Name:-Priyal Makwana
TY04_B-35

Aim:-Implementation of Classification Algorithm (Decision Tree / Naïve Bayes) using Python

Introduction: A Decision Tree classifier works by splitting the dataset into smaller subsets based on feature values, forming a tree-like structure that helps in making decisions. It is easy to understand and interpret.

The Naive Bayes classifier is a probabilistic algorithm based on Bayes' Theorem, which assumes that the features are independent of each other. Despite this assumption, Naive Bayes performs efficiently and is widely used for classification problems.

Procedure:- 1) **Decision Tree Classification**

1. Import required libraries.
2. Load the Iris dataset.
3. Separate features (X) and target labels (y).
4. Split the dataset into training and testing data.
5. Create the Decision Tree classifier.
6. Train the classifier using training data.
7. Predict class labels for test data.
8. Evaluate the model using accuracy and classification report.
9. Visualize the decision tree.

2) **Naive Bayes Classification**

1. Import required libraries.
2. Load the Iris dataset.
3. Separate features (X) and target labels (y).

4. Split the dataset into training (70%) and testing (30%) data.
5. Create a Gaussian Naive Bayes classifier.
6. Train the classifier using training data.
7. Predict class labels for test data.
8. Evaluate the model using accuracy and classification report.

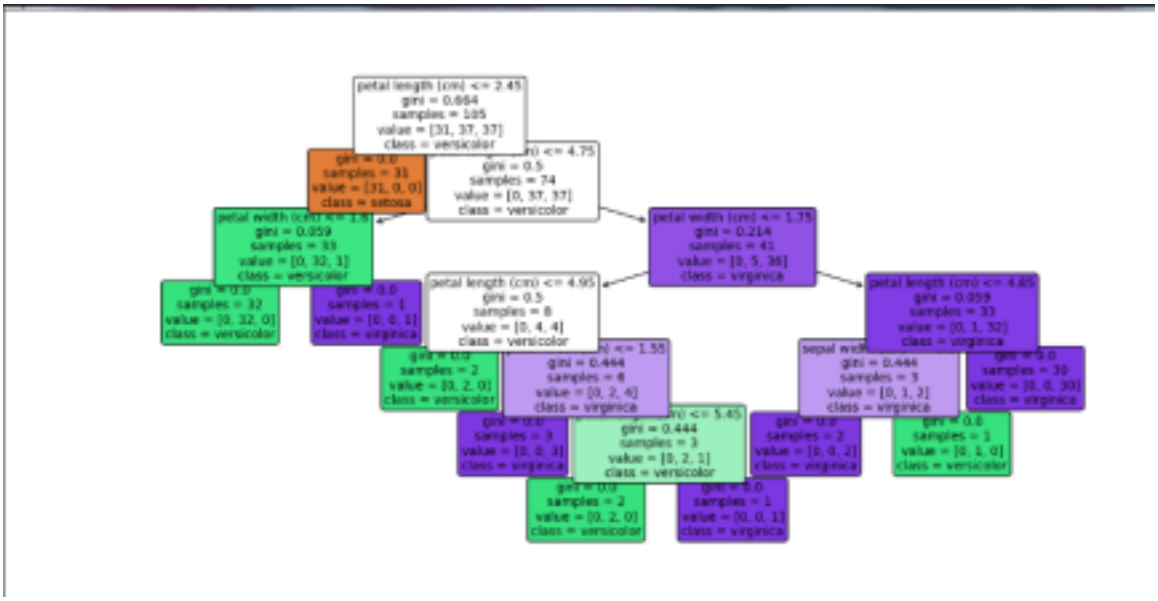
Programe and output

```
# Step 0: Import required libraries

import numpy as np

import pandas as pd

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report

import matplotlib.pyplot as plt

from sklearn.tree import plot_tree


# Step 1: Load the dataset (Iris dataset)

iris = load_iris()

X = iris.data # Features (independent variables) y =

iris.target # Target labels (dependent variable)
```

```
# Step 2: Split the data 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
)

# Step 3: Initialize the Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Step 4: Train the classifier with training data
dt_classifier.fit(X_train, y_train)

# Step 5: Predict the target labels on the test data
y_pred = dt_classifier.predict(X_test)

# Step 6: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Additional evaluation: Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))

# Step 7: Plot the decision tree
```

```
plt.figure(figsize=(14, 14))

plot_tree(

 dt_classifier,

 filled=True,
 feature_names=iris.feature_names,

 class_names=iris.target_names,

 rounded=True,

 fontsize=10

)

plt.show()
```

```
========== RESTART: C:/Users/lab505/Downloads/dt.py ==========
Matplotlib is building the font cache; this may take a moment.
Accuracy: 100.00%

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
```



Naive Bayes Classification on Iris Dataset

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
import numpy as np
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.3, random_state=42
)

# Create a Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Train the classifier
gnb.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = gnb.predict(X_test)
```

```
# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

```
========== RESTART: C:/Users/lab505/Downloads/nb.py ==========
Accuracy: 97.78%

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      0.92      0.96        13
   virginica       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
```

Conclusion:- Decision Tree and Naïve Bayes are supervised classification algorithms. Decision Tree is rule-based and easy to interpret, while Naïve Bayes is probability-based and fast. Both are effective and commonly used for classification tasks**.**

Review Questions:-

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

A Decision Tree is a supervised learning algorithm that classifies data by splitting it into branches based on feature values. It selects the best feature at each node using a splitting criterion and continues until a class label is reached at the leaf node.

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

Naïve Bayes is a probabilistic classifier based on Bayes' Theorem. It predicts the class with the highest probability assuming that all features are independent of each other.

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

Decision Tree	Naïve Bayes
Rule-based classification	Probability-based classification
Uses feature splitting	Uses Bayes' Theorem
No independence assumption	Assumes feature independence
Easy to visualize	Fast and simple

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

Gini Index
Information Gain
Entropy
Gain Ratio

Github link:-https://github.com/priyalmakwana04/DWM-EXPERIMENTS