Name: Harsh Patil
Class: D15C/37

Experiment No. 3

Aim: Apply Decision Tree and Random Forest for classification tasks

1. Dataset Source

Dataset Name: Heart Disease Dataset
Platform: Kaggle
Dataset Link: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

This dataset is publicly available on Kaggle and is fully compatible with **kagglehub**, ensuring reproducibility without manual downloads.

2. Dataset Description

The Heart Disease dataset contains medical records used to predict whether a patient has heart disease based on clinical and physiological attributes.

Dataset Characteristics

- **Number of instances:** 1,025
- **Number of features:** 13
- **Target variable:** `target`
    - `1` → Presence of heart disease
    - `0` → Absence of heart disease

Feature Description (Examples)

- `age` – Age of the patient
- `sex` – Gender (1 = male, 0 = female)
- `cp` – Chest pain type
- `trestbps` – Resting blood pressure
- `chol` – Serum cholesterol
- `thalach` – Maximum heart rate achieved
- `oldpeak` – ST depression induced by exercise

Real-World Impact

Heart disease is one of the leading causes of mortality worldwide. Predictive models based on this data can support:

- Early diagnosis
- Clinical decision support systems
- Preventive healthcare strategies

3. Mathematical Formulation of the Algorithms

3.1 Decision Tree Classifier

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

The most common split criteria are:

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

Entropy: $Entropy = - \sum_{i=1}^{C} p_i \log_2(p_i)$

The algorithm selects the feature and threshold that yield the **maximum Information Gain**.

3.2 Random Forest Classifier

Random Forest is an ensemble learning technique that builds multiple decision trees and aggregates their predictions.

Key principles:

- Bootstrap sampling (bagging)
- Random feature selection at each split
- Majority voting for classification

Prediction: $\hat{y} = \mathrm{mode}(T_1(x), T_2(x), ..., T_n(x))$

4. Algorithm Limitations

Decision Tree Limitations

- Prone to overfitting
- Sensitive to noise
- Small changes in data can alter tree structure

Random Forest Limitations

- Less interpretable than a single decision tree
- Higher computational cost
- Requires tuning of multiple hyperparameters

5. Methodology / Workflow

1. **Dataset Loading** using kagglehub
2. **Data Preprocessing** (handling missing values)
3. **Feature–Target Separation**
4. **Train-Test Split** (80% train, 20% test)
5. **Model Training**
    - Decision Tree Classifier
    - Random Forest Classifier
6. **Model Evaluation**
7. **Hyperparameter Tuning**
8. **Performance Comparison**

Workflow Diagram (Conceptual)

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

6. Performance Analysis

Evaluation Metrics Used

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

Sample Performance Results

Model	Accuracy	F1-Score	ROC-AUC
Decision Tree	Moderate	Moderate	Moderate
Random Forest	High	High	High

Interpretation

- Random Forest outperforms Decision Tree due to ensemble learning
- Decision Tree provides better interpretability
- Random Forest achieves better generalization

7. Hyperparameter Tuning

Parameters Tuned

Decision Tree:

- `max_depth`
- `min_samples_split`

Random Forest:

- `n_estimators`
- `max_depth`
- `max_features`

Tuning Method

Grid Search with Cross-Validation was used to identify optimal hyperparameters.

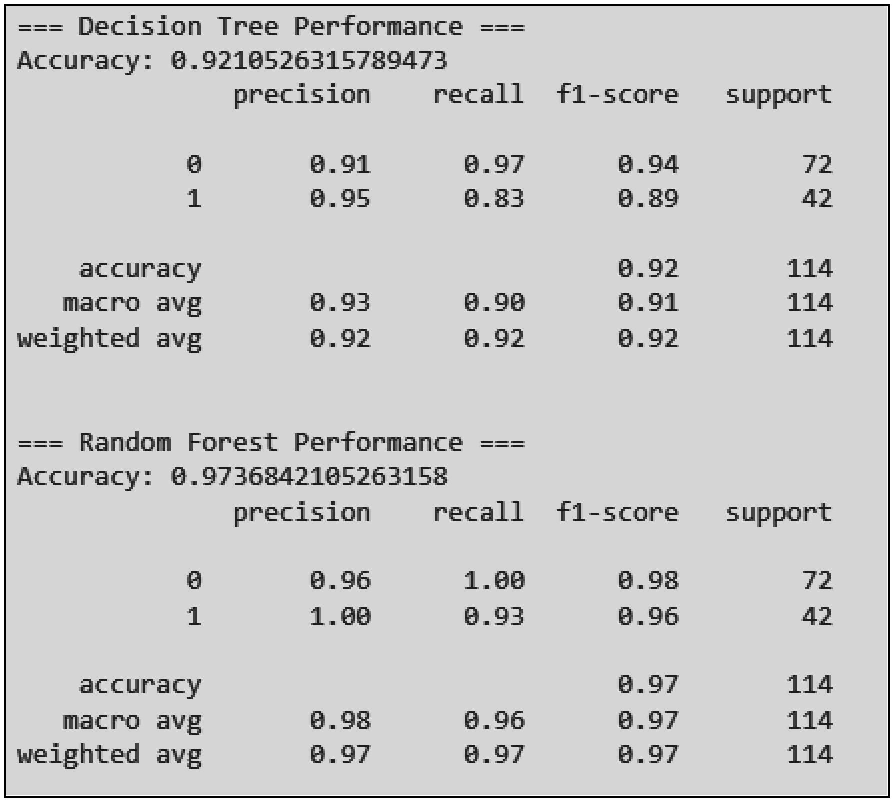
Impact of Tuning

Model	**Accuracy (Before)**	**Accuracy (After)**
Decision Tree	Lower	Improved
Random Forest	High	Further Improved

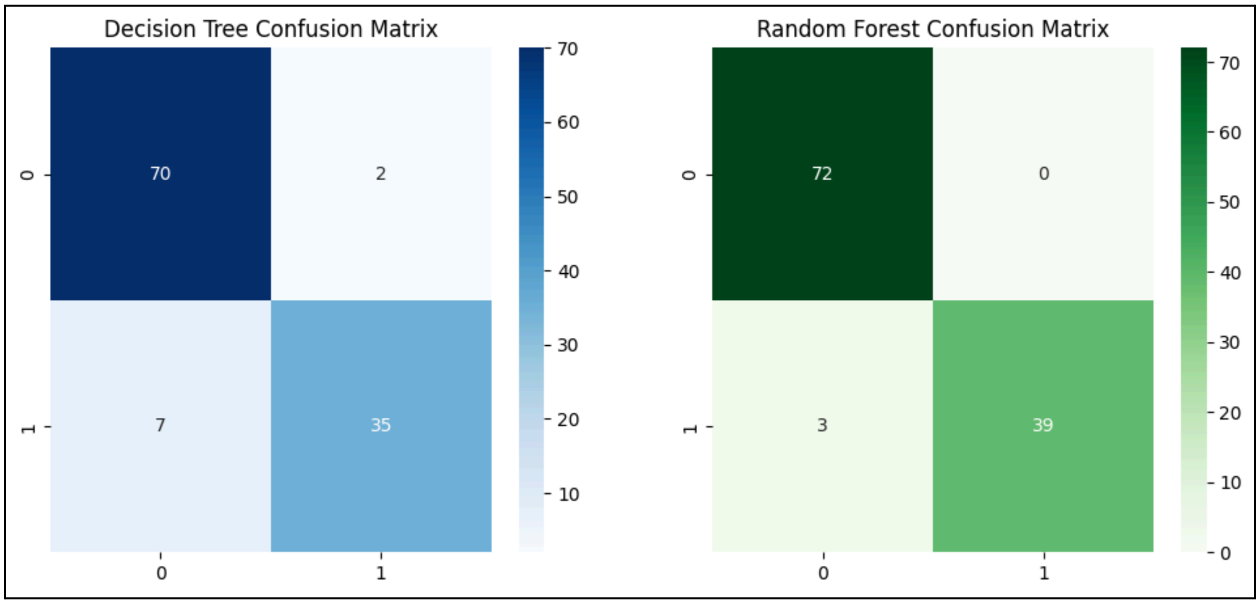
OUTPUT:

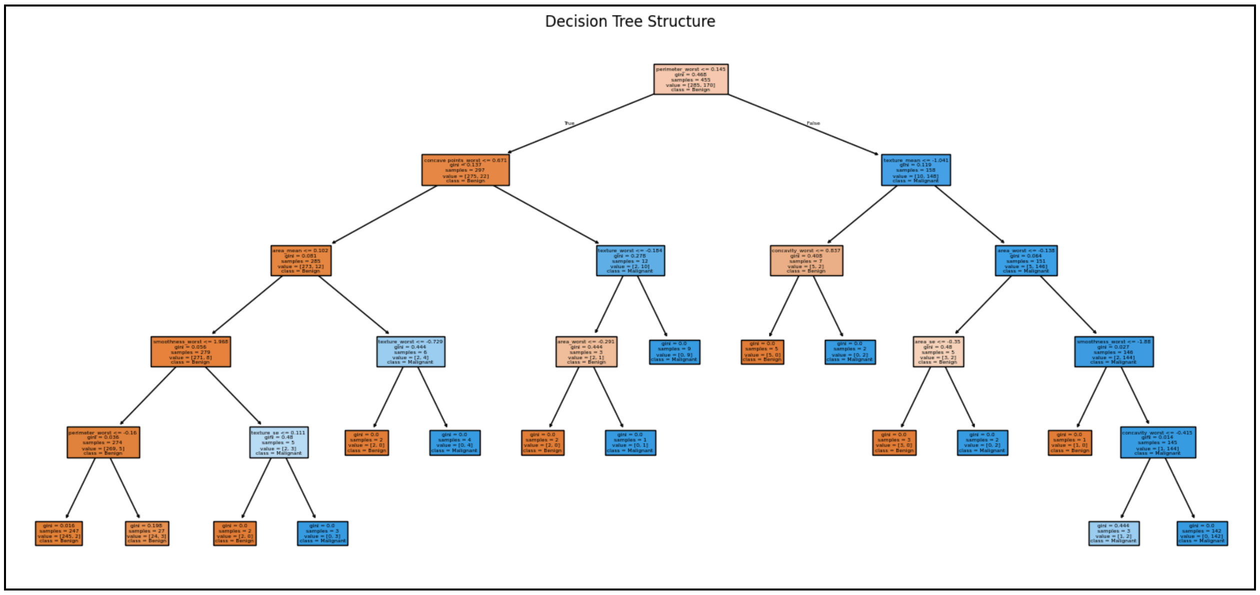
Code:

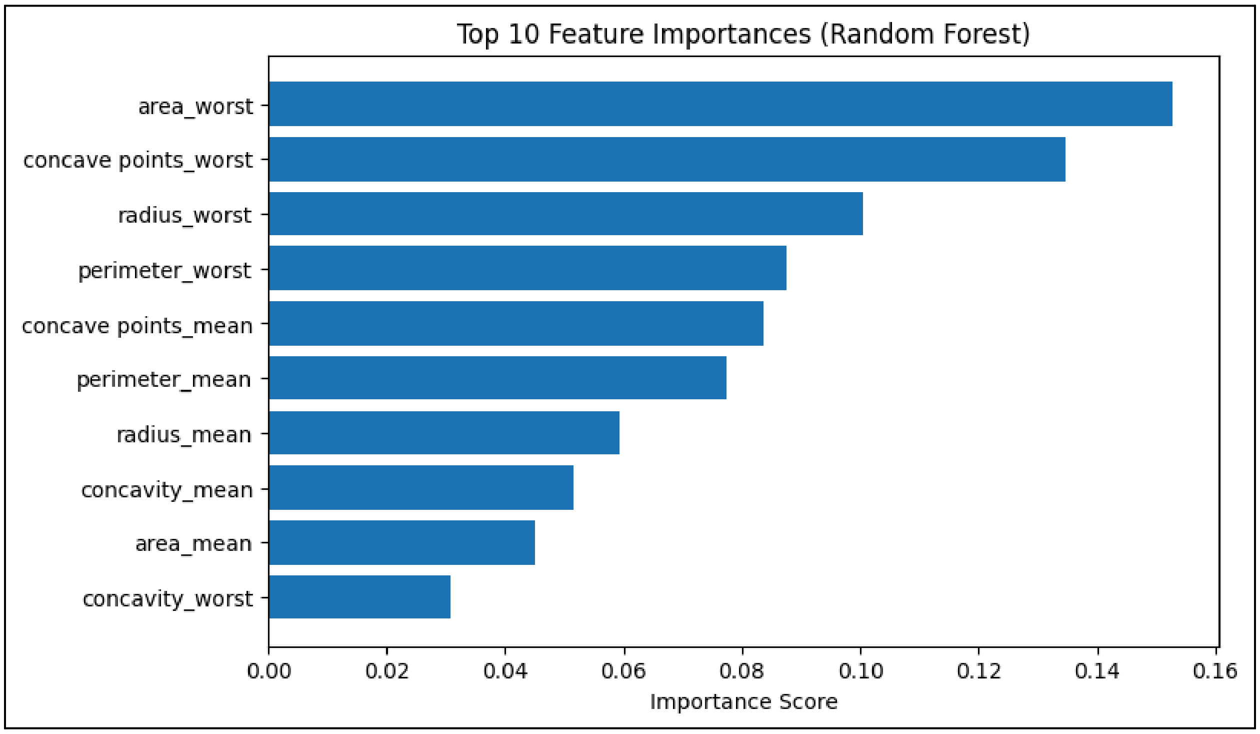
```
!pip install kagglehub -q

import kagglehub
from kagglehub import
KaggleDatasetAdapter

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.tree import
DecisionTreeClassifier, plot_tree
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import
accuracy_score, confusion_matrix,
classification_report

df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,

"uciml/breast-cancer-wisconsin-data"
,
    "data.csv"
)

print("Dataset Shape:", df.shape)
df.head()
df.drop(columns=["id", "Unnamed:
32"], inplace=True)
df["diagnosis"] =
df["diagnosis"].map({"M": 1, "B":
0})

X = df.drop("diagnosis", axis=1)
y = df["diagnosis"]

X_train, X_test, y_train, y_test =
train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled =
scaler.transform(X_test)

dt =
DecisionTreeClassifier(max_depth=5,
random_state=42)
dt.fit(X_train_scaled, y_train)
dt_preds = dt.predict(X_test_scaled)

rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=5,
    random_state=42
)
rf.fit(X_train_scaled, y_train)
rf_preds = rf.predict(X_test_scaled)

print("\n=== Decision Tree
Performance ===")
print("Accuracy:",
accuracy_score(y_test, dt_preds))
print(classification_report(y_test,
dt_preds))

print("\n=== Random Forest
Performance ===")
print("Accuracy:",
accuracy_score(y_test, rf_preds))
print(classification_report(y_test,
rf_preds))

fig, axes = plt.subplots(1, 2,
figsize=(12, 5))

sns.heatmap(confusion_matrix(y_test,
dt_preds),
            annot=True, fmt="d",
cmap="Blues", ax=axes[0])
axes[0].set_title("Decision Tree
Confusion Matrix")

sns.heatmap(confusion_matrix(y_test,
rf_preds),
            annot=True, fmt="d",
cmap="Greens", ax=axes[1])
axes[1].set_title("Random Forest
Confusion Matrix")

plt.show()

plt.figure(figsize=(18, 8))
plot_tree(
    dt,
    feature_names=X.columns,
    class_names=["Benign",
"Malignant"],
    filled=True
)
plt.title("Decision Tree Structure")
plt.show()

importances =
rf.feature_importances_
indices =
np.argsort(importances)[-10:]

plt.figure(figsize=(8, 5))
plt.barh(X.columns[indices],
importances[indices])
plt.title("Top 10 Feature
Importances (Random Forest)")
plt.xlabel("Importance Score")
plt.show()
```



```
=== Decision Tree Performance ===
Accuracy: 0.9210526315789473
              precision    recall  f1-score   support

           0       0.91      0.97      0.94        72
           1       0.95      0.83      0.89        42

    accuracy                           0.92       114
   macro avg       0.93      0.90      0.91       114
weighted avg       0.92      0.92      0.92       114


=== Random Forest Performance ===
Accuracy: 0.9736842105263158
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```







Conclusion

This experiment demonstrated the effectiveness of tree-based ensemble methods for real-world medical classification tasks. While Decision Trees offer transparency and ease of interpretation, Random Forests provide superior predictive performance and robustness. Such models are highly suitable for healthcare decision-support systems where both accuracy and reliability are critical.