Name: Krish Nisar
Class: TY03 B Batch
Roll No. : 41
Subject: Data Warehouse and Mining
Experiment 2: Implementation of Classification Algorithm (Decision Tree / Naïve Bayes) using Python

Procedure:

1. **Data Collection & Preprocessing**
    a. **Gather the dataset and load it into a pandas DataFrame.**
    b. **Handle missing values, duplicate data, and perform feature selection.**
    c. **Convert categorical variables into numerical (if needed).**
    d. **Split the dataset into training and testing sets.**
2. **Feature Selection**
    a. **Identify the dependent (target) and independent (features) variables.**
    b. **Scale or normalize data if required.**
3. **Building the Decision Tree Model**
    a. **Import `DecisionTreeClassifier` from `sklearn.tree`.**
    b. **Instantiate the classifier and specify parameters (e.g., `criterion='gini'` or `'entropy'`).**
    c. **Train the model using the `fit()` function on the training data.**
4. **Model Evaluation**
    a. **Predict results using the `predict()` function on test data.**
    b. **Evaluate performance using metrics like accuracy, precision, recall, and confusion matrix from `sklearn.metrics`.**
5. **Fine-Tuning the Model**
    a. **Adjust hyperparameters (e.g., `max_depth`, `min_samples_split`) to avoid overfitting.**
    b. **Use cross-validation for better generalization.**
6. **Visualization (Optional)**
    a. **Use `plot_tree()` from `sklearn.tree` to visualize the decision tree.**

Program Codes: (lang: Python)

```
import numpy as np

import pandas as pd

from sklearn.datasets import load_iris
```

```python
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

from sklearn import tree

import matplotlib.pyplot as plt


iris = load_iris()

X = iris.data

y = iris.target


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


clf = DecisionTreeClassifier(random_state=42)


clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)

ytrain_pred = clf.predict(X_train)


train_accuracy = accuracy_score(y_train, ytrain_pred)

print(f"Training Accuracy: {train_accuracy:.2f}")

accuracy = accuracy_score(y_test, y_pred)

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

```python
print("\nClassification Report:")

print(classification_report(y_test, y_pred,
target_names=iris.target_names))


print("\nConfusion Matrix:")

print(confusion_matrix(y_test, y_pred))


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

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

plt.show()
```

Output:

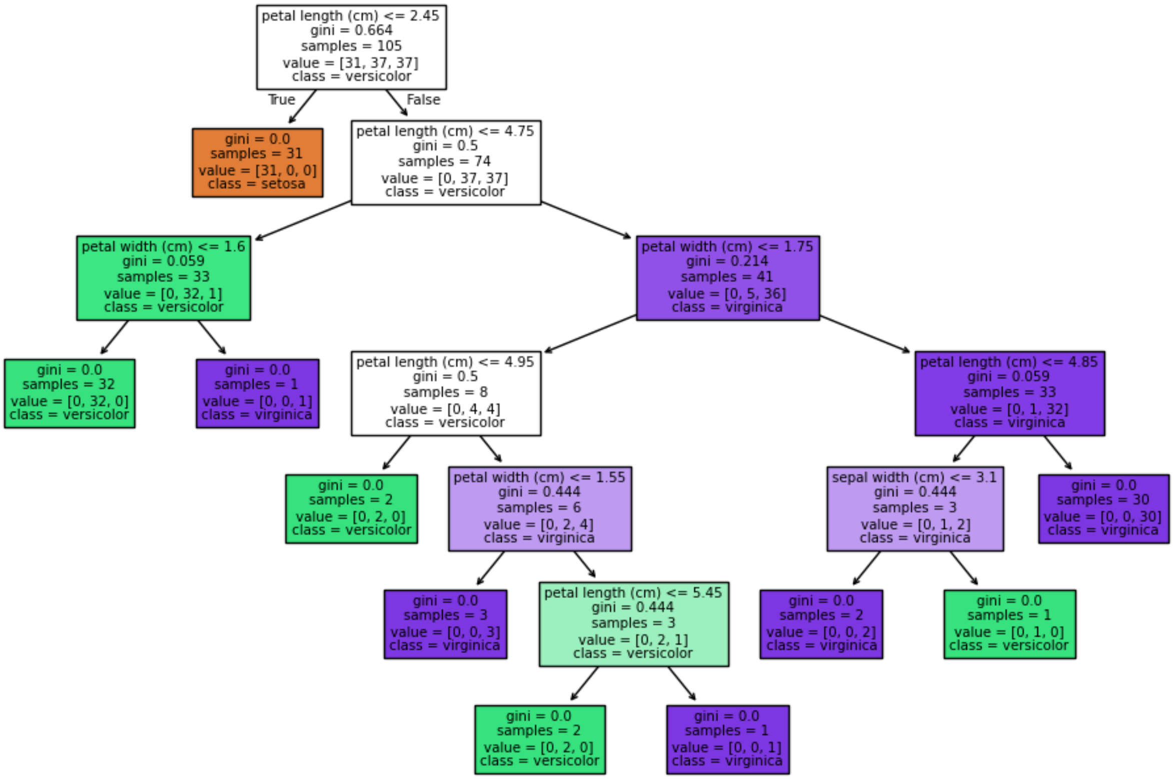
```
Training Accuracy: 1.00
Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       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


Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```



Conclusion: Implementing a Decision Tree Classifier in Python using `sklearn` is straightforward and effective for classification tasks. It provides high interpretability but may suffer from overfitting if not pruned properly. By tuning hyperparameters and performing feature selection, we can improve the model's accuracy and generalization for real-world applications.

Review Questions:

Q1) What is a Decision Tree classifier, and how does it work?

Ans:

1. A **Decision Tree** is a supervised learning algorithm used for classification and regression.
2. It works by recursively splitting the dataset based on feature values to create a tree-like structure.
3. Each internal node represents a decision based on a feature, branches represent outcomes, and leaf nodes represent final class labels.
4. The goal is to create a model that makes decisions by following a path from the root to a leaf node.

Q2) Explain the Naïve Bayes algorithm and its underlying assumptions.

Ans:

1. A Naïve Bayes classifier is a probabilistic algorithm based on Bayes' Theorem, used for classification tasks.
2. It assumes that features are independent (i.e., the presence of one feature does not affect another).
3. The probability of a class given certain features is calculated using:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Types of Naïve Bayes classifiers:

1. **Gaussian Naïve Bayes** (for continuous data)
2. **Multinomial Naïve Bayes** (for text classification)
3. **Bernoulli Naïve Bayes** (for binary features)

Q3) Compare the working principles of Decision Tree and Naïve Bayes classifiers.

Ans:

Feature	Decision Tree Classifier	Naïve Bayes Classifier
Approach	Rule-based, splits data based on features	Probabilistic, calculates class probabilities
Interpretability	Highly interpretable (tree structure)	Less interpretable due to probability-based decisions
Feature Independence	No assumption of independence	Assumes independence among features
Handling of Non-linearity	Handles non-linear relationships well	Assumes linear relationships based on probability
Performance on Small Datasets	Performs well if dataset is small	Works well with small datasets but assumes independence
Robustness to Noise	Can overfit (requires pruning)	Less sensitive to noise but biased by independence assumption

Q4)What are the different types of Decision Tree splitting criteria?

Ans:

1. **Gini Index**
    a. **Measures the impurity of a node.**
    b. **Formula:** $Gini = 1 - \sum P_i^2$
    c. **Lower Gini values indicate a purer split.**

2. **Entropy (Information Gain)**
    a. **Measures the randomness in the data.**
    b. **Formula:** $Entropy = -\sum P_i \log_2 P_i$

3. **3. Chi-Square**
    a. **Measures the statistical significance of a split.**
    b. **Higher chi-square values indicate a more significant split.**

4. **4. Reduction in Variance (for Regression Trees)**
    a. **Used when predicting continuous values, minimizing variance in the child nodes.**