EXPERIMENT 7

Student Name: Samuel	**UID: 24MAI10018**
Branch: CSE-AIML	**Section/Group: 24MAI-1**
Semester: 2	**Date of Performance:**
Subject Name: Machine Learning Lab	**Subject Code: 24CSH–651**

AIM:

Implementing Support Vector Machine using Python.

SOFTWARE REQUIREMENTS:

- Python IDE (e.g., Jupyter Notebook, PyCharm, etc.)
- NumPy Library.
- Pandas Library.
- Scikit-Learn Library.
- Matplotlib & Seaborn Libraries.

THEORY:

Support Vector Machine (SVM) Algorithm: Support Vector Machine (SVM) is a supervised learning algorithm used for both classification and regression tasks. It finds the optimal hyperplane that best separates different classes in a high-dimensional space.

Key Features of SVM:

- **Maximizes Margin**: Finds the hyperplane that maximizes the margin between classes.
- **Works with High-Dimensional Data**: Effective even in cases where the number of features is greater than the number of samples.
- **Kernel Trick**: Can be used for non-linearly separable data by mapping it to a higher-dimensional space.

Mathematically: For a dataset with features x_i and labels y_i, the SVM classifier optimizes:

$$\frac{}{1} \quad = 2\| \|^2$$

subject to:

$$y_i\,(w.\ x_i + b) \geq 1,\ \forall i,$$

where w is the weight vector and b is the bias term.

Types of SVM Kernels:

- **Linear Kernel**: Used when data is linearly separable.
- **Polynomial Kernel**: Maps data to a higher degree polynomial space.
- **Radial Basis Function (RBF) Kernel**: Useful for non-linear classification.
- **Sigmoid Kernel**: Works similarly to an artificial neural network activation function.

Advantages of SVM:

- Effective in high-dimensional spaces.
- Works well with small datasets.
- Robust to overfitting when using appropriate regularization.

Disadvantages:

- Computationally expensive for large datasets.
- Sensitive to parameter tuning (C and gamma).

ALGORITHM:

1. Load and preprocess the dataset.
2. Choose the SVM kernel (Linear, RBF, Polynomial, etc.).
3. Train the SVM model on the training data.
4. Predict the labels for the test data.
5. Evaluate the model using accuracy, precision, and recall.
6. Visualize the decision boundary (for 2D datasets).

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import
train_test_split from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

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

# Split data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
      random_state=42)

# Initialize SVM Classifier with RBF kernel
svm_classifier = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)

# Train the model
svm_classifier.fit(X_train, y_train)

# Make predictions
y_pred = svm_classifier.predict(X_test)

# Compute accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Display Classification Report
print("\nClassification Report:")
print(classification_report(y_test,
y_pred, target_names=iris.target_names))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Visualizing Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d",
      xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix -
SVM") plt.show()
```

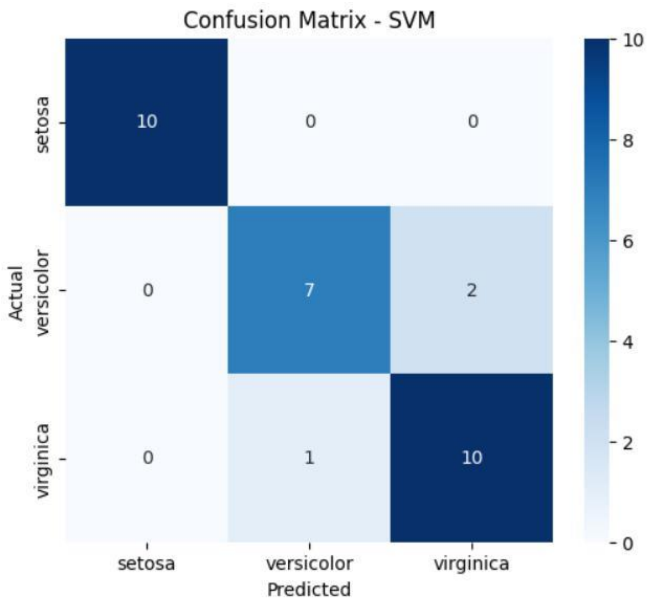
OUTPUT:

```
Model Accuracy: 90.00%

 Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.88      0.78      0.82         9
   virginica       0.83      0.91      0.87        11

    accuracy                           0.90        30
   macro avg       0.90      0.90      0.90        30
weighted avg       0.90      0.90      0.90        30
```



LEARNING OUTCOMES:

1. Understood the Support Vector Machine Algorithm and its mathematical formulation.
2. Learned how kernels transform data for linear and non-linear classification.
3. Implemented SVM Classifier using Scikit-Learn.
4. Evaluated model performance using accuracy, classification report, and confusion matrix.
5. Understood the importance of hyperparameter tuning (C, gamma) for better results.