**Aim**

To implement a decision tree algorithm in Python for classification tasks using the scikit-learn library.

**Procedure**

1. **Import Libraries**: Load necessary Python libraries (sklearn, pandas, numpy).
2. **Load Dataset**: Use a dataset suitable for classification (e.g., Iris dataset).
3. **Preprocess Data**: Handle missing values, encode categorical variables if needed.
4. **Split Data**: Divide the dataset into training and testing sets.
5. **Train Model**: Use DecisionTreeClassifier from sklearn to train the model.
6. **Evaluate Model**: Test the model on unseen data and measure accuracy.
7. **Visualize Tree**: Plot the decision tree using plot\_tree().

**Code**

**import pandas as pd**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier, plot\_tree**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import load\_iris**

**# Load dataset**

**iris = load\_iris()**

**X, y = iris.data, iris.target**

**# Split data into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Initialize and train decision tree classifier**

**clf = DecisionTreeClassifier()**

**clf.fit(X\_train, y\_train)**

**# Evaluate model**

**accuracy = clf.score(X\_test, y\_test)**

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

**# Visualize decision tree**

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

**plot\_tree(clf, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True)**

**plt.show()**

**Output**

**Model Accuracy: 0.95**