Support Vector Machines

Presented by:

(1) Easwari Arumuga Perumal
(2) Maniraaj Senthil Nathan
(3) Sooriya Mathy

Instructor: Dr.Yasser Alginahi

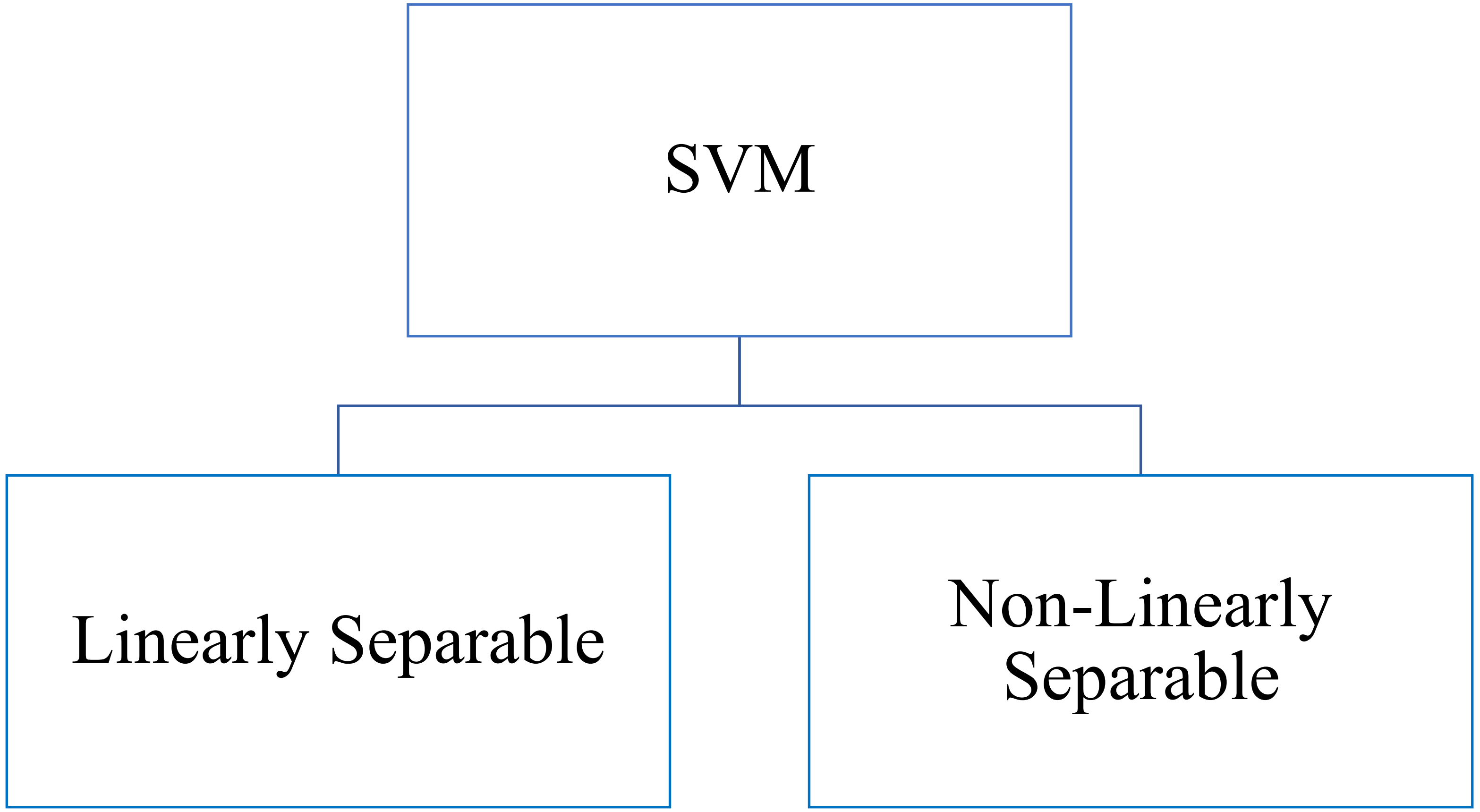
Date:10/20/2023

University of Windsor

Introduction

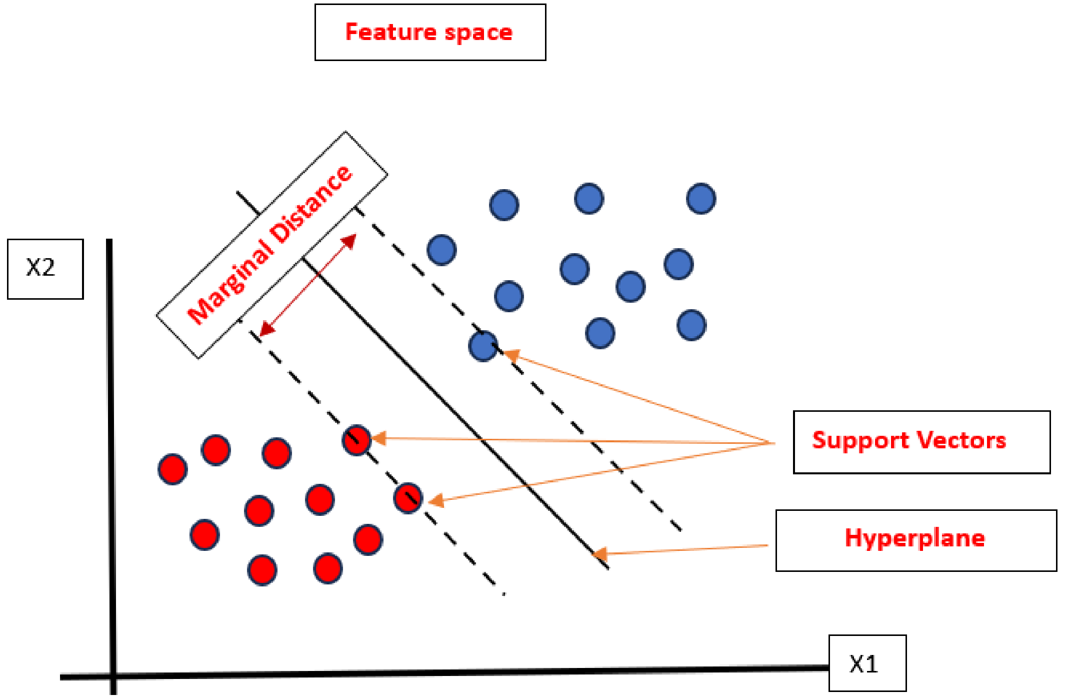
- What is Support Vector Machines?
- Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It has a high accuracy in many applications, and it uses a concept of hyperplane that best separates the data[1].
- Used across various domains, from image recognition to finance and medical diagnosis.

Types of SVM



SVM Working

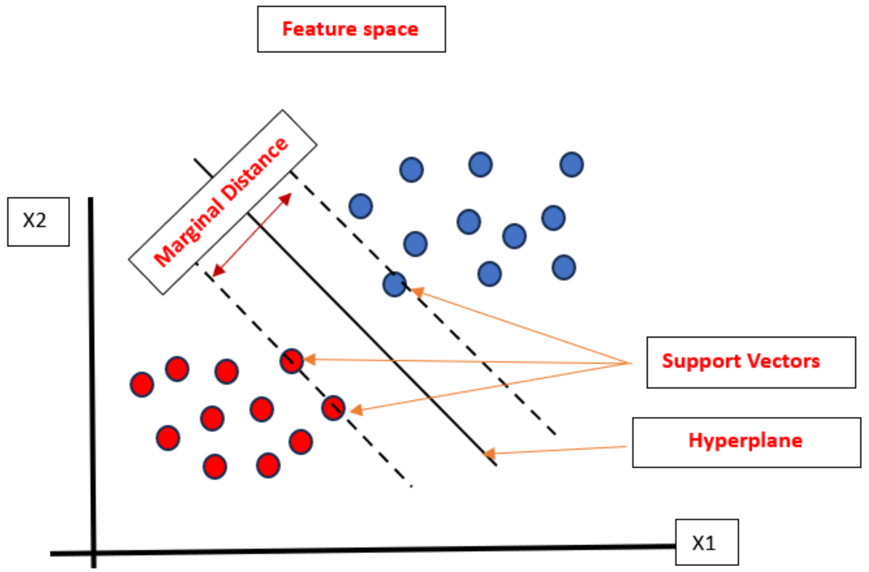
- The main objective of SVM algorithm is to find a ***hyperplane*** in an ***n-dimensional feature*** space that separates the data with the largest ***marginal distance*** possible. The attributes of the hyperplane depend on the ***Support Vectors***.



Scatter Plot Showing Support Vector Machine Working

Implementation

- Find a hyperplane in an n-dimensional feature space.
- Maximize the margin between the hyperplane and data points.
- The attributes of the hyperplane depend on the Support Vectors.
- Achieve optimal data separation for accurate classification and regression.



Scatter Plot Showing State Vector Machine Working

Terminologies

- **Hyperplane**: The n- dimensional plane that separates the data . The dimension of the hyperplane depends on the number of features (n).
- **Support Vectors**: Support vectors are the data points that are close to the decision boundary, they are the data points most difficult to classify, they hold the key for SVM to be optimal decision surface[2].
- **Marginal distance**: It is the perpendicular distance from the optimal hyperplane to the nearest support vectors. It is essentially the distance between the hyperplane and the support vectors[2].

Mathematical Intuition of SVM

- We know that the linear hyperplane is given as,

$$w^T.x + b = 0$$

Where,

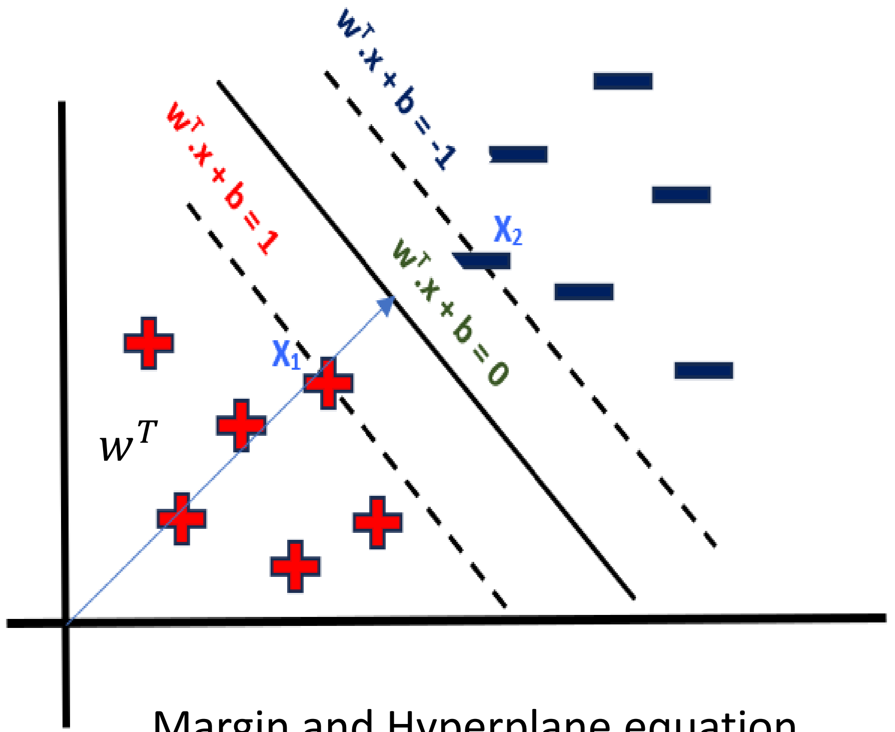
w^T is the weight vector normal to hyperplane

b is the offset of hyperplane from origin

- Thus, the margins of the positive side (X_1) and negative side (X_2) of hyperplane is taken as,

$$w^T.x + b = 1 \ ,$$
$$w^T.x + b = -1$$

respectively.



Margin and Hyperplane equation

Mathematical Intuition of SVM

- Firstly, to classify a data point as negative or positive, we define a decision rule such that, if

$$\boxed{\begin{array}{c} \text{Y}= w^T \cdot X_i + b > 0, \text{ for positive data} \\ \text{Y}= w^T \cdot X_i + b < 0, \text{ for negative data} \end{array}}$$

- Now it is also significant to consider the constraint that, there should be no data points lying in between the hyperplane and the margins. Thus,

$$\boxed{\begin{array}{c} w^T \cdot X_1 + b \geq 1, \text{ for positive data} \\ w^T \cdot X_2 + b \leq -1, \text{ for negative data} \end{array}}$$

Mathematical Intuition of SVM

- Thus, considering the positive classes have y = +1 and y = -1 for negative class the decision condition can be optimized as

$$\boxed{\text{Yi}\,(w^T \cdot X + b\,) \geq 1}$$

Mathematical Intuition of SVM

- Now, to model a hyperplane that has the largest marginal distance with the
- above constraint. Hence,
- Distance can be derived as,

$$d = (x_1 - x_2) \cdot \frac{\vec{W}}{\|W\|}$$

- As the support vectors x1 and x2 lies on the hyperplane, Yi (w^T.x + b) = 1
- Thus, substituting x1 and x2 and deriving to the lowest form we get,

$$d = \frac{2}{|w|}$$

Mathematical Intuition of SVM

- To achieve the optimal SVM model it is essential to maximize the distance $\frac{2}{|w|}$ which is similar to minimizing $\frac{|w|}{2}$
- Therefore, the optimization function can be derived as,

$$\boxed{argmin(w^*, b^*)\frac{1}{2}||w||^2 \text{ such that, } y_i\,(w^T.x + b) \geq 1}$$

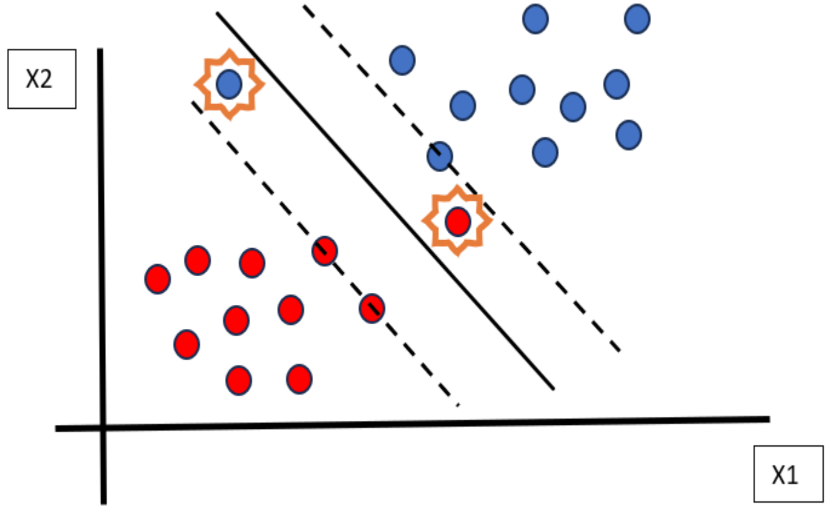
- This function does not include any data point that is lying on the wrong side of hyper plane. This is termed as **Hard Margin** and it does not tolerate any misclassification[1].

Dual Problem

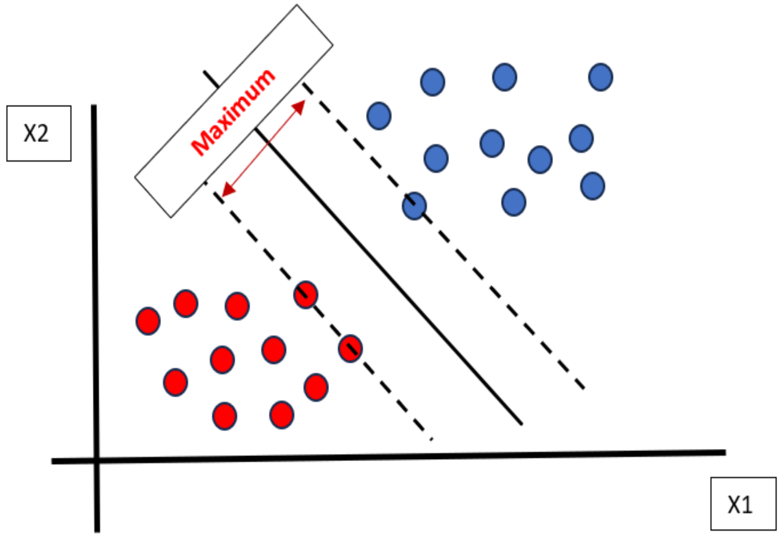
- But just minimizing the w gives only a constraint optimization problem as there would be outliers in real world data.
- The "dual problem" is a mathematical formulation used to find the Lagrangian multipliers that are essential for solving the primal problem of maximizing the margin while minimizing classification errors[3].
- Lagrangian multipliers are used in functions to account for constraint conditions while it simultaneously maximizes or minimizes the objective function.

Dual Problem

- The two constraints in the dual problem of SVM are,
  - ➢Outliers that lies on wrong side of hyperplane should be optimized.
  - ➢Maximizing the marginal distance by minimizing the $|w|$.
- This can be achieved my introducing a Lagrangian multiplier ζ that maximizes the dual objective function and a cost function C.
- Thus, we arrive at SVM with **Soft Margin** that allows some degree of misclassification.



Misclassified data points constraint



Maximizing margin constraint

Optimization Function

- Thus, we derive at the optimization function of SVM for Soft Margin,

$$\boxed{\boldsymbol{argmin}(\boldsymbol{w}^{*}, \boldsymbol{b}^{*})\frac{\mathbf{1}}{\mathbf{2}}||\boldsymbol{w}|| + \boldsymbol{C}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{\zeta}_{\boldsymbol{i}}}$$

Where, C is the cost function hyperparameter that can be tuned

ζ represents the distance of misclassified point from their respective margins

i is the datapoint lying on wrong side of hyperplane.

Optimization Function

- The Cost function C controls the trade-off between maximizing the margin and minimizing the classification error.
- the value of C is determined by trial and error process and choosing the Value of C that best fits the model using the validation metrics.
- This type of SVM is called C-SVM and to resolve this discrepancy Hyperparameter tuning, we use a alternative formulation of SVM called **nu-SVM.**
- **Nu-SVM** introduces the "**nu**" **parameter** , which is a user defined value and has a range from 0 to 1. The value of nu depends on the fraction of training errors and support vectors.
- Thus, this is the final optimized model for linear SVM.

Why we need Non-Linear SVM:

- Datasets that are linearly separable with some noise work out great:

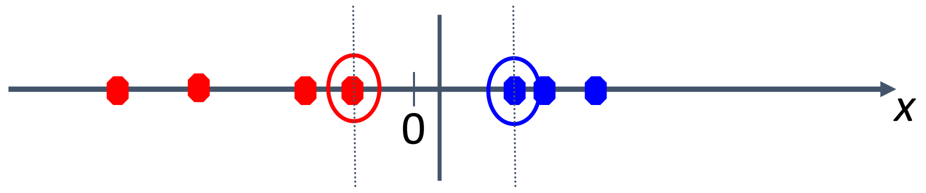


Figure showing linear data in 1-Dimension

- But what are we going to do if the dataset is complex?

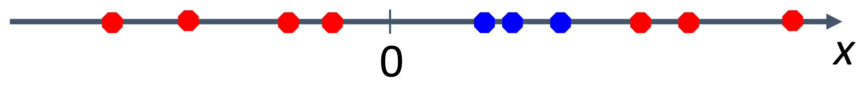


Figure showing non-linear data in 1-Dimension

Why we need Non-Linear SVM:

- Let’s map the given data to high-dimensional Space:

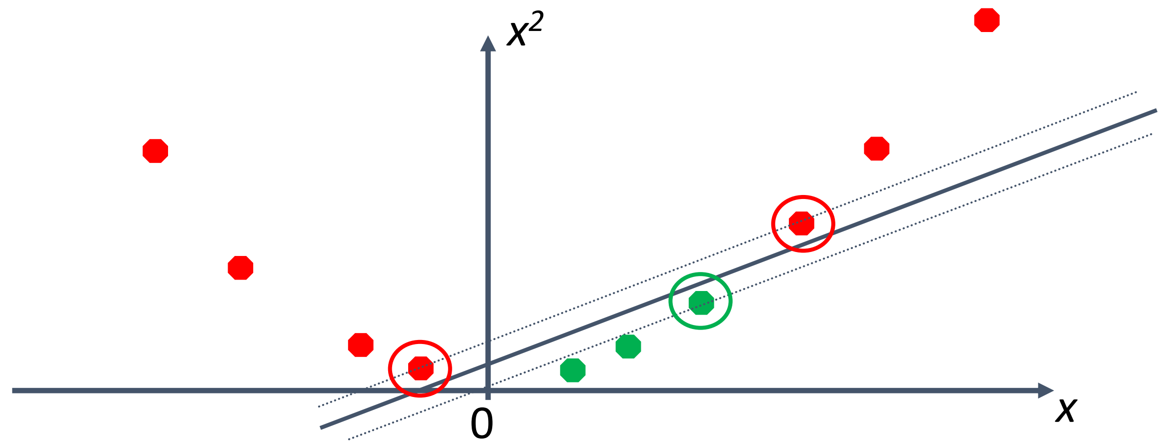


Figure showing Non-Linear Data in 2-Dimension

Non-Linear SVM and Kernel Function

- From the previous slide we can witness that if the data points are remapped into high-dimensional space, the non-linear data becomes easily separable. This is done using a Kernel function in SVM.
- The **Kernel Function** is a critical component employed to transform data from its original feature space into a higher-dimensional space.
- A kernel function, denoted as K(x, y), takes two data points, x and y, as input and computes the dot product of these points in the higher-dimensional space without explicitly mapping them to that space. Mathematically, this can be represented as:

$$K(x, y) = \Phi(x) \cdot \Phi(y)$$

Where, $\Phi(x)$ and $\Phi(y)$ represent the transformation of the data points x and y into the higher-dimensional space.

Types of SVM Kernels

- Th various types of kernel function used to transform non- linear data are,
  - ➢Linear Kernel:
  - ➢K$(x_1, x_2) = x_1^T . x_2$
  - ➢Polynomial Kernel:
  - ➢K$(x_1, x_2) = (x_1^T .\ x_2 + r)^d$
  - ➢Radial Bias Function (rbf) Kernel:
  - ➢K(x_1, x_2) =exp$(- \gamma . \| x_1 - x_2 \|)^2$

Advantages of SVM:

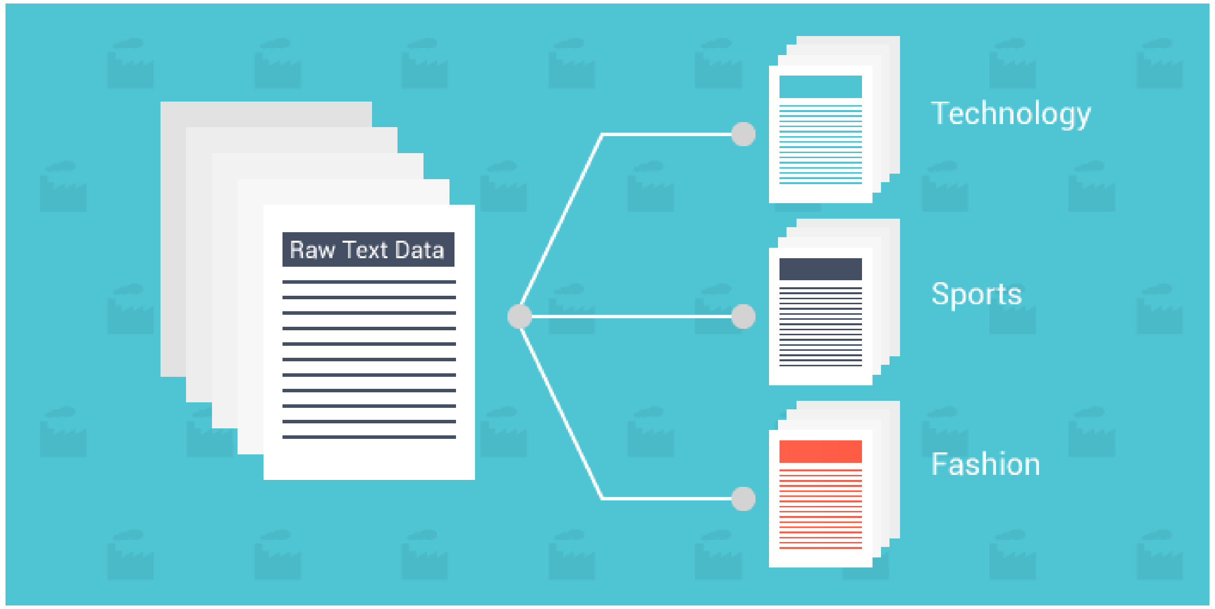
- Effective in high-dimensional cases.
- Its memory is efficient as it uses a subset of training points in the decision function called support vectors.
- SVM are robust to overfitting as it uses a large margin or soft margin approach. This makes them suitable for scenarios where the data is noisy, or the number of features is high.
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels.

Disadvantages of SVM:

- Complexity in Parameter Tuning
- Slower Training Time for Large Datasets
- SVM is inherently a binary classifier, and extensions to multi-class problems may require techniques like one-vs-all (OvA) or one-vs-one (OvO), which can be computationally expensive.

Applications:

- Text Classification
- Cancer Diagnosis
- Object Detection



[4]

University of Windsor

Code for Example

Support Vector Machine Example-1

Support Vector Machine Example-2

Code Explanation Example 1

- The aim of the program is to understand how the Support Vector machine works using a simple 2D hard coded data set.
- Import the necessary packages.
- Using numpy array, a data set is created and named as X
- An array of 0’s and 1’s is created and named as Y. This is the Binary labels. 0 and 1 are two labels indicating 2 classes.

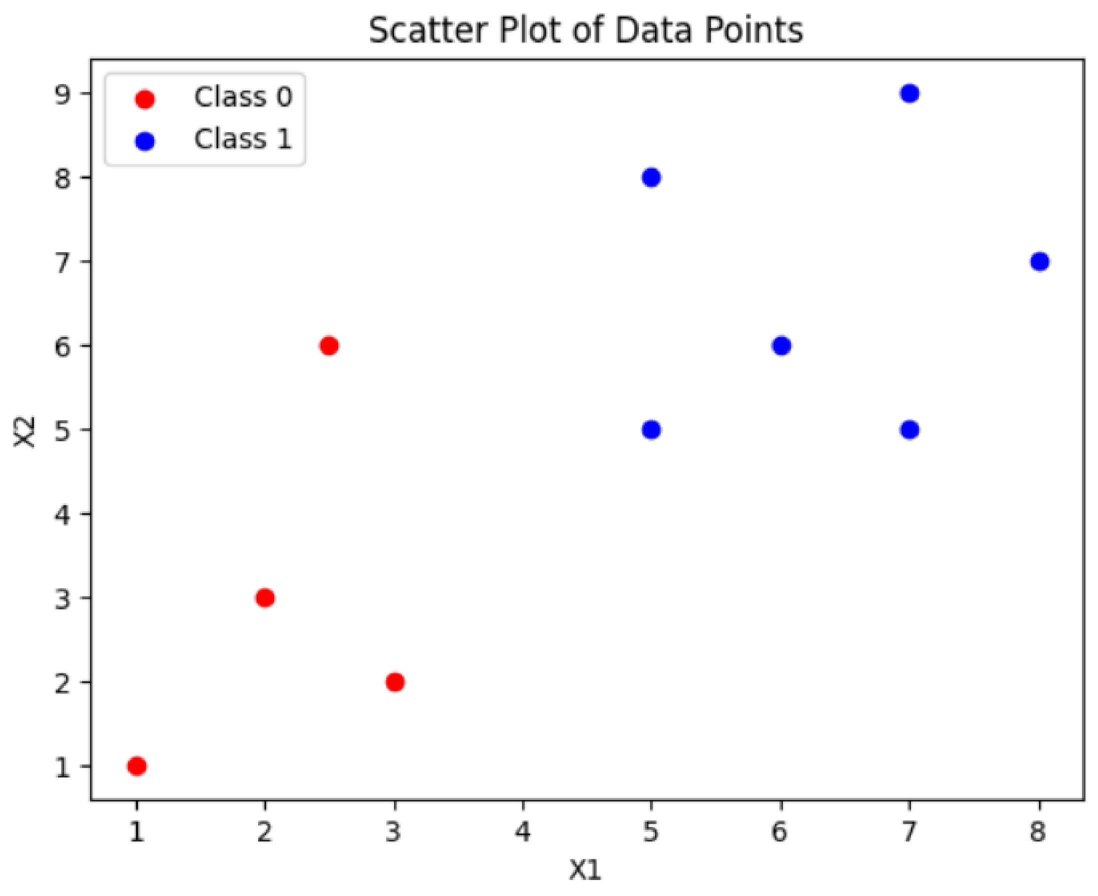
```
X = np.array([[1,1],
             [2,3],
             [2.5,6],
             [3,2],
             [5,5],
             [7,5],
             [8,7],
             [6,6],
             [5,8],
             [7,9]])
```

```
y = np.array([0,0,0,0,1,1,1,1,1,1])
```

Code Explanation Example 1

- Map the y labels with the data set created in X.
- Scatter plot of data points mapped to the labels is created.

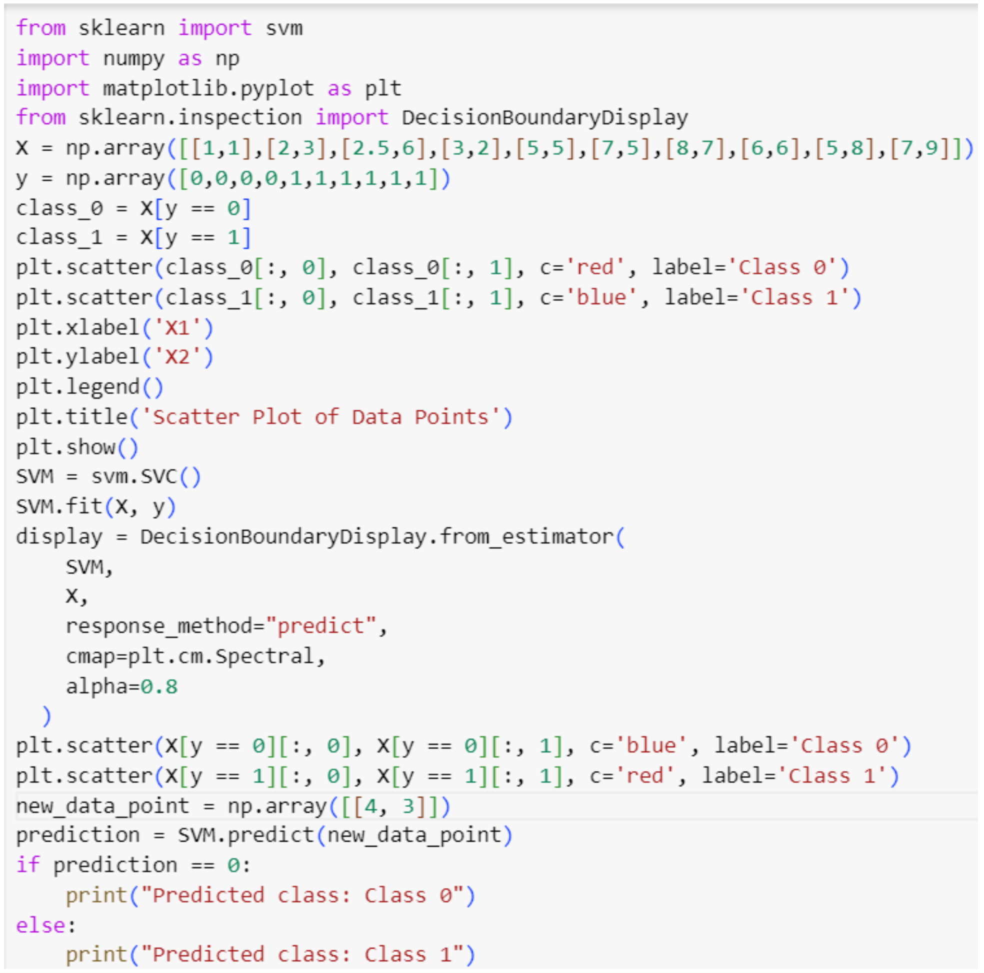
```
class_0 = X[y == 0]
class_1 = X[y == 1]
plt.scatter(class_0[:, 0], class_0[:, 1], c='red', label='Class 0')
plt.scatter(class_1[:, 0], class_1[:, 1], c='blue', label='Class 1')
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend()
plt.title('Scatter Plot of Data Points')
plt.show()
```



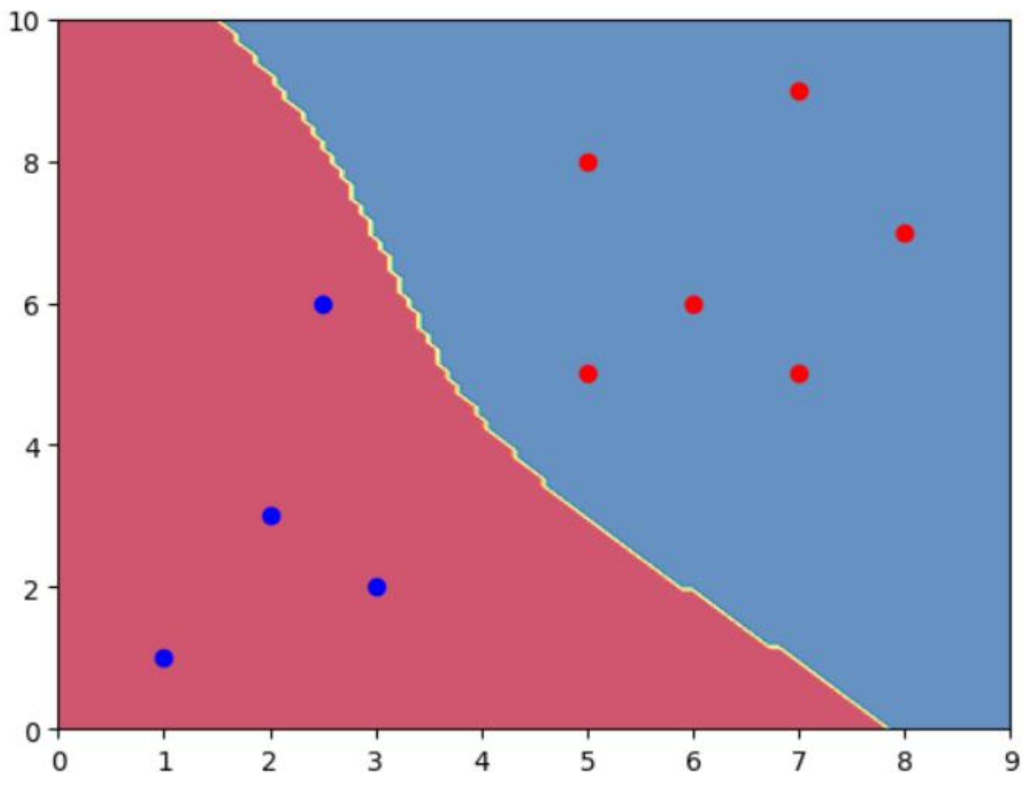
Code Explanation Example 1

- SVM model is built for the above dataset using SVC function
- Train the SVM model with the X and y values.
- With the help of scatter plot and DecisionBoundayDisplay function, we can visualise the working of SVM model.
- To check the model, take any data point from the plot and using if condition check which label the data point belongs to.

Code and Output Example 1



```python
from sklearn import svm
import numpy as np
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
X = np.array([[1,1],[2,3],[2.5,6],[3,2],[5,5],[7,5],[8,7],[6,6],[5,8],[7,9]])
y = np.array([0,0,0,0,1,1,1,1,1,1])
class_0 = X[y == 0]
class_1 = X[y == 1]
plt.scatter(class_0[:, 0], class_0[:, 1], c='red', label='Class 0')
plt.scatter(class_1[:, 0], class_1[:, 1], c='blue', label='Class 1')
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend()
plt.title('Scatter Plot of Data Points')
plt.show()
SVM = svm.SVC()
SVM.fit(X, y)
display = DecisionBoundaryDisplay.from_estimator(
    SVM,
    X,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8
  )
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], c='blue', label='Class 0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], c='red', label='Class 1')
new_data_point = np.array([[4, 3]])
prediction = SVM.predict(new_data_point)
if prediction == 0:
    print("Predicted class: Class 0")
else:
    print("Predicted class: Class 1")
```



Code Explanation Example 2

- The aim of the program is to use the existing breast cancer data set and an SVM model to predict whether a cancer's stage is malignant or benign.
- Import necessary packages and load the breast cancer dataset.
- Prediction is done using the dataset's mean radius and mean texture data. This is taken as X.
- Target is taken as y variable as it contains binary values 0 and 1. 0 for malignant and 1 for benign.

Code Explanation Example 2

- SVM model is built for the above dataset using SVC function. Here we are using kernel as linear. Alternatively, we can use any type of kernel.
- Train the model with the X and y values.
- With the help of predict method, the stage of the cancer is predict based on the trained values.
- The accuracy of the model is then calculated with accuracy_score.
- The model is visualised with the help of an inbuilt DecisionBoundaryDisplay function and scatter plot.

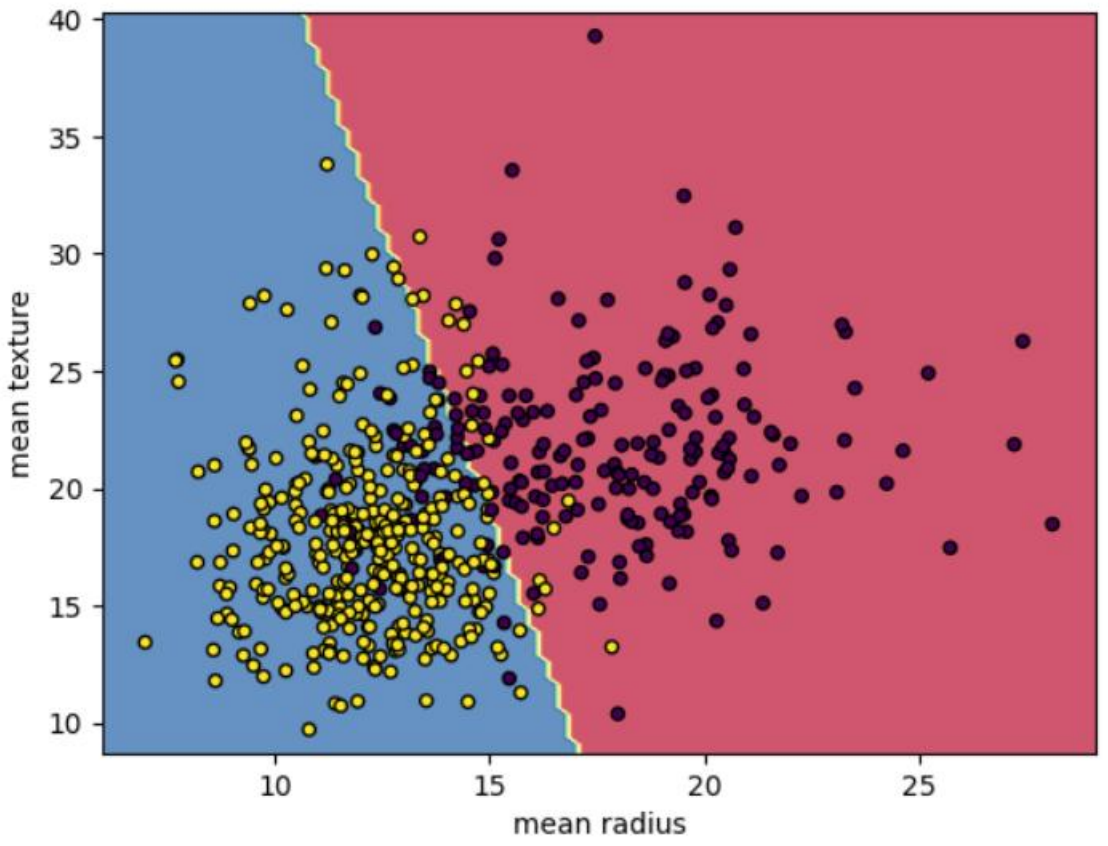
Code Explanation Example 2

- To perform regression, the values are split into training and test variables.
- The training variables are trained in SVM model.
- Prediction is done with the test data and accuracy score is calculated with test data and predicted data.
- The result is visualised with the help of DecisionBoundaryDisplay and Scatter plot.

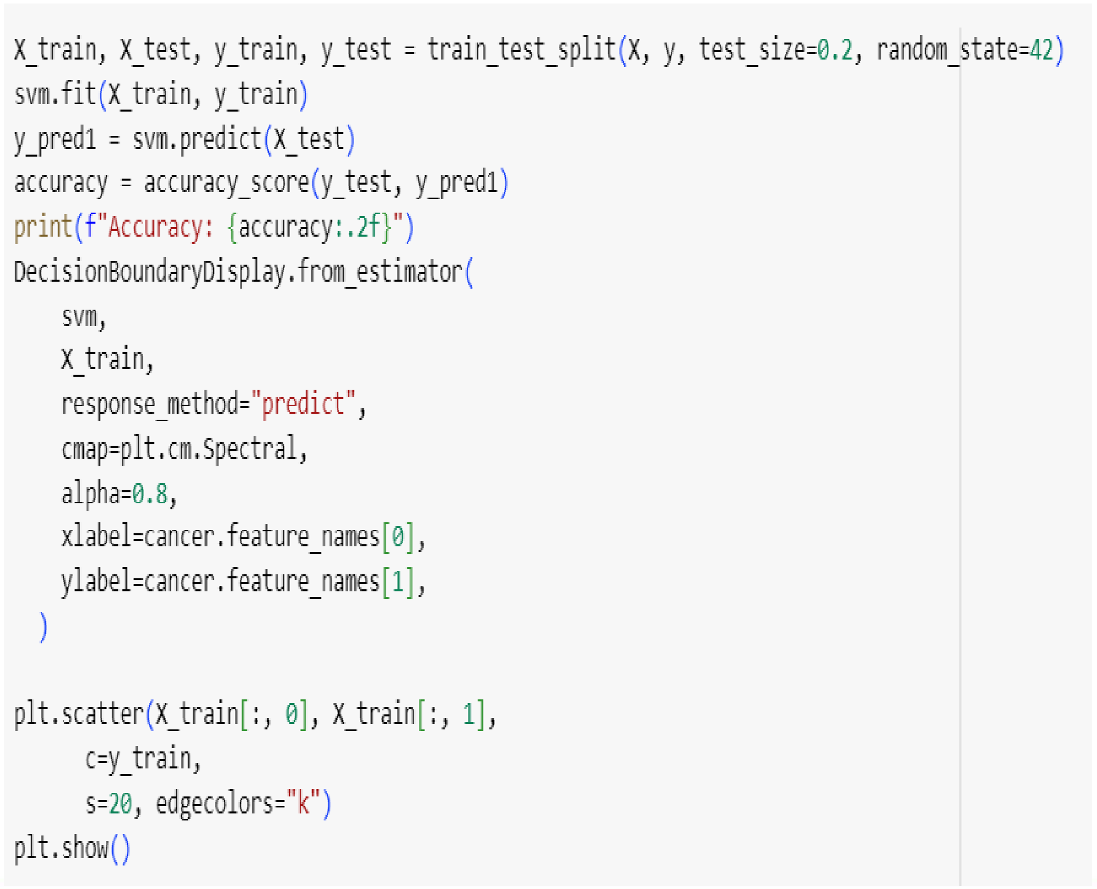
Code and Output Example 2

```python
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
import pandas as pd
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer()
data = pd.DataFrame(cancer.data, columns=cancer.feature_names)
X = cancer.data[:, :2]
y = cancer.target
svm = SVC(kernel="linear")
svm.fit(X, y)
y_pred = svm.predict(X)
accuracy = accuracy_score(y, y_pred)
print(f"Accuracy: {accuracy:.2f}")
DecisionBoundaryDisplay.from_estimator(
    svm,
    X,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8,
    xlabel=cancer.feature_names[0],
    ylabel=cancer.feature_names[1],
  )
plt.scatter(X[:, 0], X[:, 1],
      c=y,
      s=20, edgecolors="k")
plt.show()
```

```
Accuracy: 0.89
```



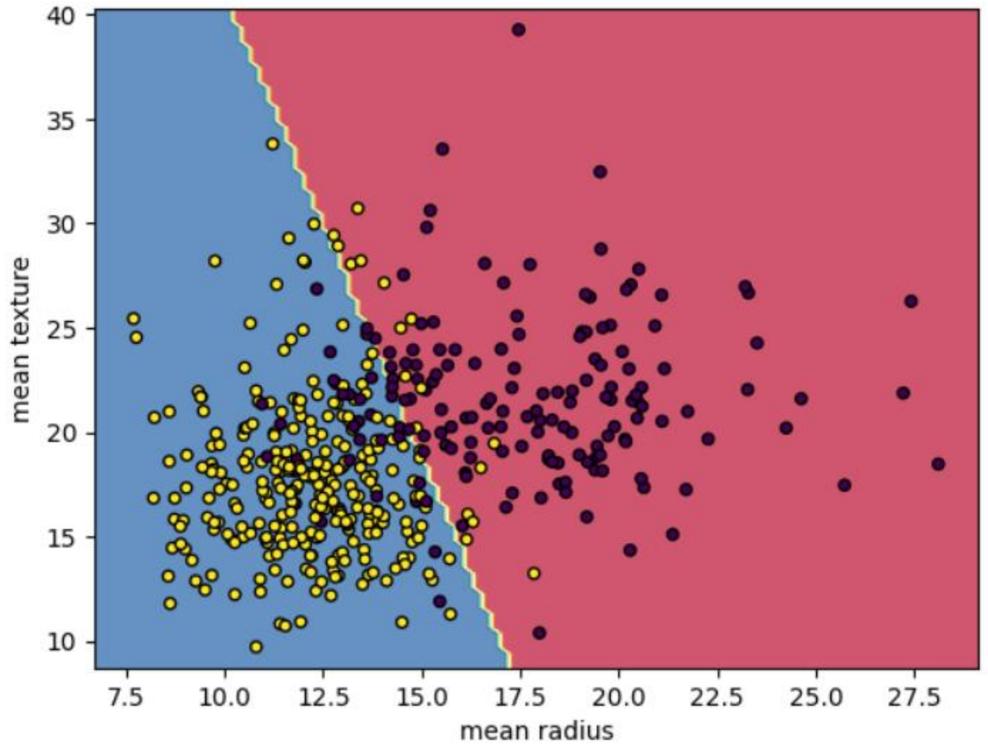
Code and Output Example 2



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm.fit(X_train, y_train)
y_pred1 = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred1)
print(f"Accuracy: {accuracy:.2f}")
DecisionBoundaryDisplay.from_estimator(
    svm,
    X_train,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8,
    xlabel=cancer.feature_names[0],
    ylabel=cancer.feature_names[1],
  )

plt.scatter(X_train[:, 0], X_train[:, 1],
      c=y_train,
      s=20, edgecolors="k")
plt.show()
```

```
Accuracy: 0.90
```



For Further Reference

Learning Support Vector Machines

[Source: https://www.youtube.com/watch?v=_PwhiWxHK8o&t=2551s]

Reference:

[1] "SSVM: a simple SVM algorithm," ieeexplore.ieee.org. https://ieeexplore.ieee.org/abstract/document/1007516/ (accessed Oct. 20, 2023).

[2] V. Jakkula, "Tutorial on Support Vector Machine (SVM)." Available: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf

[3] aswathisasidharan, "Support Vector Machine Algorithm," GeeksforGeeks, Jan. 20, 2021. https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[4] G. Bedi, "Simple guide to Text Classification(NLP) using SVM and Naive Bayes with Python," *Medium*, Nov. 09, 2018. https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34