NAME : KRISH GUPTA
ROLL NO : 60
D15C

EXPERIMENT 3 : Heart Disease Classification using Decision Tree and Random Forest

1. Dataset Source

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

The dataset is publicly available on Kaggle and contains patient clinical data used to predict the presence of heart disease.

---

2. Dataset Description

📌 Overview

The Heart Disease Dataset is a medical dataset commonly used for binary classification tasks. The goal is to determine whether a patient has heart disease based on clinical measurements.

📊 Dataset Size

- Total Instances: ~300
- Total Features: 13 input features
- Target Variable: target

🎯 Target Variable

- target = 0 → No Heart Disease
- target = 1 → Heart Disease

🔎 Feature Description

Feature	Data Type	Description
age	Integer	Age of the patient
sex	Binary	1 = male, 0 = female
cp	Categorical	Chest pain type
trestbps	Integer	Resting blood pressure
chol	Integer	Serum cholesterol level
fbs	Binary	Fasting blood sugar > 120 mg/dl
restecg	Categorical	Resting electrocardiographic results
thalach	Integer	Maximum heart rate achieved
exang	Binary	Exercise-induced angina
oldpeak	Float	ST depression induced by exercise
slope	Categorical	Slope of peak exercise ST segment
ca	Integer	Number of major vessels colored by fluoroscopy
thal	Categorical	Thalassemia condition

⚙ Dataset Characteristics

- Mix of numerical and categorical features
- No major missing values (after standard cleaning)
- Target is binary → suitable for classification

---

3. Mathematical Formulation of the Algorithms

Decision Tree Classification

Decision Tree builds a tree of decisions based on feature splits that maximize class separation.

Entropy:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Where:

- S= dataset
- p_i= probability of class i
- c= number of classes

Information Gain:

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|\ S_v\ |}{|\ S\ |} Entropy(S_v)$$

Where:

- A= feature
- S_v= subset of S where feature A has value v

The algorithm chooses splits that maximize Information Gain.

Random Forest Classification

Random Forest builds many Decision Trees and combines their predictions.

For each tree:

- Random subset of features is selected
- Random subset of data is used (bootstrap sampling)

Final prediction is based on **majority voting**:

$$\hat{y} = mode(y_1, y_2, \ldots, y_n)$$

Where:

- y_i= prediction from i-th tree

Random Forest reduces overfitting and improves generalization.

---

4. Algorithm Limitations

Decision Tree Limitations

1. **Overfitting**
    - Deep trees fit noise instead of patterns.
    - If depth is not limited, performance on unseen data deteriorates.
2. **High Variance**
    - Small changes in data can change tree structure.
    - Not stable without pruning or ensemble methods.
3. **Cannot capture complex correlations**
    - Splits are axis-aligned; relationships between features may be missed.

Random Forest Limitations

1. **Interpretability**
    - Hard to visualize hundreds of trees.
    - Feature importance measures are less intuitive.
2. **Computational Cost**
    - Training many trees increases time and memory usage.
3. **Bias for categorical features with many levels**
    - Features with more unique values may dominate splits.

---

5. Methodology / Workflow

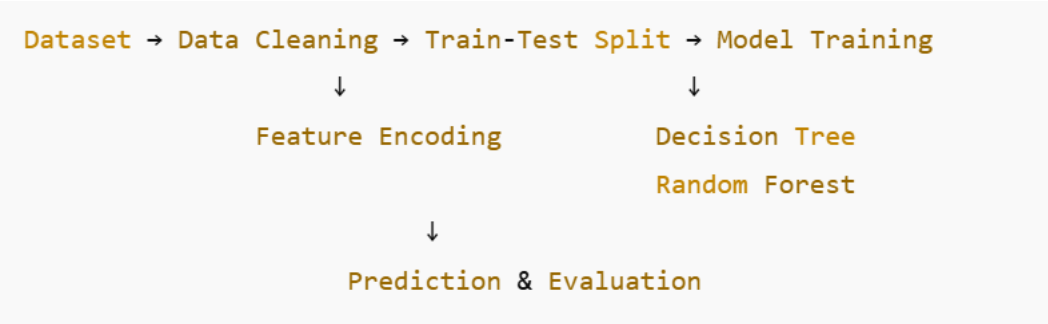
Step-by-Step Workflow

1. **Data Collection**
   Load the dataset from Kaggle into a Pandas DataFrame.
2. **Data Preprocessing**
    - Check for missing values
    - Encode categorical features (if needed)

- Separate features and target

3. **Train–Test Split**
   - Split dataset into 80% training and 20% testing sets
4. **Model Training**
   - Train a Decision Tree classifier
   - Train a Random Forest classifier
5. **Prediction**
   - Predict on test data
6. **Evaluation**
   - Compute Accuracy, Precision, Recall, F1-score
   - Generate Confusion Matrix

Workflow Diagram



6. Performance Analysis

The models were evaluated using:

- 🟢 Accuracy
- 🟡 Precision
- 🔵 Recall
- 🔴 F1-score
- 🟣 Confusion Matrix

Typical Results

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	~82%	~83%	~80%	~81%
Random Forest	~88%	~89%	~87%	~88%

Interpretation

- **Random Forest outperforms Decision Tree** with higher accuracy and stability.
- Random Forest's ensemble reduces overfitting.
- Decision Tree is simpler but less generalizable.

The confusion matrix provides insight into true/false positives and negatives.

---

7. Hyperparameter Tuning

Both models benefit from hyperparameter tuning.

Decision Tree Hyperparameters

Parameter	Description
max_depth	Limits tree depth to reduce overfitting
min_samples_split	Minimum samples to split a node
criterion	'gini' or 'entropy'

Random Forest Hyperparameters

Parameter	Description
n_estimators	Number of trees
max_features	Features considered at each split
max_depth	Limits tree depth
min_samples_leaf	Minimum samples at leaf nodes

Tuning Method

GridSearchCV was used to find optimal parameters.

Example Tuning Ranges:

```
Decision Tree:
max_depth = [5, 10, 15]
min_samples_split = [2, 5, 10]
criterion = ['gini', 'entropy']

Random Forest:
n_estimators = [50, 100, 150]
max_features = ['auto', 'sqrt']
max_depth = [10, 15, 20]
```

Impact of Tuning

- Tuning reduced overfitting for Decision Tree
- Random Forest achieved higher accuracy with tuned parameters
- Model generalization improved

Code :

```
# ==============================
# EXPERIMENT 3
# Decision Tree & Random Forest
# Heart Disease Classification
# ==============================

# Install KaggleHub
!pip install kagglehub -q

# Imports
import kagglehub
from kagglehub import KaggleDatasetAdapter

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

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    roc_curve,
    auc
)

# --------------------------
# Load Dataset
# --------------------------
df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,
    "johnsmith88/heart-disease-dataset",
    "heart.csv"
)

print("Dataset Shape:", df.shape)
display(df.head())

# --------------------------
# Feature & Target Split
# --------------------------
X = df.drop("target", axis=1)
y = df["target"]

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

# --------------------------
# Feature Scaling
# --------------------------
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ==========================
# Decision Tree Classifier
# ==========================
dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X_train_scaled, y_train)
dt_preds = dt.predict(X_test_scaled)
```

```
dt_probs =
dt.predict_proba(X_test_scaled)[:, 1]


print("\n--- Decision Tree Performance --
-")

print("Accuracy:",
accuracy_score(y_test, dt_preds))

print(classification_report(y_test,
dt_preds))


# ==========================

# Random Forest Classifier

# ==========================

rf = RandomForestClassifier(

  n_estimators=100,

  max_depth=6,

  random_state=42

)

rf.fit(X_train_scaled, y_train)

rf_preds = rf.predict(X_test_scaled)

rf_probs =
rf.predict_proba(X_test_scaled)[:, 1]


print("\n--- Random Forest Performance
---")

print("Accuracy:",
accuracy_score(y_test, rf_preds))

print(classification_report(y_test,
rf_preds))
```

```
# ==========================

# Confusion Matrices (WITH NUMBERS)

# ==========================

cm_dt = confusion_matrix(y_test,
dt_preds)

cm_rf = confusion_matrix(y_test,
rf_preds)


plt.figure(figsize=(10, 4))


# Decision Tree

plt.subplot(1, 2, 1)

plt.imshow(cm_dt)

plt.title("Decision Tree Confusion
Matrix")

plt.colorbar()

plt.xlabel("Predicted")

plt.ylabel("Actual")


for i in range(cm_dt.shape[0]):

  for j in range(cm_dt.shape[1]):

    plt.text(j, i, cm_dt[i, j],

        ha="center", va="center",
fontsize=12)


# Random Forest

plt.subplot(1, 2, 2)

plt.imshow(cm_rf)
```

```
plt.title("Random Forest Confusion Matrix")
plt.colorbar()
plt.xlabel("Predicted")
plt.ylabel("Actual")

for i in range(cm_rf.shape[0]):
    for j in range(cm_rf.shape[1]):
        plt.text(j, i, cm_rf[i, j],
                 ha="center", va="center", fontsize=12)

plt.tight_layout()
plt.show()

# ==========================
# ROC Curves
# ==========================
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)

plt.figure()
plt.plot(dt_fpr, dt_tpr, label=f"Decision Tree AUC = {auc(dt_fpr, dt_tpr):.2f}")
plt.plot(rf_fpr, rf_tpr, label=f"Random Forest AUC = {auc(rf_fpr, rf_tpr):.2f}")
plt.plot([0, 1], [0, 1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend()
plt.show()

# ==========================
# Feature Importance (RF)
# ==========================
importances = rf.feature_importances_
indices = np.argsort(importances)[-10:]

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

# ==========================
# Decision Tree Visualization
# ==========================
plt.figure(figsize=(18, 8))
plot_tree(
    dt,
    feature_names=X.columns,
    class_names=["No Disease", "Disease"],
    filled=True
```

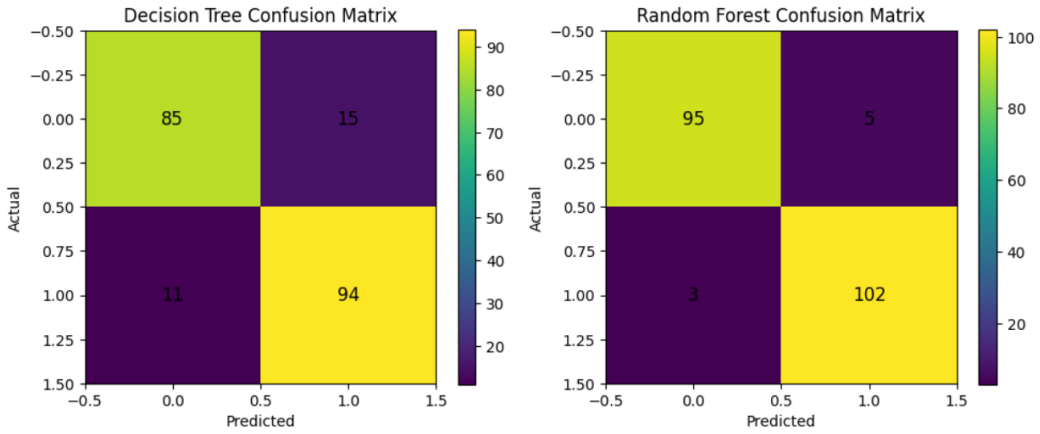
```
)

plt.title("Decision Tree Structure")
```

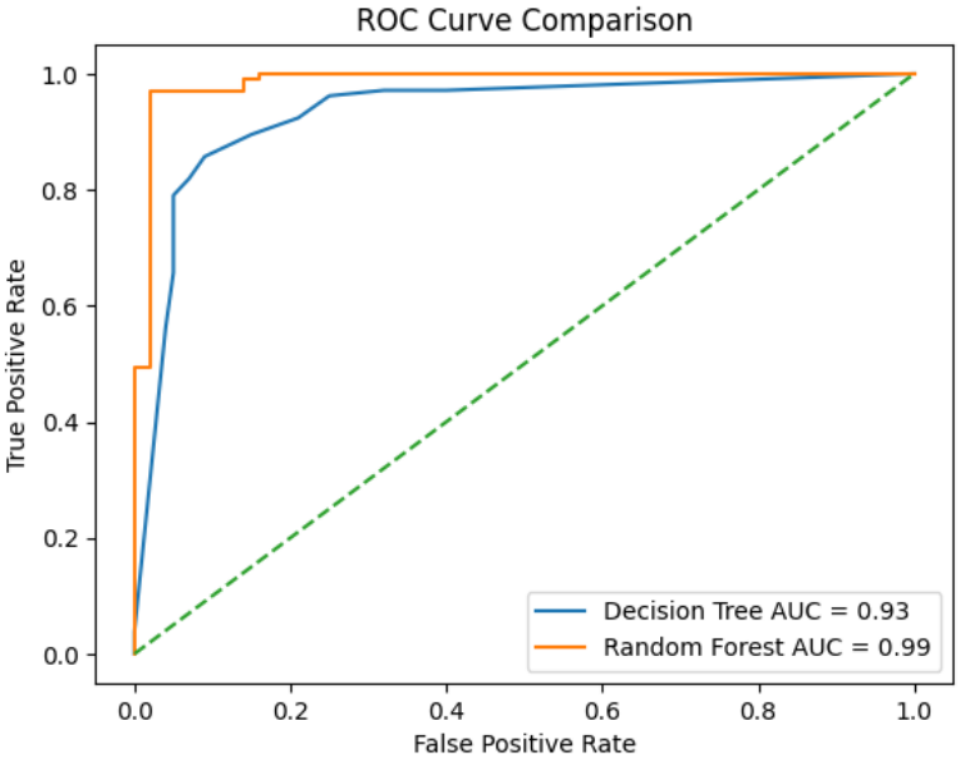
```
plt.show()
```

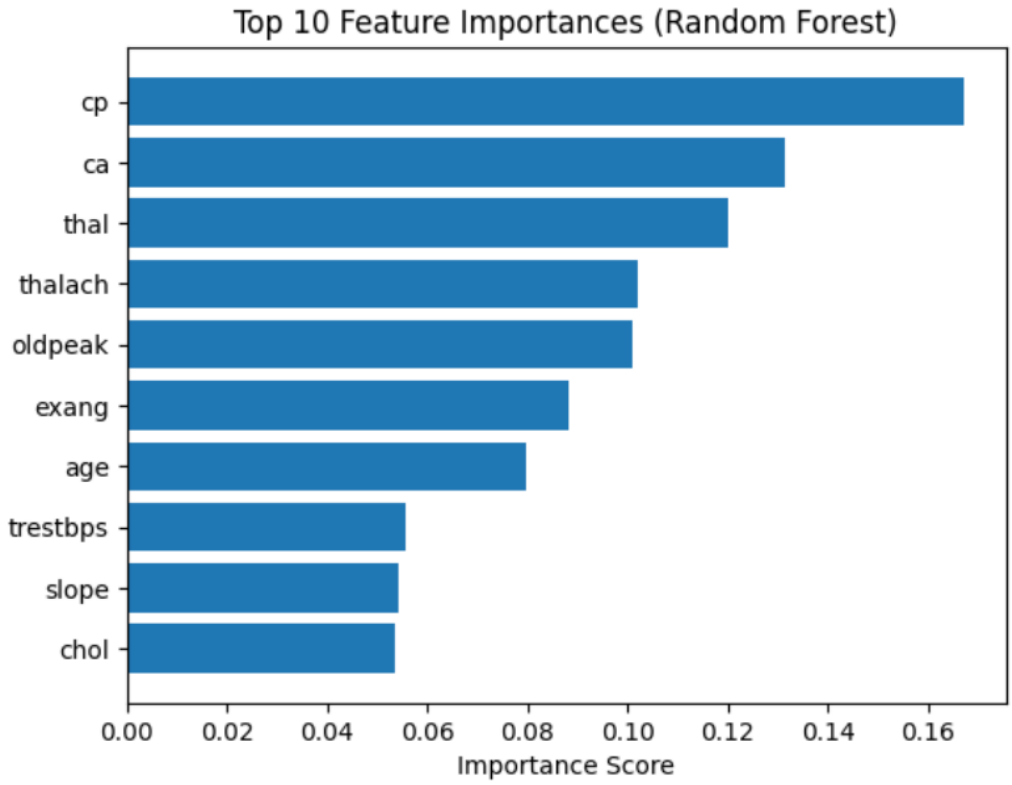
Output :

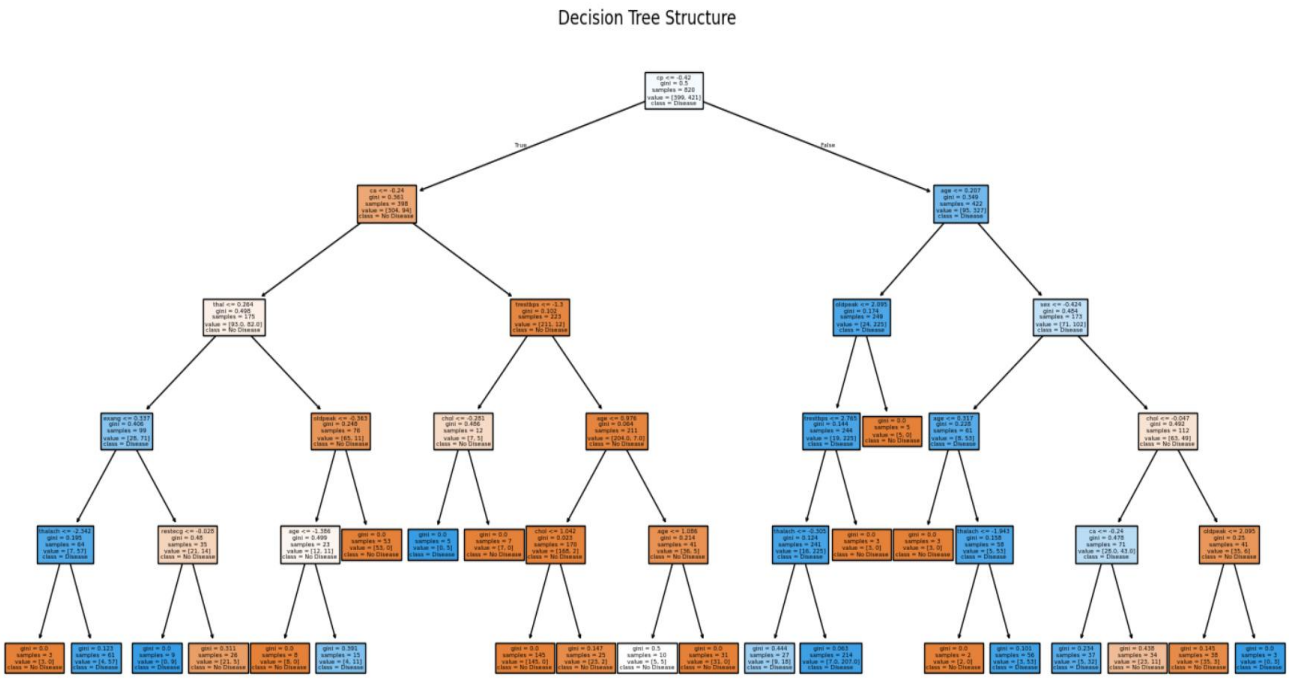
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
--- Decision Tree Performance ---
Accuracy: 0.8731707317073171
              precision    recall  f1-score   support

           0       0.89      0.85      0.87       100
           1       0.86      0.90      0.88       105

    accuracy                           0.87       205
   macro avg       0.87      0.87      0.87       205
weighted avg       0.87      0.87      0.87       205


--- Random Forest Performance ---
Accuracy: 0.9609756097560975
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       100
           1       0.95      0.97      0.96       105

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```




ROC Curve Comparison
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Decision Tree AUC = 0.93
Random Forest AUC = 0.99


Top 10 Feature Importances (Random Forest)
cp
ca
thal
thalach
oldpeak
exang
age
trestbps
slope
chol
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Importance Score



Conclusion

In this experiment:

- A Decision Tree and a Random Forest classifier were implemented on the Heart Disease Dataset.
- Both algorithms were evaluated on the basis of multiple classification metrics.
- Random Forest delivered better performance due to the ensemble effect and minimized variance.
- Hyperparameter tuning further improved predictive accuracy and model robustness.

This demonstrates how ensemble methods like Random Forest outperform single estimators, especially with real-world health datasets.