Practical No. 8

Submitted By:		
Name: Mahak Mor	**Roll No.:** 20	**Section:** A

Aim: To implement Support Vector Machine (SVM) Classifier.

Theory: Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. In classification, SVM aims to find the hyperplane that best separates different classes of data points with the maximum margin. The data points that lie closest to the hyperplane are called support vectors, and they play a critical role in determining the decision boundary.

Key Concepts:

- Hyperplane: A decision boundary that separates different classes in the feature space. In a 2D space, it's a line, in a 3D space, it's a plane, and so on.
- Support Vectors: The data points closest to the hyperplane, which influence the position and orientation of the hyperplane.
- Margin: The distance between the hyperplane and the closest support vectors. SVM aims to maximize this margin.
- Linear SVM: Used when data is linearly separable, i.e., can be separated by a straight line (or plane).
- Non-linear SVM: When data is not linearly separable, a kernel trick is used to map the data into a higher-dimensional space where it becomes separable.

Advantages:

- Effective in high-dimensional spaces.
- Works well when there is a clear margin of separation between classes.
- Can handle both linear and non-linear classification problems using kernels.

Disadvantages:

- Can be inefficient in large datasets with a high number of features.
- Requires careful tuning of parameters like C and the kernel type.

Code and Output:

```
# Importing necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

import seaborn as sns


# Loading the Iris dataset

from sklearn.datasets import load_iris

iris = load_iris()

X = iris.data  # Features

y = iris.target  # Labels


# Splitting the dataset 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)


# Creating the SVM classifier (with a linear kernel)

svm_classifier = SVC(kernel='linear', random_state=42)


# Training the model
```

```
svm_classifier.fit(X_train, y_train)


# Predicting the test results

y_pred = svm_classifier.predict(X_test)


# Evaluating the model

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

class_report = classification_report(y_test, y_pred)


# Output results

print(f"Accuracy: {accuracy:.2f}")

print("\nConfusion Matrix:")

print(conf_matrix)

print("\nClassification Report:")

print(class_report)


# Visualizing the confusion matrix

plt.figure(figsize=(6,4))

sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=iris.target_names, yticklabels=iris.target_names)

plt.title("Confusion Matrix")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.show()
```

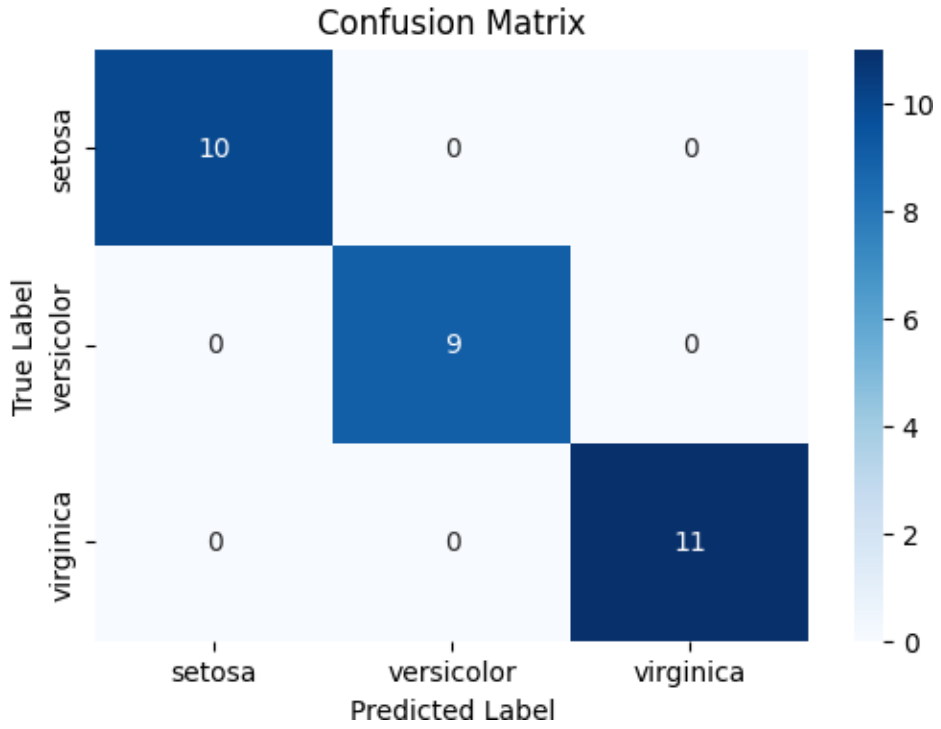
```
Accuracy: 1.00

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

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



Conclusion: In this practical, I implemented a Support Vector Machine (SVM) classifier to classify the Iris dataset. The model achieved a perfect accuracy of 100% on the test set using a linear kernel. The confusion matrix and classification report further validate the performance of the model. SVM is a robust algorithm, especially for datasets with a clear margin of separation between classes, and it can be effectively used for both linear and non-linear classification tasks.