AAYUSH PATIL - 56

MLDL EXPERIMENT 3

AIM -

Apply Decision Tree and Random Forest for classification tasks

DATASET SOURCE -

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

DATASET DESCRIPTION -

This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Attribute Information:

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholesterol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by fluoroscopy
13. thal: 0 = normal; 1 = fixed defect; 2 = reversible defect

    The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.

THEORY –

Decision Tree

A Decision Tree is a supervised machine learning algorithm used for classification and regression tasks. It works by splitting the dataset into smaller subsets based on feature conditions, forming a tree-like structure of decisions.

It is called a "tree" because:

- The top node is called the **Root Node**
- Each decision point is an **Internal Node**
- Final output nodes are called **Leaf Nodes**

For classification problems like heart disease prediction, it classifies data into discrete categories (0 or 1).

1. Working Principle

The Decision Tree works by:

1. Selecting the best feature to split the data.
2. Dividing the dataset into branches based on a condition.
3. Repeating the process recursively.
4. Stopping when data becomes pure (all 0s or all 1s).

The algorithm selects splits using impurity measures such as:

- Gini Impurity
- Entropy (Information Gain)

2. Gini Impurity Formula

Gini impurity is calculated as:

Gini = $1 - \Sigma (pi^2)$

Where:

- pi = probability of each class
  Lower Gini means better separation.
- The algorithm chooses the split that minimizes Gini impurity.

3. Steps in Decision Tree Algorithm

1. Start with a full dataset.
2. Try all features and possible split values.
3. Calculate impurity for each split.
4. Choose the best split.
5. Repeat for child nodes.
6. Stop when:

    a. All samples belong to one class
    b. Maximum depth reached
    c. Minimum samples limit reached

4. Advantages

- Easy to understand and interpret
- No need for data scaling
- Works well for nonlinear relationships
- Can handle both numerical and categorical data

5. Disadvantages

- Prone to overfitting
- Sensitive to small data variations
- Less stable compared to ensemble models

6. Application in Our Experiment

In our experiment:

- Input: Medical parameters (age, cholesterol, chest pain, etc.)
- Output: Heart disease (0 = No, 1 = Yes)
- The model learned splitting rules to classify patients.
- Feature importance was calculated to understand major contributing factors.

Random Forest

Random Forest is an ensemble learning algorithm used for classification and regression. It builds multiple Decision Trees and combines their predictions.

It is called "Random" because:

- It selects random subsets of data.
- It selects random subsets of features.

It improves accuracy and reduces overfitting.

1. Working Principle

Random Forest works in the following way:

1. Creates multiple Decision Trees.
2. Each tree is trained on a random subset of data (Bootstrap Sampling).
3. Each tree uses a random subset of features.
4. Each tree gives a prediction.
5. Final prediction is made by Majority Voting.

For classification:

Final Output = Class predicted by majority of trees.

2. Key Concepts

Bootstrap Sampling :

Random sampling of data with replacement.

Feature Randomness :

At each split, only a subset of features is considered.

Ensemble Learning :

Combining multiple models to improve performance.

3. Algorithm Steps

1. Select N random samples from the dataset.
2. Build a Decision Tree.
3. Repeat for multiple trees.
4. Collect predictions from all trees.
5. Use majority voting to decide the final class.

4. Advantages

- Reduces overfitting
- High accuracy
- Less variance compared to single Decision Tree

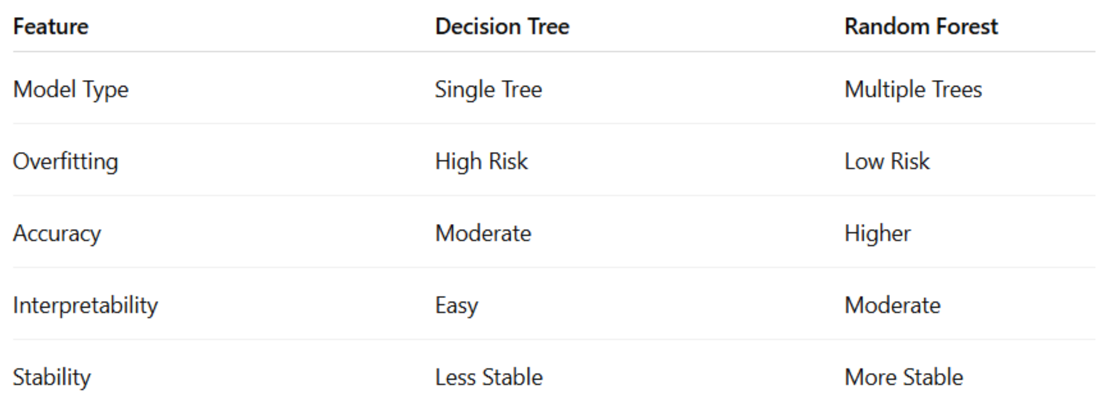
5. Disadvantages

- More computationally expensive
- Less interpretable than Decision Tree
- Requires more memory

6. Application in Our Experiment

- Multiple trees were built using medical attributes.
- Each tree predicted heart disease independently.
- Final classification was determined by majority voting.
- Feature importance was derived from aggregated trees.

COMPARISION



Feature	Decision Tree	Random Forest
Model Type	Single Tree	Multiple Trees
Overfitting	High Risk	Low Risk
Accuracy	Moderate	Higher
Interpretability	Easy	Moderate
Stability	Less Stable	More Stable

CODE -

Decision tree

```python
# ============================================
# DECISION TREE - HEART DISEASE PREDICTION
# ============================================
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load Dataset
df = pd.read_csv('/content/heart.csv')

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

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

# Train Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Accuracy
accuracy = accuracy_score(y_test, dt_model.predict(X_test))
print("Decision Tree Accuracy:", accuracy)

# ============================================
# USER INPUT (WITH LIMITS)
# ============================================
print("\nEnter Patient Details Within Limits:")

age = float(input("Age (29 - 77): "))
sex = float(input("Sex (0 = Female, 1 = Male): "))
cp = float(input("Chest Pain Type (0 - 3): "))
trestbps = float(input("Resting BP (94 - 200): "))
chol = float(input("Cholesterol (126 - 564): "))
fbs = float(input("Fasting Blood Sugar (0 or 1): "))
restecg = float(input("Rest ECG (0 - 2): "))
thalach = float(input("Max Heart Rate (71 - 202): "))
exang = float(input("Exercise Angina (0 or 1): "))
oldpeak = float(input("Oldpeak (0.0 - 6.2): "))
slope = float(input("Slope (0 - 2): "))
ca = float(input("CA (0 - 4): "))
```

```
thal = float(input("Thal (0 - 3): "))

user_data = pd.DataFrame([[age, sex, cp, trestbps, chol,
                           fbs, restecg, thalach,
                           exang, oldpeak, slope,
                           ca, thal]], columns=X.columns)

# Prediction
prediction = dt_model.predict(user_data)[0]
probability = dt_model.predict_proba(user_data)[0]

print("\nPrediction:",
      "Heart Disease" if prediction == 1 else "No Heart Disease")

# ============================================
# GRAPH 1: Prediction Probability
# ===========================================
plt.figure()
plt.bar(["No Disease", "Heart Disease"], probability)
plt.title("Decision Tree Prediction Probability")
plt.ylabel("Probability")
plt.show()
# ============================================
# GRAPH 2: Feature Importance (Decision Tree)
# ============================================
plt.figure()
plt.bar(user_data.columns, user_data.iloc[0])
plt.xticks(rotation=90)
plt.title("Patient Input Values")
plt.ylabel("Value")
plt.show()

feature_to_compare = "chol"   # Change feature name if needed

plt.figure()
plt.hist(df[feature_to_compare], bins=20)
plt.axvline(user_data[feature_to_compare].values[0])
plt.title(f"Distribution of {feature_to_compare} with Your Value")
plt.xlabel(feature_to_compare)
plt.ylabel("Frequency")
plt.show()
# ============================================
# GRAPH 3: Combined Risk Score
# ============================================
risk_score = probability[1] * 100   # probability of heart disease

plt.figure()
plt.bar(["Heart Disease Risk"], [risk_score])
plt.ylim(0,100)
plt.title("Heart Disease Risk Score (%)")
plt.ylabel("Risk Percentage")
plt.show()
# ==============================================
# GRAPH 4: Above/Below Dataset Average
# ==============================================
dataset_mean = df.mean()
difference = user_data.iloc[0] - dataset_mean

plt.figure()
plt.bar(difference.index, difference)
plt.xticks(rotation=90)
plt.title("Difference From Dataset Average")
plt.ylabel("Above (+) or Below (-) Average")
plt.show()
```

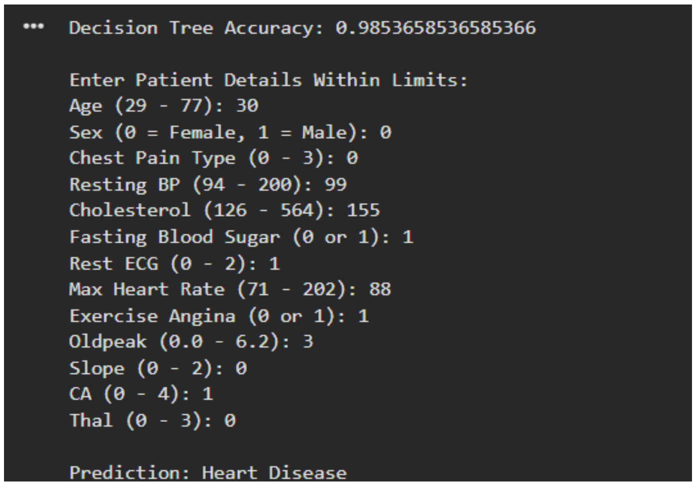
Random Forest

```
# ==============================================
# RANDOM FOREST - HEART DISEASE PREDICTION
# ==============================================
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load Dataset
df = pd.read_csv('/content/heart.csv')

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

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

# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Accuracy
accuracy = accuracy_score(y_test, rf_model.predict(X_test))
print("Random Forest Accuracy:", accuracy)
# ==============================================
# USER INPUT (WITH LIMITS)
# ==============================================
print("\nEnter Patient Details Within Limits:")

age = float(input("Age (29 - 77): "))
```

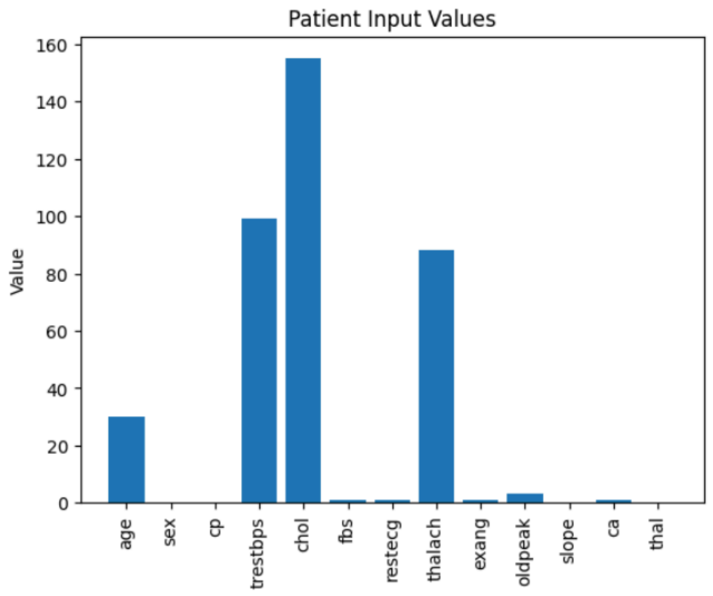
```python
sex = float(input("Sex (0 = Female, 1 = Male): "))
cp = float(input("Chest Pain Type (0 - 3): "))
trestbps = float(input("Resting BP (94 - 200): "))
chol = float(input("Cholesterol (126 - 564): "))
fbs = float(input("Fasting Blood Sugar (0 or 1): "))
restecg = float(input("Rest ECG (0 - 2): "))
thalach = float(input("Max Heart Rate (71 - 202): "))
exang = float(input("Exercise Angina (0 or 1): "))
oldpeak = float(input("Oldpeak (0.0 - 6.2): "))
slope = float(input("Slope (0 - 2): "))
ca = float(input("CA (0 - 4): "))
thal = float(input("Thal (0 - 3): "))

user_data = pd.DataFrame([[age, sex, cp, trestbps, chol,
                           fbs, restecg, thalach,
                           exang, oldpeak, slope,
                           ca, thal]], columns=X.columns)

# Prediction
prediction = rf_model.predict(user_data)[0]
probability = rf_model.predict_proba(user_data)[0]

print("\nPrediction:",
      "Heart Disease" if prediction == 1 else "No Heart Disease")
# ============================================
# GRAPH 1: Prediction Probability
# ===========================================
plt.figure()
plt.bar(["No Disease", "Heart Disease"], probability)
plt.title("Random Forest Prediction Probability")
plt.ylabel("Probability")
plt.show()
# ============================================
# GRAPH 2: Feature Importance (Random Forest)
# ============================================
plt.figure()
plt.bar(X.columns, rf_model.feature_importances_)
plt.xticks(rotation=90)
plt.title("Random Forest Feature Importance")
plt.ylabel("Importance")
plt.show()

import matplotlib.pyplot as plt
import numpy as np

risk = probability[1] * 100

plt.figure()
plt.barh(["Risk"], [risk])
plt.xlim(0,100)
plt.title("Heart Disease Risk Gauge (%) - Random Forest")
plt.xlabel("Risk Percentage")
plt.show()

plt.figure()
plt.bar(["No Disease", "Heart Disease"], probability * 100)
plt.title("Random Forest Prediction Probability (%)")
plt.ylabel("Probability %")
plt.show()

import pandas as pd

importance = pd.Series(rf_model.feature_importances_, index=df.drop('target', axis=1).columns)
top5 = importance.sort_values(ascending=False).head(5)

plt.figure()
plt.bar(top5.index, top5.values)
plt.title("Top 5 Important Features - Random Forest")
plt.ylabel("Importance Score")
plt.xticks(rotation=45)
plt.show()

dataset_mean = df.mean()

comparison_df = pd.DataFrame({
    "Your Value": user_data.iloc[0],
    "Dataset Mean": dataset_mean
})

comparison_df = comparison_df.drop("target")

plt.figure()
comparison_df.plot(kind="bar")
plt.title("Your Values vs Dataset Average")
plt.xticks(rotation=90)
plt.ylabel("Value")
plt.show()

zones = ["Low Risk", "Medium Risk", "High Risk"]
zone_values = [0, 0, 0]

if risk < 40:
    zone_values[0] = risk
elif risk < 70:
    zone_values[1] = risk
else:
    zone_values[2] = risk

plt.figure()
plt.bar(zones, zone_values)
plt.title("Heart Disease Risk Zone")
plt.ylabel("Risk Percentage")
plt.show()
```

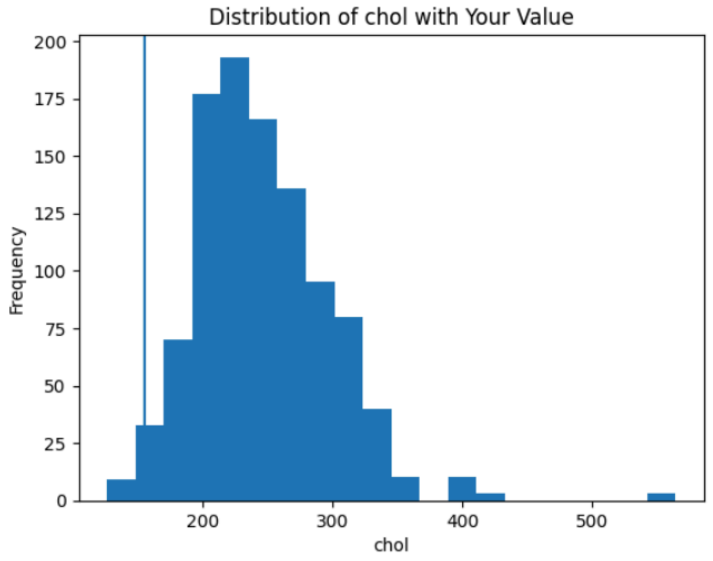
OUTPUT -

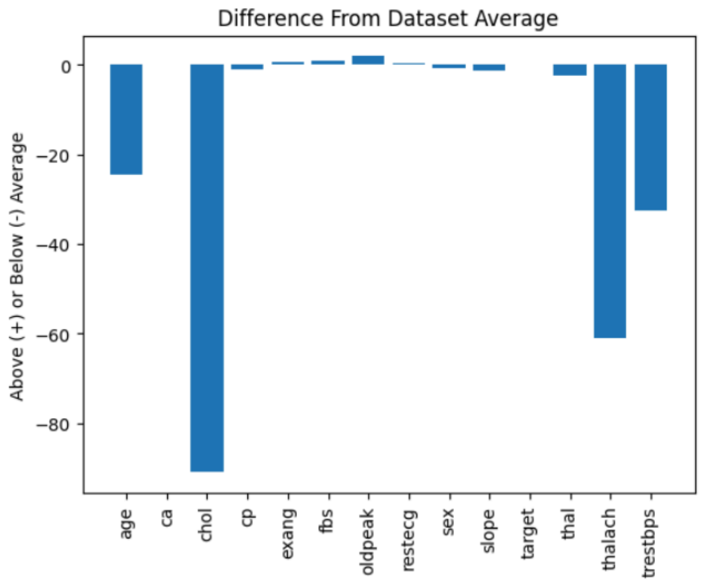
Decision Tree

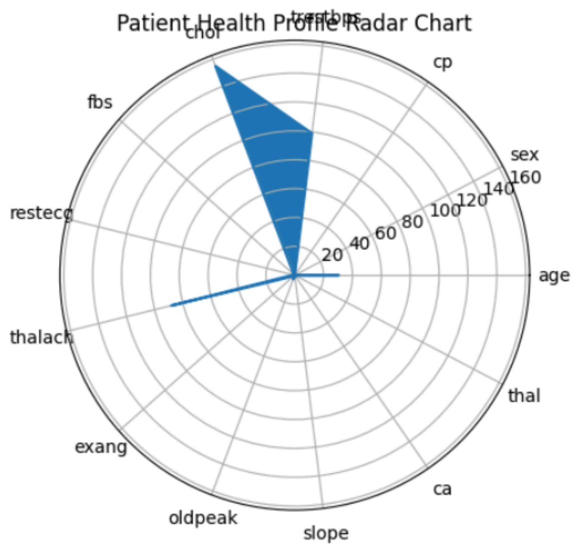


```
•••  Decision Tree Accuracy: 0.9853658536585366

     Enter Patient Details Within Limits:
     Age (29 - 77): 30
     Sex (0 = Female, 1 = Male): 0
     Chest Pain Type (0 - 3): 0
     Resting BP (94 - 200): 99
     Cholesterol (126 - 564): 155
     Fasting Blood Sugar (0 or 1): 1
     Rest ECG (0 - 2): 1
     Max Heart Rate (71 - 202): 88
     Exercise Angina (0 or 1): 1
     Oldpeak (0.0 - 6.2): 3
     Slope (0 - 2): 0
     CA (0 - 4): 1
     Thal (0 - 3): 0

     Prediction: Heart Disease
```

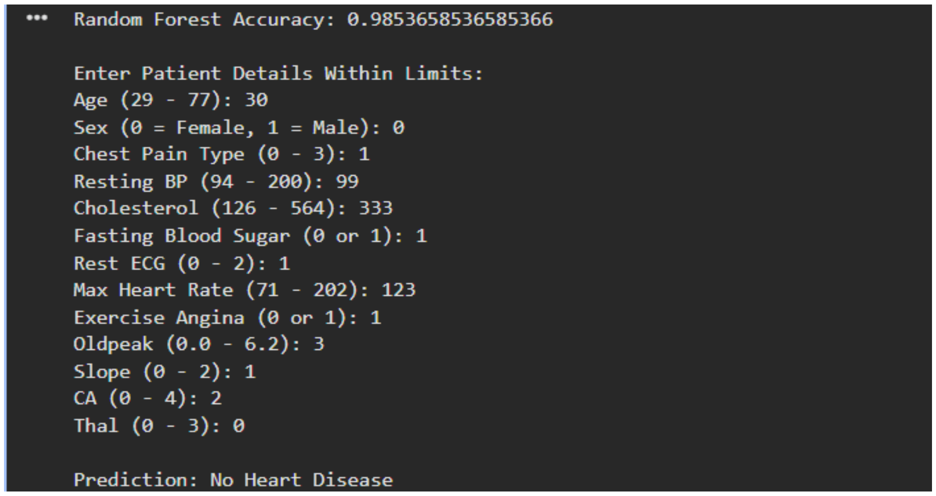




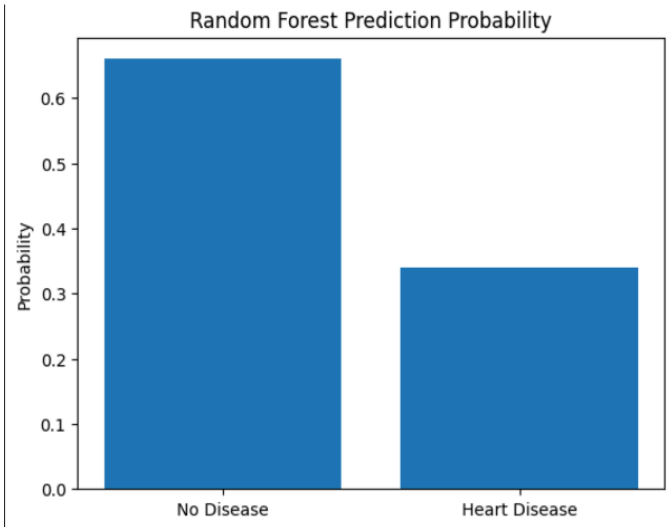


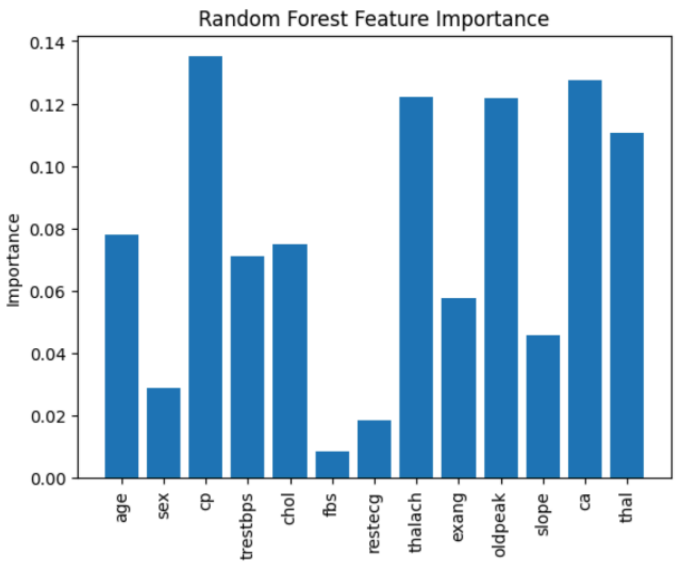


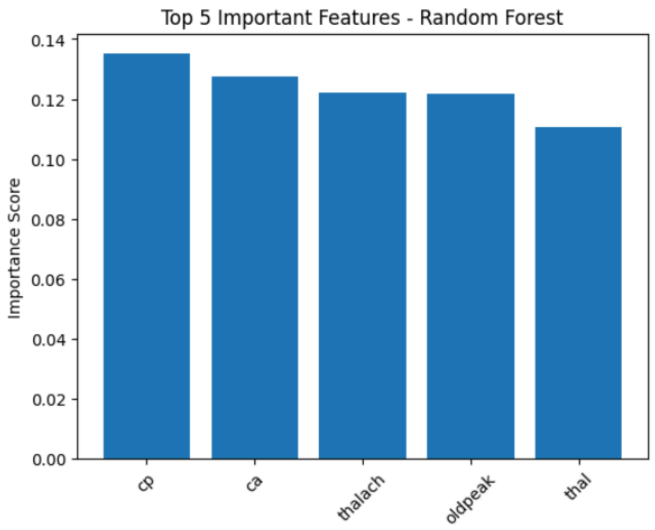
Random forest

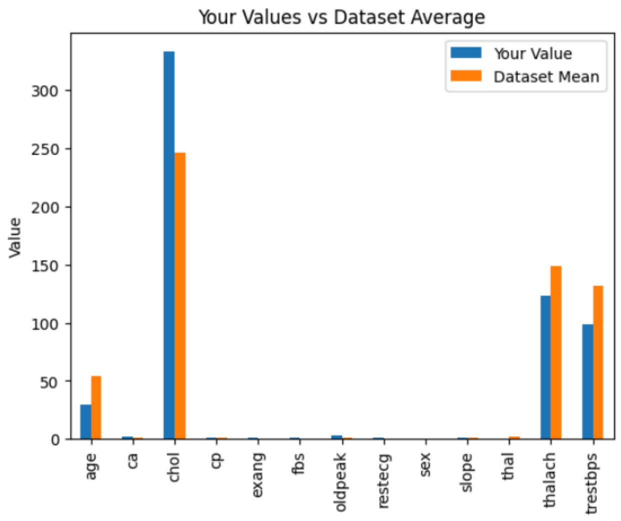


```
Random Forest Accuracy: 0.9853658536585366

Enter Patient Details Within Limits:
Age (29 - 77): 30
Sex (0 = Female, 1 = Male): 0
Chest Pain Type (0 - 3): 1
Resting BP (94 - 200): 99
Cholesterol (126 - 564): 333
Fasting Blood Sugar (0 or 1): 1
Rest ECG (0 - 2): 1
Max Heart Rate (71 - 202): 123
Exercise Angina (0 or 1): 1
Oldpeak (0.0 - 6.2): 3
Slope (0 - 2): 1
CA (0 - 4): 2
Thal (0 - 3): 0

Prediction: No Heart Disease
```









CONCLUSION -

In this experiment, Decision Tree and Random Forest algorithms were applied for heart disease classification.

- Decision Tree provided interpretable decision rules.
- Random Forest improved accuracy using ensemble learning.
- Random Forest generally performed better due to reduced variance and improved generalization.

Both algorithms successfully classified patients based on medical features.