Terna Engineering College

Department of Artificial Intelligence and Data Science

Program : Sem VI

Course: Machine Learning Lab

Experiment No.06

PART A

(PART A: TO BE REFERRED BY STUDENTS)

A.1 Aim: To implement Support Vector Machines using Python.

A.2 Theory:

Support Vector Machines:

Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

Support Vectors

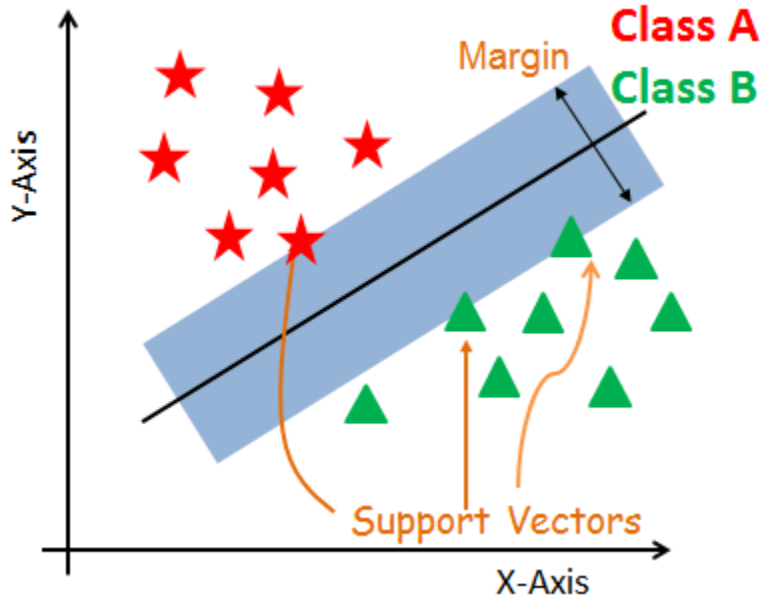
Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.

Hyperplane

A hyperplane is a decision plane which separates between a set of objects having different class memberships.

Margin

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points. If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.



e.g. The geo-sounding problem is one of the widespread use cases for SVMs, wherein the process is employed to track the planet's layered structure. This entails solving the inversion problems where the observations or results of the issues are used to factor in the variables or parameters that produced them.

In the process, linear function and support vector algorithmic models separate the electromagnetic data. Moreover, linear programming practices are employed while developing the supervised models in this case. As the problem size is considerably small, the dimension size is inevitably tiny, which accounts for mapping the planet's structure.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll. No. A12	Name: Sufiyan Khan
Class: TE – AI & DS	Batch: A1
Date of Experiment: 11-03-24	Date of Submission: 13-03-24
Grade:	

B.1 Input and Output:

```
Click here to ask Blackbox to help you code faster
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

2] ✓ 4.5s

+ Code + Markdown

```
Click here to ask Blackbox to help you code faster
print("Features: ", cancer.feature_names)
print("Labels: ", cancer.target_names)
```

3] ✓ 0.0s

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

```
Click here to ask Blackbox to help you code faster
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_state=109)
```

✓ 0.3s

```
Click here to ask Blackbox to help you code faster
from sklearn import svm

clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

✓ 1.3s

```
Click here to ask Blackbox to help you code faster
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

✓ 0.0s

```
Accuracy: 0.9649122807017544
```

B.2 Conclusion:

Thus we have successfully implemented Support Vector Machines in Python and understood how the model is fitted to the given data.