Chapter 13
Support Vector Machine

13.1 The Problem

The more the dimensions of a feature space, the more is the computing power needed to classify. Support vector machines (SVMs) main advantages are (1) their effectiveness in a high-dimensional space and in cases where the number of dimensions is higher than the number of instances in the dataset, and (2) their low use of memory and hence their memory efficiency.

The aim of the SVM algorithm is to find the best hyperplane (a line in a two-dimensions space, a plane in a three-dimension space) that divides a dataset into two (or more) classes.

To understand how SVM works, we will take a binary classification problem (Fig. 13.1). Since there could be many lines that can separate the two classes (Fig. 13.1 left), SVM looks for the instances in the datasets (points on the graph) that are closest to the dividing line. The lines passing by these points are called the support vectors. The chosen optimal classification line is the one that maximized the distance between the two support vectors. This is called a maximal margin classification.

Of course, the instances may not that perfectly separable by a line, we need to find a way to classify determine how much should we relax the constraint related to maximizing the margin. This is called a soft margin classification.

The other problem to solve is when the classes are not linearly separable, in this case we need a non-straight line to separate the instances. In SVM, this is done using a kernel. A linear kernel allows to separate linearly separable classes, a polynomial kernel allows the use of a curved line to separate the classes and a radial kernel uses a radial-based function (RBF) to solve complex separations, for example, using a polygon in a two-dimension space).

C. El Morr et al., *Machine Learning for Practical Decision Making*, International Series in Operations Research & Management Science 334,
https://doi.org/10.1007/978-3-031-16990-8_13

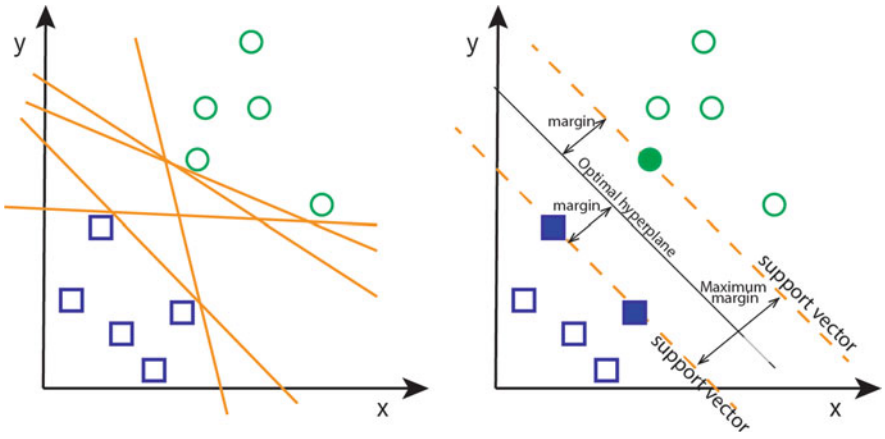


Fig. 13.1 Two datasets green (circle) and blue (square) to be classified. Each point has two features *x* and *y*

13.2 The Algorithm

SVMs are a collection of similar supervised learning algorithms that are used for classification and regression [1–4]. The most effective way to grasp the fundamentals of support vector machines and how they function is to use a simple example (Fig. 13.1). Consider the following scenario: we have two tags, one each of green (circle shape) and blue (square shape), and our data contains two characteristics, *x,* and *y*. We are looking for a classifier that, when given a pair of (x, y) coordinates, outputs whether the pair is red or blue in color. On a plane, we plot the training data that has previously been labeled:

When given these data points, a support vector machine will produce the hyperplane (in two dimensions, a hyperplane is simply a line) that will optimally divide the tags. The hyperplane is the decision border. In 2D, each side of the line will be considered a class (i.e., blue class and green class).

But, more specifically, what is the finest hyperplane? It is the one that optimizes the margins from both tags in the case of SVM. The hyperplane (remember, it is a line in this case) with the greatest distance to the nearest element of each tag is known as the maximum distance hyperplane.

The SVM's goal is to find the best hyperplane (or decision boundary) [5] that divides two different classes while also maximizing the distance between data points from both classes. There could be several hyperplanes to divide the two classes; our aim is to find the hyperplane that is at the greatest distance between data points from both classes (i.e., the greatest margin) [6]. Maximizing the margin distance allows subsequent data points to be categorized with more certainty.

Obviously, the number of features dictates the hyperplane's dimension; in Fig. 13.2, we have two features *x* and *y*, the hyperplane was a straight line [5]. If

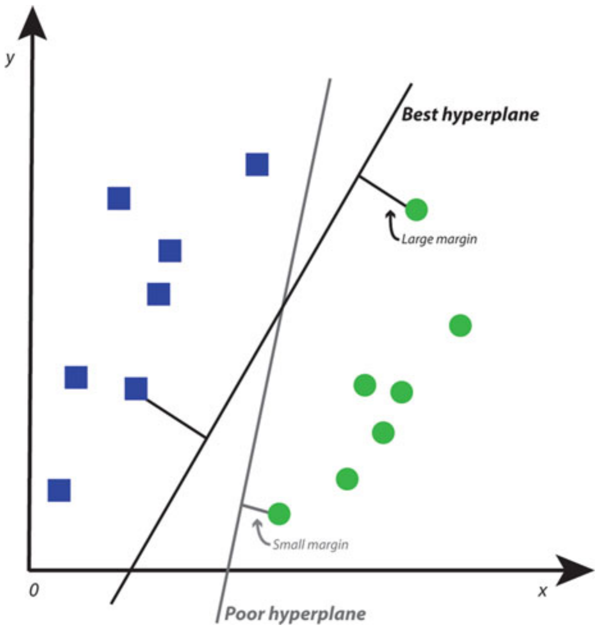


Fig. 13.2 The hyperplane (remember, it is a line in this case) with the greatest distance to the nearest element of each tag

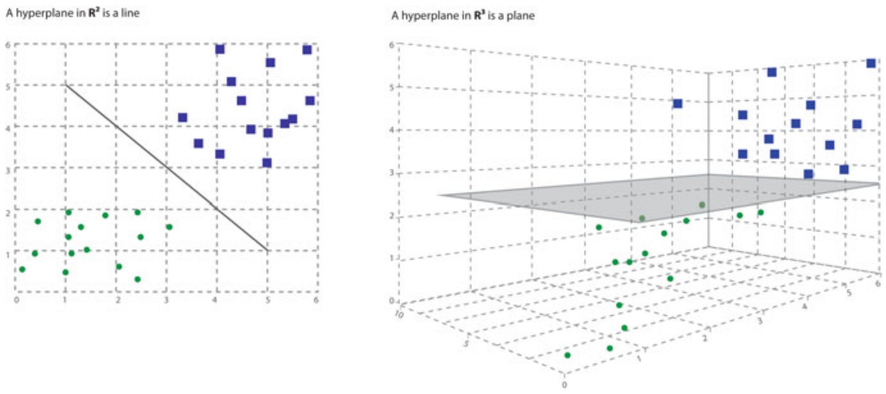


Fig. 13.3 A line hyperplane in a 2D space (left) vs. a two-dimensional hyperplane in a 3D space (right)

the number of features is three, then the hyperplane becomes a 2D plane. Beyond three features, we cannot visualize the hyperplane (Fig. 13.3).

Support vector machines are widely used in machine learning research all around the world, particularly in the United States. When SVMs were used in a handwriting recognition test, they gained popularity since they achieved performance equivalent to that of complex neural networks with elaborated features when employing pixel maps as input [2, 7].

13.2.1 Important Concepts

Support Vectors Support vectors are the data points that are closest to the hyperplane and are used to calculate the distance between them. With the aid of these data points, a dividing line will be drawn between them. It is possible to demonstrate that the optimal hyperplane is derived from the function class with the lowest "capacity" = number of independent features/parameters that can be twiddled [8]. In other words, they are the data points that are closest to a decision surface (or hyperplane). They are also the data points that are most difficult to classify. They have a direct bearing on the optimal location of the decision surface.

Hyperplane As we can see in the diagrams above, a hyperplane is a decision plane or space that is partitioned between a collection of objects belonging to distinct classes. In two dimensions, the hyperplane can be represented by the following equation. This is identical to the equation of affine combination; however, the bias b has been included in this case [9].

$$\beta_1 x_1 + \beta_2 x_2 + b$$

For d-dimensional space, we may generalize this and express it in vectorized form.

$$\begin{aligned} h(x) &= \beta_1 x_1 + \cdots + \beta_d x_d + b \\ &= \left(\sum_{i=1}^{d} \beta_i x_i \right) + b \\ &= \beta^T x + b \end{aligned}$$

For any point $X = (x_1, \ldots, x_d)$, if $h(X) = 0$, then X lies on the hyperplane; otherwise $h(X) < 0$ or $h(X) > 0$, which implies that X falls to one side of the hyperplane. If we now make a very significant assumption about the coefficient weight vector β and assume that x_1 and x_2 are two random locations that lie on the hyperplane, we may write:

$$h(x_1) = \beta^T x_1 + b = 0$$

$$h(x_2) = \beta^T x_2 + b = 0$$

Hence,

$$\beta^T x_1 + b = \beta^T x_2 + b$$

and

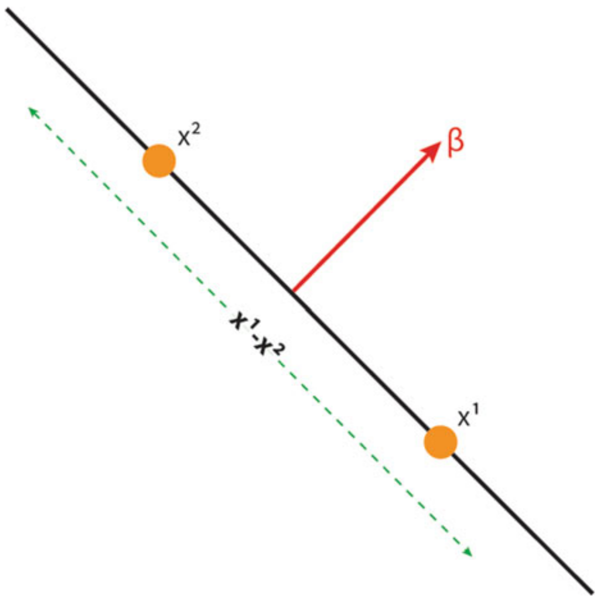


Fig. 13.4 The weight vector beta points in the direction that is normal to the hyperplane of the weight vector

$$\beta^T(x_1 - x_2) = 0$$

If the dot product of two vectors is 0, we know that the vectors are orthogonal to one another, and vice versa. The weight vector β in this case is orthogonal to $(x_1 - x_2)$. Being that $(x_1 - x_2)$ is located on the hyperplane, it follows that the weight vector β is also orthogonal to the hyperplane. That is to say, the weight vector beta points in the direction that is normal to the hyperplane of the weight vector. When the hyperplane is shifted in d-dimensional space, this is expressed as a bias ($\boldsymbol{b}$) [9] (Fig. 13.4).

13.2.2 Margin

The distance between two lines drawn through the closest data points of distinct classifications can be described as a margin. The minimal distance (normal distance) between each observation and a specific separating hyperplane can be used to establish the margin between two observations. See how we may utilize the margin to determine the best hyperplane for our situation. It may be computed by taking the perpendicular distance between the line and the support vectors and dividing it by two.

A large margin is seen as a good margin, while a small margin is regarded as a bad margin in business. The size of the margin determines the confidence level of the classifier; as a result, the largest possible margin should be used. Let us choose two

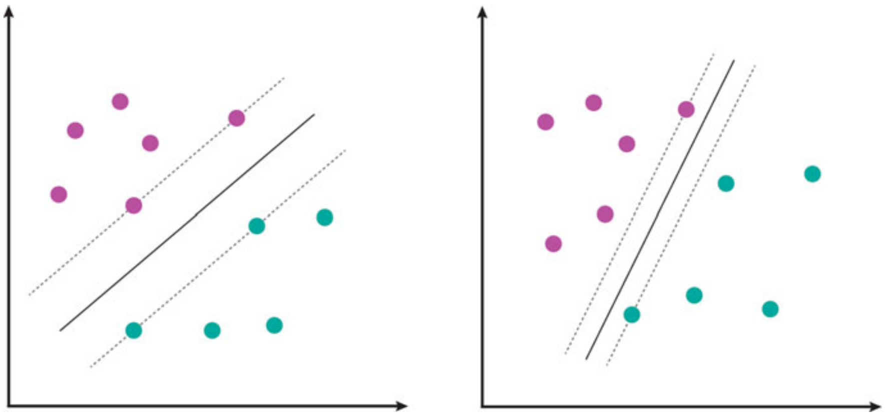


Fig. 13.5 Large (left) vs. small (right) margin

hyperplanes based on their distance from the center (Fig. 13.5). The one to the left has a significantly larger margin than the one to the right, and as a result, the first hyperplane is more optimal than the second one.

We may conclude that in the maximal margin classifier, in order to categorize the data, we will utilize a separation hyperplane that is the greatest (maximum) and smallest (minimum) distance away from the observations in order to classify the data. Let us keep in mind that the margin will still be used to pick the ideal separating hyperplane. Furthermore, Jana margins are divided into two categories: functional margin and geographic margin [9]. They are both summed up below.

13.2.2.1 Functional Margin

To define the theoretical side of the margin, the term "functional margin" is employed. In the presence of a training example (x_i, y_i), the functional margin of (β, b) with regard to the training example will be as follows:

$$y_i\left(\beta^T X_i + b\right) = \widehat{\gamma}_i$$

As opposed to just specifying that the number is larger than 0, we have established a value for the margin by using γ. Thus, the below requirements may be established:

$$\text{if } y_i = 1, \text{then } \widehat{\gamma}_i > 0$$
$$\text{if } y_i = 0, \text{then } \widehat{\gamma}_i = 0$$

But there is a problem with the functional margin, which is that its value is reliant on the values of β and b. The equation of the hyperplane remains the same when β

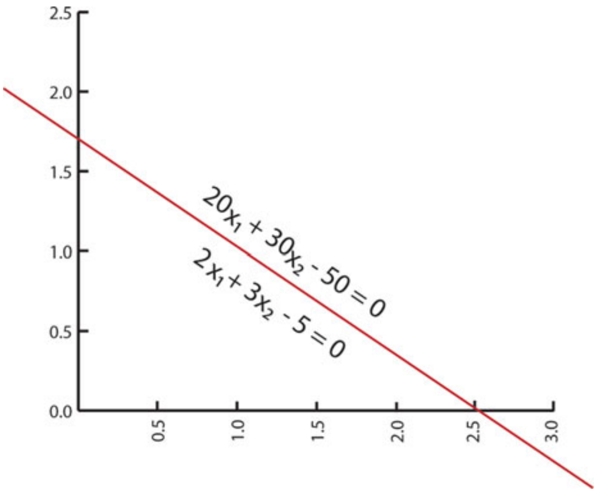


Fig. 13.6 The same hyperplane representing two equations

and b are scaled (multiplied by some scaler s), but the margin increases. If you plot the following two equations, they will both represent the same hyperplane, but in this case, the width of the margins will change between the two equations (Fig. 13.6).

$$2x_1 + 3x_2 – 5 = 0$$

$$20x_1 + 30x_2 – 50 = 0$$

13.2.2.2 Geometric Margin

Let us make considerations regarding the visuals below (Fig. 13.7):

Along with the vector w, the decision boundary corresponding to (w, b) is depicted in Fig. 13.7. It should be noted that w is orthogonal (i.e., at 90°) to the separation hyperplane.

You must convince yourself that this is a fact. Consider the point A, which represents the input $x^{(i)}$ of a training example with the label $y^{(i)} = 1$, as represented by the point B. The line segment AB determines the distance between it and the decision border, denoted by $\gamma^{(i)}$.

The value of $\gamma^{(i)}$ can be determined in several ways. To explain it more clearly, the unit-length vector $w/\|w\|$ indicates that w is moving in the same direction. Since A represents $x^{(i)}$, we may conclude that the point B is given by $x^{(i)} - \gamma^{(i)} \times w/\|w\|$. The problem is that this point is located on the decision boundary, and all points x on the decision boundary are satisfied by the equation $w^T x + b = 0$; therefore:

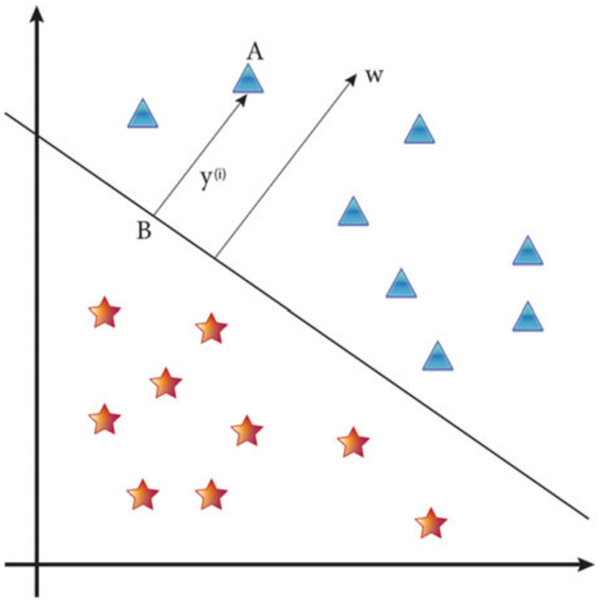


Fig. 13.7 Two datasets separated by a hyperplane with weigh vector w and a decision boundary (w, b)

$$w^T\left(x^{(i)} - \gamma^{(i)}\frac{w}{\|w\|}\right) + b = 0$$

And solving for $\gamma^{(i)}$ yields:

$$\gamma^{(i)} = \frac{w^T x^{(i)} + b}{\|w\|} = \left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}$$

Specifically, this was calculated for the situation of a positive training example at A in Fig. 13.7, in which being on the "positive" side of the decision border is advantageous.

Furthermore, we define the geometric margin of (w, b) regarding a training example $(x^{(i)}, y^{(i)})$ as follows:

$$\gamma^{(i)} = y^{(i)}\left(\left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}\right)$$

It is important to note that if $\|w\| = 1$, then the functional margin equals the geometric margin—this provides a means of connecting these two disparate ideas of margin together. As a result of this property, the geometric margin is invariant to rescaling of the parameters; that is, if we substitute two values for w and two values for b, the geometric margin remains unchanged. Furthermore, because of this invariance to scaling of the parameters, we can apply any arbitrary scaling constraint to w without producing important changes [6]; for example, we can demand that $\|w\| = 1$, or that $|w_1 + b| + |w_2| = 2$, and any other constraint can be satisfied by just rescaling w and the parameters; however, this is not recommended.

13.2.3 Types of Support Vector Machines

Support vector machines are generally classified into only two types. They are both detailed below:

13.2.3.1 Linear Support Vector Machine

This type only works with data that can be divided into two categories by a single perfect line, in which case the dataset is considered linearly separable, and the linear SVM classifier is used. This is further divided into two types and is visually displayed below.

13.2.3.2 Soft Margin Classifier

A soft margin classifier is an SVM that where the threshold is allowed to make an tolerable number of misclassifications, while allowing new data instances to be classified correctly [10]. The famous cross-validation technique can be used to determine the best classification (Fig. 13.8).

In a real-world scenario, it is unlikely that a perfectly distinct line would be drawn between the data points included inside the space [11]. Furthermore, we might have a curved decision boundary. It is possible to have a hyperplane that precisely separates the data; however, this may not be desired if the data contains noise. Jakkula agrees it is preferable for the smooth border to disregard a small number of data points rather than being curved or going in loops around outliers [2].

The assumption that the dataset is perfectly linearly separable has been made up to this point. This assumption does not hold up to scrutiny when dealing with a real-world dataset. As a result, let us look at a slightly more challenging scenario. The linear SVM is still in the works; however, this time, some of the classes overlap in such a way that a perfect separation is unattainable, yet the data is still linearly separable [9]. Consider a dataset with two dimensions shown in Fig. 13.9. There are two primary options available:

- When a single outlier occurs, the decision boundary might be pushed significantly, resulting in an extremely tight margin of safety.
- The data may not be separable using a straight line, even when a linear decision boundary can correctly categorize the target classes (no clear boundary).

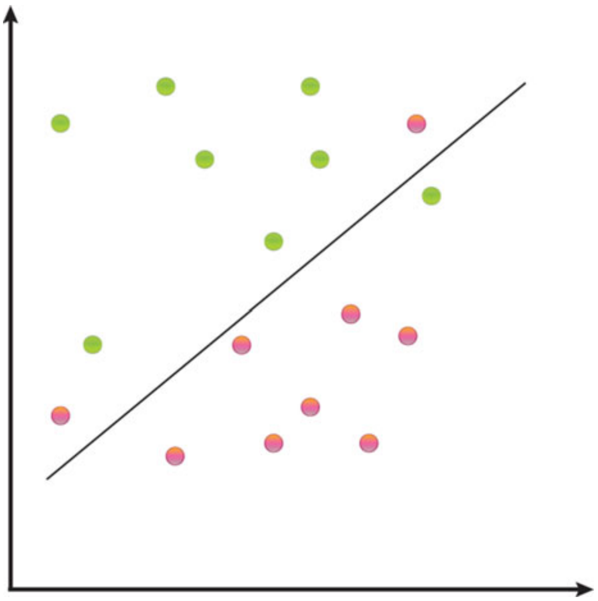


Fig. 13.8 Linear SVM—soft margin classifier

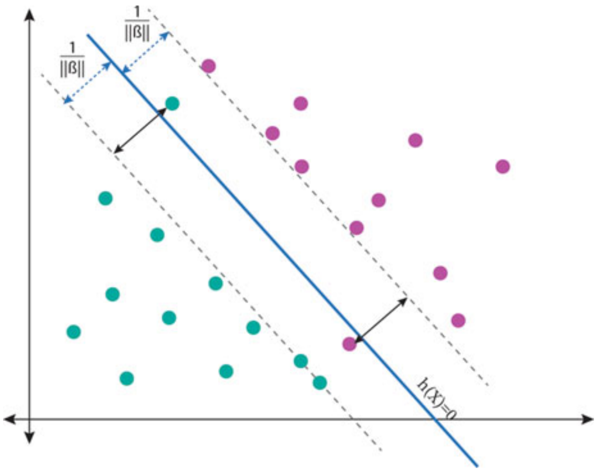


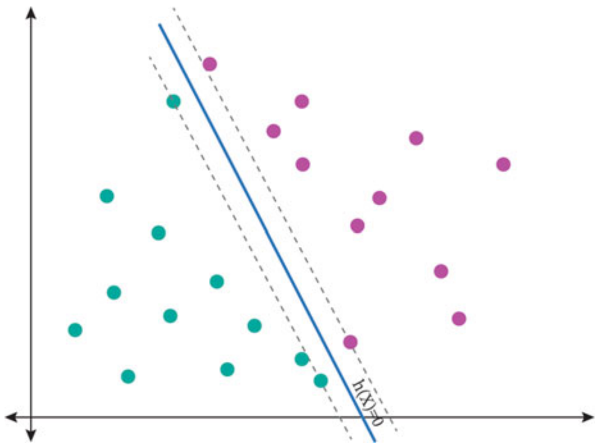
Fig. 13.9 A dataset with two dimensions

In other words, the hard margin classifier that is visualized in Fig. 13.10 would not operate owing to the inequality restriction $y_i(\beta^T x_i + 1) \geq 1$.

13.2.3.2.1 Hard Margin Classifier

As previously stated, the idea of SVM is to execute an affine discrimination of observations with the greatest amount of margin possible, that is, to identify an

Fig. 13.10 Linear SVM—hard margin classifier



element ($w \in X$) with the lowest norm and the greatest possible real value b, such that the value of $y_i((w_i, x_i) + b) \geq 1$ is the same for all i. But to do so, we must first solve the quadratic programming issue described below:

$$\min_{w,b} < w, \quad w >$$
$$\text{subject to} \quad y_{i(<w_i,\ x_i>+b)} \geq 1, \qquad 1 \leq i \leq N$$

The classification rule that relates to (w, b) is simply referred to as $f(x) = \sin((w, x) + b)$. In this circumstance (which is referred to as the hard margin SVM), we require that the rule have zero error on the learning set (Fig. 13.10).

13.2.3.3 Nonlinear Support Vector Machine

This classifier is used for nonlinearly separated data, which means that if a dataset cannot be classified using a straight line, it is considered nonlinear data, and the classifier used is the nonlinear SVM classifier. A graphical representation is shown below (Fig. 13.11) [11].

A mathematical example of nonlinear support vector machines is described below:

$$K(x, y) = (x.y + 1)^p$$
$$\left\{ -\|x-y\|^2/2\sigma^2 \right\}$$
$$K(x, y) = \tan h(kx.y–\delta)$$

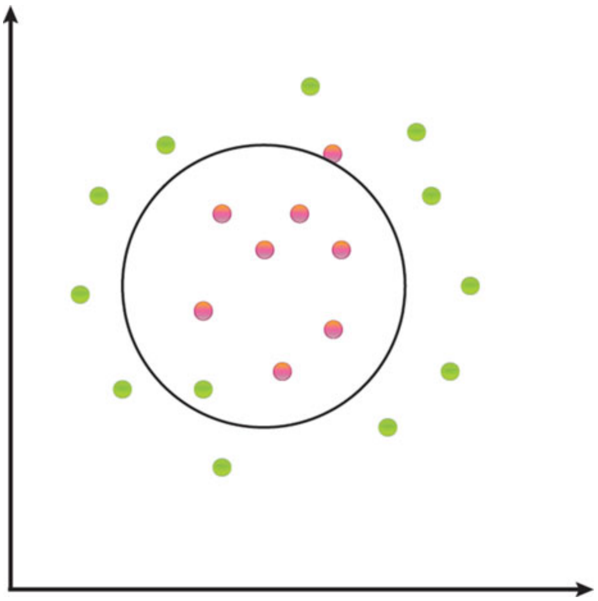


Fig. 13.11 Representation of a nonlinear Support Vector Machine

The first equation is a polynomial, while the second equation is a radial basis function (Gaussians), and the third is a sigmoid (neural net activation function) [8]. Some of these are visualized just below.

13.2.4 Classification

SVM is a data classification approach that is beneficial. The employment of neural networks, despite the fact that they are regarded as more user-friendly than SVM, might result in disappointing outcomes at times [12]. Training and testing data for classification tasks typically comprise a small number of data examples [2]. A target value and a number of characteristics are contained inside each instance of the training set. The SVM model allows us to predicts target values of the instances in the testing dataset [13].

Supervised learning may be seen in the classification process of SVM. Known labels assist in determining whether or not the system is operating in the proper manner. This information either points to a desired reaction, thus verifying the correctness of the system, or it may be utilized to assist the system in learning to behave in the appropriate manner. In SVM classification [2, 13], one phase is the identification of classes that are tightly related to the classes that are already recognized. This is referred to as feature selection, or feature extraction in technical terms. Even when the prediction of unknown samples is not required, the combination of feature selection with SVM classification might be beneficial. In order to separate the classes, they can be utilized to identify key sets that are engaged in the procedures that distinguish them.

13.2.5 *Regression*

Through the use of an alternate loss function, it is possible to apply SVMs to regression situations [13, 14]. It is necessary to modify the loss function in order to add a distance measure. There are two types of regression: linear and nonlinear. Linear models are composed mostly of the loss functions listed below: e-intensive loss functions, quadratic loss functions, and the Huber loss function.

It is common for nonlinear models to be required for data modeling challenges, much as it is for classification difficulties. A technique similar to the nonlinear SVC approach, nonlinear mapping, may be used to map the data into a high-dimensional feature space, where linear regression can then be done on the information.

When it comes to dealing with the curse of dimensionality [15], the kernel technique is once again used. Considerations based on past knowledge of the problem and the distribution of the noise are taken into account while employing the regression approach. The robust loss function of Huber has been proved to be a good substitute in the absence of such information [13].

13.2.6 *Tuning Parameters*

13.2.6.1 Regularization

For each training dataset, the support vector machine is instructed on the optimal degree of misclassification to avoid by adjusting the regularization parameter, which is also known as the C parameter in Python's sklearn module. When larger numbers are used for the C parameter in a support vector machine, the optimizer will automatically choose a hyperplane margin that is smaller if it is successful in separating and classifying all the training data points during the optimization process. Alternately, when dealing with extremely small values, the algorithm will seek a larger margin for the hyperplane to separate, even if the hyperplane misclassifies some data points.

13.2.6.2 Gamma

An influence on a single training data sample is repeated several times using this tuning parameter. Lower gamma values reflect distance from the hyperplane, whereas higher gamma values show proximity to the hyperplane. Data points with both low and high gamma (far from and near to the hyperplane, respectively) are included in the computation of the separation line.

13.2.6.3 Margins

The margin is the final but not the least important characteristic. It is also a critical parameter for fine-tuning and a vital characteristic of a support vector machine classifier. The margin, as previously established, is the distance between the line and the data points from the classes. When using the support vector approach, it is critical to have a good and appropriate margin. When the difference between the two groups of data is higher than one standard deviation, it is a good margin. A sufficient margin ensures that the individual data points remain inside their respective classes and do not cross over into another class.

13.2.7 Kernel

When using SVM, a kernel turns the input data space into the desired format. SVM employs the kernel trick to turn a low-dimensional input space into a higher-dimensional space. For the uninitiated, this means that kernel adds new dimensions to an issue that would otherwise be impossible to separate.

Generally speaking, it is most useful in nonlinear separation situations. Simply said, the kernel performs a number of incredibly sophisticated data transformations before determining the best method of separating the data depending on the labels or outputs that have been established.

As a result, SVM gains higher scalability, adaptability, and accuracy. Kernels utilized by SVM include those listed below.

13.2.7.1 Linear Kernel

All observations can be combined in this way. Here is the equation for a linear kernel:

$$K(x, x_i) = \text{sum}(x \times x_i)$$

The product between two vectors, x and x_i, may be represented as the total of the products of each pair of input values in the formula above.

13.2.7.2 Polynomial Kernel

Curved or nonlinear input spaces can be distinguished using this generalized linear kernel. A polynomial kernel may be expressed using the following formula:

$$K(x, x_i) = 1 + \mathrm{sum}(x \times x_i)^d$$

Here, d is the degree of the polynomial, which we must manually enter into the learning algorithm.

13.2.7.3 Radial Basis Function (RBF) Kernel

When used in SVM classification, the RBF kernel transforms the input space into an infinitum of three-dimensional spaces. It is widely used in SVM classification tasks. The following formula provides a mathematical explanation:

$$K(x, x_i) = \exp\left(-\mathrm{gamma} \times \mathrm{sum}\left(x - {x_i}^2\right)\right)$$

In this case, gamma is between 0 and 1. We must explicitly define it in the learning algorithm; the default value of gamma is 0.1, which is the industry-accepted default.

13.3 Advantages, Disadvantages, and Best Practices

Nonetheless, the SVM's greatest benefit is the kernel technique, which allows it to classify extremely nonlinear situations by creating complicated boundary shapes, rather than by using simple classification rules [16]. These qualities have enabled the SVM to find widespread use in a variety of disciplines throughout the course of the previous few years. SVM has been utilized for fault diagnostics [17], quality improvement [18], and quality assessment [19].

SVMs have been used in the field of computer vision for a variety of tasks, such as face detection, picture categorization, hand gesture recognition, and background removal. SVMs have been utilized in finance for a variety of purposes, including financial time series forecasting and the prediction of bankruptcy. Aside from hydrology, other uses of SVM include forecasting of solar and wind resources, prediction of atmospheric temperature, bioinformatics, speaker recognition, agricultural forecasting, and electrical design. The quality of the datasets, on the other hand, has an impact on the performance of basic support vector machines (SVMs). Typically, noise may be found in real-world datasets. Noise is defined as anything that obscures the link between the attributes of an instance and the characteristics of its class. The noise might express itself as feature-noise (or feature uncertainty), which has the effect of altering the observed value of the corresponding feature. Certainly, uncertainties may arise as a result of the constraints of observational material, as well as the restricted resources available for data collection, storage, transformation, and analysis.

Overall, the training of SVM is quite simple, which is one of its key advantages. It scales rather well to large amounts of high-dimensional data, and the trade-off between classifier complexity and error may be carefully adjusted. It is necessary to have a good kernel function, which is one of the weaknesses [13, 20]. Overall, it is a good idea to standardize to avoid the optimal hyperplane being influenced by the scale of the features.

13.4 Key Terms

1. Statistical learning theory
2. Hyperplane
3. Structural risk minimization
4. Support vectors
5. Coefficient weight vector
6. Functional margin
7. Geometric margin
8. Soft margin classifier
9. Hard margin classifier
10. Curse of dimensionality
11. Gamma

13.5 Test Your Understanding

1. What are support vectors?
2. How do support vector machines function?
3. When have we achieved a maximum distance hyperplane?
4. What is a hyperplane? Highlight its purpose(s).
5. Explain the structural risk management concept.
6. Describe an optimization theory-based learning algorithm.
7. What is the difference between the functional margin and the geometric margin?
8. List the two types of support vectors.
9. Distinguish between soft and hard margin classifiers.
10. Describe the maximal margin classifier.
11. Why do SVMs use the kernel trick?
12. Highlight the tuning parameters of a support vector machine.

13.6 Read More

1. K. R. Song et al., "Resting-state connectome-based support-vector-machine predictive modeling of internet gaming disorder," (in eng), *Addict Biol,* vol. 26, no. 4, p. e12969, Jul 2021, doi: 10.1111/adb.12969.
2. A. Fleury, N. Noury, and M. Vacher, "Supervised classification of Activities of Daily Living in Health Smart Homes using SVM," (in eng), *Annu Int Conf IEEE Eng Med Biol Soc,* vol. 2009, pp. 6099–102, 2009, doi: 10.1109/iembs.2009.5334931.
3. L. Squarcina et al., "Automatic classification of autism spectrum disorder in children using cortical thickness and support vector machine," (in eng), *Brain Behav,* vol. 11, no. 8, p. e2238, Aug 2021, doi: 10.1002/brb3.2238.
4. P. Unnikrishnan, D. K. Kumar, S. Poosapadi Arjunan, H. Kumar, P. Mitchell, and R. Kawasaki, "Development of Health Parameter Model for Risk Prediction of CVD Using SVM," (in eng), *Comput Math Methods Med,* vol. 2016, p. 3016245, 2016, doi: 10.1155/2016/3016245.
5. A. Fleury, M. Vacher, and N. Noury, "SVM-based multimodal classification of activities of daily living in Health Smart Homes: sensors, algorithms, and first experimental results," (in eng), *IEEE Trans Inf Technol Biomed,* vol. 14, no. 2, pp. 274–83, Mar 2010, doi: 10.1109/titb.2009.2037317.
6. S. Wang et al., "Abnormal regional homogeneity as a potential imaging biomarker for adolescent-onset schizophrenia: A resting-state fMRI study and support vector machine analysis," (in eng), *Schizophr Res,* vol. 192, pp. 179–184, Feb 2018, doi: 10.1016/j.schres.2017.05.038.
7. S. Wang, G. Wang, H. Lv, R. Wu, J. Zhao, and W. Guo, "Abnormal regional homogeneity as potential imaging biomarker for psychosis risk syndrome: a resting-state fMRI study and support vector machine analysis," (in eng), *Sci Rep,* vol. 6, p. 27619, Jun 8 2016, doi: 10.1038/srep27619.
8. C. Cavaliere et al., "Computer-Aided Diagnosis of Multiple Sclerosis Using a Support Vector Machine and Optical Coherence Tomography Features," (in eng), *Sensors (Basel),* vol. 19, no. 23, Dec 3 2019, doi: 10.3390/s19235323.
9. M. Kang, S. Shin, G. Zhang, J. Jung, and Y. T. Kim, "Mental Stress Classification Based on a Support Vector Machine and Naive Bayes Using Electrocardiogram Signals," (in eng), *Sensors (Basel),* vol. 21, no. 23, Nov 27 2021, doi: 10.3390/s21237916.
10. G. Cohen and R. Meyer, "Optimal asymmetrical SVM using pattern search. A health care application," (in eng), *Stud Health Technol Inform,* vol. 169, pp. 554–8, 2011

13.7 Lab

13.7.1 Working Example in Python

In this section, we will create a support vector machine classifier model, test it, and optimize it. Start by downloading Iris dataset using the following link: https://www.kaggle.com/datasets/arshid/iris-flower-dataset. Alternatively, you can use the following code to load the dataset directly into your code.

```
iris = datasets.load_iris()
x = iris.data[:, :4]
y = iris.target
```

The iris dataset describes the properties of flowers. It includes three iris species within 50 samples. This dataset includes the following columns:

- Petal length: petal length for the Iris
- Petal width: petal width for the Iris
- Sepal length: sepal length for the Iris
- Sepal width: sepal width for the Iris
- Species: class of the iris (there are three species in the dataset)

13.7.1.1 Loading Iris Dataset

Start by importing the required libraries and loading the dataset (Fig. 13.12).

13.7.1.1.1 Visualize Iris Dataset

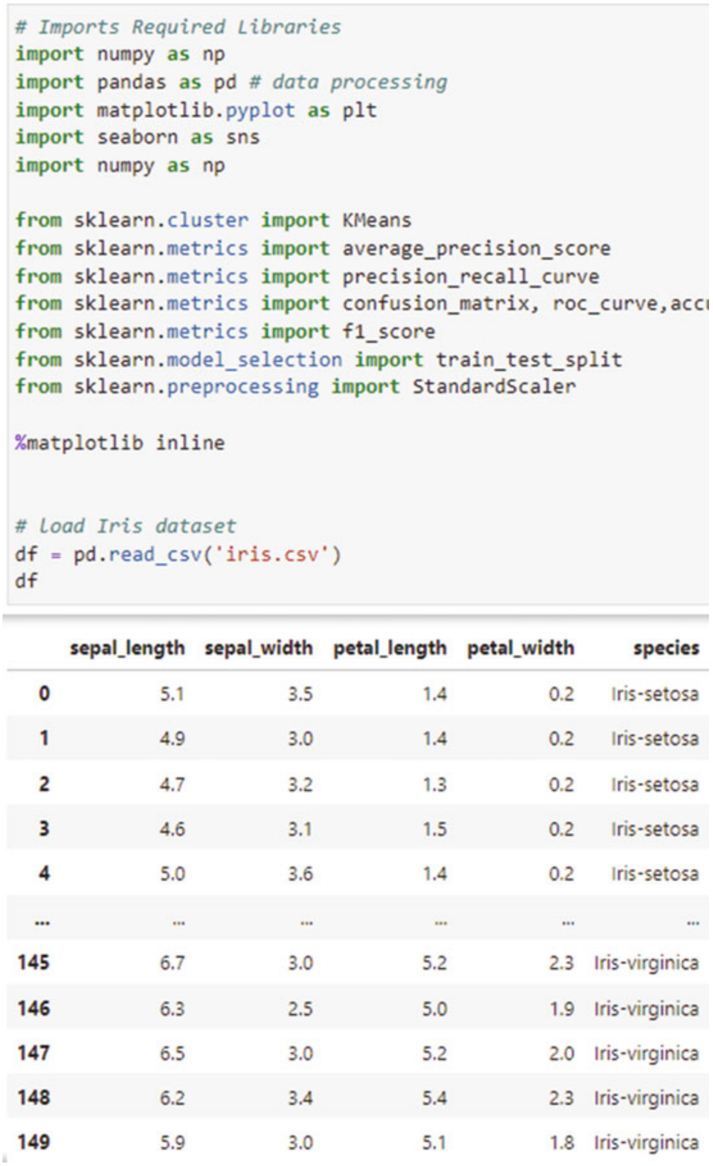
Visualizing the dataset can be done in many ways, one is demonstrated in Fig. 13.13.

13.7.1.2 Preprocess and Scale Data

We need to replace the categorical target with numeric values, split the dataset into training and testing datasets, and standardize both sets (Fig. 13.14).

13.7.1.3 Dimension Reduction

We can now create a support vector model (SVM) using an RBF kernel and C=100. The dimension of the feature matrix is low (i.e., 4); however, for illustration purposes, we will use the Principal Component analysis (PCA) to reduce the number

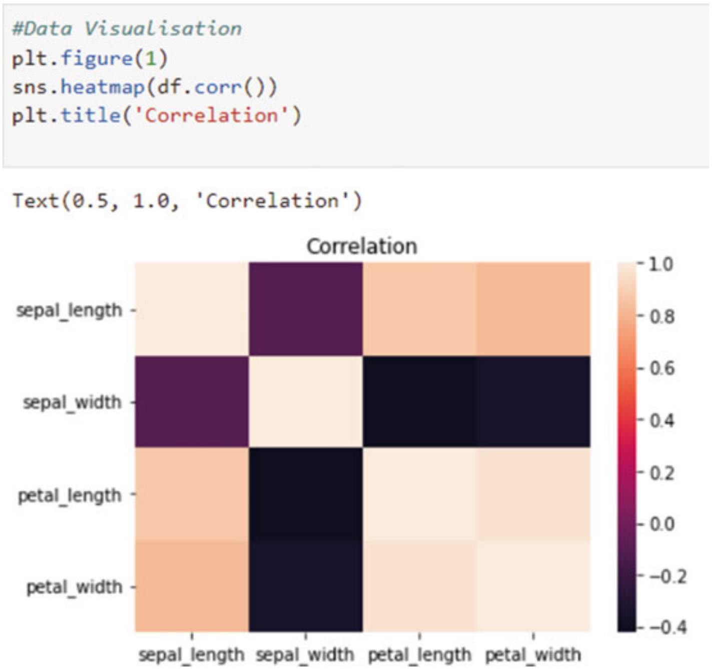


```
# Imports Required Libraries
import numpy as np
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.cluster import KMeans
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix, roc_curve,acc
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

%matplotlib inline


# Load Iris dataset
df = pd.read_csv('iris.csv')
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Fig. 13.12 Loading the Iris dataset into pandas

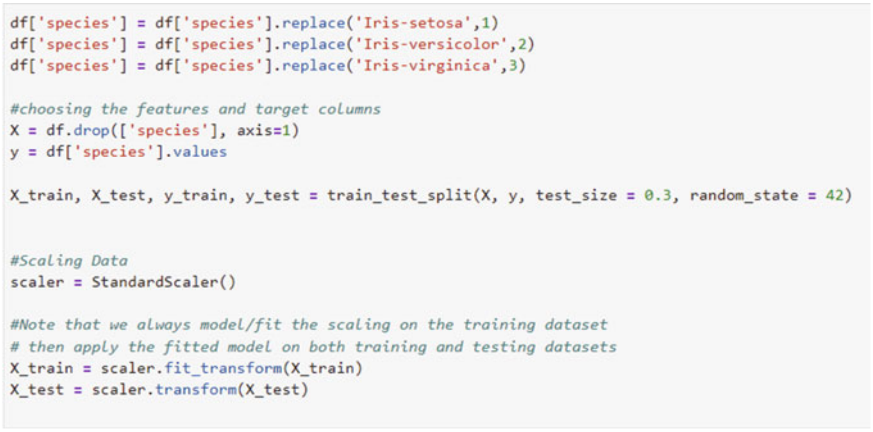
of features to 2. Once PCA is create, we apply it to the x_tran and x_test. A two-dimension feature matrix will allow us to plot the SVM results in two dimensions which clarifies the end result. Instead of using PCA, and for illustration purposes, you could have opted to choose two of the four dimensions, such as sepal width and petal width (Fig. 13.15).



```
#Data Visualisation
plt.figure(1)
sns.heatmap(df.corr())
plt.title('Correlation')
```

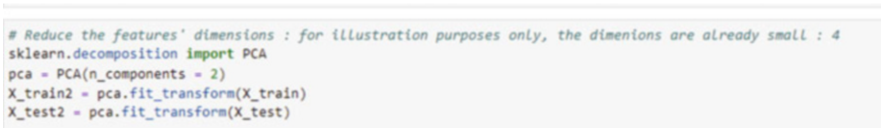
```
Text(0.5, 1.0, 'Correlation')
```

Fig. 13.13 Visualizing iris dataset



```
df['species'] = df['species'].replace('Iris-setosa',1)
df['species'] = df['species'].replace('Iris-versicolor',2)
df['species'] = df['species'].replace('Iris-virginica',3)

#choosing the features and target columns
X = df.drop(['species'], axis=1)
y = df['species'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)


#Scaling Data
scaler = StandardScaler()

#Note that we always model/fit the scaling on the training dataset
# then apply the fitted model on both training and testing datasets
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Fig. 13.14 Preprocess and scale iris dataset



```
# Reduce the features' dimensions : for illustration purposes only, the dimenions are already small : 4
sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
X_test2 = pca.fit_transform(X_test)
```

Fig. 13.15 Creating support vector machine

13.7.1.4 Hyperparameter Tuning and Performance Measurements

Using GridSearch, we can now seek hyperparameter tuning for the SVC. One the optimal model is found, we fit it to the training dataset and make predictions on the testing dataset to display the classification report and the AUC (Fig. 13.16).

Optionally, we can display the confusion matrix (Fig. 13.17).

13.7.1.5 Plot the Decision Boundaries

Finally, we can plot the decision boundaries between classes (Figs. 13.18 and 13.19).

13.7.2 Do It Yourself

13.7.2.1 The Iris Dataset Revisited

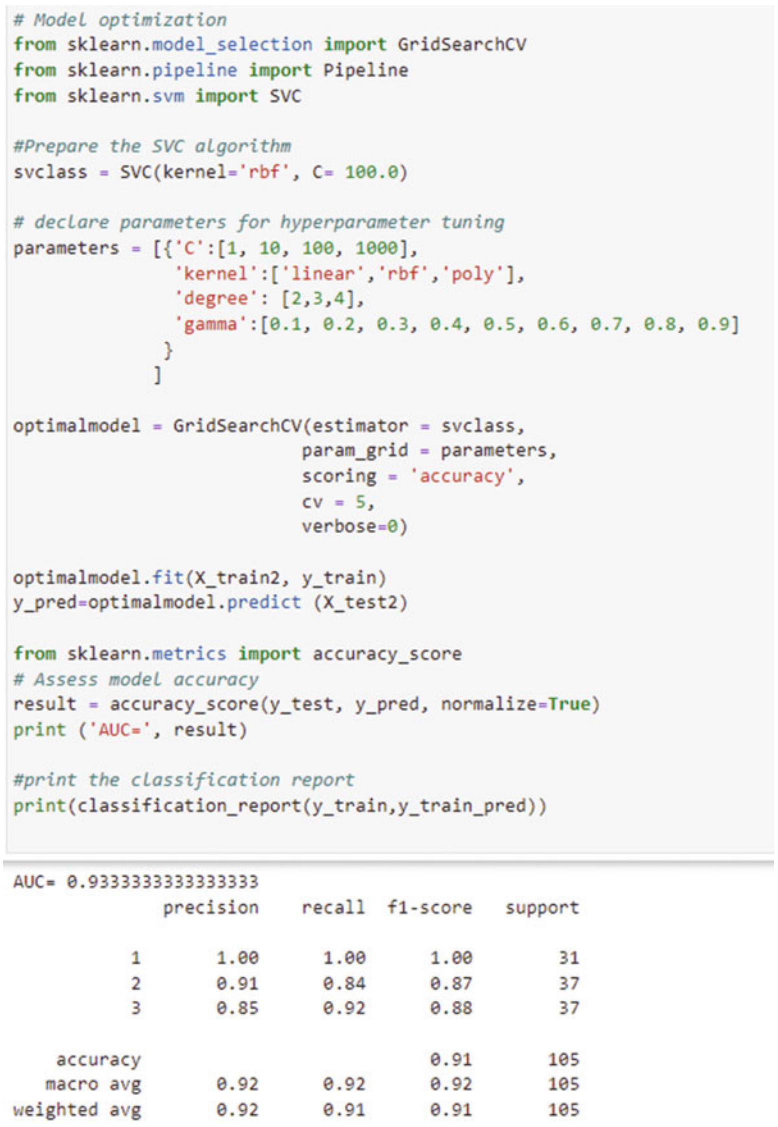
In Sect. 13.7 above, we applied PCA to reduce the number of features to 2

1. Instead of PCA choose to drop the petal length and sepal length and check how the MVC performance chance.
2. Instead of PCA choose to drop the petal width and sepal width and check how the MVC performance chance.
3. Which one lead to better results? Can you know in advance what is more likely to lead to good performance by looking at the pair plots? We did not display the pair plots, so display them and check visually to see if you can gain an insight about the better choice.

13.7.2.2 Breast Cancer

Use the breast cancer dataset that can download from the following link: https://www.kaggle.com/code/buddhiniw/breast-cancer-prediction/data.

1. Create an SVM model to solve this classification problem.
2. Now that you know several classifiers, create a lab where you use three classifiers including an SVM and compare their performance. Conclude by choosing the best performing classifier. Always give a rational for your choices.



```
# Model optimization
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

#Prepare the SVC algorithm
svclass = SVC(kernel='rbf', C= 100.0)

# declare parameters for hyperparameter tuning
parameters = [{'C':[1, 10, 100, 1000],
               'kernel':['linear','rbf','poly'],
               'degree': [2,3,4],
               'gamma':[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
              }
             ]

optimalmodel = GridSearchCV(estimator = svclass,
                            param_grid = parameters,
                            scoring = 'accuracy',
                            cv = 5,
                            verbose=0)

optimalmodel.fit(X_train2, y_train)
y_pred=optimalmodel.predict (X_test2)

from sklearn.metrics import accuracy_score
# Assess model accuracy
result = accuracy_score(y_test, y_pred, normalize=True)
print ('AUC=', result)

#print the classification report
print(classification_report(y_train,y_train_pred))
```

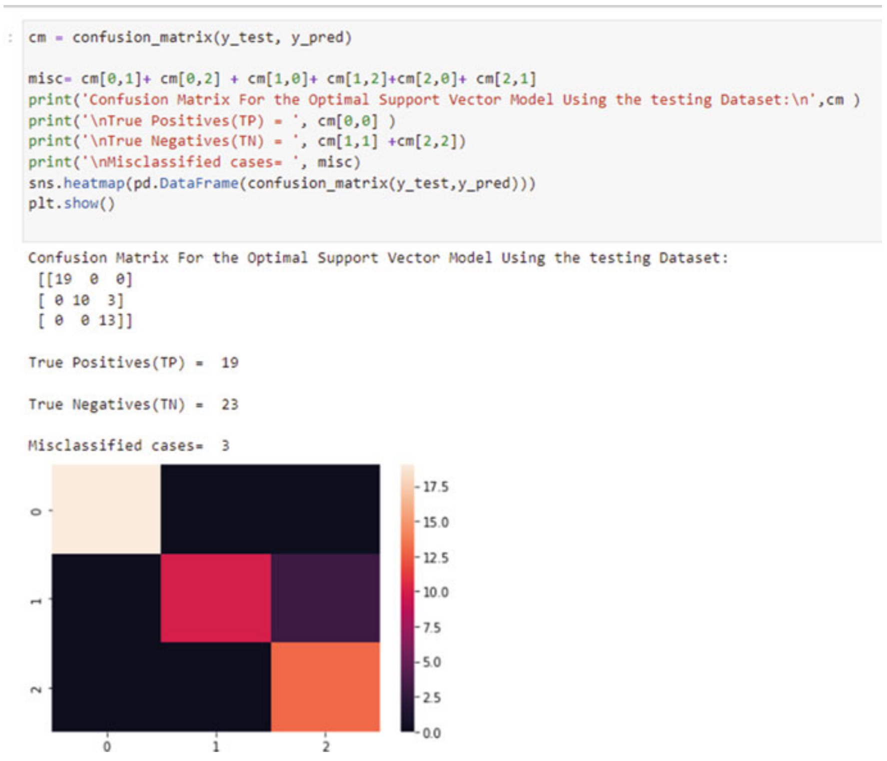
```
AUC= 0.9333333333333333
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        31
           2       0.91      0.84      0.87        37
           3       0.85      0.92      0.88        37

    accuracy                           0.91       105
   macro avg       0.92      0.92      0.92       105
weighted avg       0.92      0.91      0.91       105
```

Fig. 13.16 Decision plot for iris species

13.7.2.3 Wine Classification

Use the wine dataset that can be downloaded from the following link: https://archive.ics.uci.edu/ml/datasets/wine

You can also contemplate using the built “load_wine”



```
cm = confusion_matrix(y_test, y_pred)

misc= cm[0,1]+ cm[0,2] + cm[1,0]+ cm[1,2]+cm[2,0]+ cm[2,1]
print('Confusion Matrix For the Optimal Support Vector Model Using the testing Dataset:\n',cm )
print('\nTrue Positives(TP) = ', cm[0,0] )
print('\nTrue Negatives(TN) = ', cm[1,1] +cm[2,2])
print('\nMisclassified cases= ', misc)
sns.heatmap(pd.DataFrame(confusion_matrix(y_test,y_pred)))
plt.show()
```

```
Confusion Matrix For the Optimal Support Vector Model Using the testing Dataset:
 [[19  0  0]
 [ 0 10  3]
 [ 0  0 13]]

True Positives(TP) =  19

True Negatives(TN) =  23

Misclassified cases=  3
```

Fig. 13.17 Confusion matrix resulting from the optimal model

```
from sklearn.datasets import load_wine
wine_data= sklearn.datasets.load_wine()
```

There are three types of wine, so this is a multi-class problem. Create a model to predict the wine the using SVM (hint: use the SVC with decision_functino_shape =‘ovr’ and degree=3).

13.7.2.4 Face Recognition

You might need to install the Python image library called pillow: pip install pillow

1. Load the images dataset using the following code

```
from sklearn.datasets import fetch_lfw_people
data = fetch_lfw_people(min_faces_per_person=50) # read only
those with 50 images or more
```

```
from mlxtend.plotting import plot_decision_regions # you need to: pip install mlxtend
#plot the classification
pldec = plot_decision_regions(X_test2, y_test, clf=optimalmodel, legend=0)

# Plot decision regions
# Adding axes annotations
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('SVM classification result on the Iris dataset')
handles, labels = pldec.get_legend_handles_labels()
pldec.legend(handles,
          ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
           framealpha=0.3, scatterpoints=1)

plt.show()
```

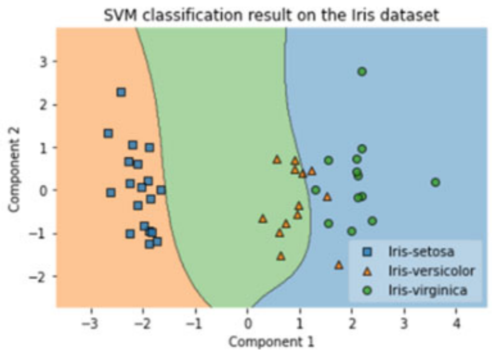


Fig. 13.18 Plotting the decision boundaries in 2D

```
# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=100.0  using testing dataset: {0:0.4f} %'.
      format(accuracy_score(y_test, y_test_pred)*100))
print(classification_report(y_test,y_test_pred))
```

```
Model accuracy score with rbf kernel and C=100.0  using testing dataset: 86.6667 %
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        19
           2       1.00      0.54      0.70        13
           3       0.68      1.00      0.81        13

    accuracy                           0.87        45
   macro avg       0.89      0.85      0.84        45
weighted avg       0.91      0.87      0.86        45
```

Fig. 13.19 Calculating accuracy, recall, and precision metrics for SVM using testing dataset

2. Display on the screen the number of instances in each target class
3. Do you notice any imbalance in the classes? Clarify.
4. Try to plot few images on the screen

5. Split the dataset into training and testing datasets
6. Create an SVM variable (if there were imbalance in classes, then use a class_weight parameter)
7. Grid search for the optimal model
8. Display the best parameters and the best model
9. Fit the optimal model on the training dataset
10. Use the fitted model to predict using the testing dataset
11. Display a classification report (use classification_report from Sklearn)
12. Let's take one further step. PCA might help you boost the model's performance. Apply PCA before rerunning the SVM grid search and check if the performance is better.

13.7.2.5 SVM Regressor: Predict House Prices with SVR

Support vector machine can be used not only as classifiers but as regressors too. Create a support vector model regressor to predict house prices, using the housing dataset that can be downloaded using the following link: https://www.kaggle.com/datasets/huyngohoang/housingcsv.

The housing dataset provides the sale price of houses across the United States. This dataset includes the following columns:

- Avg. Area Income: the average income in the area where the house is located.
- Avg. Area House Age: the average house age in the area where the house is located.
- Avg. Area Number of Rooms: the average number of rooms for a house in the area where the house is located.
- Avg. Area Number of Bedrooms: the average number of bedrooms for a house in the area where it is located.
- Area Population: the population in the area where the house is located.
- Price: the sale price of the house.
- Address: the house address.

Hint: explore the SVR following this link: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

13.7.2.6 SVM Regressor: Predict Diabetes with SVR

1. Load the diabetes dataset using the following code:

```
SVM Regressor: Predict house prices with SVR
```

2. Store at least 30 samples in a testing dataset
3. Proceed with GridSearch using the following values
   (a) alpha: [1e-7, 1e-6, 1e-5, 1e-4]
   (b) penalty: [None, ‘l2’]
   (c) eta0: [0.03, 0.04, 0.05, 0.1]
   (d) max_itr: [500, 1000]
4. After fitting the optimal model, display the best parameters and best estimstor
5. Make prediction and display the optimal model performance (i.e., MAE, MSE, R2)

13.7.2.7 Unsupervised SVM

Support vector machine can be used not only as supervised but unsupervised too.

Create an unsupervised support vector machine regressor to predict house prices, using the housing dataset.

Hint: explore the One class SVM following this link: https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

13.7.3 Do More Yourself

Use the following datasets and create linear and nonlinear support vector machines to solve the classification problems associated with these datasets. Also, try several algorithms to solve each and choose the best model.

- https://www.kaggle.com/datasets/paultimothymooney/stock-market-data
- https://www.kaggle.com/code/startupsci/titanic-data-science-solutions/data
- https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction
- https://www.kaggle.com/datasets/elikplim/forest-fires-data-set

References

1. S. Osowski, K. Siwek, T. Markiewicz, MLP and SVM networks—a comparative study, in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004, 11-11 June 2004,* pp. 37–40 (2004).
2. V.R. Jakkula, Tutorial on support vector machine (SVM). Sch. EECS Wash. State Uni. **37**(2.5), 3 (2011). [Online]. Available: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf.
3. S. Huang, N. Cai, P.P. Pacheco, S. Narrandes, Y. Wang, W. Xu, Applications of support vector machine (SVM) learning in cancer genomics (in eng). Cancer Genomics Proteomics **15**(1), 41–51 (2018). https://doi.org/10.21873/cgp.20063
4. V.N. Vapnik, Pattern recognition using generalized portrait method. Autom. Remote Control **24**, 774–780 (1963)

5. R. Gandhi, Support vector machine—introduction to machine learning algorithms. *Medium* (2022). https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed 11 Mar 2022.
6. A. Ng, Part V, Support Vector Machines. See stanford.edu (2022). https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf. Accessed 14 Mar 2022.
7. W. Zouhri, L. Homri, J.-Y. Dantan, Handling the impact of feature uncertainties on SVM: A robust approach based on Sobol sensitivity analysis. Expert Syst. Appl. **189**, 115691 (2022). https://doi.org/10.1016/j.eswa.2021.115691
8. R. Berwick, An idiot's guide to support vector machines (SVMs). Village Idiot (2022). https://web.mit.edu/6.034/wwwbob/svm.pdf. Accessed 14 Mar 2022
9. A. Jana, Support vector machines for beginners—Linear SVM—A developer diary. *A Developer Diary* (2022). http://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-linear-svm/. Accessed 11 Mar 2022.
10. K. Jain, What is support vector machine? *Medium* (2022). https://towardsdatascience.com/what-is-support-vector-machine-870a0171e690. Accessed 14 Mar 2022.
11. F. Rossi, N. Villa, Support vector machine for functional data classification. Neurocomputing **69**(7), 730–742 (2006). https://doi.org/10.1016/j.neucom.2005.12.010
12. E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, in *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop, 24–26 Sept. 1997*, pp. 276–285 (1997). https://doi.org/10.1109/NNSP.1997.622408.
13. N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, Cambridge, 2000)
14. A.J. Smola, Regression estimation with support vector learning machines, Technische Universität München, München (1996). [Online]. Available: http://alex.smola.org/papers/1996/Smola96.pdf
15. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
16. M.E. Cholette, P. Borghesani, E.D. Gialleonardo, F. Braghin, Using support vector machines for the computationally efficient identification of acceptable design parameters in computer-aided engineering applications. Expert Syst. Appl. **81**(C), 39–52 (2017). https://doi.org/10.1016/j.eswa.2017.03.050
17. L.M.R. Baccarini, V.V. Rocha e Silva, B.R. de Menezes, W.M. Caminhas, SVM practical industrial application for mechanical faults diagnostic. Expert Syst. Appl. **38**(6), 6980–6984 (2011). https://doi.org/10.1016/j.eswa.2010.12.017
18. Z. Wei, Y. Feng, Z. Hong, R. Qu, J. Tan, Product quality improvement method in manufacturing process based on kernel optimisation algorithm. Int. J. Prod. Res. **55**(19), 5597–5608 (2017). https://doi.org/10.1080/00207543.2017.1324223
19. H. Rostami, J.-Y. Dantan, L. Homri, Review of data mining applications for quality assessment in manufacturing industry: support vector machines. Int. J. Metrol. Qual. Eng. **6**(4), 401 (2015). [Online]. Available: https://doi.org/10.1051/ijmqe/2015023.
20. C.J.C. Burges, B. Schölkopf, A.J. Smola, *Advances in Kernel Methods: Support Vector Learning* (MIT Press, Cambridge, MA, 1999)