EXP-2

IMPLEMENT A CLASSIFIER USING OPEN-SOURCE DATASET

Aim:- To implement a simple classifier using an open source dataset.

Description:-

This experiment uses the iris dataset from scikit-learn, a well known open source dataset, to classify iris flowers into three species using the K-Nearest Neighbors (KNN) algorithm. The classifier is evaluated using a confusion matrix.

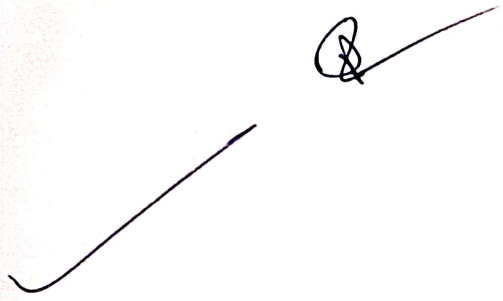
Procedure:-

1) Import the dataset and necessary libraries.
2) Split the dataset into training and test sets.
3) Train the KNN classifier on the training data.
4) Predict the test data.
5) Evaluate the model using confusion matrix, classification report, precision etc

Program:-

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import kNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
    ConfusionMatrixDisplay
import matplotlib.pyplot as plt.
```

```
iris = load_iris()
import pandas as pd
df = pd.DataFrame(X, columns = iris.feature_names)
df['Target'] = [iris.target_names[i] for i in y]
df
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    = 0.3, random_state = 42)
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_predicted = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_predicted)
print(f"Accuracy : {accuracy}")
cm = confusion_matrix(y_test, y_predicted)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels
                              = iris.target_names)
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```



Result:- A KNN classifier was successfully implemented using the Iris dataset. The model was evaluated using a confusion matrix.

Output :-

Confusion Matrix :-

True \ Predicted	setosa	versicolor	virginica
setosa	19	0	0
versicolor	0	13	0
virginica	0	0	13

```
4.8 matplotlib-3.10.3 pillow-11.3.0

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: pip install --upgrade pip
```

In [3]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDispla
import matplotlib.pyplot as plt
```

In [4]:
```
iris = load_iris()
```

X = iris.data y = iris.target

In [30]:
```
import pandas as pd
```

In [33]:
```
df = pd.DataFrame(X, columns = iris.feature_names)
```

In [ ]:
```

```

In [43]:
```
df['Target'] = [iris.target_names[i] for i in y]
df
```

Out[43]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Training the Model

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_s
```

In [12]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
```

In [13]:
```
knn.fit(X_train, y_train)
```

Out[13]:
```
KNeighborsClassifier(n_neighbors=3)
```

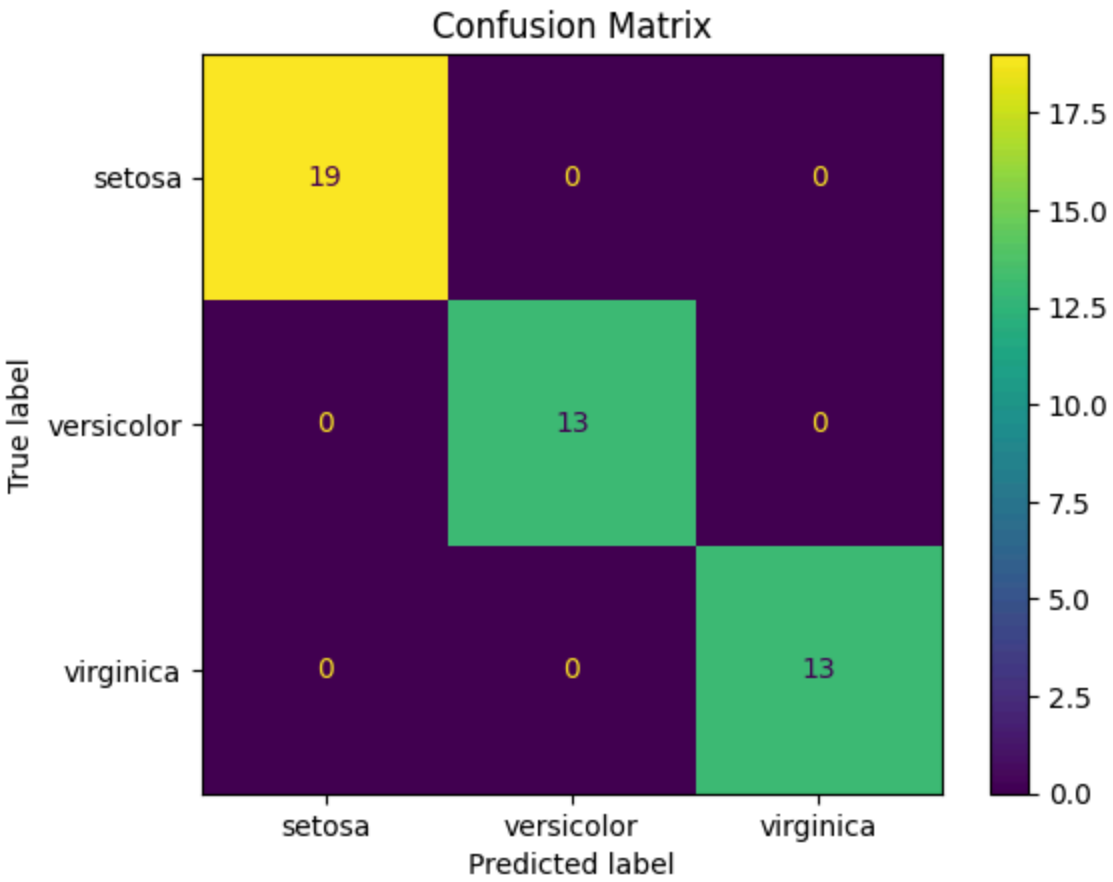
In [14]:
```
y_predicted = knn.predict(X_test)
```

Evaluation of the Model

In [15]:
```
accuracy = accuracy_score(y_test, y_predicted)
print(f"Accuracy: {accuracy} ")
```

```
Accuracy: 1.0
```

In [25]:
```
cmf = confusion_matrix(y_test, y_predicted)
disp = ConfusionMatrixDisplay(confusion_matrix = cmf, display_labels = iris.target_
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```



In [26]:
```
from sklearn.metrics import classification_report
```

In [28]:
```
print(classification_report(y_test, y_predicted))
```

```
              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
```