Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL604	**Course Name:**	Machine Learning Lab

Name of Student:	Parth Manoj Raut
Roll No.:	44
Experiment No.:	4
Title of the Experiment:	Implementation of Support Vector Machine
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr Raunak Joshi

Signature :

Date :

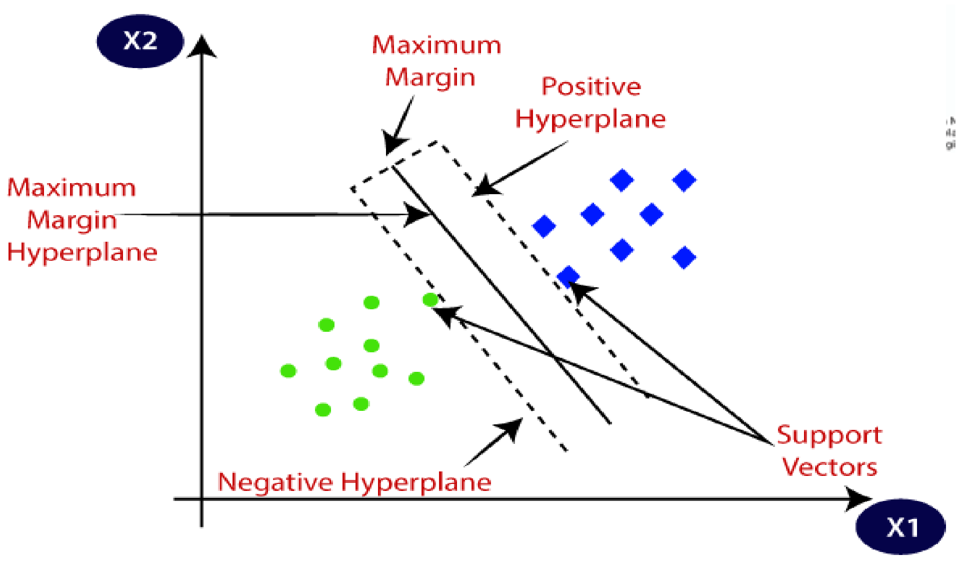
Aim: Implementation of Support Vector Machine Algorithm.

Objective: Ablility to perform various feature engineering tasks, apply Support Vector Machine and create the Confusion Matrix.

Theory:

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:



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.

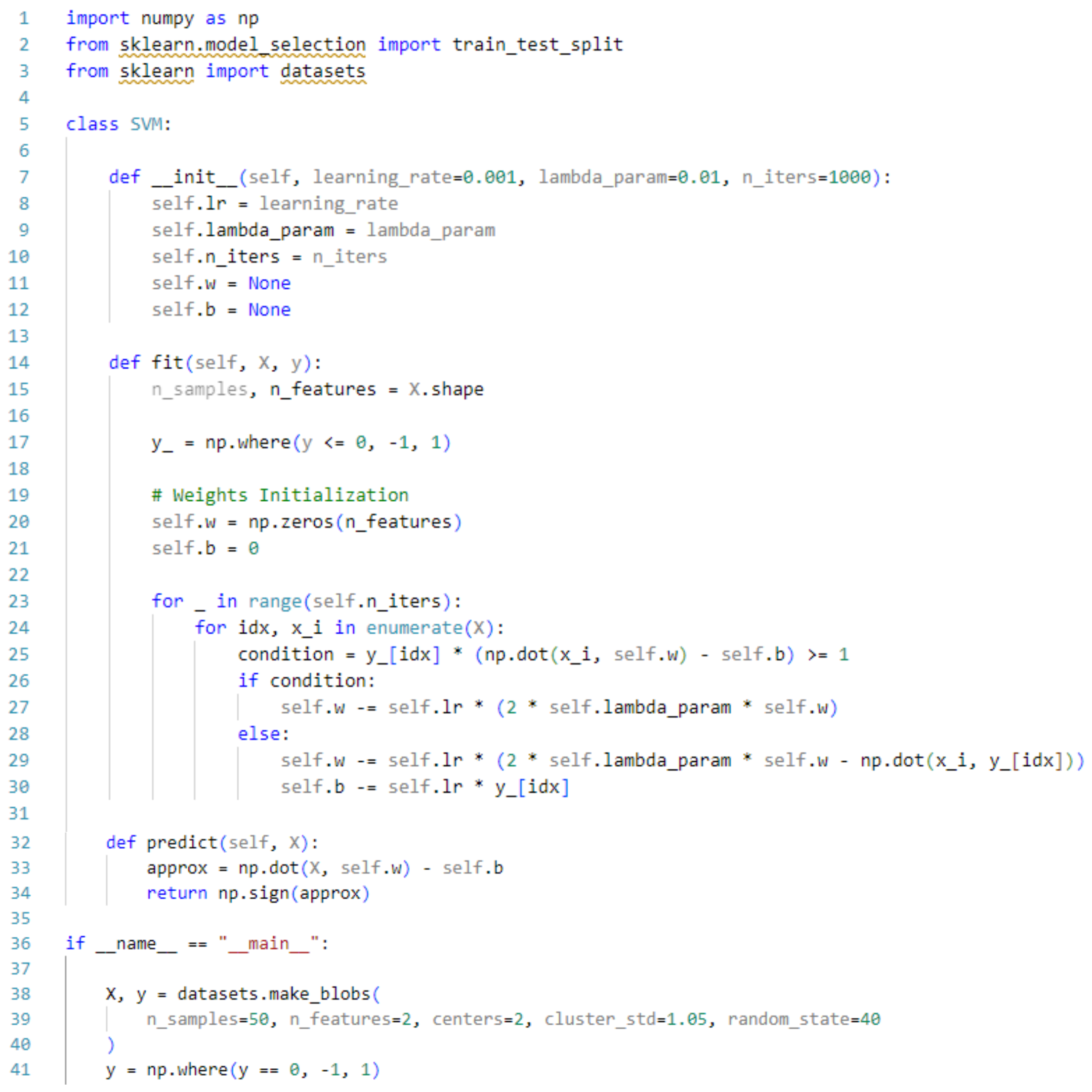
Support Vectors:

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.

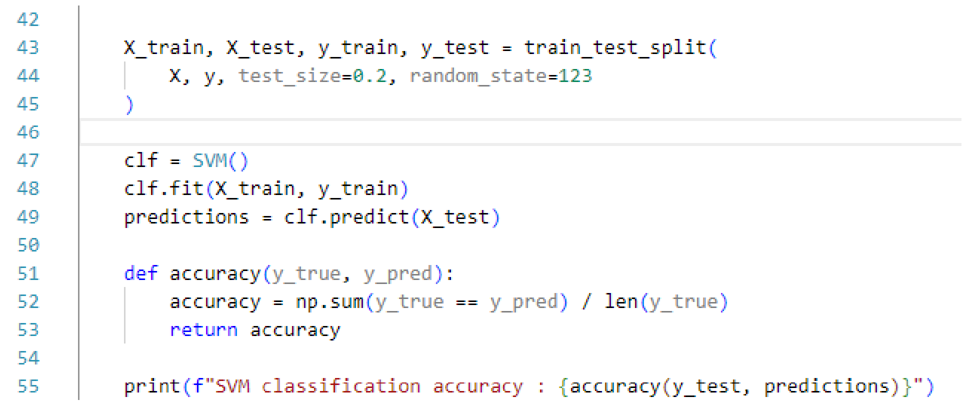
Implementation of Support Vector Machine

1. Data Pre-processing step
2. Fitting the SVM classifier to the training set
3. Predicting the test set result
4. Creating the confusion matrix
5. Visualizing the training set result
6. Visualizing the test set result

Implementation:

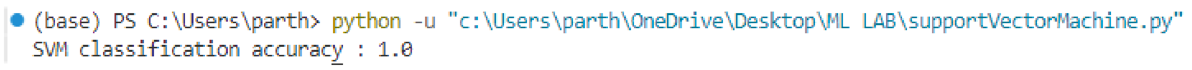


```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets

class SVM:

    def __init__(self, learning_rate=0.001, lambda_param=0.01, n_iters=1000):
        self.lr = learning_rate
        self.lambda_param = lambda_param
        self.n_iters = n_iters
        self.w = None
        self.b = None

    def fit(self, X, y):
        n_samples, n_features = X.shape

        y_ = np.where(y <= 0, -1, 1)

        # Weights Initialization
        self.w = np.zeros(n_features)
        self.b = 0

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1
                if condition:
                    self.w -= self.lr * (2 * self.lambda_param * self.w)
                else:
                    self.w -= self.lr * (2 * self.lambda_param * self.w - np.dot(x_i, y_[idx]))
                    self.b -= self.lr * y_[idx]

    def predict(self, X):
        approx = np.dot(X, self.w) - self.b
        return np.sign(approx)

if __name__ == "__main__":

    X, y = datasets.make_blobs(
        n_samples=50, n_features=2, centers=2, cluster_std=1.05, random_state=40
    )
    y = np.where(y == 0, -1, 1)
```



```python

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=123
)

clf = SVM()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)

def accuracy(y_true, y_pred):
    accuracy = np.sum(y_true == y_pred) / len(y_true)
    return accuracy

print(f"SVM classification accuracy : {accuracy(y_test, predictions)}")
```

Output:



```
● (base) PS C:\Users\parth> python -u "c:\Users\parth\OneDrive\Desktop\ML LAB\supportVectorMachine.py"
  SVM classification accuracy : 1.0
```

Conclusion:

The SVM implementation has high classification accuracy, attaining 100% prediction on test data. Its capacity to properly distinguish binary classes shows that it can handle linearly separable datasets. However, more testing on different datasets and hyperparameter adjustment could improve its generalization and performance in a variety of settings.