Experiment No:6

Course Outcome: CO2 **Blooms Level: L3**

Aim: To implement Linear Support Vector Machine

Abstract:

Support Vector Machine (SVM) is a powerful machine learning algorithm used for classification tasks, demonstrated through the Iris and Breast Cancer datasets. In the Iris dataset, SVM classifies different species of iris flowers based on sepal and petal measurements, achieving high accuracy in multi-class classification. For the Breast Cancer dataset, SVM differentiates between malignant and benign tumors, showcasing its strength in binary classification. By using the kernel trick, SVM handles non-linearly separable data, making it effective for both datasets. Despite computational complexity in larger datasets, SVM's strong generalization ability and robust performance make it an ideal choice for classification in domains like botany and medical diagnostics.

Sample Input and Output:

Case 1:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa

Sample Output

```
Confusion Matrix:
 [[17  0]
 [ 0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        13

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Theory:

Support Vector Machine (SVM):

SVM is a supervised machine learning algorithm used for classification and regression tasks.

It finds the optimal hyperplane that best separates data into classes.

SVM aims to maximize the margin between data points of different classes while minimizing classification errors. 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.

Preprocessing Data:

Data preprocessing includes handling missing values, scaling, and preparing data for classification.

Implementing SVM:

Libraries like scikit-learn are commonly used for implementing SVM.

The SVM model is trained on data to create a decision boundary that separates classes.

Model Evaluation:

Metrics like accuracy, precision, recall, and F1-score are used to evaluate the performance of the SVM model.

Visualizations of decision boundaries provide insights into class separation.

Program:

Dataset Information

The Iris dataset consists of 150 samples of iris flowers, 3 classes (Setosa, Versicolor, and Virginica) of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2. The latter are not linearly seperable from each other.

Attribute information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm.
5. class:-Iris Setosa--Iris Versicolour—Iris Virginica

Iris Versicolor   Iris Setosa   Iris Virginica

Output:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset (Iris)
iris = datasets.load_iris()
X = iris.data[:, :2]   # take only first 2 features for easy visualization
y = iris.target

# For binary classification, use only classes 0 and 1
X = X[y != 2]
y = y[y != 2]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create linear SVM
svm_clf = SVC(kernel="linear", C=1.0)
svm_clf.fit(X_train, y_train)

# Predictions
y_pred = svm_clf.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```python

# Plot decision boundary
def plot_svm(clf, X, y):
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k')

    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Create grid
    xx = np.linspace(xlim[0], xlim[1], 30)
    yy = np.linspace(ylim[0], ylim[1], 30)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    # Plot decision boundary & margins
    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], linestyles=['--', '-', '--'])

    # Plot support vectors
    ax.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
               s=100, facecolors='none', edgecolors='k')
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.title("Linear SVM Decision Boundary")
    plt.show()

plot_svm(svm_clf, X, y)
```

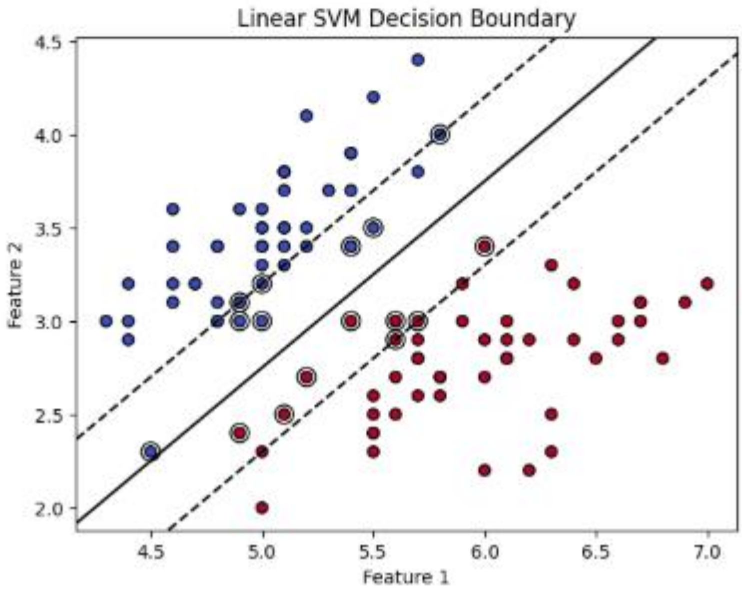
Output

```
Confusion Matrix:
 [[17  0]
 [ 0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        13

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



Conclusion: Implementing the Support Vector Machine algorithm using Python allows for the creation of powerful classifiers. By preprocessing data, training the model, and evaluating its performance, you can gain practical experience in building SVM models for classification tasks, such as image recognition, medical diagnosis, and more.

Exercise 1:

Apply a Linear Support Vector Machine (SVM) to the Breast Cancer dataset to classify tumors as malignant or benign based on features such as cell radius, texture, and smoothness. The problem involves training a linear SVM model to identify the optimal hyperplane that separates the two classes while maximizing the margin between them. The objective is to achieve high classification accuracy and evaluate the model's performance in distinguishing between benign and malignant tumors.

Students shall draw flowchart of exercise question in the writeup and submit.

Exercise 2:

Generate and side-by-side bar chart comparing accuracy, precision, recall, and F1-score between Linear vs RBF SVM and also do cross-validation comparison using Breast cancer dataset.

Students shall draw flowchart of exercise question i n the writeup and submit.