**EXERCISE-7**

**AIM: IMPLEMENT SUPPORT VECTOR MACHINE FOR A SAMPLE TRAINING DATA SET STORED AS A ‘.CSV’ FILE. COMPUTE THE ACCURACY OF THE CLASSIFIER, CONSIDERING FEW TEST DATA SETS.**

**DESCRIPTION:**

**A support vector machine (SVM)** is a machine learning algorithm that uses **supervised learning models** to solve complex **classification, regression, and outlier detection problems** by performing **optimal data transformations** that determine **boundaries between data points** based on **predefined classes, labels, or outputs.**

SVMs are widely adopted across disciplines such as **healthcare, natural language processing, signal processing applications, and speech & image recognition fields.**

Technically, the **primary objective of the SVM algorithm** is to **identify a hyperplane that distinguishably segregates the data points of different classes**. The hyperplane is localized in such a manner that the **largest margin separates the classes under consideration.**

**Types of Support Vector Machine (SVM) Algorithms:**

1. **Linear SVM:** When the data is **perfectly linearly separable only** then we can **use Linear SVM.** Perfectly linearly separable means that the ***data points can be classified into 2 classes by using a single straight line(if 2D).***
2. **Non-Linear SVM:** When the data is not linearly separable then **we can use Non-Linear SVM**, which means when the **data points cannot be separated into 2 classes by using a straight line (if 2D)** then we use some advanced techniques **like kernel tricks to classify them**. In most real-world applications we do not find linearly separable Datapoints hence we **use kernel trick to solve them.**

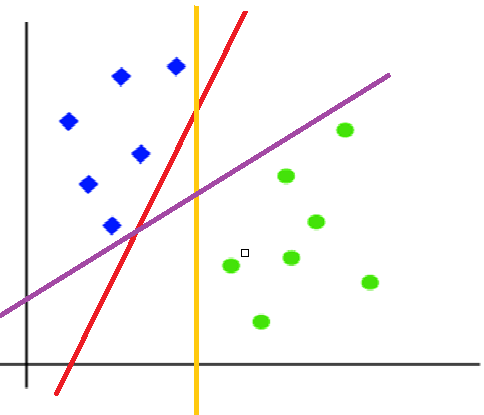
**Important Terms:**

1. **Support Vectors:** These are the points that are **closest to the hyperplane.** A separating line will be defined with the help of these data points.
2. **Margin:** it is the **distance between the hyperplane and the observations** closest to the hyperplane (support vectors). In SVM large margin is **considered a good margin.** There are two types of margins **hard margin and soft margin**.



**How does Support Vector Machine Works?** The working of SVM using an example. Suppose we have a dataset that has two classes (green and blue). We want to classify that the new data point as either blue or green.

To **classify these points,** we can have **many decision boundaries,** but the question is which is the best and how do we find it?

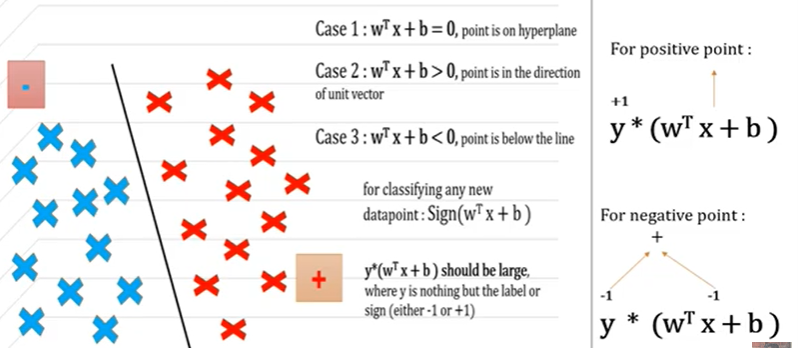
 **NOTE:** Since we are plotting the data points in a 2-dimensional graph we call this decision **boundary a straight line** but if we have more dimensions, we call this decision boundary **a “hyperplane”.**

The **best hyperplane** is that plane that has the **maximum distance from both the classes**, and this is the main aim of SVM.

This is done by finding different hyperplanes which classify the labels in the best way then it will choose the **one which is farthest from the data poi**nts or the one **which has a maximum margin.**



**Mathematical Representation of Linear SVM:**



**Implementation of the Support Vector Machine involves below steps:**

1. **Install the Packages:**
2. **Numpy:** Numpy Python library is used for including any type **of mathematical operation in the code**. It is the fundamental package for scientific calculation in Python. It also supports to **add large, multidimensional arrays and matrices**. **So, in Python, we can import it as:**

**import numpy as np**

**(b) Matplotlib:** The second library is matplotlib, which is a **Python 2D plotting library**, and with this library, we need to import a sub-library pyplot. This library is **used to plot any type of charts in Python** for the code. **It will be imported as below:**

**import matplotlib.pyplot as plt**

Here we have used **plt a**s a short name for this library.

**c) Pandas:** The last library is the Pandas library, which is one of the most famous Python libraries and used for **importing and managing the datasets**. It is an **open-source data manipulation and analysis library**. It will be imported as below:

**import pandas as pd**

Here, we have used **pd** as a short name for this library.

1. **Importing the Dataset:**

**read\_csv() function:** Now to import the dataset, we will use **read\_csv() function** of **pandas library**, which is used to read a csv file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL. **We can use read\_csv function as below:**

**For Eg: dataset = pd.read\_csv('Social\_Network\_Ads.csv')**

1. **Separating Independent and Dependent Variables:**

In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

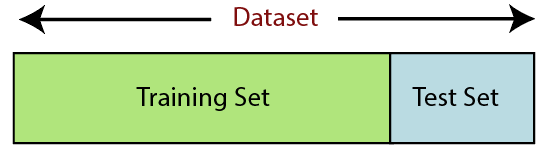
**For Eg:** In our dataset, there are **Two independent variables** that are  **Age** and **Salary**, and **one is a dependent variable** which is **purchased**.

**# Seperating Independent and Dependent Variable**

**X = dataset.iloc[:, :-1].values**

**y = dataset.iloc[:, -1].values**

**4. Splitting dataset into training and test set:** we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.



**Training Set:** A subset of dataset to train the machine learning model, and we already know the output.

**Test set:** A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**For splitting the dataset, we will use the below lines of code:**

**# training and testing data**

**from sklearn.model\_selection import train\_test\_split**

**# assign test data size 25%**

**X\_train, X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size= 0.25, random\_state=0)**

**Explanation:** We set test\_size=0.25, which means **25%** of the whole data set will be assigned to the **testing** part, and the remaining **75%** will be used for the model’s **training**.

**5. Feature Scaling:** Feature scaling is the final step of data pre-processing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put in the same range and in the same scale so that no any variable dominate the other variable.

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.transform(X\_test)**

### 6.Training model using SVM Classifier : Now, let’s train our model using the Linear SVM Classifier

**# import SVM classifier from SK Learn**

**from sklearn.svm import SVC**

**# create a Linear SVM Classifier**

**classifier = SVC(kernel = 'linear', random\_state = 0)**

**# training the model**

**classifier.fit(X\_train, y\_train)**

**7.Test the model using SVM Classifier :**

**y\_pred = classifier.predict(X\_test)**

**print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))**

**8. Find the Accuracy of the Model:** Accuracy score in machine learning is an **evaluation metric** that **measures the number of correct predictions made by a model** in relation to **the total number of predictions made.** We calculate it by dividing the number of correct predictions by the total number of predictions.



**# importing accuracy score**

**from sklearn.metrics import accuracy\_score**

**# printing the accuracy of the model**

**print(accuracy\_score(y\_test,y\_pred1))**

**Print the Confusion Matrix and Accuracy Score:** The **confusion matrix** is one of the most popular and widely used performance measurement techniques for **classification models.**

**Confusion Matrix** as the name suggests gives us a **matrix as output and describes the complete performance of the model** and it also used to **determine the performance of the classification models for a given set of test data.**

**from sklearn.metrics import confusion\_matrix, accuracy\_score**

**# passing actual and predicted values**

**cm = confusion\_matrix(y\_test, y\_pred)**

**print(cm)**

****

**# Print the Accuracy Score**

**accuracy\_score(y\_test, y\_pred)**

### 9. Visualizing the Clusters:

The last step is to visualize the clusters. To visualize the clusters will use scatter plot using matplotlib. Here we can use the **Colormap object generated** from a li**st of colors.** This may be most useful when indexing directly into a **colormap**, but it can also be used to generate special colormaps for ordinary mapping.

**Parameters:** colorslist, array. Sequence of Matplotlib color specifications (color names or RGB(A) values).

**from matplotlib.colors import ListedColormap**

**X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train**

You use **meshgrid()** to convert the 1D vectors representing the axes into 2D arrays. You can then use those arrays in place of the x and y variables in the mathematical equation.

**X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10, step = 0.25),**

**np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))**

The **contourf() function** in pyplot module of matplotlib library is used to **plot contours**. But contourf **draw filled contours**, while contourf draws **contour lines.**

**plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),**

**alpha = 0.75, cmap = ListedColormap(('red', 'green')))**

**plt.xlim(X1.min(), X1.max())**

**plt.ylim(X2.min(), X2.max())**

**PROGRAM:**

**# importing the libraries**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

**# importing the dataset**

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

**# Seperating Independent and Dependent Variable**

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

**# Splitting the Dataset into training and testing**

from sklearn.model\_selection import train\_test\_split

**# assign test data size 25%**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

**# Feature Scaling**

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

**# Training the SVM Model on the Training Set**

from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

**# Predicting the New Result**

print(classifier.predict(sc.transform([[30,87000]])))

**# Predicting the Test Result**

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

**# Making the Confusion Matrix**

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

## #Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10, step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

## #Visualising the Test set results

## from matplotlib.colors import ListedColormap

## X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test

## X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10, step = 0.25),

## np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step = 0.25))

## plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),

## alpha = 0.75, cmap = ListedColormap(('red', 'green')))

## plt.xlim(X1.min(), X1.max())

## plt.ylim(X2.min(), X2.max())

## for i, j in enumerate(np.unique(y\_set)):

## plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)

## plt.title('SVM (Test set)')

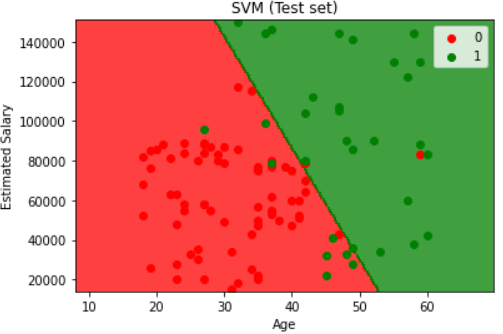
## plt.xlabel('Age')

## plt.ylabel('Estimated Salary')

## plt.legend()

## plt.show()

**INPUT / OUTPUT:**

****

**CONCLUSION: Program is executed successfully without any error.**