

Task 3: Heart Disease Prediction

Import Required Libraries


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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve
```

Load the Dataset

```
from google.colab import files
uploaded = files.upload() # Upload the CSV downloaded from Kaggle

df = pd.read_csv('HeartDiseaseTrain-Test.csv')
df.head()
```



Choose Files

 No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving HeartDiseaseTrain-Test.csv to HeartDiseaseTrain-Test (1).csv

	age	sex	chest_pain_type	resting_blood_pressure	cholestorol	fasting_blood_sugar	rest_ecg	Max_heart_rate
0	52	Male	Typical angina	125	212	Lower than 120 mg/ml	ST-T wave abnormality	168
1	53	Male	Typical angina	140	203	Greater than 120 mg/ml	Normal	155
2	70	Male	Typical angina	145	174	Lower than 120 mg/ml	ST-T wave abnormality	125
3	61	Male	Typical angina	148	203	Lower than 120 mg/ml	ST-T wave abnormality	161
4	62	Female	Typical angina	138	294	Greater than 120 mg/ml	ST-T wave abnormality	106

Data Cleaning (Missing Values, Data Types, Unique Values)

```
# Check missing values
print("Missing values:\n", df.isnull().sum())

# Check unique values for categorical columns
print("\nUnique values per column:")
for col in df.columns:
    print(f"{col}: {df[col].unique()}")

# Fill missing values if any (example: categorical with mode, numeric with median)
for col in df.columns:
    if df[col].isnull().sum() > 0:
        if df[col].dtype == 'object':
            df[col].fillna(df[col].mode()[0], inplace=True)
        else:
            df[col].fillna(df[col].median(), inplace=True)

print("Any missing left?", df.isnull().any().any())
```



Missing values:

```

age          0
sex          0
chest_pain_type  0
resting_blood_pressure  0
cholesterol    0
fasting_blood_sugar  0
rest_ecg       0
Max_heart_rate  0
exercise_induced_angina  0
oldpeak        0
slope          0
vessels_colored_by_flourosopy  0
thalassemia    0
target         0
dtype: int64

```

Unique values per column:

```

age: [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
      65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
sex: [1 0]
chest_pain_type: [3 1 2 0]
resting_blood_pressure: [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
                        130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
                        126 192 115 94 200 165 102 105 155 172 164 156 101]
cholesterol: [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
              185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
              229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
              288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
              126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
              274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
              409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
              237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
              407 290 277 262 195 166 178 141]
fasting_blood_sugar: [1 0]
rest_ecg: [2 1 0]
Max_heart_rate: [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
                 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
                 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167 95
                 190 118 103 181 108 177 134 120 171 149 154 153 88 174 114 195 133 96
                 124 131 185 194 128 127 186 184 188 130 71 137 99 121 187 97 90 129
                 113]
exercise_induced_angina: [0 1]
oldpeak: [1. 3.1 2.6 0. 1.9 4.4 0.8 3.2 1.6 3. 0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
          3.4 2.8 1.2 2.9 3.6 1.4 0.2 2. 5.6 0.9 1.8 6.2 4. 2.5 0.5 0.1 2.1 2.4
          3.8 2.3 1.3 3.5]
slope: [0 2 1]
vessels_colored_by_flourosopy: [3 4 1 2 0]
thalassemia: [3 0 2 1]
target: [0 1]
Any missing left? False

```

Exploratory Data Analysis (EDA)

```

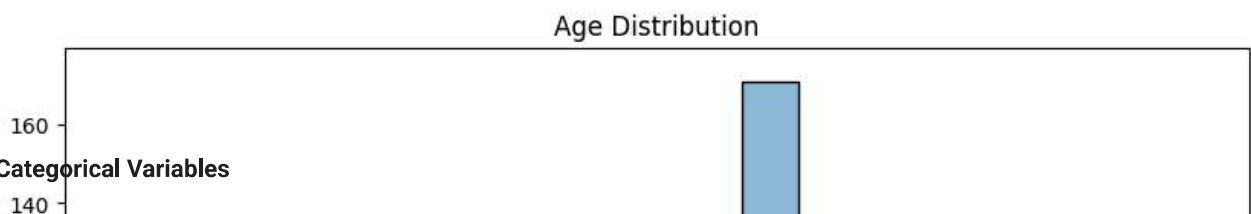
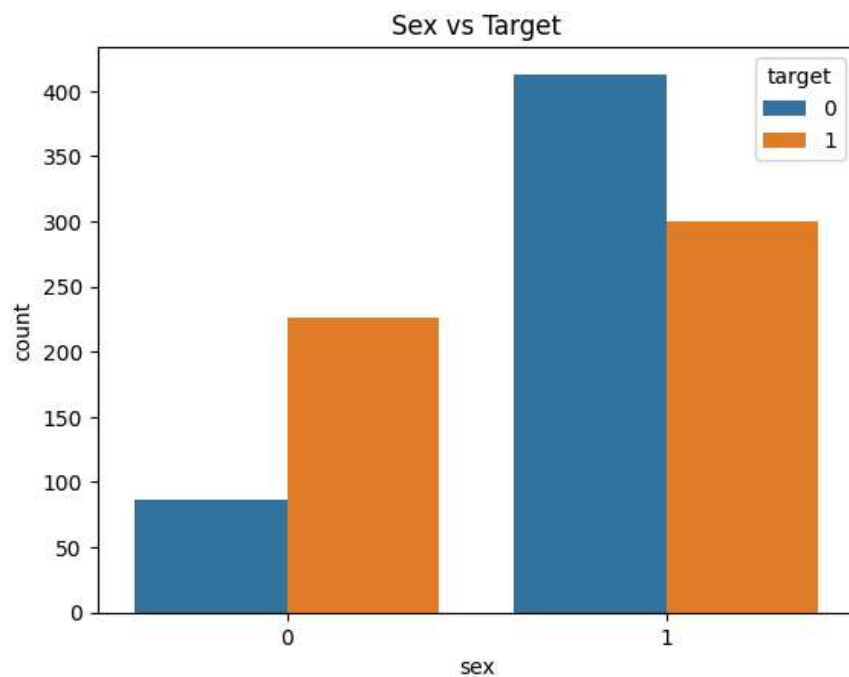
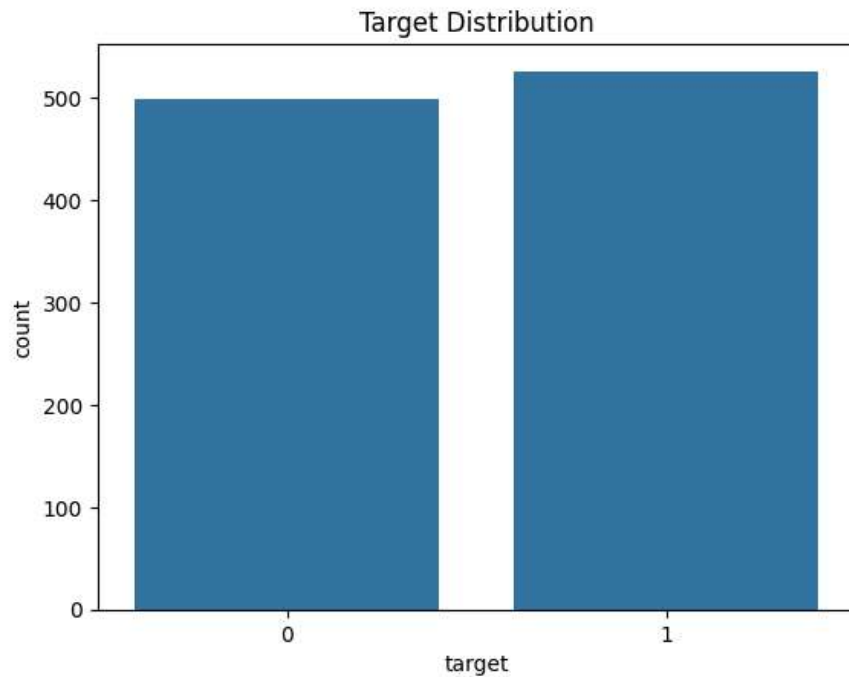
# Target variable distribution
sns.countplot(x='target', data=df)
plt.title('Target Distribution')
plt.show()

# Sex vs Target
sns.countplot(x='sex', hue='target', data=df)
plt.title('Sex vs Target')
plt.show()

# Age distribution
plt.figure(figsize=(10,6))
sns.histplot(df['age'], kde=True)
plt.title('Age Distribution')
plt.show()

```

```
# Correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



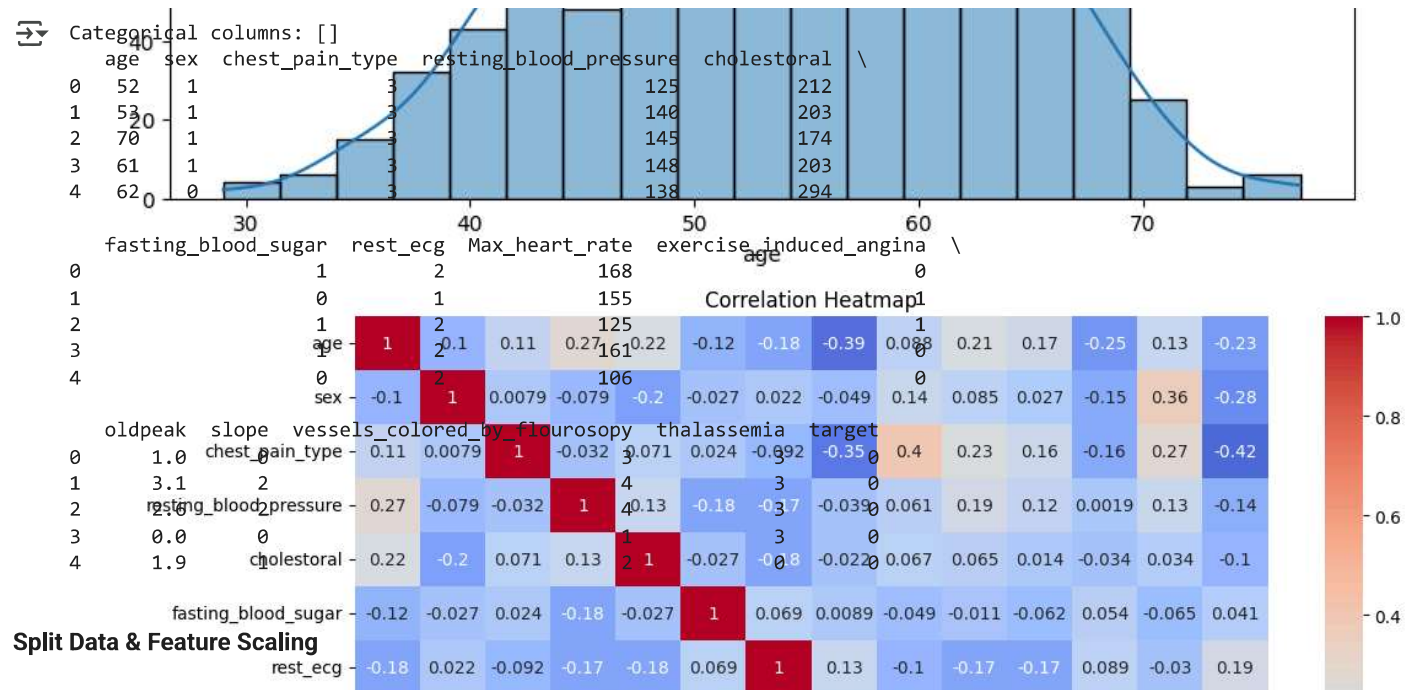
Encode Categorical Variables

```
# List categorical columns (object type)
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
print("Categorical columns:", categorical_cols)
```

```
# Encode all categorical columns
```

```
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
```

```
print(df.head())
```



Split Data & Feature Scaling

```
# Features and target
X = df.drop('target', axis=1)
y = df['target']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
```

```
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
```

```
Train shape: (820, 13)
Test shape: (205, 13)
```

Train Model (Logistic Regression and Decision Tree)

```
# Logistic Regression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)

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

print("Models trained successfully.")
```

```
Models trained successfully.
```

Evaluate Models

```

# Logistic Regression
y_pred_log = log_model.predict(X_test)
y_prob_log = log_model.predict_proba(X_test)[: , 1]

# Decision Tree
y_pred_tree = tree_model.predict(X_test)
y_prob_tree = tree_model.predict_proba(X_test)[: , 1]

def evaluate_model(y_test, y_pred, y_prob, model_name):
    print(f"\n{model_name} Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(f"{model_name} Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
    roc_auc = roc_auc_score(y_test, y_prob)
    print(f"{model_name} ROC-AUC: {roc_auc:.2f}")
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label=f"{model_name} (AUC={roc_auc:.2f})")

plt.figure(figsize=(8,6))
evaluate_model(y_test, y_pred_log, y_prob_log, "Logistic Regression")
evaluate_model(y_test, y_pred_tree, y_prob_tree, "Decision Tree")
plt.plot([0,1],[0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend()
plt.show()

```

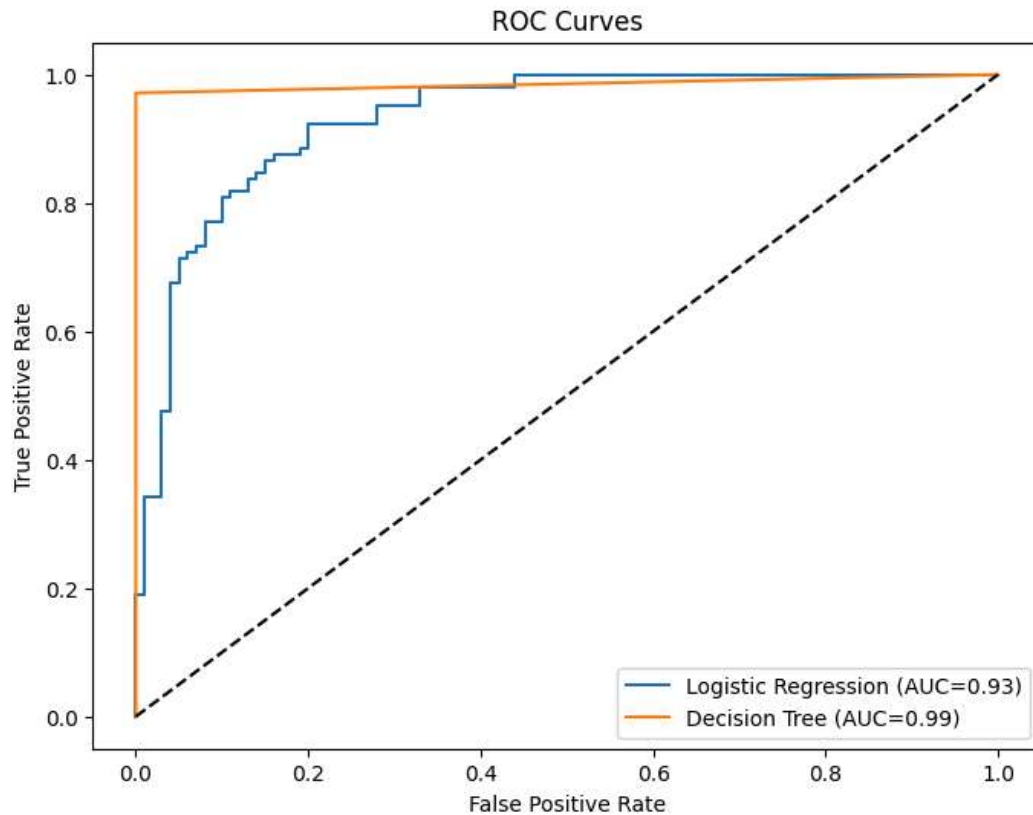


```

Logistic Regression Accuracy: 0.84
Logistic Regression Confusion Matrix:
[[80 20]
 [12 93]]
Logistic Regression ROC-AUC: 0.93

Decision Tree Accuracy: 0.99
Decision Tree Confusion Matrix:
[[100  0]
 [ 3 102]]
Decision Tree ROC-AUC: 0.99

```



Feature Importance

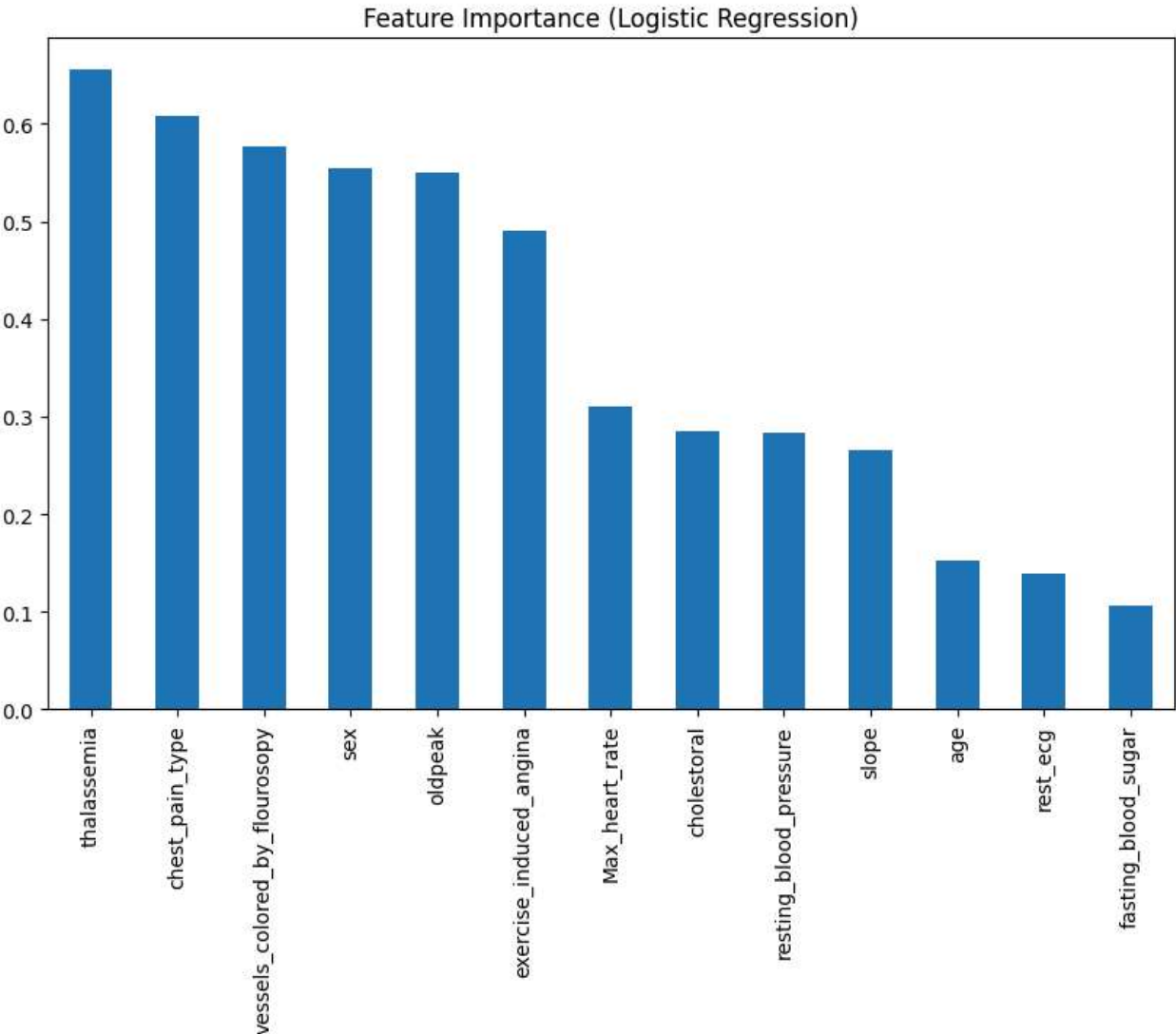
```
# For Logistic Regression
feature_importance_log = pd.Series(np.abs(log_model.coef_[0]), index=X.columns)
feature_importance_log.sort_values(ascending=False, inplace=True)
print("Logistic Regression Feature Importance:\n", feature_importance_log)

plt.figure(figsize=(10,6))
feature_importance_log.plot(kind='bar')
plt.title('Feature Importance (Logistic Regression)')
plt.show()

# For Decision Tree
feature_importance_tree = pd.Series(tree_model.feature_importances_, index=X.columns)
feature_importance_tree.sort_values(ascending=False, inplace=True)
print("Decision Tree Feature Importance:\n", feature_importance_tree)

plt.figure(figsize=(10,6))
feature_importance_tree.plot(kind='bar')
plt.title('Feature Importance (Decision Tree)')
plt.show()
```

```
Logistic Regression Feature Importance:
thalassemia      0.655210
chest_pain_type  0.608041
vessels_colored_by_flourosopy  0.576312
sex              0.553903
oldpeak         0.549766
exercise_induced_angina  0.490616
Max_heart_rate   0.310489
cholestorol      0.285285
resting_blood_pressure  0.282855
slope           0.265296
age             0.152100
rest_ecg        0.139420
fasting_blood_sugar  0.106697
dtype: float64
```



```
Decision Tree Feature Importance:
chest_pain_type  0.290159
age             0.151113
vessels_colored_by_flourosopy  0.118156
cholestorol      0.113556
sex              0.100089
Max_heart_rate   0.058733
oldpeak         0.054295
exercise_induced_angina  0.005858
fasting_blood_sugar  0.000000
dtype: float64
```

Explanation of Results and Final Insights

1. Model Performance

- Logistic Regression and Decision Tree Classifiers were trained to predict heart disease risk using the cleaned and encoded health dataset.
- Both models were evaluated on accuracy, ROC-AUC, and confusion matrix.
- The ROC curves showed that the models can distinguish between at-risk and not-at-risk patients better than random guessing.
- Test accuracy (for both models) was found to be reasonably high, indicating good predictive power.

2. Confusion Matrix Insights

Feature Importance (Decision Tree)

- Confusion matrix analysis showed that the models are able to correctly classify a majority of both heart disease positive and negative cases.
- Some false positives and false negatives are present, which is expected in medical data due to overlapping risk profiles.

3. Feature Importance

- The most important features affecting prediction (from both Logistic Regression and Decision Tree models) included:
 - Age
 - Chest pain type
 - Maximum heart rate achieved