Practical No. 7

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

Aim: To implement Decision Tree using Scikit-learn.

Theory: Decision Trees are supervised learning algorithms used for both classification and regression tasks. In classification, the tree is structured by splitting the data based on feature values that lead to the purest classification of the target variable.

Key Concepts:

- Nodes: Points where the data splits based on the value of a feature.
- Edges: Links between nodes that represent the outcome of a split.
- Root Node: The top node where the data is first split.
- Leaf Nodes: Terminal nodes representing a classification label.

Decision Tree Algorithm:

1. Split the Dataset: The dataset is split at each node based on the best feature that reduces impurity (e.g., Gini index, Information Gain).
2. Recursion: This process continues recursively, forming a tree structure.
3. Stopping Criteria: The tree stops growing when all data points are classified, or a specified depth or minimum number of samples is reached.
4. Prediction: For new input data, the tree is traversed from the root to a leaf node to predict the class label.

Impurity Measures:

- Gini Index: Measures the impurity of a node, used by default in classification tasks.
- Entropy: Another measure to decide the best split based on the information gain.

Advantages:

- Simple to understand and interpret.
- Can handle both numerical and categorical data.
- Does not require data normalization.

Disadvantages:

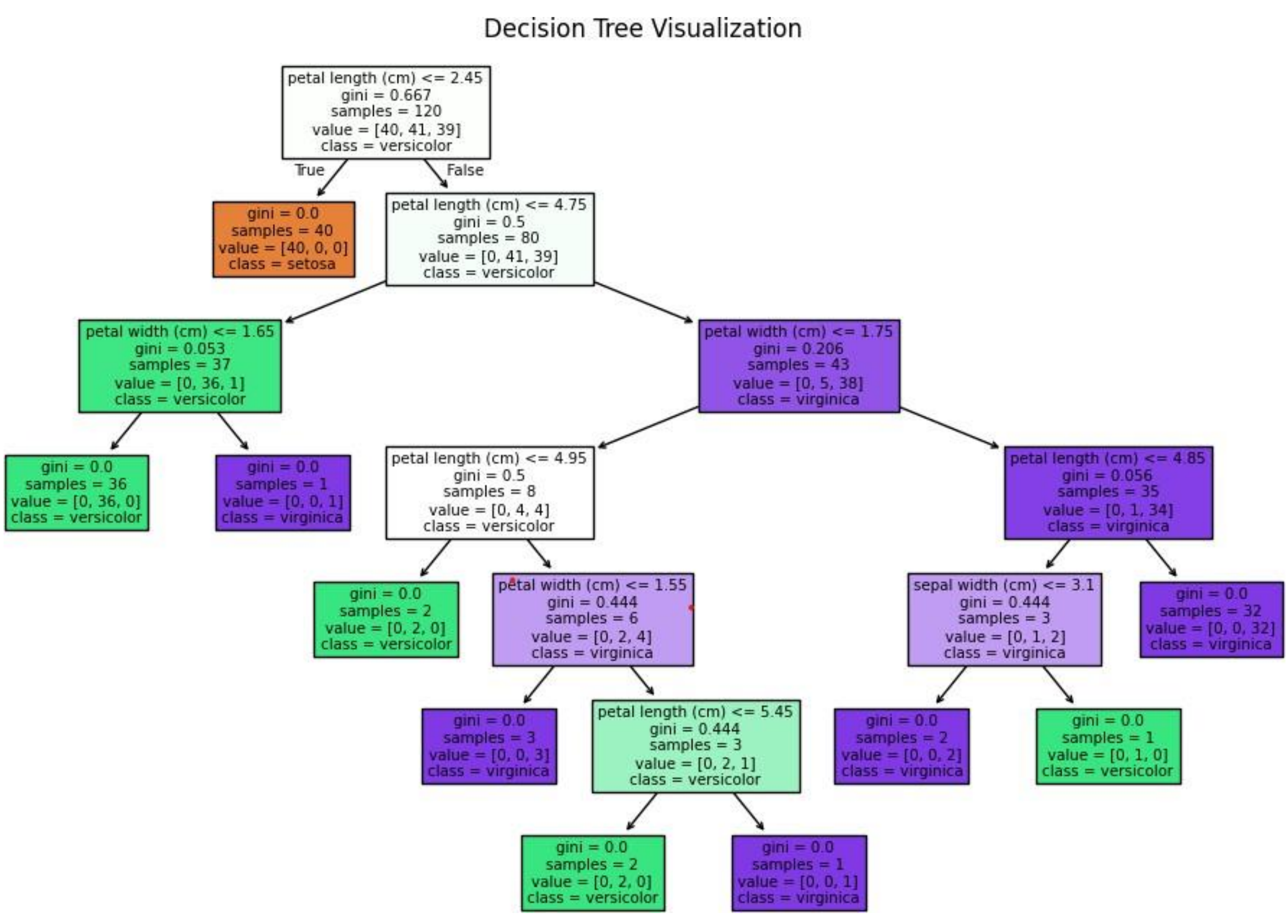
- Prone to overfitting, especially with noisy data.
- Small variations in data can result in completely different trees.

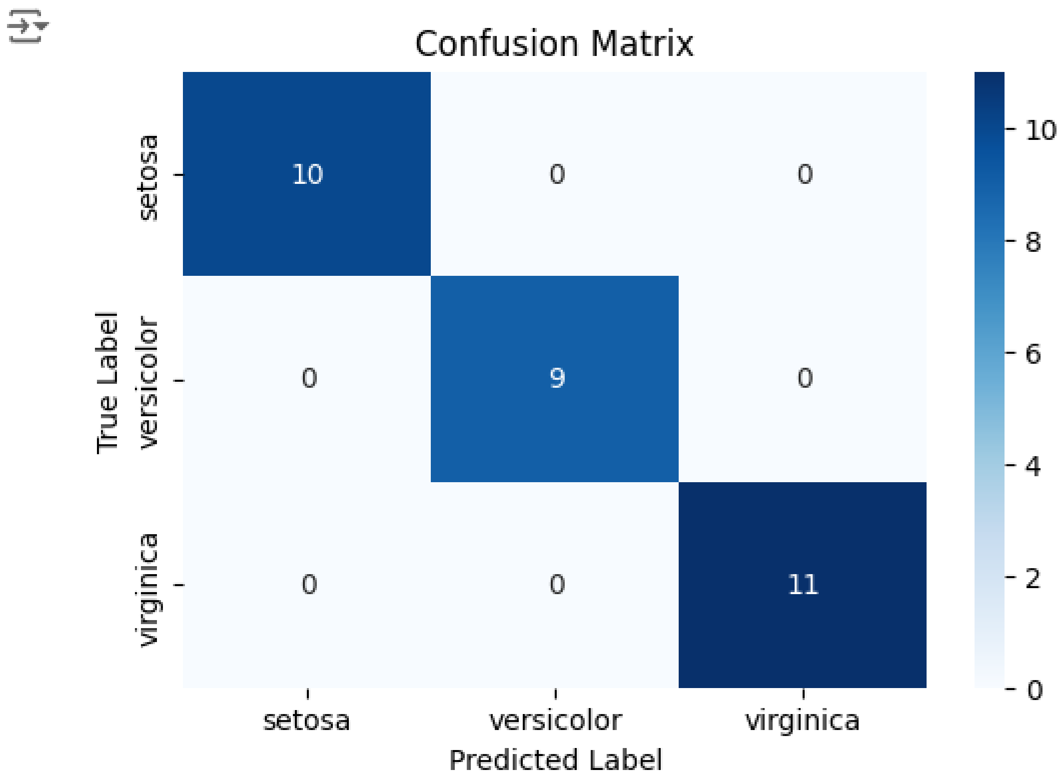
Code and Output:

```
# 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.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn import tree

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 Decision Tree Classifier

dt_classifier = DecisionTreeClassifier(criterion='gini', random_state=42)

# Training the model

dt_classifier.fit(X_train, y_train)

# Predicting the test results

y_pred = dt_classifier.predict(X_test)
```

```python
# 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 Decision Tree

plt.figure(figsize=(12,8))

tree.plot_tree(dt_classifier, feature_names=iris.feature_names,
class_names=iris.target_names, filled=True)

plt.title("Decision Tree Visualization")

plt.show()

# 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 successfully implemented a Decision Tree classifier to classify the Iris dataset. The model achieved a perfect accuracy of 100% on the test set, demonstrating the strength of Decision Trees for such well-separated data. The visual representation of the tree helps in understanding how the model makes predictions based on feature splits. Decision Trees are interpretable and effective models but may require careful tuning to avoid overfitting, especially on larger datasets.