Name : Bhoomi Mangesh Naik
Class : D15C
Roll No. : 60
Practical No. : 4

Aim : To implement Classification Algorithms : Decision tree and Naive Bayes' algorithms using Python

Theory :

1. Decision Tree

- A **Decision Tree** is a supervised learning algorithm used for classification and regression tasks.
- It works by splitting the dataset into subsets based on the most significant attribute, chosen using measures like **Gini Index** or **Information Gain**.
- Each internal node represents a decision rule, each branch represents an outcome, and each leaf node represents a class label.
- Advantages: Easy to understand, interpretable, handles both numerical & categorical data.
- Disadvantage: Can overfit the dataset if not pruned properly.

2. Naïve Bayes

- Naïve Bayes is a **probabilistic classifier** based on **Bayes' Theorem** with the assumption of independence among predictors.
- Formula:

$$P(C \mid X) = \frac{P(X \mid C) \cdot P(C)}{P(X)}$$

- It calculates the probability of a class given the input features and predicts the class with the highest probability.
- Commonly used for text classification (spam filtering, sentiment analysis).
- Advantage: Works well with small data and high-dimensional datasets.
- Limitation: Assumes features are independent, which may not always be true.

Syntax

Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='gini', random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Naïve Bayes:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Widgets and Properties (Applicable Concepts in Python/Sklearn)

- **DecisionTreeClassifier Parameters**
  - criterion: Splitting rule (gini or entropy).
  - max_depth: Maximum depth of the tree (prevents overfitting).
  - random_state: Ensures reproducibility.
  - min_samples_split: Minimum samples required to split a node.
- **GaussianNB Properties**
  - priors: Prior probabilities of classes.
  - var_smoothing: Portion of largest variance added to variances for stability.
- **Common Methods (Both Models)**
  - .fit(X_train, y_train) → Train the model.
  - .predict(X_test) → Predict target values.
  - .score(X_test, y_test) → Accuracy score.

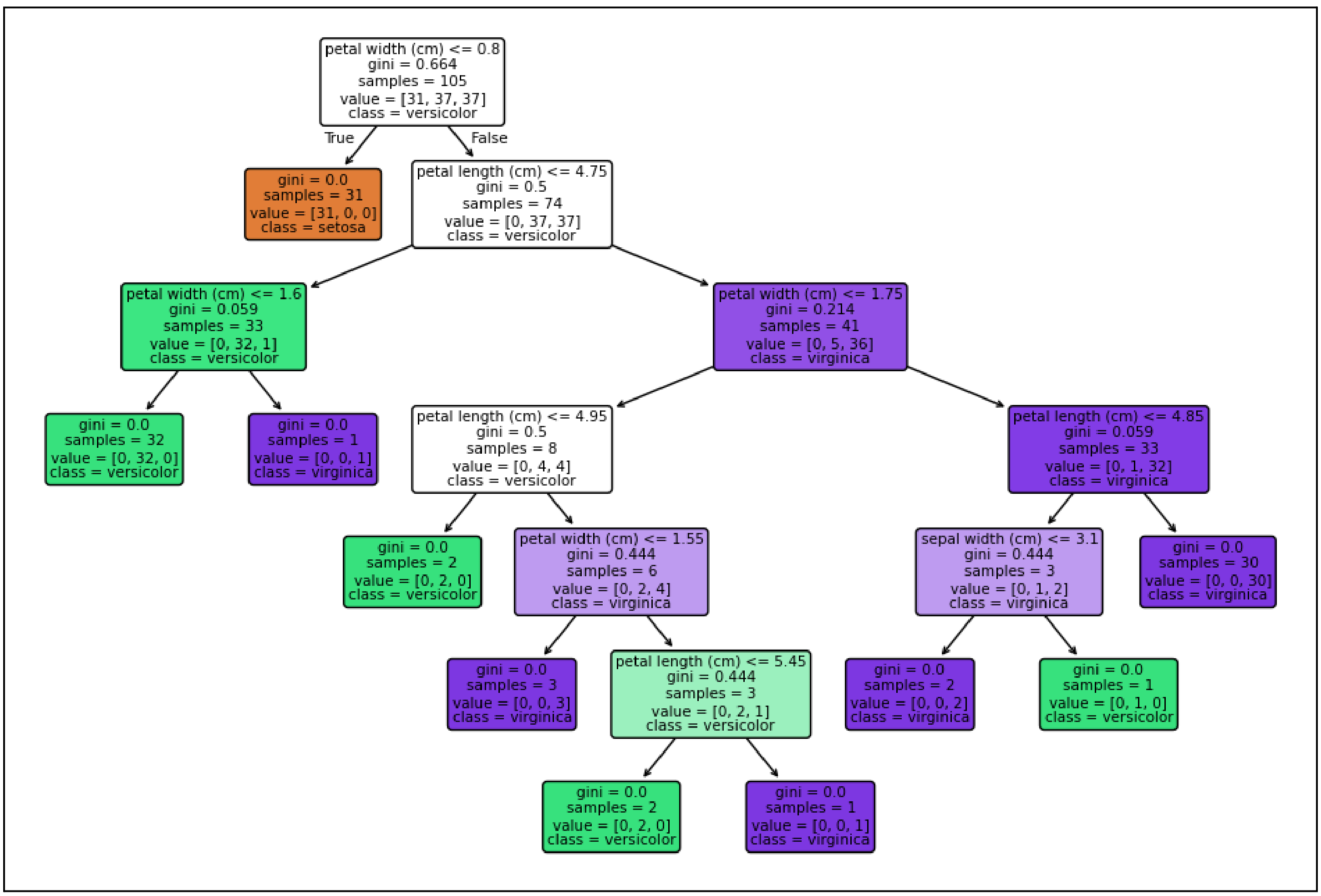
Code and Output :

Decision Tree :

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt

iris = load_iris()
X, y = iris.data, iris.target
feature_names = iris.feature_names
class_names = iris.target_names

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

dt_model = DecisionTreeClassifier(criterion='gini', random_state=0)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_test)

print("Decision Tree Results:")
print("Accuracy:", accuracy_score(y_test, dt_pred))
print(classification_report(y_test, dt_pred, target_names=class_names))
```

```
plt.figure(figsize=(12,8))
plot_tree(dt_model, filled=True, feature_names=feature_names, class_names=class_names, rounded=True)
plt.show()
```

```
Decision Tree Results:
Accuracy: 1.0
              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
```

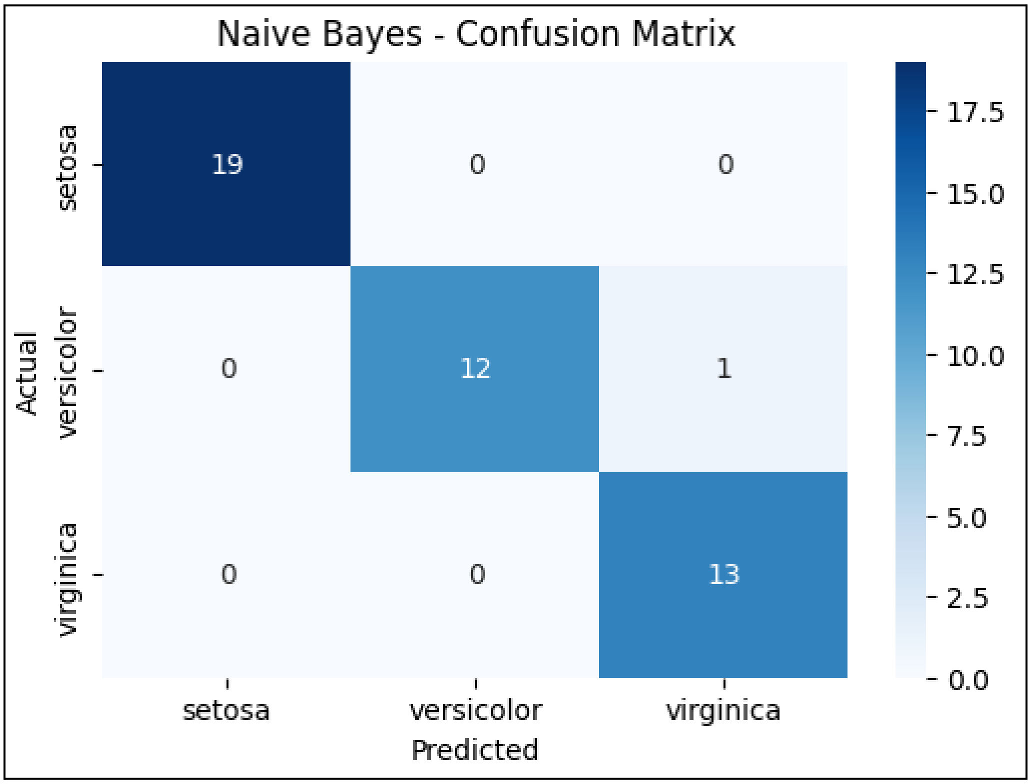


Naïve Bayes:

```
import pandas as pd
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, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

iris = load_iris()
X, y = iris.data, iris.target
class_names = iris.target_names

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

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)

print("Naive Bayes Results:")
print("Accuracy:", accuracy_score(y_test, nb_pred))
print(classification_report(y_test, nb_pred, target_names=class_names))

cm = confusion_matrix(y_test, nb_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names,
yticklabels=class_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Naive Bayes - Confusion Matrix")
plt.show()
```

```
Naive Bayes Results:
Accuracy: 0.9777777777777777
              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 :

In this experiment, we implemented two classification algorithms, Decision Tree and Naïve Bayes, using Python. The Decision Tree gave very high accuracy on the Iris dataset as it can perfectly separate the classes, while Naïve Bayes also performed well though with slightly lower accuracy. From this, we can conclude that both algorithms are useful for classification problems, but their performance can vary depending on the dataset.