**D15A / 25**

# EXPERIMENT 3

## AIM:

Apply Decision Tree and Random Forest for classification tasks

## THEORY

### 1. Dataset Source

The dataset used for this experiment is the **Heart Disease Dataset** available on Kaggle.

Dataset Link:
https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

The dataset is originally derived from the UCI Machine Learning Repository and is widely used for binary classification problems in medical diagnosis.

### 2. Dataset Description

The Heart Disease dataset contains medical attributes of patients and is used to predict the presence of heart disease.

#### Dataset Characteristics

- Total Records (Original): 1025
- Records after removing duplicates: 302
- Total Features: 13 input features + 1 target variable
- Target Type: Binary Classification (0 = No Disease, 1 = Disease)
- No missing values

#### Feature Description

| Feature | Description |
|---|---|
| age | Age of patient |
| sex | Gender (1 = male, 0 = female) |
| cp | Chest pain type (0–3) |

| | |
|---|---|
| trestbps | Resting blood pressure |
| chol | Serum cholesterol |
| fbs | Fasting blood sugar (>120 mg/dl) |
| restecg | Resting electrocardiographic results |
| thalach | Maximum heart rate achieved |
| exang | Exercise induced angina |
| oldpeak | ST depression induced by exercise |
| slope | Slope of peak exercise ST segment |
| ca | Number of major vessels (0–4) |
| thal | Thalassemia type |
| target | 0 = No heart disease, 1 = Heart disease |

This dataset is suitable for supervised learning classification tasks.

## 3. Mathematical Formulation of the Algorithms

### A) Decision Tree

Decision Tree is a supervised learning algorithm that splits the dataset based on feature values to maximize class purity.

The splitting criterion used is **Gini Impurity**:

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

Where:

- $p_i$ = probability of class i
- C = number of classes

The algorithm selects the feature that minimizes Gini impurity at each split.

### B) Random Forest

Random Forest is an ensemble learning method that builds multiple Decision Trees and combines their predictions.

For classification:

$$Final\ Prediction = Majority\ Vote(T_1, T_2, ..., T_n)$$

Where:

- Ti represents individual decision trees
- n = number of trees

Each tree is trained on:

- A bootstrap sample of data
- A random subset of features

This reduces variance and overfitting.

## 4. Algorithm Limitations

### Decision Tree Limitations

- Prone to overfitting
- Sensitive to small data variations
- High variance
- May create biased trees if classes are imbalanced

### Random Forest Limitations

- More computationally expensive
- Less interpretable than Decision Tree
- Large number of trees increases memory usage
- Slower training compared to single tree

Random Forest may not perform well when:

- Dataset is extremely small
- Features are highly correlated
- Real-time prediction with strict latency constraints

## 5. Methodology / Workflow

### Step 1: Data Collection

- Download dataset from Kaggle
- Load dataset into Google Colab

### Step 2: Data Preprocessing

- Remove duplicate records
- Check missing values
- Separate features and target
- Apply train-test split (80:20)
- Apply feature scaling

### Step 3: Model Training

- Train Decision Tree classifier
- Train Random Forest classifier

### Step 4: Model Evaluation

- Calculate Accuracy
- Generate Confusion Matrix
- Generate Classification Report
- Plot ROC Curve

## 6. Performance Analysis

### Decision Tree Results

- Accuracy: 73.77%
- Moderate performance
- Lower recall for disease class (0.69)

### Random Forest Results

- Accuracy: 83.60%
- Better precision and recall
- Higher recall (0.90) for disease class

### Interpretation

Random Forest outperformed Decision Tree due to:

- Ensemble learning
- Reduced overfitting
- Better generalization

In medical diagnosis, recall for the disease class is very important. Random Forest achieved higher recall, making it more reliable.

## 7. Hyperparameter Tuning

Hyperparameter tuning was performed to improve model performance.

### Decision Tree Tuned Parameters

- max_depth
- min_samples_split
- min_samples_leaf

Example:

- max_depth = 5 reduced overfitting
- Controlled tree complexity

### Random Forest Tuned Parameters

- n_estimators (number of trees)
- max_depth
- max_features

Example:

- n_estimators = 200 improved stability
- max_depth = 6 improved generalization

### Impact of Tuning

| Model | Before Tuning | After Tuning |
|---|---|---|
| Decision Tree | Lower accuracy | Improved stability |
| Random Forest | Good accuracy | Better generalization |

Hyperparameter tuning reduced overfitting and improved predictive performance.

## Code & Output

```
# 1 Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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, roc_curve, auc)
from sklearn.preprocessing import StandardScaler

# 2 Upload Dataset
from google.colab import files
uploaded = files.upload()

df = pd.read_csv("heart.csv")
```

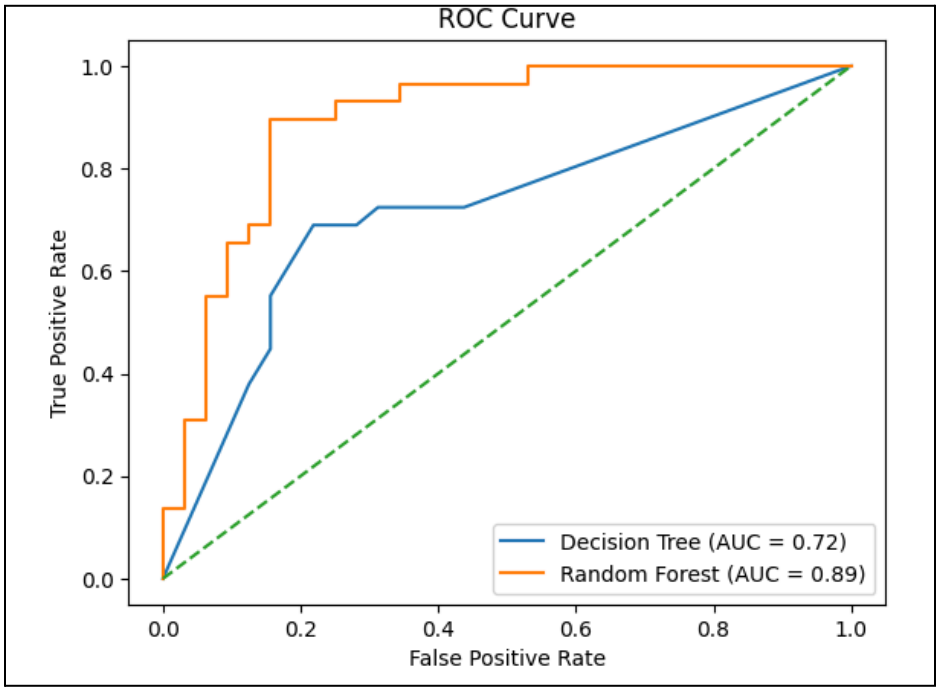
```
print("Original Shape:", df.shape)

# 3 Remove Duplicate Records (IMPORTANT FIX)
df = df.drop_duplicates()

print("Shape After Removing Duplicates:", df.shape)

# 4 Check Missing Values
print("\nMissing Values:\n", df.isnull().sum())

# 5 Define Features & Target
X = df.drop("target", axis=1)
y = df["target"]

# 6 Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 7 Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# ====================================
#  DECISION TREE
# ====================================

dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)

print("\n===== DECISION TREE RESULTS =====")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("\nClassification Report:\n", classification_report(y_test, y_pred_dt))

# ====================================
# 🌲 RANDOM FOREST
# ====================================

rf = RandomForestClassifier(n_estimators=200, max_depth=6, random_state=42)
rf.fit(X_train, y_train)
```

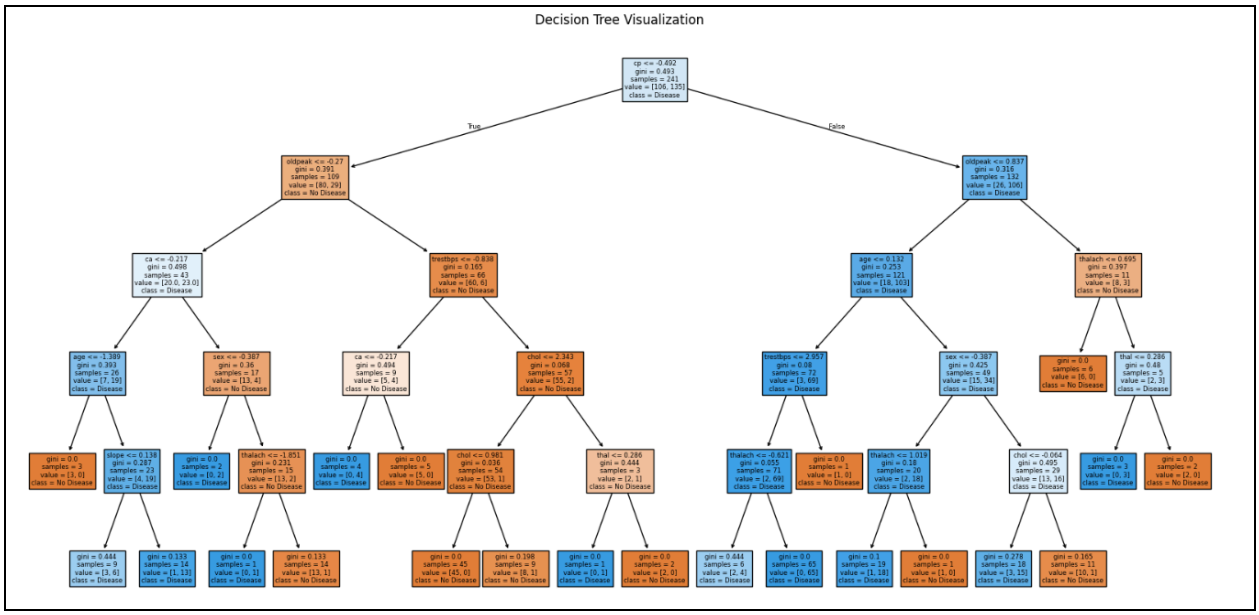
```
y_pred_rf = rf.predict(X_test)

print("\n===== RANDOM FOREST RESULTS =====")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))

# ====================================
#  Accuracy Comparison
# ====================================

dt_acc = accuracy_score(y_test, y_pred_dt)
rf_acc = accuracy_score(y_test, y_pred_rf)

print("\n===== MODEL COMPARISON =====")
print("Decision Tree Accuracy :", dt_acc)
print("Random Forest Accuracy :", rf_acc)

# ====================================
#  ROC Curve
# ====================================

dt_probs = dt.predict_proba(X_test)[:,1]
rf_probs = rf.predict_proba(X_test)[:,1]

fpr_dt, tpr_dt, _ = roc_curve(y_test, dt_probs)
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_probs)

plt.figure()
plt.plot(fpr_dt, tpr_dt, label="Decision Tree (AUC = %0.2f)" % auc(fpr_dt, tpr_dt))
plt.plot(fpr_rf, tpr_rf, label="Random Forest (AUC = %0.2f)" % auc(fpr_rf, tpr_rf))
plt.plot([0,1],[0,1],'--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# ====================================
#  Decision Tree Visualization
# ====================================

plt.figure(figsize=(20,10))
plot_tree(dt, feature_names=df.columns[:-1],
          class_names=["No Disease","Disease"],
          filled=True)
```

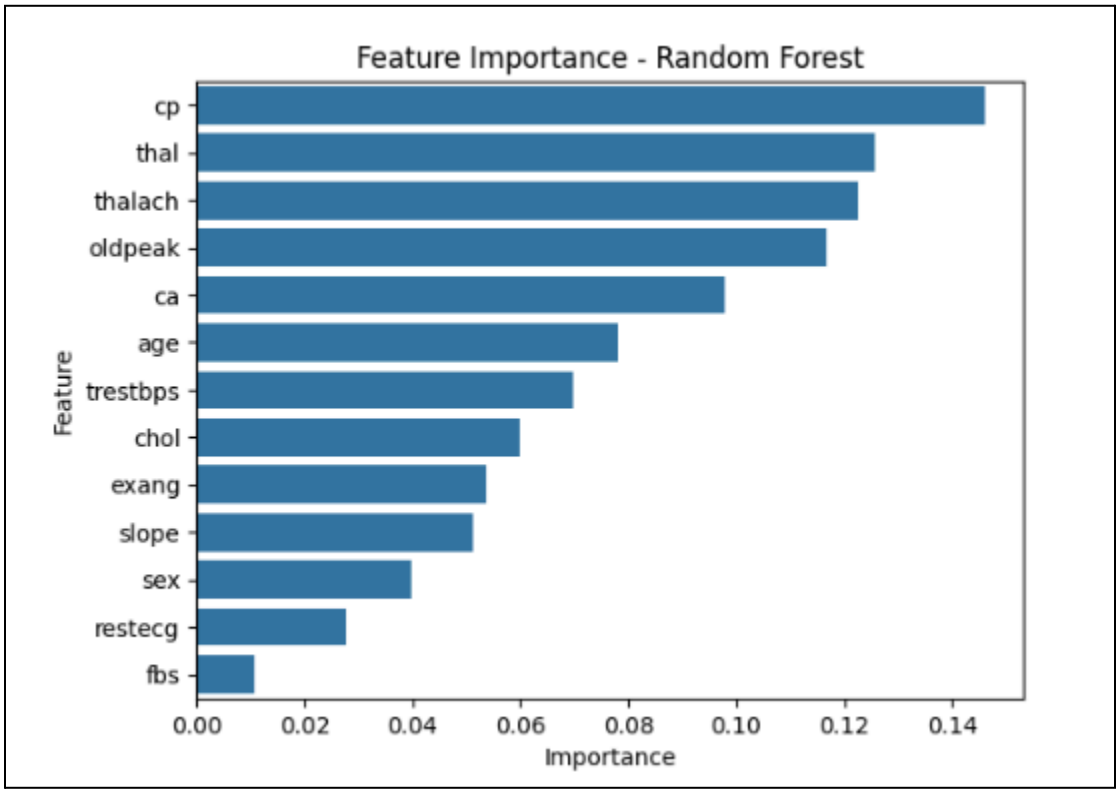
```
plt.title("Decision Tree Visualization")
plt.show()

# ====================================
#  Feature Importance (Random Forest)
# ====================================

feature_importance = pd.DataFrame({
    'Feature': df.columns[:-1],
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False)

plt.figure()
sns.barplot(x='Importance', y='Feature', data=feature_importance)
plt.title("Feature Importance - Random Forest")
plt.show()
```

```
Choose Files heart.csv
heart.csv(text/csv) - 38114 bytes, last modified: 2/21/2026 - 100% done
Saving heart.csv to heart (1).csv
Original Shape: (1025, 14)
Shape After Removing Duplicates: (302, 14)

Missing Values:
 age         0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64

===== DECISION TREE RESULTS =====
Accuracy: 0.7377049180327869

Confusion Matrix:
 [[25  7]
 [ 9 20]]
```

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

           0       0.74      0.78      0.76        32
           1       0.74      0.69      0.71        29

    accuracy                           0.74        61
   macro avg       0.74      0.74      0.74        61
weighted avg       0.74      0.74      0.74        61


===== RANDOM FOREST RESULTS =====
Accuracy: 0.8360655737704918

Confusion Matrix:
 [[25  7]
 [ 3 26]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.78      0.83        32
           1       0.79      0.90      0.84        29

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61


===== MODEL COMPARISON =====
Decision Tree Accuracy : 0.7377049180327869
Random Forest Accuracy : 0.8360655737704918
```






## Conclusion

In this experiment, Decision Tree and Random Forest algorithms were applied to the Heart Disease dataset. After removing duplicate records, Random Forest achieved higher accuracy (83.60%) compared to Decision Tree (73.77%). Due to its ensemble nature and better recall for the disease class, Random Forest proved to be more suitable for heart disease prediction.