# Experiment-2.3

**Student Name: Ashish Kumar** **UID: 23MAI10008**
**Branch: ME CSE AIML** **Section/Group: 23MAI-1**
**Semester: 02** **Date of Performance: 13/03/2024**
**Subject Name: Machine Learning Lab** **Subject Code: 23CSH-651**

**Aim of the Experiment :**

Implementing Support Vector Machine using Python.

**Theory :**

**Support Vector Machine** (SVM) is a supervised machine learning algorithm used for both classification and regression. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

**Types of SVM:**

**1) Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

**2) Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**Support Vector Machine Terminology:**

**1) Hyperplane:** Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.

**2) Support Vectors:** Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.

**3) Margin:** Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin. The wider margin indicates better classification performance.

**4) Kernel:** Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space.Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.

**Code for Experiment :**

```
# Import the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Load the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
dataset

# Split Dataset into X and Y
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Split the X and Y Dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
# Perform Feature Scaling as all values are not in the same range
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#  Fit SVM to the Training set
# 'rbf' is nonlinear and gives better results as compared to linear.
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)

# Predict the Test Set Results
y_pred = classifier.predict(X_test)
print("Actual values:")
print(y_test)
print("Predicted values:")
print(y_pred)

# Make the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
print("\nResults of SVM Model:")
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix: ")
print(cm)
print("Accuracy Score: ",accuracy_score(y_test,y_pred))

# Visualise the Test set results
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
```

```
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(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],color = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

**Result/Output :**

jupyter ML_Experiment7_ASHISH_23MAI10008 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

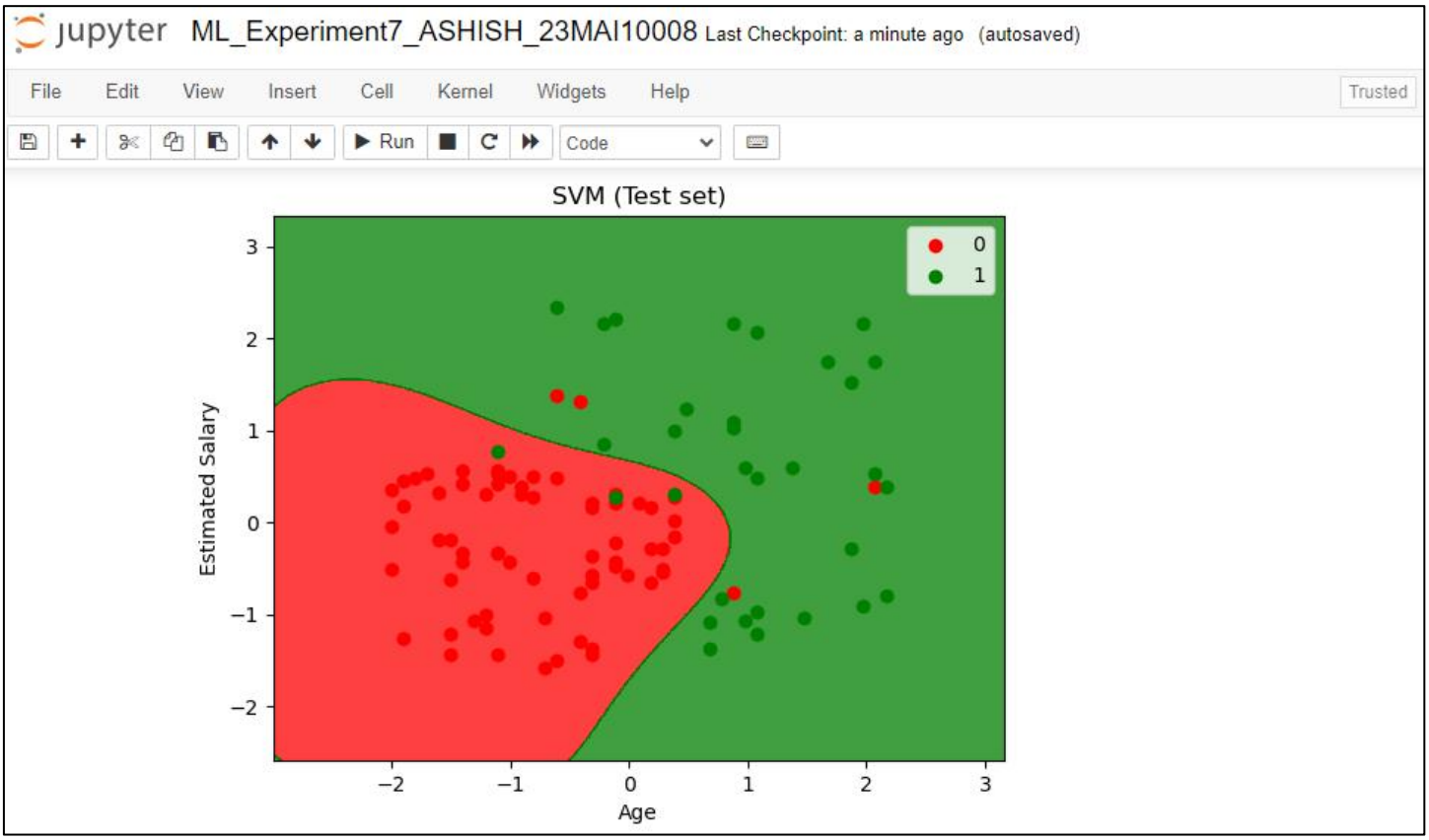
Out[1]:

| | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| 0 | 15624510 | Male | 19 | 19000 | 0 |
| 1 | 15810944 | Male | 35 | 20000 | 0 |
| 2 | 15668575 | Female | 26 | 43000 | 0 |
| 3 | 15603246 | Female | 27 | 57000 | 0 |
| 4 | 15804002 | Male | 19 | 76000 | 0 |
| ... | ... | ... | ... | ... | ... |
| 395 | 15691863 | Female | 46 | 41000 | 1 |
| 396 | 15706071 | Male | 51 | 23000 | 1 |
| 397 | 15654296 | Female | 50 | 20000 | 1 |
| 398 | 15755018 | Male | 36 | 33000 | 0 |
| 399 | 15594041 | Female | 49 | 36000 | 1 |

400 rows × 5 columns

```
Actual values:
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
Predicted values:
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 1 1 1]

Results of SVM Model:
Confusion Matrix:
[[64  4]
 [ 3 29]]
Accuracy Score:  0.93
```




**Learning outcomes (What I have learnt):**

1. I learnt about various python libraries like pandas, numpy and sklearn.
2. I learnt about the concept of Support Vector Machine Algorithm.
3. I learnt about Hyperplane, Support vectors and Margin in SVM algorithm.
4. I learnt about the linear and non linear SVM algorithm.
5. I learnt about different types of kernels used in SVM Algorithm.