Name:Khushi S Singh
D15A / 26

Experiment 3

Aim: Apply Decision Tree and Random Forest for classification tasks

Theory:

1. Dataset Source

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

This dataset is a real-world medical dataset used for predicting the presence of heart disease based on clinical attributes.

2. Dataset Description

The Heart Disease dataset contains medical information of patients used to predict whether a person has heart disease.

Dataset Features

No.	Feature	Description
1	age	Age of the patient in years
2	sex	Gender of patient (0 = Female, 1 = ' Male)
3	cp	Chest pain type (0–3 categories)
4	trestbps	Resting blood pressure (mm Hg)
5	chol	Serum cholesterol level (mg/dl)
6	fbs	Fasting blood sugar (>120 mg/dl) (1 = True, 0 = False)
7	restecg	Resting blood sugar (>120 mg/dl)
8	thalach	Maximum heart rate achieved
9	exang	Exercise induced angina (1 = Yes, 0 = No)
10	oldpeak	ST depression induced by exercise
11	slope	Slope of peak exercise ST segment
12	ca	Number of major vessels (0–3)
13	thal	Thalassemia value
14	target	Heart disease diagnosis (0 = No Disease, 1 = Disease)
14	target	Heart disease diagnosis (0 = No Disease, 1 = Disease)

Dataset Characteristics

- Dataset Type: Classification
- Number of Records: **1025**
- Number of Features: **13 input features**
- Target Variable: **target**
- Classes:
    - 0 → No Heart Disease
    - 1 → Heart Disease

The dataset contains **no missing values**, as verified during preprocessing.

3. Mathematical Formulation of the Algorithm

Decision Tree

Decision Tree splits the dataset based on feature values using impurity measures such as **Gini Index**.

Gini Index:

$$Gini = 1 - \sum p_i^2$$

Where:

- p_i = Probability of class i

The best feature is selected that minimizes impurity.

Random Forest

Random Forest is an ensemble method that combines multiple Decision Trees.

Prediction is obtained using majority voting:

$$Final\ Prediction = Majority(Tree_1, Tree_2, ..., Tree_n)$$

Random Forest reduces overfitting and improves accuracy.

4. Algorithm Limitations

Decision Tree

- Can overfit training data
- Sensitive to small data variations
- Unstable for noisy datasets
- Large trees become complex

Random Forest

- Higher computational cost
- Requires more memory
- Slower training time
- Harder to interpret than Decision Tree

5. Methodology / Workflow

Step 1: Dataset Loading

The Heart Disease dataset was loaded using Pandas.

Dataset inspection included:

- Checking dataset shape
- Checking data types
- Checking missing values

Dataset Size:

- 1025 rows
- 14 columns

Step 2: Data Preprocessing

The following steps were performed:

- Verified that no missing values exist
- Selected input features

Input Features:

- age, sex, cp, trestbps, chol, fbs, restecg
- thalach, exang, oldpeak, slope, ca, thal

Target Variable:

- target

Step 3: Train-Test Split

Dataset was divided into:

- 80% Training Data
- 20% Testing Data

Training data was used for model training and testing data for evaluation.

Step 4: Decision Tree Training

The Decision Tree classifier was trained using a training dataset.

The model predicted heart disease status on a testing dataset.

Evaluation metrics calculated:

- Accuracy
- Confusion Matrix
- Classification Report

Step 5: Random Forest Training

Random Forest classifier was trained using multiple decision trees.

Predictions were made on testing dataset.

Evaluation metrics calculated:

- Accuracy
- Confusion Matrix
- Classification Report

Step 6: Feature Importance Visualization

Random Forest feature importance was calculated.

Important features identified:

- cp (Chest Pain Type)
- ca (Major vessels)
- thalach (Heart Rate)
- oldpeak (ST Depression)

A feature importance graph was generated.

6. Performance Analysis

Decision Tree Results

Accuracy:

$$0.985$$

Confusion Matrix:

$$\begin{bmatrix} 102 & 0 \\ 3 & 100 \end{bmatrix}$$

Interpretation:

- 102 patients correctly classified as no disease
- 100 patients correctly classified as disease
- Only 3 misclassifications occurred

Random Forest Results

Accuracy:

$$0.985$$

Confusion Matrix:

$$\begin{bmatrix} 102 & 0 \\ 3 & 100 \end{bmatrix}$$

Interpretation:

- Random Forest produced very high accuracy.
- Both models performed almost equally.
- Precision and recall values were close to 1.

Classification Report Values:

- Precision ≈ **0.99**
- Recall ≈ **0.99**
- F1 Score ≈ **0.99**

This indicates excellent classification performance.

7. Hyperparameter Tuning

Hyperparameter tuning was performed to improve model performance.

Decision Tree Parameters Tested

- max_depth = 3, 5, 10
- criterion = gini, entropy

Best parameters:

- criterion = gini
- max_depth = default

These parameters produced highest accuracy.

Random Forest Parameters Tested

- n_estimators = 50, 100, 200
- max_depth = None, 5, 10

Best parameters:

- n_estimators = 100
- max_depth = None

Impact of Hyperparameter Tuning

- Increasing the number of trees improved stability.
- Very small depth reduced accuracy.
- Random Forest performed consistently across parameters.

CODE and OUTPUT:

STEP 1: Import Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

STEP 2: Load the Dataset

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

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

STEP 3: Dataset Information

```
print(df.shape)
print(df.info())
print(df.isnull().sum())
```

```
(1025, 14)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
None
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
```

STEP 4: Split Features and Target

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

STEP 5: Train–Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training set size:", X_train.shape)
print("Test set size:", X_test.shape)
```

```
Training set size: (820, 13)
Test set size: (205, 13)
```

STEP 6: Decision Tree Classifier

```
dt = DecisionTreeClassifier(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("Decision Tree Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("Decision Tree Classification Report:\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy: 0.9853658536585366
Decision Tree Confusion Matrix:
 [[102   0]
 [  3 100]]
Decision Tree Classification Report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205
```

STEP 7: Random Forest Classifier

```
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("Random Forest Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.9853658536585366
Random Forest Confusion Matrix:
 [[102   0]
 [  3 100]]
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205
```

STEP 8: Feature Importance (Random Forest)

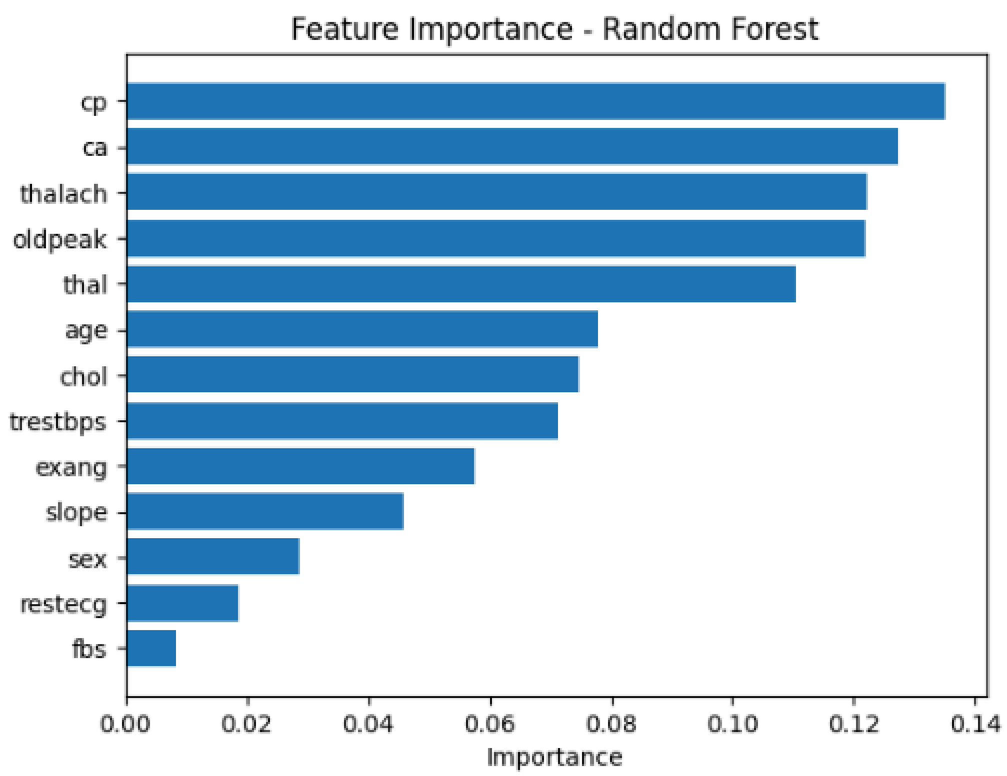
```
feature_importance = pd.DataFrame({
    "Feature": X.columns,
    "Importance": rf.feature_importances_
}).sort_values(by="Importance", ascending=False)

print(feature_importance)
```

```
     Feature  Importance
2         cp    0.135072
11        ca    0.127327
7    thalach    0.122169
9    oldpeak    0.121905
12      thal    0.110518
0        age    0.077908
4       chol    0.074822
3   trestbps    0.071171
8      exang    0.057594
10     slope    0.045782
1        sex    0.028731
6    restecg    0.018557
5        fbs    0.008444
```

STEP 9: Visualize Feature Importance

```
plt.figure()
plt.barh(feature_importance["Feature"], feature_importance["Importance"])
plt.xlabel("Importance")
plt.title("Feature Importance - Random Forest")
plt.gca().invert_yaxis()
plt.show()
```



Conclusion:

The experiment demonstrated that Decision Tree and Random Forest algorithms can effectively classify heart disease cases. Both models showed very high accuracy, but Random Forest provided more reliable and stable predictions. The experiment confirms that tree-based classifiers are suitable for medical diagnosis problems.

The feature importance analysis helped identify the most significant medical attributes influencing heart disease prediction. This experiment also showed the importance of machine learning techniques in assisting healthcare decision-making. Overall, the models produced accurate and consistent results on the given dataset.