

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

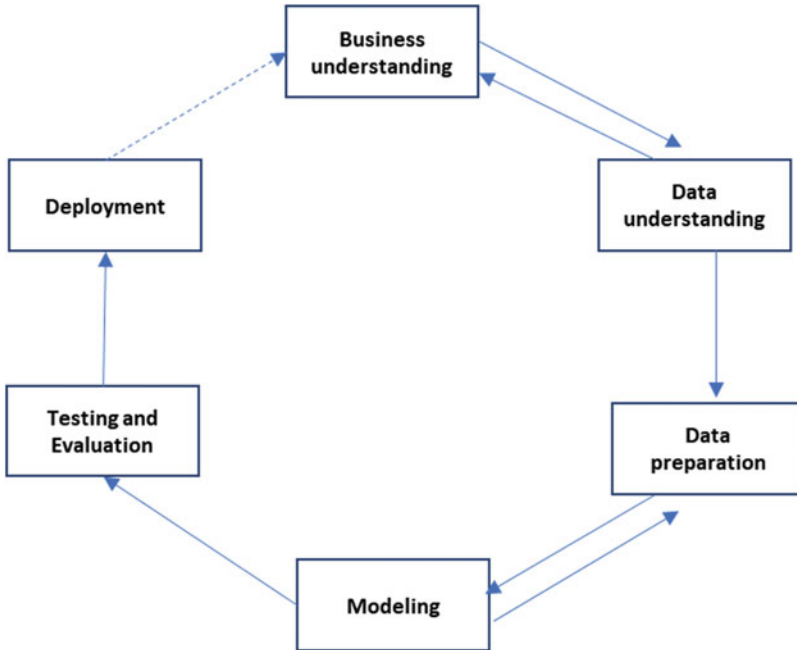


Fig. 3.1 CRISP-DM data mining life cycle (adapted from [3])

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]

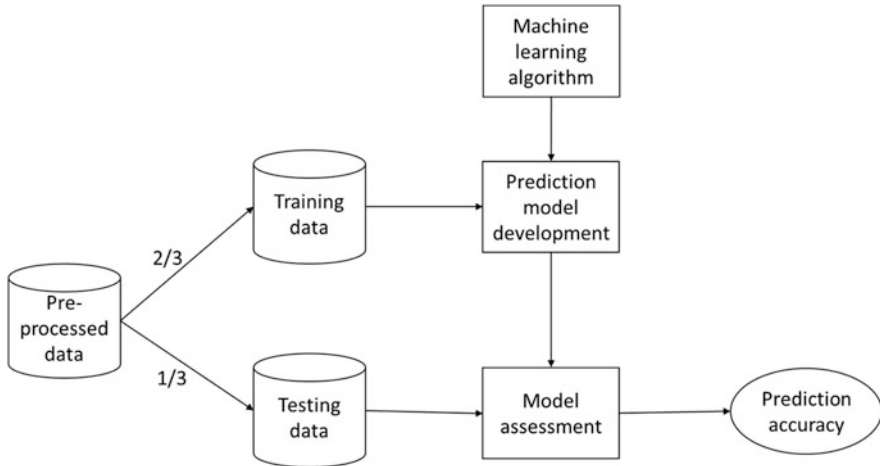


Fig. 3.3 Building a prediction model (adapted from [9])

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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Artificial Neural Networks Genetic Algorithms Hidden Markov Model Hierarchical Clustering K-Means Clustering Self-Organizing Map
		Dimensionality Reduction	<ul style="list-style-type: none"> 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:

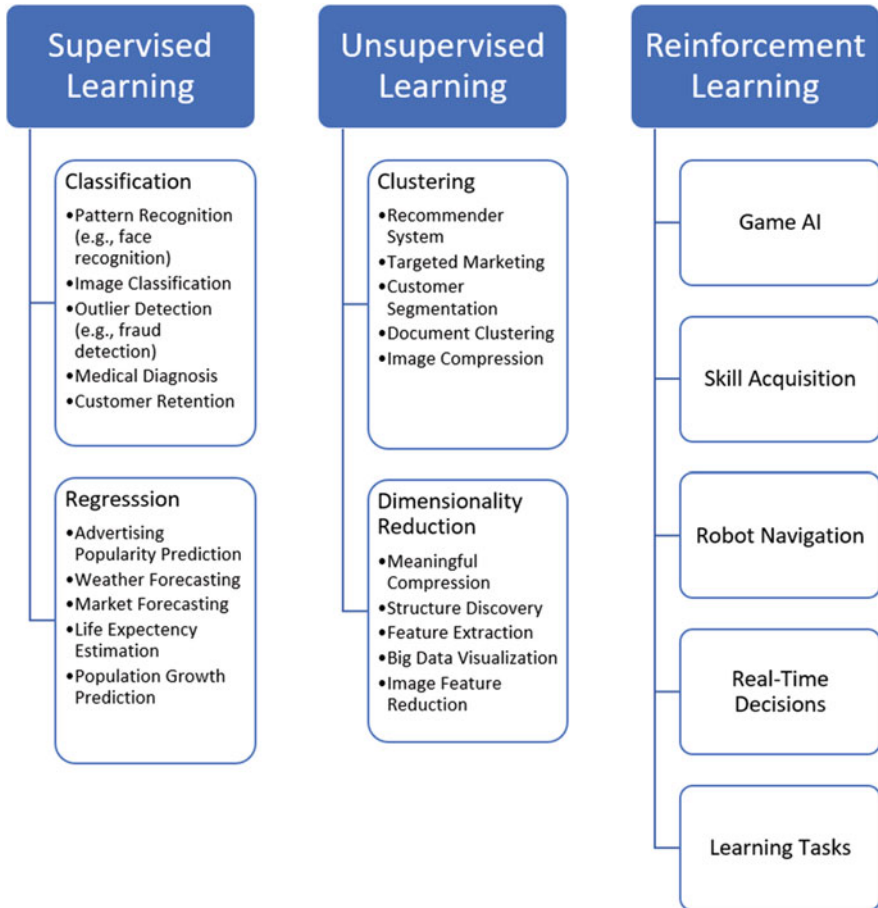


Fig. 3.5 Machine learning applications from a variety of fields (adapted from [13, 14])

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 = $a \times (\text{age}) + b \times (\text{gender}) + c \times (\text{Carlson comorbidity score}) + d$.

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, $\text{cost} = 0.5 \times (\text{age}) + 3 \times (\text{gender}) + 0.2 \times (\text{Carlson comorbidity score}) + 4$ [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 $0.5 \times 50 + 3 \times 1 + 0.2 \times 6 + 4 = 25 + 3 + 3 + 4 = 35$; 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 $35 \times \$2000 = \$75,000$ [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].

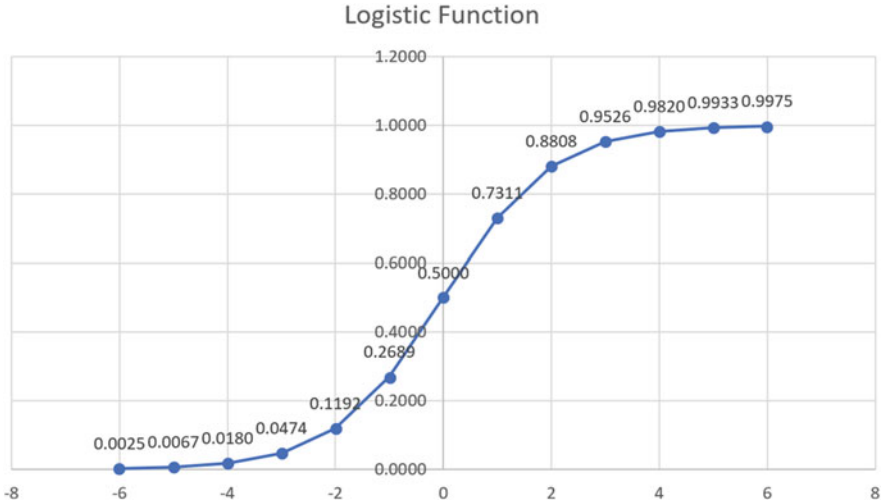


Fig. 3.6 Logistic regression function for data varying between -6 and $+6$

3.4.2 Decision Trees

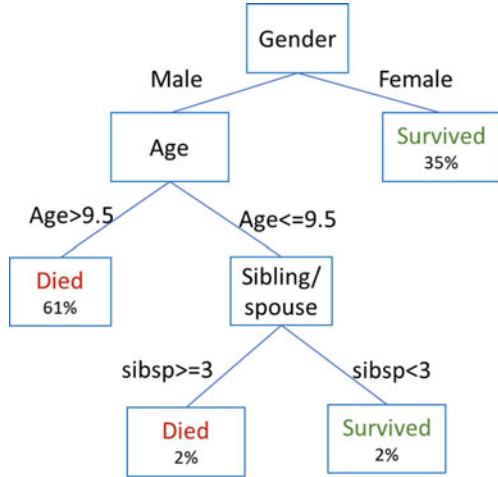
Decision trees have made it possible to find features and patterns in large databases that can be used for discrimination and predictive modeling. In addition to their intuitive interpretation, these characteristics have led decision trees to be widely used for both exploratory data analysis and predictive 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* ($w_0 \dots w_n$), and data *input* ($x_0 \dots x_n$) to generate an *output* $y = f\left(\sum_{i=1}^n (w_i \times x_i)\right) = f(w_0 \times x_0 + w_1 \times x_1 + w_1 \times x_1 \dots)$. The input vector (x_0, x_1, x_2, \dots) 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]

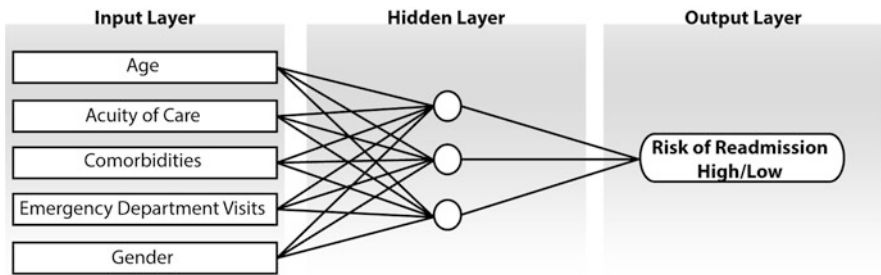
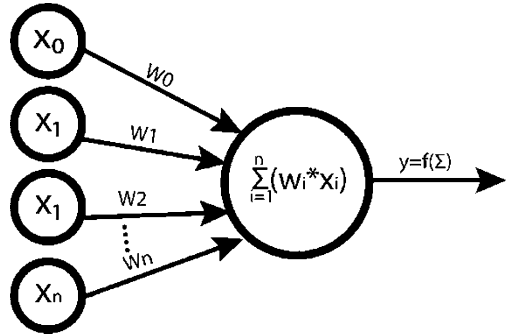


Fig. 3.10 An artificial neural network with an input layer with five nodes each for one variable of the input data, one hidden layer, and one node in the output layer representing two classes: High Risk of Hospital Readmission and Low Risk of Hospital Readmission (adapted from [24])

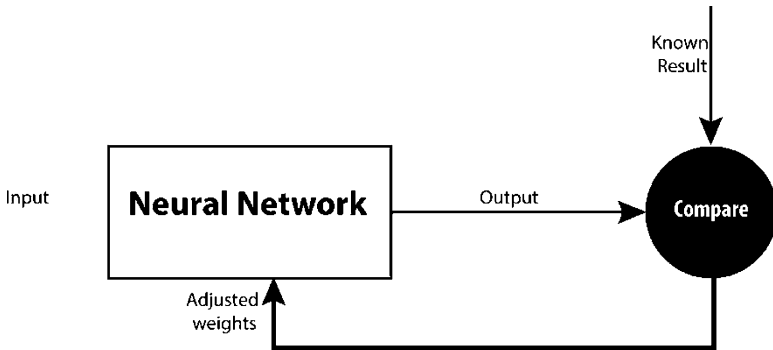


Fig. 3.11 Overall functioning of artificial neural network supervised learning [7]

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, \dots, x_m can be expressed as a feature vector X :

$$X = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{pmatrix}$$

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

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

Mathematically, the multiplication of two vectors W and X is written as WX and is equal to $w_0 \times x_0 + w_1 \times x_1 + \dots + w_m \times x_m$, and the perceptron output can be written as a mathematical equation $y = f(w_0 \times x_0 + w_1 \times x_1 + \dots + w_m \times x_m)$, 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, \dots, 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 : $y_j(t) = f(w_0(t) \times x_{j,0} + w_1(t) \times x_{j,1} + \dots + w_m(t) \times x_{j,m})$. When the process starts, the weights can be initialized to zero.

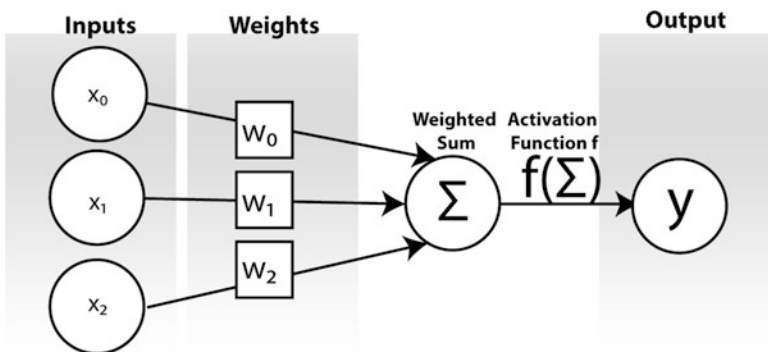


Fig. 3.12 An example of a perceptron [7]

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 ($x_0, x_1, x_2, \dots, x_n$) 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 $P(x_i)$: $P(x|C) = \prod_{i=1}^n P(x_i|C)$.

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: $P(C_j|x)$ $j=1$ to M , where M is the number of classes. Each $P(C_j|x)$ is computed as $P(C_j|x) = P(C) \times \prod_{i=1}^n P(x_i|C_j)$ (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."

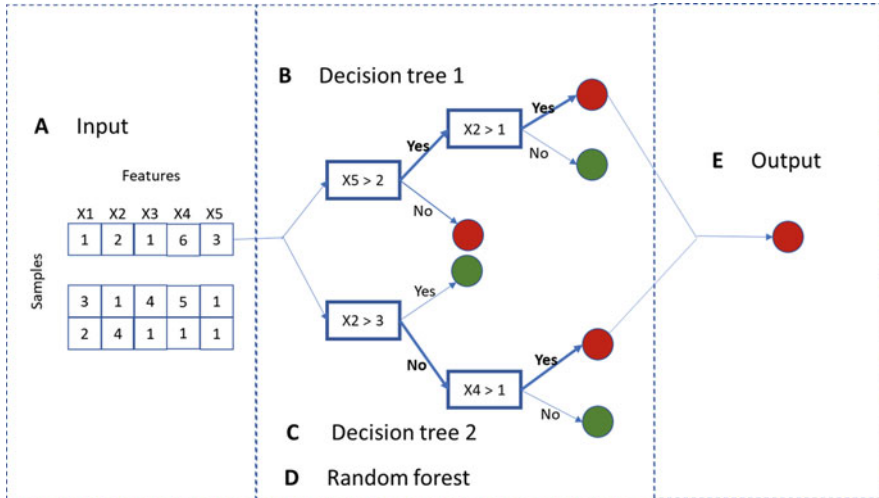


Fig. 3.13 Example of a random forest (adapted from [27])

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: $\{h(x, k), k = 1, \dots\}$ 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 ($x_1, x_2, x_3, x_4,$ and x_5) 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 converge [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

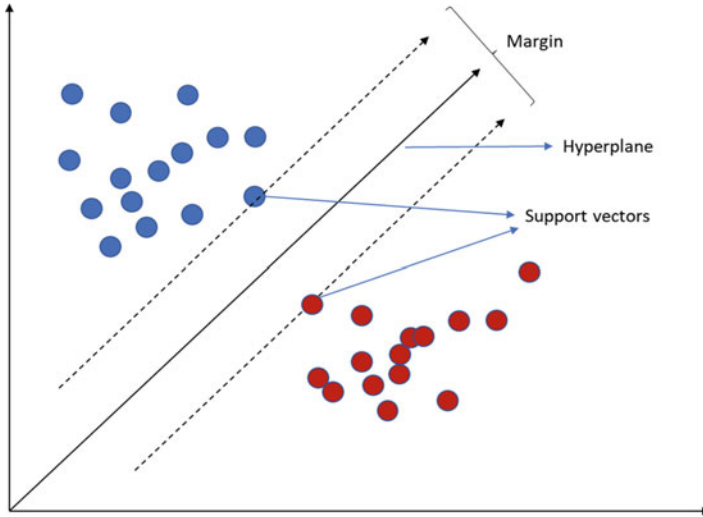


Fig. 3.14 Representation of a hyperplane in two dimensions (adapted from [32])

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: $\beta_0 + \beta_1X_1 + \beta_2X_2 = 0$.

Equation of a hyperplane in p dimensions: $\beta_0 + \beta_1X_1 + \dots + \beta_pX_p = 0$

With X as a feature vector defined by $X=[X_1, \dots, X_p]^T$, we can make a classification by considering one class defined by $\beta_0 + \beta_1X_1 + \dots + \beta_pX_p > 0$ with the other defined by $\beta_0 + \beta_1X_1 + \dots + \beta_pX_p < 0$.

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]

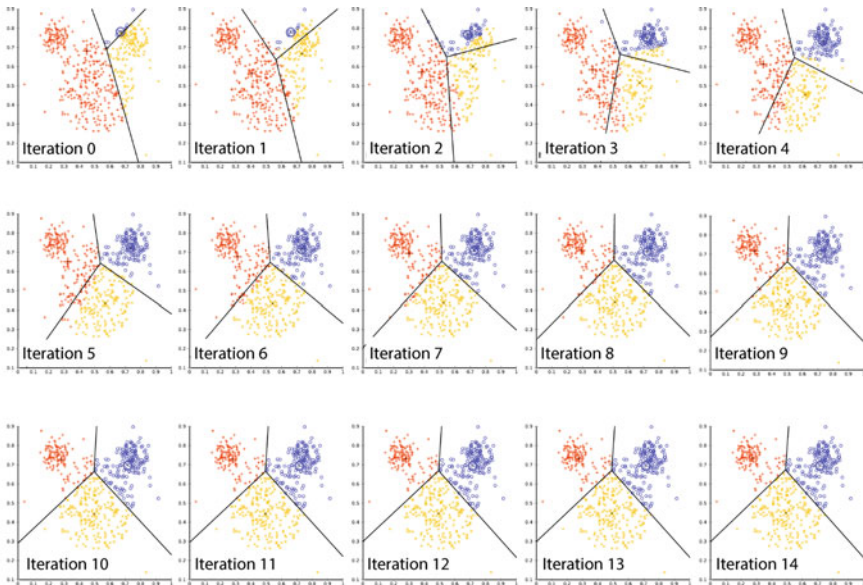
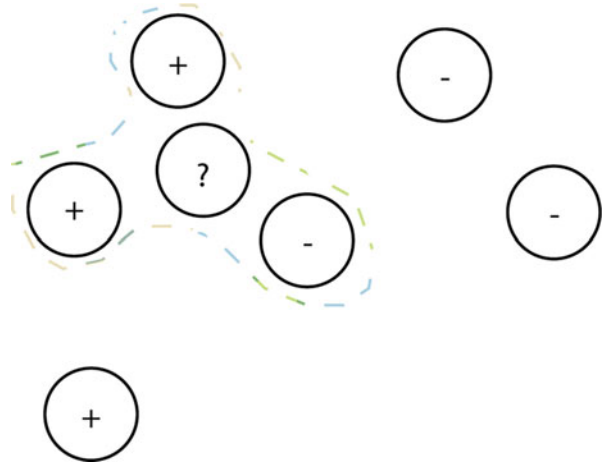


Fig. 3.16 Example of *k*-means convergence (by Chire CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0>), from Wikimedia Commons)

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 converges 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, cyclicity, 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,]
```

```
Console Terminal Jobs x
R 4.1.0 - ~/R
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
version 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

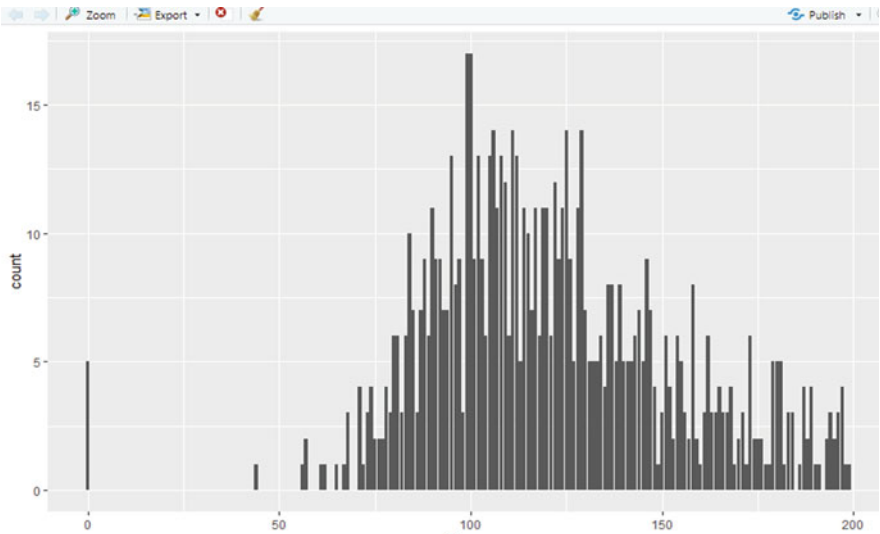


Fig. 3.19 Visualizing glucose values in a bar chart

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.

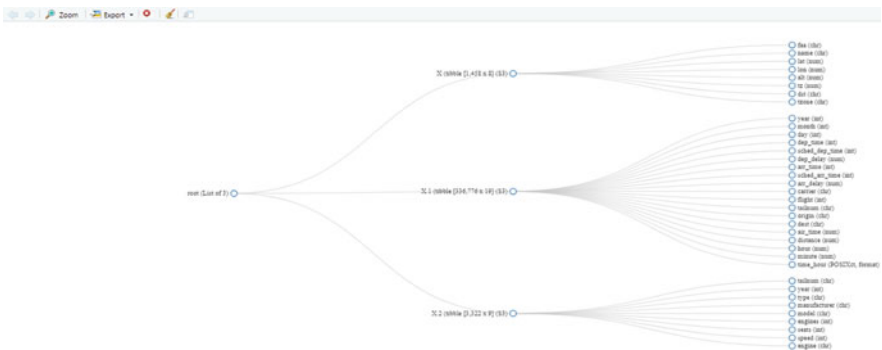


Fig. 3.20 Using DataExplorer package to visualize and analyze flights data

3.11.1.8 Mr3 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

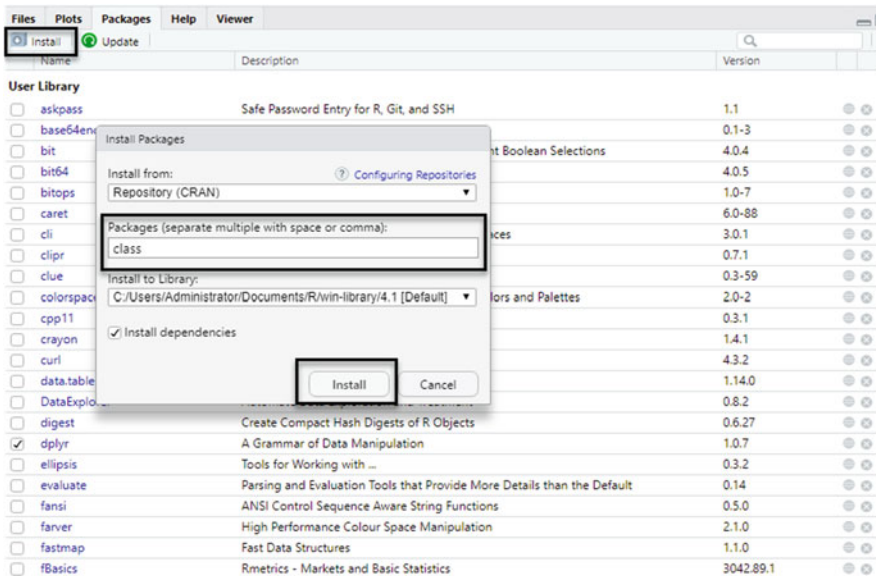


Fig. 3.21 Loading class package in RStudio program

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 “`mapvalues`” 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 ...
 $ 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: $k = \sqrt{\text{size of sample observations}}$. 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

```

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

```

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

```

```

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

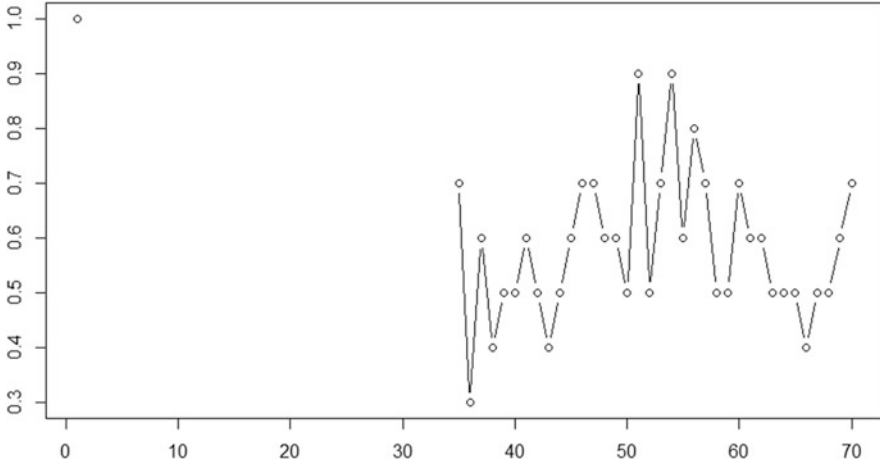


Fig. 3.24 Diamond price accuracy vs. k-value of KNN algorithm

```
> 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, ...
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.

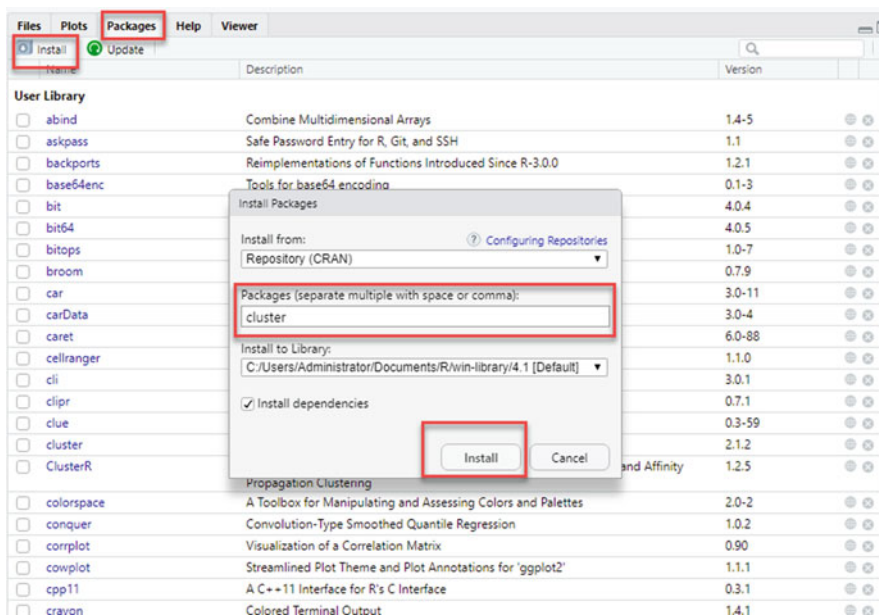


Fig. 3.26 Installing Cluster package in RStudio

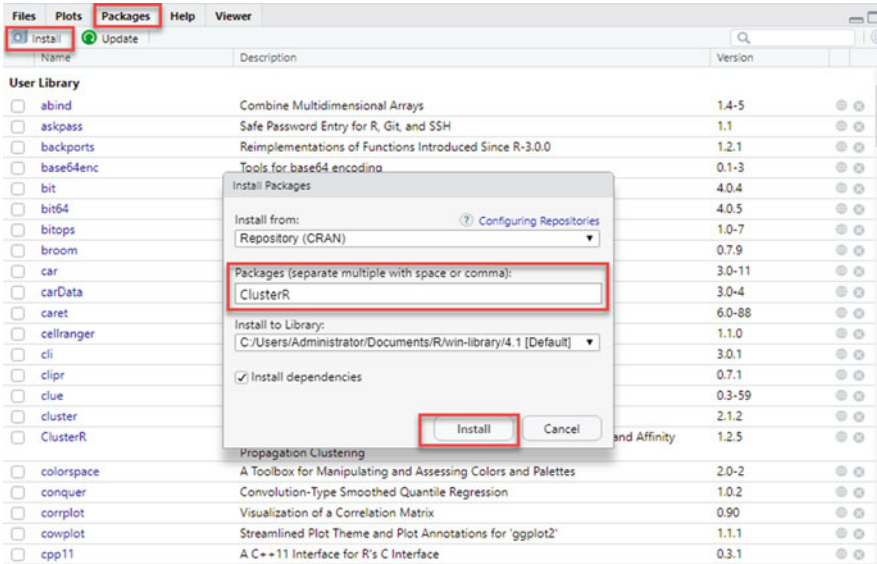


Fig. 3.27 Installing ClusterR package in RStudio

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,

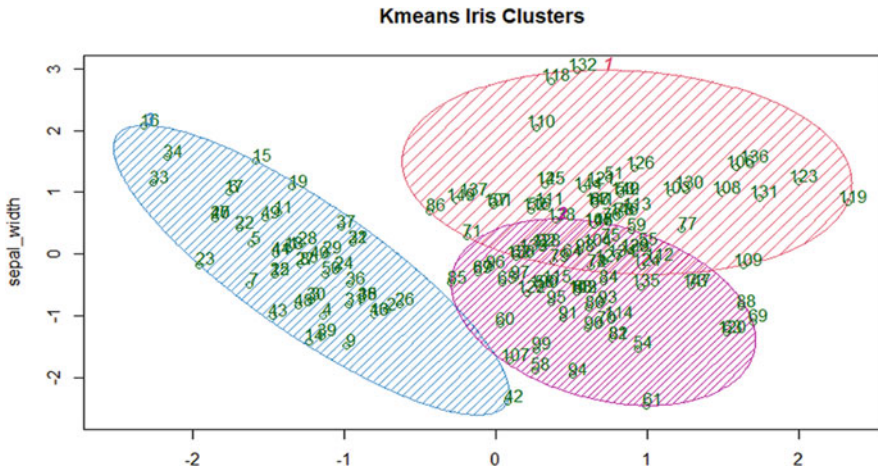


Fig. 3.28 Iris plot for k -means algorithm

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.

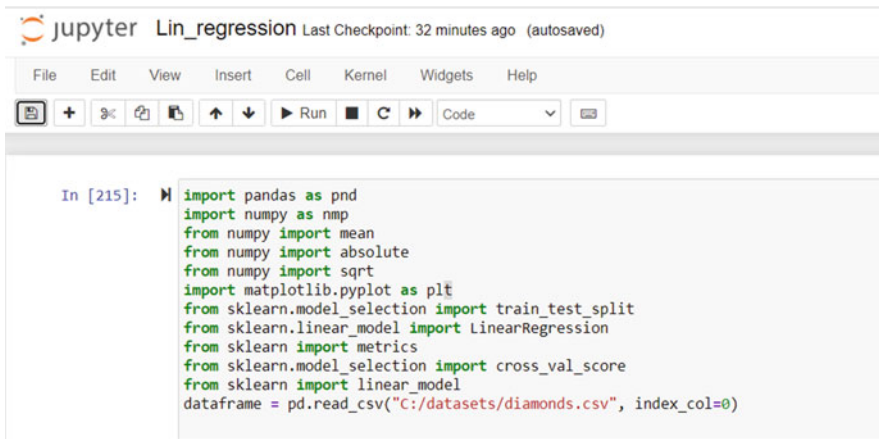
3.11.5.1 Using Scikit-Learn Package

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



The screenshot shows a Jupyter Notebook window titled 'Lin_regression' with a 'Last Checkpoint: 32 minutes ago (autosaved)' status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, navigation, and execution. A code cell is active, showing the following Python code:

```
In [215]: import pandas as pd
import numpy as np
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	1.118600	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]: M import sklearn
          from sklearn.linear_model import SGDRegressor

          dataframe = sklearn.util.shuffle(dataframe)

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

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

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

          y_pred = regressor.predict(X_test)

In [245]: M 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

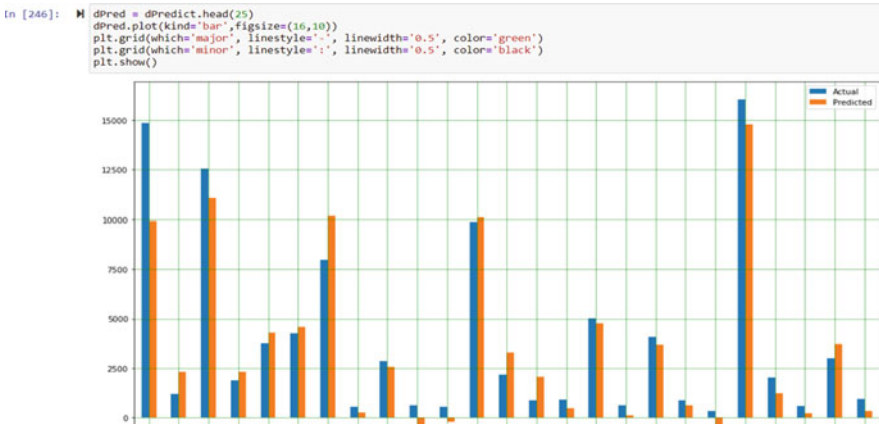


Fig. 3.34 Bar diagram for predicted vs. actual diamond prices

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.

Fig. 3.35 Calculating cross-validation score

```
In [258]: > c1 = linear_model.LinearRegression()
           > scores = cross_val_score(c1, X, y, cv=10)
```

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

plt.show()
```

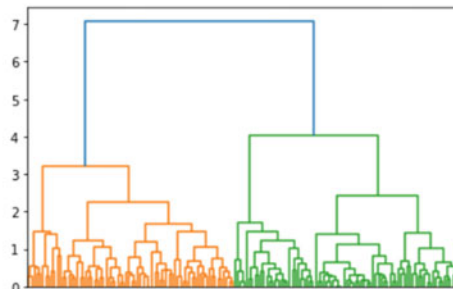


Fig. 3.38 Hierarchical clustering algorithm plot using IRIS dataset

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. Dhani, 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-c98cdbcbaeb3f>

34. R. Sharda, D. Delen, E. Turban, J. Aronson, T.P. Liang, *Business 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, February 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)