Aim: To obtain thesupport vectors for classification of the data points using support vector machine technique

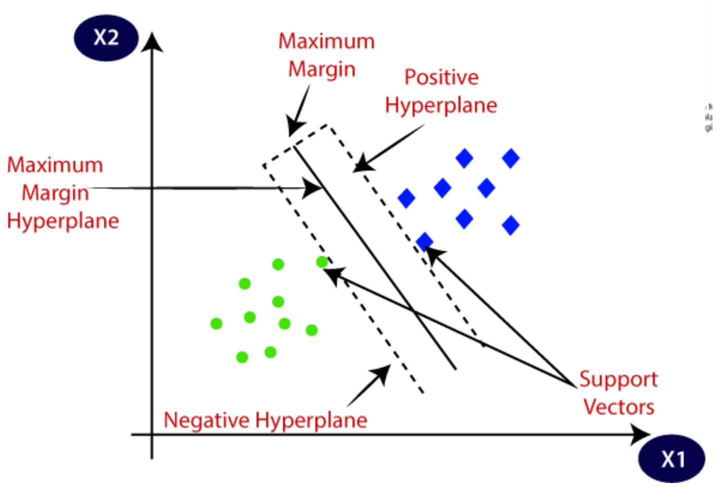
IDE: Google Colab

Theory:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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.

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. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **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.

- **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.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Methodology:

1. Load the basic libraries and packages
2. Load the dataset
3. Analyse the dataset
4. Normalize the data
5. Pre-process the data
6. Load Different Support Vector classifiers
7. Train the classifier
8. Predict over the test dataset
9. Obtain the confusion matrix for each of the classifier output

Program (Code):

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm, datasets
```

```
from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC


# import some data to play with

iris = datasets.load_iris()

X = iris.data[:, :2]

y = iris.target


y


C = 1  # SVM regularization parameter


svc = svm.SVC(kernel='linear', C=C).fit(X, y)

rbf_svc = svm.SVC(kernel='rbf', gamma=0.5, C=C).fit(X, y)

poly_svc = svm.SVC(kernel='poly', degree=5, C=C).fit(X, y)

lin_svc = svm.LinearSVC(C=C).fit(X, y)


y_pred = rbf_svc.predict(X)


print("RBF", accuracy_score(y,y_pred))


y_pred = lin_svc.predict(X)

print("linear", accuracy_score(y,y_pred))
```

```
y_pred=poly_svc.predict(X)
print("Polynomial",accuracy_score(y,y_pred))

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score

y_pred = rbf_svc.predict(X)
print("RBF", accuracy_score(y,y_pred))

print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
print(accuracy_score(y,y_pred))

y_pred = lin_svc.predict(X)
print(confusion_matrix(y, y_pred))
print("Linear")
print(classification_report(y, y_pred))

y_pred = poly_svc.predict(X)
print(confusion_matrix(y, y_pred))
print("Polynomial")
print(classification_report(y, y_pred))

# create a mesh to plot in
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
```

```
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

# title for the plots
titles = ['SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel (Gamma=0.5)',
          'SVC with polynomial (degree 5) kernel']


for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
```

```
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```

Results:

To be attached with

a. Correlation graph between various features

```
RBF 0.8266666666666667
linear 0.8
Polynomial 0.7933333333333333
```

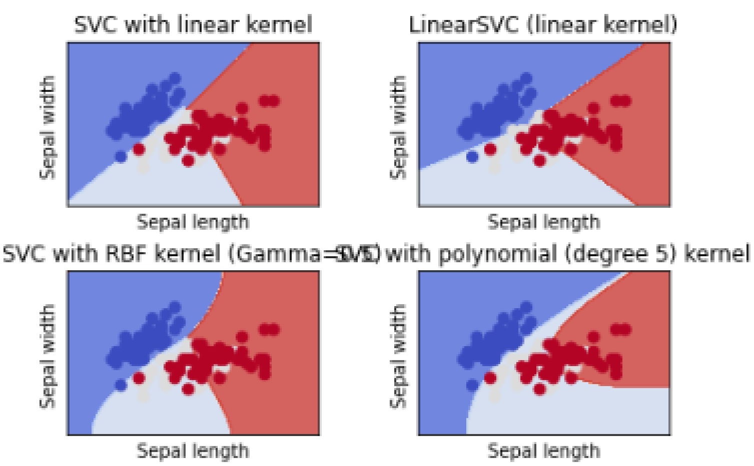
b. Confusion matrix and classification report for (i) RBF, (ii) Linear and (iii) Polynomial

```
RBF 0.8266666666666667
[[50  0  0]
 [ 0 38 12]
 [ 0 14 36]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.73      0.76      0.75        50
           2       0.75      0.72      0.73        50

    accuracy                           0.83       150
   macro avg       0.83      0.83      0.83       150
weighted avg       0.83      0.83      0.83       150

0.8266666666666667
```

```
[[49  1  0]
 [ 2 30 18]
 [ 0  9 41]]
Linear
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        50
           1       0.75      0.60      0.67        50
           2       0.69      0.82      0.75        50

    accuracy                           0.80       150
   macro avg       0.80      0.80      0.80       150
weighted avg       0.80      0.80      0.80       150
```

```
[[50  0  0]
 [ 0 38 12]
 [ 0 19 31]]
Polynomial
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.67      0.76      0.71        50
           2       0.72      0.62      0.67        50

    accuracy                           0.79       150
   macro avg       0.80      0.79      0.79       150
weighted avg       0.80      0.79      0.79       150
```

c. Contour graph for different support vector classifiers



Observation and Result Analysis:

a. Nature of the dataset

b. During Training Process

c. After the training Process

Post Lab Exercise:

a. What is the geometric intuition behind SVM?

b. What do know about Hard Margin SVM and Soft Margin SVM?

c. What's the "kernel trick" and how is it useful?

d. What is the role of C in SVM? How does it affect the bias/variance trade-off?

e. SVM being a large margin classifier, is it influenced by outliers?

Post Lab Activity:

Consider any dataset from **https://archive.ics.uci.edu/ml/datasets.php** and obtain the hyperplane for the classification of various classes. Make sure that the dataset is not matching with your classmates. You can also select the dataset from other ML repositories with prior permission from your concerned subject faculty.