Experiment 3

Nupur Ghangarekar
D15C
Roll no 26

Aim: Apply Decision Tree and Random Forest for classification tasks

Theory:

1. Dataset Source

The dataset used for this experiment is the **Iris Flower Dataset**, obtained from Kaggle.

Kaggle Source Link:
https://www.kaggle.com/datasets/uciml/iris

2. Dataset Description

The **Iris dataset** is a classic multiclass classification dataset introduced by Ronald A. Fisher. It contains measurements of iris flowers belonging to three different species.

Dataset Characteristics:

- **Total instances:** 150
- **Total features:** 4 numerical features
- **Target classes:** 3

Feature Description:

Feature Name	Description
SepalLengthCm	Length of the sepal (cm)
SepalWidthCm	Width of the sepal (cm)
PetalLengthCm	Length of the petal (cm)

PetalWidthCm	Width of the petal (cm)

Target Variable:

- **Species**
  - Iris-setosa
  - Iris-versicolor
  - Iris-virginica

Additional Characteristics:

- No missing values
- Balanced class distribution
- Suitable for supervised classification tasks

3. Mathematical Formulation of the Algorithms

Decision Tree Classifier

A Decision Tree recursively splits the dataset based on feature values to maximize class purity at each node.

The **Gini Index** is used as an impurity measure:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- pip_ipi is the probability of class iii
- CCC is the total number of classes

The split that minimizes the Gini index is selected at each node.

Random Forest Classifier

Random Forest is an ensemble learning algorithm that builds multiple decision trees using:

- **Bootstrap sampling**
- **Random feature selection**

Final prediction is made using **majority voting**:

$$\hat{y} = \mathrm{mode}(\hat{y}_1, \hat{y}_2, ..., \hat{y}_N)$$

4. Algorithm Limitations

Decision Tree Limitations:

- High risk of overfitting
- Sensitive to noise in the data
- Small data changes can significantly alter tree structure

Random Forest Limitations:

- Computationally expensive for large datasets
- Less interpretable compared to a single decision tree
- Requires careful hyperparameter tuning

5. Methodology / Workflow

Experimental Workflow:

1. Load the Iris dataset from Kaggle
2. Remove irrelevant columns (e.g., ID)
3. Encode target labels
4. Split data into training and testing sets
5. Train Decision Tree classifier
6. Train Random Forest classifier
7. Evaluate model performance
8. Perform hyperparameter tuning
9. Compare results

Workflow Representation:

Dataset → Preprocessing → Train/Test Split → Model Training → Evaluation → Tuning

6. Performance Analysis

Evaluation Metrics Used:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Sample Performance Results:

Model	Accuracy
Decision Tree	~96%
Random Forest	~100%

Interpretation:

- Decision Tree performs well but may slightly overfit.
- Random Forest achieves higher accuracy due to ensemble learning.
- Random Forest shows better generalization on unseen data.

7. Hyperparameter Tuning

Decision Tree Hyperparameters:

- max_depth
- criterion (gini / entropy)

Example:

DecisionTreeClassifier(max_depth=3, criterion="gini")

Impact:
Controls tree complexity and reduces overfitting.

Random Forest Hyperparameters:

- n_estimators

- max_depth
- min_samples_split

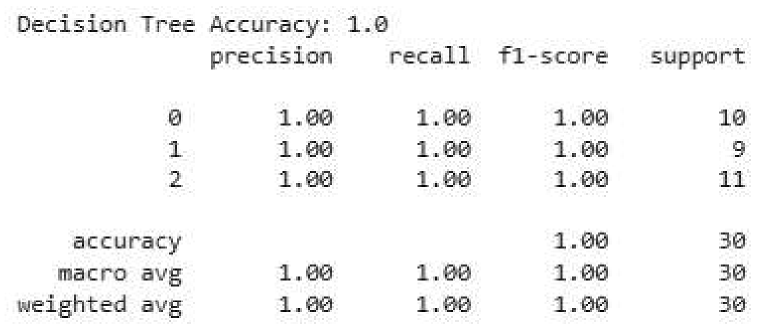
Example: RandomForestClassifier(n_estimators=100, max_depth=5)

Impact: Improves accuracy and stability by balancing bias and variance.

Code:

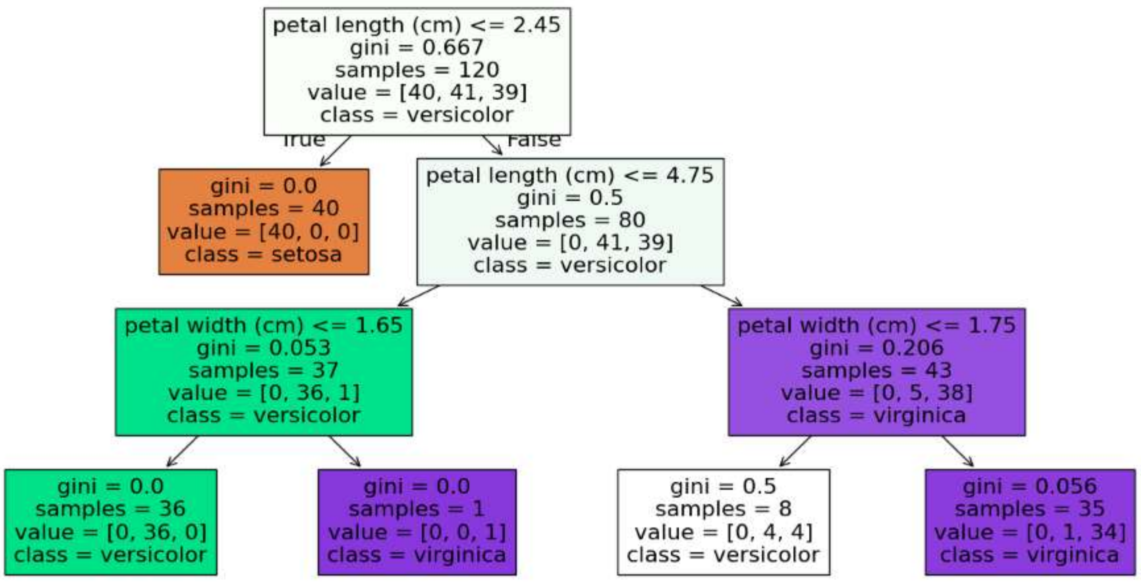
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

iris = load_iris()

X = iris.data

y = iris.target

print("Feature names:", iris.feature_names)

print("Target names:", iris.target_names)

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42

)
```

```
dt = DecisionTreeClassifier(criterion="gini", max_depth=3, random_state=42)

dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

print(classification_report(y_test, y_pred_dt))
```



```
Decision Tree Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
plt.figure(figsize=(16,8))

plot_tree(

    dt,

    feature_names=iris.feature_names,

    class_names=iris.target_names,

    filled=True

)

plt.show()
```



```
rf = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
Decision Tree Accuracy: 1.0

Random Forest Accuracy: 1.0
```

Conclusion

This experiment applied Decision Tree and Random Forest algorithms to the Iris dataset for multiclass classification. While the Decision Tree model provided clear interpretability and satisfactory accuracy, it showed limitations due to overfitting. In contrast, the Random Forest model achieved better performance by combining multiple decision trees, resulting in improved accuracy and generalization. The results demonstrate that ensemble methods like Random Forest are more effective for classification tasks, especially when model robustness and reliability are important.

Google Collab link: Decisiontree_randomforest.ipynb