Lab Program 10

Aim:
Implement and demonstrate classification algorithm using Support vector machine Algorithm.

Program:
Implement and demonstrate the working of SVM algorithm for classification.

Import necessary libraries:

In [2]:

```
import numpy as np
import pandas as pd
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, accuracy_score
```

Load and Visualize the dataset

In [10]:

```
iris.target_names
```

Out[10]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [13]:

```
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
print("Features:\n", iris.feature_names)
print("Classes:\n", iris.target_names)
```

```
# Convert to DataFrame for better visualization
df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
columns=iris['feature_names'] + ['target'])
print("\nFirst 5 rows of the dataset:\n", df.head())

# Plotting (optional)
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', edgecolor='k', s=50)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Iris Data Sepal Length vs Sepal Width')

plt.show()
```

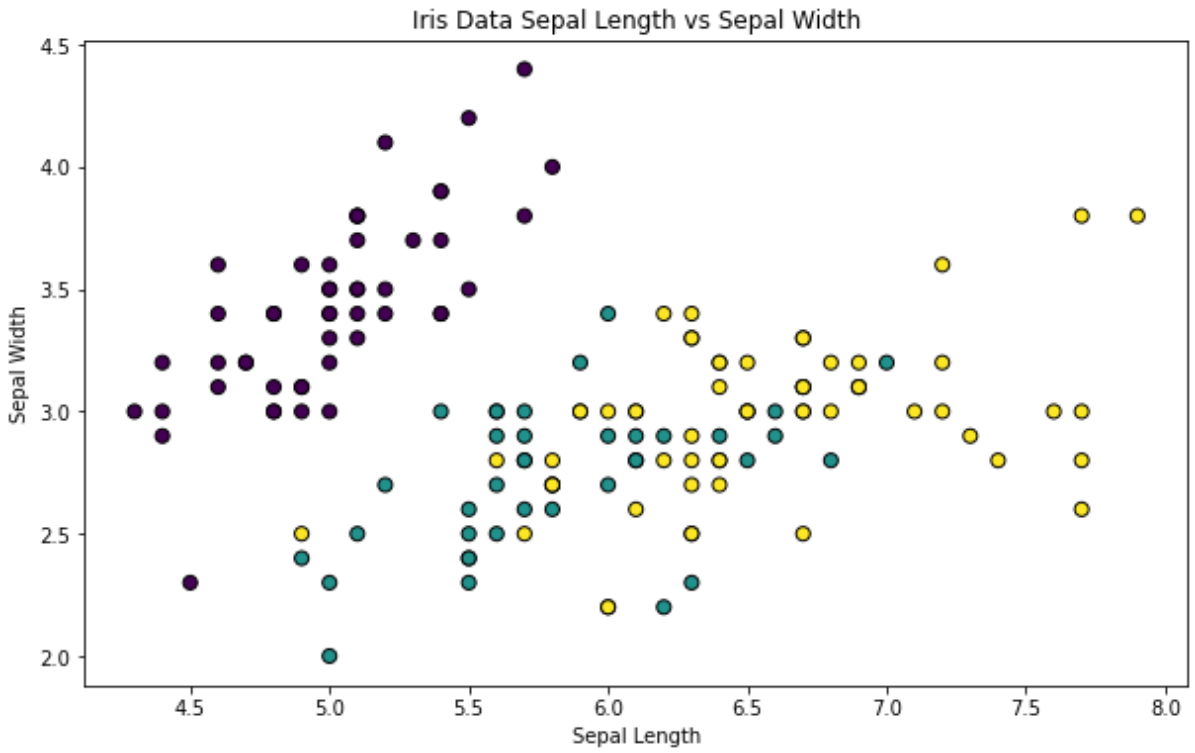
```
Features:
 ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Classes:
 ['setosa' 'versicolor' 'virginica']

First 5 rows of the dataset:
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0     0.0
```

1 0.0

2 0.0

3 0.0

4 0.0



Split the dataset

In [4]:

 # Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Train the SVM classifier

In [5]:

 # Create an SVM classifier

svm_classifier = SVC(kernel='linear', C=1.0, random_state=42)

 # Train the classifier

svm_classifier.fit(X_train, y_train)

Out[5]:

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

Make predictions and evaluate the model:

In [6]:

 # Make predictions

y_pred = svm_classifier.predict(X_test)

 # Evaluate the model

print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))

print("\nAccuracy Score:\n", accuracy_score(y_test, y_pred))

Confusion Matrix:

[[19 0 0]

[0 13 0]

[0 0 13]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Accuracy Score:

1.0

In [15]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
0	sepal length (cm)	150 non-null	float64
1	sepal width (cm)	150 non-null	float64
2	petal length (cm)	150 non-null	float64
3	petal width (cm)	150 non-null	float64
4	target	150 non-null	float64

dtypes: float64(5)

memory usage: 6.0 KB

In [22]:

df['target'].value_counts()

Out[22]:

2.0 50

1.0 50

0.0 50

Name: target, dtype: int64

In [25]:

svm_classifier_rbf = SVC(random_state = 42)

svm_classifier_rbf.fit(X_train, y_train)

Out[25]:

SVC(random_state=42)

In [26]:

 # Make predictions

y_pred = svm_classifier_rbf.predict(X_test)

 # Evaluate the model

print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))

print("\nAccuracy Score:\n", accuracy_score(y_test, y_pred))

Confusion Matrix:

[[19 0 0]

[0 13 0]

[0 0 13]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Accuracy Score:

1.0

In [27]:

```
svm_classifier_poly = SVC(kernel = 'poly', random_state = 42)
svm_classifier_poly.fit(X_train, y_train)
```

Out[27]:

```
SVC(kernel='poly', random_state=42)
```

In [28]:

```
# Make predictions
y_pred = svm_classifier_poly.predict(X_test)

# Evaluate the model
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:\n", accuracy_score(y_test, y_pred))
```

Confusion Matrix:

```
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	0.92	0.96	13
2	0.93	1.00	0.96	13

accuracy			0.98	45
macro avg	0.98	0.97	0.97	45
weighted avg	0.98	0.98	0.98	45

Accuracy Score:

0.9777777777777777

In [ ]: