Name : Bhavna Sanjay More

Div : TY3

Batch : B

Roll no : 38

Subject : Data WareHouse And Mining

Experiment 3 : Implementation of Classification Algorithm (Decision Tree / Naïve Bayes) using Python

Aim:

To implement and understand the working of Decision Tree and Naïve Bayes classifiers using Python, and to compare their working principles and performance.

Introduction:

Classification is a supervised machine learning technique used to predict the class or category of a given data point. Two popular classification algorithms are:

- **Decision Tree Classifier:** A tree-like model that splits the dataset into subsets based on feature values to make predictions.
- **Naïve Bayes Classifier:** A probabilistic algorithm based on Bayes' theorem, which assumes independence between features.

Procedure:

Data Preparation: Load or generate a dataset suitable for classification.

Algorithm Implementation:

- **For Decision Tree:** Use the DecisionTreeClassifier class from sklearn.tree.
- **For Naïve Bayes:** Use the GaussianNB class from sklearn.naive_bayes.

Training and Testing: Split the dataset into training and testing sets, train the models, and evaluate their performance.

Visualization: Plot the Decision Tree and compare the results of both classifiers.

Program Codes:

Decision Tree Classifier:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import load_iris


# Load a sample dataset (Iris dataset in this case)
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)


# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Create a DecisionTreeClassifier model
model = DecisionTreeClassifier(random_state=42)


# Train the model
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
# Make predictions
y_pred = model.predict(X_test)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

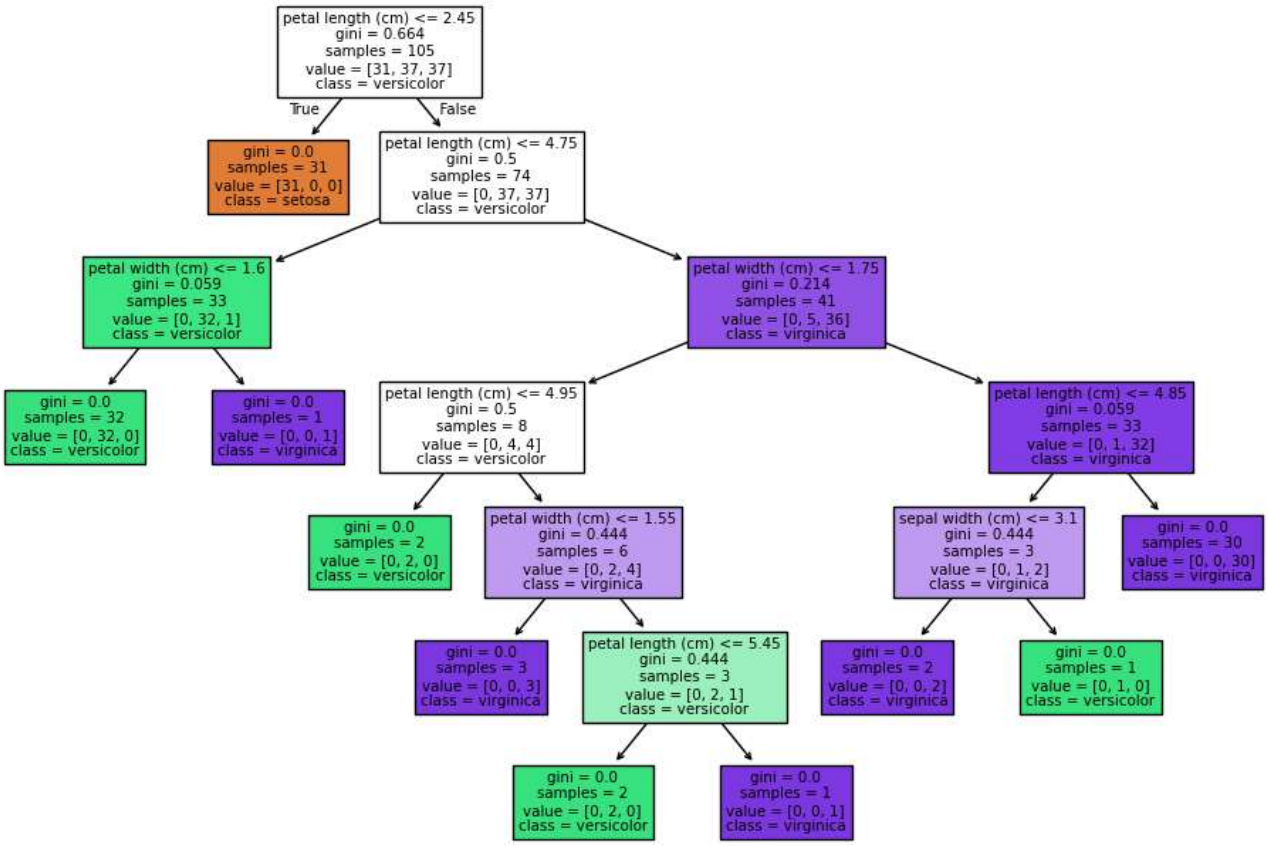
```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
```

```
                2       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
```

```
# Visualize the tree (Optional)
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
plot_tree(model, filled=True, feature_names=data.feature_names, class_names=data.target_name
plt.show()
```



Naïve Bayes Classifier:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report


# Train Naïve Bayes
nb_clf = GaussianNB()
nb_clf.fit(X_train, y_train)
```

```
GaussianNB()
```

```
# Predict and evaluate
y_pred_nb = nb_clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Classification Report:\n", classification_report(y_test, y_pred_nb))
```

```
Accuracy: 0.9777777777777777
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       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 Trees are intuitive and easy to interpret but can overfit if not pruned properly.

Naïve Bayes is computationally efficient and works well with high-dimensional data but relies on the assumption of feature independence.

Both algorithms are powerful tools for classification tasks, and the choice depends on the dataset and problem requirements.

Review Questions:

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

- A Decision Tree classifier is a tree-like model that splits data into subsets based on feature values to make predictions.

- It starts at the root, splits the data using the best feature, and repeats this process recursively until it reaches leaf nodes, which represent class labels.
- It is easy to interpret but can overfit if not pruned.

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

- Naïve Bayes is a probabilistic classifier based on Bayes' Theorem.
- It assumes that all features are independent of each other given the class label (feature independence).
- It works well with high-dimensional data and is computationally efficient but may struggle if the independence assumption is violated.

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

- **Decision Tree:** Splits data based on feature values, handles both numerical and categorical data, and is interpretable but prone to overfitting.
- **Naïve Bayes:** Uses probability to classify data, assumes feature independence, is fast and efficient but less interpretable and relies on strong assumptions.

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

Gini Impurity: Measures the likelihood of incorrect classification.

Entropy: Measures disorder or uncertainty in the dataset.

Information Gain: Measures the reduction in entropy after a split.

Chi-Square: Used for categorical targets to measure statistical significance.

Variance Reduction: Used for regression tasks to minimize variance.

Github

https://github.com/bhavnamore/DWM_Kmeans/blob/main/DWM_KMeans%20(1).ipynb

Start coding or generate with AI.