Scratch* (Jason Brownlee, 2016)
18. P. Domingos, M. Pazzani, On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. Mach. Learn. **29**(2), 103–130 (1 Nov 1997). https://doi.org/10.1023/A:1007413511361

Chapter 10
K-Nearest Neighbors

10.1 The Problem

Like decision trees, k-nearest neighbors (KNN) is a non-parametric algorithm that can perform classification and regression.

KNN tries to classify a data instance based on its neighboring instances, assuming that the instance should be of the same class as the majority of its neighbors; an example would be to try to guess which party would a person vote for by looking at how most of her neighbors voted. Of course, if we have more information about the person and her neighbors, such as age, income, and education level, we will get a better prediction by looking at the neighbors with similar features. In the rest of the chapter, all references to KNN are concerned with the supervised approach.

KNN keeps the training dataset; when a new instance x is available, KNN finds the k training instances that are "closest" to x and assigns x to the majority class (i.e., the *mode*) in the case of a classification scenario, or to the *mean* (or the *median*) in the case of a regression scenario (i.e., continuous variables) [1].

Since we need to find the k instances "closest" to x, we need to define a distance function to measure "closeness." We are used to using a direct line to measure the distance between two physical objects, but that is only one type of existing distance function, called the *Euclidean distance*; there are other types that we will discover in this chapter.

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_10

10.2 A Practical Example

10.2.1 A Classification

We will start with a practical example to illustrate how the algorithm functions in the case of classification.

Consider a dataset describing the weights and heights of nine people (Table 10.1) with a classification for each as normal weight or overweight based on the body mass index (BMI). BMI is computed as weight/(height)$^2\times$10,000, and the decision is based on the following criteria: underweight if BMI$<$18.5, normal weight if BMI = 18.5–24.9, overweight if BMI = 25–29.9, and obese if BMI is 30 or greater.

The scatter graph in Fig. 10.1 shows the dataset based on the axes of weight and height; the two points on the weight axis at 167 cm and 174 cm represent the two data points in the overweight class.

We would like to classify a new data point (73 kg, 173 cm) to determine if it belongs to the normal or overweight class. We will compute the distance between

Table 10.1 A dataset of weights and heights

Weight (kg)	Height (cm)	Class
71	167	Overweight
65	182	Normal
72	176	Normal
68	173	Normal
60	172	Normal
55	169	Normal
70	173	Normal
76	174	Overweight
57	170	Normal
73	173	**?**

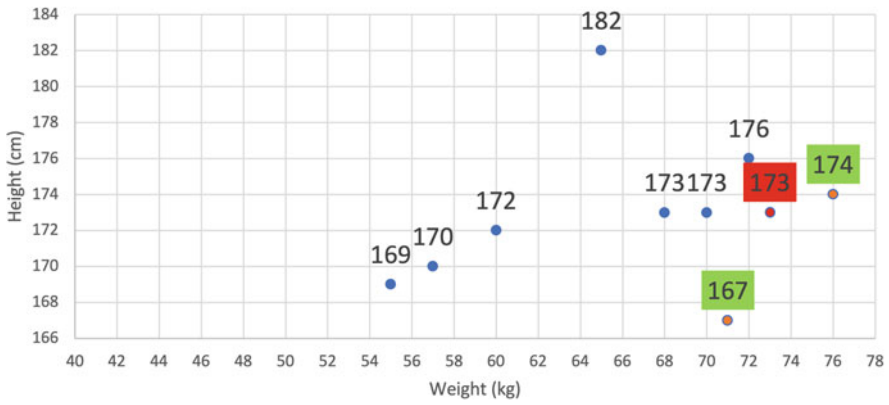


Fig. 10.1 Scatter graph for the dataset and the new instance (73 kg, 173 cm)

Table 10.2 Distances between (73 kg, 173 cm) and all available instances in the dataset

Weight (kg)	Height (cm)	Class	Distance
71	167	Overweight	6.324555
65	182	Normal	10.041595
72	176	Normal	3.162278
68	173	Normal	5.000000
60	172	Normal	13.038405
55	169	Normal	18.439089
70	173	Normal	3.000000
76	174	Overweight	3.162278
57	170	Normal	16.278821
73	173	**?**	

this new instance and the available dataset points and find the nearest neighbors; the question is how many neighbors (i.e., the parameter K). We can consider an odd number like 3 or 5 or 7 or 21; this will facilitate voting for the majority class and overcome the possibility of equal votes for the two classes. The distance between two points (x_1, y_1) and (x_2, y_2) $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$; this distance is called the Euclidean distance.

To decide on the new instance (73 kg, 173 cm), we need to compute the Euclidean distance between it and every point of the dataset. The result is displayed in Table 10.2.

Then we choose the closest three points, which are (70 kg, 173 cm), (72 kg, 176 cm), and (76 kg, 174 cm). The first two belong to the normal class and the third to the overweight class. We, therefore, classify (73 kg, 173 cm) in the majority class: normal.

10.2.2 Regression

Let us take the Sacramento real estate dataset containing 932 real estate transactions in Sacramento, California, and originally reported in the *Sacramento Bee* newspaper [2]. You can download the full dataset from the following link: https://support.spatialkey.com/spatialkey-sample-csv-data/.

We will take a subsample of 15 instances out of the 932 instances. The dataset contains the prices as well as the characteristics of houses that were sold in Sacramento. Our aim is to predict the house price (i.e., a number) of a 1050-square-foot (sq. ft.) house using nothing but the house size. Since we are trying to predict a number, our problem is one of regression and we will use the KNN algorithm for prediction. Suppose $k = 5$ and that we will use the first 15 instances of the dataset for our prediction (Table 10.3).

We will first compute the Euclidean distance between our new house of 1050 sq. ft. and the 15 instances and sort them in ascending order (Table 10.4).

Table 10.3 House sizes and prices

sq. ft.	Price
836	$ 59,222
1167	$ 68,212
796	$ 68,880
852	$ 69,307
797	$ 81,900
1122	$ 89,921
1104	$ 90,895
1177	$ 91,002
941	$ 94,905
1146	$ 98,937
909	$ 100,309
1289	$ 106,250
871	$ 106,852
1020	$ 107,502
1022	$ 108,750

Table 10.4 Distances between the new house of 1050 sq. ft. and the 15 instances in terms of house size, sorted in ascending order

sq. ft.	Price	Distance
1022	$ 108,750	28
1020	$ 107,502	30
1104	$ 90,895	54
1122	$ 89,921	72
1146	$ 98,937	96
941	$ 94,905	109
1167	$ 68,212	117
1177	$ 91,002	127
909	$ 100,309	141
871	$ 106,852	179
852	$ 69,307	198
836	$ 59,222	214
1289	$ 106,250	239
797	$ 81,900	253
796	$ 68,880	254

The five nearest neighbors to the new house are (1022 sq. ft., $108,750), (1020 sq. ft., $107,502), (1104 sq. ft., $90,895), (1122 sq. ft., $89,921), and (1146 sq. ft., $98,937). To compute the value of the 1050 sq. ft. house, we will take the average of the prices of the nearest neighbors.

KNN predicts the price of the new house as $\frac{(\$108{,}750+\$107{,}502+\$90{,}895+\$89{,}921+\$98{,}937)}{5} = \$99{,}201$.

10.3 The Algorithm

10.3.1 Distance Function

The KNN algorithm depends on the use of a distance; while we have used the Euclidean distance in the example above, many other distance functions exist to measure closeness or similarities, such as Hamming distance, Manhattan distance, Minkowski distance, Mahalanobis distance, cosine similarity, Tanimoto distance, Chebychev distance, and Jaccard distance.

10.3.1.1 Euclidean Distance

The Euclidean distance between two vectors v $(v_1, v_2 \ldots v_n)$ and w $(w_1, w_2 \ldots w_n)$ is given by the following formula:

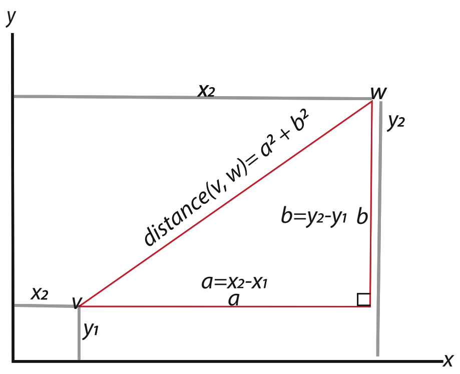
$$\text{Euclidean Distance}(v, w) = \sqrt{\sum\nolimits_{i=1}^{n} (v_i - w_i)^2}$$

In a machine learning dataset, v and w are two vectors with attributes v_i and w_i, $i = 1$ to n, where n is the number of available attributes. In the case of a two-dimensional space, the Euclidean distance between two points v (x_1, y_1) and w (x_2, y_2) is computed by the Pythagorean theorem (Fig. 10.2).

$$\text{Euclidean Distance}(v, w) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

When input variables are similar in type, Euclidean distance is usually used.

Fig. 10.2 Euclidean distance based on the Pythagorean theorem



10.3.1.2 Manhattan Distance

Manhattan distance calculates the sum of the absolute difference between two vectors v (v_1, $v_2 \ldots v_n$) and w (w_1, $w_2 \ldots w_n$).

$$\text{Manhattan Distance}(v, w) = \sqrt{\sum\nolimits_{i=1}^{n} \mid v_i - w_i \mid}$$

Manhattan distance can be computed as follows:

$$\text{Manhattan Distance}(v, w) = \sqrt{|x_2 - x_1| + |y_2 - y_1|}$$

When input variables are *not* similar in type (e.g., age, gender, weight), Manhattan distance is usually used.

10.3.1.3 Minkowski Distance

Minkowski distance is a generalized form of Euclidean and Manhattan distances; its formula is given by the following:

$$\text{Minkowski Distance}(v, w) = \sum\nolimits_{i=1}^{n} \left(|v_i - w_i|^p\right)^{1/p}$$

where p represents the order of the distance. When $p = 2$, the Minkowski distance corresponds to the Euclidean distance, and when $p = 1$, it corresponds to the Manhattan distance. When p tends towards infinity, the Minkowski distance corresponds to the Chebyshev distance.

10.3.1.4 Cosine Similarity

Cosine similarity measures the similarity between the directions of two vectors; it is equal to the cosine of the angle between the two vectors. Considering that cos (90°) = 0, cos(0°) = 1, and cos(180°) = −1, the cosine similarity can be used in the following way:

– If the vectors are pointing in the same direction, the cosine similarity is 1.
– If the vectors are pointing in the opposite direction, the cosine similarity is −1.
– If the angle between two vectors is 90°, then two vectors are orthogonal, and the cosine similarity is equal to 0.

When used as a distance metric, the cosine similarity needs to be multiplied by −
1.

$$\text{similarity} = \cos(\theta) = \frac{\sum_{i=1}^{D} v_i, w_i}{\sqrt{\sum_{i=1}^{D} {v_i}^2}\sqrt{\sum_{j=1}^{D} {w_i}^2}}$$

where D is the domain or dataset for the two vectors v (v_1, $v_2 \ldots v_n$) and w (w_1, $w_2 \ldots w_n$).

We use cosine similarity to measure distance when the size of the vectors (i.e., the term frequency) does not matter, for example, if we are measuring similarities between two documents and we just want to know if they both contain similar words but are not interested in the frequency of those words in the documents.

10.3.1.5 Hamming Distance

Hamming distance measures the similarity between two strings of the same length; it represents the number of positions at which the characters of the strings differ. If string S_1 is "Health Informati**cs**" and string S_2 is "Health Informati**on**," then the Hamming distance between S_1 and S_2 is 2; the higher the distance, the more dissimilar the two strings. A Hamming distance of 0 indicates identical strings. Hamming distance works only when the strings are of the same length.

10.3.2 KNN for Classification

In general, you will have a dataset and a corresponding set of classes, also called labels, such as readmitted/not readmitted; satisfied/not satisfied, true/false, indicating the presence or absence of an event (e.g., spam, movie enjoyable to watch), or it could be categories (e.g., movie ratings).

The algorithm is as follows:

```
BEGIN
Get a new instance $v$.
  For all points $w_i$ in the dataset
    Compute the distance between $v$ and $w_i$
    Store the distance for later comparison
  End for
  Sort the distances
  Choose the $k$ instances closest to $v$ (i.e., with lowest $k$ distances ($v$, $w_i$))
  Count the votes for each class in the $k$ instances
  Classify $v$ in the winning class (i.e., class with highest votes).
END
```

10.3.3 KNN for Regression

Regression is used to use existing information to predict future numerical values. For example, we could use the number of hours students spend studying each week to predict their future GPA; we can also use information about a house (e.g., number of rooms, size, location) to predict its sale price.

For regression problems, KNN tries to predict a value; it detects the *k* closest (i.e., most similar) instances to the new instance in question and computes the mean (or the median) of the k-nearest neighbors' outcome needed. That mean (or median) is the predicted outcome for the new instance. Hence the algorithm stays the same while the outcome is chosen as the average of the k-nearest neighbors instead of voting on a winning class.

10.4 Final Notes: Advantages, Disadvantages, and Best Practices

The distance metric and the value for *k* are hyperparameters that are chosen before running the algorithm. The distance metric could also be learned from data as opposed to guessing it, but this is beyond the scope of this introductory book [1]. It is possible to try different values for *k* (e.g., 1, 3, 5, 7, 11) and find out what works for your dataset [3]. *K* can also be chosen as the square root of the number of instances in the dataset.

When working with KNN, it will be a good idea to:

1. **Normalize your data**, as KNN performs better if the data has the same scale.
2. **Tackle missing data**: If data is missing, the algorithm cannot calculate distances between the instances. Either exclude instances with missing values or impute the missing values.
3. **Decrease data dimensionality**: KNN suffers from high dimensionality (hundreds or thousands of input variables). An increased number of dimensions leads to an increase in the average distance between points [4]. You can reduce dimensionality by performing feature selection.

10.5 Key Terms

1. K-nearest neighbors
2. KNN
3. Supervised KNN
4. Mode
5. Mean
6. Median

7. Euclidean distance
8. Hamming distance
9. Manhattan distance
10. Minkowski distance
11. Mahalanobis distance
12. Cosine similarity
13. Tanimoto distance
14. Chebychev distance
15. Jaccard distance

10.6 Test Your Understanding

1. Define the concept "term frequency" or TF.
2. Define the term "inverse document frequency" or IDF.
3. How can you use the cosine distance, TF, and IDF in measuring document similarities?
4. Mention at least one way of determining *k*.
5. Give three examples where we can use KNN for regression.
6. Give three examples where we can use KNN for classification.

10.7 Read More

1. Almanjahie, I. M., Kaid, Z., Laksaci, A., & Rachdi, M. (2021). Predicting temperature curve based on fast kNN local linear estimation of the conditional distribution function. PeerJ, 9, e11719. https://doi.org/10.7717/peerj.11719
2. Bian, Z., Vong, C. M., Wong, P. K., & Wang, S. (2020). Fuzzy KNN Method With Adaptive Nearest Neighbors. IEEE Trans Cybern, Pp. https://doi.org/10.1109/tcyb.2020.3031610
3. Cao, S., Lingao, W., Ji, R., Wang, C., Yao, L., Kai, L., Abdalla, A. N., & k., S. (2020). Clinical Decision Support System Based on KNN/Ontology Extraction Method Proceedings of the 2020 third International Conference on Signal Processing and Machine Learning, Beijing, China. https://doi.org/10.1145/3432291.3432305
4. Chen, L., Li, M., Su, W., Wu, M., Hirota, K., & Pedrycz, W. (2021). Adaptive Feature Selection-Based AdaBoost-KNN With Direct Optimization for Dynamic Emotion Recognition in Human–Robot Interaction. IEEE Transactions on Emerging Topics in Computational Intelligence, 5(2), 205–213. https://doi.org/10.1109/TETCI.2019.2909930
5. Ehsani, R., & Drabløs, F. (2020). Robust Distance Measures for kNN Classification of Cancer Data. Cancer Inform, 19, 1,176,935,120,965,542. https://doi.org/10.1177/1176935120965542

6. Gui, J., Cao, Y., Qi, H., Li, K., Ye, J., Liu, C., & Xu, X. (2021a). Fast kNN Search in Weighted Hamming Space With Multiple Tables. IEEE Trans Image Process, 30, 3985–3994. https://doi.org/10.1109/tip.2021.3066907
7. Gui, J., Cao, Y., Qi, H., Li, K., Ye, J., Liu, C., & Xu, X. (2021b). Fast kNN Search in Weighted Hamming Space With Multiple Tables. IEEE Transactions on Image Processing, 30, 3985–3994. https://doi.org/10.1109/TIP.2021.3066907
8. Hamed, A., Sobhy, A., & Nassar, H. (2021). Accurate Classification of COVID-19 Based on Incomplete Heterogeneous Data using a KNN Variant Algorithm. Arab J Sci Eng, 1–10. https://doi.org/10.1007/s13369-020-05212-z
9. Kherif, O., Benmahamed, Y., Teguar, M., Boubakeur, A., & Ghoneim, S. S. M. (2021). Accuracy Improvement of Power Transformer Faults Diagnostic Using KNN Classifier With Decision Tree Principle. IEEE Access, 9, 81,693–81,701. https://doi.org/10.1109/ACCESS.2021.3086135
10. Li, C., Liu, M., Cai, J., Yu, Y., & Wang, H. (2021). Topic Detection and Tracking Based on Windowed DBSCAN and Parallel KNN. IEEE Access, 9, 3858–3870. https://doi.org/10.1109/ACCESS.2020.3047458
11. Ma, Y., & Zhao, X. (2021). <italic>POD</italic>: A Parallel Outlier Detection Algorithm Using Weighted kNN. IEEE Access, 9, 81,765–81,777. https://doi.org/10.1109/ACCESS.2021.3085605
12. Mahfouz, M. A., Shoukry, A., & Ismail, M. A. (2021). EKNN: Ensemble classifier incorporating connectivity and density into kNN with application to cancer diagnosis. Artif Intell Med, 111, 101,985. https://doi.org/10.1016/j.artmed.2020.101985
13. Nayak, K. V., Arunalatha, J., & Venugopal, K. (2021). IR-FF-kNN: Image Retrieval Using Feature Fusion with k-Nearest Neighbour Classifier 2021 Workshop on Algorithm and Big Data, Fuzhou, China. https://doi.org/10.1145/3456389.3456405
14. Quan, Y., Fei, C., Ren, W., Wang, L., Niu, G., Zhao, J., Zhuang, J., Zhang, J., Zheng, K., Lin, P., Sun, X., Chen, Q., Ye, Z. G., & Karaki, T. (2021). Lead-Free KNN-Based Textured Ceramics for High-Frequency Ultrasonic Transducer Application. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 68(5), 1979–1987. https://doi.org/10.1109/TUFFC.2020.3039120
15. Quy, T. B., & Kim, J. M. (2021). Real-Time Leak Detection for a Gas Pipeline Using a k-NN Classifier and Hybrid AE Features. Sensors (Basel), 21(2). https://doi.org/10.3390/s21020367
16. Rashid, M., Bari, B. S., Hasan, M. J., Razman, M. A. M., Musa, R. M., Ab Nasir, A. F., & A, P. P. A. M. (2021). The classification of motor imagery response: an accuracy enhancement through the ensemble of random subspace k-NN. PeerJ Comput Sci, 7, e374. https://doi.org/10.7717/peerj-cs.374
17. Ren, J., Zhou, R., Farrow, M., Peiris, R., Alosi, T., Guenard, R., & Romero-Torres, S. (2020). Application of a kNN-based similarity method to biopharmaceutical manufacturing. Biotechnol Prog, 36(2), e2945. https://doi.org/10.1002/btpr.2945

18. Shaban, W. M., Rabie, A. H., Saleh, A. I., & Abo-Elsoud, M. A. (2020). A new COVID-19 Patients Detection Strategy (CPDS) based on hybrid feature selection and enhanced KNN classifier. Knowl Based Syst, 205, 106,270. https://doi.org/10.1016/j.knosys.2020.106270
19. Ying, S., Wang, B., Wang, L., Li, Q., Zhao, Y., Shang, J., Huang, H., Cheng, G., Yang, Z., & Geng, J. (2021). An Improved KNN-Based Efficient Log Anomaly Detection Method with Automatically Labeled Samples. ACM Trans. Knowl. Discov. Data, 15(3), Article 34. https://doi.org/10.1145/3441448
20. Yu, J., Wang, X., Chen, X., & Guo, J. (2021). Automatic Premature Ventricular Contraction Detection Using Deep Metric Learning and KNN. Biosensors (Basel), 11(3). https://doi.org/10.3390/bios11030069
21. Zhou, Y., Liu, L., Li, R., Shen, J., Li, L., Luo, Y., Zhou, S., & Xu, J. (2020). Distribution Network Electrical Topology Identification Based on Edge Computing and Improved KNN Proceedings of the 2020 fourth International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China. https://doi.org/10.1145/3443467.3443858

10.8 Lab

10.8.1 Working Example in Python

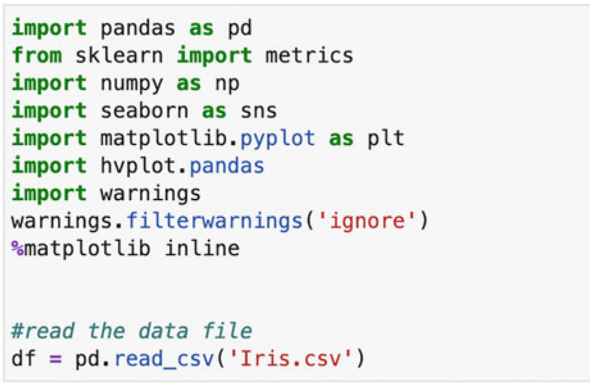
In this lab, we will use the k-nearest neighbors (KNN) algorithm to classify flowers in the Iris dataset. This dataset can be downloaded from the link below: https://www.kaggle.com/datasets/uciml/iris

It contains three species of iris as well as some properties for each flower. This dataset contains the following columns:

- Id: an internal ID for the iris dataset
- SepalLengthCm: sepal length of the iris measured in cm
- SepalWidthCm: sepal width of the iris measured in cm
- PetalLengthCm: petal length of the iris measured in cm
- PetalWidthCm: petal width of the iris measured in cm
- Species: the species of the flower (*Iris virginica*, *Iris versicolor*, and *Iris setosa*)

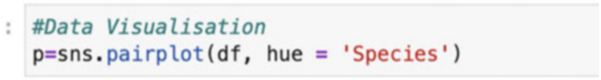
10.8.1.1 Load Iris Dataset

The first task is to import the needed libraries and load the dataset (Fig. 10.3).



```
import pandas as pd
from sklearn import metrics
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline


#read the data file
df = pd.read_csv('Iris.csv')
```

Fig. 10.3 Loading iris dataset into pandas for KNN model



```
#Data Visualisation
p=sns.pairplot(df, hue = 'Species')
```

Fig. 10.4 Data cleanup by converting species column values to numeric

10.8.1.2 Data Cleaning and Visualization

We will then display a pair plot giving a different color to each specie (Figs. 10.4, 10.5). As discussed earlier, other libraries can be used to visualize data to plot histograms, bar charts, etc.

We can notice that while the Iris Setosa can be easily identified, the Iris Virnica and Iris Versicolor data overlap and would be more challenging to classify.

10.8.1.3 Split and Scale Data

Then we will proceed to replace the values of the target variable (Species) to numeric values using the data frame's .replace() method (can you find another way to replace these values?).

Following the value replacement, we divide the dataset into feature vector x and target vector y. We proceed to split the dataset (70%, 30% ratio) between training and testing datasets (Fig. 10.6).

10.8.1.4 Optimize KNN Model Using Grid Search Cross-Validation

We then proceed to optimize the KNN model by tuning its hyperparameter using Grid search cross-validation. Finally, the best hyper parameters, the accuracy, the AUC and a classification report are computed and displayed (Fig. 10.7).

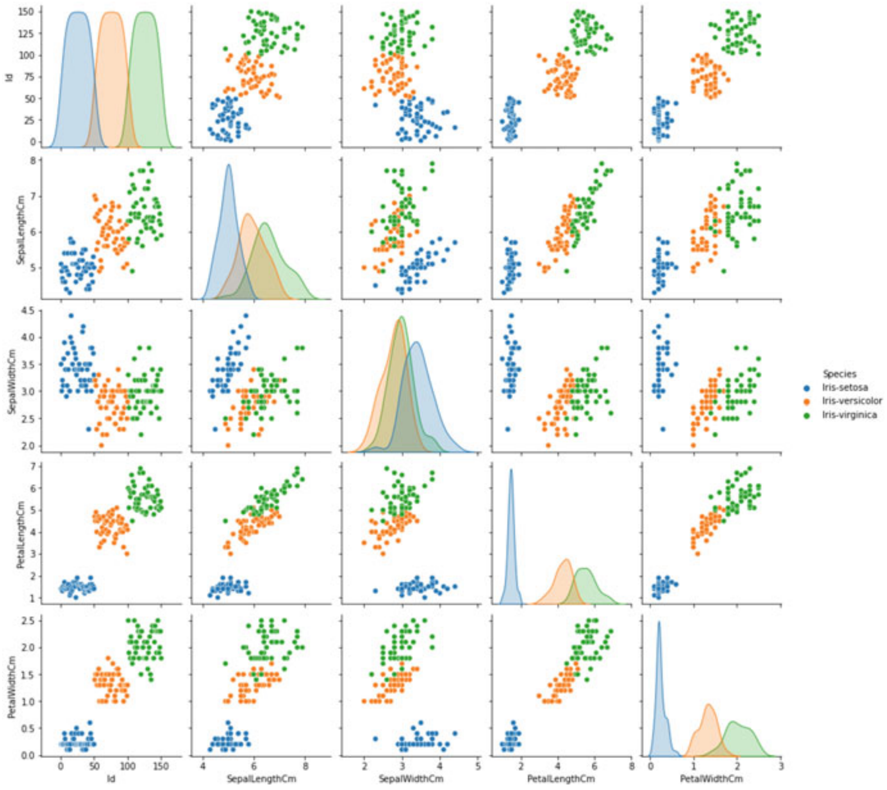
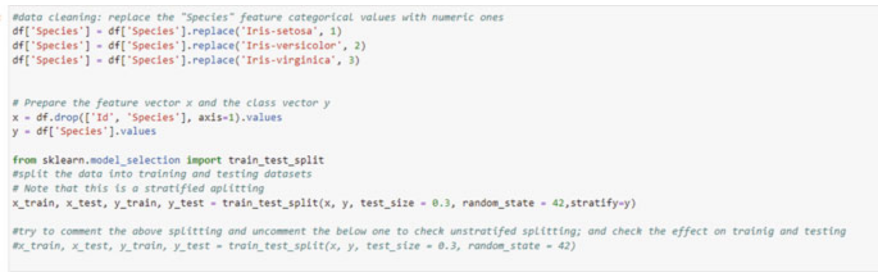


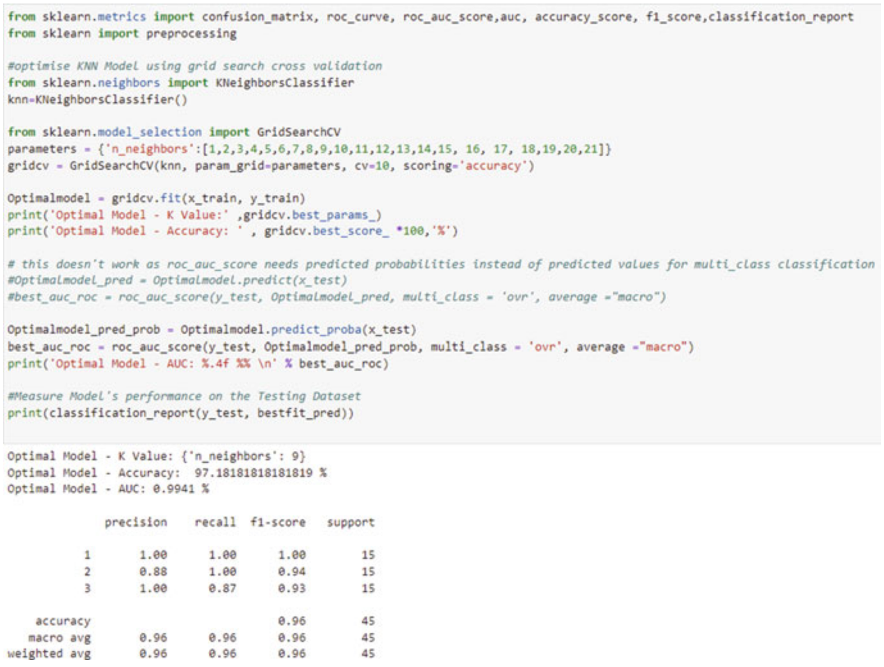
Fig. 10.5 Visualizing iris data using Pairplot



```
#data cleaning: replace the "Species" feature categorical values with numeric ones
df['Species'] = df['Species'].replace('Iris-setosa', 1)
df['Species'] = df['Species'].replace('Iris-versicolor', 2)
df['Species'] = df['Species'].replace('Iris-virginica', 3)


# Prepare the feature vector x and the class vector y
x = df.drop(['Id', 'Species'], axis=1).values
y = df['Species'].values

from sklearn.model_selection import train_test_split
#split the data into training and testing datasets
# Note that this is a stratified aplitting
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42,stratify=y)

#try to comment the above splitting and uncomment the below one to check unstratifed splitting; and check the effect on trainig and testing
#x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)
```

Fig. 10.6 Split data into training and testing datasets and scale them

Note that we are in a multi-class classification situation (i.e., 3 classes of Iris). The usual roc_auc_score will not work, and you need to use a multi_class parameter. The mult_class parameter can have a value of "ovr" or "ovo": ovr (one-vs-rest) mode



```python
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score,auc, accuracy_score, f1_score,classification_report
from sklearn import preprocessing

#optimise KNN Model using grid search cross validation
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()

from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 16, 17, 18,19,20,21]}
gridcv = GridSearchCV(knn, param_grid=parameters, cv=10, scoring='accuracy')

Optimalmodel = gridcv.fit(x_train, y_train)
print('Optimal Model - K Value:' ,gridcv.best_params_)
print('Optimal Model - Accuracy: ' , gridcv.best_score_ *100,'%')

# this doesn't work as roc_auc_score needs predicted probabilities instead of predicted values for multi_class classification
#Optimalmodel_pred = Optimalmodel.predict(x_test)
#best_auc_roc = roc_auc_score(y_test, Optimalmodel_pred, multi_class = 'ovr', average ="macro")

Optimalmodel_pred_prob = Optimalmodel.predict_proba(x_test)
best_auc_roc = roc_auc_score(y_test, Optimalmodel_pred_prob, multi_class = 'ovr', average ="macro")
print('Optimal Model - AUC: %.4f %% \n' % best_auc_roc)

#Measure Model's performance on the Testing Dataset
print(classification_report(y_test, bestfit_pred))
```

```
Optimal Model - K Value: {'n_neighbors': 9}
Optimal Model - Accuracy:  97.18181818181819 %
Optimal Model - AUC: 0.9941 %

              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       0.88      1.00      0.94        15
           3       1.00      0.87      0.93        15

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

Fig. 10.7 Optimize KNN model using grid search cross-validation

computes the AUC for each class versus the others which makes it sensitive to class imbalance, while the ovo mode computes the AUC for all possible combination of classes which makes it insensitive to class imbalance. In our case, we have the same number of instances in each target class, so the AUC will not be affected by class imbalance. Note the commented code, if you try to uncomment it and run it you will notice an error because for multi-class clasficiation roc_auc_score expects predicted probabilities (optimal_pred_prob) not predicted values (optimal_pred). Finally, we can display a confusion matrix (Fig. 10.8).

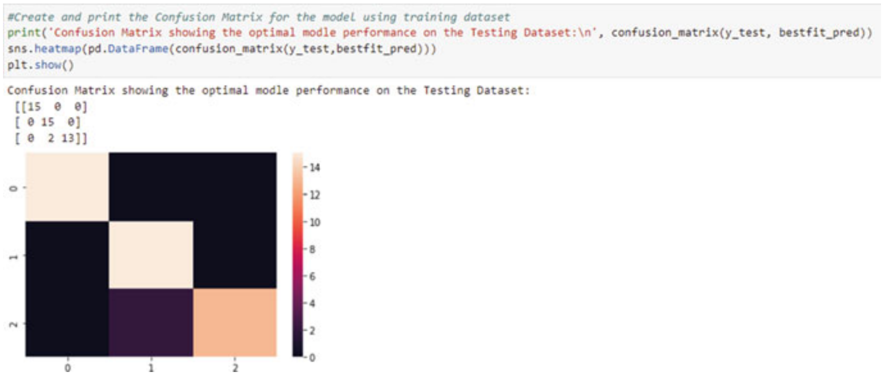
10.8.2 Working Example in Weka

Download the iris dataset using the following link: https://www.kaggle.com/arshid/iris-flower-dataset.

Open the Iris.csv file and choose the IBk classifier from the "lazy classifiers" set (Fig. 10.9).

Click on the algorithm's name to open the list of parameters associated with it (Fig. 10.10).

The KNN parameter allows you to specify the size of the neighborhood. *K* is set to 1 by default; other more common values are 3, 7, 11, and 21 [3]. However, if you



```
#Create and print the Confusion Matrix for the model using training dataset
print('Confusion Matrix showing the optimal modle performance on the Testing Dataset:\n', confusion_matrix(y_test, bestfit_pred))
sns.heatmap(pd.DataFrame(confusion_matrix(y_test,bestfit_pred)))
plt.show()
```

```
Confusion Matrix showing the optimal modle performance on the Testing Dataset:
 [[15  0  0]
 [ 0 15  0]
 [ 0  2 13]]
```

Fig. 10.8 Confusion matrix showing the performance of the optimal model on the testing dataset

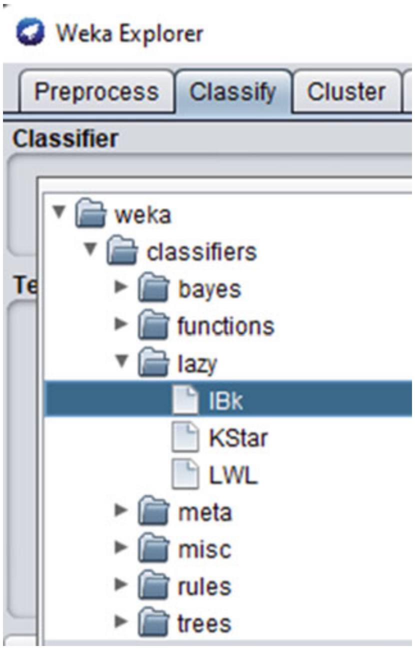
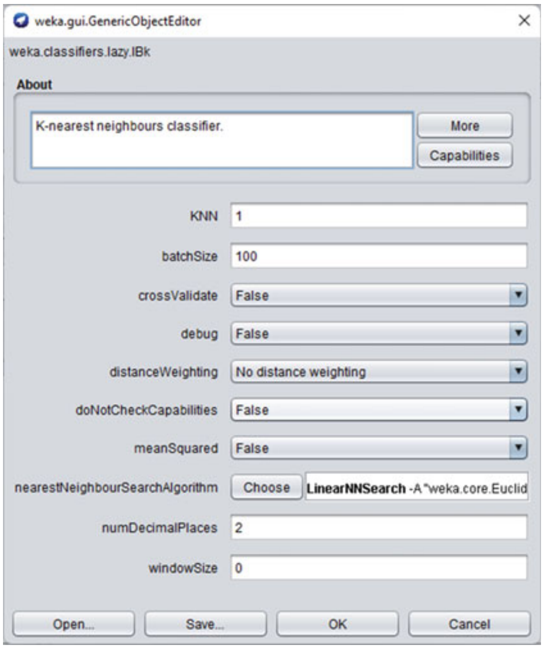


Fig. 10.9 Choosing IBk (i.e., KNN implementation) from Weka

set the *crossValidate* parameter to true, then Weka will find the most appropriate value for *k*.

The *nearestNeighbourSearchAlgorithm* controls the way the algorithm searches and stores the training dataset; it allows you to set the distance function. Click on the textbox and you will be able to change the distance parameter; the default is the Euclidean distance. Since in our case the variables are numerical, the Euclidean distance is appropriate.



weka.gui.GenericObjectEditor

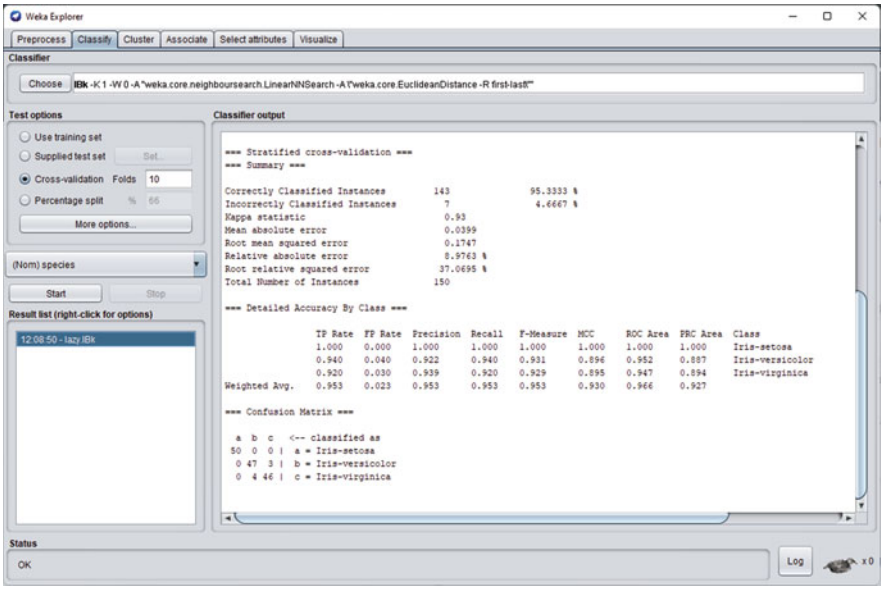
weka.classifiers.lazy.IBk

About

K-nearest neighbours classifier.

KNN	1
batchSize	100
crossValidate	False
debug	False
distanceWeighting	No distance weighting
doNotCheckCapabilities	False
meanSquared	False
nearestNeighbourSearchAlgorithm	LinearNNSearch -A "weka.core.Euclid
numDecimalPlaces	2
windowSize	0

Fig. 10.10 The parameters of KNN in Weka



Weka Explorer

Classifier

IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options

Use training set
Supplied test set
[✓ Cross-validation] Folds 10
Percentage split % 66

(Nom) species

Result list (right-click for options)

12:08:50 - lazy.IBk

Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         143               95.3333 %
Incorrectly Classified Instances         7                4.6667 %
Kappa statistic                          0.93
Mean absolute error                      0.0399
Root mean squared error                  0.1747
Relative absolute error                  8.9763 %
Root relative squared error             37.0695 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.940    0.040    0.922      0.940    0.931      0.896    0.952     0.887     Iris-versicolor
                 0.920    0.030    0.939      0.920    0.929      0.895    0.947     0.894     Iris-virginica
Weighted Avg.    0.953    0.023    0.953      0.953    0.953      0.930    0.966     0.927

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

Status

OK

Fig. 10.11 Result of executing the KNN in Weka

Run the algorithm with $k = 1$, $k = 3$, $k = 7$, and $k = 21$. Which one gives you a better result? The next figure displays the result for $k = 1$ (Fig. 10.11).

10.8.3 Do It Yourself

10.8.3.1 Iris Data Set Revisited

In the presentation above, we did not normalize or standardize the dataset. Investigate if any would enhance the KNN accuracy or AUC.

10.8.3.2 Predict the Age of Abalone from Physical Measurement

You can download the dataset from the following link: https://archive.ics.uci.edu/ml/datasets/abalone

Here is a new way to do so:

```
import pandas as pd
url = ("https://archive.ics.uci.edu/ml/machine-learning-databases/
abalone/abalone.data")
abalone = pd.read_csv(url, header=None)
```

Use KNN to predict the age of an abalone from its physical measurements.

10.8.3.3 Prostate Cancer

Download the prostate cancer dataset from the following link: https://www.kaggle.com/sajidsaifi/prostate-cancer.

Answer the following:

1. Use KNN to classify any new instance as benign (B) or malignant (M).
2. Suggest one or more algorithm for classification (You can follow the steps presented in Chap. 4).
3. Which algorithm led to a better result? Explain your answer.

10.8.4 Do More Yourself

Download the digitized breast cancer image features dataset created by Dr. William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian at the University of Wisconsin, Madison. You can use either of the following two links:

- http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
- https://www.kaggle.com/roustekbio/breast-cancer-csv

1. Get acquainted with the new dataset. Read articles published about it if necessary.
2. Can we use the dataset and KNN to predict if a new instance with unknown diagnosis derives from a benign or malignant tumor? Use KNN to answer the question.
3. Generate a new instance and use KNN to classify it as malignant or benign.

References

1. A. Burkov, *The Hundred-Page Machine Learning Book* (Andriy Burkov, 2019)
2. T.-A. Timbers, T. Campbell, M. Lee, Data Science: A first introduction, (2021). [Online]. Available: https://ubc-dsci.github.io/introduction-to-datascience/regression1.html#exploring-a-data-set-1
3. J. Brownlee, *Master Machine Learning Algorithms: Discover how they Work and Implement Them from Scratch* (Jason Brownlee, 2016)
4. J. Grus, *Data Science from Scratch: First Principles with Python* (O'Reilly Media, 2015)

Chapter 11
Neural Networks

11.1 The Problem

Rule-based systems and Bayesian networks cannot effectively solve problems such as image or speech recognition. Artificial neural networks (ANNs), or simply neural networks, are effective in solving complex problems, i.e., in modeling complex nonlinear functions. ANNs model the functioning of the brain's neurons; ANN can be trained to "learn" how to recognize patterns and classify data [1].

11.2 A Practical Example

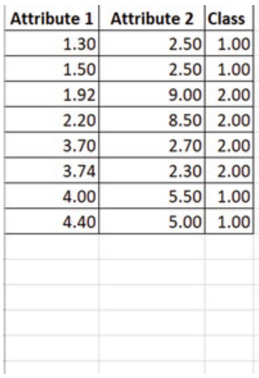
11.2.1 Example 1

Let us take an example of a dataset that has four instances with two variables, x and y, and two classes (i.e., class "grey" and class "black"), which are drawn in Fig. 11.1. We can notice two groups of instances: those in black and those in grey. But there is no way that one straight line can classify these instances into two classes/categories. If we have two lines like those present in Fig. 11.2, we can correctly classify the instances. So, the function that separates these two classes cannot be linear; we therefore have a nonlinear solution to this classification problem (Fig. 11.2). Every time a linear classification cannot work, we can make use of an artificial neural network (ANN), or more accurately, an ANN with hidden layers.

To make our point clear, we can draw two straight lines to separate the two classes (Fig. 11.3).

Each line is expressed as $y = ax + b$, or to write it slightly differently, $y - ax - b = 0$, which is equivalent to

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_11



Attribute 1	Attribute 2	Class
1.30	2.50	1.00
1.50	2.50	1.00
1.92	9.00	2.00
2.20	8.50	2.00
3.70	2.70	2.00
3.74	2.30	2.00
4.00	5.50	1.00
4.40	5.00	1.00

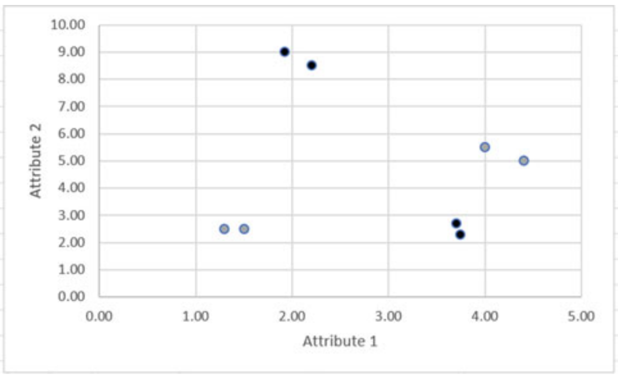


Fig. 11.1 Eight instances belonging to two classes represented by black dots and grey dots

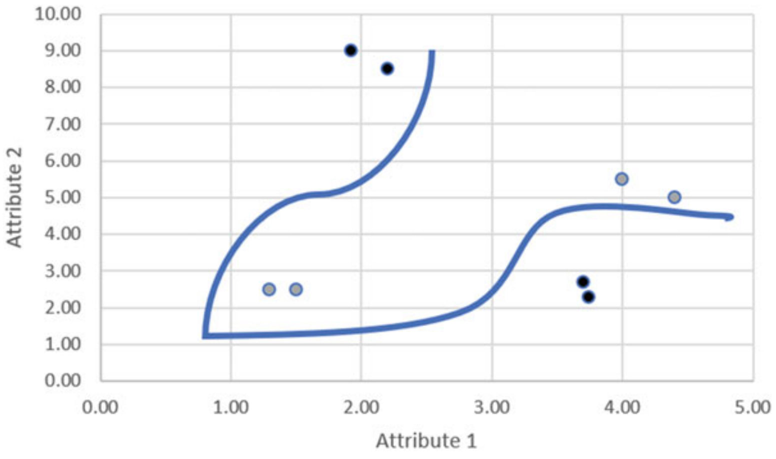


Fig. 11.2 Eight instances belonging to two classes separated by nonlinear function

$$w_2x_2 + w_1x_1 + w_0 = 0,$$

where w_2=1, w_1=−a, and w_0=−b.

We will see in the Multilayer Perceptron paragraph how an artificial neural network can solve this problem.

11.3 The Algorithm

A biological neuron can be schematized typically in the following figure (Fig. 11.4).

A brain neuron can be considered an information-processing unit. Neurons communicate through electrical signals. By discharging chemicals known as

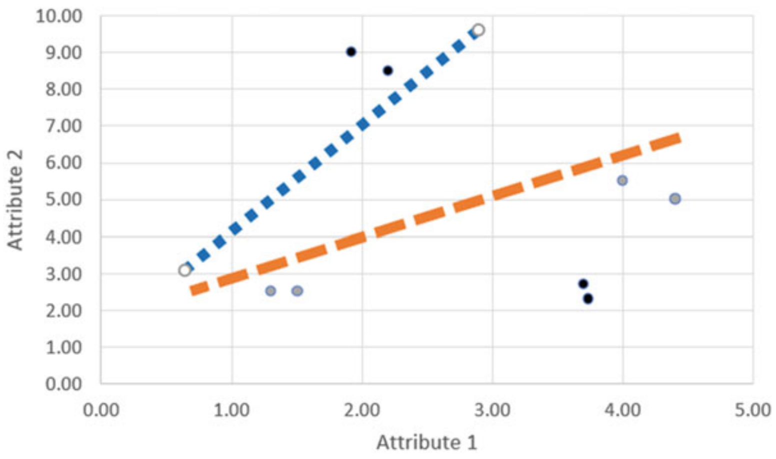


Fig. 11.3 An example of two straight lines drawn in an attempt to separate the two classes

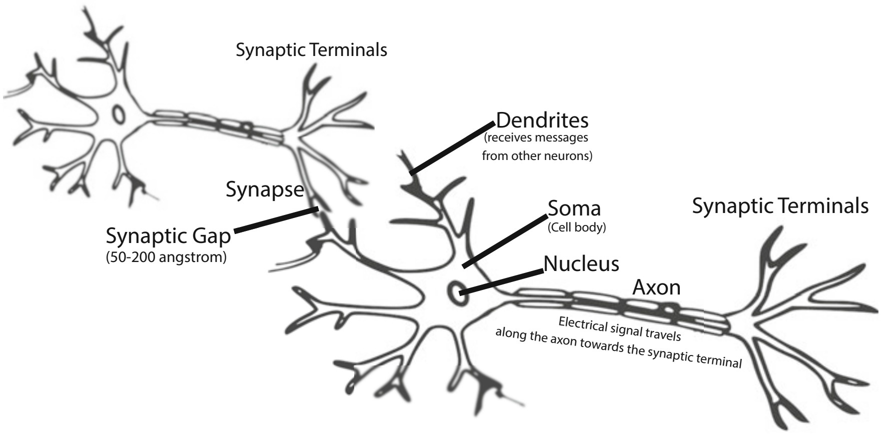

Synaptic Terminals
Dendrites
(receives messages from other neurons)
Synapse
Synaptic Gap
(50-200 angstrom)
Soma
(Cell body)
Synaptic Terminals
Nucleus
Axon
Electrical signal travels along the axon towards the synaptic terminal

Fig. 11.4 Typical biological neuron

neurotransmitters, the synaptic terminals of one neuron produce a voltage pulse which is communicated to the soma through the dendrites of another neuron. At the soma, the potentials are added, and when the summation rises above a critical threshold, then an electrical signal travels through the axon to the synaptic terminals [2, 3].

Hence, dendrites play the role of input, and the axon, the role of output. As a processing unit, the neuron has many inputs and one output that is connected to many other processing units [3].

Synapses might excite or inhibit the dendrites; exciting a dendrite results in a positive direction of its potential, while inhibiting it results in a negative direction of its potential. Hence, the inputs communicated through the dendrites are "weighted":

some signals are positive (excite), and others are negative (inhibit). At the soma, the weighted inputs are added, and if the sum crosses a threshold, the neuron fires (i.e., gives output). A neuron can fire between 0 and 1500 times per second [2]. The neuron either fires or does not, but what changes is the rate of firing.

11.3.1 The McCulloch–Pitts Neuron

In 1943, Warren McCulloch and Walter Pitts proposed a mathematical model of the neuron known today as the McCulloch–Pitts (M-P) neuron [4, 5] (Fig. 11.5). The inputs (e.g., dendrites) of an M-P neuron are either 0 or 1, and it can be thought of as formed of two parts: the first part sums up all input values, and the second makes a decision about the resulting sum. The decision function f will provide an output of 1 if the sum of the inputs is greater than or equal to a certain threshold θ (pronounced theta) and 0 otherwise.

Let us use an M-P neuron to decide whether to go to the movie theater or not. Suppose that we base our decision on four binary parameters: it is a weekday (x_1), it is after 6:00 p.m. (x_2), it is not during the COVID-19 pandemic (x_3), and the actor is Shah Rukh Khan (x_4). A decision will be made to go to watch the movie if three out of the four conditions are met ($\theta = 3$): $f(g(x)) = 1$ if $g(x) \geq \vartheta$; $f(g(x)) = 0$ *if* $g(x) < \vartheta$.

θ is called the bias; we can think of it as the prior prejudice. For example, for a certain group of people, it might be enough that two of the conditions are met to decide to go to the movie theater; for others, the threshold could be 4 or even 0.

What would be the M-P decision on a Tuesday at 5:00 p.m. during the pandemic if the actor was Shah Rukh Khan?

$$o = g(x) = x_1 + x_2 + x_3 + x_4 = 1 + 0 + 0 + 1 = 2$$

$f(\text{o}) = f(g(x)) = f(2) = 0$; the decision is not to go to the movie theater (i.e., the neuron will not fire).

What would be the M-P decision on a Tuesday at 7:00 p.m. during the pandemic if the actor was Shah Rukh Khan?

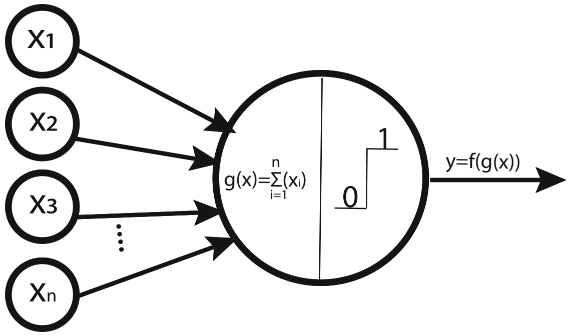


Fig. 11.5 A McCulloch–Pitts neuron

$$g(x) = x_1 + x_2 + x_3 + x_4 = 1 + 1 + 0 + 1 = 3$$

$f(g(x)) = f(3) = 1$; the decision is to go to the movie theater (i.e., the neuron will fire).

The M-P neuron was the first step towards today's neural network; however, it was very restrictive. First, not all our inputs are binary; they can be numerical or categorical. Also, the output we desire is not always binary—we might want to predict a number in the case of regression or predict a class out of multiple existing classes (more than 2) in the case of classification problems.

11.3.2 The Perceptron

To overcome these limitations, the perceptron model was proposed by Frank Rosenblatt in 1958; the model was refined by Minsky and Papertin 1969. Mainly, the perceptron proposed to add adaptive weights to the inputs (Fig. 11.6).

The neuron has many inputs ($x_0 \ldots x_n$) and adaptive weights ($w_0 \ldots w_n$); each input x_i is multiplied by a corresponding weight w_i, and then the results are summed up, mimicking the dendrites-soma-axon behavior. When the summed-up result is higher than a threshold ϑ, the outcome y is set to 1; otherwise, y is set to 0; y is in fact a function of the weighted sum $y = f(g(x)) = f\left(\sum_{i=1}^{n} (w_i \times x_i)\right)$.

If we go back to the same problem above—the decision to go to the movie theater—but we add to it the weight for each input, the weights can be decided based on knowledge about the importance of each input: a highly important input for making the right decision can be assigned a high weight, and inputs that do not play a major role can be assigned lower weights. Finding ways to determine the best weights and θ for a decision problem is the main goal in the next paragraphs.

Suppose that the perceptron is deciding for a group of Shah Rukh Khan diehard fans, hence the weight w_4 could be 10, while the other weights are set as follows: $w_1 = 2$, $w_2 = 3$, $w_3 = -5$. For the sake of this example, let us change x_3 to represent pandemic if it is equal to 1 and no pandemic if it is equal to 0.

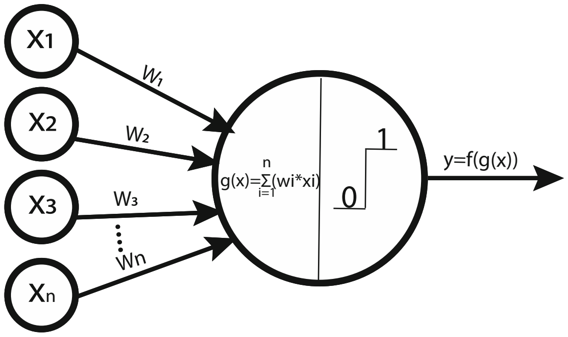


Fig. 11.6 The perceptron

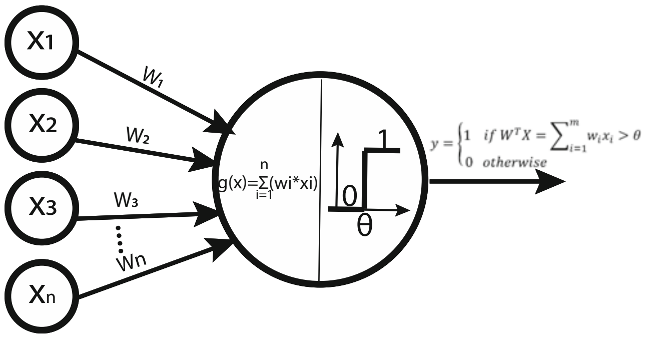

X1
X2
X3
Xn
W1
W2
W3
Wn
$g(x)=\sum_{i=1}^{n}(wi*xi)$
1
0
θ
$y = \begin{cases} 1 & if\ W^TX = \sum_{i=1}^{m} w_i x_i > \theta \\ 0 & otherwise \end{cases}$

Fig. 11.7 A perceptron with threshold ϑ

What would the perceptron's decision be on a Tuesday at 7:00 p.m. during the pandemic if the actor was Shah Rukh Khan?

$g(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 = 2x_1 + 3x_2 + -5x_3 + 10x_4 = 10 > \vartheta=3$, the decision is to go to watch the movie.

Now, we can change the weights for people who are more reasonable and decide that watching a movie with Shah Rukh Khan (or any other actor) is not of a higher value than their and others' lives; we can decide that $w_3 = -100$, which will push the perceptron's decision "do not go to theater" to always fire under a pandemic.

Mathematically, we could look at the inputs $(x_0 \ldots x_n)$ and the weights $(w_0 \ldots w_n)$ as vectors.

$$\text{The input vector } x \text{ is defined as } x = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}.$$

And the weights' vector transpose is defined as $w^T = [w_1\ w_2 \ldots w_n]$.

In mathematics, the multiplication of two vectors w^T and x is written w^Tx and is expressed as follows:

$$w^Tx = [w_1\ w_2 \ldots w_n] \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} (w_i \times x_i)$$

The function f that we have used to provide the output y is a function of the total weighted sum (i.e., $g(x)$) and is called the activation function because it allows us to activate the neuron when the value is greater than or equal to ϑ.

The activation function compares the weighted sum to ϑ and decides to activate the neuron if the weighted sum is greater than ϑ (Fig. 11.7).

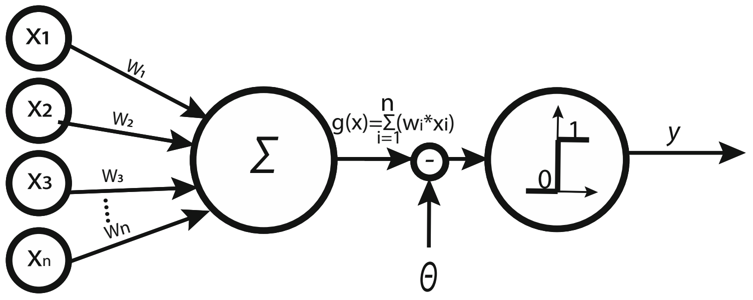


Fig. 11.8 A perceptron with the threshold ϑ subtracted from the weighted sum

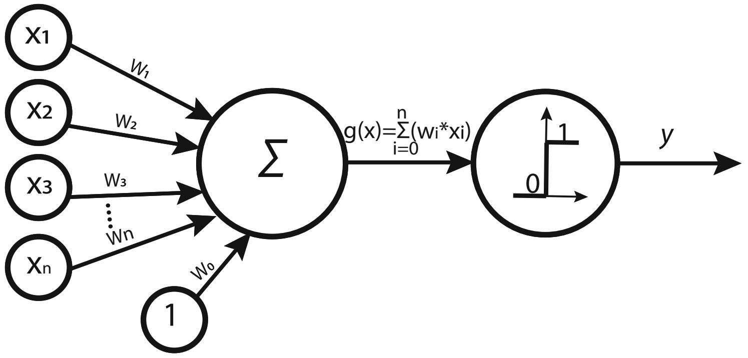


Fig. 11.9 -ϑ as an input weight W_0 for an attribute x_0 of value 1

The same result can be achieved if we subtract the value ϑ from the sum (Fig. 11.8). The result y is the same; however, the activation decision is made based on whether $\sum_{i=0}^{n} (w_i \times x_i) - \theta > 0$ or not.

We can move one step further by treating -ϑ as an extra weight called w_0 multiplied by an attribute x_0 of value 1 (Fig. 11.9).

w_0 is the bias of the model, which we sometimes represent with the letter b, which is familiar in linear functions (i.e., $y = ax + b$).

Hence the following:

$$g(x) = \sum_{i=\mathbf{1}}^{n} (w_i \times x_i) + b$$

$$y = f(g(x)) = f\left(\sum_{i=\mathbf{1}}^{n} (w_i \times x_i) + b\right)$$

can also be written

$$g(x) = \sum_{i=1}^{n} (w_i \times x_i) + (-\theta) = \sum_{i=\mathbf{1}}^{n} (w_i \times x_i) + (w_0 \times 1) = \sum_{i=\mathbf{0}}^{n} (w_i \times x_i)$$

$$y = f(g(x) = f\left(\sum_{i=\mathbf{0}}^{n} (w_i \times x_i)\right)$$

11.3.3 The Perceptron as a Linear Function

In fact, the perceptron estimates a linear function. Let us take the following example with two input variables, x_1 and x_2, and their corresponding weights, w_1 and w_2. Suppose that we have the following values for the dataset we are trying to model using the perceptron (Table 11.1). That dataset is plotted in Fig. 11.10, where the points corresponding to a zero output are in grey.

Table 11.1 A training dataset

x_1	x_2	y
1	2	1
2	3	1
1	5	1
3	0.5	0
4	1	0
5	2.3	0
0.7	4	1
0.1	4	1
0.2	5	1

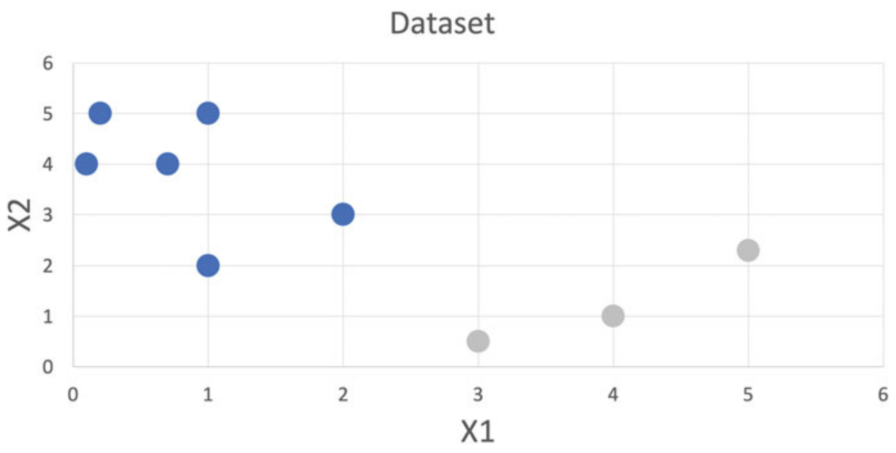


Fig. 11.10 The dataset plotted on a graph

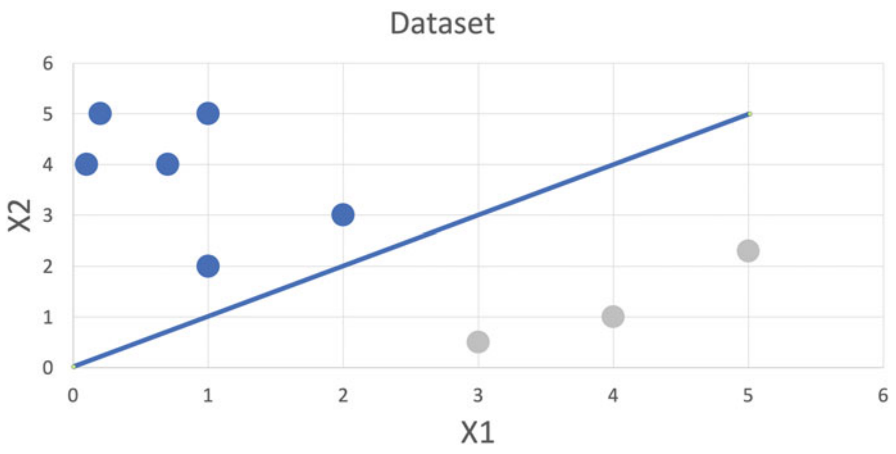


Fig. 11.11 A line ($f(x) = x$) separating the data points that belong to two different categories

It is obvious that we can find a solution to differentiate between the points in grey and the others: we can plot a straight line that separates the two sets of data points. A line such as $y = x$ will do the job (Fig. 11.11).

If we use a perceptron with weights $w_1 = 1$, $w_2 = -1$, and $w_0 = 0$, the perceptron behaves exactly like $f(x) = x$.

Let us start with $f(x) = x$; we can rewrite it as $y = x$.

We can also write it as $x_2 - x_1 = 0$, or $x_2 - x_1\_0 = 0$, or even $w_2x_2 + w_1x_1 + w_0 = 0$, where $w_0 = 0$, $w_1 = -1$, and $w_2 = 1$.

All the points on the line have in common the property $x_2 - x_1\_0 = 0$.

The points on both sides of the lines satisfy either of these two conditions: $x_2 - x_1 - 0 > 0$ or $x_2 - x_1 - 0 < 0$.

We are in a situation of a summation $\sum_{i=0}^{2} (w_i \times x_i)$ and then a decision based on comparison of the resulting sum with a threshold $\theta = 0$. We are in the domain of the perceptron. The perceptron is modeling a straight line, so it is a linear model.

11.3.4 Activation Functions

The activation function f can be different than the one mentioned above; it can, for example, propose that the output be -1 instead of 0; such a function is called bipolar as opposed to unipolar (i.e., output positive or zero). The passage from one output to another (0 to 1 or $-$ 1 to 1) was abrupt in the previous paragraph, but we can use activation functions with a smoother passage; such functions are called soft-limiting (Fig. 11.12).

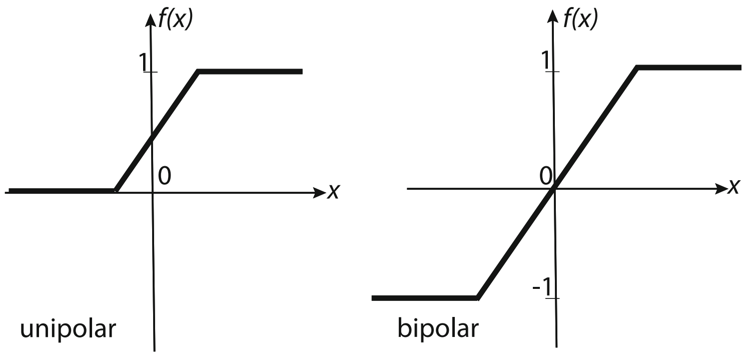


Fig. 11.12 Unipolar and bipolar activation functions

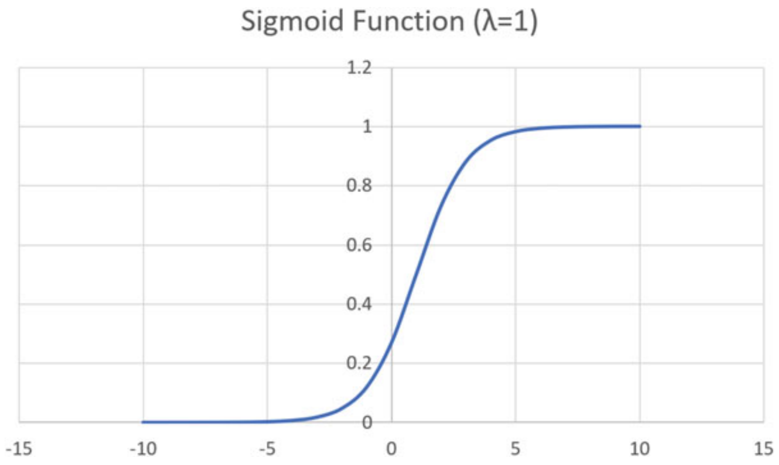


Fig. 11.13 Sigmoid function for $\lambda = 1$

However, for reasons, we will discuss below (i.e., gradient descent), we will need to compute the derivative of the activation function, i.e., it must be differentiable. We have many activation functions to choose from [6].

11.3.4.1 The Sigmoid Function

A function that satisfies the differentiability criterion and that can play the role of a soft-limiting activation function is the sigmoid function, defined as:

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

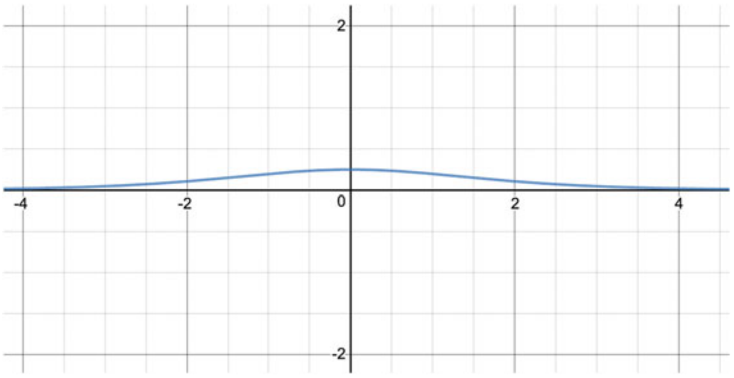


Fig. 11.14 A graph showing the gradient of the sigmoid function

The graph of the sigmoid function is given in Fig. 11.13:where λ determines the steepness of the sigmoid function. We can notice that the outcome of the sigmoid function varies between 0 and 1.

The gradient of the sigmoid is defined as follows:

$$f'(x) = \text{sigmoid}(x) \times (1 - \text{sigmoid}(x))$$

Since the sigmoid function's output is always positive, the gradient of the sigmoid will always be positive, whatever the value of x (Fig. 11.14). In fact, the gradient approaches 0 above +3 and below −3, which indicates that little learning is done above +3 or below −3.

We will overcome this issue if we scale the sigmoid function, and that is the solution proposed by the tanh function.

11.3.4.2 The Tanh Function

The hyperbolic tangent function is defined as follows:

$$f(x) = \tanh\left(\frac{1}{2}\lambda x\right)$$

$$\text{for } \lambda = 2, \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The graph of the tanh function is like that of the sigmoid, but it is scaled so that it is symmetric around zero (Fig. 11.15).

The gradient of the tanh function is defined as follows:

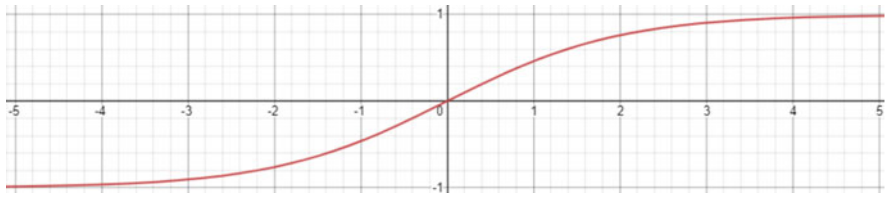


Fig. 11.15 The tanh function for λ = 2

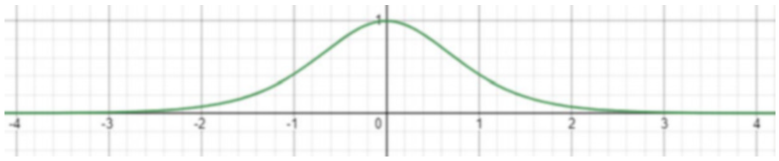


Fig. 11.16 A graph showing the gradient of the tanh function

$$f'(x) = 1 - (\tanh(x))^2$$

We can notice that the graph of the tanh gradient is symmetric around zero, and hence it can be positive or negative (Fig. 11.16)

11.3.4.3 The ReLU Function

The rectified unit function (ReLU) is defined as follows:

$$\mathrm{ReLU}(x) = f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

The ReLU graph is shown in Fig. 11.17.

For all values that are below 0, the activation function will have 0 as an output; hence, ReLU might activate a subset of all the neurons, which makes it more efficient than other activation functions. The gradient of ReLU is a constant (0 or 1) and is defined as

$$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

Since the gradient might be 0 for some neurons, during backpropagation, some weights and biases will not be updated, and the corresponding neurons might never get activated; we call such neurons "dead neurons." The leaky ReLU activation function addresses this problem.

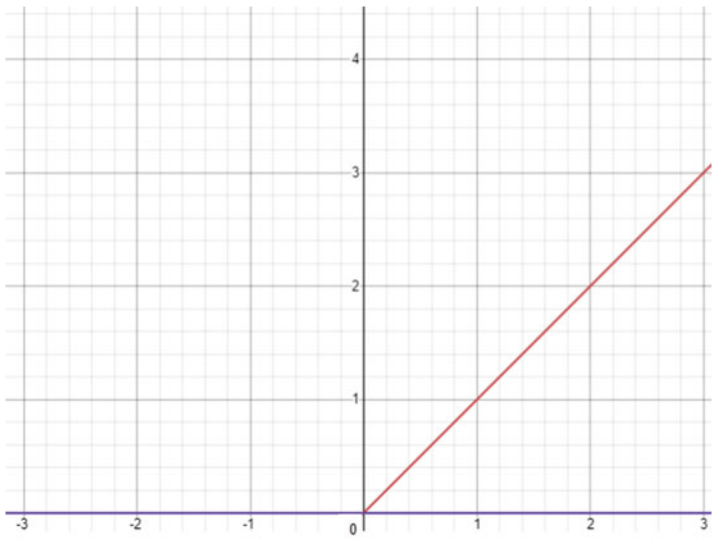


Fig. 11.17 The ReLU function

11.3.4.4 The Leaky ReLU Function

Leaky ReLU is defined as follows:

$$\mathrm{ReLU}(x) = f(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

The leaky ReLU graph is shown in Fig. 11.18
The leaky ReLU gradient function is defined as follows:

$$\mathrm{ReLU}(x) = f(x) = \begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

With leaky ReLU, the gradient of the negative inputs will never be 0; hence, there will be no dead neurons.

11.3.4.5 The Parameterized ReLU Function

The parameterized ReLU function adds flexibility for the negative values of x as it introduces the slope as a parameter (instead of the constant slope 0.01). The function is defined as follows:

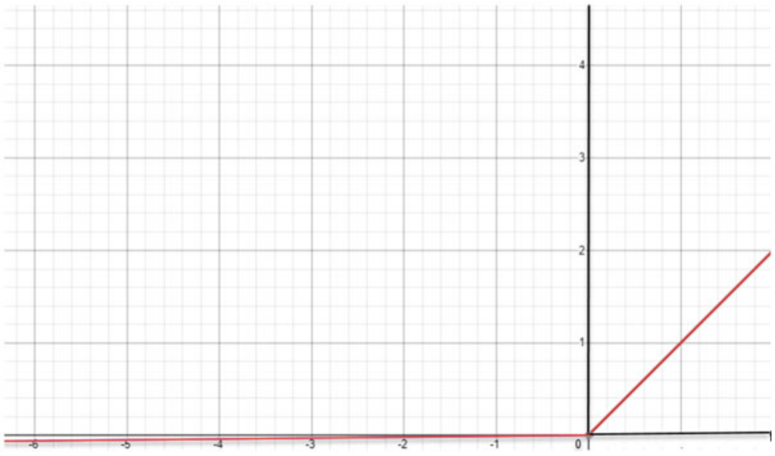


Fig. 11.18 The leaky ReLU function

$$\mathrm{ReLU}(x) = f(x) = \begin{cases} ax & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

The only caveat is that the artificial neural network will also learn the slope a for an optimal convergence.

The parameterized ReLU gradient function is defined as follows:

$$\mathrm{ReLU}(x) = f(x) = \begin{cases} a & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

11.3.4.6 The Swish Function

The swish activation function shows better performance than ReLU and is very efficient; it is defined as follows (Fig. 11.19):

$$f(x) = \frac{x}{1 + e^{-x}}$$

11.3.4.7 The SoftMax Function

The SoftMax function turns a *vector* x of k real values x_j, $j = 1$ to k, into a vector of k real values that sum to 1; it is defined as:

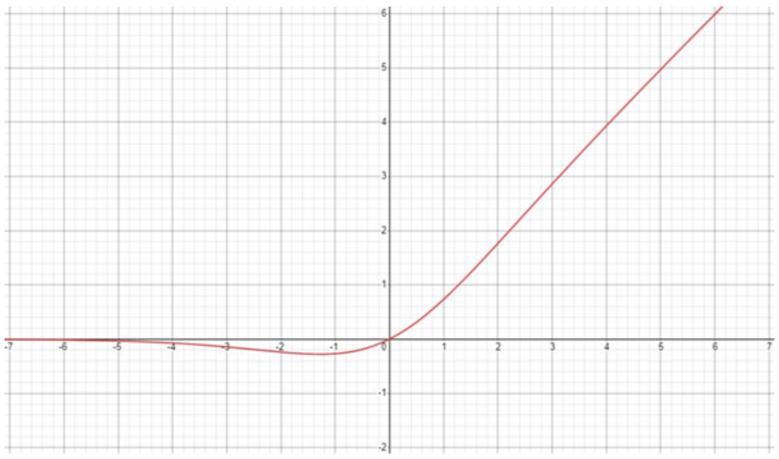


Fig. 11.19 The swish Function

$$\sigma(x_i) = \frac{e^{-x_i}}{\sum_{j=1}^{K} e^{-x_j}}$$

Since the SoftMax function returns values between 0 and 1, we can treat these values as probabilities that an input belongs to a particular class. The SoftMax activation function is very useful for multiclass classification, where the ANN has multiple neurons as output.

11.3.4.8 Which Activation Function to Choose?

There is no formula; however, the following are rules of thumb:

– Sigmoid functions work well in classification problems.
– Sigmoid and tanh functions have one notable drawback: the vanishing gradient.
– The ReLU function is generic and is widely used.
– In the case of dead neurons, use leaky ReLU.
– Use ReLU first; if it does not provide you with a good solution, then you can try other activation functions.
– Use SoftMax for multiclass classification problems.

11.3.5 Training the Perceptron

The question is how to find the right weights for the perceptron. We will do that by gradient descent, which we have seen during regression.

Let us define an error function E for the perceptron. If we take the error function as the mean squared error (MSE), and suppose that we have a training set of N instances (x^i, y^i), then E can be formulated as:

$$E = \frac{1}{2N} \sum\nolimits_{i=1}^{N} \left(y^i - \widehat{y}^i\right)^2 = \frac{1}{2N} \sum\nolimits_{i=1}^{N} \left(y^i - f\left(w^T x^i + w_0\right)\right)^2$$

- The value $\frac{1}{2}$ is chosen for convenience in later calculations (i.e., derivatives).
- The training dataset is formed of N instances $\{(x^1, y^1), (x^2, y^2)\ldots(x^N, y^N)\}$. Each x^i is an input vector with n attributes/features $(x^i{}_1, x^i{}_2, \ldots x^i{}_n)$, and each y^i is an expected output for vector x^i.
- w^T is the transpose of the weight vector $(w_1, w_2, \ldots w_n)$, where w_0 is the bias.
- $\widehat{y}^i$ is the thresholded output computed by the perceptron for the input vector x^i.

Our aim is to find the set of weights that minimizes E.

The error value depends on the values of w_0, which is the bias b, and on all other weights represented by the vector w, so E is a function of both variables. To obtain $E(w, b)$, we replace $f(w^T x^i + b)$ with $w^T x^i + b$. The error function is then expressed as follows:

$$E(w, b) = \frac{1}{2N} \sum\nolimits_{i=1}^{N} \left(y^i - \left(w^T x^i + b\right)\right)^2$$

When $w^T x^i + b = y^i$ for all x^i, $i = 1$ to N, then $E = 0$; our aim is to find a set of weights that makes E as close as possible to 0. When the perceptron learns how to fit the unthresholded outputs $w^T x^i + b$ to the desired outputs y^i, it is simple to take the same weights, apply them to the input vectors x^i, and then use a threshold function f to obtain perceptron outputs $\widehat{y}^i$ that correctly classify the x^i. For example, suppose the vectors x^i are in two classes, $y^i = 1$ and $y^i = 0$; then if $(w^T x^i + b)$ correctly classifies a vector x^i into one of the two classes (e.g., 1), that means that $w^T x^i + b$ is equal to either 1 or 0; if we apply an activation function $f(w^T x^i + w_0)$ that produces 1 if $w^T x^i + b = 1$ and produces 0 if $w^T x^i + b = 0$, the perceptron's output $f(w^T x^i + w_0)$ will correctly classify x^i.

So, we will be interested in minimizing the error function for the output $\widehat{y}^i$:

$$E(w, b) = \frac{1}{2N} \sum\nolimits_{i=1}^{N} \left(y^i - \left(w^T x^i + b\right)\right)^2$$

As was the case with the regression, we will use gradient descent to minimize the error function until convergence is reached.

$$\frac{\partial E}{\partial w_j} = \frac{\partial\left(\frac{1}{2N}\sum_{i=1}^{N}\left(y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)\right)^2\right)}{\partial w_j}$$

The error e_i made for the ith sample can be expressed as follows:

$$e^i = y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)$$

Hence

$$\frac{\partial E}{\partial w_j} = \frac{\partial\left(\frac{1}{2N}\sum_{i=1}^{N}(e^i)^2\right)}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{2N}\,\frac{\partial\left(\sum_{i=1}^{N}(e^i)^2\right)}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{2N}\sum_{i=1}^{N}\frac{\partial(e^i)^2}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{2N}\sum_{i=1}^{N}2e^i\,\frac{\partial(e^i)}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}e^i\,\frac{\partial(e^i)}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}e^i\,\frac{\partial\left(y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)\right)}{\partial w_j}$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}\left(e^i\left(-x_j^i\right)\right)$$

$$\frac{\partial E}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}e^i x_j^i = -\frac{1}{N}\sum_{i=1}^{N}\left(\left(\left(y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)\right)x_j^i\right)\right.$$

Now, we will compute $\frac{\partial E}{\partial b}$, which is $\frac{\partial E}{\partial w_0}$, where w_0 is the weight for $x^i{}_0$ and $x_0^i = 1$.

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial w_0} = -\frac{1}{N}\sum\nolimits_{i=1}^{N}\left(\left(\left(y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)\right)x_0^i\right) = \right.$$
$$-\frac{1}{N}\sum\nolimits_{i=1}^{N}\left(y^i - \sum_{j=1}^{n}\left(w_j x_j^i + w_0\right)\right)$$

We can start at the first iteration at a random value for the weights *w* and the bias *b*; then, we adjust these values at the next iteration based on the following formula, where *k* is the current iteration:

$$w_{(k+1)} = w_{(k)} - \alpha\frac{\partial E}{\partial w_{(k)}}$$

$$b_{(k+1)} = b_{(k)} - \alpha\frac{\partial E}{\partial b_{(k)}}$$

where α is the learning rate (e.g., 0.01) and $\frac{\partial E}{\partial w_{(k)}}$ and $\frac{\partial E}{\partial b_{(k)}}$ are the gradient of *E* with respect to *w* and *b*, at iteration *k*, respectively.

The updates of the perceptron parameters *w* and *b* are calculated as the difference (represented by delta Δ) between their values in the next iteration *k* and in the current one [7]:

$$w_{(k+1)} = w_{(k)} - \alpha\frac{\partial E}{\partial w_{(k)}}$$

$$w_{(k+1)} = w_{(k)} + \alpha\frac{1}{N}\sum\nolimits_{i=1}^{N} e^i x_j^i$$

$$\boldsymbol{w_{(k+1)} = w_{(k)} + \frac{\alpha}{N}\sum\nolimits_{i=1}^{N}\left(\left(y^i - \sum_{j=1}^{n}\left(w_{j(k)} x_j^i + w_{0(k)}\right)\right)x_j^i\right)}$$

$$b_{(k+1)} = b_{(k)} - \alpha\frac{\partial E}{\partial b_{(k)}}$$

$$b_{(k+1)} = b_{(k)} + \alpha\frac{1}{N}\sum\nolimits_{i=1}^{N} e^i$$

$$\boldsymbol{b_{(k+1)} = b_{(k)} + \frac{\alpha}{N}\sum\nolimits_{i=1}^{N}\left(y^i - \sum_{j=1}^{n}\left(w_{j(k)} x_j^i + w_{0(k)}\right)\right)},$$

which is equivalent to writing $w_{0(k+1)} = w_{0(k)} + \alpha\frac{1}{N}\sum_{i=1}^{N}\left(y^i - \sum_{j=1}^{n}\left(w_{j(k)} x_j^i + w_{0(k)}\right)\right)$.

To train the perceptron, we can proceed by:

1. Forward calculation: Calculating $w^T x_i + b$ for all x_i. Such a run into the N instances of the training set is called an epoch.
2. Updating the weights and the bias $w_{(k+1)}$ and $w_{0(k+1)}$.
3. Repeating steps 1 and 2 until $E(w, b)$ converges.
4. Using the last calculated weights and bias to predict the output y for any new input x.

As we can guess, the perceptron is a linear model and cannot solve a nonlinear problem.

In practice, using all the available instances to make a single update of the weights might be extremely slow, so instead, we sample a random smaller batch of the training dataset to compute every update. This method is called the minibatch **stochastic gradient descent**.

11.3.6 Perceptron Limitations: XOR Modeling

The exclusive-or (XOR) function is a function with two input variables, x_1 and x_2, that has an output of 1 if either x_1 or x_2 is 1; otherwise, it is 0. The truth table is shown in Table 11.2, and the corresponding plot is in Fig. 11.20.

To model the XOR function, we need to find a line that separates outputs 1 (black dots in Fig. 11.20) from outputs 0 (grey dots in Fig. 11.20). However, we cannot find a linear function that separates those outputs; we can see an example failing to represent XOR in Fig. 11.20.

The perceptron is a linear classifier and hence cannot find a model to classify correctly an XOR function. We can, however, extend the perceptron by adding more layers so that it becomes a multilayer perceptron (MLP), which will enable it to model nonlinear complex functions and virtually any function.

11.3.7 Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) has one or more hidden layers. Figure 11.21 shows an example of two hidden layers, one input layer and one output layer. It is important to note that the perceptron principles function on the hidden layer and the output

Table 11.2 XOR function truth table

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

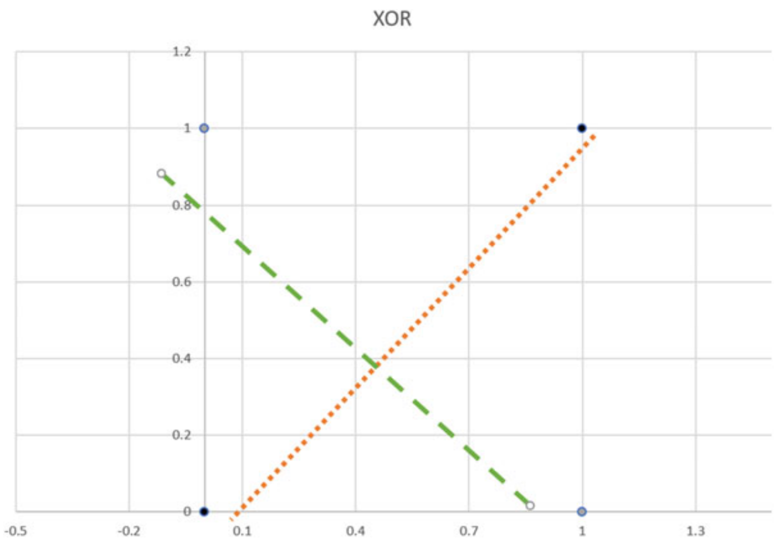


Fig. 11.20 The XOR function with two discriminant lines

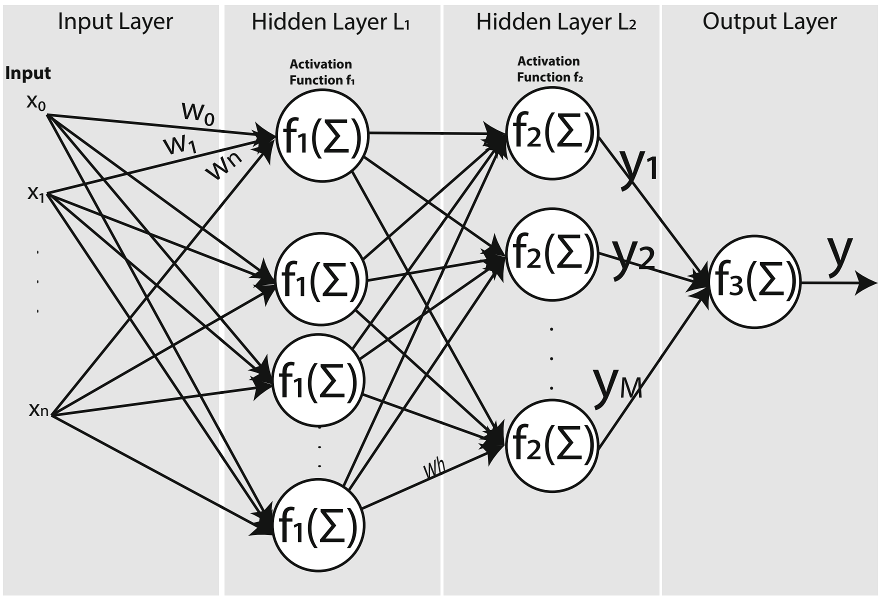


Fig. 11.21 An MLP with two hidden layers and one output

layer, where the hidden layer is formed of the data instances from the training dataset but no perceptron is involved in it; that is why many authors do not count the input layer as part of the total number of layers; however, some authors do.

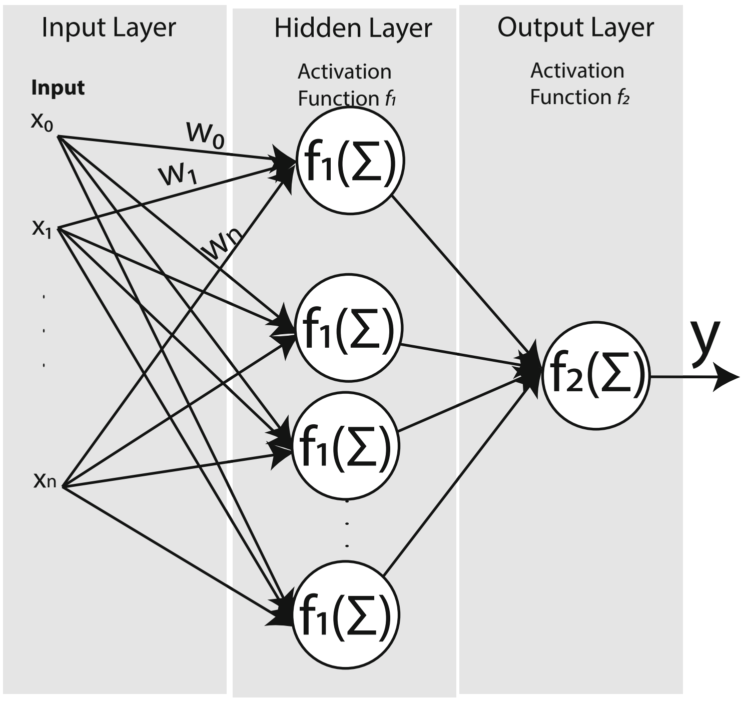


Fig. 11.22 An MLP with one hidden layer and one output y

The inputs in Fig. 11.21 and their corresponding weights are fed into the first hidden layer, composed of several neurons, which in turn generate their own outputs using an activation function f_1 and feed them with their own weights to the second hidden layer, which in turn applies an activation function f_2 and feeds its outputs to the output layer; the latter applies an activation function f_3 and generates the final MLP output.

The activation functions in each layer (i.e., f_1, f_2, and f_3) can be the same or different. If the activation function of the output layer is a linear function, the MLP generates a regression model, while if it is nonlinear, then the model is nonlinear (i.e., if the function is logistic, then the model is logistic regression or binary classification) [8]. Figure 11.22 shows an MLP with one hidden layer and one output, while Fig. 11.23 shows an MLP with one hidden layer and three outputs.

Within an MLP, we have different weights and different biases (i.e., constant b) for each layer; hence, expressing the learning problem becomes more elaborate, but it follows the same principle as in the case of one perceptron.

What do hidden layers do exactly? We will continue the practical example to answer this question.

We have seen in Fig. 11.3 that we need two lines to separate the two given classes. We know that a perceptron models a linear function; needing n lines to separate two classes is equivalent to say that we need n perceptrons. In our example,

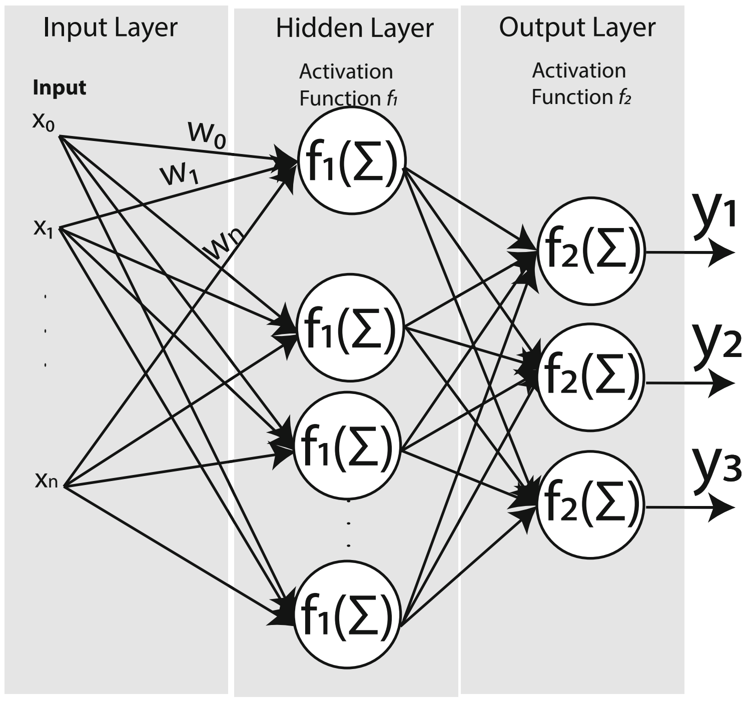


Fig. 11.23 An MLP with one hidden layer and three outputs y_1, y_2, and y_3

we will need two perceptrons in one hidden layer, where each hidden perceptron (i.e., neuron) produces one line. Since we need to join the two lines in order to have one model that separates the two classes, then we will need to join the two neurons' outputs into one neuron (the output neuron). The result is shown in Fig. 11.24.

This example is to illustrate the benefit of hidden neurons; in complex real-life problems, we cannot just guess the number of hidden neurons and the number of hidden layers required to create the model.

11.3.8 MLP Algorithm Overview

The MLP follows the algorithm below:

Initialize the input layer
Initialize the weights' vectors and bias vectors for all layers
PHASE 1: forward computation

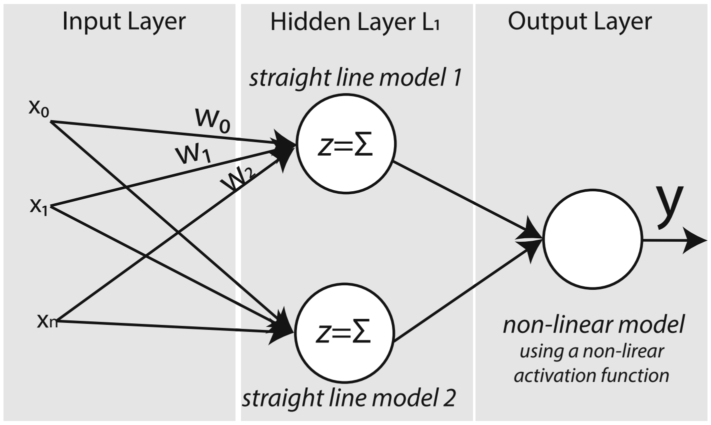


Fig. 11.24 A multilayer perceptron with a two-neuron hidden layer to model a nonlinear classifier solving the problem in Fig. 11.3

```
For each layer l from layer 2 to the output layer L (layer 1 being the MLP inputs)
  For each neuron i in layer l
      Compute the sum based on layer l's weights, bias, and the previous layer outputs
          $z_i^{(l)} = \sum_{j=1}^{N^{(l-1)}} w_{ij}^{(l)} \times a_j^{(l-1)} + b^{(l)}$
      Compute the output based on the previous sum
          $a_i^{(l)} = f^{(l)}\left(z_i^{(l)}\right)$
  End For
End For
```

Learning the model entails iteratively calculating the gradient for a cost function such as the mean squared error (MSE) until the minimum is found (the algorithm converges).

The example here is for MSE.

$$\text{Use } E(X) = \frac{1}{2N}\sum_{i=1}^{N}(\widehat{y}_i - y_i)^2$$

Starting with the last layer and working backward, compute for every neuron

$$\frac{\partial E(X)}{\partial w_{ji}^l} \text{ and } \frac{\partial E(X)}{\partial b_i^l}$$

PHASE 2: Backward propagation

We start from the output layer and update the weights and biases of each layer backward: layer L first, then L-1, and we continue until the weights of layer 0 are updated.

Weights' update:

$$w_{i+1}^l = w_i - \alpha \frac{\partial E(X)}{\partial w_{ji}^l}$$

$$b_{i+1}^l = b_i - \alpha \frac{\partial E(X)}{\partial b_i^l}$$

Then we repeat the two phases until convergence, i.e., the cost is less than a certain threshold.

11.3.9 Backpropagation

The problem we are facing is to find a method to minimize the error the MLP can produce by minimizing the error function that estimates the difference between the final output of the MLP and the expected outcome.

Backpropagation is a technique that allows us to achieve such aim; it performs a gradient descent by working backward from the output layer to the input layer, calculating in each layer the gradient of the error function with respect to the neural network's weights. The gradients of the last layer of weights are computed first and then used in computation of the gradient for the previous layer; the process continues until we reach the first layer of weights [9].

The mathematical notation is complex if we want to take a fully connected neural network, so we will start with an example and move towards the fully connected situation.

We will use the following denotations:

– E denotes our error (i.e., cost) function
– L denotes the number of layers
– N_l denotes the number of neurons in layer l
– $w_{ij}^{(l)}$ denotes the weight for neuron i in layer l in relation to the incoming neuron j in layer l-1
– $b_i^{(l)}$ denotes the bias for neuron i in layer l
– $z_i^{(l)}$ denotes the product sum plus bias for neuron i in layer l: $z_i^{(l)} =$

$\sum_{j=1}^{N^{(L-1)}} w_{ij} a_j^{(l-1)} + b_j^{(l-1)}$

– σ denotes a nonlinear activation function in layer l
– $a_i^{(l)}$ denotes the output at a neuron i in layer l: $a_i^{(l)} = \sigma\left(z_i^{(l)}\right)$
– $a^{(l)}$ denotes the output vector for layer l: $a^{(l)} = \left\{a_1^{(l)}, a_2^{(l)}, \ldots a_{N_l}^{(l)}\right\}$;
– $w_i^{(l)}$ denotes the weight vector for neuron i in layer l; $w_i^{(l)} = \left\{ w_1^{(l)}, w_2^{(l)}, \ldots w_{N_l}^{(l)}\right\}$
– $w_{ij}^{(l)}$ denotes the weight vector connecting the neuron i in layer l to neuron j in layer *l-1*

11.3.9.1 Simple 1–1–1 Network

Let us take an example of a three-layer neural network ($L = 3$) with an input layer with 1 neuron, a hidden layer with 1 neuron, and an output layer with 1 neuron (Fig. 11.25).

There are only three layers: layer L (output), layer L-1 (hidden), and layer L-2 (input). We have one neuron in each layer, so we will not use the subscript i; for example, instead of a_i^l we will use a^l, and the same applies for all other notations.

$$a^{(L)} = \sigma\left(z^{(L)}\right)$$

$$z^{(L)} = w^{(L-1)} a^{(L-1)} + b^{(L-1)}$$

$$z^{(L-1)} = w^{(L-2)} a^{(L-2)} + b^{(L-2)}$$

We need to compute the gradient (partial derivative) of the error function E (or cost function C) with respect to the weights and the biases. That is, we would like to know how our cost function would change if we changed the weights and biases of the network.

Starting in the last layer, we then investigate how this gradient propagates backward through the network.

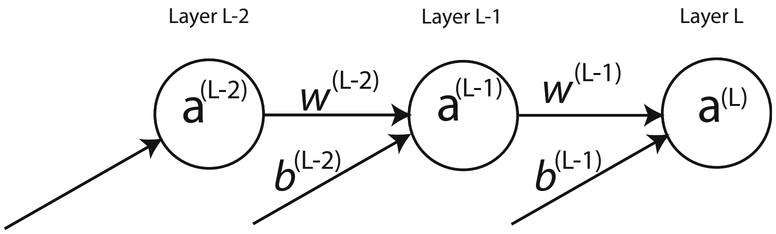


Fig. 11.25 A three-layer neural network formed; each layer is 1 neuron

11.3.9.1.1 Computation with Respect to Layer *L*-1

Having one node per layer will help us understand the computational work and its implications. We will start with the layer L and calculate the gradient of our error function E with respect to the weights and biases of neurons in the previous layer $L-1$, $\frac{\partial E}{\partial w^{(L-1)}}$. Figure 11.26 clarifies the relationship between the cost function and the weights and biases.

Let us consider the mean squared error as an error function. Since we have only one neuron in the output:

$$E = \frac{1}{2}\left(a^{(L)} - y\right)^2$$

$$\left(\text{the } \frac{1}{2} \text{ is for convenience}\right)$$

Using the chain rule, we can write:

$$\frac{\partial E}{\partial w^{(L-1)}} = \frac{\partial E}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L-1)}}$$

$$\frac{\partial E}{\partial a^{(L)}} = \frac{1}{2} \frac{\partial \left(a^{(L)} - y\right)^2}{\partial a^{(L)}} = \frac{1}{2} 2\left(a^{(L)} - y\right) = \left(a^{(L)} - y\right)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \frac{\partial \sigma\left(z^{(L)}\right)}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$\frac{\partial z^{(L)}}{\partial w^{(L-1)}} = \frac{\partial \left(w^{(L-1)} a^{(L-1)} + b^{(L-1)}\right)}{\partial w^{(L-1)}} = a^{(L-1)}$$

Hence, we can solve $\frac{\partial E}{\partial w^{(L-1)}}$:

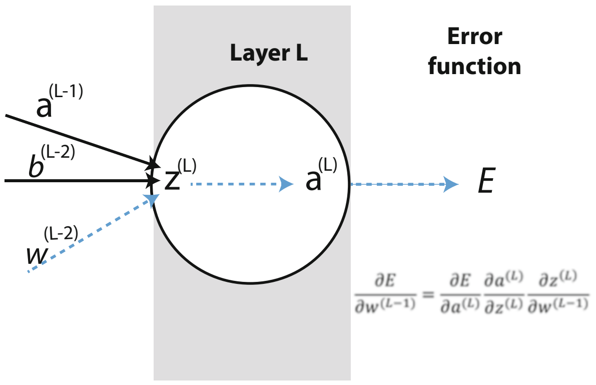

Layer L
Error function
$a^{(L-1)}$
$b^{(L-2)}$
$w^{(L-2)}$
$z^{(L)}$
$a^{(L)}$
E
$\frac{\partial E}{\partial w^{(L-1)}} = \frac{\partial E}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L-1)}}$

Fig. 11.26 The cost function *E*'s relationship with the weights and biases passes through a chain from *E* to *a*, from *a* to *z*, and from *z* to the weights and biases

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}^{(\boldsymbol{L}-\mathbf{1})}} = \left(\boldsymbol{a}^{(\boldsymbol{L})} - \boldsymbol{y}\right)\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(\boldsymbol{L})}\right)\boldsymbol{a}^{(\boldsymbol{L}-\mathbf{1})}$$

Just note that the sigmoid is used as the nonlinear activation function; then, $\sigma = \frac{1}{1+e^{(-z)}}$, and its derivative is $\sigma' = \frac{e^{(-z)}}{\left(1+e^{(-z)}\right)^2}$. Also note that we would like to see how the cost function changes with the change of the weight $w^{(L-2)}$; we will see that in a moment. First, let us see how the cost function changes with the change of the bias $b^{(L-1)}$. We will use the chain rule $\frac{\partial E}{\partial b^{(L-1)}}$, which can be written as follows:

$$\frac{\partial E}{\partial b^{(L-1)}} = \frac{\partial E}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial b^{(L-1)}}$$

$$\frac{\partial E}{\partial a^{(L)}} = \left(a^{(L)} - y\right)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$\frac{\partial z^{(L)}}{\partial b^{(L-1)}} = \frac{\partial\left(w^{(L-1)}a^{(L-1)} + b^{(L-1)}\right)}{\partial w^{(L-1)}} = 1$$

Therefore,

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}^{(\boldsymbol{L}-\mathbf{1})}} = \left(\boldsymbol{a}^{(\boldsymbol{L})} - \boldsymbol{y}\right)\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(\boldsymbol{L})}\right)$$

So, based on the weights and biases' initial values, we have used a training instance to compute the predicted output $a^{(L)}$, then computed the gradient of the cost (i.e., error) with respect to the weights and biases, as we have just seen. We can use those gradients in the following equations to update the weights and biases before going forward with another training round (time t + 1):

$$w^{(L-1)}(t+1) = w^{(L)}(t) - \alpha\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}^{(\boldsymbol{L}-\mathbf{1})}}$$

$$b^{(L-1)}(t+1) = b^{(L)}(t) - \alpha\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}^{(\boldsymbol{L}-\mathbf{1})}}$$

where α is the training rate, *t* denotes a round of training.

11.3.9.1.2 Computation with Respect to Layer *L*-2

We will now proceed further up the network and calculate the gradient of our error function *E* with respect to the weights and biases of neurons in the previous layer

L-2. This is a measurement of how much E changes with respect to changes in weights and biases at level L-2.

But before we can know that, we will need $\frac{\partial E}{\partial a^{(L-1)}}$, so let us compute that derivative.

$$\frac{\partial E}{\partial a^{(L-1)}} = \frac{\partial E}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

$$\frac{\partial E}{\partial a^{(L)}} = \left(a^{(L)} - y\right)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$\frac{\partial z^{(L)}}{\partial a^{(L-1)}} = w^{(L-1)}$$

Hence,

$$\frac{\partial E}{\partial a^{(L-1)}} = \left(a^{(L)} - y\right)\sigma'\left(z^{(L)}\right)w^{(L-1)}$$

Now that we have found the gradient of E with respect to $a^{(L-1)}$, we can proceed with our investigation:

$$\frac{\partial E}{\partial w^{(L-2)}} = \frac{\partial E}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-2)}}$$

$$\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} = \frac{\partial \sigma\left(z^{(L-1)}\right)}{\partial z^{(L-1)}} = \sigma'\left(z^{(L-1)}\right)$$

$$\frac{\partial z^{(L-1)}}{\partial w^{(L-2)}} = \frac{\partial\left(w^{(L-2)}a^{(L-2)} + b^{(L-2)}\right)}{\partial w^{(L-2)}} = a^{(L-2)}$$

$$\frac{\partial E}{\partial w^{(L-2)}} = \frac{\partial E}{\partial a^{(L-1)}} \boldsymbol{\sigma'}\left(\boldsymbol{z^{(L-1)}}\right) \boldsymbol{a^{(L-2)}}$$

$$\boldsymbol{\frac{\partial E}{\partial w^{(L-2)}} = \left(a^{(L)} - y\right)\sigma'\left(z^{(L)}\right)w^{(L-1)}\sigma'\left(z^{(L-1)}\right)a^{(L-2)}}$$

Similarly, $\frac{\partial E}{\partial b^{(L-1)}}$ can be computed as follows:

$$\frac{\partial E}{\partial b^{(L-2)}} = \frac{\partial E}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-2)}}$$

$$\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} = \sigma'\left(z^{(L-1)}\right)$$

$$\frac{\partial z^{(L-1)}}{\partial b^{(L-2)}} = \frac{\partial\left(w^{(L-2)}a^{(L-2)} + b^{(L-2)}\right)}{\partial w^{(L-2)}} = 1$$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}^{(L-2)}} = \frac{\partial E}{\partial a^{(L-1)}}\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(L-1)}\right)$$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}^{(L-2)}} = \left(\boldsymbol{a}^{(L)} - \boldsymbol{y}\right)\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(L)}\right)\boldsymbol{w}^{(L-1)}\boldsymbol{\sigma}'\left(\boldsymbol{z}^{(L-1)}\right)$$

We can also update the weights in level L-2 using the usual formula:

$$w^{(L-2)}(t+1) = w^{(L)}(t) - \alpha\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}^{(L-2)}}$$

$$b^{(L-2)}(t+1) = b^{(L)}(t) - \alpha\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}^{(L-2)}}$$

11.3.9.2 Fully Connected Neural Network

There are a few adjustments that we have to consider when we have a fully connected neural network.

The mean squared error function *E* is still a function of the weights vector and the bias *b* but is now expressed as an average:

$$E(w, b) = \frac{1}{2N}\sum\nolimits_{i=1}^{N}(a_i - y_i)^2$$

where *N* is the number of instances in the training set.

11.3.9.2.1 Computation with Respect to Layer *L*-1

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w_{ij}}^{(L-1)}} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{a_i}^{(L)}}\frac{\partial \boldsymbol{a_i}^{(L)}}{\partial \boldsymbol{z_i}^{(L)}}\frac{\partial \boldsymbol{z_i}^{(L)}}{\partial \boldsymbol{w_{ij}}^{(L-1)}}$$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w_{ij}}^{(L-1)}} = \left(\boldsymbol{a_i}^{(L)} - \boldsymbol{y_i}\right)\boldsymbol{\sigma}'\left(\boldsymbol{z_i}^{(L)}\right)\boldsymbol{a_i}^{(L-1)}$$

$$\frac{\partial E}{\partial a_j{}^{(L-1)}} = \sum_{i=1}^{N^{(l-1)}} \frac{\partial E}{\partial a_i{}^{(L)}} \frac{\partial a_i{}^{(L)}}{\partial z_i{}^{(L)}} \frac{\partial z_i{}^{(L)}}{\partial a_j{}^{(L-1)}}$$

The sum is added, as the activation from every neuron a_j from layer L-1 will affect all the activations of the neurons in layer L, which will affect the cost of the neural network.

11.3.9.2.2 Computation with Respect to Layer *L*-2

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_{\boldsymbol{ij}}{}^{(\boldsymbol{L}-\boldsymbol{2})}} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{a}_{\boldsymbol{i}}{}^{(\boldsymbol{L}-\boldsymbol{1})}} \frac{\partial \boldsymbol{a}_{\boldsymbol{i}}{}^{(\boldsymbol{L}-\boldsymbol{1})}}{\partial \boldsymbol{z}_{\boldsymbol{i}}{}^{(\boldsymbol{L}-\boldsymbol{1})}} \frac{\partial \boldsymbol{z}_{\boldsymbol{i}}{}^{(\boldsymbol{L}-\boldsymbol{1})}}{\partial \boldsymbol{w}_{\boldsymbol{ij}}{}^{(\boldsymbol{L}-\boldsymbol{2})}}$$

We can see clearly from the above that the error in a layer *l* depends on the error in the next layer *l* + *1*; hence, the errors propagate backward from the last to the first layer. Once we compute the error at the output layer and once the partial derivatives for all the neurons are known, the weights can be updated. The process is repeated until convergence.

Note that strictly speaking, the term “backpropagation” refers not to the learning process but to the method used to compute the gradient [10].

11.3.10 Backpropagation Algorithm

The backpropagation algorithm runs in four steps:

1. Forward phase: Proceeding from the input layer to the output layer, for each input-output pair in the training dataset, calculate the predicted output and save the result for each neuron.
2. Backward phase: Proceeding from the output layer to the input layer, calculate and save the resulting gradients.
3. Combine the individual gradients to obtain the total gradient.
4. Update the weights using α and total gradient.
5. Repeat until the minimum cost is reached.

11.4 Final Notes: Advantages, Disadvantages, and Best Practices

Neural networks are considered one of most prominent ML models given its ability to deal with different type of outputs, including discrete, real-value, vectors, images, and many others. Those models can learn and model complex, nonlinear and highly volatile data. Their architecture allows those models to be robust to any noises during the training period. Even with long training period, neural networks can generate interesting results. Note that neural networks can also be used for anomaly detection (even if we are dealing with unlabeled data); in this case, the learning results can be used to give fast second opinion with good accuracy in any used application.

Like an ML model, neural networks need parallel processing power, which makes it hardware dependence in a way. Although it gives promising results, the latter are unexplainable in many cases in terms of why and how we reached such decisions which might affect the trust in such models. In terms of its technical structure, there is no well-defined rule on how to design such architecture (number of hidden layers, number of hidden nodes, error thresholds for best training time and optimal results); it is more of trial-and-error process

With this in mind, we tend to depend on best practices to try and optimize the neural networks results. Some of key practices include the following:

- Always check the size of the training data; if it is not enough, it is important to increase.
- If the model overfits, you can either use simpler network (a smaller number of hidden layers/nodes), use dropout layers, increase data samples, or remove some features (execute preprocessing of data again).
- If the mode underfits, you can add more features (using feature engineering techniques).
- Starting with large batch size can reduce the training time in some cases.
- If the model suffers from vanishing gradient problem, using lower learning rate might allow the model to converge.
- Normalizing the inputs in every layer might help the stability and performance of the model.

11.5 Key Terms

1. Artificial neural networks (ANN)
2. McCulloch–Pitts (M-P) neuron
3. Perceptron
4. Linear function
5. Linear model

6. Bipolar activation functions
7. Unipolar activation functions
8. Sigmoid function
9. Hyperbolic tangent function
10. Tanh function
11. Rectified unit function
12. ReLU function
13. Leaky ReLU function
14. Parameterized ReLU function
15. Swish function
16. SoftMax function
17. Training the perceptron
18. Gradient descent
19. Stochastic gradient descent
20. XOR
21. Exclusive OR
22. Multilayer perceptron
23. MLP
24. Backpropagation
25. Chain rule
26. Fully connected neural network

11.6 Test Your Understanding

1. Can we identify a perceptron as a linear classifier or a nonlinear one?
2. What type of problems does a perceptron solve?
3. Why should the activation function of a multilayer perceptron be nonlinear?
4. What is the aim of backpropagation?
5. Explain backpropagation in simple words for a specialist.
6. The hyperbolic tangent function overcomes a problem we find in the sigmoid functions. What is it?
7. Why does ReLU perform better than tanh and sigmoid functions?
8. Explain the "dead" neuron problem and how to overcome it.
9. What kind of issues does a leaky ReLU overcome in comparison with a ReLU?
10. SoftMax is very useful to solve a specific kind of problem; what is it?

11.7 Read More

1. Cao, J., Qian, S., Zhang, H., Fang, Q., & Xu, C. (2021). Global Relation-Aware Attention Network for Image-Text Retrieval Proceedings of the 2021 International Conference on Multimedia Retrieval, Taipei, Taiwan. https://doi.org/10.1145/3460426.3463615

2. Chatterjee, B., & Sen, S. (2021). Energy-Efficient Deep Neural Networks with Mixed-Signal Neurons and Dense-Local and Sparse-Global Connectivity Proceedings of the 26th Asia and South Pacific Design Automation Conference, Tokyo, Japan. https://doi.org/10.1145/3394885.3431614
3. Collins, J., Sun, S., Guo, C., Podgorsak, A., Rudin, S., & Bednarek, D. R. (2021). Estimation of Patient Eye-Lens Dose During Neuro-Interventional Procedures using a Dense Neural Network (DNN). Proc SPIE Int Soc Opt Eng, 11,595. https://doi.org/10.1117/12.2580723
4. Grasemann, U., Peñaloza, C., Dekhtyar, M., Miikkulainen, R., & Kiran, S. (2021). Predicting language treatment response in bilingual aphasia using neural network-based patient models. Sci Rep, 11(1), 10,497. https://doi.org/10.1038/s41598-021-89443-6
5. Hasan, N. (2021). A Hybrid Method of Covid-19 Patient Detection from Modified CT-Scan/Chest-X-Ray Images Combining Deep Convolutional Neural Network And Two- Dimensional Empirical Mode Decomposition. Comput Methods Programs Biomed Update, 1, 100,022. https://doi.org/10.1016/j.cmpbup.2021.100022
6. Kimura, Y., Kadoya, N., Oku, Y., Kajikawa, T., Tomori, S., & Jingu, K. (2021). Error detection model developed using a multi-task convolutional neural network in patient-specific quality assurance for volumetric-modulated arc therapy. Med Phys. https://doi.org/10.1002/mp.15031
7. Maharaj, S., Qian, T., Ohiba, Z., & Hayes, W. (2021). Common Neighbors Extension of the Sticky Model for PPI Networks Evaluated by Global and Local Graphlet Similarity. IEEE/ACM Trans. Comput. Biol. Bioinformatics, 18(1), 16–26. https://doi.org/10.1109/tcbb.2020.3017374
8. Minagawa, A., Koga, H., Sano, T., Matsunaga, K., Teshima, Y., Hamada, A., Houjou, Y., & Okuyama, R. (2021). Dermoscopic diagnostic performance of Japanese dermatologists for skin tumors differs by patient origin: A deep learning convolutional neural network closes the gap. J Dermatol, 48(2), 232–236. https://doi.org/10.1111/1346-8138.15640
9. Pan, Q., Zhang, L., Jia, M., Pan, J., Gong, Q., Lu, Y., Zhang, Z., Ge, H., & Fang, L. (2021). An interpretable 1D convolutional neural network for detecting patient-ventilator asynchrony in mechanical ventilation. Comput Methods Programs Biomed, 204, 106,057. https://doi.org/10.1016/j.cmpb.2021.106057
10. Shihada, B., Elbatt, T., Eltawil, A., Mansour, M., Sabir, E., Rekhis, S., & Sharafeddine, S. (2021). Networking research for the Arab world: from regional initiatives to potential global impact. Commun. ACM, 64(4), 114–119. https://doi.org/10.1145/3447748
11. Shorfuzzaman, M., Masud, M., Alhumyani, H., Anand, D., & Singh, A. (2021). Artificial Neural Network-Based Deep Learning Model for COVID-19 Patient Detection Using X-Ray Chest Images. J Healthc Eng, 2021, 5,513,679. https://doi.org/10.1155/2021/5513679
12. Sridhara, S., Wirz, F., Ruiter, J. d., Schutijser, C., Legner, M., & Perrig, A. (2021). Global Distributed Secure Mapping of Network Addresses Proceedings of the ACM SIGCOMM 2021 Workshop on Technologies, Applications,

and Uses of a Responsible Internet, Virtual Event, USA. https://doi.org/10.1145/3472951.3473503

13. Valizadeh, A., Jafarzadeh Ghoushchi, S., Ranjbarzadeh, R., & Pourasad, Y. (2021). Presentation of a Segmentation Method for a Diabetic Retinopathy Patient's Fundus Region Detection Using a Convolutional Neural Network. Comput Intell Neurosci, 2021, 7,714,351. https://doi.org/10.1155/2021/7714351
14. Xiao, Y., Wang, X., Li, Q., Fan, R., Chen, R., Shao, Y., Chen, Y., Gao, Y., Liu, A., Chen, L., & Liu, S. (2021). A cascade and heterogeneous neural network for CT pulmonary nodule detection and its evaluation on both phantom and patient data. Comput Med Imaging Graph, 90, 101,889. https://doi.org/10.1016/j.compmedimag.2021.101889
15. Zhong, Y. W., Jiang, Y., Dong, S., Wu, W. J., Wang, L. X., Zhang, J., & Huang, M. W. (2021). Tumor radiomics signature for artificial neural network-assisted detection of neck metastasis in patient with tongue cancer. J Neuroradiol. https://doi.org/10.1016/j.neurad.2021.07.006
16. Zhu, Y., Xie, R., Zhuang, F., Ge, K., Sun, Y., Zhang, X., Lin, L., & Cao, J. (2021). Learning to Warm Up Cold Item Embeddings for Cold-start Recommendation with Meta Scaling and Shifting Networks. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1167–1176). Association for Computing Machinery. https://doi.org/10.1145/3404835.3462843

11.8 Lab

11.8.1 Working Example in Python

The diabetes dataset that is used in this lab can be downloaded from the following link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

This is a binary classification problem. This dataset contains the following information:

- Pregnancies: number of pregnancies
- Glucose: plasma glucose concentration
- Blood Pressure: diastolic blood pressure measurement
- SkinThikness: triceps skinfold thickness (mm)
- Insulin: 2-hour serum insulin
- BMI: body mass index (BMI)
- DiabetesPedigreeFunction: diabetes pedigree function
- Age: the person's age
- Outcome: tested positive for diabetes or not (1 or 0)

11.8.1.1 Load Diabetes for Pima Indians Dataset

Before loading the Pima Indians dataset, it is important to note that we need to install the Keras, TensorFlow, and SciKeras libraries using the pip install command to create the sequential neural network model.

For visualizing neural network, the Graphviz library is used. Graphviz Python can be downloaded from the following link: https://www.graphviz.org/download/

After downloading Graphviz, the path in the system environment variables needs to be edited to include:

C:\Program Files\Graph viz.\bin

C:\Program Files\Graphviz\bin\dot.exe

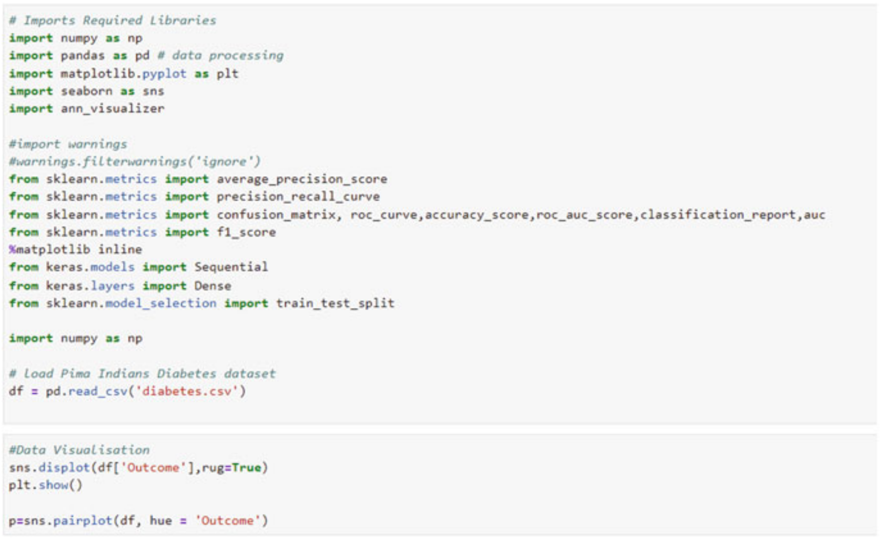
We start by importing the required libraries and loading the dataset and display a bar chart for the outcomes a well as pair plots for the features (Fig. 11.27). The displayed graphs are partially shown in Fig. 11.28.

11.8.1.2 Visualize Data

We explore the data visually (Fig. 11.28).

11.8.1.3 Split Dataset into Training and Testing Datasets

The next task is to choose features and target (the "Outcome"). The next step is to split the dataset into training and testing and standardize both (Fig. 11.29).



```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import ann_visualizer

#import warnings
#warnings.filterwarnings('ignore')
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix, roc_curve,accuracy_score,roc_auc_score,classification_report,auc
from sklearn.metrics import f1_score
%matplotlib inline
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

import numpy as np

# Load Pima Indians Diabetes dataset
df = pd.read_csv('diabetes.csv')
```

```
#Data Visualisation
sns.displot(df['Outcome'],rug=True)
plt.show()

p=sns.pairplot(df, hue = 'Outcome')
```

Fig. 11.27 Load Pima Indians diabetes dataset

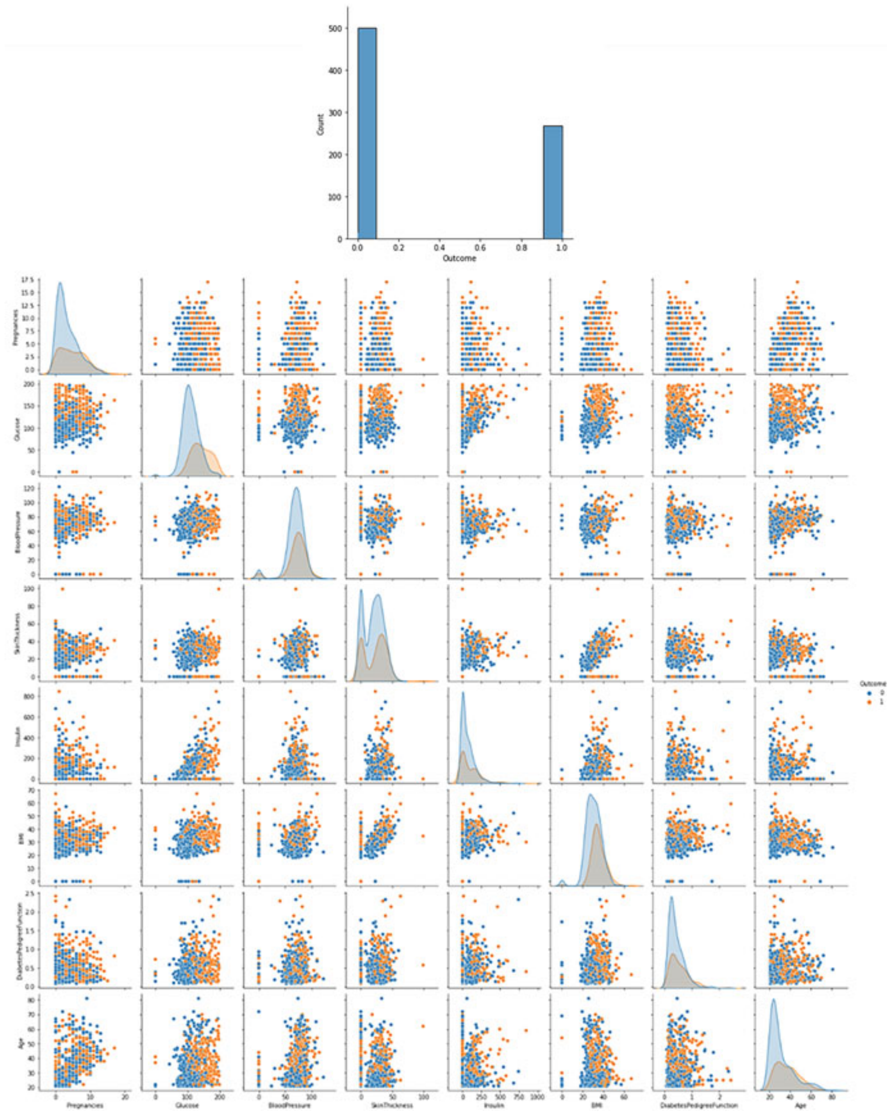
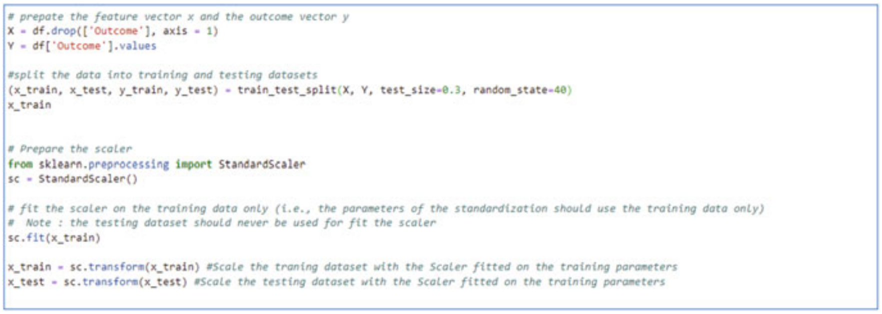


Fig. 11.28 Visualizing diabetic vs. nondiabetic Pima Indians

11.8.1.4 Create Neural Network Model

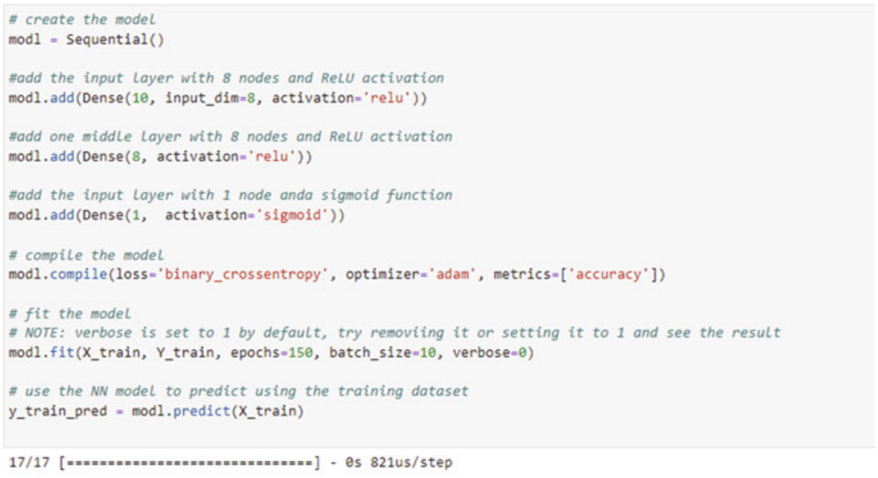
The next task is to create the sequential neural network model using the Keras library. As expected the input layer has eight nodes to accommodate the 8 features. WE have chosen to add two hidden layers are added, one with 10 nodes and the other with eight nodes (Fig. 11.30).

We can display the NN structure using the graphviz library (Fig. 11.31).



```
# prepate the feature vector x and the outcome vector y
X = df.drop(['Outcome'], axis = 1)
Y = df['Outcome'].values

#split the data into training and testing datasets
(x_train, x_test, y_train, y_test) = train_test_split(X, Y, test_size=0.3, random_state=40)
x_train


# Prepare the scaler
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

# fit the scaler on the training data only (i.e., the parameters of the standardization should use the training data only)
#  Note : the testing dataset should never be used for fit the scaler
sc.fit(x_train)

x_train = sc.transform(x_train) #Scale the traning dataset with the Scaler fitted on the training parameters
x_test = sc.transform(x_test) #Scale the testing dataset with the Scaler fitted on the training parameters
```

Fig. 11.29 Splitting and scaling Pima Indians diabetes dataset



```
# create the model
modl = Sequential()

#add the input layer with 8 nodes and ReLU activation
modl.add(Dense(10, input_dim=8, activation='relu'))

#add one middle layer with 8 nodes and ReLU activation
modl.add(Dense(8, activation='relu'))

#add the input layer with 1 node anda sigmoid function
modl.add(Dense(1,  activation='sigmoid'))

# compile the model
modl.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the model
# NOTE: verbose is set to 1 by default, try removiing it or setting it to 1 and see the result
modl.fit(X_train, Y_train, epochs=150, batch_size=10, verbose=0)

# use the NN model to predict using the training dataset
y_train_pred = modl.predict(X_train)
```

```
17/17 [==============================] - 0s 821us/step
```

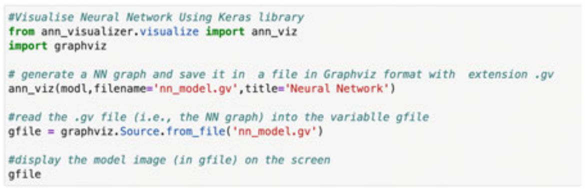
Fig. 11.30 Creating sequential neural network model

11.8.1.5 Optimize Neural Network Model Using Hyperparameter

For model optimization, we use the grid search cross-validation approach (Fig. 11.32). The hyperparameters used for the grid search are the batch size and the number of epochs. We conclude that the model has fair performance (AUC = 72% and accuracy 75%) and can be used on an unseen dataset.

11.8.2 Working Example in Weka

Download the Boston housing dataset from the following website:
https://www.kaggle.com/prasadperera/the-boston-housing-dataset



```
#Visualise Neural Network Using Keras library
from ann_visualizer.visualize import ann_viz
import graphviz

# generate a NN graph and save it in  a file in Graphviz format with  extension .gv
ann_viz(modl,filename='nn_model.gv',title='Neural Network')

#read the .gv file (i.e., the NN graph) into the variablle gfile
gfile = graphviz.Source.from_file('nn_model.gv')

#display the model image (in gfile) on the screen
gfile
```

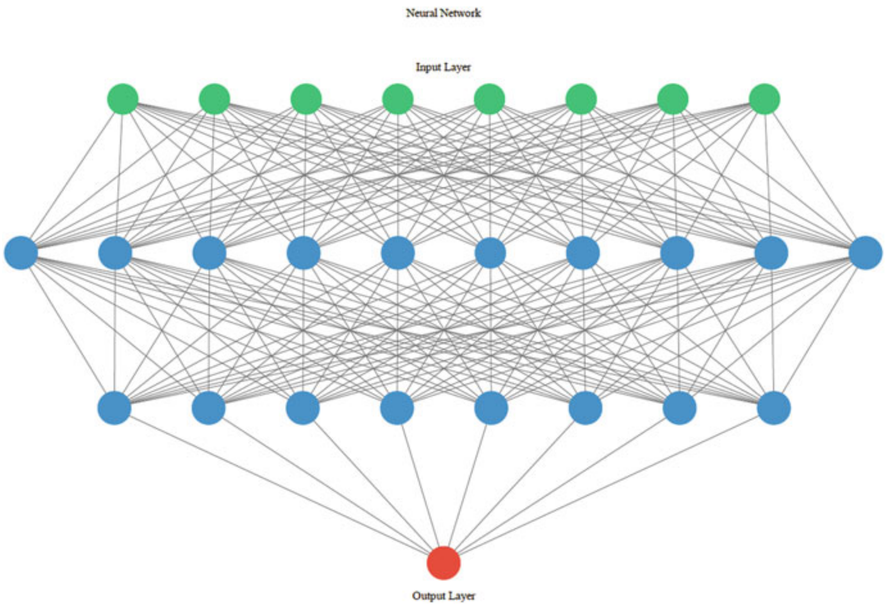
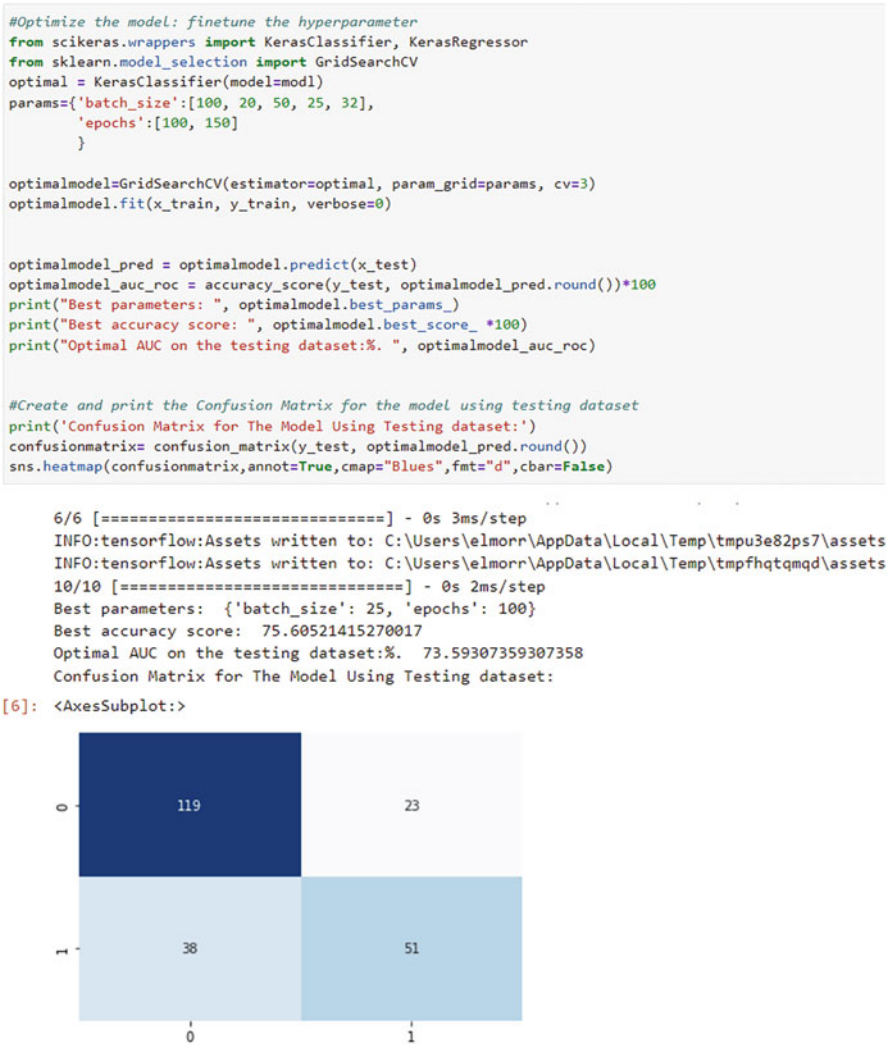


Fig. 11.31 Displaying a sequential neural network model

Open the file in Weka, go to the Classify tab, and choose the Multilayer Perceptron algorithm from Functions (Fig. 11.33).

Click on the function and notice the parameters for the algorithm (Fig. 11.34).

One of the most important parameters is the number of hidden layers; it is set to automatic by default (i.e., the letter *a* denotes automatic), but it can be set to any number you want. The learning rate can be changed; the default is 0.3. Click on GUI and make it True, then click OK, then click Start to run the algorithm. The result is shown in Fig. 11.35, and its graphical representation is shown in Fig. 11.36.



```
#Optimize the model: finetune the hyperparameter
from scikeras.wrappers import KerasClassifier, KerasRegressor
from sklearn.model_selection import GridSearchCV
optimal = KerasClassifier(model=modl)
params={'batch_size':[100, 20, 50, 25, 32],
        'epochs':[100, 150]
        }

optimalmodel=GridSearchCV(estimator=optimal, param_grid=params, cv=3)
optimalmodel.fit(x_train, y_train, verbose=0)


optimalmodel_pred = optimalmodel.predict(x_test)
optimalmodel_auc_roc = accuracy_score(y_test, optimalmodel_pred.round())*100
print("Best parameters: ", optimalmodel.best_params_)
print("Best accuracy score: ", optimalmodel.best_score_ *100)
print("Optimal AUC on the testing dataset:%. ", optimalmodel_auc_roc)


#Create and print the Confusion Matrix for the model using testing dataset
print('Confusion Matrix for The Model Using Testing dataset:')
confusionmatrix= confusion_matrix(y_test, optimalmodel_pred.round())
sns.heatmap(confusionmatrix,annot=True,cmap="Blues",fmt="d",cbar=False)
```

```
     6/6 [==============================] - 0s 3ms/step
     INFO:tensorflow:Assets written to: C:\Users\elmorr\AppData\Local\Temp\tmpu3e82ps7\assets
     INFO:tensorflow:Assets written to: C:\Users\elmorr\AppData\Local\Temp\tmpfhqtqmqd\assets
     10/10 [==============================] - 0s 2ms/step
     Best parameters:  {'batch_size': 25, 'epochs': 100}
     Best accuracy score:  75.60521415270017
     Optimal AUC on the testing dataset:%.  73.59307359307358
     Confusion Matrix for The Model Using Testing dataset:
[6]: <AxesSubplot:>
```

Fig. 11.32 Optimize the neural network model using grid search and its performance

11.8.3 Do it Yourself

11.8.3.1 Diabetes Revisited

How can you enhance the results of the Neural Network above? Hint: think of changing the number of hidden layers, and the number of nodes in each. We have used above standardization but neural network expects values between 0 and 1, would normalization allow the NN to perform better?

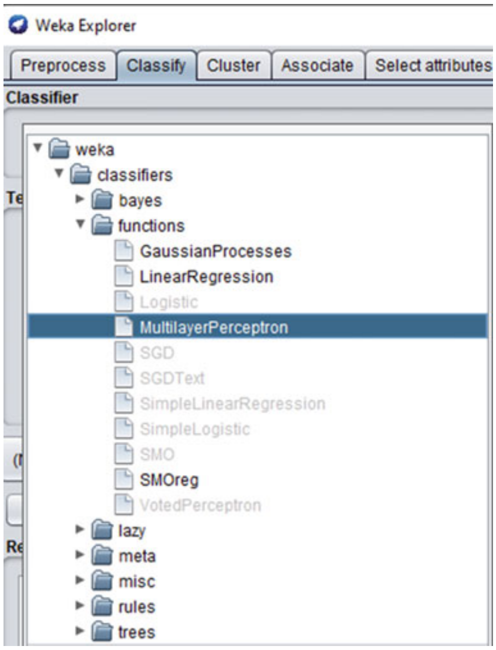
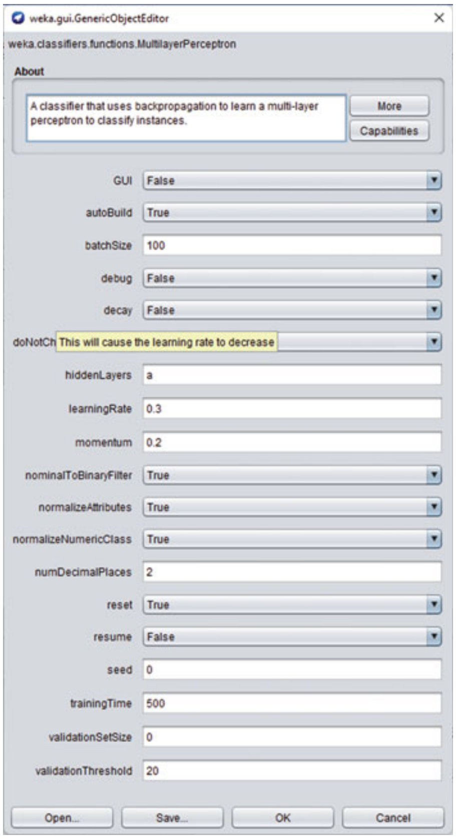


Fig. 11.33 Weka multilayer perceptron algorithm


weka.gui.GenericObjectEditor

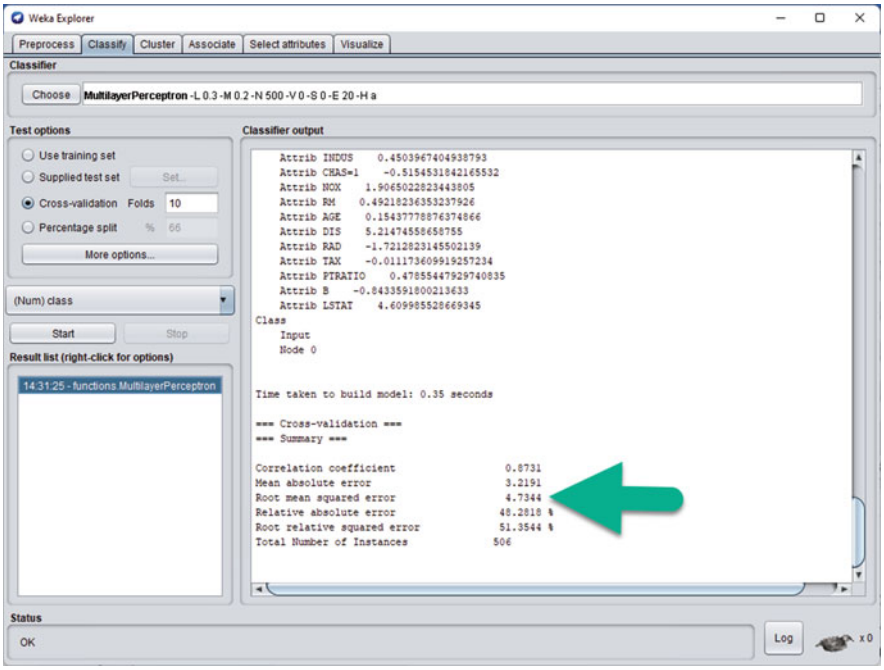
weka.classifiers.functions.MultilayerPerceptron

About

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

GUI	False
autoBuild	True
batchSize	100
debug	False
decay	False
doNotCh	This will cause the learning rate to decrease
hiddenLayers	a
learningRate	0.3
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
resume	False
seed	0
trainingTime	500
validationSetSize	0
validationThreshold	20

Fig. 11.34 Multilayer perceptron parameters window in Weka



MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

(•) Cross-validation Folds 10

% 66

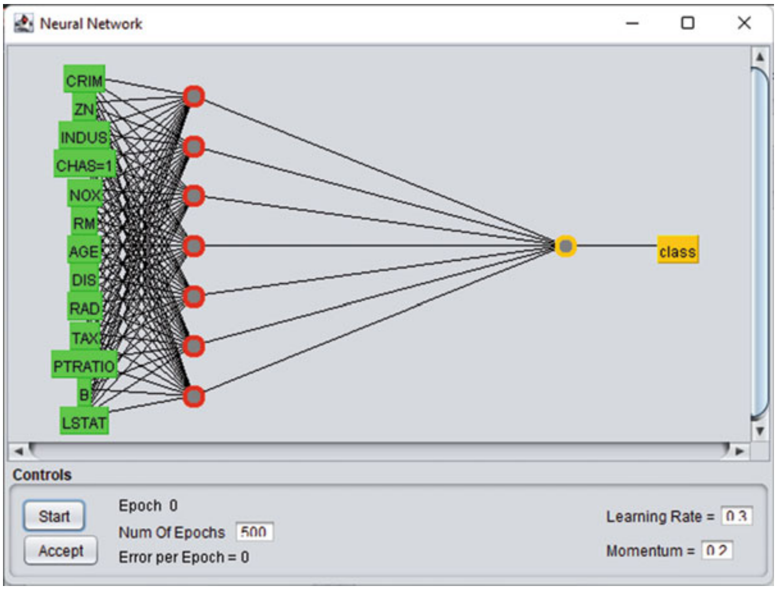
(Num) class

14:31:25 - functions.MultilayerPerceptron

```
    Attrib INDUS    0.4503967404938793
    Attrib CHAS=1    -0.5154531842165532
    Attrib NOX    1.9065022823443805
    Attrib RM    0.49218236353237926
    Attrib AGE    0.15437778876374866
    Attrib DIS    5.21474558658755
    Attrib RAD    -1.7212823145502139
    Attrib TAX    -0.011173609919257234
    Attrib PTRATIO    0.47855447929740835
    Attrib B    -0.8433591800213633
    Attrib LSTAT    4.609985528669345
Class
    Input
    Node 0


Time taken to build model: 0.35 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8731
Mean absolute error                      3.2191
Root mean squared error                  4.7344
Relative absolute error                 48.2818 %
Root relative squared error             51.3544 %
Total Number of Instances              506
```

OK

Fig. 11.35 MLP results in Weka; we can notice RMSE = 4.73



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

Fig. 11.36 The neural network's graphical representation

11.8.3.2 Choose your Own Problem

Pick a problem of your own and apply NN. Discuss the results with another person. Compare your result with someone else who used NN to solve the same problem. Note the differences in the results.

11.8.4 Do More Yourself

Solve each of the following predictive problems using neural networks.

1. Boston house prices:
   You can load the Boston house prices data file (as well as many other datasets) from within python by writing: boston = dataset.load_boston()
2. Predicting stock prices using neural networks.
   Download the dataset from https://www.kaggle.com/datasets/paultimothymooney/stock-market-data
3. Handwritten digit recognition.
   Download the dataset from http://yann.lecun.com/exdb/mnist/

References

1. Z.H. Zhou, *Introduction, Ensemble Methods: Foundations and Algorithms* ((Chapman & Hall/CRC Machine Learning & Pattern Recognition Series: CRC Press, 2012)
2. M. Gopal, *Applied Machine Learning* (McGraw-Hill Education, 2018)
3. T. Trappenberg, *Fundamentals of Machine Learning* (OUP, Oxford, 2019)
4. W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1 Dec 1943). https://doi.org/10.1007/BF02478259
5. G. Palm, Warren McCulloch and Walter Pitts: A logical calculus of the ideas immanent in nervous activity, in *Brain Theory*, (Springer, Berlin, Heidelberg, 1986), pp. 229–230
6. D. Gupta, *Fundamentals of deep learning – activation functions and when to use them?* https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/#. Accessed 7 Sep 2021
7. Brilliant.org, *Backpropagation*. Brilliant.org. https://brilliant.org/wiki/backpropagation/. Accessed 25 Aug 2021
8. A. Burkov, *The Hundred-Page Machine Learning Book* (Andriy Burkov, 2019)
9. S. Kostadinov, Understanding Backpropagation Algorithm: Learn the nuts and bolts of a neural network's most important ingredient. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd. Accessed 7 Sep 2021
10. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016)

Chapter 12
K-Means

12.1 The Problem

So far, we have covered supervised learning algorithms for which the classes are predefined and the class of each instance of the training dataset is known in advance. The problem was to find a model that correctly classifies instances into their appropriate classes with a minimal cost (i.e., minimal error rate).

The problem in unsupervised learning is different; we have a dataset, but neither the classes nor the way to classify each instance in the training dataset is known in advance, i.e., the samples in the dataset are not labeled. For example, one might have data about residents of a city and want to cluster them geographically based on their assumed support for a candidate for election, or a medical image and want to cluster the pixel to perform image segmentation into similar regions, or a dataset about houses' characteristics (e.g., number of bedrooms, number of bathrooms, area, price, postal code) and want to cluster them by their value. Hospitals would be interested in uncovering clusters of patients who are high users of their services. Businesses might want to perform customer segmentation, i.e., to cluster customers based on some criteria, such as their purchases and their website activities; then, the businesses can recommend products for customers in the same clusters [1].

Our aim is to build a model that uncovers the clusters of data latent in the dataset based on feature similarities and classifies the instances into these clusters with a minimal error rate. How to label these clusters is the job of the data analyst.

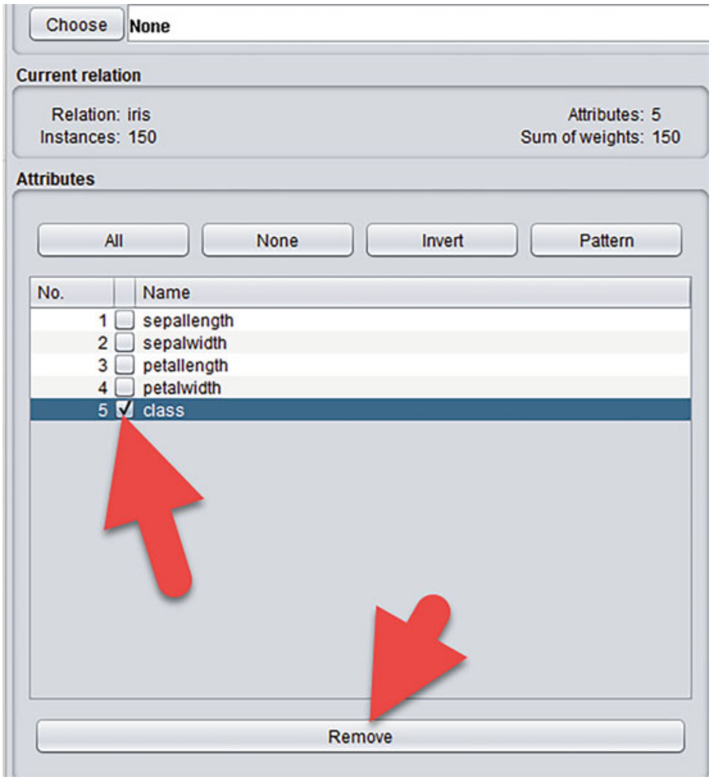
C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_12

12.2 A Practical Example

Let us consider the iris database and discount our knowledge of the class attribute. The instances are now not labeled, and we are in front of a clustering problem. We would like to use *K*-means to cluster the data into *n* clusters/categories. In this problem, we know that there are three clusters (*Iris setosa*, *Iris versicolor*, and *Iris virginica*); however, usually, *n* is suggested by a domain expert who understands the reality the data describes.

The dataset contains the length and width of the sepal and petal of the three types of irises. The dataset is labeled (i.e., class attribute); we can either delete the label or, if we are using Weka, we can ask the *K*-means algorithm to ignore the label. Figure 12.1 shows how to delete the label in Weka, while Fig. 12.6 shows how to ignore it when executing *K*-means in Weka; as usual, in the lab, you will be using Python and R.

We can start by exploring the dataset to have an understanding of the data trends and the problem. Figure 12.2 shows the histograms for each attribute by class. The dataset contains 50 instances for each type of iris. The *petallength* and *petalwidth*



Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150

No.		Name
1	☐	sepallength
2	☐	sepalwidth
3	☐	petallength
4	☐	petalwidth
5	✓	class

Fig. 12.1 Remove the class attribute

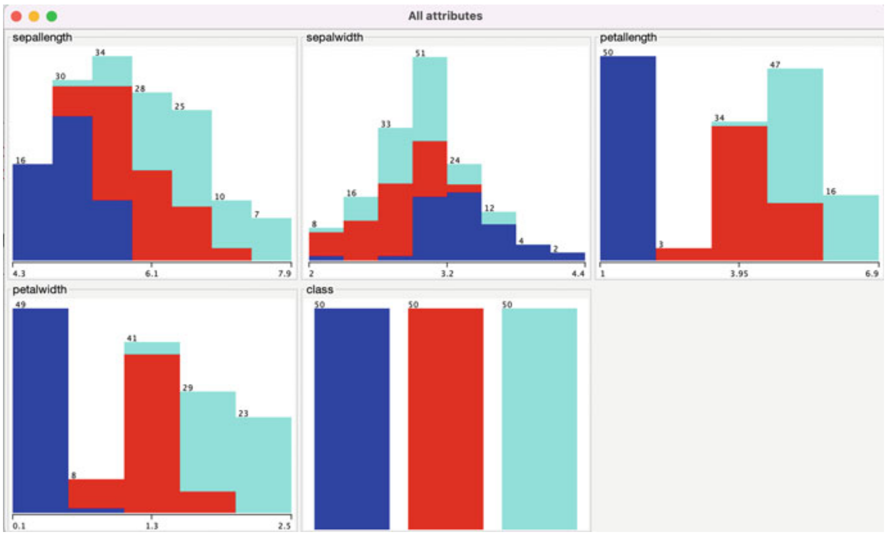


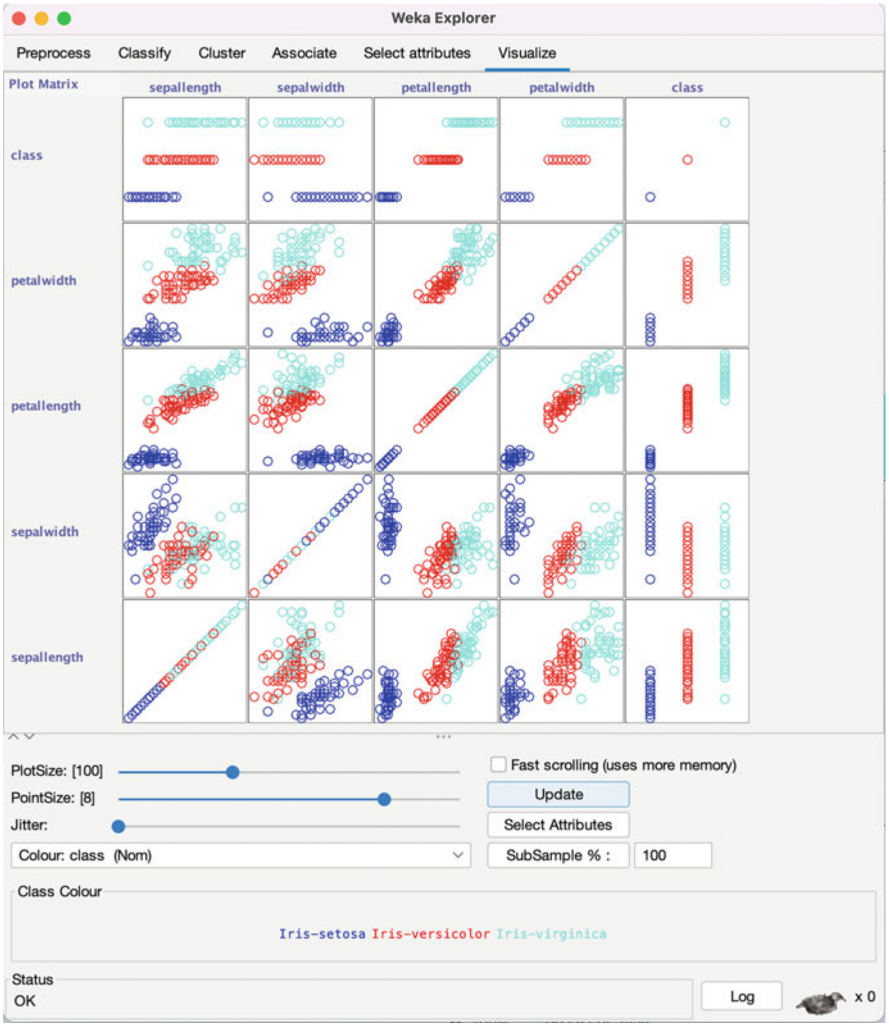
Fig. 12.2 Histograms for the different dataset features

histograms indicate that the petal length and width of one of the iris types (*Iris setosa*) are clearly distinct from the other two types, and hence these two attributes will help us identify *Iris setosa*. *Iris versicolor* and *Iris virginica* have common petal length and width, and the sepal length and width do not provide enough information to distinguish a separate class, as all three types of flowers have common values for these two attributes. We can already conclude that we will encounter errors in clustering, especially between *Iris versicolor* and *Iris virginica*.

When we explore the graph plots for the dataset features (Fig. 12.3), the scatterplot shows the feature on the *X*-axis (i.e., in the columns) against all other features on the *Y*-axis (i.e., in the rows). Again, we notice that the *Iris setosa* is separated from the other two species in each of the scatterplots. *Iris versicolor* and *Iris virginica* have many of their instances intermixed and are hard to distinguish in all the scatterplots (e.g., sepal width vs. sepal length).

Given these observations, let us apply the *K*-means algorithm to the dataset and explore the result (Fig. 12.4). Click on the algorithm's name to open the list of its parameters; since we know that we are seeking to detect three clusters, we will change the numClusters (i.e., *K*) parameter to 3 (Fig. 12.5); the default distance used is Euclidean, which fits our problem (i.e., length and width). Finally, we choose to ignore the Class parameter, as the label is not part of the features that we would use in a clustering problem (Fig. 12.6), and we run the algorithm.

The resulting window (Fig. 12.7) displays important information. We can notice that the algorithm converged after three iterations. The clusters are enumerated from 0 to $K - 1$, and Weka will display the centroid positions for each iteration (e.g., Cluster 0: 6.1, 2.9, 4.7, 1.4); the sum of the square error that we are trying to minimize is 6.99, and the number of instances that were assigned to the three clusters



PlotSize: [100]
PointSize: [8]
Jitter:
Colour: class (Nom)
SubSample % : 100
Class Colour
Iris-setosa Iris-versicolor Iris-virginica
Status
OK

Fig. 12.3 Scatterplots for the dataset features

is 61, 50, and 39; since we know the labels, in this particular example, we know that there were instances assigned to the wrong cluster, but this is already expected given the instances' intermixing that we noticed during the data visualization phase.

Since the dataset contains the labels, we can ask *K*-means to consider the labels instead of ignoring them to let us know how many instances were assigned to the wrong cluster. In Weka, we can do so by choosing the "classes to clusters evaluation" and selecting the class from the list of features (Fig. 12.8). Clicking on Start this time shows an extra output (Fig. 12.9), the incorrectly clustered instances. We notice that the *K*-means algorithm incorrectly clustered 17 instances: 14 *Iris virginica* were

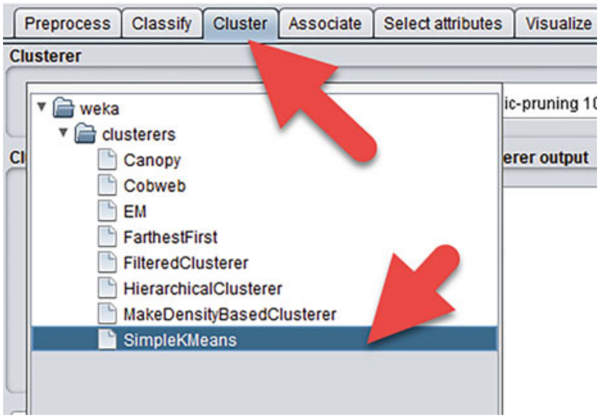


Fig. 12.4 Choose the SimpleKMeans algorithm from the Cluster tab in Weka

clustered with the *Iris versicolor*, and three *Iris versicolor* were clustered with the *Iris virginica*; the 50 instances of *Iris setosa* were all correctly clustered. As expected, the errors were related to the distinction between the *Iris versicolor* and *Iris virginica* species.

We can always visualize the cluster assignments by right-clicking on the name of the algorithm in the Result window and clicking on "Visualize Cluster Assignments" (Fig. 12.10). A new window opens and displays a scatterplot for data on the *X*- and *Y*-axes. To display a scatterplot showing the classes on the *Y*-axis, click on the Y dropdown list and choose the Cluster attribute. The scatterplot shows us how the instances were assigned, and we can see clearly that cluster 1 is distinct, 13 instances are assigned to cluster 0 while they clearly belong to cluster 2, and three instances are assigned to class 0 while they seem to belong to class 2 (Fig. 12.11).

We can check the cluster to which each instance was assigned by using a filter called AddCluster (Fig. 12.12).

We can click on the filter to choose the clustering algorithm and set its parameters. In our case, we would like to set *K* to 3 (Fig. 12.13).

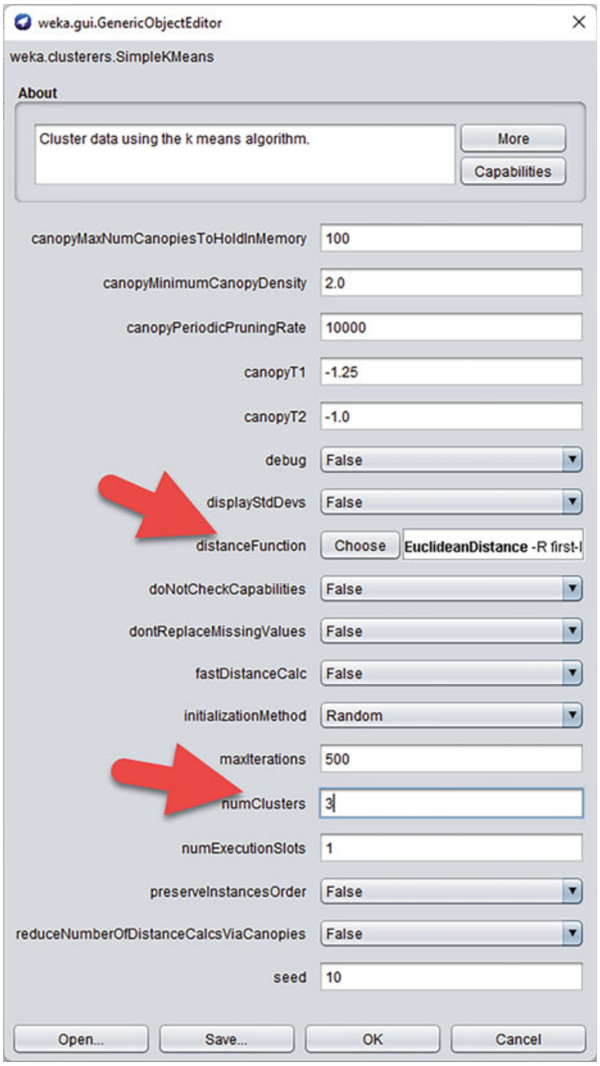
When we apply the filter, a new attribute called "cluster" is created; it indicates the cluster associated with each instance. To display the data, it is enough to click on the Edit button (Fig. 12.14).

12.3 The Algorithm

The *K*-means algorithm involves the following steps:

- Specify the number of clusters *K*.
- Initialize the centroids by randomly selecting *K* samples from the training dataset.
- Assign samples to the clusters based on the closest centroid. The closeness is determined using a distance (e.g., Euclidean, Manhattan).

Fig. 12.5 Set the numClusters (i.e., *K*) parameters of the *K*-means algorithm to 3


weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

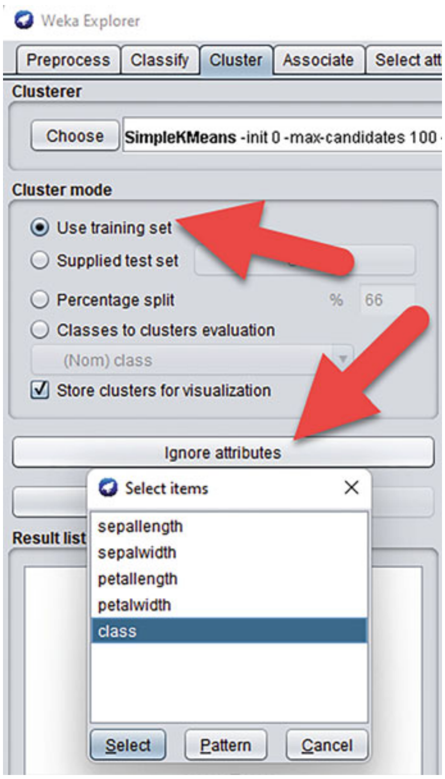
About

Cluster data using the k means algorithm.

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	3
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

– Update the centroid for each cluster by recalculating the center point for each cluster. The latter is the mean of the cluster's samples, hence the name *K*-means.
– Repeat assigning samples and updating centroids until model convergence, i.e., there is little change in the centroids, or a certain number of iterations is completed.

After convergence, the model is represented by the centroids. Each new sample is assigned to the closest centroid; that is, each new sample is assigned to one of the clusters 1 to *K* [2].



SimpleKMeans -init 0 -max-candidates 100

● Use training set

☑ Store clusters for visualization

sepallength
sepalwidth
petallength
petalwidth
class

Fig. 12.6 Choose to ignore the class label before executing *K*-means

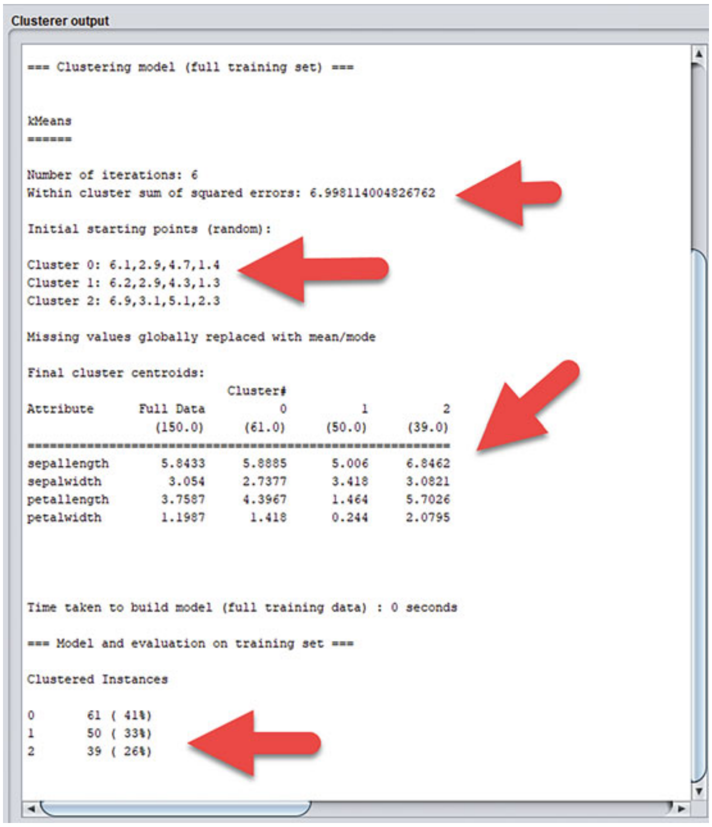
We can represent the dataset with a matrix of N instances with n features, the instances being the rows and the features, the columns of the matrix.

$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_n^{(N)} \end{bmatrix}$$

Each vector $x^{(i)} = \left[x_1^{(i)}\ x_2^{(i)} \ldots x_n^{(i)}\right]^T$ and $X = [x^{(1)}\ x^{(2)} \ldots x^{(N)}]^T$.

The K-means algorithm computes the distance between each vector $x^{(i)}$ and the centroids of the cluster $\mu_k\ k = 1, \ldots, K$. If N_k is the number of instances in the cluster k, then the centroid μ_k is calculated as follows:

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} x^{(i)}$$



Clusterer output

```
=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 6.998114004826762

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4
Cluster 1: 6.2,2.9,4.3,1.3
Cluster 2: 6.9,3.1,5.1,2.3

Missing values globally replaced with mean/mode

Final cluster centroids:
                          Cluster#
Attribute      Full Data         0          1          2
                 (150.0)    (61.0)     (50.0)     (39.0)
==========================================================
sepallength       5.8433    5.8885      5.006     6.8462
sepalwidth         3.054    2.7377      3.418     3.0821
petallength       3.7587    4.3967      1.464     5.7026
petalwidth        1.1987     1.418      0.244     2.0795




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       61 ( 41%)
1       50 ( 33%)
2       39 ( 26%)
```

Fig. 12.7 *K*-means clustering result

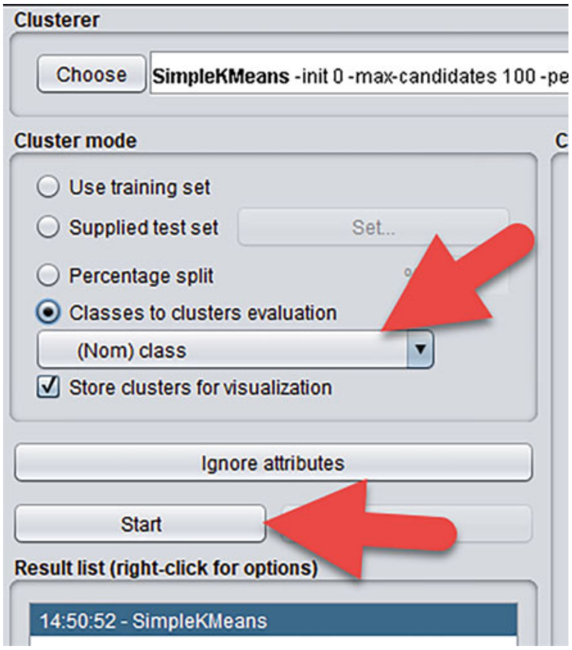
Using the Euclidean distance (note that other distances can be used) to measure similarities between instances [3], the distance between an instance μ_k and a centroid μ_k can be computed as follows:

$$d_{ik} = \sqrt{\sum_{i=1}^{n} \left(x_j^{(i)} - \mu_{kj}\right)^2}$$

If we use the matrix notation, we can rewrite the same as follows:

$$x^{(i)} = \left(\left[x^{(i)} - \mu_k\right]^T \left[x^{(i)} - \mu_k\right]\right)^{1/2} = \left\|x^{(i)} - \mu_k\right\|$$

Each $x^{(i)}$ is considered as belonging to a cluster K if it is closest to it and hence satisfies the following condition:



Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -pe

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- [✓ Classes to clusters evaluation]

(Nom) class

[✓] Store clusters for visualization

Result list (right-click for options)

14:50:52 - SimpleKMeans

Fig. 12.8 Analyzing the incorrectly clustered instances

$$\left\|x^{(i)}-\mu_k\right\|<\left\|x^{(i)}-\mu_j\right\|;j=1,\ldots,K,j\neq k$$

The algorithm stops when the change in the number of instances belonging to the clusters is minimal (less than a certain constant) or the clusters' center's location is minimal; the stopping criteria is

either

$$\sum_{k=1}^{K}\left|N_k^{t+1}-N_k^{t}\right|<\varepsilon$$

or

$$\sum_{k=1}^{K}\left|\mu_k^{t+1}-\mu_k^{t}\right|<\varepsilon$$

The algorithm can be summarized as follows:

1. Initialize k, and $\mu_1^{(t_0)}$ to $\mu_k^{(t_0)}$, and set the time $t = 1$.
2. Classify the N instances according to the nearest $\mu_k^{(t-1)}$:

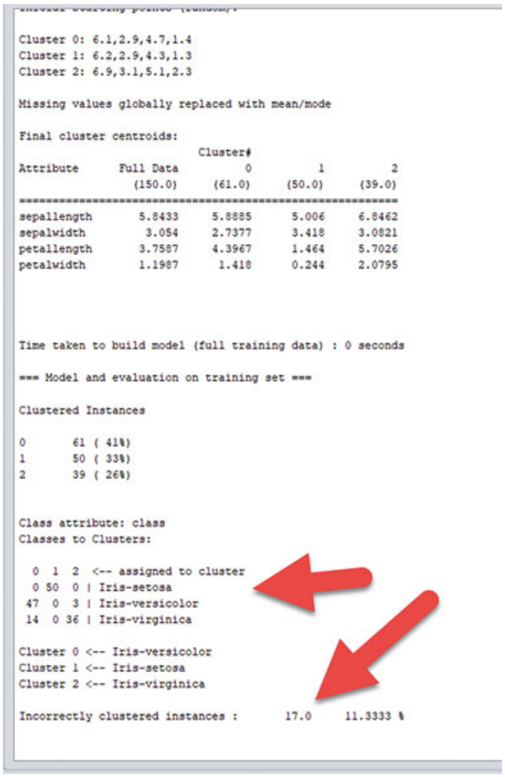
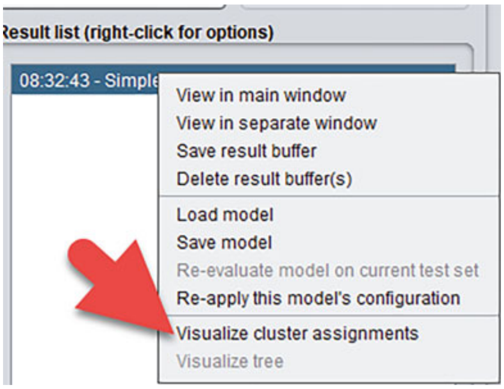


Fig. 12.9 Seventeen instances were incorrectly clustered

```
Cluster 0: 6.1,2.9,4.7,1.4
Cluster 1: 6.2,2.9,4.3,1.3
Cluster 2: 6.9,3.1,5.1,2.3

Missing values globally replaced with mean/mode

Final cluster centroids:
                          Cluster#
Attribute      Full Data         0          1          2
                 (150.0)    (61.0)     (50.0)     (39.0)
=========================================================
sepallength       5.8433    5.8885      5.006     6.8462
sepalwidth         3.054    2.7377      3.418     3.0821
petallength       3.7587    4.3967      1.464     5.7026
petalwidth        1.1987     1.418      0.244     2.0795



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       61 ( 41%)
1       50 ( 33%)
2       39 ( 26%)


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 50  0 | Iris-setosa
 47  0  3 | Iris-versicolor
 14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances :       17.0      11.3333 %
```



Result list (right-click for options)
08:32:43 - Simple
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize cluster assignments
Visualize tree

Fig. 12.10 Visualize cluster assignments

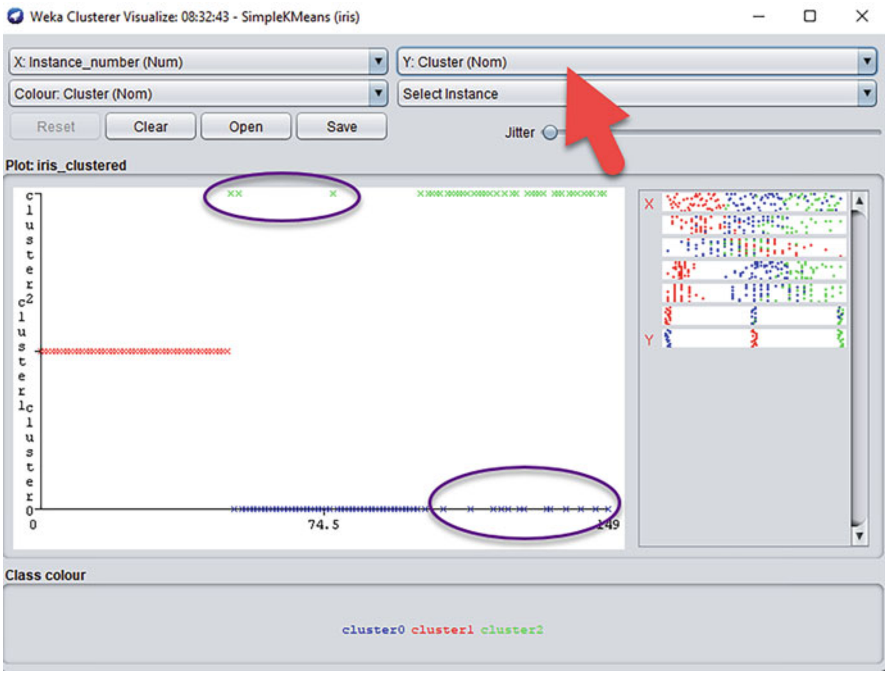
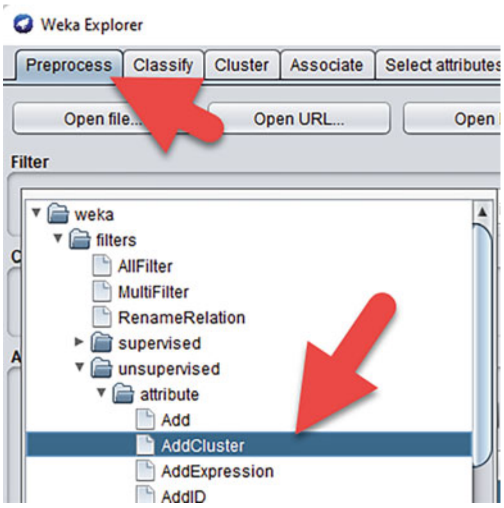
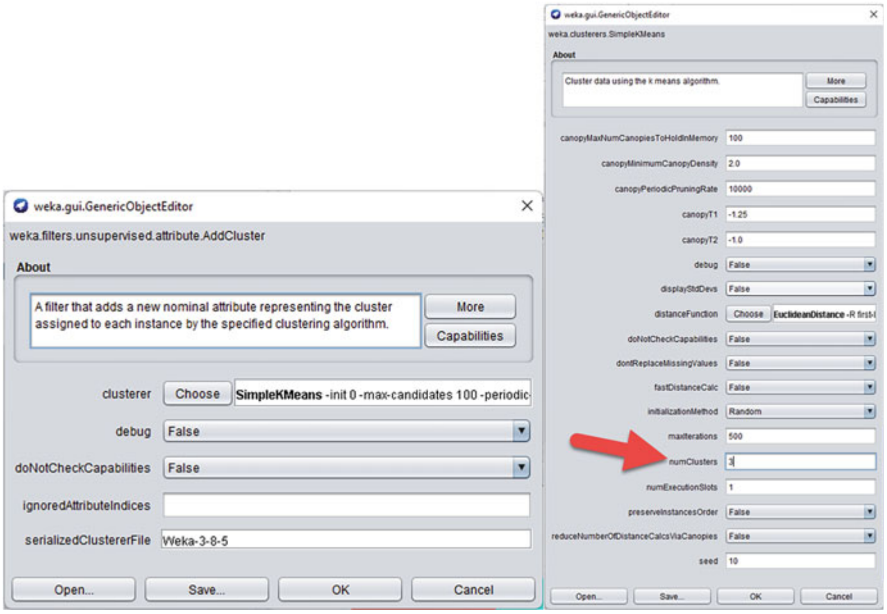


Fig. 12.11 Cluster visualization

Fig. 12.12 AddCluster filter





weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.AddCluster

About

A filter that adds a new nominal attribute representing the cluster assigned to each instance by the specified clustering algorithm.

clusterer	Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-
debug	False
doNotCheckCapabilities	False
ignoredAttributeIndices	
serializedClustererFile	Weka-3-8-5

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose **EuclideanDistance** -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	3
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

Fig. 12.13 The *K*-means parameters in the AddCluster filter

$$\left\| x^{(i)} - \mu_k^{(t-1)} \right\| < \left\| x^{(i)} - \mu_k^{(t-1)} \right\|; j = 1, \ldots, K, j \neq k$$

3. For each group of instances $N_k^{(t-1)}$ in a cluster K, $k = 1, \ldots K$, recompute:

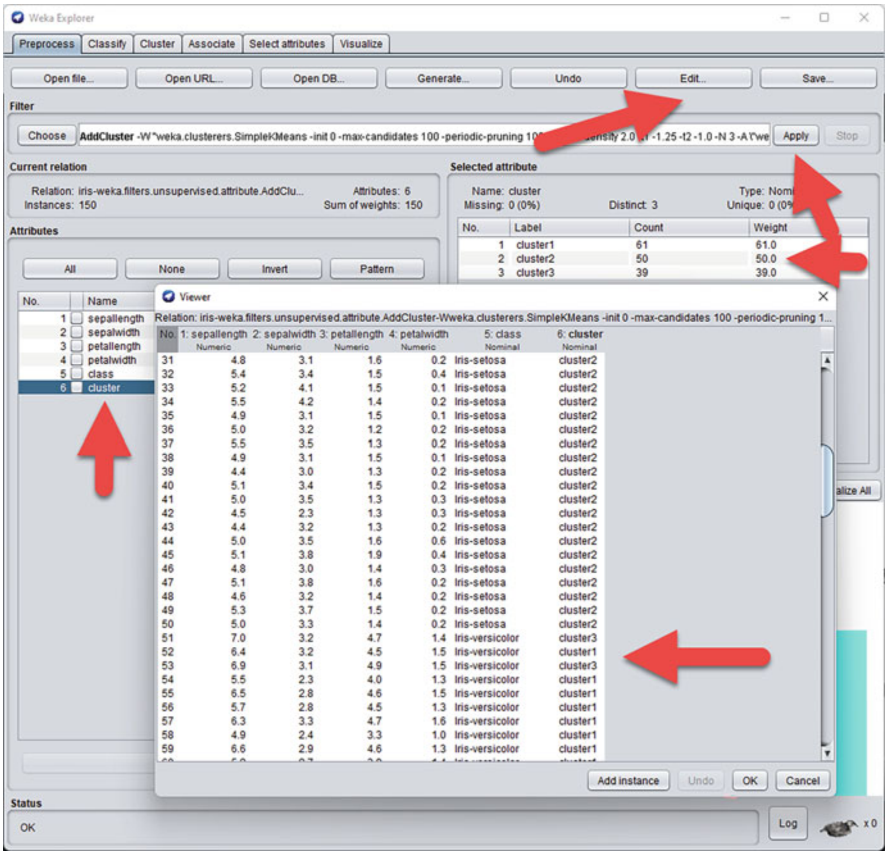
$$\mu_k = \frac{1}{N_k^{(t-1)}} \sum_{i=1}^{N_k^{(t-1)}} x^{(i)}$$

4. Stop when the stopping criteria are satisfied; otherwise, increment t: $t = t + 1$ and repeat from step 2.
5. Return the result $\mu_1^{(t)}$ to $\mu_k^{(t)}$.

12.4 Inertia

Once the learning is one (using .fit() in Python), the algorithm can always assign a new instance to the closest centroid (using .predict()), which is called a hard clustering.

Alternatively to hard clustering, we can perform soft clustering that assigns scores to the new instance, each score is equivalent to the distance of the new instance to one of the centroids (.transform() in Python). In this way, each data instance can be



Filter

AddCluster -W "weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10… …ensity 2.0 … -1.25 -t2 -1.0 -N 3 -A \"we

Current relation

Relation: iris-weka.filters.unsupervised.attribute.AddClu... Attributes: 6
Instances: 150 Sum of weights: 150

Selected attribute

Name: cluster Type: Nomi
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%

No.	Label	Count	Weight
1	cluster1	61	61.0
2	cluster2	50	50.0
3	cluster3	39	39.0

Attributes

No.		Name
1	☐	sepallength
2	☐	sepalwidth
3	☐	petallength
4	☐	petalwidth
5	☐	class
6	☐	cluster

Relation: iris-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 1...

No.	1: sepallength Numeric	2: sepalwidth Numeric	3: petallength Numeric	4: petalwidth Numeric	5: class Nominal	6: cluster Nominal
31	4.8	3.1	1.6	0.2	Iris-setosa	cluster2
32	5.4	3.4	1.5	0.4	Iris-setosa	cluster2
33	5.2	4.1	1.5	0.1	Iris-setosa	cluster2
34	5.5	4.2	1.4	0.2	Iris-setosa	cluster2
35	4.9	3.1	1.5	0.1	Iris-setosa	cluster2
36	5.0	3.2	1.2	0.2	Iris-setosa	cluster2
37	5.5	3.5	1.3	0.2	Iris-setosa	cluster2
38	4.9	3.1	1.5	0.1	Iris-setosa	cluster2
39	4.4	3.0	1.3	0.2	Iris-setosa	cluster2
40	5.1	3.4	1.5	0.2	Iris-setosa	cluster2
41	5.0	3.5	1.3	0.3	Iris-setosa	cluster2
42	4.5	2.3	1.3	0.3	Iris-setosa	cluster2
43	4.4	3.2	1.3	0.2	Iris-setosa	cluster2
44	5.0	3.5	1.6	0.6	Iris-setosa	cluster2
45	5.1	3.8	1.9	0.4	Iris-setosa	cluster2
46	4.8	3.0	1.4	0.3	Iris-setosa	cluster2
47	5.1	3.8	1.6	0.2	Iris-setosa	cluster2
48	4.6	3.2	1.4	0.2	Iris-setosa	cluster2
49	5.3	3.7	1.5	0.2	Iris-setosa	cluster2
50	5.0	3.3	1.4	0.2	Iris-setosa	cluster2
51	7.0	3.2	4.7	1.4	Iris-versicolor	cluster3
52	6.4	3.2	4.5	1.5	Iris-versicolor	cluster1
53	6.9	3.1	4.9	1.5	Iris-versicolor	cluster3
54	5.5	2.3	4.0	1.3	Iris-versicolor	cluster1
55	6.5	2.8	4.6	1.5	Iris-versicolor	cluster1
56	5.7	2.8	4.5	1.3	Iris-versicolor	cluster1
57	6.3	3.3	4.7	1.6	Iris-versicolor	cluster1
58	4.9	2.4	3.3	1.0	Iris-versicolor	cluster1
59	6.6	2.9	4.6	1.3	Iris-versicolor	cluster1

Status

OK

Fig. 12.14 Display dataset values

represented by its position to the *K* clusters, if *K* is lower than the dataset dimension, the result of clustering would be a reduction in the data dimensions; so, *K*-means could be used for dimensionality reduction.

The *K*-means is sensitive to the initial centroids and can provide you with different solutions if you start with different initial centroids. So how to choose the best of many solutions? What is the cost function to minimize? Since this is an unsupervised learning, we do not have labels to measure the performance of the algorithm, what we have seen in the Weka example above was only for learning purposes. However, there is one performance measurement for *K*-means called inertia, that consist of the within-cluster sum-of-squares that measures the sum of squared distances of each instance to its centroid. Inertia measures the internal coherence of the clusters; a low inertia means a better clustering solution. We can run *K*-Means multiple times (in Python this is controlled by *K*-Means *n_init* hyperparameter), and *K*-means will use inertia to find an optimal clustering solution.

12.5 Minibatch *K*-Means

The original *K*-means algorithm was proposed by Stuart Lloyd in 1957 [4]. To accelerate the execution of *K*-Means, a faster new algorithm was proposed in 2003 by Charles Elkan [5]; both version can be set by the *K*-Means *algorithm* hyperparameter in Python, the default being Lloyd's. An even faster version of *K*-Means was proposed in 2010 by David Sculley [6] that uses min-batches of the dataset instead of the full dataset at each iteration, it is implemented under the name MiniBatchKMeans in Python.

12.6 Final Notes: Advantages, Disadvantages, and Best Practices

K-means models are widely used because of their simplicity. They can easily scale and generalize to different data size/shapes and easily adapts to new scenarios. As convergence is key for accuracy and optimal decisions, *K*-means model has shown good convergence results. Choosing the number of centroid (K) is a challenge, and we will see a method to find the optimal K in Sect. 12.10 below.

Like any ML model, *K*-means has its own disadvantages, including the manual choice of K value, which makes it dependent on initial values. One of key issues of *K*-means is the impact of outliers on clusters generation; if the data is not well preprocessed, outliers can have their own clusters. Lastly, highly dimensional data can generate curse of dimensionality problem, thus affecting the convergence of the model. Using feature engineering techniques can mitigate this issue.

When working with *K*-means, consider trying many values for K, as well as many clustering algorithms (e.g., *K*-Means, DBSCAN, and test which setup provides better results. You may also want to take special care to avoid overfitting, mainly you need to increase the value of K to eliminate noises in clusters.

12.7 Key Terms

1. *K*-means
2. Clustering
3. Cluster
4. Centroids
5. Matrix
6. Segmentation
7. Image segmentation
8. Customer segmentation

9. Hard clustering
10. Soft clustering

12.8 Test Your Understanding

1. Give examples of situations from different fields where we can use clustering.
2. How do you decide which cluster a data instance belongs to?
3. How do you decide the centroid of each cluster?
4. How do you determine the centroids of the first clusters?
5. How do you determine the number of clusters *K*?
6. What is the stopping criterion for *K*-means?
7. There is a method called "elbow" that allows us to compute the optimum number of clusters in a dataset. Explore this method; you will be using it in the lab.
8. Cite some of the hyperparameters of *K*-means.
9. What is inertia in *K*-Means? What is it used for?

12.9 Read More

1. Aris, T. A., Nasir, A. S. A., & Mohamed, Z. (2021). A Robust Segmentation of Malaria Parasites Detection using Fast K-Means and Enhanced K-Means Clustering Algorithms. 2021 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 128–133. doi: 10.1109/ICSIPA52582.2021.9576799
2. Armstrong, J. J., Zhu, M., Hirdes, J. P., & Stolee, P. (2012). K-means cluster analysis of rehabilitation service users in the Home Health Care System of Ontario: examining the heterogeneity of a complex geriatric population. Arch Phys Med Rehabil, 93(12), 2198–2205. doi: 10.1016/j.apmr.2012.05.026
3. Fahim, A. (2021). K and starting means for K-means algorithm. Journal of Computational Science, 55(Complete). doi: 10.1016/j.jocs.2021.101445
4. Gesicho, M. B., Were, M. C., & Babic, A. (2021). Evaluating performance of health care facilities at meeting HIV-indicator reporting requirements in Kenya: an application of K-means clustering algorithm. BMC Med Inform Decis Mak, 21(1), 6. doi: 10.1186/s12911-020-01367-9
5. Grant, R. W., McCloskey, J., Hatfield, M., Uratsu, C., Ralston, J. D., Bayliss, E., & Kennedy, C. J. (2020). Use of Latent Class Analysis and K-Means Clustering to Identify Complex Patient Profiles. JAMA Netw Open, 3(12), e2029068. doi: 10.1001/jamanetworkopen.2020.29068
6. Jabi, M., Pedersoli, M., Mitiche, A., & Ayed, I. B. (2021). Deep Clustering: On the Link Between Discriminative Models and K-Means. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(6), 1887–1896. doi: 10.1109/TPAMI.2019.2962683

7. Khan, A. R., Khan, S., Harouni, M., Abbasi, R., Iqbal, S., & Mehmood, Z. (2021). Brain tumor segmentation using K-means clustering and deep learning with synthetic data augmentation for classification. Microsc Res Tech, 84(7), 1389–1399. doi: 10.1002/jemt.23694
8. Kwedlo, W., & Lubowicz, M. (2021). Accelerated K-Means Algorithms for Low-Dimensional Data on Parallel Shared-Memory Systems. IEEE Access, 9, 74286–74301. doi: 10.1109/ACCESS.2021.3080821
9. Li, Y., Zhang, Y., Tang, Q., Huang, W., Jiang, Y., & Xia, S.-T. (2021). t-k-means: A ROBUST AND STABLE K-means VARIANT. ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 3120–3124. doi: 10.1109/ICASSP39728.2021.9414687
10. Sinaga, K. P., Hussain, I., & Yang, M.-S. (2021). Entropy K-Means Clustering With Feature Reduction Under Unknown Number of Clusters. IEEE Access, 9, 67736–67751. doi: 10.1109/ACCESS.2021.3077622
11. Tavallali, P., Tavallali, P., & Singhal, M. (2021). K-means tree: an optimal clustering tree for unsupervised learning. The Journal of Supercomputing, 77(5), 5239–5266. doi: 10.1007/s11227-020-03436-2
12. Yuan, R. (2021). An improved K-means clustering algorithm for global earthquake catalogs and earthquake magnitude prediction. Journal of Seismology, 25(3), 1005–1020. doi: 10.1007/s10950-021-09999-8
13. Zukotynski, K., Black, S. E., Kuo, P. H., Bhan, A., Adamo, S., Scott, C. J. M., . . . Gaudet, V. (2021). Exploratory Assessment of K-means Clustering to Classify 18F-Flutemetamol Brain PET as Positive or Negative. Clin Nucl Med, 46(8), 616–620. doi: 10.1097/rlu.0000000000003668

12.10 Lab

12.10.1 Working Example in Python

Download the dataset using the following link: https://www.kaggle.com/datasets/uciml/adult-census-income

This dataset includes demographics and income level (above or less or equal to 50K). It includes the following columns:

- Age: person's age
- Workclass: work class type the person belongs to
- Fnlwgt: final weight estimate for the specified socioeconomic characteristics of the population
- Education: education level
- Education.num: education level as a number
- Marital.status: person's marital status
- Occupation: person's occupation
- Relationship: person's relationship status
- Race: person's race

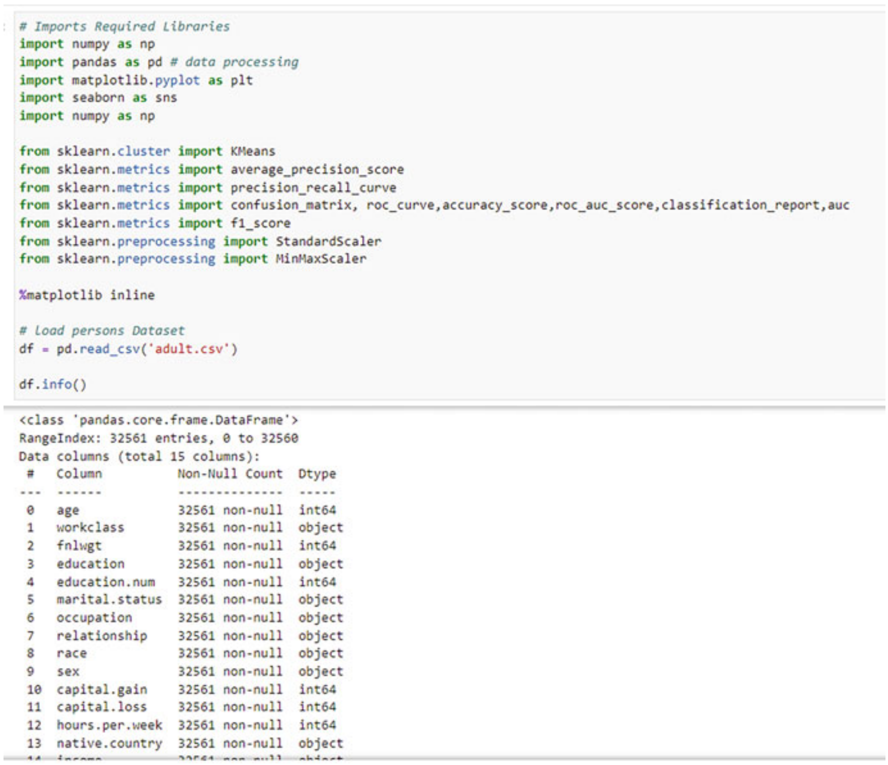
- Sex: person's gender
- Capital.gain: an increase in the person's profit
- Capital.loss: a decrease in the person's profit
- Hours.per.week: number of working hours per week for the person
- Native.country: the person's original country
- Income: the person's salary

12.10.1.1 Load Person's Demographics

After downloading the adult census dataset, load the dataset (Fig. 12.15).

12.10.1.2 Data Visualization and Cleaning

You can explore the data visually; we will content with one plot (Fig. 12.16).



```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.cluster import KMeans
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix, roc_curve,accuracy_score,roc_auc_score,classification_report,auc
from sklearn.metrics import f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

%matplotlib inline

# Load persons Dataset
df = pd.read_csv('adult.csv')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
```

Fig. 12.15 Load persons' census income dataset into pandas

```
#Data Visualisation
f, ax = plt.subplots(figsize=(7, 5))
sns.countplot(x='income', data=df)
_ = plt.title('(Person\'s income <= 50k) vs (Persons Income > 50k)')
_ = plt.xlabel('Person\'s Income')
_ = plt.ylabel('Frequency')
```

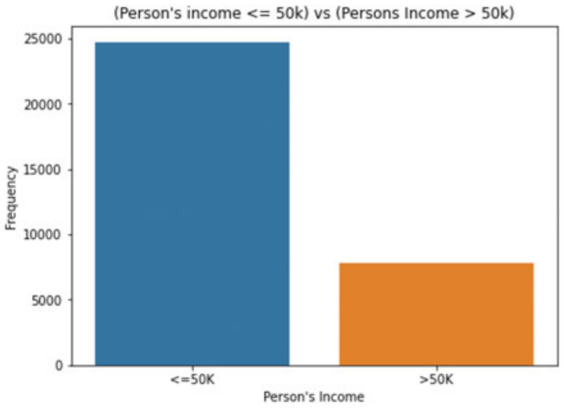


Fig. 12.16 Visualizing persons' census income data in histogram

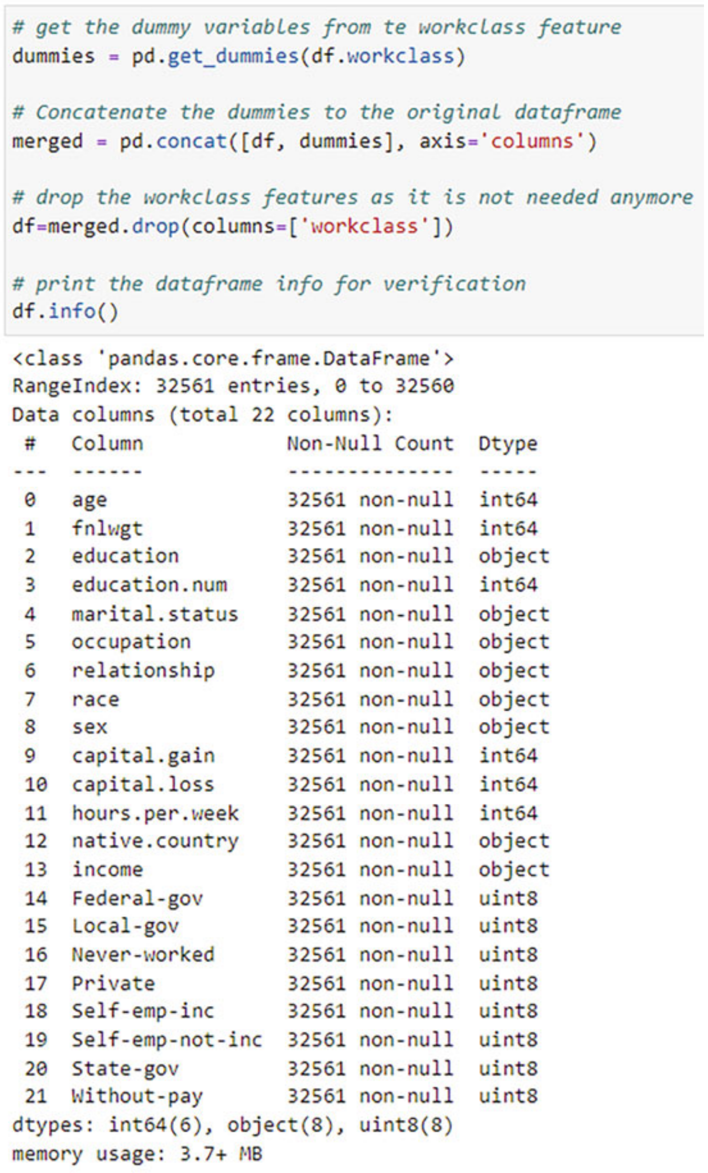
```
df["workclass"] = df["workclass"].replace('?','unknown')
df["occupation"] = df["occupation"].replace('?','unknown')

#drop two irrelevant features
df =df.drop(["native.country"], axis =1)
```

Fig. 12.17 Replacing "?" with "unknown" and dropping the "native.country"

12.10.1.3 Data preprocessing

"Workclass" and "occupation" features that contain interrogation marks, and we will start by replacing the "?" with the word "unknown," while the "native.country" feature is irrelevant for our work and we will drop it altogether (Fig. 12.17).

Many features (workclass, marital.status, occupation, relationship, race, sex) as well as the target (income) are categorical data and hence need one-hot-encoding. The features that are one-hot encoded will be split into many new "dummy" features, if we are working with a linear regression algorithm that would be a problem as the values of any of the features can be deduced if we know the values of all other ones (if all features other than the dropped one are zero, then for sure the dropped feature is 1, and if any one of the other features is 1 then for sure the dropped feature value is zero). This means that one of the features is always correlated with all others. With *K*-means we do not need to do so. We show how to proceed with splitting the categorical features into new features in Fig. 12.18.

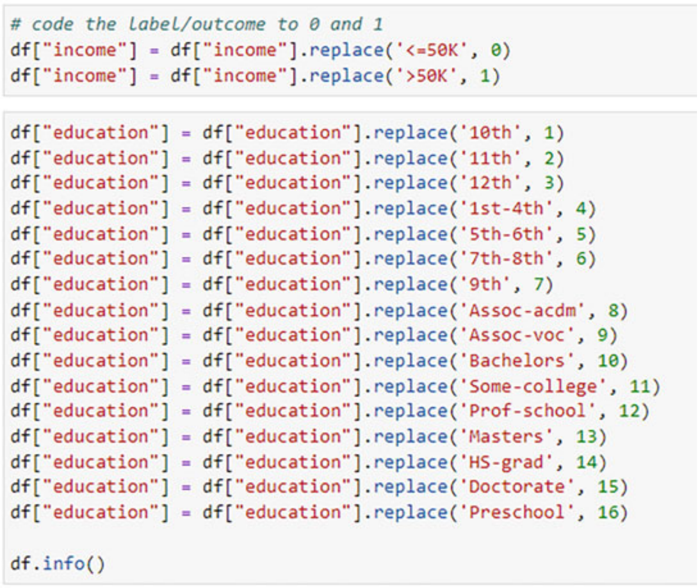


```
# get the dummy variables from te workclass feature
dummies = pd.get_dummies(df.workclass)

# Concatenate the dummies to the original dataframe
merged = pd.concat([df, dummies], axis='columns')

# drop the workclass features as it is not needed anymore
df=merged.drop(columns=['workclass'])

# print the dataframe info for verification
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   fnlwgt          32561 non-null  int64 
 2   education       32561 non-null  object
 3   education.num   32561 non-null  int64 
 4   marital.status  32561 non-null  object
 5   occupation      32561 non-null  object
 6   relationship    32561 non-null  object
 7   race            32561 non-null  object
 8   sex             32561 non-null  object
 9   capital.gain    32561 non-null  int64 
 10  capital.loss    32561 non-null  int64 
 11  hours.per.week  32561 non-null  int64 
 12  native.country  32561 non-null  object
 13  income          32561 non-null  object
 14  Federal-gov     32561 non-null  uint8 
 15  Local-gov       32561 non-null  uint8 
 16  Never-worked    32561 non-null  uint8 
 17  Private         32561 non-null  uint8 
 18  Self-emp-inc    32561 non-null  uint8 
 19  Self-emp-not-inc 32561 non-null  uint8 
 20  State-gov       32561 non-null  uint8 
 21  Without-pay     32561 non-null  uint8 
dtypes: int64(6), object(8), uint8(8)
memory usage: 3.7+ MB
```

Fig. 12.18 One-hot encoding workclass categorical variable

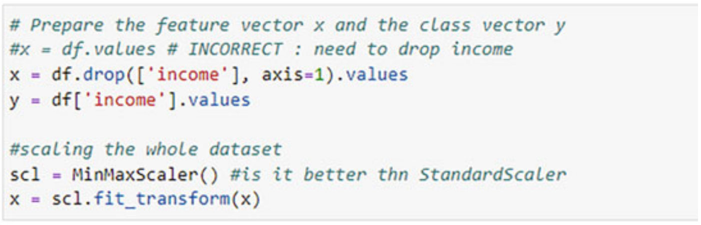
The *education* feature in ordinal in nature, so we will change the string values to numeric ordered values (Fig. 12.19).



```
# code the label/outcome to 0 and 1
df["income"] = df["income"].replace('<=50K', 0)
df["income"] = df["income"].replace('>50K', 1)
```

```
df["education"] = df["education"].replace('10th', 1)
df["education"] = df["education"].replace('11th', 2)
df["education"] = df["education"].replace('12th', 3)
df["education"] = df["education"].replace('1st-4th', 4)
df["education"] = df["education"].replace('5th-6th', 5)
df["education"] = df["education"].replace('7th-8th', 6)
df["education"] = df["education"].replace('9th', 7)
df["education"] = df["education"].replace('Assoc-acdm', 8)
df["education"] = df["education"].replace('Assoc-voc', 9)
df["education"] = df["education"].replace('Bachelors', 10)
df["education"] = df["education"].replace('Some-college', 11)
df["education"] = df["education"].replace('Prof-school', 12)
df["education"] = df["education"].replace('Masters', 13)
df["education"] = df["education"].replace('HS-grad', 14)
df["education"] = df["education"].replace('Doctorate', 15)
df["education"] = df["education"].replace('Preschool', 16)

df.info()
```

Fig. 12.19 Mapping ordinal categorical values into numeric values



```
# Prepare the feature vector x and the class vector y
#x = df.values # INCORRECT : need to drop income
x = df.drop(['income'], axis=1).values
y = df['income'].values

#scaling the whole dataset
scl = MinMaxScaler() #is it better thn StandardScaler
x = scl.fit_transform(x)
```

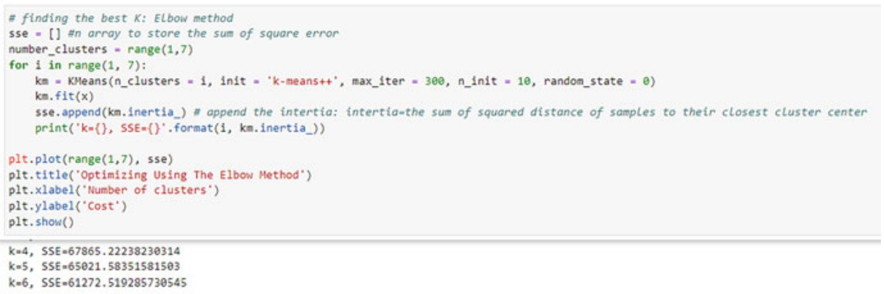
Fig. 12.20 Preparing the data for analysis

12.10.1.4 Choosing Features and Scaling Data

The feature and the target vectors are prepared and data is normalized (check the results if you do not normalize the data, how would the clustering be affected) (Fig. 12.20).

12.10.1.5 Finding the Best *K* for the *K*-Means Model

To find the best *K* for the *K*-means we will proceed manually using a method called the "the elbow" method and also we will sue the usual grid search cross-validation method.



```
# finding the best K: Elbow method
sse = [] #n array to store the sum of square error
number_clusters = range(1,7)
for i in range(1, 7):
    km = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    km.fit(x)
    sse.append(km.inertia_) # append the intertia: intertia=the sum of squared distance of samples to their closest cluster center
    print('k={}, SSE={}'.format(i, km.inertia_))

plt.plot(range(1,7), sse)
plt.title('Optimizing Using The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Cost')
plt.show()
```

```
k=4, SSE=67865.22238230314
k=5, SSE=65021.58351581503
k=6, SSE=61272.519285730545
```

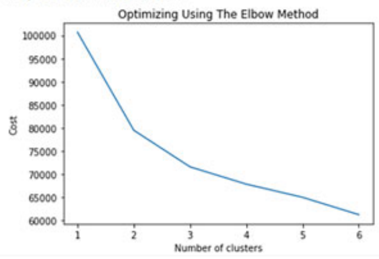
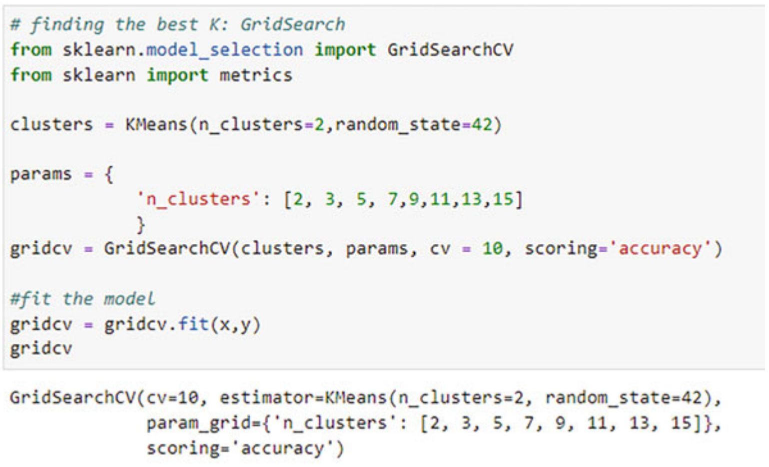


Fig. 12.21 Finding the optimal *K* using elbow method



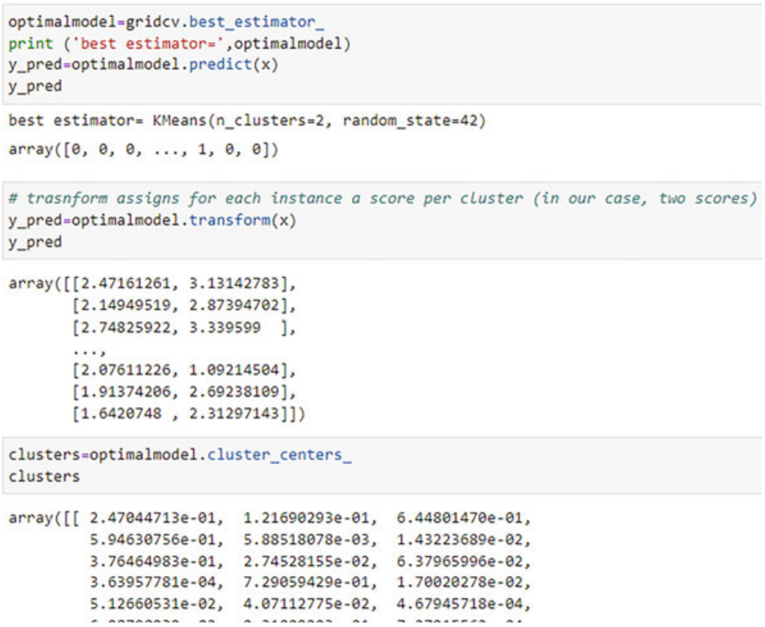
```
# finding the best K: GridSearch
from sklearn.model_selection import GridSearchCV
from sklearn import metrics

clusters = KMeans(n_clusters=2,random_state=42)

params = {
            'n_clusters': [2, 3, 5, 7,9,11,13,15]
            }
gridcv = GridSearchCV(clusters, params, cv = 10, scoring='accuracy')

#fit the model
gridcv = gridcv.fit(x,y)
gridcv
```

```
GridSearchCV(cv=10, estimator=KMeans(n_clusters=2, random_state=42),
             param_grid={'n_clusters': [2, 3, 5, 7, 9, 11, 13, 15]},
             scoring='accuracy')
```

Fig. 12.22 Optimizing *K*-Means hyperparameters using grid search cross-validation

The elbow method consists of drawing for each possible *K* the inertia cost function. (Fig. 12.21), we can notice that cost declines significantly at the beginning from $K = 1$ to $K = 2$, afterwards the decline is less pronounced. It seems like $K = 2$ is the optimal number of clusters. This indeed reflects the actual data, where we have two clusters: people who earn more than 50k and others who earn 50K or less.

Using grid search cross-validation, we can also find the *K* for the optimal model (Fig. 12.22). It is important to note that the GridSearch varied the *K* parameter between 2 and 15 and resulted in $K = 2$ for the optimal *K*-Means.



```
optimalmodel=gridcv.best_estimator_
print ('best estimator=',optimalmodel)
y_pred=optimalmodel.predict(x)
y_pred
```

```
best estimator= KMeans(n_clusters=2, random_state=42)
array([0, 0, 0, ..., 1, 0, 0])
```

```
# trasnform assigns for each instance a score per cluster (in our case, two scores)
y_pred=optimalmodel.transform(x)
y_pred
```

```
array([[2.47161261, 3.13142783],
       [2.14949519, 2.87394702],
       [2.74825922, 3.339599  ],
       ...,
       [2.07611226, 1.09214504],
       [1.91374206, 2.69238109],
       [1.6420748 , 2.31297143]])
```

```
clusters=optimalmodel.cluster_centers_
clusters
```

```
array([[ 2.47044713e-01,  1.21690293e-01,  6.44801470e-01,
         5.94630756e-01,  5.88518078e-03,  1.43223689e-02,
         3.76464983e-01,  2.74528155e-02,  6.37965996e-02,
         3.63957781e-04,  7.29059429e-01,  1.70020278e-02,
         5.12660531e-02,  4.07112775e-02,  4.67945718e-04,
```

Fig. 12.23 Optimizing *K*-means model using grid search cross-validation

Then you can predict the clusters of the feature vector *x* using the optimal model found by GridSearch. As mentioned above the .predict() will perform hard clustering, and you can display for each data instance the cluster to which it is assigned. However, .transform() will show perform soft clustering and, in our case, assigns for each datapoint two scores representing the distance between the instance and the two centroids. Finally, .labels_ and .cluster_centers_ allow you to display the labels of each point as well as the coordinates of clusters' centers (Fig. 12.23).

You can think of plotting the datapoints and the centers. You can try that in Sect. 12.10.2.

12.10.2 Do It Yourself

12.10.2.1 The Iris Dataset Revisited

In this section, create a *K*-means model using different values for *K* (try $K = 2, 5, 10, 15, 20$, and 25) using the iris dataset.

You can download the Iris dataset using the following code

```
import numpy as np
import pandas as pd
from sklearn import datasets
iris =datasets.load_iris()
```

Note the variable “iris” is an array and not a data frame; you should be able by now to know how to convert an array to a data frame, but that is not necessary. To learn more about other available data set, click on the following link: https://scikit-learn.org/stable/datasets.html.

You can always now the features names using “feature_names” and the target name using “target_names.”

```
iris.feature_names
iris.target_names
```

If you want to convert data to a data frame you can write the following code

```
df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
columns= iris['feature_names'] + ['class'])
df.info() # display the data frame information
df # display the first few rows
```

Note: other than the elbow method explained above, there is the silhouette score that allows you to uncover the best K value. Read about the silhouette_score to choose the best K for the Isis dataset.

12.10.2.2 *K*-Means for Dimension Reduction

Download the digits dataset using the following code

```
from sklearn.datasets import load_digits
x, y = load_digits(return_X_y=True)
```

1. Phase 1: basic multiclass classification using logistic regression
   (a) Split the data into training and testing dataset.
   (b) Use logistic regression for multiclass classification (i.e., multi_class=“ovr”) and consider max_iter=5000.
   (c) Fit the model to the training dataset.
   (d) Compute the algorithm score using the testing dataset.

2. Phase 2: seek enhancement using *K*-Means for preprocessing
    (a) Create a pipeline for *K*-Means followed by logistic regression. Since the digits are handwritten and can be present in many ways, choose the number of clusters for *K*-Means much larger than 10, try 50. The logistic regression parameters are the same as in phase 1.
    (b) Fit the pipeline on the training dataset.
    (c) Compute the pipeline score no the testing dataset.
    (d) Compared to the previous score, was this one better or worse? Discuss the possible reasons behind the new score.
3. Phase 3: search for an optimal *K*-Means and logistic regression.
    (a) Find through GridSearch the optimal model for the pipeline. Finetune only one hyperparameter for *K*-Mean: the number of clusters; vary is between 2 and 100. Note that the execution might take around 20 min depending on your computer configuration.
    (b) Fit the pipeline and check the new score.
    (c) Compared to the previous score, was this one better or worse? Discuss the possible reasons behind the new score.
    (d) What was the best parameter found by GridSearch?

12.10.3 Do More Yourself

Create *K*-means models to solve the problems presented by the following datasets:

- https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities
- https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter
- https://archive.ics.uci.edu/ml/datasets/YouTube+Multiview+Video+Games+Dataset

References

1. A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly Media, Sebastopol, CA, 2019)
2. Y. Liu, *Python Machine Learning by Example: Build Intelligent Systems Using Python, TensorFlow 2, PyTorch, and Scikit-Learn, 3rd Edition (Kindle Edition)* (Packt, 2020)
3. M. Gopal, *Applied Machine Learning* (McGraw-Hill Education, New York, 2018)
4. S.P. Lloyd, Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982). https://doi.org/10.1109/TIT.1982.1056489
5. C. Elkan, Using the triangle inequality to accelerate k-means. Presented at the Proceedings of the Twentieth International Conference on International Conference on Machine Learning, Washington, DC, USA (2003)
6. D. Sculley, Web-scale k-means clustering. Presented at the Proceedings of the 19th International Conference on World Wide Web, Raleigh, North Carolina, USA (2010) [Online]. Available: https://doi.org/10.1145/1772690.1772862

Chapter 13
Support Vector Machine

13.1 The Problem

The more the dimensions of a feature space, the more is the computing power needed to classify. Support vector machines (SVMs) main advantages are (1) their effectiveness in a high-dimensional space and in cases where the number of dimensions is higher than the number of instances in the dataset, and (2) their low use of memory and hence their memory efficiency.

The aim of the SVM algorithm is to find the best hyperplane (a line in a two-dimensions space, a plane in a three-dimension space) that divides a dataset into two (or more) classes.

To understand how SVM works, we will take a binary classification problem (Fig. 13.1). Since there could be many lines that can separate the two classes (Fig. 13.1 left), SVM looks for the instances in the datasets (points on the graph) that are closest to the dividing line. The lines passing by these points are called the support vectors. The chosen optimal classification line is the one that maximized the distance between the two support vectors. This is called a maximal margin classification.

Of course, the instances may not that perfectly separable by a line, we need to find a way to classify determine how much should we relax the constraint related to maximizing the margin. This is called a soft margin classification.

The other problem to solve is when the classes are not linearly separable, in this case we need a non-straight line to separate the instances. In SVM, this is done using a kernel. A linear kernel allows to separate linearly separable classes, a polynomial kernel allows the use of a curved line to separate the classes and a radial kernel uses a radial-based function (RBF) to solve complex separations, for example, using a polygon in a two-dimension space).

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_13

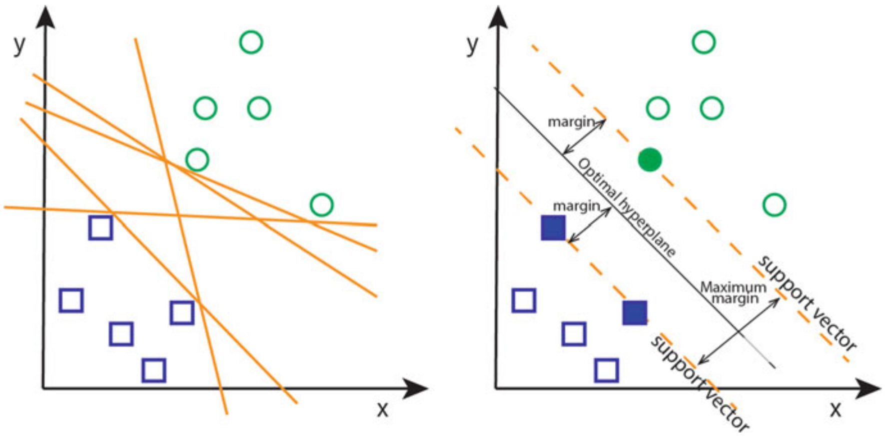


Fig. 13.1 Two datasets green (circle) and blue (square) to be classified. Each point has two features *x* and *y*

13.2 The Algorithm

SVMs are a collection of similar supervised learning algorithms that are used for classification and regression [1–4]. The most effective way to grasp the fundamentals of support vector machines and how they function is to use a simple example (Fig. 13.1). Consider the following scenario: we have two tags, one each of green (circle shape) and blue (square shape), and our data contains two characteristics, *x,* and *y*. We are looking for a classifier that, when given a pair of (x, y) coordinates, outputs whether the pair is red or blue in color. On a plane, we plot the training data that has previously been labeled:

When given these data points, a support vector machine will produce the hyperplane (in two dimensions, a hyperplane is simply a line) that will optimally divide the tags. The hyperplane is the decision border. In 2D, each side of the line will be considered a class (i.e., blue class and green class).

But, more specifically, what is the finest hyperplane? It is the one that optimizes the margins from both tags in the case of SVM. The hyperplane (remember, it is a line in this case) with the greatest distance to the nearest element of each tag is known as the maximum distance hyperplane.

The SVM's goal is to find the best hyperplane (or decision boundary) [5] that divides two different classes while also maximizing the distance between data points from both classes. There could be several hyperplanes to divide the two classes; our aim is to find the hyperplane that is at the greatest distance between data points from both classes (i.e., the greatest margin) [6]. Maximizing the margin distance allows subsequent data points to be categorized with more certainty.

Obviously, the number of features dictates the hyperplane's dimension; in Fig. 13.2, we have two features *x* and *y*, the hyperplane was a straight line [5]. If

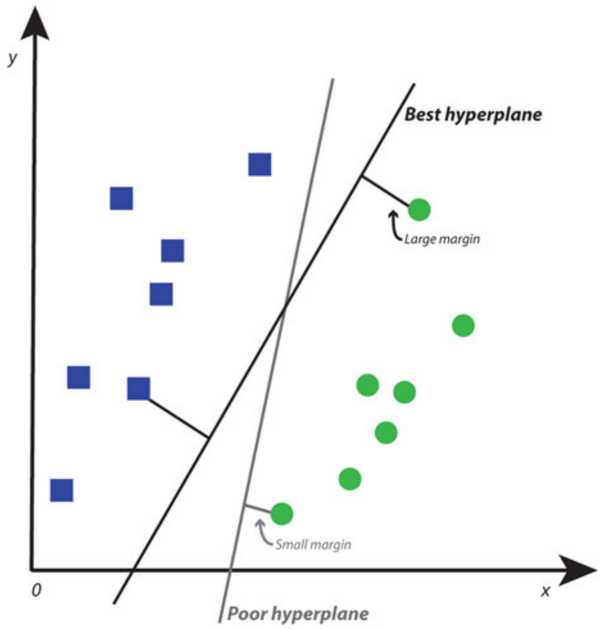


Fig. 13.2 The hyperplane (remember, it is a line in this case) with the greatest distance to the nearest element of each tag

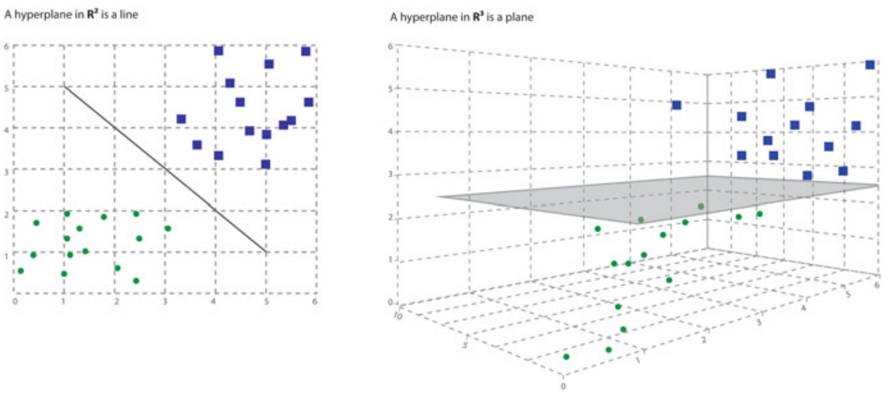


Fig. 13.3 A line hyperplane in a 2D space (left) vs. a two-dimensional hyperplane in a 3D space (right)

the number of features is three, then the hyperplane becomes a 2D plane. Beyond three features, we cannot visualize the hyperplane (Fig. 13.3).

Support vector machines are widely used in machine learning research all around the world, particularly in the United States. When SVMs were used in a handwriting recognition test, they gained popularity since they achieved performance equivalent to that of complex neural networks with elaborated features when employing pixel maps as input [2, 7].

13.2.1 Important Concepts

Support Vectors Support vectors are the data points that are closest to the hyperplane and are used to calculate the distance between them. With the aid of these data points, a dividing line will be drawn between them. It is possible to demonstrate that the optimal hyperplane is derived from the function class with the lowest "capacity" = number of independent features/parameters that can be twiddled [8]. In other words, they are the data points that are closest to a decision surface (or hyperplane). They are also the data points that are most difficult to classify. They have a direct bearing on the optimal location of the decision surface.

Hyperplane As we can see in the diagrams above, a hyperplane is a decision plane or space that is partitioned between a collection of objects belonging to distinct classes. In two dimensions, the hyperplane can be represented by the following equation. This is identical to the equation of affine combination; however, the bias b has been included in this case [9].

$$\beta_1 x_1 + \beta_2 x_2 + b$$

For d-dimensional space, we may generalize this and express it in vectorized form.

$$\begin{aligned} h(x) &= \beta_1 x_1 + \cdots + \beta_d x_d + b \\ &= \left(\sum_{i=1}^{d} \beta_i x_i \right) + b \\ &= \beta^T x + b \end{aligned}$$

For any point $X = (x_1, \ldots, x_d)$, if $h(X) = 0$, then X lies on the hyperplane; otherwise $h(X) < 0$ or $h(X) > 0$, which implies that X falls to one side of the hyperplane. If we now make a very significant assumption about the coefficient weight vector β and assume that x_1 and x_2 are two random locations that lie on the hyperplane, we may write:

$$h(x_1) = \beta^T x_1 + b = 0$$

$$h(x_2) = \beta^T x_2 + b = 0$$

Hence,

$$\beta^T x_1 + b = \beta^T x_2 + b$$

and

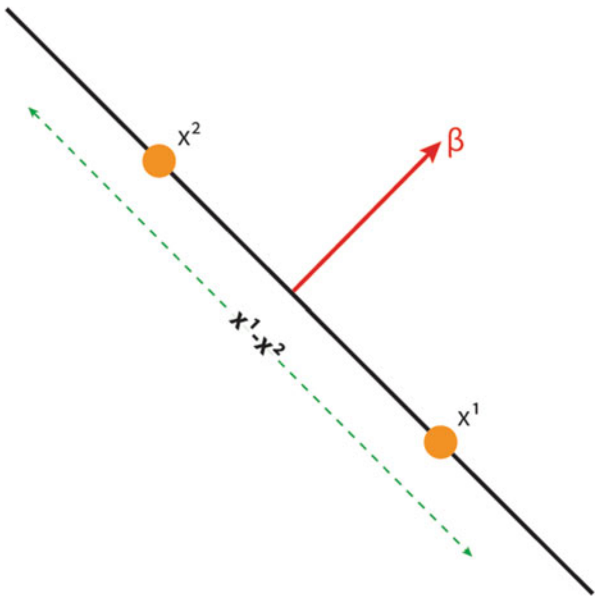


Fig. 13.4 The weight vector beta points in the direction that is normal to the hyperplane of the weight vector

$$\beta^T(x_1 - x_2) = 0$$

If the dot product of two vectors is 0, we know that the vectors are orthogonal to one another, and vice versa. The weight vector β in this case is orthogonal to $(x_1 - x_2)$. Being that $(x_1 - x_2)$ is located on the hyperplane, it follows that the weight vector β is also orthogonal to the hyperplane. That is to say, the weight vector beta points in the direction that is normal to the hyperplane of the weight vector. When the hyperplane is shifted in d-dimensional space, this is expressed as a bias ($\boldsymbol{b}$) [9] (Fig. 13.4).

13.2.2 Margin

The distance between two lines drawn through the closest data points of distinct classifications can be described as a margin. The minimal distance (normal distance) between each observation and a specific separating hyperplane can be used to establish the margin between two observations. See how we may utilize the margin to determine the best hyperplane for our situation. It may be computed by taking the perpendicular distance between the line and the support vectors and dividing it by two.

A large margin is seen as a good margin, while a small margin is regarded as a bad margin in business. The size of the margin determines the confidence level of the classifier; as a result, the largest possible margin should be used. Let us choose two

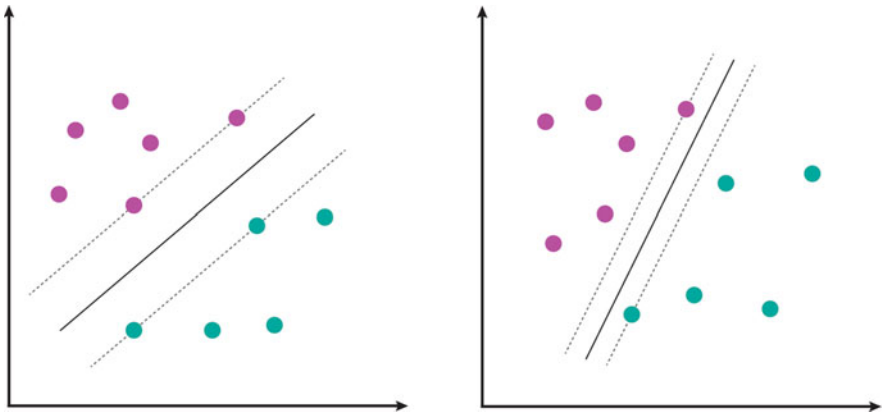


Fig. 13.5 Large (left) vs. small (right) margin

hyperplanes based on their distance from the center (Fig. 13.5). The one to the left has a significantly larger margin than the one to the right, and as a result, the first hyperplane is more optimal than the second one.

We may conclude that in the maximal margin classifier, in order to categorize the data, we will utilize a separation hyperplane that is the greatest (maximum) and smallest (minimum) distance away from the observations in order to classify the data. Let us keep in mind that the margin will still be used to pick the ideal separating hyperplane. Furthermore, Jana margins are divided into two categories: functional margin and geographic margin [9]. They are both summed up below.

13.2.2.1 Functional Margin

To define the theoretical side of the margin, the term "functional margin" is employed. In the presence of a training example (x_i, y_i), the functional margin of (β, b) with regard to the training example will be as follows:

$$y_i\left(\beta^T X_i + b\right) = \widehat{\gamma}_i$$

As opposed to just specifying that the number is larger than 0, we have established a value for the margin by using γ. Thus, the below requirements may be established:

$$\text{if } y_i = 1,\text{then } \widehat{\gamma}_i > 0$$

$$\text{if } y_i = 0,\text{then } \widehat{\gamma}_i = 0$$

But there is a problem with the functional margin, which is that its value is reliant on the values of β and b. The equation of the hyperplane remains the same when β

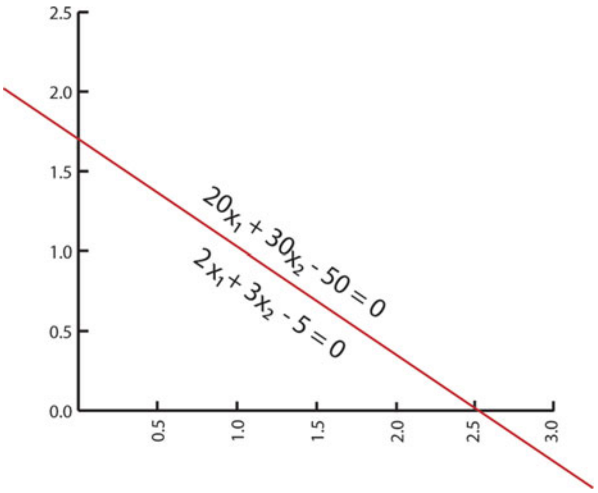


Fig. 13.6 The same hyperplane representing two equations

and b are scaled (multiplied by some scaler s), but the margin increases. If you plot the following two equations, they will both represent the same hyperplane, but in this case, the width of the margins will change between the two equations (Fig. 13.6).

$$2x_1 + 3x_2 – 5 = 0$$

$$20x_1 + 30x_2 – 50 = 0$$

13.2.2.2 Geometric Margin

Let us make considerations regarding the visuals below (Fig. 13.7):

Along with the vector w, the decision boundary corresponding to (w, b) is depicted in Fig. 13.7. It should be noted that w is orthogonal (i.e., at 90°) to the separation hyperplane.

You must convince yourself that this is a fact. Consider the point A, which represents the input $x^{(i)}$ of a training example with the label $y^{(i)} = 1$, as represented by the point B. The line segment AB determines the distance between it and the decision border, denoted by $\gamma^{(i)}$.

The value of $\gamma^{(i)}$ can be determined in several ways. To explain it more clearly, the unit-length vector $w/\|w\|$ indicates that w is moving in the same direction. Since A represents $x^{(i)}$, we may conclude that the point B is given by $x^{(i)} - \gamma^{(i)} \times w/\|w\|$. The problem is that this point is located on the decision boundary, and all points x on the decision boundary are satisfied by the equation $w^T x + b = 0$; therefore:

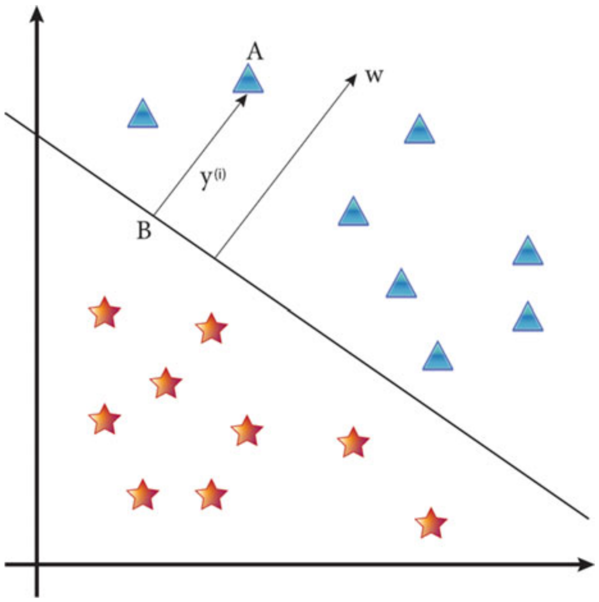


Fig. 13.7 Two datasets separated by a hyperplane with weigh vector w and a decision boundary (w, b)

$$w^T\left(x^{(i)}-\gamma^{(i)}\frac{w}{\|w\|}\right)+b=0$$

And solving for $\gamma^{(i)}$ yields:

$$\gamma^{(i)}=\frac{w^Tx^{(i)}+b}{\|w\|}=\left(\frac{w}{\|w\|}\right)^Tx^{(i)}+\frac{b}{\|w\|}$$

Specifically, this was calculated for the situation of a positive training example at A in Fig. 13.7, in which being on the "positive" side of the decision border is advantageous.

Furthermore, we define the geometric margin of (w, b) regarding a training example $(x^{(i)}, y^{(i)})$ as follows:

$$\gamma^{(i)}=y^{(i)}\left(\left(\frac{w}{\|w\|}\right)^Tx^{(i)}+\frac{b}{\|w\|}\right)$$

It is important to note that if $\|w\| = 1$, then the functional margin equals the geometric margin—this provides a means of connecting these two disparate ideas of margin together. As a result of this property, the geometric margin is invariant to rescaling of the parameters; that is, if we substitute two values for w and two values for b, the geometric margin remains unchanged. Furthermore, because of this invariance to scaling of the parameters, we can apply any arbitrary scaling constraint to w without producing important changes [6]; for example, we can demand that $\|w\| = 1$, or that $|w_1 + b| + |w_2| = 2$, and any other constraint can be satisfied by just rescaling w and the parameters; however, this is not recommended.

13.2.3 Types of Support Vector Machines

Support vector machines are generally classified into only two types. They are both detailed below:

13.2.3.1 Linear Support Vector Machine

This type only works with data that can be divided into two categories by a single perfect line, in which case the dataset is considered linearly separable, and the linear SVM classifier is used. This is further divided into two types and is visually displayed below.

13.2.3.2 Soft Margin Classifier

A soft margin classifier is an SVM that where the threshold is allowed to make an tolerable number of misclassifications, while allowing new data instances to be classified correctly [10]. The famous cross-validation technique can be used to determine the best classification (Fig. 13.8).

In a real-world scenario, it is unlikely that a perfectly distinct line would be drawn between the data points included inside the space [11]. Furthermore, we might have a curved decision boundary. It is possible to have a hyperplane that precisely separates the data; however, this may not be desired if the data contains noise. Jakkula agrees it is preferable for the smooth border to disregard a small number of data points rather than being curved or going in loops around outliers [2].

The assumption that the dataset is perfectly linearly separable has been made up to this point. This assumption does not hold up to scrutiny when dealing with a real-world dataset. As a result, let us look at a slightly more challenging scenario. The linear SVM is still in the works; however, this time, some of the classes overlap in such a way that a perfect separation is unattainable, yet the data is still linearly separable [9]. Consider a dataset with two dimensions shown in Fig. 13.9. There are two primary options available:

- When a single outlier occurs, the decision boundary might be pushed significantly, resulting in an extremely tight margin of safety.
- The data may not be separable using a straight line, even when a linear decision boundary can correctly categorize the target classes (no clear boundary).

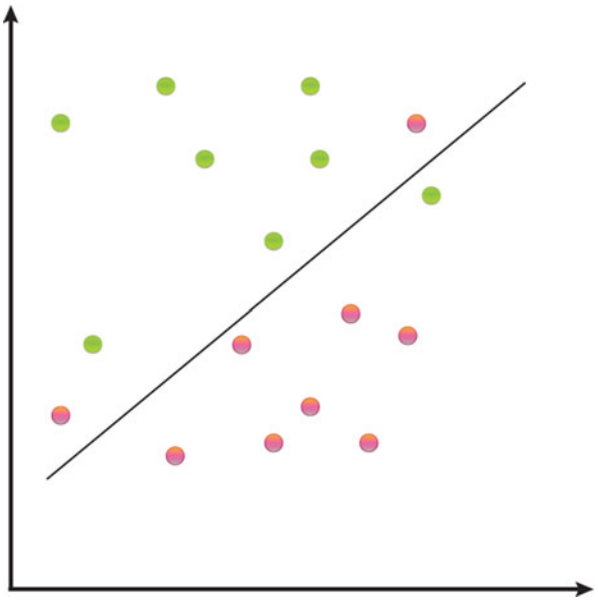


Fig. 13.8 Linear SVM—soft margin classifier

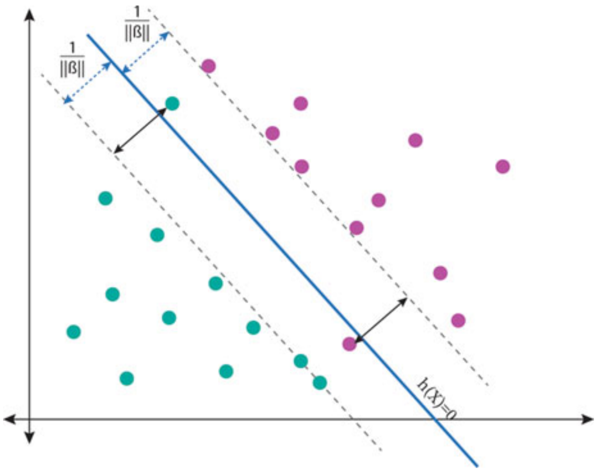


Fig. 13.9 A dataset with two dimensions

In other words, the hard margin classifier that is visualized in Fig. 13.10 would not operate owing to the inequality restriction $y_i(\beta^T x_i + 1) \geq 1$.

13.2.3.2.1 Hard Margin Classifier

As previously stated, the idea of SVM is to execute an affine discrimination of observations with the greatest amount of margin possible, that is, to identify an

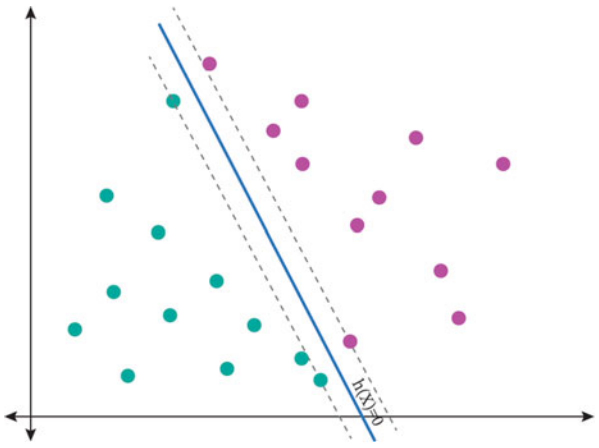


Fig. 13.10 Linear SVM—hard margin classifier

element ($w \in X$) with the lowest norm and the greatest possible real value b, such that the value of $y_i((w_i, x_i) + b) \geq 1$ is the same for all i. But to do so, we must first solve the quadratic programming issue described below:

$$\min_{w,b} < w, \ w >$$
$$\text{subject to} \quad y_{i(<w_i, \ x_i>+b)} \geq 1, \qquad 1 \leq i \leq N$$

The classification rule that relates to (w, b) is simply referred to as $f(x) = \sin((w, x) + b)$. In this circumstance (which is referred to as the hard margin SVM), we require that the rule have zero error on the learning set (Fig. 13.10).

13.2.3.3 Nonlinear Support Vector Machine

This classifier is used for nonlinearly separated data, which means that if a dataset cannot be classified using a straight line, it is considered nonlinear data, and the classifier used is the nonlinear SVM classifier. A graphical representation is shown below (Fig. 13.11) [11].

A mathematical example of nonlinear support vector machines is described below:

$$K(x, y) = (x.y + 1)^p$$
$$\left\{ - \|x - y\|^2 / 2\sigma^2 \right\}$$
$$K(x, y) = \tan h(kx.y\text{–}\delta)$$

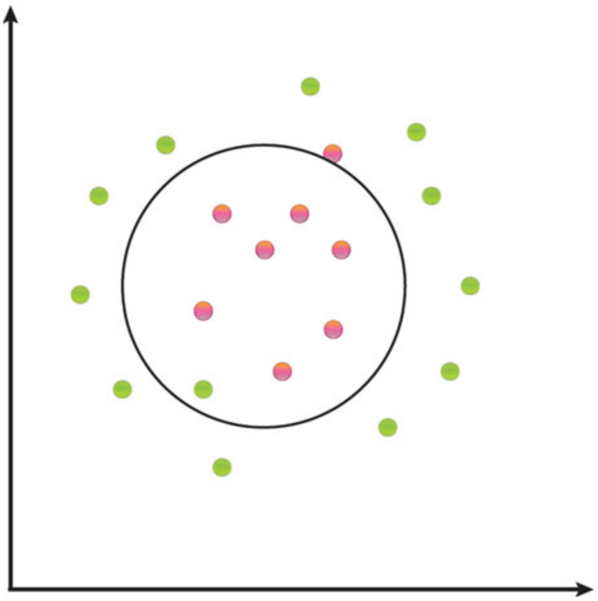


Fig. 13.11 Representation of a nonlinear Support Vector Machine

The first equation is a polynomial, while the second equation is a radial basis function (Gaussians), and the third is a sigmoid (neural net activation function) [8]. Some of these are visualized just below.

13.2.4 Classification

SVM is a data classification approach that is beneficial. The employment of neural networks, despite the fact that they are regarded as more user-friendly than SVM, might result in disappointing outcomes at times [12]. Training and testing data for classification tasks typically comprise a small number of data examples [2]. A target value and a number of characteristics are contained inside each instance of the training set. The SVM model allows us to predicts target values of the instances in the testing dataset [13].

Supervised learning may be seen in the classification process of SVM. Known labels assist in determining whether or not the system is operating in the proper manner. This information either points to a desired reaction, thus verifying the correctness of the system, or it may be utilized to assist the system in learning to behave in the appropriate manner. In SVM classification [2, 13], one phase is the identification of classes that are tightly related to the classes that are already recognized. This is referred to as feature selection, or feature extraction in technical terms. Even when the prediction of unknown samples is not required, the combination of feature selection with SVM classification might be beneficial. In order to separate the classes, they can be utilized to identify key sets that are engaged in the procedures that distinguish them.

13.2.5 Regression

Through the use of an alternate loss function, it is possible to apply SVMs to regression situations [13, 14]. It is necessary to modify the loss function in order to add a distance measure. There are two types of regression: linear and nonlinear. Linear models are composed mostly of the loss functions listed below: e-intensive loss functions, quadratic loss functions, and the Huber loss function.

It is common for nonlinear models to be required for data modeling challenges, much as it is for classification difficulties. A technique similar to the nonlinear SVC approach, nonlinear mapping, may be used to map the data into a high-dimensional feature space, where linear regression can then be done on the information.

When it comes to dealing with the curse of dimensionality [15], the kernel technique is once again used. Considerations based on past knowledge of the problem and the distribution of the noise are taken into account while employing the regression approach. The robust loss function of Huber has been proved to be a good substitute in the absence of such information [13].

13.2.6 Tuning Parameters

13.2.6.1 Regularization

For each training dataset, the support vector machine is instructed on the optimal degree of misclassification to avoid by adjusting the regularization parameter, which is also known as the C parameter in Python's sklearn module. When larger numbers are used for the C parameter in a support vector machine, the optimizer will automatically choose a hyperplane margin that is smaller if it is successful in separating and classifying all the training data points during the optimization process. Alternately, when dealing with extremely small values, the algorithm will seek a larger margin for the hyperplane to separate, even if the hyperplane misclassifies some data points.

13.2.6.2 Gamma

An influence on a single training data sample is repeated several times using this tuning parameter. Lower gamma values reflect distance from the hyperplane, whereas higher gamma values show proximity to the hyperplane. Data points with both low and high gamma (far from and near to the hyperplane, respectively) are included in the computation of the separation line.

13.2.6.3 Margins

The margin is the final but not the least important characteristic. It is also a critical parameter for fine-tuning and a vital characteristic of a support vector machine classifier. The margin, as previously established, is the distance between the line and the data points from the classes. When using the support vector approach, it is critical to have a good and appropriate margin. When the difference between the two groups of data is higher than one standard deviation, it is a good margin. A sufficient margin ensures that the individual data points remain inside their respective classes and do not cross over into another class.

13.2.7 Kernel

When using SVM, a kernel turns the input data space into the desired format. SVM employs the kernel trick to turn a low-dimensional input space into a higher-dimensional space. For the uninitiated, this means that kernel adds new dimensions to an issue that would otherwise be impossible to separate.

Generally speaking, it is most useful in nonlinear separation situations. Simply said, the kernel performs a number of incredibly sophisticated data transformations before determining the best method of separating the data depending on the labels or outputs that have been established.

As a result, SVM gains higher scalability, adaptability, and accuracy. Kernels utilized by SVM include those listed below.

13.2.7.1 Linear Kernel

All observations can be combined in this way. Here is the equation for a linear kernel:

$$K(x, x_i) = \text{sum}(x \times x_i)$$

The product between two vectors, x and x_i, may be represented as the total of the products of each pair of input values in the formula above.

13.2.7.2 Polynomial Kernel

Curved or nonlinear input spaces can be distinguished using this generalized linear kernel. A polynomial kernel may be expressed using the following formula:

$$K(x, x_i) = 1 + \mathrm{sum}(x \times x_i)^d$$

Here, *d* is the degree of the polynomial, which we must manually enter into the learning algorithm.

13.2.7.3 Radial Basis Function (RBF) Kernel

When used in SVM classification, the RBF kernel transforms the input space into an infinitum of three-dimensional spaces. It is widely used in SVM classification tasks. The following formula provides a mathematical explanation:

$$K(x, x_i) = \exp\left(-\mathrm{gamma} \times \mathrm{sum}\left(x - {x_i}^2\right)\right)$$

In this case, gamma is between 0 and 1. We must explicitly define it in the learning algorithm; the default value of gamma is 0.1, which is the industry-accepted default.

13.3 Advantages, Disadvantages, and Best Practices

Nonetheless, the SVM's greatest benefit is the kernel technique, which allows it to classify extremely nonlinear situations by creating complicated boundary shapes, rather than by using simple classification rules [16]. These qualities have enabled the SVM to find widespread use in a variety of disciplines throughout the course of the previous few years. SVM has been utilized for fault diagnostics [17], quality improvement [18], and quality assessment [19].

SVMs have been used in the field of computer vision for a variety of tasks, such as face detection, picture categorization, hand gesture recognition, and background removal. SVMs have been utilized in finance for a variety of purposes, including financial time series forecasting and the prediction of bankruptcy. Aside from hydrology, other uses of SVM include forecasting of solar and wind resources, prediction of atmospheric temperature, bioinformatics, speaker recognition, agricultural forecasting, and electrical design. The quality of the datasets, on the other hand, has an impact on the performance of basic support vector machines (SVMs). Typically, noise may be found in real-world datasets. Noise is defined as anything that obscures the link between the attributes of an instance and the characteristics of its class. The noise might express itself as feature-noise (or feature uncertainty), which has the effect of altering the observed value of the corresponding feature. Certainly, uncertainties may arise as a result of the constraints of observational material, as well as the restricted resources available for data collection, storage, transformation, and analysis.

Overall, the training of SVM is quite simple, which is one of its key advantages. It scales rather well to large amounts of high-dimensional data, and the trade-off between classifier complexity and error may be carefully adjusted. It is necessary to have a good kernel function, which is one of the weaknesses [13, 20]. Overall, it is a good idea to standardize to avoid the optimal hyperplane being influenced by the scale of the features.

13.4 Key Terms

1. Statistical learning theory
2. Hyperplane
3. Structural risk minimization
4. Support vectors
5. Coefficient weight vector
6. Functional margin
7. Geometric margin
8. Soft margin classifier
9. Hard margin classifier
10. Curse of dimensionality
11. Gamma

13.5 Test Your Understanding

1. What are support vectors?
2. How do support vector machines function?
3. When have we achieved a maximum distance hyperplane?
4. What is a hyperplane? Highlight its purpose(s).
5. Explain the structural risk management concept.
6. Describe an optimization theory-based learning algorithm.
7. What is the difference between the functional margin and the geometric margin?
8. List the two types of support vectors.
9. Distinguish between soft and hard margin classifiers.
10. Describe the maximal margin classifier.
11. Why do SVMs use the kernel trick?
12. Highlight the tuning parameters of a support vector machine.

13.6 Read More

1. K. R. Song et al., "Resting-state connectome-based support-vector-machine predictive modeling of internet gaming disorder," (in eng), *Addict Biol,* vol. 26, no. 4, p. e12969, Jul 2021, doi: 10.1111/adb.12969.
2. A. Fleury, N. Noury, and M. Vacher, "Supervised classification of Activities of Daily Living in Health Smart Homes using SVM," (in eng), *Annu Int Conf IEEE Eng Med Biol Soc,* vol. 2009, pp. 6099–102, 2009, doi: 10.1109/iembs.2009.5334931.
3. L. Squarcina et al., "Automatic classification of autism spectrum disorder in children using cortical thickness and support vector machine," (in eng), *Brain Behav,* vol. 11, no. 8, p. e2238, Aug 2021, doi: 10.1002/brb3.2238.
4. P. Unnikrishnan, D. K. Kumar, S. Poosapadi Arjunan, H. Kumar, P. Mitchell, and R. Kawasaki, "Development of Health Parameter Model for Risk Prediction of CVD Using SVM," (in eng), *Comput Math Methods Med,* vol. 2016, p. 3016245, 2016, doi: 10.1155/2016/3016245.
5. A. Fleury, M. Vacher, and N. Noury, "SVM-based multimodal classification of activities of daily living in Health Smart Homes: sensors, algorithms, and first experimental results," (in eng), *IEEE Trans Inf Technol Biomed,* vol. 14, no. 2, pp. 274–83, Mar 2010, doi: 10.1109/titb.2009.2037317.
6. S. Wang et al., "Abnormal regional homogeneity as a potential imaging biomarker for adolescent-onset schizophrenia: A resting-state fMRI study and support vector machine analysis," (in eng), *Schizophr Res,* vol. 192, pp. 179–184, Feb 2018, doi: 10.1016/j.schres.2017.05.038.
7. S. Wang, G. Wang, H. Lv, R. Wu, J. Zhao, and W. Guo, "Abnormal regional homogeneity as potential imaging biomarker for psychosis risk syndrome: a resting-state fMRI study and support vector machine analysis," (in eng), *Sci Rep,* vol. 6, p. 27619, Jun 8 2016, doi: 10.1038/srep27619.
8. C. Cavaliere et al., "Computer-Aided Diagnosis of Multiple Sclerosis Using a Support Vector Machine and Optical Coherence Tomography Features," (in eng), *Sensors (Basel),* vol. 19, no. 23, Dec 3 2019, doi: 10.3390/s19235323.
9. M. Kang, S. Shin, G. Zhang, J. Jung, and Y. T. Kim, "Mental Stress Classification Based on a Support Vector Machine and Naive Bayes Using Electrocardiogram Signals," (in eng), *Sensors (Basel),* vol. 21, no. 23, Nov 27 2021, doi: 10.3390/s21237916.
10. G. Cohen and R. Meyer, "Optimal asymmetrical SVM using pattern search. A health care application," (in eng), *Stud Health Technol Inform,* vol. 169, pp. 554–8, 2011

13.7 Lab

13.7.1 *Working Example in Python*

In this section, we will create a support vector machine classifier model, test it, and optimize it. Start by downloading Iris dataset using the following link: https://www.kaggle.com/datasets/arshid/iris-flower-dataset. Alternatively, you can use the following code to load the dataset directly into your code.

```
iris = datasets.load_iris()
x = iris.data[:, :4]
y = iris.target
```

The iris dataset describes the properties of flowers. It includes three iris species within 50 samples. This dataset includes the following columns:

- Petal length: petal length for the Iris
- Petal width: petal width for the Iris
- Sepal length: sepal length for the Iris
- Sepal width: sepal width for the Iris
- Species: class of the iris (there are three species in the dataset)

13.7.1.1 Loading Iris Dataset

Start by importing the required libraries and loading the dataset (Fig. 13.12).

13.7.1.1.1 Visualize Iris Dataset

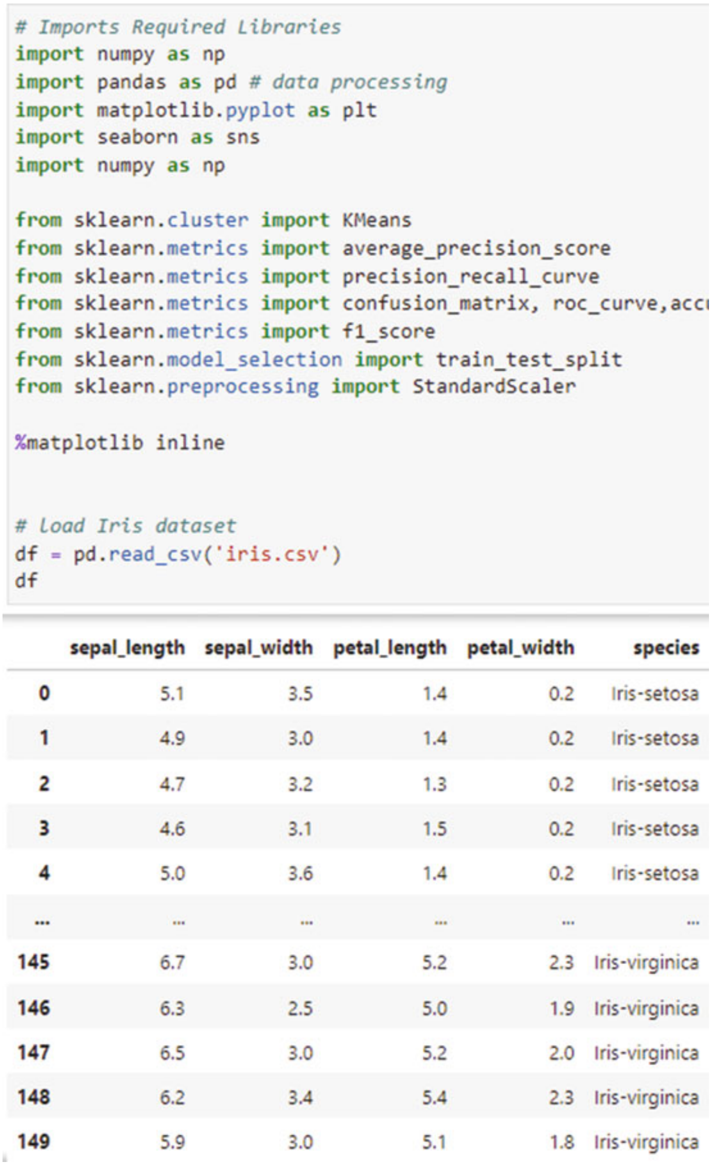
Visualizing the dataset can be done in many ways, one is demonstrated in Fig. 13.13.

13.7.1.2 Preprocess and Scale Data

We need to replace the categorical target with numeric values, split the dataset into training and testing datasets, and standardize both sets (Fig. 13.14).

13.7.1.3 Dimension Reduction

We can now create a support vector model (SVM) using an RBF kernel and C=100. The dimension of the feature matrix is low (i.e., 4); however, for illustration purposes, we will use the Principal Component analysis (PCA) to reduce the number

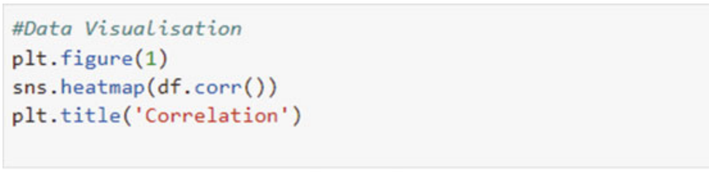


```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.cluster import KMeans
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix, roc_curve,acc
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

%matplotlib inline


# Load Iris dataset
df = pd.read_csv('iris.csv')
df
```

```
     sepal_length  sepal_width  petal_length  petal_width         species
  0           5.1          3.5           1.4          0.2     Iris-setosa
  1           4.9          3.0           1.4          0.2     Iris-setosa
  2           4.7          3.2           1.3          0.2     Iris-setosa
  3           4.6          3.1           1.5          0.2     Iris-setosa
  4           5.0          3.6           1.4          0.2     Iris-setosa
...           ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica
```

Fig. 13.12 Loading the Iris dataset into pandas

of features to 2. Once PCA is create, we apply it to the x_tran and x_test. A two-dimension feature matrix will allow us to plot the SVM results in two dimensions which clarifies the end result. Instead of using PCA, and for illustration purposes, you could have opted to choose two of the four dimensions, such as sepal width and petal width (Fig. 13.15).



```
#Data Visualisation
plt.figure(1)
sns.heatmap(df.corr())
plt.title('Correlation')
```

```
Text(0.5, 1.0, 'Correlation')
```

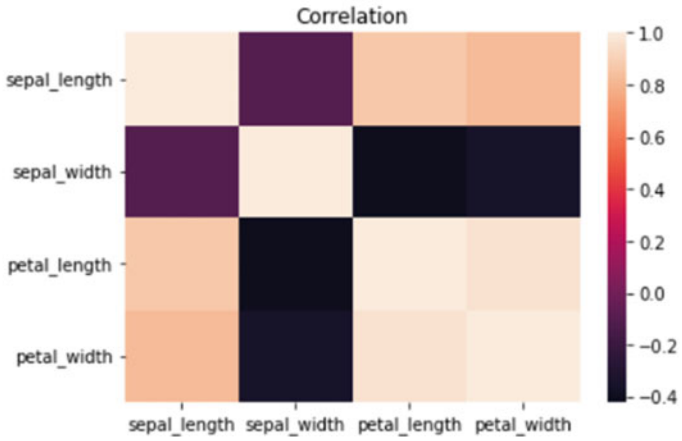
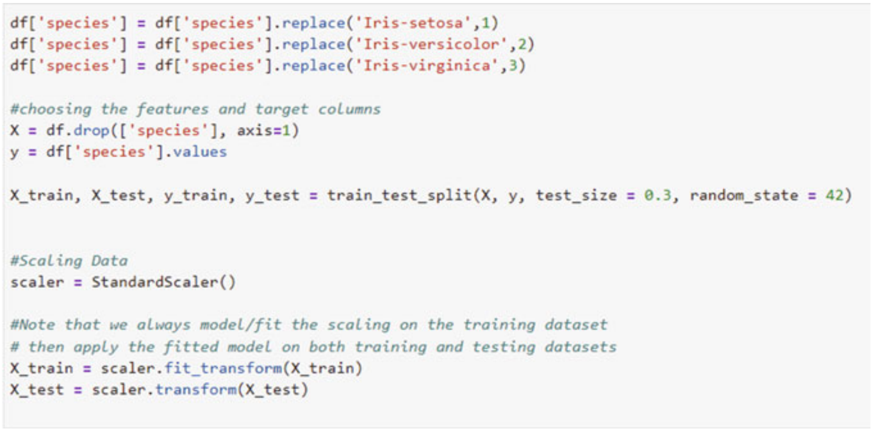
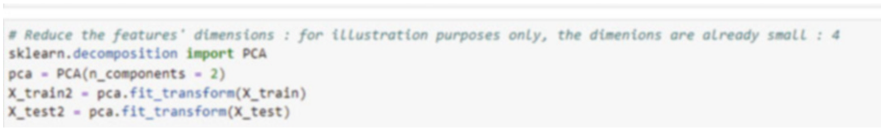


Fig. 13.13 Visualizing iris dataset



```
df['species'] = df['species'].replace('Iris-setosa',1)
df['species'] = df['species'].replace('Iris-versicolor',2)
df['species'] = df['species'].replace('Iris-virginica',3)

#choosing the features and target columns
X = df.drop(['species'], axis=1)
y = df['species'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)


#Scaling Data
scaler = StandardScaler()

#Note that we always model/fit the scaling on the training dataset
# then apply the fitted model on both training and testing datasets
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Fig. 13.14 Preprocess and scale iris dataset



```
# Reduce the features' dimensions : for illustration purposes only, the dimenions are already small : 4
sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
X_test2 = pca.fit_transform(X_test)
```

Fig. 13.15 Creating support vector machine

13.7.1.4 Hyperparameter Tuning and Performance Measurements

Using GridSearch, we can now seek hyperparameter tuning for the SVC. One the optimal model is found, we fit it to the training dataset and make predictions on the testing dataset to display the classification report and the AUC (Fig. 13.16).

Optionally, we can display the confusion matrix (Fig. 13.17).

13.7.1.5 Plot the Decision Boundaries

Finally, we can plot the decision boundaries between classes (Figs. 13.18 and 13.19).

13.7.2 Do It Yourself

13.7.2.1 The Iris Dataset Revisited

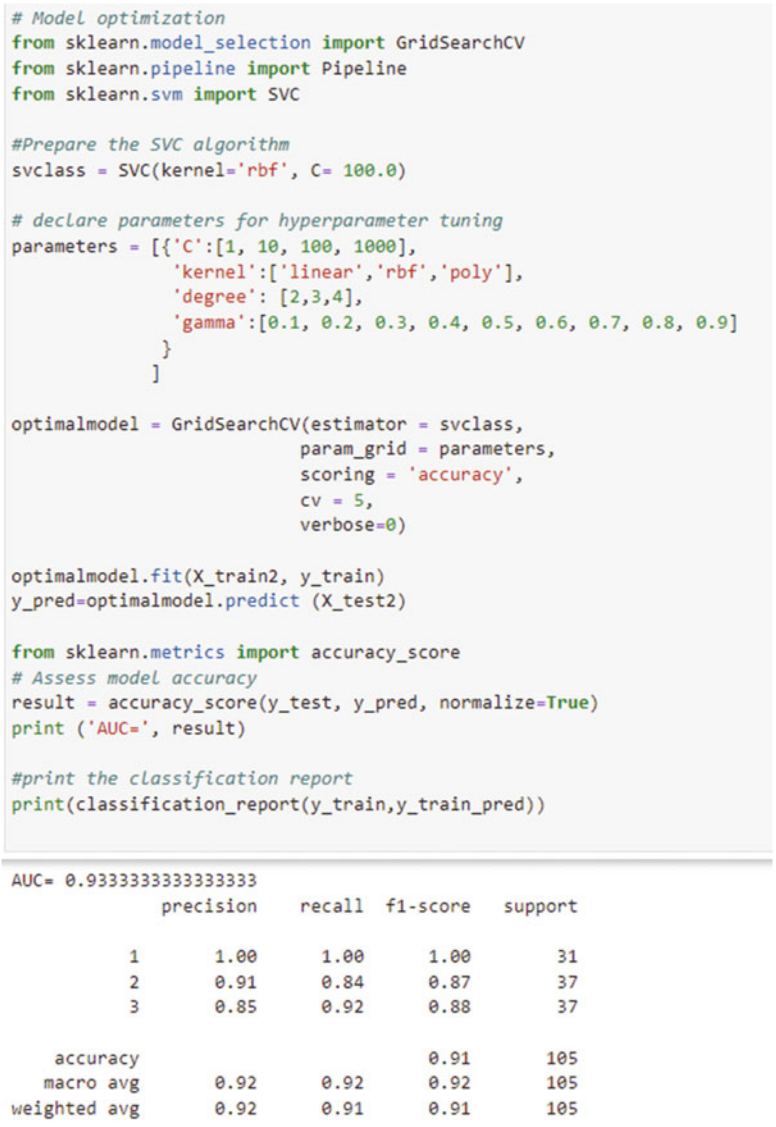
In Sect. 13.7 above, we applied PCA to reduce the number of features to 2

1. Instead of PCA choose to drop the petal length and sepal length and check how the MVC performance chance.
2. Instead of PCA choose to drop the petal width and sepal width and check how the MVC performance chance.
3. Which one lead to better results? Can you know in advance what is more likely to lead to good performance by looking at the pair plots? We did not display the pair plots, so display them and check visually to see if you can gain an insight about the better choice.

13.7.2.2 Breast Cancer

Use the breast cancer dataset that can download from the following link: https://www.kaggle.com/code/buddhiniw/breast-cancer-prediction/data.

1. Create an SVM model to solve this classification problem.
2. Now that you know several classifiers, create a lab where you use three classifiers including an SVM and compare their performance. Conclude by choosing the best performing classifier. Always give a rational for your choices.



```
# Model optimization
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

#Prepare the SVC algorithm
svclass = SVC(kernel='rbf', C= 100.0)

# declare parameters for hyperparameter tuning
parameters = [{'C':[1, 10, 100, 1000],
               'kernel':['linear','rbf','poly'],
               'degree': [2,3,4],
               'gamma':[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
              }
             ]

optimalmodel = GridSearchCV(estimator = svclass,
                            param_grid = parameters,
                            scoring = 'accuracy',
                            cv = 5,
                            verbose=0)

optimalmodel.fit(X_train2, y_train)
y_pred=optimalmodel.predict (X_test2)

from sklearn.metrics import accuracy_score
# Assess model accuracy
result = accuracy_score(y_test, y_pred, normalize=True)
print ('AUC=', result)

#print the classification report
print(classification_report(y_train,y_train_pred))
```

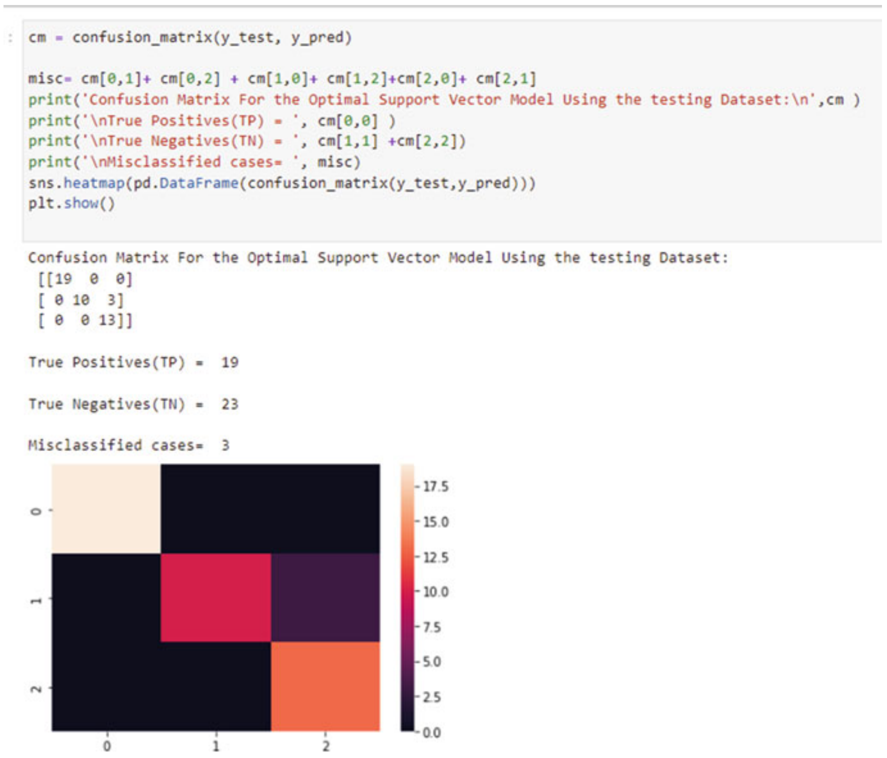
```
AUC= 0.9333333333333333
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        31
           2       0.91      0.84      0.87        37
           3       0.85      0.92      0.88        37

    accuracy                           0.91       105
   macro avg       0.92      0.92      0.92       105
weighted avg       0.92      0.91      0.91       105
```

Fig. 13.16 Decision plot for iris species

13.7.2.3 Wine Classification

Use the wine dataset that can be downloaded from the following link: https://archive.ics.uci.edu/ml/datasets/wine

You can also contemplate using the built "load_wine"



```
cm = confusion_matrix(y_test, y_pred)

misc= cm[0,1]+ cm[0,2] + cm[1,0]+ cm[1,2]+cm[2,0]+ cm[2,1]
print('Confusion Matrix For the Optimal Support Vector Model Using the testing Dataset:\n',cm )
print('\nTrue Positives(TP) = ', cm[0,0] )
print('\nTrue Negatives(TN) = ', cm[1,1] +cm[2,2])
print('\nMisclassified cases= ', misc)
sns.heatmap(pd.DataFrame(confusion_matrix(y_test,y_pred)))
plt.show()
```

```
Confusion Matrix For the Optimal Support Vector Model Using the testing Dataset:
 [[19  0  0]
 [ 0 10  3]
 [ 0  0 13]]

True Positives(TP) =  19

True Negatives(TN) =  23

Misclassified cases=  3
```

Fig. 13.17 Confusion matrix resulting from the optimal model

```
from sklearn.datasets import load_wine
wine_data= sklearn.datasets.load_wine()
```

There are three types of wine, so this is a multi-class problem. Create a model to predict the wine the using SVM (hint: use the SVC with decision_functino_shape =‘ovr’ and degree=3).

13.7.2.4 Face Recognition

You might need to install the Python image library called pillow: pip install pillow

1. Load the images dataset using the following code

```
from sklearn.datasets import fetch_lfw_people
data = fetch_lfw_people(min_faces_per_person=50) # read only
those with 50 images or more
```

```
from mlxtend.plotting import plot_decision_regions # you need to: pip install mlxtend
#plot the classification
pldec = plot_decision_regions(X_test2, y_test, clf=optimalmodel, legend=0)

# Plot decision regions
# Adding axes annotations
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('SVM classification result on the Iris dataset')
handles, labels = pldec.get_legend_handles_labels()
pldec.legend(handles,
          ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
           framealpha=0.3, scatterpoints=1)

plt.show()
```

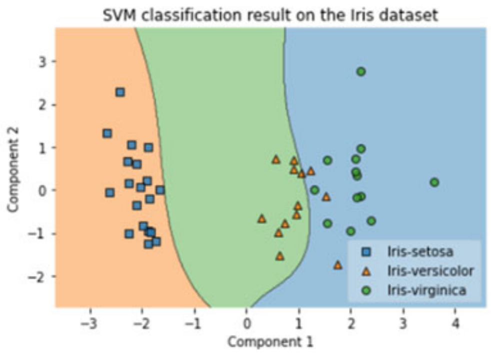


Fig. 13.18 Plotting the decision boundaries in 2D

```
# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=100.0  using testing dataset: {0:0.4f} %'.
      format(accuracy_score(y_test, y_test_pred)*100))
print(classification_report(y_test,y_test_pred))
```

```
Model accuracy score with rbf kernel and C=100.0  using testing dataset: 86.6667 %
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        19
           2       1.00      0.54      0.70        13
           3       0.68      1.00      0.81        13

    accuracy                           0.87        45
   macro avg       0.89      0.85      0.84        45
weighted avg       0.91      0.87      0.86        45
```

Fig. 13.19 Calculating accuracy, recall, and precision metrics for SVM using testing dataset

2. Display on the screen the number of instances in each target class
3. Do you notice any imbalance in the classes? Clarify.
4. Try to plot few images on the screen

5. Split the dataset into training and testing datasets
6. Create an SVM variable (if there were imbalance in classes, then use a class_weight parameter)
7. Grid search for the optimal model
8. Display the best parameters and the best model
9. Fit the optimal model on the training dataset
10. Use the fitted model to predict using the testing dataset
11. Display a classification report (use classification_report from Sklearn)
12. Let's take one further step. PCA might help you boost the model's performance. Apply PCA before rerunning the SVM grid search and check if the performance is better.

13.7.2.5 SVM Regressor: Predict House Prices with SVR

Support vector machine can be used not only as classifiers but as regressors too. Create a support vector model regressor to predict house prices, using the housing dataset that can be downloaded using the following link: https://www.kaggle.com/datasets/huyngohoang/housingcsv.

The housing dataset provides the sale price of houses across the United States. This dataset includes the following columns:

- Avg. Area Income: the average income in the area where the house is located.
- Avg. Area House Age: the average house age in the area where the house is located.
- Avg. Area Number of Rooms: the average number of rooms for a house in the area where the house is located.
- Avg. Area Number of Bedrooms: the average number of bedrooms for a house in the area where it is located.
- Area Population: the population in the area where the house is located.
- Price: the sale price of the house.
- Address: the house address.

Hint: explore the SVR following this link: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

13.7.2.6 SVM Regressor: Predict Diabetes with SVR

1. Load the diabetes dataset using the following code:

```
SVM Regressor: Predict house prices with SVR
```

2. Store at least 30 samples in a testing dataset
3. Proceed with GridSearch using the following values
    (a) alpha: [1e-7, 1e-6, 1e-5, 1e-4]
    (b) penalty: [None, ‘l2’]
    (c) eta0: [0.03, 0.04, 0.05, 0.1]
    (d) max_itr: [500, 1000]
4. After fitting the optimal model, display the best parameters and best estimstor
5. Make prediction and display the optimal model performance (i.e., MAE, MSE, R2)

13.7.2.7 Unsupervised SVM

Support vector machine can be used not only as supervised but unsupervised too.

Create an unsupervised support vector machine regressor to predict house prices, using the housing dataset.

Hint: explore the One class SVM following this link: https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

13.7.3 Do More Yourself

Use the following datasets and create linear and nonlinear support vector machines to solve the classification problems associated with these datasets. Also, try several algorithms to solve each and choose the best model.

- https://www.kaggle.com/datasets/paultimothymooney/stock-market-data
- https://www.kaggle.com/code/startupsci/titanic-data-science-solutions/data
- https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction
- https://www.kaggle.com/datasets/elikplim/forest-fires-data-set

References

1. S. Osowski, K. Siwek, T. Markiewicz, MLP and SVM networks—a comparative study, in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004, 11-11 June 2004,* pp. 37–40 (2004).
2. V.R. Jakkula, Tutorial on support vector machine (SVM). Sch. EECS Wash. State Uni. **37**(2.5), 3 (2011). [Online]. Available: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf.
3. S. Huang, N. Cai, P.P. Pacheco, S. Narrandes, Y. Wang, W. Xu, Applications of support vector machine (SVM) learning in cancer genomics (in eng). Cancer Genomics Proteomics **15**(1), 41–51 (2018). https://doi.org/10.21873/cgp.20063
4. V.N. Vapnik, Pattern recognition using generalized portrait method. Autom. Remote Control **24**, 774–780 (1963)

5. R. Gandhi, Support vector machine—introduction to machine learning algorithms. *Medium* (2022). https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed 11 Mar 2022.
6. A. Ng, Part V, Support Vector Machines. See stanford.edu (2022). https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf. Accessed 14 Mar 2022.
7. W. Zouhri, L. Homri, J.-Y. Dantan, Handling the impact of feature uncertainties on SVM: A robust approach based on Sobol sensitivity analysis. Expert Syst. Appl. **189**, 115691 (2022). https://doi.org/10.1016/j.eswa.2021.115691
8. R. Berwick, An idiot's guide to support vector machines (SVMs). Village Idiot (2022). https://web.mit.edu/6.034/wwwbob/svm.pdf. Accessed 14 Mar 2022
9. A. Jana, Support vector machines for beginners—Linear SVM—A developer diary. *A Developer Diary* (2022). http://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-linear-svm/. Accessed 11 Mar 2022.
10. K. Jain, What is support vector machine? *Medium* (2022). https://towardsdatascience.com/what-is-support-vector-machine-870a0171e690. Accessed 14 Mar 2022.
11. F. Rossi, N. Villa, Support vector machine for functional data classification. Neurocomputing **69**(7), 730–742 (2006). https://doi.org/10.1016/j.neucom.2005.12.010
12. E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, in *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop, 24–26 Sept. 1997*, pp. 276–285 (1997). https://doi.org/10.1109/NNSP.1997.622408.
13. N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, Cambridge, 2000)
14. A.J. Smola, Regression estimation with support vector learning machines, Technische Universität München, München (1996). [Online]. Available: http://alex.smola.org/papers/1996/Smola96.pdf
15. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
16. M.E. Cholette, P. Borghesani, E.D. Gialleonardo, F. Braghin, Using support vector machines for the computationally efficient identification of acceptable design parameters in computer-aided engineering applications. Expert Syst. Appl. **81**(C), 39–52 (2017). https://doi.org/10.1016/j.eswa.2017.03.050
17. L.M.R. Baccarini, V.V. Rocha e Silva, B.R. de Menezes, W.M. Caminhas, SVM practical industrial application for mechanical faults diagnostic. Expert Syst. Appl. **38**(6), 6980–6984 (2011). https://doi.org/10.1016/j.eswa.2010.12.017
18. Z. Wei, Y. Feng, Z. Hong, R. Qu, J. Tan, Product quality improvement method in manufacturing process based on kernel optimisation algorithm. Int. J. Prod. Res. **55**(19), 5597–5608 (2017). https://doi.org/10.1080/00207543.2017.1324223
19. H. Rostami, J.-Y. Dantan, L. Homri, Review of data mining applications for quality assessment in manufacturing industry: support vector machines. Int. J. Metrol. Qual. Eng. **6**(4), 401 (2015). [Online]. Available: https://doi.org/10.1051/ijmqe/2015023.
20. C.J.C. Burges, B. Schölkopf, A.J. Smola, *Advances in Kernel Methods: Support Vector Learning* (MIT Press, Cambridge, MA, 1999)

Chapter 14
Voting and Bagging

14.1 The Problem

The ensemble technique relies on the idea that aggregation of many classifiers and regressors will lead to a better prediction [1]. In this chapter, we will introduce the ensemble technique and cover two ways in which to organize an ensemble (literally, a set) of machine learning methods called voting and bagging [2] and one algorithm to perform bagging called random forest [1, 3]. The other two ways to organize the ensemble methods are called boosting and stacking, which will be covered in the next chapter.

14.2 Voting Algorithm

The voting approach to the ensemble technique relies on allowing several algorithms to learn based on the same dataset; then in the presence of a new instance, they will all predict the output class or cluster. The final output will be decided based on a majority vote among the algorithms: the majority prediction is chosen as the output for the ensemble. This is called hard voting. Alternatively, we can choose soft voting, i.e., choosing the average prediction to be the output for the ensemble. This is typically possible when all predictors are able to estimate class probabilities (the probability that an instance belongs to a certain class or another) [1]. The average vote suggests that the output is the class with the highest average probability, i.e., the highest confidence. The algorithms used in voting need to be sufficiently diverse.

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_14

14.3 Bagging Algorithm

Another way of enhancing the strength of prediction is to use bagging (or bootstrap aggregating) [4]. Bootstrapping is a procedure that creates many subsets of data from the same dataset. Each sample is used and returned to the dataset before the next sample is drawn. This is called sampling with replacement. The subsets are used to train many models of an algorithm (e.g., decision tree), and then we proceed by majority voting or average voting (the latter is typically done in the case of a regression). The difference from voting is that each predictor is trained on a subset of the original dataset, which means that it will have a higher bias than if it was trained on the original one. However, after aggregation, the bias is reduced. Since the predictors have some common data instances, the aggregation reduces the variance (Fig. 14.1).

14.4 Random Forest

Decision trees suffer from overfitting; random forest overcomes this limitation. A random forest is an ensemble of decision trees trained using bagging [5]. However, individual trees are constructed, usually taking into account the important features of a dataset; hence, in bagging, there is a high chance that the decision trees used are highly correlated. To reduce the chance of correlation between the decision trees in an ensemble, random forest uses a random subset of features when searching for the split point at each node. The result is a diverse set of decision trees (i.e., less likely to be highly correlated), which enhances the performance of the random forest [2].

14.5 Voting Example

Download the file "ionosphere" from the Weka datasets or from the Kaggle website using the following link: https://www.kaggle.com/prashant111/ionosphere. Open the file in Weka, choose the Vote algorithm in the Classify tab (Fig. 14.2), and explore the voting parameters, particularly the combination rule that specifies the voting mechanism (Fig. 14.3).

Click on "classifiers" to choose the classifiers you would like to include in the voting algorithm (Fig. 14.4). You can click on the Choose and then Add buttons to add a new algorithm other the default ZeroR. Clicking on the Edit button allows you to edit the algorithm's parameters. Add a few algorithms of different types (Fig. 14.5). Run the voting algorithm and explore the results (Fig. 14.6). The accuracy achieved by voting was 91%. Try to see what the performance of each of the algorithms used in voting would be if run alone.

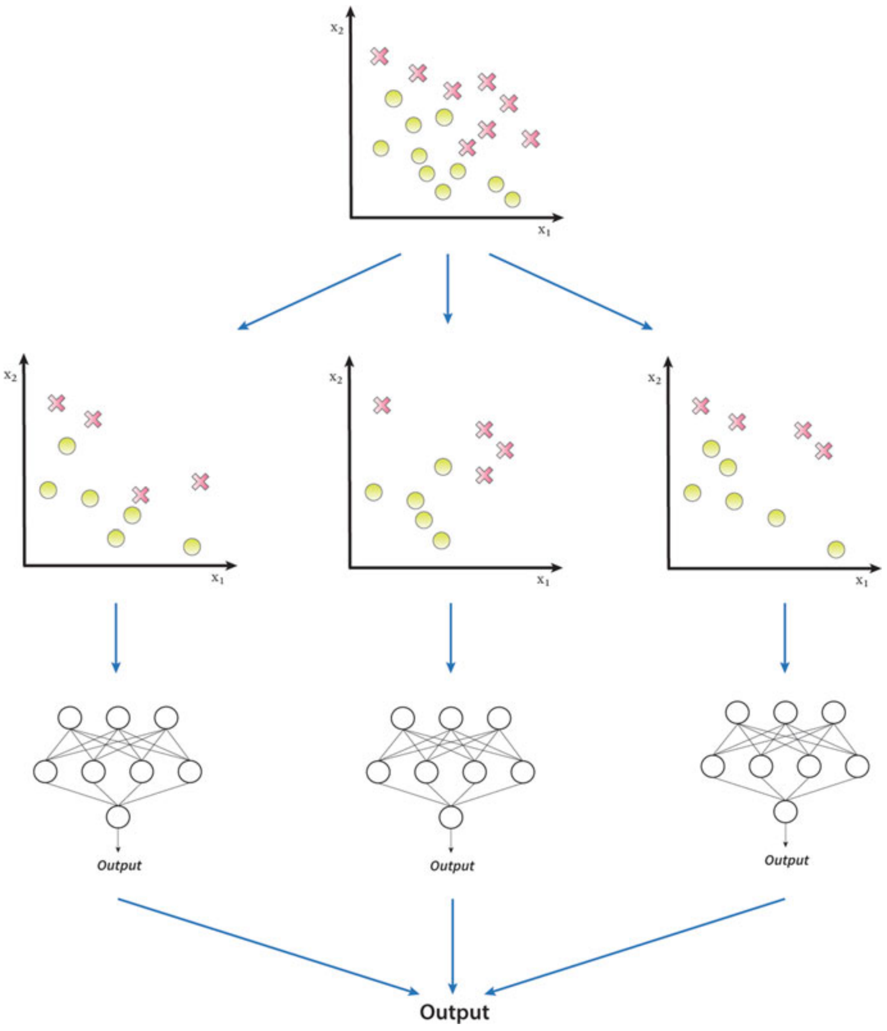


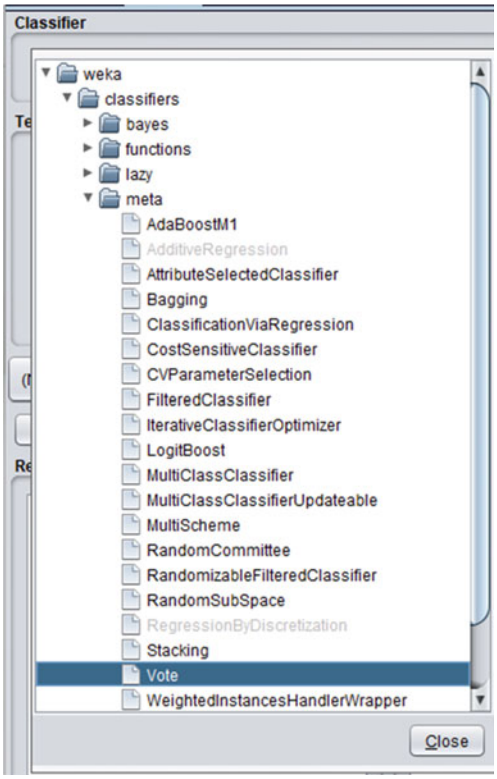
Fig. 14.1 Overview of bagging

14.6 Bagging Example: Random Forest

If not open yet, open the "ionosphere" file in Weka and choose the random forest algorithm in the Classify tab (Fig. 14.7).

Open the parameters of the bagging algorithm and explore the parameters, noting that the REP tree is the default classifier; the REP tree is the decision tree or classification and regression tree (CART) implementation in Weka (Fig. 14.8).

Choose Cross-validation with tenfolds (Fig. 14.9) and click the Start button. The output window displays the random forest results (Fig. 14.10).


Classifier

weka
classifiers
bayes
functions
lazy
meta
AdaBoostM1
AdditiveRegression
AttributeSelectedClassifier
Bagging
ClassificationViaRegression
CostSensitiveClassifier
CVParameterSelection
FilteredClassifier
IterativeClassifierOptimizer
LogitBoost
MultiClassClassifier
MultiClassClassifierUpdateable
MultiScheme
RandomCommittee
RandomizableFilteredClassifier
RandomSubSpace
RegressionByDiscretization
Stacking
Vote
WeightedInstancesHandlerWrapper

Fig. 14.2 Choosing the vote algorithm in Weka

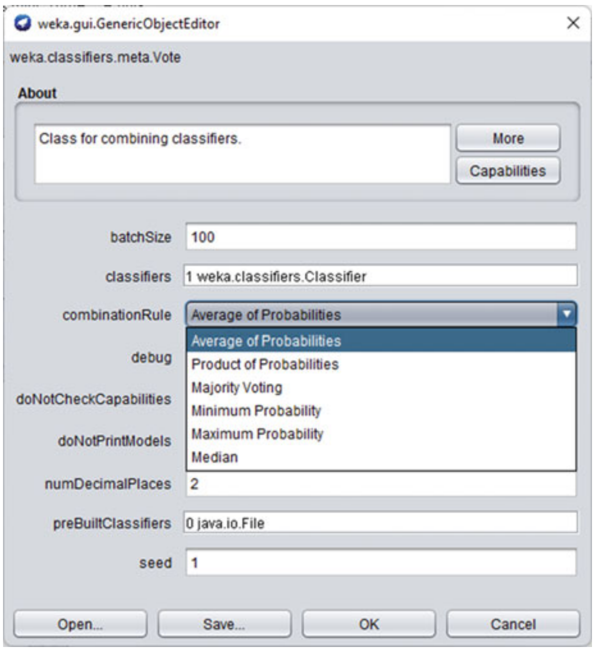
The random forest results show an RMSE of 0.26 and a precision of 91%.

14.7 Final Notes: Advantages, Disadvantages, and Best Practices

Voting is surprisingly very effective; even when the individual predictors are weak (i.e., perform slightly better than a random guess), voting provides good performance. However, voting will not provide a spectacular outcome if the predictors are not independent or not loosely correlated, and if they are not many and diverse (i.e., use a different type of algorithms).

14.8 Key Terms

1. Voting
2. Bagging



weka.gui.GenericObjectEditor

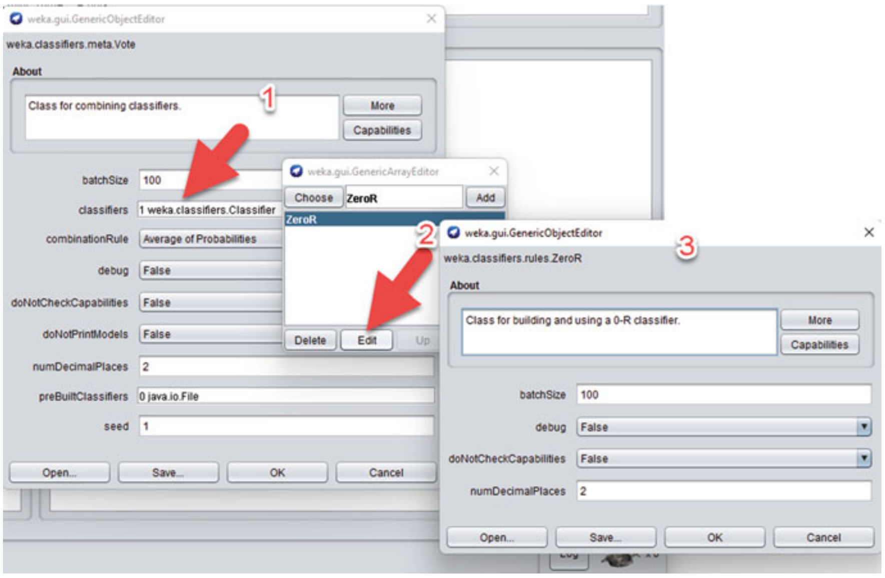
weka.classifiers.meta.Vote

About

Class for combining classifiers.

batchSize	100
classifiers	1 weka.classifiers.Classifier
combinationRule	Average of Probabilities
debug	
doNotCheckCapabilities	
doNotPrintModels	
numDecimalPlaces	2
preBuiltClassifiers	0 java.io.File
seed	1

Fig. 14.3 Vote algorithm parameters



weka.gui.GenericObjectEditor

weka.classifiers.meta.Vote

About

Class for combining classifiers.

batchSize	100
classifiers	1 weka.classifiers.Classifier
combinationRule	Average of Probabilities
debug	False
doNotCheckCapabilities	False
doNotPrintModels	False
numDecimalPlaces	2
preBuiltClassifiers	0 java.io.File
seed	1

weka.gui.GenericArrayEditor

ZeroR

weka.gui.GenericObjectEditor

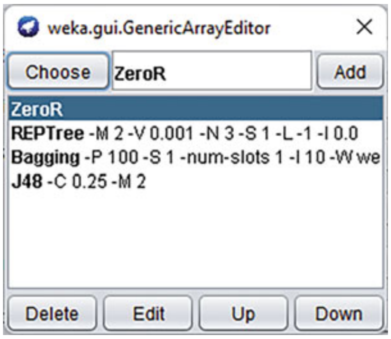
weka.classifiers.rules.ZeroR

About

Class for building and using a 0-R classifier.

batchSize	100
debug	False
doNotCheckCapabilities	False
numDecimalPlaces	2

Fig. 14.4 Algorithms included in voting and their corresponding parameters



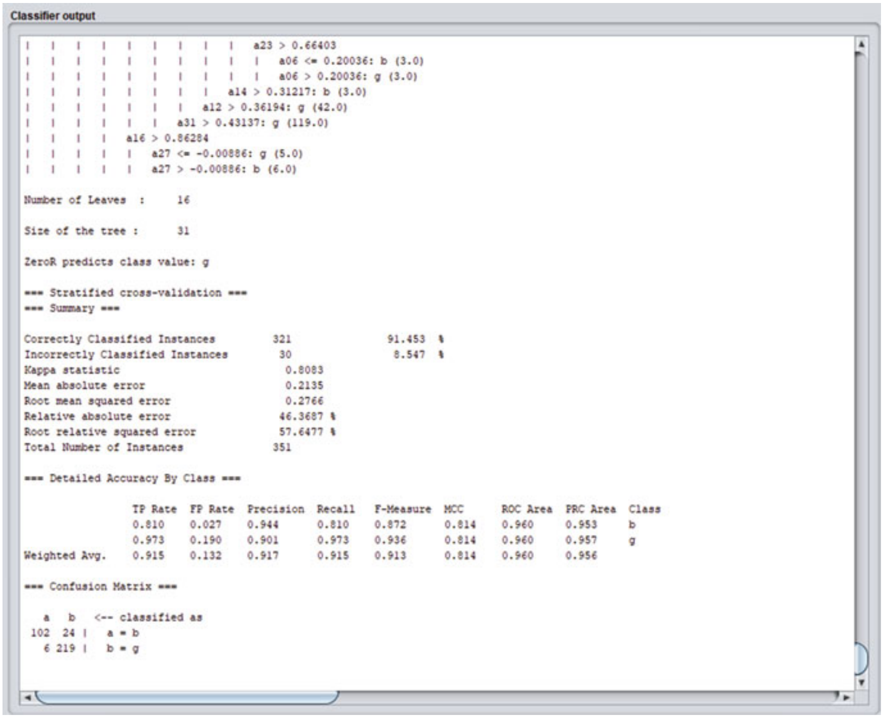
weka.gui.GenericArrayEditor

Choose ZeroR Add

ZeroR
REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Bagging -P 100 -S 1 -num-slots 1 -I 10 -W we
J48 -C 0.25 -M 2

Delete Edit Up Down

Fig. 14.5 Add J48, bagging, and REPTree algorithms to the voting algorithm



Classifier output

```
|   |   |   |   |   |   |   |   |   a23 > 0.66403
|   |   |   |   |   |   |   |   |   |   a06 <= 0.20036: b (3.0)
|   |   |   |   |   |   |   |   |   |   a06 > 0.20036: g (3.0)
|   |   |   |   |   |   |   |   a14 > 0.31217: b (3.0)
|   |   |   |   |   |   |   a12 > 0.36194: g (42.0)
|   |   |   |   |   |   a31 > 0.43137: g (119.0)
|   |   |   |   a16 > 0.86284
|   |   |   |   |   a27 <= -0.00886: g (5.0)
|   |   |   |   |   a27 > -0.00886: b (6.0)

Number of Leaves  :     16

Size of the tree :      31

ZeroR predicts class value: g

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         321               91.453  %
Incorrectly Classified Instances        30                8.547  %
Kappa statistic                          0.8083
Mean absolute error                      0.2135
Root mean squared error                  0.2766
Relative absolute error                 46.3687 %
Root relative squared error             57.6477 %
Total Number of Instances              351

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.810    0.027    0.944      0.810    0.872      0.814    0.960     0.953     b
                 0.973    0.190    0.901      0.973    0.936      0.814    0.960     0.957     g
Weighted Avg.    0.915    0.132    0.917      0.915    0.913      0.814    0.960     0.956

=== Confusion Matrix ===

   a   b   <-- classified as
 102  24 |   a = b
   6 219 |   b = g
```

Fig. 14.6 Voting results

3. Random forest
4. Hard voting
5. Soft voting
6. Sampling with replacement
7. Correlation
8. Decision trees

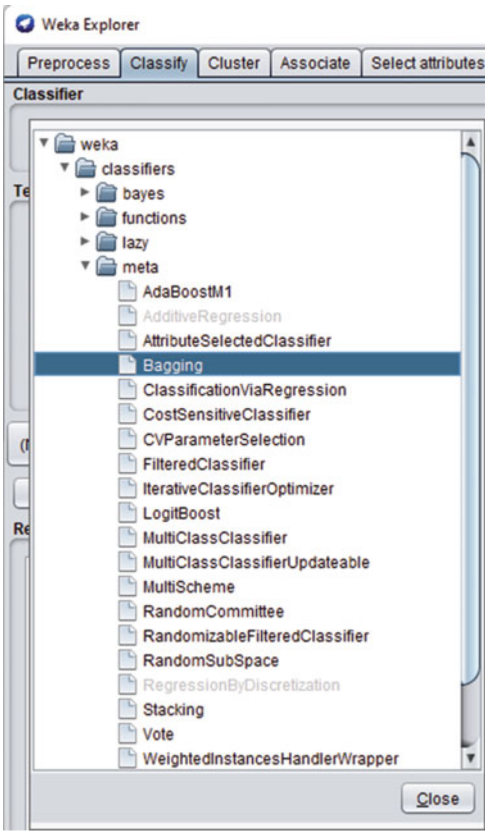
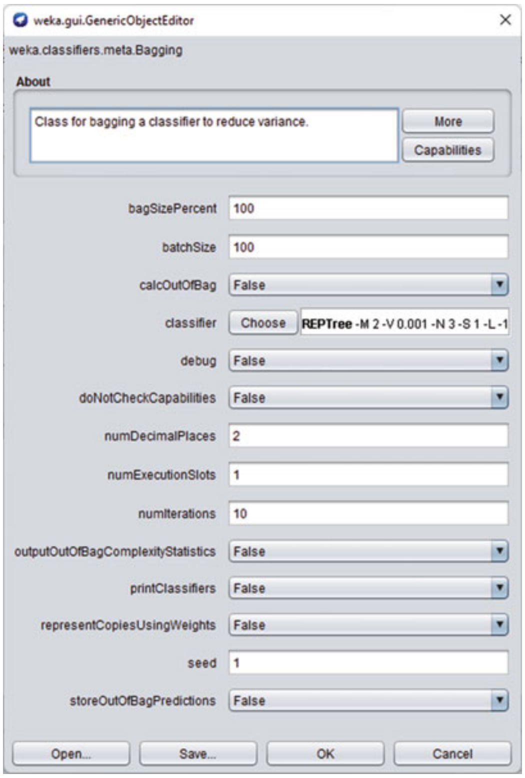


Fig. 14.7 Bagging algorithm in Weka

14.9 Test Your Understanding

1. How does voting function in machine learning?
2. Explain bagging and why it is expected to enhance performance.
3. Explain a few challenges in bagging.
4. Explain soft voting vs. hard voting.
5. Explain sampling with replacement.
6. How does sampling without replacement differ from sampling with replacement?
7. Cite some of the hyperparameters of random forest.



weka.classifiers.meta.Bagging

About

Class for bagging a classifier to reduce variance.

bagSizePercent	100
batchSize	100
calcOutOfBag	False
classifier	REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1
debug	False
doNotCheckCapabilities	False
numDecimalPlaces	2
numExecutionSlots	1
numIterations	10
outputOutOfBagComplexityStatistics	False
printClassifiers	False
representCopiesUsingWeights	False
seed	1
storeOutOfBagPredictions	False

Fig. 14.8 Bagging hyperparameters in Weka

14.10 Read More

1. Bao, S., Pan, H. Y., Zheng, W., Wu, Q. Q., Dai, Y. N., Sun, N. N., . . . Pan, H. Y. (2021). Multicenter analysis and a rapid screening model to predict early novel coronavirus pneumonia using a random forest algorithm. Medicine (Baltimore), 100(24), e26279. doi: 10.1097/md.0000000000026279
2. Best, K. B., Gilligan, J. M., Baroud, H., Carrico, A. R., Donato, K. M., Ackerly, B. A., & Mallick, B. (2021). Random forest analysis of two household surveys can identify important predictors of migration in Bangladesh. Journal of Computational Social Science, 4(1), 77–100. doi: 10.1007/s42001-020-00066-9
3. Chen, X., Yu, S., Zhang, Y., Chu, F., & Sun, B. (2021). Machine Learning Method for Continuous Noninvasive Blood Pressure Detection Based on Random Forest. IEEE Access, 9, 34112–34118. doi: 10.1109/ACCESS.2021.3062033
4. Chencho, Li, J., Hao, H., Wang, R., & Li, L. (2021). Development and application of random forest technique for element level structural damage quantification. Structural Control and Health Monitoring, 28(3), n/a-n/a. doi: 10.1002/stc.2678

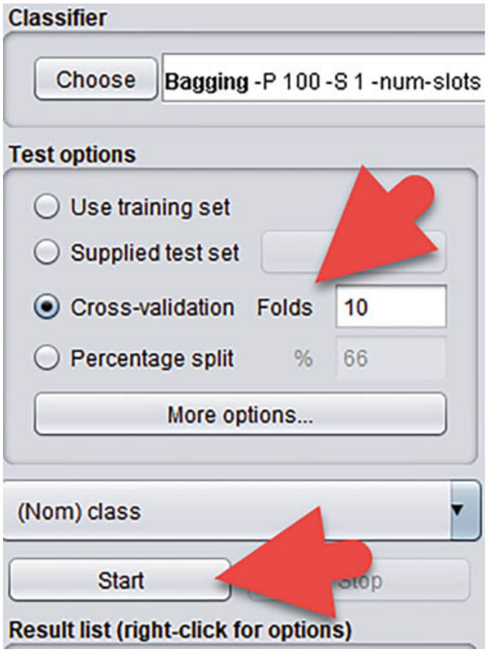
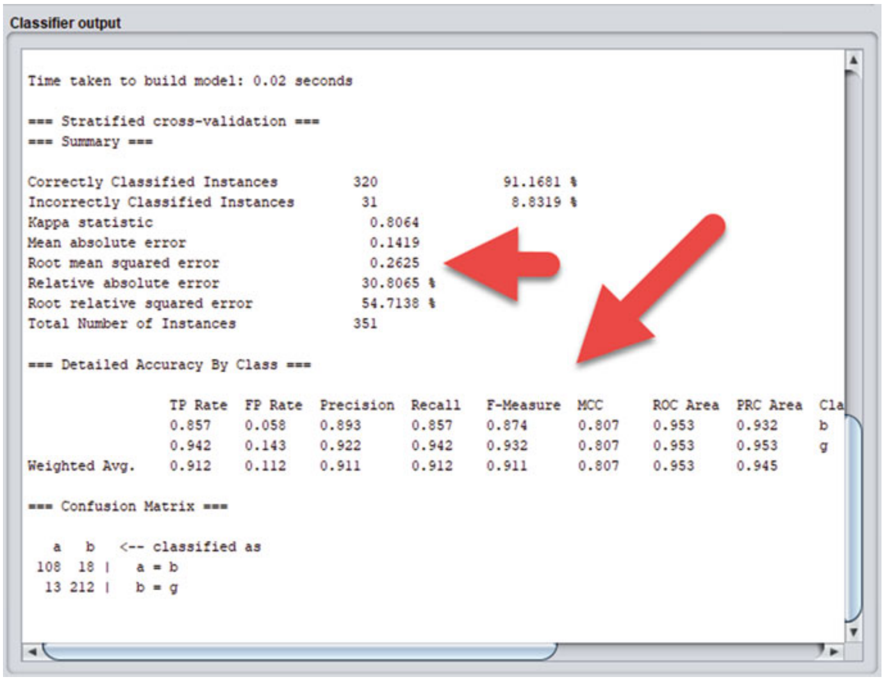


Fig. 14.9 Cross-validation with tenfolds


Classifier output

```
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         320               91.1681 %
Incorrectly Classified Instances        31                8.8319 %
Kappa statistic                          0.8064
Mean absolute error                      0.1419
Root mean squared error                  0.2625
Relative absolute error                 30.8065 %
Root relative squared error             54.7138 %
Total Number of Instances              351

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cla
                 0.857    0.058    0.893      0.857    0.874      0.807    0.953     0.932     b
                 0.942    0.143    0.922      0.942    0.932      0.807    0.953     0.953     g
Weighted Avg.    0.912    0.112    0.911      0.912    0.911      0.807    0.953     0.945

=== Confusion Matrix ===

   a   b   <-- classified as
 108  18 |   a = b
  13 212 |   b = g
```

Fig. 14.10 Random forest results

5. Fatlawi, H. K., & Kiss, A. (2021). Differential privacy based classification model for mining medical data stream using adaptive random forest. Acta Universitatis Sapientiae, Informatica, 13(1), 1–20. doi: 10.2478/ausi-2021-0001
6. Fernandez-Lozano, C., Hervella, P., Mato-Abad, V., Rodríguez-Yáñez, M., Suárez-Garaboa, S., López-Dequidt, I., … Iglesias-Rey, R. (2021). Random forest-based prediction of stroke outcome. Sci Rep, 11(1), 10071. doi: 10.1038/s41598-021-89434-7
7. GIS-based forest fire susceptibility assessment by random forest, artificial neural network and logistic regression methods. (2021). Journal of Tropical Forest Science, 33(2), 173–184. Retrieved from http://resolver.scholarsportal.info/resolve/01281283/v33i0002/173_gffsabnnalrm_2
8. Hanko, M., Grendár, M., Snopko, P., Opšenák, R., Šutovský, J., Benčo, M., … Kolarovszki, B. (2021). Random Forest-Based Prediction of Outcome and Mortality in Patients with Traumatic Brain Injury Undergoing Primary Decompressive Craniectomy. World Neurosurg, 148, e450-e458. doi: 10.1016/j.wneu.2021.01.002
9. Jianyun, H. (2021). Retracted Article: Geochemical characteristics of South China Sea based on random forest algorithm and Wushu teaching action simulation. Arabian Journal of Geosciences, 14(17). doi: 10.1007/s12517-021-08182-0
10. Jiao, S., Zou, Q., Guo, H., & Shi, L. (2021). iTTCA-RF: a random forest predictor for tumor T cell antigens. J Transl Med, 19(1), 449. doi: 10.1186/s12967-021-03084-x
11. Liu, D., Zhang, X., Zheng, T., Shi, Q., Cui, Y., Wang, Y., & Liu, L. (2021). Optimisation and evaluation of the random forest model in the efficacy prediction of chemoradiotherapy for advanced cervical cancer based on radiomics signature from high-resolution T2 weighted images. Arch Gynecol Obstet, 303(3), 811–820. doi: 10.1007/s00404-020-05908-5
12. Liu, D., Zhang, X., Zheng, T., Shi, Q., Cui, Y., Wang, Y., & Liu, L. (2021). Optimisation and evaluation of the random forest model in the efficacy prediction of chemoradiotherapy for advanced cervical cancer based on radiomics signature from high-resolution T2 weighted images. Archives of Gynecology and Obstetrics, 303(3), 811–820. doi: 10.1007/s00404-020-05908-5
13. Neto, M. P., & Paulovich, F. V. (2021). Explainable Matrix—Visualization for Global and Local Interpretability of Random Forest Classification Ensembles. IEEE Transactions on Visualization and Computer Graphics, 27(2), 1427–1437. doi: 10.1109/TVCG.2020.3030354
14. Ribeiro, M. N., Carvalho, I. A., Fonseca, G. A., Lago, R. C., Rocha, L. C., Ferreira, D. D., … Pinheiro, A. C. (2021). Quality control of fresh strawberries by a random forest model. J Sci Food Agric, 101(11), 4514–4522. doi: 10.1002/jsfa.11092
15. Simsekler, M. C. E., Alhashmi, N. H., Azar, E., King, N., Luqman, R., & Al Mulla, A. (2021). Exploring drivers of patient satisfaction using a random forest algorithm. BMC Med Inform Decis Mak, 21(1), 157. doi: 10.1186/s12911-021-01519-5

16. Simsekler, M. C. E., Alhashmi, N. H., Azar, E., King, N., Luqman, R. A. M. A., & Al Mulla, A. (2021). Exploring drivers of patient satisfaction using a random forest algorithm. BMC Medical Informatics and Decision Making, 21. doi: 10.1186/s12911-021-01519-5
17. Speiser, J. L. (2021). A random forest method with feature selection for developing medical prediction models with clustered and longitudinal data. Journal of Biomedical Informatics, 117(Complete). doi: 10.1016/j.jbi.2021.103763
18. Tracy, B. M., Finnegan, T. M., Smith, R. N., & Senkowski, C. K. (2021). Random forest modeling using socioeconomic distress predicts hernia repair approach. Surg Endosc, 35(7), 3890–3895. doi: 10.1007/s00464-020-07860-6
19. Weppler, S., Quon, H., Schinkel, C., Ddamba, J., Harjai, N., Vigal, C., … Smith, W. (2021). Determining Clinical Patient Selection Guidelines for Head and Neck Adaptive Radiation Therapy Using Random Forest Modelling and a Novel Simplification Heuristic. Frontiers in Oncology, 11. doi: 10.3389/fonc.2021.650335
20. Yan, K., Zhao, J., & Ren, Y. (2021). Electricity Theft Identification Algorithm Based on Auto-Encoder Neural Network and Random Forest. 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 5, 2641–2645. doi: 10.1109/IAEAC50856.2021.9390774
21. Yaqoob, M. K., Ali, S. F., Bilal, M., Hanif, M. S., & Al-Saggaf, U. M. (2021). ResNet Based Deep Features and Random Forest Classifier for Diabetic Retinopathy Detection. Sensors (Basel), 21(11). doi: 10.3390/s21113883

14.11 Lab

14.11.1 A working Example in Python

Download the *Titanic* dataset from the following link: https://www.kaggle.com/competitions/titanic/data?select=train.csv.

There are three files, we will use for this lab demonstration one of them “train.csv” that we have renamed to “titanic.csv.” This dataset contains information about the *Titanic*’s passengers and is formed of the following features:

- Survived: survival (0 or 1)
- Pclass: ticket class (1, 2, or 3)
- Sex: gender (M, F, or Unknown)
- Age: passenger age in years
- Sibsp: # of siblings/spouses aboard the *Titanic*
- Parch: # of parents/children aboard the *Titanic*
- Ticket: ticket number
- Fare: passenger fare
- Cabin: cabin number
- Embarked: port of embarkation: C=Cherbourg, Q=Queenstown, S=Southampton

14.11.1.1 Load Titanic Dataset

The first task is to upload the *Titanic* dataset (Fig. 14.11).

14.11.1.2 Visualizing Titanic Dataset

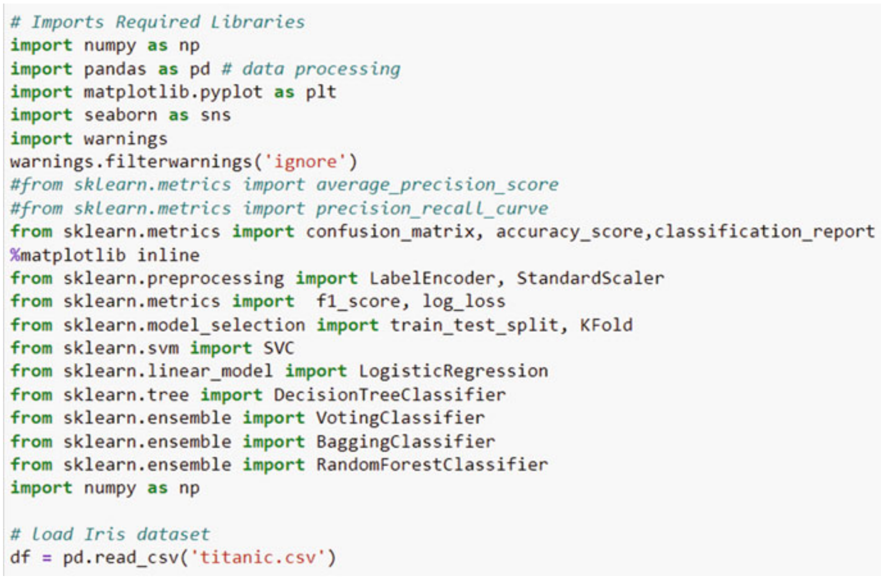
We can visualize the data in many ways. In Fig. 14.12, we are using the heatmap.

14.11.1.3 Preprocess and Manipulate Data

The next step is to drop unnecessary columns and fill in the missing values. PassengerId, Cabin, ticket, and name features are not useful and hence dropped. The sex and embarked features are mapped to numeric, and the values for the age and fare features are scaled (Fig. 14.13).

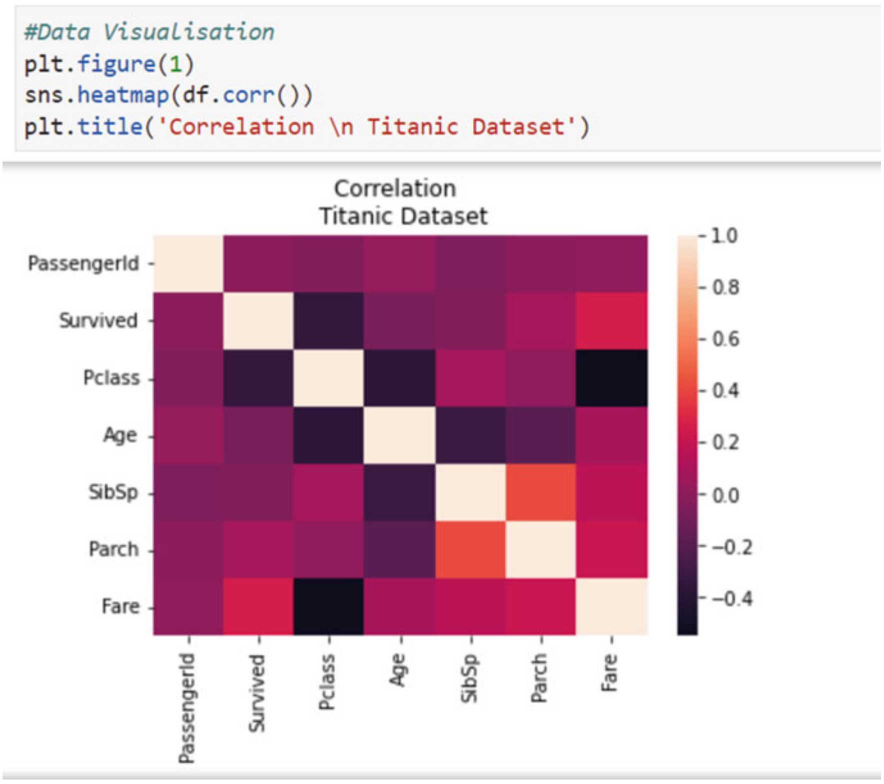
14.11.1.4 Create Bagging and Voting Models

The next step is to choose the features and the target ("Survived" column is the target). We split the dataset into testing and training datasets, and prepare for bagging based on a decision tree classifier, while for voting we opted for logistic



```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
#from sklearn.metrics import average_precision_score
#from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
%matplotlib inline
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import  f1_score, log_loss
from sklearn.model_selection import train_test_split, KFold
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Load Iris dataset
df = pd.read_csv('titanic.csv')
```

Fig. 14.11 Loading *Titanic* dataset into pandas



```
#Data Visualisation
plt.figure(1)
sns.heatmap(df.corr())
plt.title('Correlation \n Titanic Dataset')
```

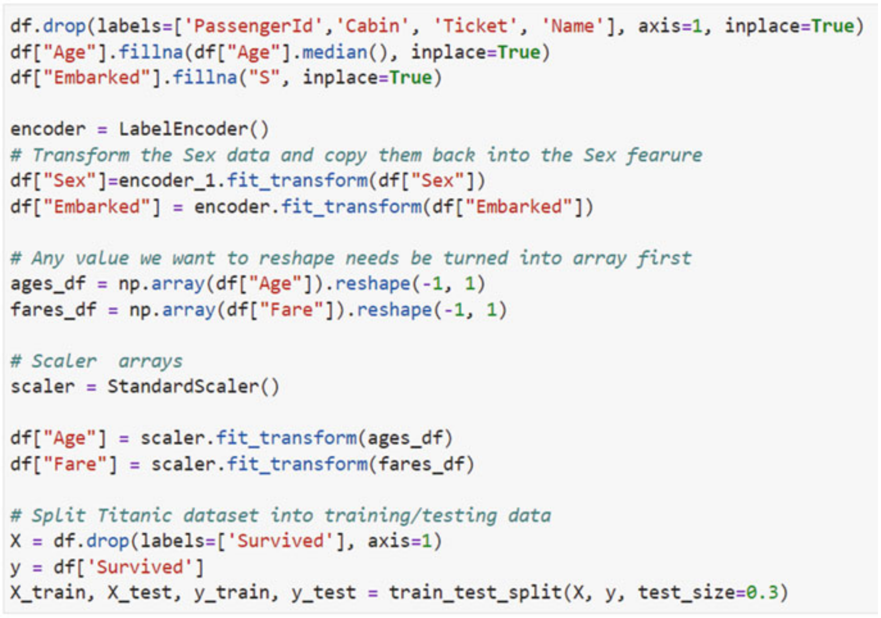
Fig. 14.12 Exploring *Titanic* data visually

regression and a support vector machine. Note that the voting parameter value is “hard” (Fig. 14.14).

For the sake of learning, we are going to change or procedure for this chapter. We will be using fit the models on the training dataset and use them to make predictions on the testing dataset. Hence, we will proceed with hyperparameter tuning using grid search.

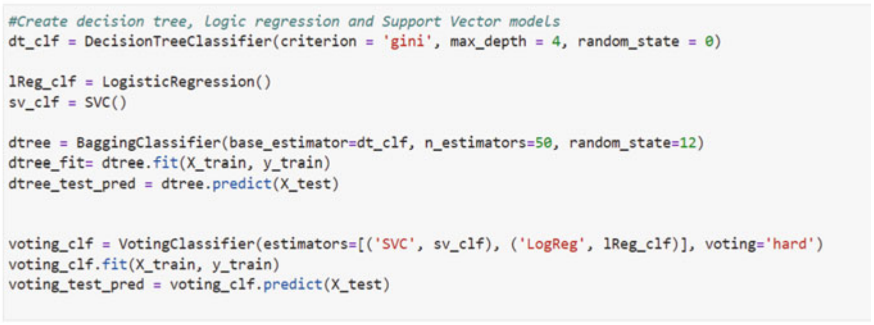
14.11.1.5 Evaluate Bagging and Voting Model’s

In Fig. 14.5 we define a procedure to print the scores of a prediction. It is then used to print the performance scores of the bagging and voting classifiers (Fig. 14.15).



```
df.drop(labels=['PassengerId','Cabin', 'Ticket', 'Name'], axis=1, inplace=True)
df["Age"].fillna(df["Age"].median(), inplace=True)
df["Embarked"].fillna("S", inplace=True)

encoder = LabelEncoder()
# Transform the Sex data and copy them back into the Sex fearure
df["Sex"]=encoder_1.fit_transform(df["Sex"])
df["Embarked"] = encoder.fit_transform(df["Embarked"])

# Any value we want to reshape needs be turned into array first
ages_df = np.array(df["Age"]).reshape(-1, 1)
fares_df = np.array(df["Fare"]).reshape(-1, 1)

# Scaler  arrays
scaler = StandardScaler()

df["Age"] = scaler.fit_transform(ages_df)
df["Fare"] = scaler.fit_transform(fares_df)

# Split Titanic dataset into training/testing data
X = df.drop(labels=['Survived'], axis=1)
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Fig. 14.13 Preprocess and scale data in *Titanic* dataset



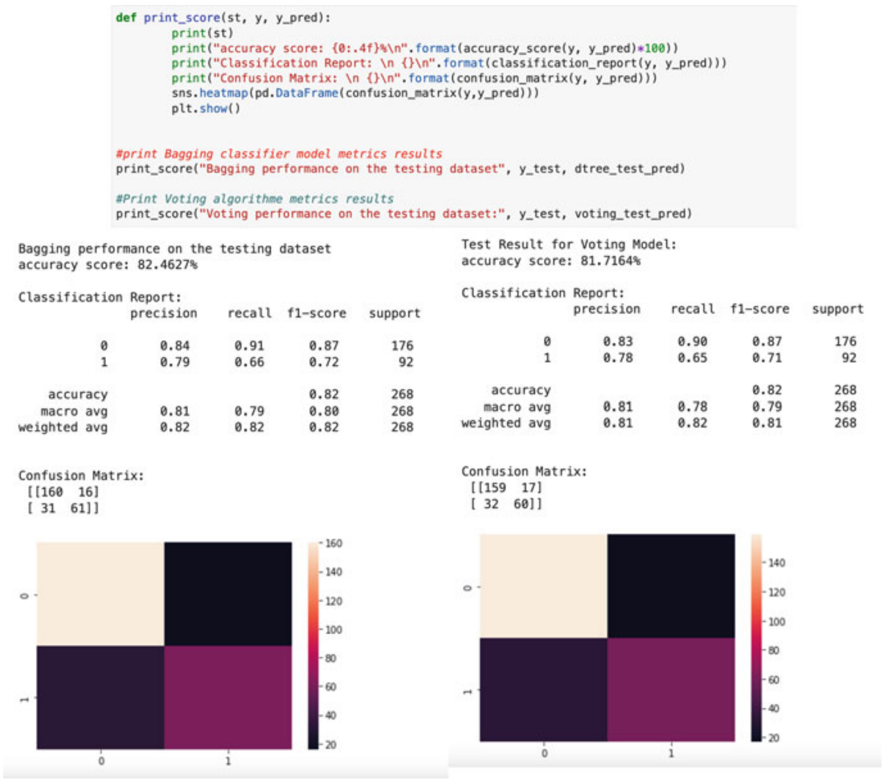
```
#Create decision tree, Logic regression and Support Vector models
dt_clf = DecisionTreeClassifier(criterion = 'gini', max_depth = 4, random_state = 0)

lReg_clf = LogisticRegression()
sv_clf = SVC()

dtree = BaggingClassifier(base_estimator=dt_clf, n_estimators=50, random_state=12)
dtree_fit= dtree.fit(X_train, y_train)
dtree_test_pred = dtree.predict(X_test)


voting_clf = VotingClassifier(estimators=[('SVC', sv_clf), ('LogReg', lReg_clf)], voting='hard')
voting_clf.fit(X_train, y_train)
voting_test_pred = voting_clf.predict(X_test)
```

Fig. 14.14 Creating bagging and voting models

14.11.1.6 Optimize the Bagging and Voting Models

The next step is to optimize the two models, and that test them, grid search cross-validation is used to optimize the bagging (Fig. 14.16) and Voting (Fig. 14.17) models.

Did you obtain the same results? What do you think about those results?



```
def print_score(st, y, y_pred):
        print(st)
        print("accuracy score: {0:.4f}%\n".format(accuracy_score(y, y_pred)*100))
        print("Classification Report: \n {}\n".format(classification_report(y, y_pred)))
        print("Confusion Matrix: \n {}\n".format(confusion_matrix(y, y_pred)))
        sns.heatmap(pd.DataFrame(confusion_matrix(y,y_pred)))
        plt.show()


#print Bagging classifier model metrics results
print_score("Bagging performance on the testing dataset", y_test, dtree_test_pred)

#Print Voting algorithme metrics results
print_score("Voting performance on the testing dataset:", y_test, voting_test_pred)
```

```
Bagging performance on the testing dataset
accuracy score: 82.4627%

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.91      0.87       176
           1       0.79      0.66      0.72        92

    accuracy                           0.82       268
   macro avg       0.81      0.79      0.80       268
weighted avg       0.82      0.82      0.82       268


Confusion Matrix:
 [[160  16]
 [ 31  61]]
```

```
Test Result for Voting Model:
accuracy score: 81.7164%

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.90      0.87       176
           1       0.78      0.65      0.71        92

    accuracy                           0.82       268
   macro avg       0.81      0.78      0.79       268
weighted avg       0.81      0.82      0.81       268


Confusion Matrix:
 [[159  17]
 [ 32  60]]
```

Fig. 14.15 Getting classification report and confusion matrix for the bagging and voting models

14.11.2 Do It Yourself

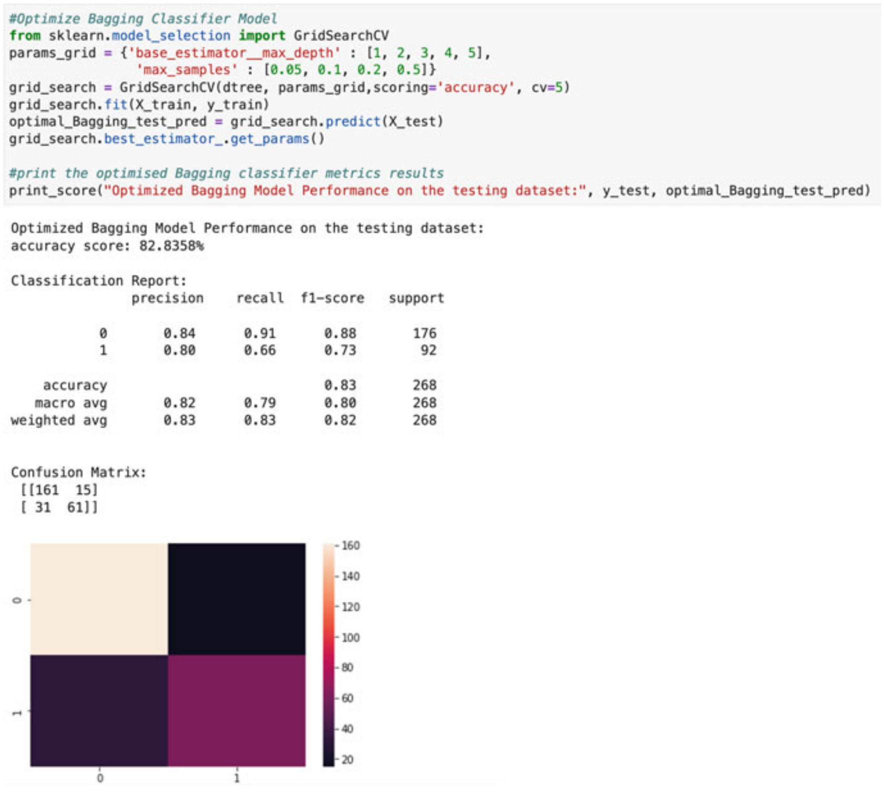
14.11.2.1 The Titanic revisited

1. Did you obtain the same results when you run your code? What do you think about those results?
2. During the evaluation step above we have just applied the models to the testing dataset. That is not the best option. What is a better approach (you might refer to Chap. 4)?
3. Use cross-validation to redo the evaluation step.

14.11.2.2 The Diabetes Dataset

Download the Pima Indians diabetes dataset using the following link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

And do the following steps:

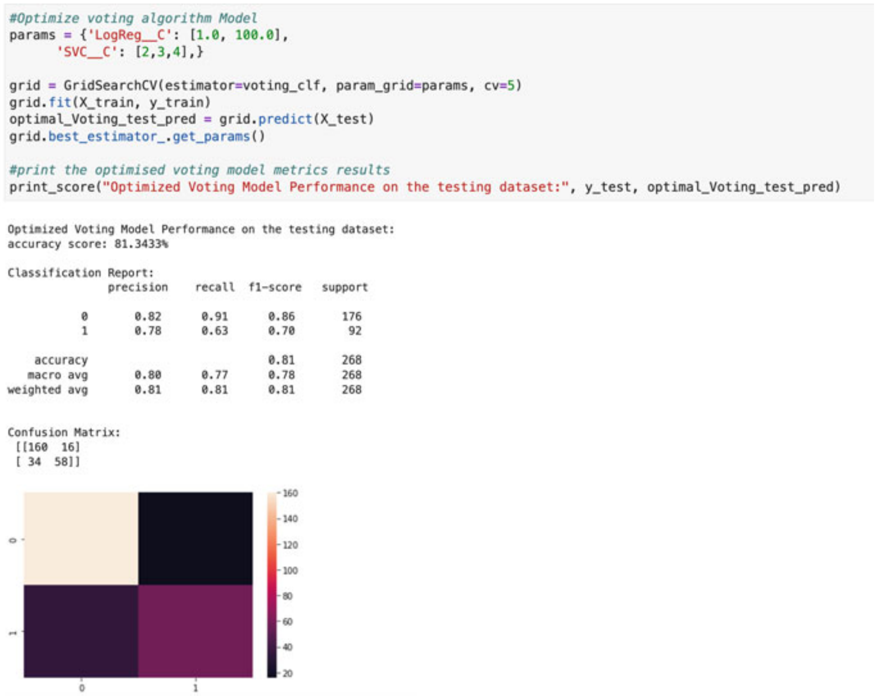


```
#Optimize Bagging Classifier Model
from sklearn.model_selection import GridSearchCV
params_grid = {'base_estimator__max_depth' : [1, 2, 3, 4, 5],
               'max_samples' : [0.05, 0.1, 0.2, 0.5]}
grid_search = GridSearchCV(dtree, params_grid,scoring='accuracy', cv=5)
grid_search.fit(X_train, y_train)
optimal_Bagging_test_pred = grid_search.predict(X_test)
grid_search.best_estimator_.get_params()

#print the optimised Bagging classifier metrics results
print_score("Optimized Bagging Model Performance on the testing dataset:", y_test, optimal_Bagging_test_pred)
```

```
Optimized Bagging Model Performance on the testing dataset:
accuracy score: 82.8358%

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.91      0.88       176
           1       0.80      0.66      0.73        92

    accuracy                           0.83       268
   macro avg       0.82      0.79      0.80       268
weighted avg       0.83      0.83      0.82       268


Confusion Matrix:
 [[161  15]
 [ 31  61]]
```

Fig. 14.16 Optimize bagging model using grid search cross-validation approach

- Load the data into pandas.
- Visualize your data using the seaborn library.
- Preprocess the data and split it into training and testing datasets using a 0.7:0.3 ratio.
- Create a random forest model using decision trees.
- Create a voting algorithm using linear regression, support vectors, and decision trees.
- Evaluate the bagging classifier and voting models using classification reports and confusion matrices for training and testing datasets.
- Optimize the bagging classifier and voting models using grid search cross-validation.
- Evaluate the optimized models using classification reports and confusion matrices and retain the optimized versions of the models.



```
#Optimize voting algorithm Model
params = {'LogReg__C': [1.0, 100.0],
      'SVC__C': [2,3,4],}

grid = GridSearchCV(estimator=voting_clf, param_grid=params, cv=5)
grid.fit(X_train, y_train)
optimal_Voting_test_pred = grid.predict(X_test)
grid.best_estimator_.get_params()

#print the optimised voting model metrics results
print_score("Optimized Voting Model Performance on the testing dataset:", y_test, optimal_Voting_test_pred)
```

```
Optimized Voting Model Performance on the testing dataset:
accuracy score: 81.3433%

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       176
           1       0.78      0.63      0.70        92

    accuracy                           0.81       268
   macro avg       0.80      0.77      0.78       268
weighted avg       0.81      0.81      0.81       268


Confusion Matrix:
 [[160  16]
 [ 34  58]]
```

Fig. 14.17 Optimize the voting model using grid search cross-validation approach

14.11.3 Do More Yourself

Download the following datasets and practice using bagging classifier and voting models:

- https://www.kaggle.com/code/shiva948/bike-sharing-systems/data
- https://www.kaggle.com/code/nandinibagga/fertilizer-type-prediction/data
- https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

References

1. A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly Media, Sebastopol, CA, 2019)

2. Y. Liu, *Python Machine Learning by Example: Build Intelligent Systems Using Python, TensorFlow 2, PyTorch, and Scikit-Learn, 3rd Edition (Kindle Edition)* (Packt, 2020)
3. J. Grus, *Data Science from Scratch: First Principles with Python* (O'Reilly Media, Sebastopol, CA, 2015)
4. L. Breiman, Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996). https://doi.org/10.1007/BF00058655
5. H. Tin Kam, The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998). https://doi.org/10.1109/34.709601

Chapter 15
Boosting and Stacking

15.1 The Problem

The ensemble technique relies on an aggregate of models' output to provide a better prediction. Other than voting and bagging, we can use boosting and stacking.

15.2 Boosting

Boosting (or hypothesis boosting) refers to an ensemble method that builds a strong learner out of a combination of weak learners (i.e., learners that perform slightly better than random guessing). The predictors are trained sequentially, and each subsequent predictor tries to correct the current one [1]. The dataset is the same for all algorithms; however, each data instance is subject to a weight based on the outcome of the previous model's success [2]; in each iteration, to factor in the prediction difficulty of incorrectly classified instances, their weight is increased. We usually use this technique when learning a new skill, as we focus our attention on difficult aspects. Boosting algorithms differ in the way they calculate the weights (Fig. 15.1).

15.3 Stacking

In stacking (or stacking generalization [3]), the outputs of several algorithms are used as the input of the main algorithm (called sometime the blender [1]) that is supposed to make the final prediction. Practically, we feed the blender with the predicted outcomes of the preceding algorithms. The training dataset is divided into

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_15

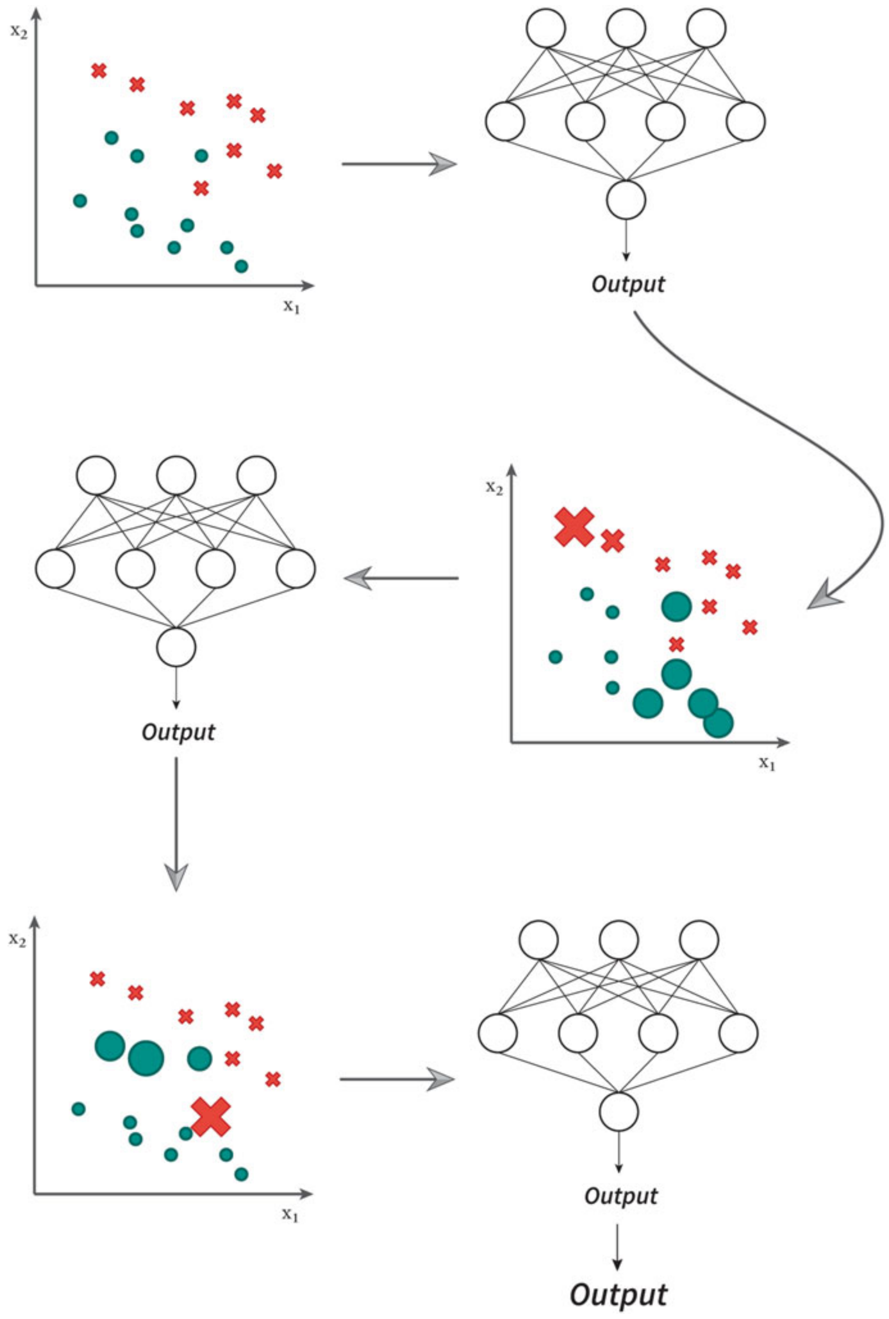


Fig. 15.1 Boosting in action

two parts, a subset (a holdout) to train the blender and a subset to train the other algorithms. The blender uses the outcomes of the other algorithms as input features and the labels from the holdout dataset to train the blender. The blender will learn to predict the labels based on the input features (i.e., the outcomes of the algorithms).

What we have just explained is a stacking mechanism with two layers and one blender. It is possible to create stacking with more than two layers; for instance, in stacking with three layers, the dataset is split into three subsets. The first is used at layer 1 to generate the outcomes, which will act as input for the first blender at layer 2, which will also use the labels of the second subset for training. The second blender at layer 3 will act similarly, i.e., it uses the outcomes of layer 2 and the labels of the third subset for training.

15.4 Boosting Example

15.4.1 AdaBoost Algorithm

AdaBoost is a boosting algorithm that focuses its attention on the training instances that the predecessor algorithm misclassified [4].

Initially, each instance of the dataset is assigned equal weight. Using the training dataset, AdaBoost trains a weak learner classifier, such as a decision tree with one level (called a decision stump). Then, AdaBoost uses the developed model to make predictions about the training dataset and increases the weight for the misclassified instances. The dataset with the updated weights is then used for training in the next iteration. The process continues until the desired number of classifiers is reached or no further improvement in classification can be made.

Once trained, AdaBoost makes predictions by calculating all the predictions of all the predictors, weighting them using the predictors' weights. The predicted class is determined by the majority of the weighted votes [1].

At each iteration, AdaBoost focuses on misclassified instances; this strategy improves the performance of the weak classifiers drastically (Fig. 15.2).

The following is a summary of the training of AdaBoost algorithm for a dataset of m instances and N predictors:

```
Initialize the weights $w^{(i)} = \frac{1}{m}$
For $j = 1$ to $N$
Begin For $j$
    1. For $i = 1$ to $m$
           Begin For $i$
       Calculate $j$th prediction $\widehat{y}_j^{(i)}$ for each instance $x^{(i)}$
           End For $i$
```

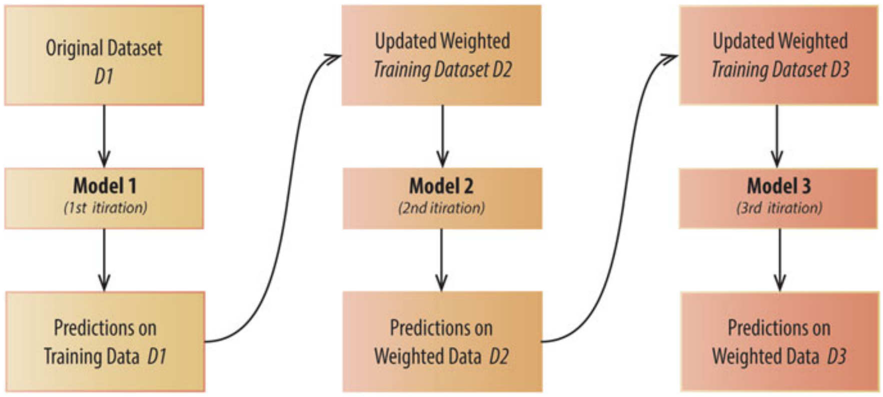

Original Dataset D1
Model 1 (1st itiration)
Predictions on Training Data D1
Updated Weighted Training Dataset D2
Model 2 (2nd itiration)
Predictions on Weighted Data D2
Updated Weighted Training Dataset D3
Model 3 (3rd itiration)
Predictions on Weighted Data D3

Fig. 15.2 Overview of AdaBoost

2. Calculate the *j*th predictor's error rate

$$r_j = \frac{\left(\text{for } \widehat{y}_j^{(i)} \neq y^{(i)}\right) \sum_{i=1}^{m} w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

where $\widehat{y}_j^{(i)}$ is the *j*th prediction for the instance *i*

3. Calculate the *j*th predictor's weight

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

where η is the learning rate (by default, $\eta = 1$).

4. Update the instances' weights

$$\text{For } i = 1 \text{ to } m$$

Begin For *i*

$$\text{if } \left(\widehat{y}_j^{(i)} \neq y^{(i)}\right) \text{ then } w^{(i)} = w^{(i)} \exp\left(\alpha_j\right)$$

End For *i*

5. Normalize the weights

```
For $i = 1$ to $m$
Begin For $i$
    $w^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$
End For $i$
End For $j$
```

To predict using AdaBoost, for a new instance, the weak learners calculate in sequence a predicted value as either +1 (for first class) or –1 (for the second class); each prediction is weighted by the predictor's weight. The weighted sum is calculated; AdaBoost assigns the instance to the first class if the weighted sum is positive and to the second class otherwise. Classifying an instance x with AdaBoost with N predictors can be summarized as follows:

$$\widehat{y}(x) = \underset{k}{\operatorname{argmax}} \sum_{\substack{j=1 \\ \widehat{y}_j(x)=k}}^{N} \alpha_j$$

15.4.2 AdaBoost Example

Download the "Iris" file from the Weka datasets or from the Kaggle website using the following link: https://www.kaggle.com/uciml/iris. Open the file in Weka and choose the AdaBoost algorithm in the Classify tab (Fig. 15.3).

Check the AdaBoost parameters and get acquainted with them (you can use the More button for more information). Accept the default parameters. Explore particularly the Classifier parameter; you can choose classifiers other than the decision stump (Fig. 15.4). Choose cross-validation with tenfolds (Fig. 15.5) and click the Start button. The output window displays the AdaBoost results (Fig. 15.6).

AdaBoost has performed ten iterations of cross-validation, as per our request. There are 143 (95.33%) correctly classified instances and seven (4.73%) incorrectly classified ones. The root mean squared error (RMSE) that we are trying to minimize is 0.1729. We can notice that the class Iris-setosa was clearly identified with a perfect area under the curve (AUC) (i.e., ROC area). The AUCs for Iris-versicolor and Iris-virginica were 0.92 and 0.93, respectively, indicating a high ability of the model to classify both types of irises. The confusion table shows five Iris-versicolor incorrectly classified as Iris-virginica and two Iris-virginica incorrectly classified

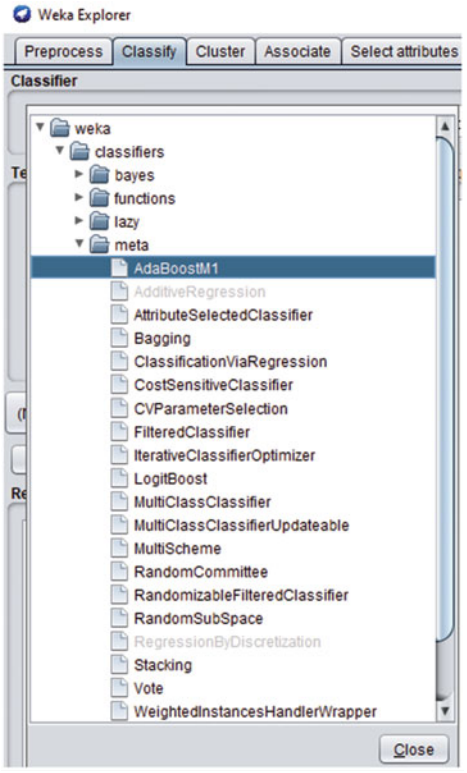


Fig. 15.3 AdaBoost classifier in Weka

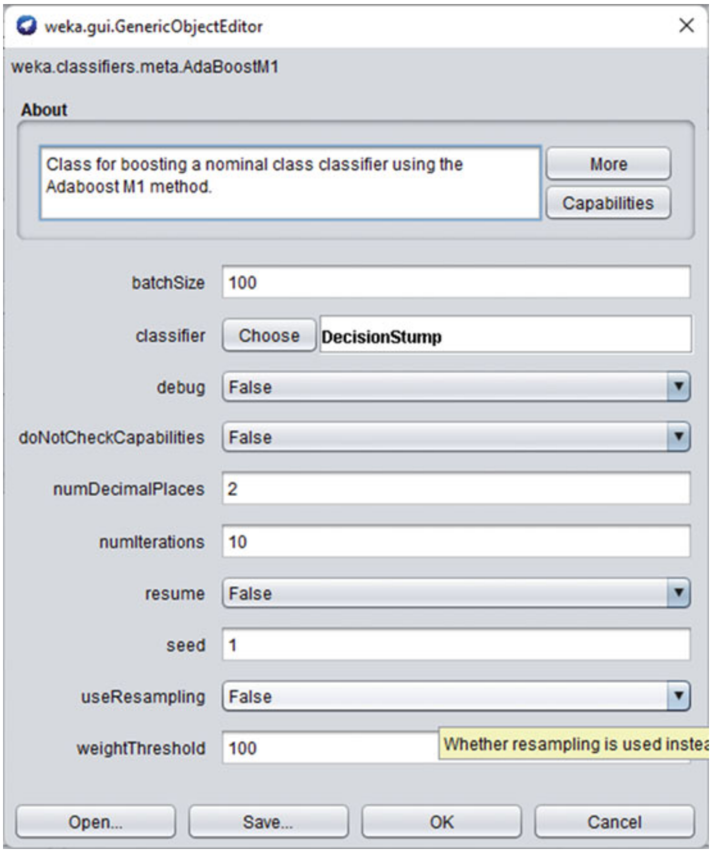
Weka Explorer

```
▼ weka
  ▼ classifiers
    ► bayes
    ► functions
    ► lazy
    ▼ meta
        AdaBoostM1
        AdditiveRegression
        AttributeSelectedClassifier
        Bagging
        ClassificationViaRegression
        CostSensitiveClassifier
        CVParameterSelection
        FilteredClassifier
        IterativeClassifierOptimizer
        LogitBoost
        MultiClassClassifier
        MultiClassClassifierUpdateable
        MultiScheme
        RandomCommittee
        RandomizableFilteredClassifier
        RandomSubSpace
        RegressionByDiscretization
        Stacking
        Vote
        WeightedInstancesHandlerWrapper
```

as Iris-versicolor. The window shows at the top that the classification was based on the petal length value 2.45 to differentiate between the Iris-setosa and the two other types, the decision being if petal length value is <2.45, then the flower is Iris-setosa.

15.5 Key Terms

1. Boosting
2. Hypothesis boosting
3. Weak learners
4. Stacking
5. Blender
6. Holdout sample
7. AdaBoost
8. Decision stump


weka.gui.GenericObjectEditor

weka.classifiers.meta.AdaBoostM1

About

Class for boosting a nominal class classifier using the Adaboost M1 method.

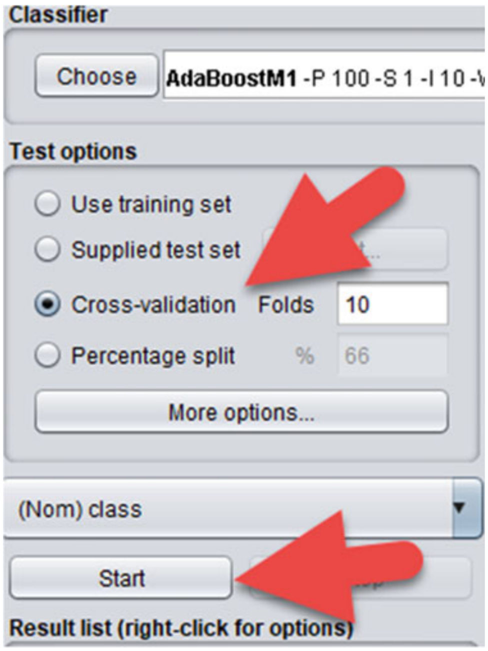
batchSize	100
classifier	DecisionStump
debug	False
doNotCheckCapabilities	False
numDecimalPlaces	2
numIterations	10
resume	False
seed	1
useResampling	False
weightThreshold	100

Whether resampling is used instea

Fig. 15.4 AdaBoost parameters in Weka

15.6 Test Your Understanding

1. Explain how stacking functions.
2. Describe boosting.
3. What are some of the challenges in boosting?
4. What is a decision stump?
5. Cite some of the hyperparameters of AdaBoost.


Classifier

AdaBoostM1 -P 100 -S 1 -I 10 -

Test options

Cross-validation Folds 10

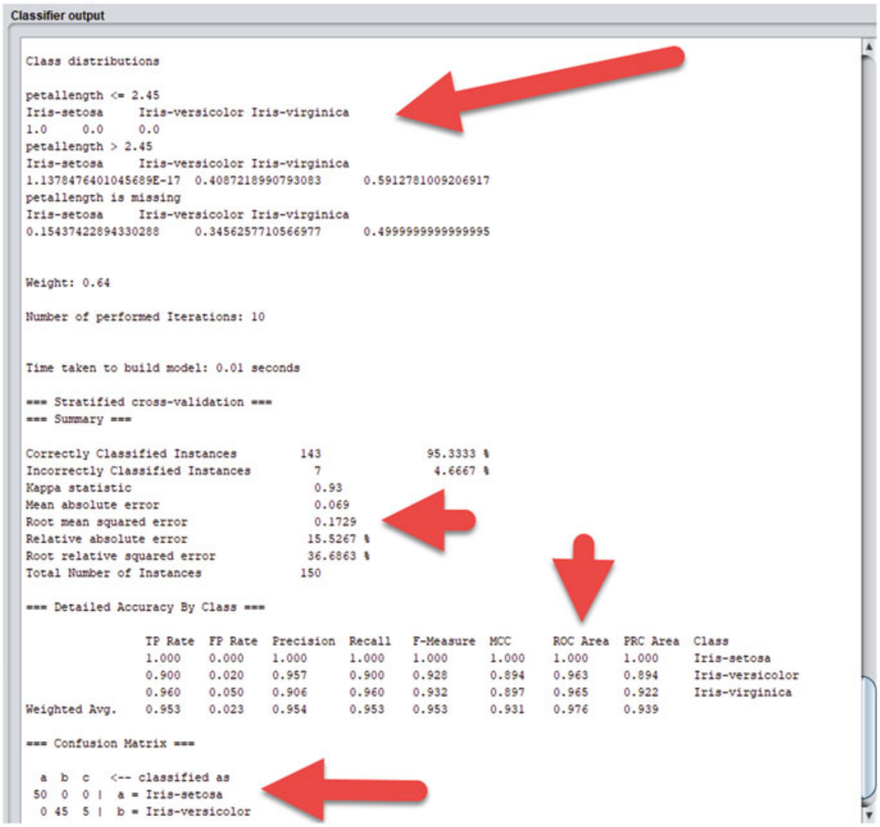
Percentage split % 66

(Nom) class

Fig. 15.5 Choosing to train the model using cross-validation with tenfolds

15.7 Read More

1. Abbruzzo, A., Tamburo, E., Varrica, D., Dongarrà, G., & Mineo, A. (2016). Penalized linear discriminant analysis and Discrete AdaBoost to distinguish human hair metal profiles: The case of adolescents residing near Mt. Etna. Chemosphere, 153, 100–106. doi: 10.1016/j.chemosphere.2016.03.029
2. Barczak, A. L. C., Johnson, M. J., & Messom, C. H. (2008). Empirical evaluation of a new structure for AdaBoost. Paper presented at the Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil. https://doi.org/10.1145/1363686.1364109
3. Bartlett, P. L., & Traskin, M. (2007). AdaBoost is Consistent. J. Mach. Learn. Res., 8, 2347–2368.
4. Cai, W., Qiu, L., Li, W., Yu, J., & Wang, L. (2019). Practical Fall Detection Algorithm based on Adaboost. Paper presented at the Proceedings of the 2019 4th International Conference on Biomedical Signal and Image Processing (ICBIP 2019), Chengdu, China. https://doi.org/10.1145/3354031.3354056
5. Carreras, X., Màrquez, L., & Padró, L. (2002). Named Entity Extraction using AdaBoost. Paper presented at the proceedings of the 6th conference on Natural language learning - Volume 20. https://doi.org/10.3115/1118853.1118857
6. Carreras, X., Màrquez, L., & Padró, L. (2003). A simple named entity extractor using AdaBoost. Paper presented at the Proceedings of the seventh conference



```
Classifier output

Class distributions

petallength <= 2.45
Iris-setosa     Iris-versicolor Iris-virginica
1.0     0.0     0.0
petallength > 2.45
Iris-setosa     Iris-versicolor Iris-virginica
1.1378476401045689E-17  0.4087218990793083      0.5912781009206917
petallength is missing
Iris-setosa     Iris-versicolor Iris-virginica
0.15437422894330288     0.3456257710566977      0.4999999999999995


Weight: 0.64

Number of performed Iterations: 10


Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         143               95.3333 %
Incorrectly Classified Instances         7                4.6667 %
Kappa statistic                          0.93
Mean absolute error                      0.069
Root mean squared error                  0.1729
Relative absolute error                 15.5267 %
Root relative squared error             36.6863 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.900    0.020    0.957      0.900    0.928      0.894    0.963     0.894     Iris-versicolor
                 0.960    0.050    0.906      0.960    0.932      0.897    0.965     0.922     Iris-virginica
Weighted Avg.    0.953    0.023    0.954      0.953    0.953      0.931    0.976     0.939

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 45  5 |  b = Iris-versicolor
```

Fig. 15.6 AdaBoost results when run on the iris dataset

on Natural language learning at HLT-NAACL 2003 - Volume 4, Edmonton, Canada. https://doi.org/10.3115/1119176.1119197

7. Chen, Y., Li, X., & Sun, W. (2020). Research on Stock Selection Strategy Based on AdaBoost Algorithm. Paper presented at the Proceedings of the 4th International Conference on Computer Science and Application Engineering, Sanya, China. https://doi.org/10.1145/3424978.3425084
8. Du, N., Li, K., Mahajan, S. D., Schwartz, S. A., Nair, B. B., Hsiao, C. B., & Zhang, A. (2011). Gene Co-Adaboost: a semi-supervised approach for classifying gene expression data. Paper presented at the Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine, Chicago, Illinois. https://doi.org/10.1145/2147805.2147892
9. Frost, J., Beekers, N., Hengst, B., & Vendeloo, R. (2012). Meeting cancer patient needs: designing a patient platform. Paper presented at the CHI '12 Extended Abstracts on Human Factors in Computing Systems, Austin, Texas, USA. https://doi.org/10.1145/2212776.2223806

10. Gutiérrez-Tobal, G. C., Álvarez, D., Del Campo, F., & Hornero, R. (2016). Utility of AdaBoost to Detect Sleep Apnea-Hypopnea Syndrome From Single-Channel Airflow. IEEE Trans Biomed Eng, 63(3), 636–646. doi: 10.1109/tbme.2015.2467188
11. He, B., Huang, C., Sharp, G., Zhou, S., Hu, Q., Fang, C., . . . Jia, F. (2016). Fast automatic 3D liver segmentation based on a three-level AdaBoost-guided active shape model. Med Phys, 43(5), 2421. doi: 10.1118/1.4946817
12. Hsu, K.-W. (2017). Heterogeneous AdaBoost with stochastic algorithm selection. Paper presented at the Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, Beppu, Japan. https://doi.org/10.1145/3022227.3022266
13. Hu, W., & Hu, W. (2005). Network-Based Intrusion Detection Using Adaboost Algorithm. Paper presented at the Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence. https://doi.org/10.1109/WI.2005.107
14. Kadlček, F., & Fučík, O. (2013). Fast and energy efficient AdaBoost classifier. Paper presented at the Proceedings of the 10th FPGAworld Conference, Stockholm, Sweden. https://doi.org/10.1145/2513683.2513685
15. Memari, N., Ramli, A. R., Bin Saripan, M. I., Mashohor, S., & Moghbel, M. (2017). Supervised retinal vessel segmentation from color fundus images based on matched filtering and AdaBoost classifier. PLoS One, 12(12), e0188939. doi: 10.1371/journal.pone.0188939
16. Mukherjee, I., Rudin, C., & Schapire, R. E. (2013). The rate of convergence of AdaBoost. J. Mach. Learn. Res., 14(1), 2315–2347.
17. Park, S. Y., & Chen, Y. (2017). Patient Strategies as Active Adaptation: Understanding Patient Behaviors During an Emergency Visit. Paper presented at the Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, Colorado, USA. https://doi.org/10.1145/3025453.3025978
18. Reiss, A., Hendeby, G., & Stricker, D. (2013). Confidence-based multiclass AdaBoost for physical activity monitoring. Paper presented at the Proceedings of the 2013 International Symposium on Wearable Computers, Zurich, Switzerland. https://doi.org/10.1145/2493988.2494325
19. Rudin, C., & Schapire, R. E. (2009). Margin-based Ranking and an Equivalence between AdaBoost and RankBoost. J. Mach. Learn. Res., 10, 2193–2232.
20. Saravanakumar, S., & Thangaraj, P. (2019). A Computer Aided Diagnosis System for Identifying Alzheimer's from MRI Scan using Improved Adaboost. J Med Syst, 43(3), 76. doi: 10.1007/s10916-018-1147-7
21. Song, X., Rui, T., Zha, Z., Wang, X., & Fang, H. (2015). The AdaBoost algorithm for vehicle detection based on CNN features. Paper presented at the Proceedings of the 7th International Conference on Internet Multimedia Computing and Service, Zhangjiajie, Hunan, China. https://doi.org/10.1145/2808492.2808497

22. Sun, J., Wang, F., Hu, J., & Edabollahi, S. (2012). Supervised patient similarity measure of heterogeneous patient records. SIGKDD Explor. Newsl., 14(1), 16–24. doi: 10.1145/2408736.2408740
23. Thongkam, J., Xu, G., Zhang, Y., & Huang, F. (2008). Breast cancer survivability via AdaBoost algorithms. Paper presented at the Proceedings of the second Australasian workshop on Health data and knowledge management - Volume 80, Wollongong, NSW, Australia.
24. Wang, B., Qi, Z., Chen, S., Liu, Z., & Ma, G. (2017). Multi-vehicle detection with identity awareness using cascade Adaboost and Adaptive Kalman filter for driver assistant system. PLoS One, 12(3), e0173424. doi: 10.1371/journal.pone.0173424
25. Wang, M. Y., Li, P., & Qiao, P. L. (2016). The Virtual Screening of the Drug Protein with a Few Crystal Structures Based on the Adaboost-SVM. Comput Math Methods Med, 2016, 4809831. doi: 10.1155/2016/4809831
26. Wang, Q., & Wei, X. (2020). The Detection of Network Intrusion Based on Improved Adaboost Algorithm. Paper presented at the Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy, Nanjing, China. https://doi.org/10.1145/3377644.3377660
27. Wang, Y., Ru, J., Jiang, Y., & Zhang, J. (2019). Adaboost-SVM-based probability algorithm for the prediction of all mature miRNA sites based on structured-sequence features. Sci Rep, 9(1), 1521. doi: 10.1038/s41598-018-38048-7
28. Yang, Y., Liu, C., & Liu, N. (2019). Credit Card Fraud Detection based on CSat-Related AdaBoost. Paper presented at the Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition, Beijing, China. https://doi.org/10.1145/3373509.3373548
29. Yousefi, M., Yousefi, M., Ferreira, R. P. M., Kim, J. H., & Fogliatto, F. S. (2018). Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments. Artif Intell Med, 84, 23–33. doi: 10.1016/j.artmed.2017.10.002

15.8 Lab

15.8.1 A Working Example in Python

The heart dataset that will be used for this lab can be downloaded from the following link: https://www.kaggle.com/code/ysthehurricane/heart-failure-prediction-using-adaboost-xgboost/data.

That dataset contains 11 features that will be used to predict heart disease events:

- Age: person's age in years
- Sex: person's gender
- ChestPainType: chest pain type
- RestingBP: resting blood pressure in mm Hg

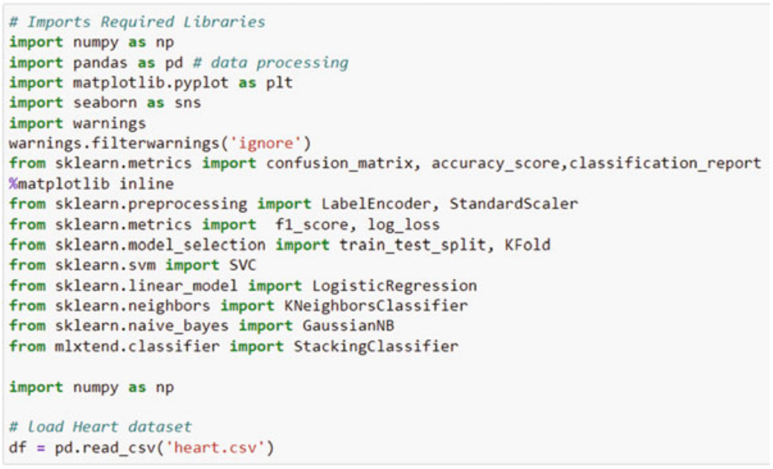
- Cholesterol: serum cholesterol in mm/dL
- FastingBS: blood sugar measurement on fasting
- RestingECG: electrocardiogram results in resting
- MaxHR: maximum heart rate achieved
- ExerciseAngina: exercise-induced angina flag
- Oldpeak: old peak
- ST_Slope: the slope of the peak exercise
- HeartDisease: target class (1 for having heart disease and 0 for not)

15.8.1.1 Loading Heart Dataset

We start by importing the required libraries and loading the heart dataset (Fig. 15.7). When you run the code, if you have not installed previously a needed library you will receive an error message stating that the module was not found, in such cases you need install the missing library using pip.

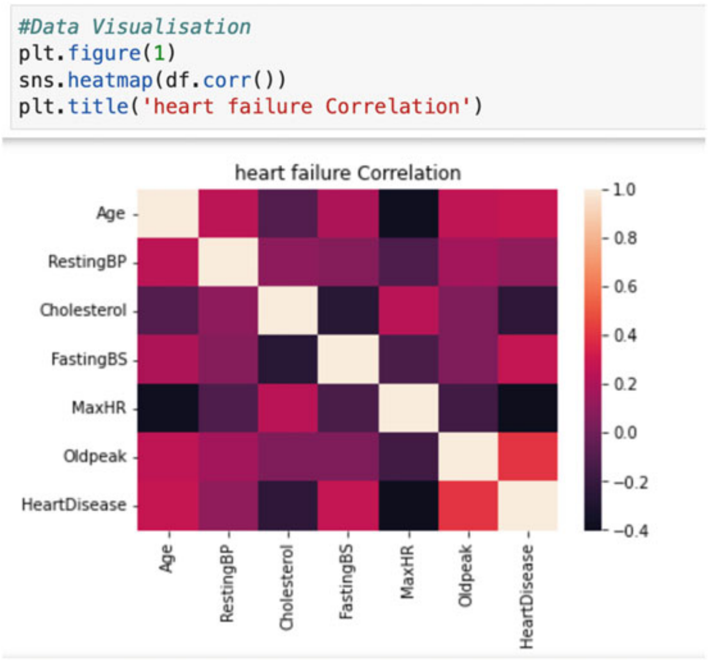
15.8.1.2 Visualizing Heart Dataset

The next step is to explore the heart dataset visually. We have opted to display the plot heatmap correlations between features' pairs (Fig. 15.8).



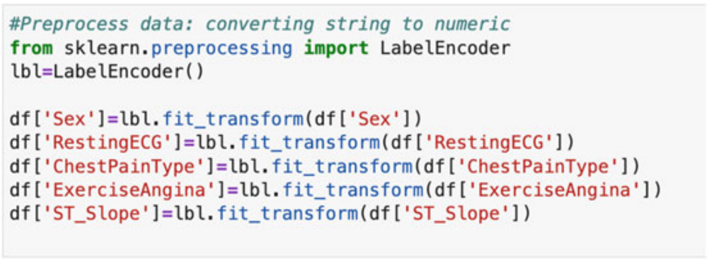
```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
%matplotlib inline
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import  f1_score, log_loss
from sklearn.model_selection import train_test_split, KFold
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from mlxtend.classifier import StackingClassifier

import numpy as np

# Load Heart dataset
df = pd.read_csv('heart.csv')
```

Fig. 15.7 Loading heart dataset into pandas



```
#Data Visualisation
plt.figure(1)
sns.heatmap(df.corr())
plt.title('heart failure Correlation')
```

Fig. 15.8 Visualizing heart dataset in heatmap



```
#Preprocess data: converting string to numeric
from sklearn.preprocessing import LabelEncoder
lbl=LabelEncoder()

df['Sex']=lbl.fit_transform(df['Sex'])
df['RestingECG']=lbl.fit_transform(df['RestingECG'])
df['ChestPainType']=lbl.fit_transform(df['ChestPainType'])
df['ExerciseAngina']=lbl.fit_transform(df['ExerciseAngina'])
df['ST_Slope']=lbl.fit_transform(df['ST_Slope'])
```

Fig. 15.9 Preprocess data by mapping string values into numeric ones

15.8.1.3 Preprocess Data

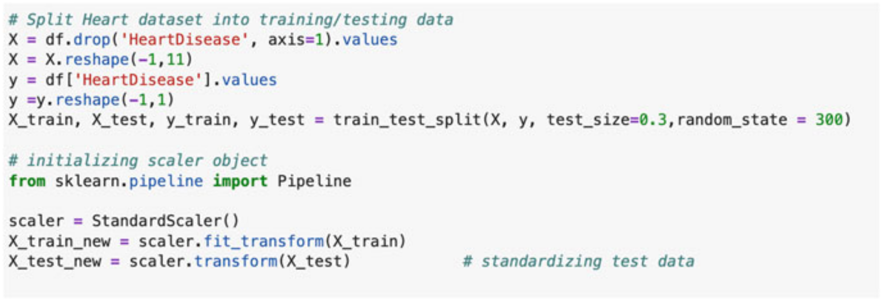
The next step is to convert string values to numeric ones. We have used the LabelEncoder do so (Fig. 15.9), can you achieve the same result using a different approach? Try.

15.8.1.4 Split and Scale Data

We can now choose the features and target, split the original dataset into training and testing datasets and standardize both (Fig. 15.10).

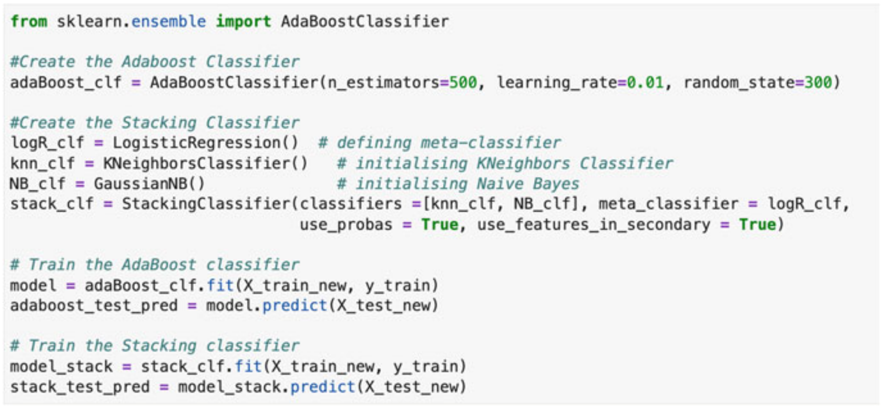
15.8.1.5 Create AdaBoost and Stacking Models

We will use AdaBoost to create a boosting model with a learning_rate=0.01 and n_estimators=500. We will also use a stacking approach using k-nearest neighbors and Gaussian naïve Bayes algorithms as classifiers and logistic regression as a metaclassifier. Then, we train both models on the training dataset and make associated predictions using the testing dataset (Fig. 15.11).



```
# Split Heart dataset into training/testing data
X = df.drop('HeartDisease', axis=1).values
X = X.reshape(-1,11)
y = df['HeartDisease'].values
y =y.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state = 300)

# initializing scaler object
from sklearn.pipeline import Pipeline

scaler = StandardScaler()
X_train_new = scaler.fit_transform(X_train)
X_test_new = scaler.transform(X_test)          # standardizing test data
```

Fig. 15.10 Split and scale heart dataset

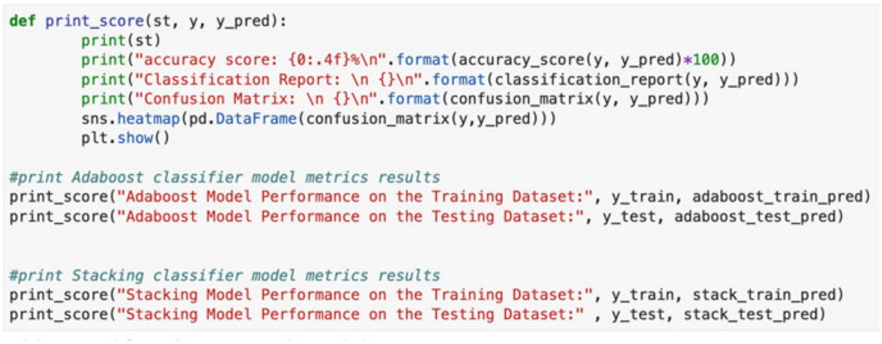


```
from sklearn.ensemble import AdaBoostClassifier

#Create the Adaboost Classifier
adaBoost_clf = AdaBoostClassifier(n_estimators=500, learning_rate=0.01, random_state=300)

#Create the Stacking Classifier
logR_clf = LogisticRegression()  # defining meta-classifier
knn_clf = KNeighborsClassifier()   # initialising KNeighbors Classifier
NB_clf = GaussianNB()              # initialising Naive Bayes
stack_clf = StackingClassifier(classifiers =[knn_clf, NB_clf], meta_classifier = logR_clf,
                               use_probas = True, use_features_in_secondary = True)

# Train the AdaBoost classifier
model = adaBoost_clf.fit(X_train_new, y_train)
adaboost_test_pred = model.predict(X_test_new)

# Train the Stacking classifier
model_stack = stack_clf.fit(X_train_new, y_train)
stack_test_pred = model_stack.predict(X_test_new)
```

Fig. 15.11 Create AdaBoost and stacking models

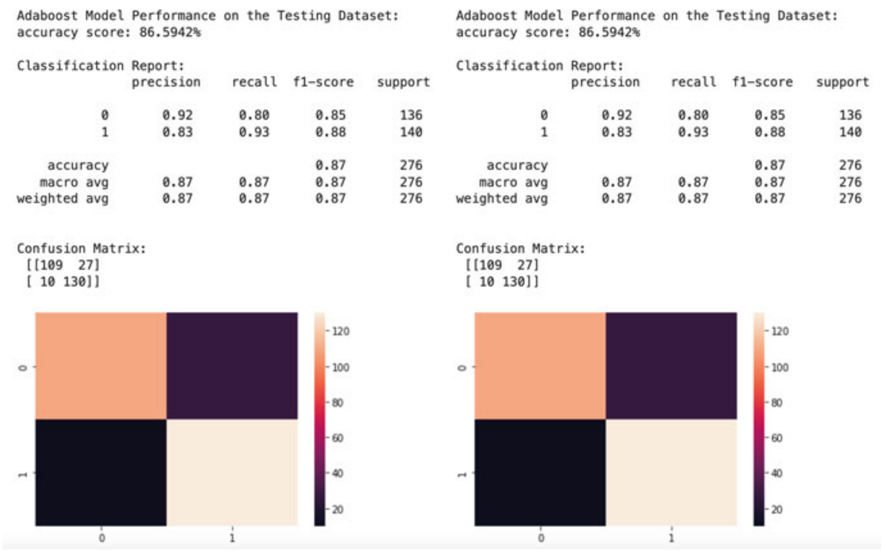
15.8.1.6 Evaluate the AdaBoost and the Stacking Models

The next step is to evaluate the performance of the AdaBoost and Stacking models. We opted to show in this lab the performance on the training and testing datasets for exploration/learning purposes. Figure 15.12 shows the code and Figs. 15.13 and 15.14 show the performance results displayed for AdaBoost and Stacking, respectively.



```python
def print_score(st, y, y_pred):
        print(st)
        print("accuracy score: {0:.4f}%\n".format(accuracy_score(y, y_pred)*100))
        print("Classification Report: \n {}\n".format(classification_report(y, y_pred)))
        print("Confusion Matrix: \n {}\n".format(confusion_matrix(y, y_pred)))
        sns.heatmap(pd.DataFrame(confusion_matrix(y,y_pred)))
        plt.show()

#print Adaboost classifier model metrics results
print_score("Adaboost Model Performance on the Training Dataset:", y_train, adaboost_train_pred)
print_score("Adaboost Model Performance on the Testing Dataset:", y_test, adaboost_test_pred)


#print Stacking classifier model metrics results
print_score("Stacking Model Performance on the Training Dataset:", y_train, stack_train_pred)
print_score("Stacking Model Performance on the Testing Dataset:" , y_test, stack_test_pred)
```

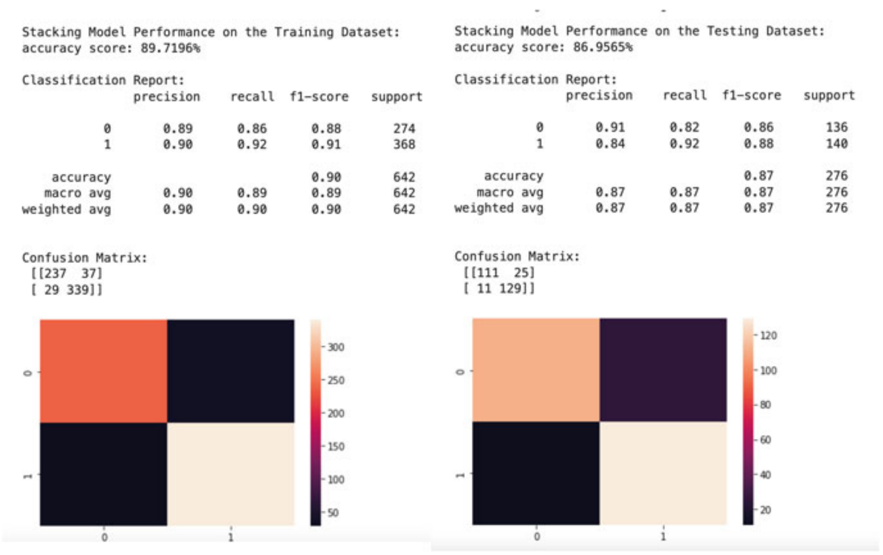
Fig. 15.12 Calculating accuracy and confusion matrix for AdaBoost model



```
Adaboost Model Performance on the Testing Dataset:
accuracy score: 86.5942%

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.80      0.85       136
           1       0.83      0.93      0.88       140

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276


Confusion Matrix:
 [[109  27]
 [ 10 130]]
```

```
Adaboost Model Performance on the Testing Dataset:
accuracy score: 86.5942%

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.80      0.85       136
           1       0.83      0.93      0.88       140

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276


Confusion Matrix:
 [[109  27]
 [ 10 130]]
```

Fig. 15.13 AdaBoost Model Performance on the training and testing Datasets


```
Stacking Model Performance on the Training Dataset:
accuracy score: 89.7196%

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88       274
           1       0.90      0.92      0.91       368

    accuracy                           0.90       642
   macro avg       0.90      0.89      0.89       642
weighted avg       0.90      0.90      0.90       642

Confusion Matrix:
 [[237  37]
 [ 29 339]]
```

```
Stacking Model Performance on the Testing Dataset:
accuracy score: 86.9565%

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.82      0.86       136
           1       0.84      0.92      0.88       140

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276

Confusion Matrix:
 [[111  25]
 [ 11 129]]
```

Fig. 15.14 Stacking Model Performance on the training and testing Datasets

15.8.1.7 Optimizing the Stacking and AdaBoost Models

The models' performances on the testing datasets are fair. Let us explore the performance of the optimized models after hyperparameter tuning. The results for the Stacking and AdaBoost models are presented in Figs. 15.15 and 15.16, respectively.

15.8.2 Do It Yourself

15.8.2.1 The Heart Disease Dataset Revisited

1. Have you noticed any possible overfitting in the example above?
2. Did you obtain the same results when you run your code? What do you think about those results?
3. During the evaluation step above, we have just applied the models to the testing dataset. That is not the best option. What is a better approach?
4. Use cross-validation to redo the evaluation step.

15.8.2.2 The Iris Dataset

Download the iris dataset and do the following:

```
from sklearn.model_selection import GridSearchCV

sclf = StackingClassifier(classifiers=[knn_clf, NB_clf], meta_classifier=logR_clf)
params = {'kneighborsclassifier__n_neighbors': [1, 5],
          'meta_classifier__C': [0.1, 10.0]}
grid = GridSearchCV(estimator=sclf,
                    param_grid=params,
                    cv=5,
                    refit=True)
grid.fit(X_test_new, y_test)
optimal_Stacking_pred= grid.predict(X_test_new)
print_score("Optimized Stacking Model Performance on the Testing Dataset:", y_test, optimal_Stacking_pred)
```

```
Optimized Stacking Model Performance on the Testing Dataset:
accuracy score: 88.0435%

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.83      0.87       136
           1       0.85      0.93      0.89       140

    accuracy                           0.88       276
   macro avg       0.88      0.88      0.88       276
weighted avg       0.88      0.88      0.88       276


Confusion Matrix:
 [[113  23]
 [ 10 130]]
```

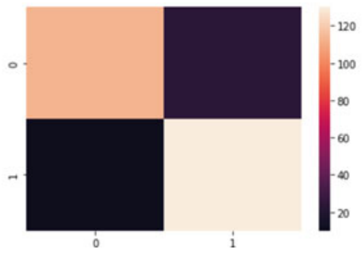


Fig. 15.15 Optimal stacking model performance

1. Load the dataset into pandas.
2. Visualize the dataset and calculate the highest correlations.
3. Preprocess the data.
4. Split the data.
5. Create an AdaBoost model.
6. Evaluate the AdaBoost model.
7. Optimize the AdaBoost model.
8. Create a stack model.
9. Evaluate the stack model.
10. Optimize the stack model.
11. Compare the results between both models and deduce a conclusion.

```
ada_params={'n_estimators':[50,100,200],'learning_rate':[.001,0.01,.1]}
gsearch=GridSearchCV(estimator=adaBoost_clf,param_grid=ada_params,scoring='accuracy',n_jobs=1,cv=5)
gsearch.fit(X_test_new, y_test)
optimal_boost_pred= grid.predict(X_test_new)
print_score("Optimized AdaBoost Model Performance on the Testing Dataset:" , y_test, optimal_boost_pred)
```

```
Optimized AdaBoost Model Performance on the Testing Dataset:
accuracy score: 88.0435%

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.83      0.87       136
           1       0.85      0.93      0.89       140

    accuracy                           0.88       276
   macro avg       0.88      0.88      0.88       276
weighted avg       0.88      0.88      0.88       276


Confusion Matrix:
 [[113  23]
 [ 10 130]]
```

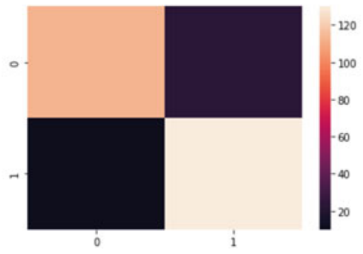


Fig. 15.16 Optimal AdaBoost model performance

15.8.3 *Do More Yourself*

- https://www.kaggle.com/code/treina/titanic-with-adaboost/data
- https://www.kaggle.com/datasets/zaurbegiev/my-dataset
- https://www.kaggle.com/code/sid321axn/house-price-prediction-gboosting-adaboost-etc/data

References

1. A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly Media, Sebastopol, CA, 2019)
2. Y. Liu, *Python Machine Learning By Example: Build Intelligent Systems Using Python, TensorFlow 2, PyTorch, and Scikit-Learn, 3rd Edition (Kindle Edition)* (Packt, 2020)
3. D.H. Wolpert, Stacked generalization. Neural Netw. **5**(2), 241–259 (1992). https://doi.org/10.1016/S0893-6080(05)80023-1
4. R.E. Schapire, Y. Freund, *Boosting: Foundations and Algorithms* (MIT Press, Cambridge, MA, 2014)

Chapter 16
Future Directions and Ethical Considerations

16.1 Introduction

Artificial intelligence and machine learning have significantly advanced in the last few years and are expected to continue a trajectory of increased adoption and impact. This growth is driven by a number of technological and social developments over the past few years. According to Gartner, a technology research and consulting firm, the drivers for growth in AI are (1) the increasing volume and availability of big data and the developments in parallel processing systems that can cost-effectively store and process data at massive scale; (2) the advancements in computer hardware, particularly the emergence of powerful graphics processing units (GPUs) for complex computations; (3) the development of new machine learning (ML) techniques; (4) the emergence of cloud computing, which enables faster experimentation with and operationalization of AI with lower complexity; and (5) the vibrant open-source ecosystem, which has enabled many deep learning frameworks and resulted in an explosion of startups [1].

According to Gartner, the worldwide AI software market is expected to reach $62 billion in 2022, an increase of 21.3% from 2021, to be spent mostly on knowledge management, virtual assistants, autonomous vehicles, digital workplaces, and crowdsourced data [2]. While the interest in and spending on AI are increasing, there is still a lag in AI maturity and a lack of making the new technology part of organizations' standard operations. This is mainly due to reluctance to embrace AI, lack of trust in AI, and difficulties delivering business value from AI [2]. Nonetheless, new technological developments and applications of artificial intelligence and machine learning are expected to emerge, which will lead to further growth and adoption by organizations. Gartner predicts that it will take until 2025 for half of organizations worldwide to reach the "stabilization stage" of their AI maturity model [2].

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_16

16.2 Current AI Applications

Artificial intelligence and machine learning applications can be found in many fields and industries. Their applications can be found in healthcare, water management, agriculture, animal farming, electrical power grids, car insurance, banking, marketing, customer churn analysis, smart buildings, human resource management, traffic management, fraud detection, and many more. A well-known classic example in healthcare is Google Flu Trends. Based on the large volume of data from individuals searching for information about influenza when they are sick, Google is able to detect trends and estimate the current flu outbreak levels in the USA and other regions. Not only is Google's estimate highly accurate, but it is also 2 weeks faster than the traditional method used by the US Centers for Disease Control. According to *Fortune Magazine* [3], AI has changed business in many ways. AI is playing a role in hiring, such as screening candidates, and in some cases making managerial decisions. Companies like Unilever, a multinational consumer goods company, and Vodaphone, a telecommunication company, are using AI to build a better workforce. AI is used in banking and investment for picking stocks and approving loans and mortgages. Banks like Wells Fargo rely on AI to streamline their loan approval process. Financial services firms like Charles Schwab use the technology to help customers invest their savings, while many Wall Street firms are relying on AI for selecting stocks to invest in. Car manufacturers like BMW and Mercedes-Benz use collaborative robots or Cobots, which are computer-controlled robotic devices, to assist human workers in assembling cars more efficiently. High-tech firms like SAP are also using AI to develop software to spot worker fatigue before accidents happen. AI is also used in cybersecurity, helping banks detect fraud and money laundering and helping businesses prevent hackings. Chatbots, which are AI-powered computer programs that simulate and process human conversations, are commonly used for customer service. AI is also used by companies like Home Depot and Walmart to help their customers find products more quickly. Finally, AI will help in detecting diseases and illnesses, and in developing drugs and treatments. Companies like IBM, with its Watson Health AI system, are pioneers in that field [3]. In agriculture, animal farmers use artificial intelligence and machine learning with big data and sensor data, including local weather data, air quality data, voice signals of animals, visual data of various animal movements, food intake, sleep cycles, and other animal behavior data, to lower production costs, increase efficiencies, enhance animal welfare, and raise more animals per hectare [4]. For example, technology allows farmers to employ different facial recognition methods using non-invasive imaging systems to recognize faces of individual animals in a real farm setting with 96.7% accuracy. Such systems can replace inefficient and costly RFID tags and help farmers monitor their animals efficiently at scale, significantly reducing their costs and labor requirements [4]. NotCo, a Chilean company founded in 2015, designed an algorithm that uses AI to predict which vegetal ingredients can be mixed and which plants can be combined to create food products similar to milk, mayo, or meat in their color, smell, texture, flavor, and nutritional content. The company's Giuseppe, their AI chef, also known as Genius G, is capable of analyzing the animal base molecular structure and replicating it by using plants only, all 300,000 species of them. It uses machine learning to automatically learn and improve based on existing data. The company's products NotMayo, NotMilk, NotIce Cream, NotBurger, and NotChicken can be found in many grocery chains and stores in North and South America [5].

16.3 Future Directions

According to *Forbes Magazine*, there are seven areas where breakthroughs in AI are expected in 2022 [6]. The first area is the augmented workforce, where machines and humans will be working together, with the former using smart and cognitive functionality to boost the latter's abilities and skills. For example, in engineering, AI tools help in preventive maintenance by predicting when machines need servicing and repair. In marketing, AI tools identify customer leads for salespersons to target and follow. Major advances are also expected in the area of language modeling, which is the process that allows machines to understand and communicate with humans in a language they understand. OpenAI, an AI research and deployment company, is working on the fourth version of its Generative Pre-trained Transformer (GPT-4), getting closer to allowing machines to hold conversations that may be undifferentiated from those with a human. It might also take natural human languages and turn them into computer instructions that can run software applications. The second predicted area of growth of AI is cybersecurity, where hacking and cybercrime pose a major risk to society. The increasing number of devices connected to the Internet has increased the number of potential points of failure and made the network more complex to manage and secure. AI can be used to analyze network traffic and can use smart algorithms to recognize patterns that suggest potential criminal intentions. AI is also expected to play a role in the metaverse, or the future digital and virtual environment where humans will interact together and have immersive experiences. AI is expected to play a role in creating online environments based on humans' impulsive needs and providing help with tasks. The fourth area of growth is in low-code or no-code AI, where humans will be able to create smart programs or applications by plugging together premade modules and complementing them with domain-specific data and knowledge. Natural language processing and language modeling will allow us to use our voice or written instructions to build smart applications, making AI reachable to a much larger population. AI is the brain guiding autonomous vehicles like cars and boats. Advances in this area are evident, and it is expected that autonomous cars with full self-driving capabilities may be ready for general use in 2022. Car companies like Tesla, GM, and Ford and technology companies like Apple and Google's Waymo are all expected to make major advances in this area. In 2022, the Mayflower Autonomous Ship, designed by IBM in partnership with PromAre, a non-profit organization promoting marine research and exploration, is expected to cross the Atlantic autonomously. Finally, AI is expected to make more headway in an area that has been considered explicitly a human skill: creativity. AI is expected to create art in the form of music, drawings, and poetry, in addition to performing routine creative tasks like writing articles and newsletters and designing logos and infographics [6].

In the longer run, Gartner has identified a number of significant emergent AI trends that will shape the future of AI in the enterprise and will have an impact on us by 2025. We will describe four of them: (1) democratized AI, or making AI accessible to a wide set of users beyond experts; (2) edge AI, or harnessing AI for real-time analytics closer to data sources; (3) responsible AI, or the ethical use of AI; and (4) generative AI, or the use of AI to generate new artifacts and create new products [1].

16.3.1 Democratized AI

Democratized AI consists of techniques and tools aimed at improving the construction of AI applications while reducing their dependence on humans and expert knowledge. It will make AI accessible to a wide set of users beyond experts. Democratized AI will affect everyone's life, both at work and at home. It will assist professionals by automatically identifying relationships, trends, patterns, and exceptions hidden in large datasets. It will augment and assist workers by complementing their knowledge. Applications will be more intelligent and will support decision-making, in addition to processing transactions [1].

Democratized AI consists of four sub-trends. Everyday AI is the seamless integration of AI techniques into everyday productivity tools for email sorting and routing, automated meetings scheduling, proactive information distribution, and many more. Augmented AI is where AI techniques provide additional functionality to enterprise systems, for example, resulting in intelligent automation, data-driven insights and guided recommendations, improved productivity and decision-making, and even a personalized interface for users. Citizen AI, or easier-to-use AI, would enable new users like business analytics professionals to quickly tune highly accurate models with limited coding knowledge. Finally, human-centered AI (HCAI) calls for AI to benefit people and society. It is a model of people and AI working together to enhance cognitive performance, including learning and decision-making, and is sometimes referred to as "augmented intelligence" [1].

16.3.2 Edge AI

Edge AI refers to the use of AI techniques embedded in Internet of Things (IoT) sensors, gateways, and edge servers in a wide range of applications, including autonomous vehicles and streaming analytics. Edge AI can be used to manage and improve business operations and processes and improve asset management and operational intelligence in areas like mining, construction, healthcare diagnostic centers, and other businesses with high-value assets. An application in energy and manufacturing is the AI-based visual inspection of video streams or images, where edge AI is closer to the data sources for real-time training and inference in a highly distributed model [1].

Edge AI technological developments and applications are enabled by the advances and growth in AI, the Internet of Things (IoT), and big data processing. The drivers for edge AI adoption are the need for real-time processing, the growth in IoT endpoints due to a decrease in price, and enterprise infrastructure and workforces becoming more distributed. Gartner predicts that by 2025, 75% of data will be generated outside enterprises' centralized data centers, and 11.7 billion IoT devices will be connected. The demand for data processing at the point of data creation for real-time insights will push AI applications to the edge, closer to the IoT endpoints [1].

16.3.3 Responsible AI

Responsible artificial intelligence refers to making appropriate ethical choices when adopting AI in relation to societal value, trust, transparency, fairness, bias mitigation, accountability, safety, privacy, regulatory compliance, and many more. Organizations will follow responsible AI practices by ensuring positive and accountable AI development and exploitation [1].

The drivers behind responsible AI are the growing guidelines by governments, regulatory authorities, and industry bodies. AlgorithmWatch, a "non-profit research and advocacy organization that is committed to watch, unpack and analyze automated decision-making (ADM) systems and their impact on society," has cataloged more than 80 AI ethics guidelines. Business Roundtable, an "association of chief executive officers of America's leading companies working to promote a thriving U.S. economy and expanded opportunity for all Americans through sound public policy," has published a roadmap and policy guidelines for responsible AI. Another driver is consumer trust, which is critical for any company and may be eroded if it uses AI without paying attention to ethics. Finally, the pressure to meet sustainability goals is pushing companies to use AI to assess and reduce the impact of climate change [1].

16.3.4 Generative AI

Generative AI is a term that describes AI techniques that are able to learn from data a representation of artifacts and use it to generate new unique artifacts. Generative AI can produce synthetic data, models of physical objects, and novel media content (e.g., text, audio, image, video). Photorealistic images of people and things are the most commonly seen outcomes of generative AI today. It can also be used to generate computer code, thus reducing the burden on human developers and software engineers. It can be used to create synthetic data, thereby protecting privacy and controlling bias in AI training data. It can be used in drug discovery or materials design for a much faster process than traditional methods. It can also be used to dub films in different languages using the original actors' voices [1].

However, these AI developments and the spread of AI applications within organizations and in society will come with increased ethical concerns, both old and new. Generative AI when used for content creation with malicious intent like deepfakes can result in negative outcomes by spreading false information and bringing harm to individuals and society. Increased scrutiny will be required, and AI experts need to ensure the ethical, bias-free, and responsible implementation of AI models. Edge AI, for example, should be applied without the violating privacy needs of the IoT endpoints and physical locations [1].

16.4 Ethical Concerns

The ability of information technology to collect, analyze, transfer, and store massive amounts of data has given rise to several concerns. A major concern is the risk to data security from hackers and data breaches. In the past few years, data breaches have grown in size and frequency, with one of the largest breaches affecting almost 80 million people. The exposed data were highly sensitive and included patients' identifying information, health insurance information, and medical histories [7]. In addition to hacking databases and stealing data, criminals may be able to remotely access certain medical devices, such as implantable cardiac pacemakers, if they have IoT capabilities but lax cybersecurity and cause serious physical harm to patients [16]. Another major concern is related to privacy and the risk that personal sensitive information may be unlawfully accessed or shared. The personal records of hundreds of high school students in Australia were mistakenly published on their school's intranet, including medical and mental health conditions, learning disabilities, behavioral difficulties, and medications used [8]. Other ethical concerns and questions will arise with advances in AI and machine learning. AI in hiring raised serious concerns of discrimination when it was unveiled that Amazon, and possibly other high-tech companies, were not hiring women. This was due to a bias in the data caused by a lack of enough representation of women in these companies, which led its AI hiring algorithms to discriminate against women [9]. Another example of AI-generated discrimination is in law enforcement, where machine learning algorithms, like the ones used for criminal risk assessment, try to find patterns in data which may be historically biased against certain groups, like low-income and minority groups [10]. In addition to the bias inherent in the historical data used to train machine learning algorithms, there is also a risk of artificial intelligence making erroneous decisions due to flaws in algorithms leading to serious problems, like insurance companies denying coverage for individuals whose data predict that they will develop certain illnesses or behavioral problems in the future. A common AI application today is facial recognition, used in shopping centers, museums, casinos, and many public places like city streets. The technology is used by law enforcement agencies to match people against watchlists. However, the facial recognition technology does not reach 100% accuracy and does not work very well on people with darker skin, women, and children, putting almost half of the population of certain countries at risk of being misidentified and having to prove their innocence [11]. An additional ethical concern with AI is its potential ability to exceed the cognitive performance of humans, which could lead to mass unemployment and wealth redistribution, and to making decisions that humans cannot understand or control [12].

16.4.1 Ethical Frameworks

While the benefits of artificial intelligence and machine learning are evident and their adoption will continue to increase, it is important to find the proper safeguards and policies to protect the data and ensure its ethical use. It is important to make sure AI applications do not cause harm to individuals and society.

Ethical concerns have emerged over the years with the development and adoption of new technologies, and multiple ethical frameworks have been developed to guide their use. Table 16.1 presents an overview of some key ethical frameworks that can be used to address the ethical and moral issues with AI [12].

In this chapter, we focus on Richard Mason's (1986) [14] four categories of ethical issues for the information age, which are privacy, accuracy, property, and accessibility, otherwise known by the acronym PAPA. Mason introduced the PAPA framework to inform people about the threats to human dignity that the information age introduces. These issues were raised in 1986 but are still relevant today. The first ethical concern is privacy, or "what information should one be required to divulge about one's self to others? Under what conditions? What information should one be able to keep strictly to one's self?" ([14] page 5). Privacy is threatened by information technology's capacity for surveillance, communication, computation, storage, and retrieval. It is also threatened by the use of data in decision-making and developing policies. Accuracy concerns lead to the following questions: "Who is responsible for the authenticity, fidelity and accuracy of information? Similarly, who is to be held accountable for errors in information and how is the injured party to be made whole?" ([14] page 5). Misinformation can hurt individuals, especially when the one inaccurate (or even fake) information are in the hands of those in power and authority. This inaccuracy can have negative financial, professional, personal, and even health implications. Property is about "Who owns information? What are the just and fair prices for its exchange? Who owns the channels, especially the airways, through which information is transmitted? How should access to this scarce resource be allocated?" ([14] page 5). Intellectual property falls in this category, as it is extremely difficult and costly to create, but extremely easy and cheap to reproduce and share once it is digitally available. Unlike tangible property, information is

Table 16.1 Examples of ethical frameworks in AI

Source	Title	Ethical framework's basics
Belmont [13]	Ethical principles and guidelines in the protection of human subjects of research	1. Respect for subject: the right to decide whether to participate 2. Beneficence: do no harm to participants 3. Justice: fairly distribute costs and benefits of research
Mason [14]	Four ethical concerns of the information age	Issues with information technology: privacy, accuracy, property, and accessibility (PAPA)
Bentham [15]	An introduction to the principles of morals and legislation	Act utilitarianism: tally the consequences of each action first and then determine on a case-by-case basis whether an action is morally right or wrong Hedonistic utilitarianism: pleasure and pain are the only consequences that matter in determining whether the conduct is moral or not
Wallach [16]	Big data, machine learning, and the social sciences: Fairness, accountability, and transparency	Ethical principles: 1. Fairness: bias, fairness, and inclusion 2. Accountability 3. Transparency
Hursthouse and Pettigrove [17]	Virtue ethics	Having ethical thoughts and ethical characters
Sinnott-Armstrong [18]	Consequentialism	Engaging in actions that cause more good than harm
Alexander and Moore [19]	Deontological ethics	Conforming to rules, laws, and other statements of ethical duty (religious texts, industry codes of ethics)

Adapted from [12]

difficult to keep for oneself, and it is very difficult to get reimbursement from its use. Finally, accessibility ethical concerns lead to the following questions: "What information does a person or an organization have a right or a privilege to obtain, under what conditions and with what safeguards?" ([14] page 5). To have access to information and benefit from it, one should have certain prerequisite knowledge and economic level. For many people, economic or social factors prevent them from getting access to information and gaining literacy [14].

The moral requirement in Mason's framework was obvious: information technology should be used to protect and enhance human dignity. Mason affirmed the need for society to ensure everyone's right to fulfill their own human potential. He called for a new social contract where information systems do not unjustifiably breach a person's privacy. Information systems should communicate accurate information to avoid misinformation, and be accessible to avoid the information illiteracy and deprivation [14]. The same threats of information technology to human dignity that Mason described back in 1986, long before the wide spread of the Internet, still exist today. Artificial intelligence used by automated devices like Alexa and Siri for voice recognition can encroach on one's privacy by continuously "listening" and even recording our conversations. China's Social Credit System, which uses AI to track and evaluate citizens, creates serious privacy and accuracy concerns. AI has limited presence and applications in many developing countries due to a growing Digital Divide and is hence inaccessible to the population there. Like with any other information technology, the PAPA framework should be taken into consideration when AI is applied and adopted in order to enhance the dignity of mankind.

16.5 Conclusion

New growth in technologies is changing the way we process data and make decisions. Like other technologies, artificial intelligence and machine learning are here to stay; they provide immense benefits in many disciplines and fields, as we discovered in this book. However, AI and ML raise immense ethical concerns. They also raise societal issues; for example, a mathematician, Cathy O'Neil, felt compelled to write a book to address the impact of big data on inequality and democracy [20]; the book's name (*Weapons of Math Destruction*) reflects a general human acknowledgment and concern about these changes and a call to reflect on them [21].

Artificial intelligence and machine learning have immense benefits for individuals, organizations, and society. Professionals in some fields are feeling the change first; radiologists, for instance, are raising questions about the future of radiology and radiography [22]; an AI software that is able to read an image better than a trained human is not a far-fetched idea anymore, and a robot that can take an X-ray in a more precise way than humans can be feasible. If radiographers and radiologists do not disappear, their work will undergo an immense change; other fields will probably face similar challenges, society as a whole needs to make decisions, and citizens need to weigh in about the directions these technologies should be taking and how far they should go. Ultimately, if something can be created, it does not necessarily mean that it ought to be created; our future should be decided by our personal and collective efforts and reflections being translated into policies [21].

16.6 Key Terms

1. Democratized AI
2. Edge AI
3. Responsible AI

4. Generative AI
5. Ethical frameworks
6. PAPA
7. Privacy
8. Accuracy
9. Property
10. Accessibility

16.7 Test Your Understanding

1. What are the technological drivers behind the growth of AI?
2. What is a GPU, and what is its role in AI and ML?
3. How is AI benefitting from the prevalence of cloud computing?
4. What are some potential downsides to AI?
5. What role can ethical frameworks play in the context of AI?
6. What does the acronym PAPA stand for?
7. What questions does the PAPA framework try to answer?
8. What is Richard Mason's concern, and what does he call for?
9. What are some concerns with traditional information systems that apply to AI today?
10. What are new concerns that AI has generated?

16.8 Read More

1. Hagendorff, T. (2020). The ethics of AI ethics: An evaluation of guidelines. Minds and Machines, 30(1), 99–120.
2. Holzinger, A., Kieseberg, P., Weippl, E., & Tjoa, A. M. (2018, August). Current advances, trends and challenges of machine learning and knowledge extraction: from machine learning to explainable AI. In International Cross-Domain Conference for Machine Learning and Knowledge Extraction (pp. 1–8). Springer, Cham.
3. Lu, Y. (2019). Artificial intelligence: a survey on evolution, models, applications and future trends. Journal of Management Analytics, 6(1), 1–29.
4. Richardson, S. M., Petter, S., & Carter, M. (2021). Five ethical issues in the big data analytics age. Communications of the Association for Information Systems, (1), 18.
5. Ruiz-Real, J. L., Uribe-Toril, J., Torres, J. A., & De Pablo, J. (2021). Artificial intelligence in business and economics research: trends and future. Journal of Business Economics and Management, 22(1), 98–117.
6. Shinde, P. P., & Shah, S. (2018, August). A review of machine learning and deep learning applications. In 2018 Fourth international conference on computing communication control and automation (ICCUBEA) (pp. 1–6). IEEE.

7. Siau, K., & Wang, W. (2020). Artificial intelligence (AI) ethics: ethics of AI and ethical AI. Journal of Database Management (JDM), 31(2), 74–87.
8. Stahl, B. C. (2021). From PAPA to PAPAS and Beyond: Dealing with Ethics in Big Data, AI and other Emerging Technologies. Communications of the Association for Information Systems, 49(1), 20.

References

1. A. Chandrasekaran, B. Burke, E. Brethenoux, Building a digital future: emergent AI trends. *Gartner* (2022). [Online]. Available: https://www.gartner.com/document/4014200
2. *Gartner*, Gartner forecasts worldwide artificial intelligence software market to reach $62 billion in 2022. https://www.gartner.com/en/newsroom/press-releases/2021-11-22-gartner-forecasts-worldwide-artificial-intelligence-software-market-to-reach-62-billion-in-2022. Accessed 4 May 2022.
3. *Fortune*, 25 ways A.I. is changing business. https://fortune.com/2018/10/22/artificial-intelligence-ai-changing-business/. Accessed 7 May 2022.
4. S. Neethirajan, The role of sensors, big data and machine learning in modern animal farming. Sens. Bio-Sens. Res. **29**, 100367 (2020)
5. M.J.A. Ruiz, NotCo aims to create delicious plant-based food with AI. *TechAcute*. https://techacute.com/notco-aims-to-create-delicious-plant-based-food-with-ai/. Accessed 7 May 2022
6. B. Marr, The 7 biggest artificial intelligence (AI) trends in 2022. *Forbes*. https://www.forbes.com/sites/bernardmarr/2021/09/24/the-7-biggest-artificial-intelligence-ai-trends-in-2022/?sh=2e12ad522015. Accessed 5 May 2022
7. N. Lord, Top 10 biggest healthcare data breaches of all time. *Data Insider*, 25 June 2018 [Online]. Available: https://digitalguardian.com/blog/top-10-biggest-healthcare-data-breaches-all-time
8. Australian Associated Press, Melbourne student health records posted online in 'appalling' privacy breach. *The Guardian* (2018)
9. J. Dastin, Amazon scraps secret AI recruiting tool that showed bias against women. *Reuters*. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G. Accessed 5 May 2022
10. K. Hao, AI is sending people to jail—and getting it wrong. MIT Technol. Rev. https://www.technologyreview.com/2019/01/21/137783/algorithms-criminal-justice-ai/. Accessed 5 May 2022
11. S. Hare, Facial recognition is now rampant. The implications for our freedom are chilling. *The Guardian* (2019)
12. W. Wang, K. Siau, Ethical and moral issues with AI, in *Twenty-fourth Americas Conference on Information Systems, New Orleans* (2018)
13. Belmont, *The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research*. Department of Health, Education, and Welfare, National Commission for the . . . (1978)
14. R.O. Mason, Four ethical issues of the information age. MIS Q., 5–12 (1986).
15. J. Bentham, *The Collected Works of Jeremy Bentham: An Introduction to the Principles of Morals and Legislation* (Clarendon Press, Oxford, 1996)
16. H. Wallach, Big data, machine learning, and the social sciences: Fairness, accountability, and transparency, in *NIPS Workshop on Fairness, Accountability, and Transparency in Machine Learning* (2014)
17. R. Hursthouse, G. Pettigrove, Virtue ethics, in *The Stanford Encyclopedia of Philosophy*, ed. by E. N. Zalta, (Metaphysics Research Lab, Stanford University, Stanford, CA, 2018)

18. W. Sinnott-Armstrong, Consequentialism, in *The Stanford Encyclopedia of Philosophy*, ed. by E. N. Zalta, (Metaphysics Research Lab, Stanford University, Stanford, CA, 2021)
19. L. Alexander, M. Moore, Deontological ethics, in *The Stanford Encyclopedia of Philosophy*, ed. by E. N. Zalta, (Metaphysics Research Lab, Stanford University, Stanford, CA, 2021)
20. C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy* (Crown/Archetype, New York, 2016)
21. C. El Morr, H. Ali-Hassan, *Analytics in Healthcare: A Practical Introduction* (Springer, Cham, 2019)
22. O.A. Paiva, L.M. Prevedello, The potential impact of artificial intelligence in radiology. Radiologia Brasileira **50**(5), V–VI (2017). https://doi.org/10.1590/0100-3984.2017.50.5e1

Index