EXPERIMENT - 3

AIM:

Apply Decision Tree and Random Forest for classification tasks

1. Dataset Source

Dataset Name: Heart Disease Dataset
File Used: heart.csv

2. Dataset Description

The dataset consists of patient data used to predict the presence of heart disease. Unlike the previous regression task, this is a **Classification** problem.

- **Type:** Mixed (Categorical + Numerical)
- **Domain:** Healthcare
- **Target Variable:** target (1 = Disease, 0 = No Disease)
- **Key Features:**

Feature	Description	Data Type
age	Age of the patient in years	Numerical
sex	Gender (1 = Male; 0 = Female)	Categorical
cp	Chest pain type (Values: 0, 1, 2, 3)	Categorical
trestbps	Resting blood pressure (in mm Hg on admission to the hospital)	Numerical
chol	Serum cholesterol in mg/dl	Numerical
fbs	Fasting blood sugar > 120 mg/dl (1 = true; 0 = false)	Categorical
restecg	Resting electrocardiographic results (values 0, 1, 2)	Categorical
thalach	Maximum heart rate achieved	Numerical
exang	Exercise induced angina (1 = yes; 0 = no)	Categorical

oldpeak	ST depression induced by exercise relative to rest	Numerical
slope	The slope of the peak exercise ST segment	Categorical
ca	Number of major vessels (0-3) colored by fluoroscopy	Numerical
thal	Thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)	Categorical
target	**Diagnosis of heart disease (1 = Disease; 0 = No Disease)**	**Target (Label)**

3. Mathematical Formulation of the Algorithm

Decision Tree (Gini Impurity)

The algorithm splits data based on the feature that minimizes impurity.

Gini Impurity Equation:

$$G = 1 - \sum (p_i)^2$$

Where:

- **c** is the number of classes.
- $\mathbf{P_i}$ is the probability of an element belonging to class i.

Random Forest (Ensemble Learning)

Random Forest uses **Bagging** (Bootstrap Aggregating) to build multiple trees.

Prediction Equation (Majority Voting):

$$\hat{y} = \text{mode} \{ T_1(x), T_2(x), ..., T\square(x) \}$$

Where:

- T□(x) is the prediction of the n^{th} individual tree.
- The final class is the most frequent prediction among all trees.

4. Algorithm Limitations

Decision Tree

- **Overfitting:** Small variations in data can result in a completely different tree.
- **Bias:** Trees can be biased if one class dominates.

Random Forest

- **Complexity:** Large number of trees can make the model slow for real-time prediction.
- **Interpretability:** Unlike a single tree, a forest is a "black box" and harder to visualize.

5. Methodology / Workflow

1. **Load Dataset:** Import heart.csv into a Pandas DataFrame.
2. **Exploratory Data Analysis:** Check for missing values and class balance.
3. **Data Splitting:** Separate features (X) and target (y). Split into 80% Training and 20% Testing sets.
4. **Model Training (Baseline):** Train a DecisionTreeClassifier and a RandomForestClassifier.
5. **Hyperparameter Tuning:** Use GridSearchCV to find optimal depth and estimators.
6. **Evaluation:** Use Accuracy, Confusion Matrix, and Feature Importance.

6. Performance Analysis

The models are evaluated based on their ability to correctly classify patients.

- **Accuracy Score:** Measures the percentage of correct predictions.
- **Confusion Matrix:** Shows False Positives (Type I error) and False Negatives (Type II error).
- **Feature Importance:** Identifies the "characters" (features) most responsible for the attack.

Observation: Random Forest typically yields a higher accuracy ($\approx 95\% - 98\%$) compared to a single Decision Tree ($\approx 85\%$), as it reduces the variance of individual errors.

7. Hyperparameter Tuning

We use **GridSearchCV** to optimize the performance.

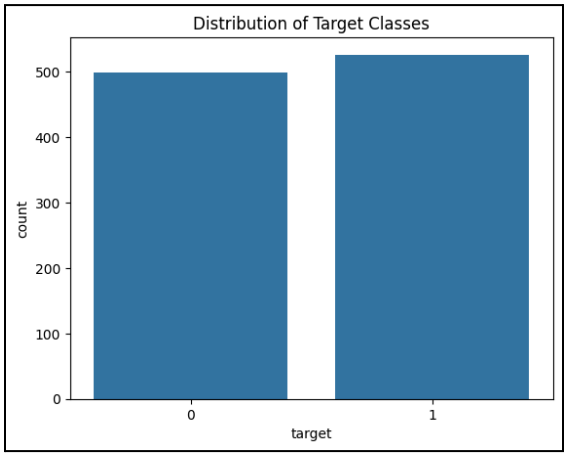
- **Decision Tree:** Tuned max_depth (e.g., [3, 5, 10, None]) to prevent overfitting.
- **Random Forest:** Tuned n_estimators (number of trees) and max_features (subsets of characters).

Code:

1. **Load and Explore Dataset**

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

print(df.head())

print(df.info())


sns.countplot(x='target', data=df)

plt.title('Distribution of Target Classes')

plt.show()
```

```
   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
```



2. **Splitting the dataset for training and testing.**

```
# Define Features (X) and Target (y)

X = df.drop('target', axis=1)

y = df['target']
```

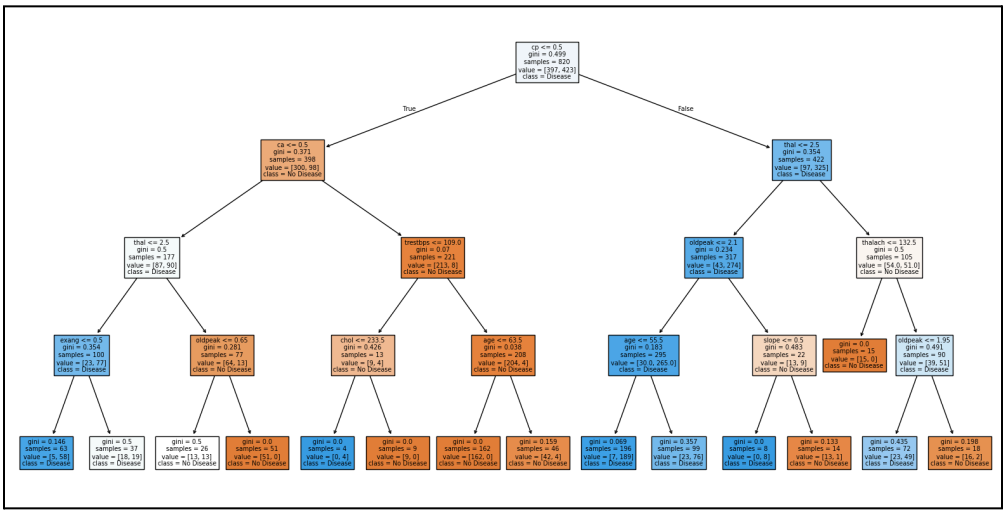
```
# Split data: 80% Training, 20% Testing

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


print(f"Training set size: {X_train.shape}")

print(f"Testing set size: {X_test.shape}")
```

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

3. **Decision Tree**

```
# Initialize and Train

dt_model = DecisionTreeClassifier(max_depth=4, random_state=52)

dt_model.fit(X_train, y_train)


# Make Predictions

dt_preds = dt_model.predict(X_test)


# Evaluation

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

print(f"Accuracy: {accuracy_score(y_test, dt_preds):.2f}")

print(classification_report(y_test, dt_preds))


# Visualize the Tree

plt.figure(figsize=(20,10))

plot_tree(dt_model, feature_names=X.columns, class_names=['No Disease', 'Disease'],
filled=True)

plt.show()
```

```
--- Decision Tree Performance ---
Accuracy: 0.80
              precision    recall  f1-score   support

           0       0.88      0.70      0.78       102
           1       0.75      0.90      0.82       103

    accuracy                           0.80       205
   macro avg       0.81      0.80      0.80       205
weighted avg       0.81      0.80      0.80       205
```

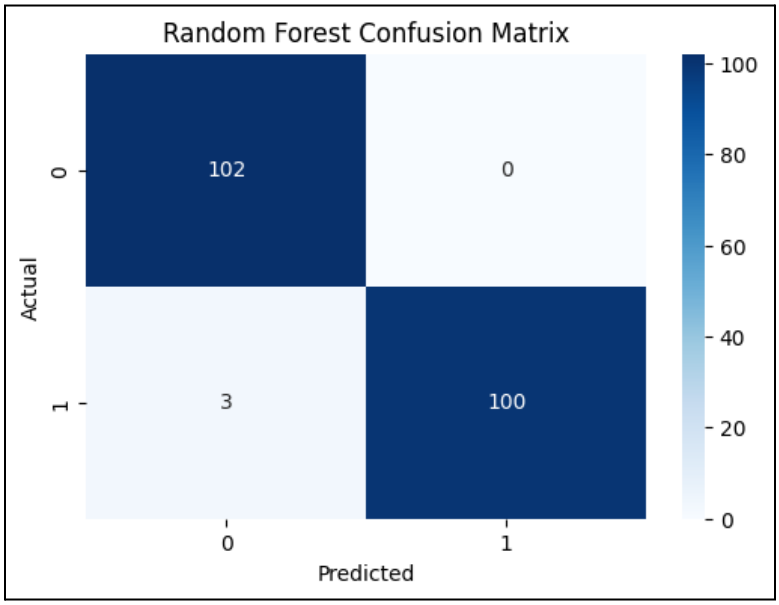


4. Random Forest Algorithm

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=52)

rf_model.fit(X_train, y_train)



rf_preds = rf_model.predict(X_test)


# Evaluation

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

print(f"Accuracy: {accuracy_score(y_test, rf_preds):.2f}")
```

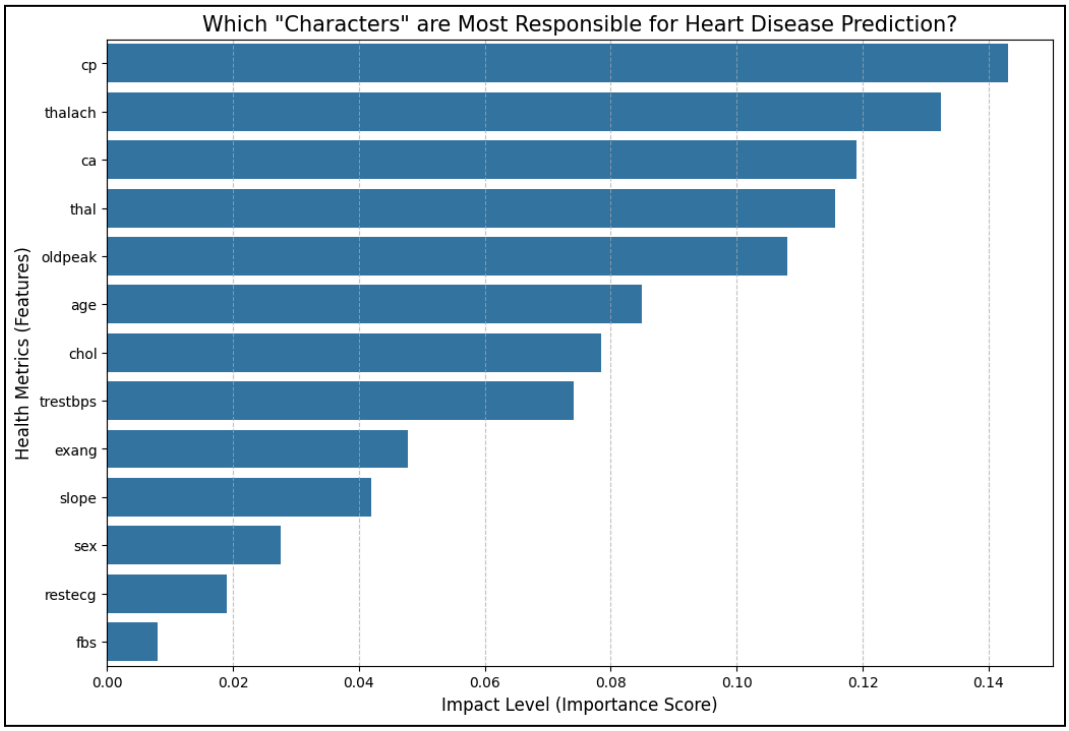
```
print(classification_report(y_test, rf_preds))


# Confusion Matrix Heatmap
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, rf_preds), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')
plt.show()
```

```
--- Random Forest Performance ---
Accuracy: 0.99
              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
```



5. **Characters responsible for Heart Disease**

```
# 1. Extract feature importances from the Random Forest model
importances = rf_model.feature_importances_
feature_names = X.columns


# 2. Create a DataFrame for easy plotting
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance Score': importances
}).sort_values(by='Importance Score', ascending=False)


plt.figure(figsize=(12, 8))
sns.barplot(
    x='Importance Score',
    y='Feature',
    data=feature_importance_df
)
plt.title('Which "Characters" are Most Responsible for Heart Disease Prediction?', fontsize=15)
plt.xlabel('Impact Level (Importance Score)', fontsize=12)
plt.ylabel('Health Metrics (Features)', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



CONCLUSION

- **Decision Tree and Random Forest** were successfully implemented for the classification of heart disease.
- The **Random Forest** model generally outperformed the single Decision Tree in terms of accuracy and robustness.
- **Feature Importance** visualization allowed us to identify which medical attributes are most responsible for a high-risk diagnosis, making the model transparent and useful for clinical decision-making.