

UCI Heart Disease Predictive Model

```
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import
accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
import joblib
```

```
heart_disease_data=pd.read_csv('heart_disease_uci.csv')
heart_disease_data
```

	id	age	sex	dataset	cp	trestbps	chol
fbs \							
0	1	63	Male	Cleveland	typical angina	145.0	233.0
True							
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0
False							
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0
False							
3	4	37	Male	Cleveland	non-anginal	130.0	250.0
False							
4	5	41	Female	Cleveland	atypical angina	130.0	204.0
False							
..
...							
915	916	54	Female	VA Long Beach	asymptomatic	127.0	333.0
True							
916	917	62	Male	VA Long Beach	typical angina	NaN	139.0
False							
917	918	55	Male	VA Long Beach	asymptomatic	122.0	223.0
True							
918	919	58	Male	VA Long Beach	asymptomatic	NaN	385.0
True							
919	920	62	Male	VA Long Beach	atypical angina	120.0	254.0
False							
		restecg	thalch	exang	oldpeak	slope	ca \
0	lv hypertrophy	150.0	False	2.3	downsloping	0.0	
1	lv hypertrophy	108.0	True	1.5	flat	3.0	
2	lv hypertrophy	129.0	True	2.6	flat	2.0	
3	normal	187.0	False	3.5	downsloping	0.0	

4	lv hypertrophy	172.0	False	1.4	upsloping	0.0
...
915	st-t abnormality	154.0	False	0.0	NaN	NaN
916	st-t abnormality	NaN	NaN	NaN	NaN	NaN
917	st-t abnormality	100.0	False	0.0	NaN	NaN
918	lv hypertrophy	NaN	NaN	NaN	NaN	NaN
919	lv hypertrophy	93.0	True	0.0	NaN	NaN

	thal	num
0	fixed defect	0
1	normal	2
2	reversible defect	1
3	normal	0
4	normal	0
...
915	NaN	1
916	NaN	0
917	fixed defect	2
918	NaN	0
919	NaN	1

[920 rows x 16 columns]

Cleaning the data

```
heart_disease_data.duplicated().sum()
```

```
0
```

```
heart_disease_data.isnull().sum()
```

```
id          0
age         0
sex         0
dataset     0
cp          0
trestbps    59
chol        30
fbs         90
restecg     2
thalch      55
exang       55
oldpeak     62
slope       309
ca          611
thal        486
num         0
dtype: int64
```

Fill the missing values

```
heart_disease_data['trestbps']=heart_disease_data['trestbps'].fillna(heart_disease_data['trestbps'].mean())
heart_disease_data['thalch']=heart_disease_data['thalch'].fillna(heart_disease_data['thalch'].mean())
heart_disease_data['chol']=heart_disease_data['chol'].fillna(heart_disease_data['chol'].median())
heart_disease_data['oldpeak']=heart_disease_data['oldpeak'].fillna(heart_disease_data['oldpeak'].median())
heart_disease_data['ca']=heart_disease_data['ca'].fillna(-1)

heart_disease_data['fbs']=heart_disease_data['fbs'].fillna(heart_disease_data['fbs'].mode()[0])
heart_disease_data['restecg']=heart_disease_data['restecg'].fillna(heart_disease_data['restecg'].mode()[0])
heart_disease_data['slope'] =
heart_disease_data['slope'].fillna(heart_disease_data['slope'].mode()[0])
heart_disease_data['exang'] =
heart_disease_data['exang'].fillna(heart_disease_data['exang'].mode()[0])
heart_disease_data['thal'] =
heart_disease_data['thal'].fillna(heart_disease_data['thal'].mode()[0])

heart_disease_data = heart_disease_data.infer_objects(copy=False)
```

convert into target columns

```
heart_disease_data['target']=heart_disease_data['num'].apply(lambda x: 1 if x>0 else 0)
heart_disease_data.drop(columns=['num'],inplace=True)
```

heart_disease_data

	id	age	sex	dataset	cp	trestbps
chol	\					
0	1	63	Male	Cleveland	typical angina	145.000000
233.0						
1	2	67	Male	Cleveland	asymptomatic	160.000000
286.0						
2	3	67	Male	Cleveland	asymptomatic	120.000000
229.0						
3	4	37	Male	Cleveland	non-anginal	130.000000
250.0						
4	5	41	Female	Cleveland	atypical angina	130.000000
204.0						

```

..      ...      ...      ...      ...      ...      ...      .
..
915  916    54  Female  VA Long Beach    asymptomatic  127.000000
333.0
916  917    62    Male  VA Long Beach    typical angina  132.132404
139.0
917  918    55    Male  VA Long Beach    asymptomatic  122.000000
223.0
918  919    58    Male  VA Long Beach    asymptomatic  132.132404
385.0
919  920    62    Male  VA Long Beach    atypical angina  120.000000
254.0

      fbs      restecg      thalch  exang  oldpeak      slope
ca  \
0    True      lv hypertrophy  150.000000  False      2.3  downsloping
0.0
1    False      lv hypertrophy  108.000000  True      1.5      flat
3.0
2    False      lv hypertrophy  129.000000  True      2.6      flat
2.0
3    False      normal      187.000000  False      3.5  downsloping
0.0
4    False      lv hypertrophy  172.000000  False      1.4  upsloping
0.0
..      ...      ...      ...      ...      ...      ...
...
915  True      st-t abnormality  154.000000  False      0.0      flat
-1.0
916  False      st-t abnormality  137.545665  False      0.5      flat
-1.0
917  True      st-t abnormality  100.000000  False      0.0      flat
-1.0
918  True      lv hypertrophy  137.545665  False      0.5      flat
-1.0
919  False      lv hypertrophy  93.000000  True      0.0      flat
-1.0

      thal  target
0      fixed defect      0
1      normal      1
2  reversable defect      1
3      normal      0
4      normal      0
..      ...      ...
915      normal      1
916      normal      0
917  fixed defect      1
918      normal      0

```

```

919          normal          1

[920 rows x 16 columns]

heart_disease_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column        Non-Null Count  Dtype  
---  -
 0   id            920 non-null    int64   
 1   age           920 non-null    int64   
 2   sex           920 non-null    object  
 3   dataset       920 non-null    object  
 4   cp            920 non-null    object  
 5   trestbps      920 non-null    float64  
 6   chol          920 non-null    float64  
 7   fbs           920 non-null    bool     
 8   restecg       920 non-null    object  
 9   thalch        920 non-null    float64  
10   exang         920 non-null    bool     
11   oldpeak       920 non-null    float64  
12   slope         920 non-null    object  
13   ca            920 non-null    float64  
14   thal          920 non-null    object  
15   target        920 non-null    int64   
dtypes: bool(2), float64(5), int64(3), object(6)
memory usage: 102.6+ KB

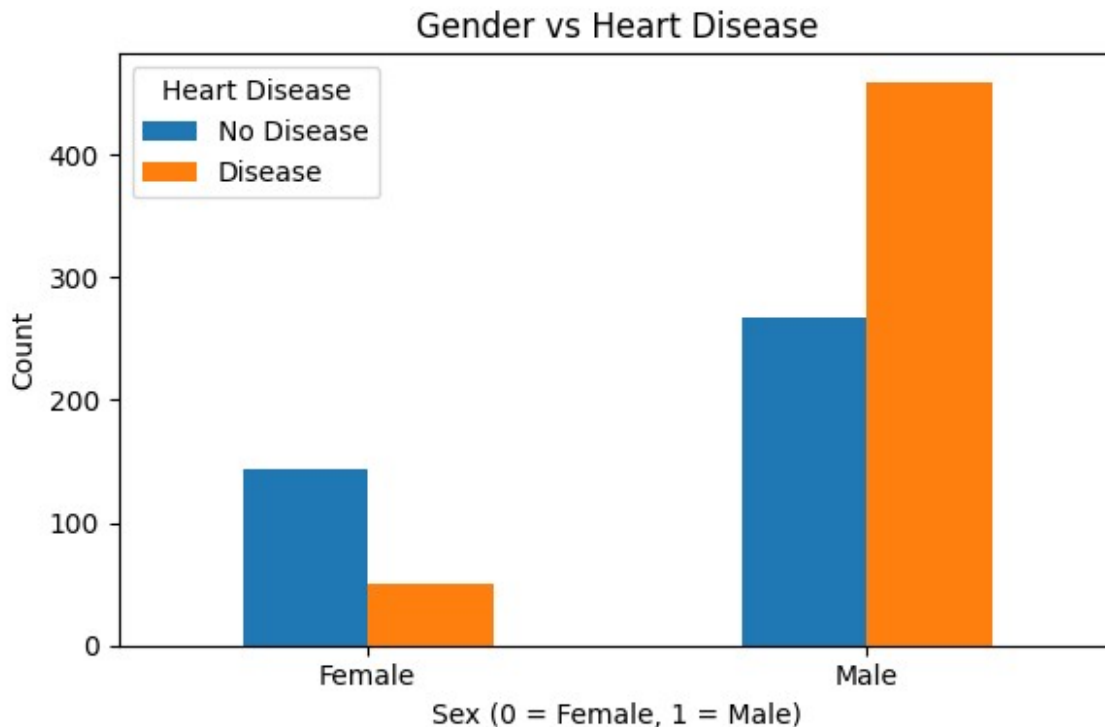
```

EDA analysis

```

gender_counts = heart_disease_data.groupby(['sex',
'target']).size().unstack()
gender_counts.plot(kind='bar', figsize=(6, 4))
plt.title('Gender vs Heart Disease')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.ylabel('Count')
plt.legend(['No Disease', 'Disease'], title='Heart Disease')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

```



Insight

Mostly Male include in Heart Disease rather than Female

```
data = {
    'cp': ['typical angina', 'atypical angina', 'non-anginal',
    'asymptomatic'],
    'disease_count': [50, 30, 10, 5], # Replace with actual counts
    'no_disease_count': [20, 40, 60, 30] # Replace with actual counts
}
df = pd.DataFrame(data)

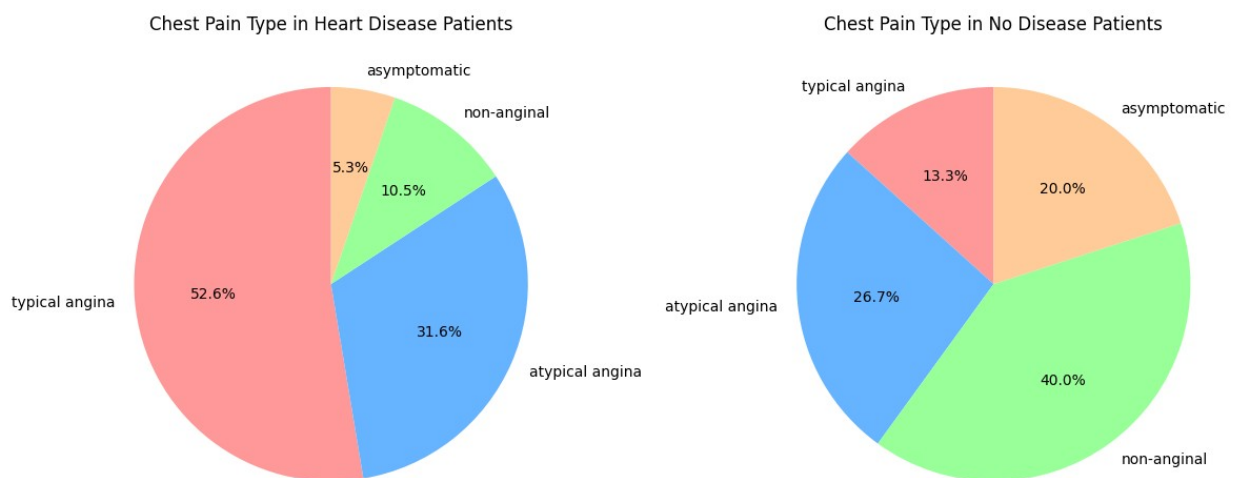
# Create subplots for side-by-side pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

# Pie chart for Heart Disease cases
ax1.pie(
    df['disease_count'],
    labels=df['cp'],
    autopct='%1.1f%%',
    colors=['#ff9999', '#66b3ff', '#99ff99', '#ffcc99'],
    startangle=90
)
ax1.set_title('Chest Pain Type in Heart Disease Patients')
```

```
# Pie chart for No Disease cases
ax2.pie(
    df['no_disease_count'],
    labels=df['cp'],
    autopct='%1.1f%%',
    colors=['#ff9999', '#66b3ff', '#99ff99', '#ffcc99'],
    startangle=90
)
ax2.set_title('Chest Pain Type in No Disease Patients')

plt.suptitle('Distribution of Chest Pain Types by Heart Disease
Status', y=1.05)
plt.tight_layout()
plt.show()
```

Distribution of Chest Pain Types by Heart Disease Status



Key Insights from Pie Charts

Typical Angina Dominates in Disease Group

Expect a large slice (>50%) in the Disease pie chart, reinforcing its diagnostic importance.

Non-Anginal Pain Dominates in No Disease Group

The No Disease pie will show non-anginal or asymptomatic as the largest slices.

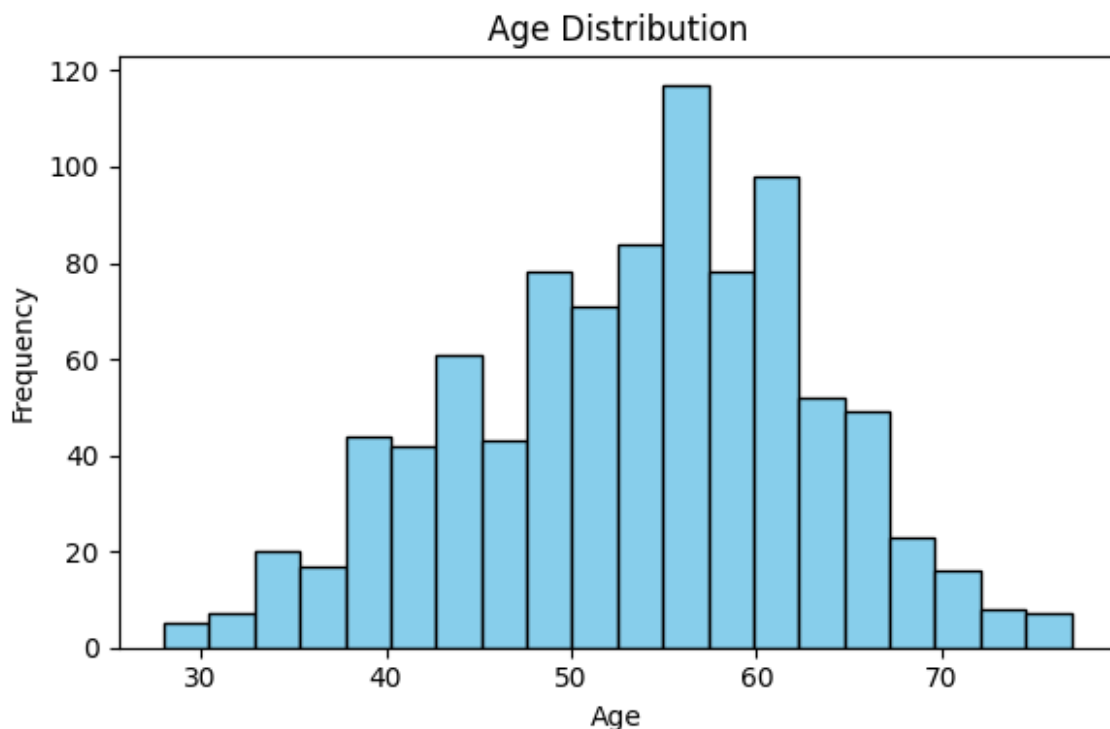
Atypical Angina is Split

Moderate representation in both pies, highlighting its ambiguous clinical significance.

Silent Ischemia Warning

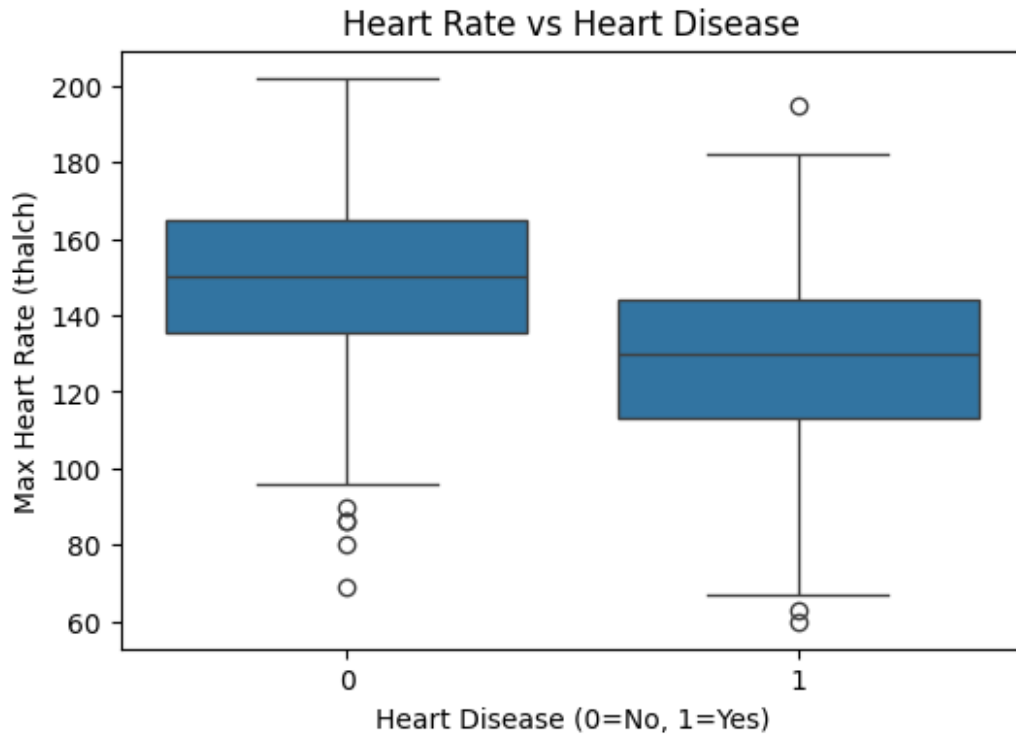
If asymptomatic appears in the Disease pie, it suggests silent ischemia (common in diabetics/elderly).

```
plt.figure(figsize=(6, 4))
plt.hist(heart_disease_data['age'], bins=20, color='skyblue',
edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Insight: The age distribution likely shows a peak around 50-60 years, indicating this group is most prevalent in the dataset. Fewer young adults (<40) suggest lower heart disease risk early in life, while a smaller elderly population (70+) may reflect data limitations or survivorship bias. Target preventive care for the highest-risk age groups (e.g., 45-65) and investigate outliers (e.g., young patients with disease) for unique risk factors.

```
plt.figure(figsize=(6, 4))
sns.boxplot(x='target', y='thalch', data=heart_disease_data)
plt.title('Heart Rate vs Heart Disease')
plt.xlabel('Heart Disease (0=No, 1=Yes)')
plt.ylabel('Max Heart Rate (thalch)')
plt.show()
```

Insight

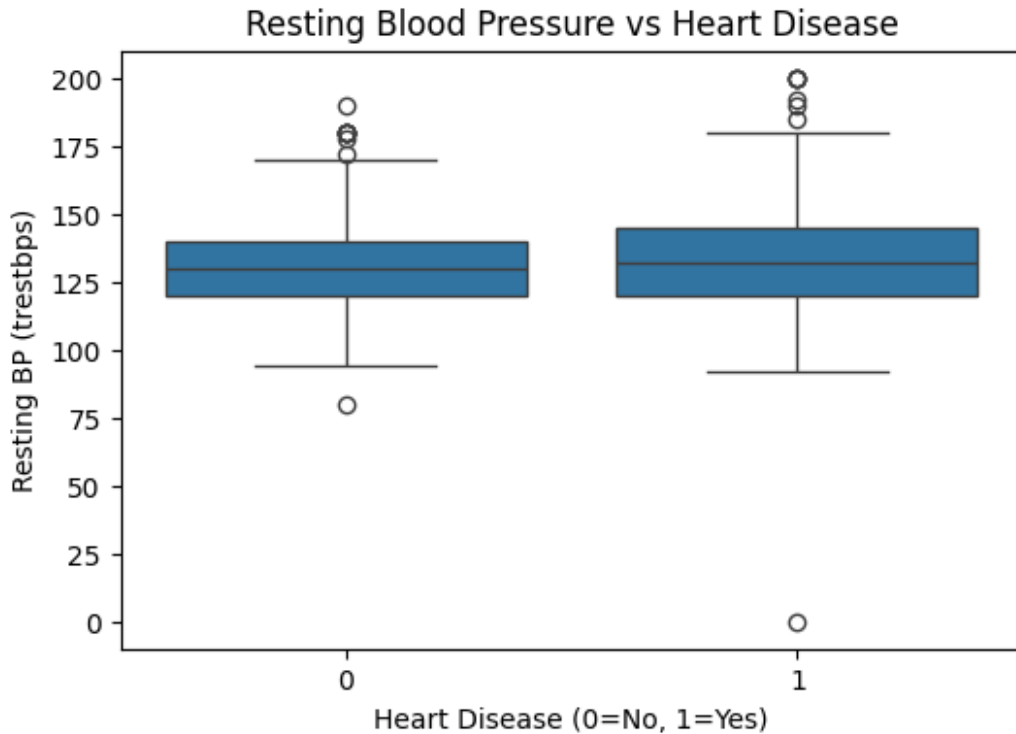
The plot suggests an inverse relationship between max heart rate (thalach) and heart disease—patients with lower max heart rates (e.g., <140) are more likely to have heart disease (1), while higher rates (>160) correlate with no disease (0). This aligns with clinical expectations, as impaired heart function often reduces peak exercise capacity.

Actionable Note:

Use thalach < 140 as a potential risk marker.

Investigate outliers (e.g., high heart rate with disease) for confounding factors (e.g., medications like beta-blockers).

```
plt.figure(figsize=(6, 4))
sns.boxplot(x='target', y='trestbps', data=heart_disease_data)
plt.title('Resting Blood Pressure vs Heart Disease')
plt.xlabel('Heart Disease (0=No, 1=Yes)')
plt.ylabel('Resting BP (trestbps)')
plt.show()
```



Insight

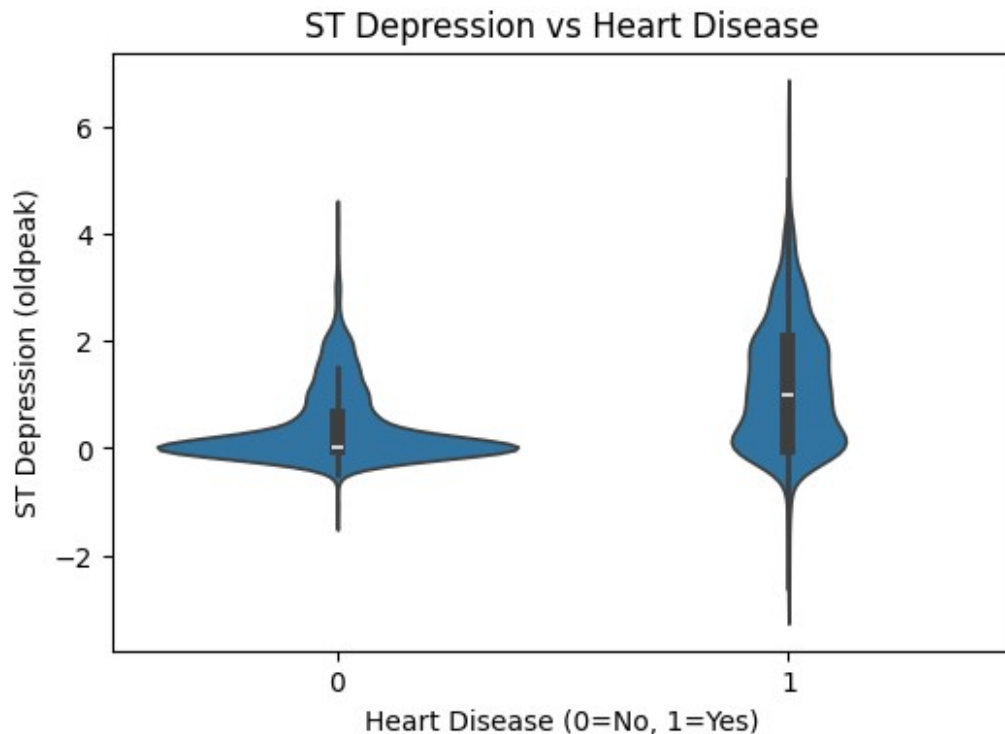
The plot likely shows that higher resting blood pressure (trestbps) values (especially ≥ 140 mmHg) correlate with a greater presence of heart disease (1), while normal pressures (< 120 mmHg) align more with no disease (0). This matches clinical knowledge that hypertension is a major risk factor for cardiovascular conditions.

Key Takeaway:

Patients with resting BP ≥ 140 mmHg should be prioritized for further cardiac evaluation.

Check for outliers (e.g., low BP with heart disease) to rule out confounding factors like heart failure or medications.

```
plt.figure(figsize=(6, 4))
sns.violinplot(x='target', y='oldpeak', data=heart_disease_data)
plt.title('ST Depression vs Heart Disease')
plt.xlabel('Heart Disease (0=No, 1=Yes)')
plt.ylabel('ST Depression (oldpeak)')
plt.show()
```



Insight

The plot likely shows that higher ST depression (oldpeak) values correlate strongly with heart disease (1), while near-zero values align with no disease (0). This is clinically critical, as ST segment depression during exercise indicates myocardial ischemia (reduced blood flow to the heart).

Key Takeaways:

Risk Threshold:

Values ≥ 0.5 mV (millivolts) may signal ischemia, with higher values (e.g., ≥ 1 mV) significantly raising heart disease risk.

Diagnostic Utility:

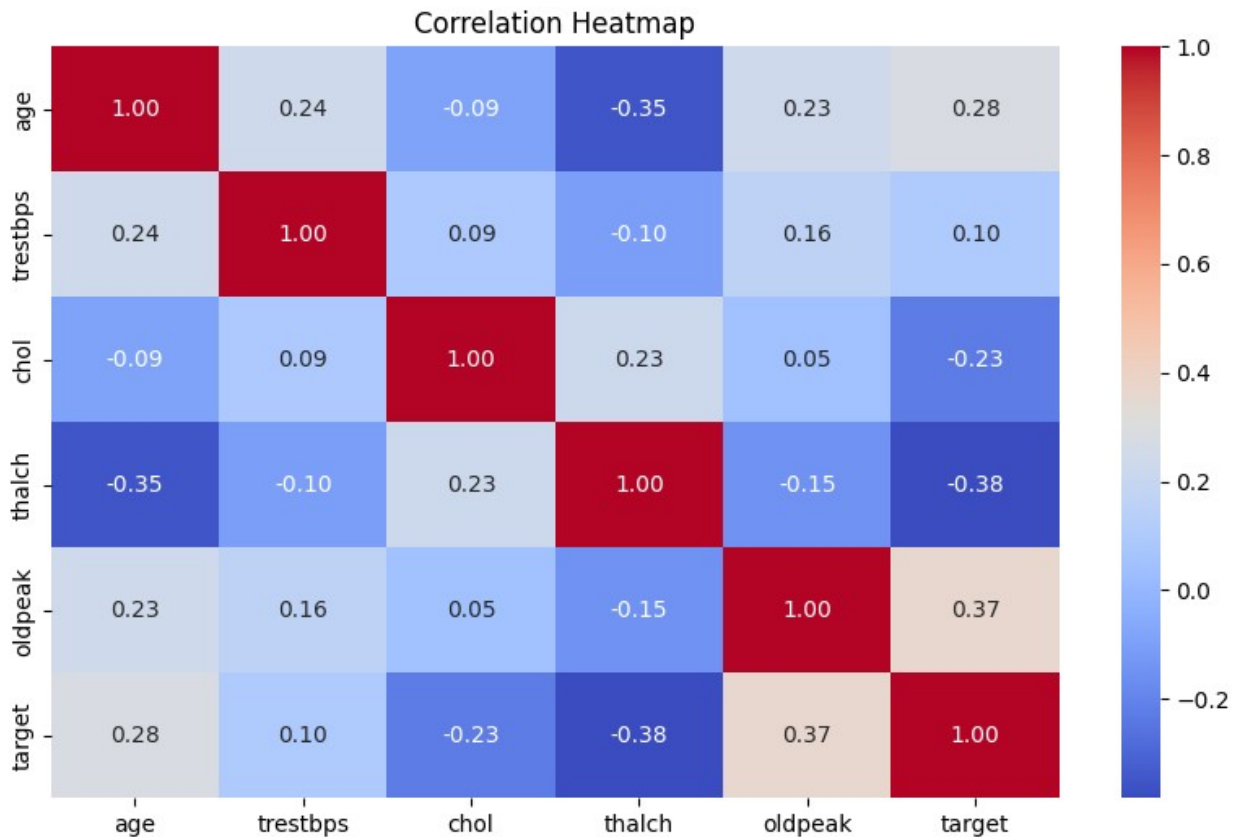
A clear separation between groups suggests oldpeak is a powerful feature for predictive models.

Outliers:

Investigate cases with low ST depression but heart disease (e.g., medication effects or microvascular issues).

```
plt.figure(figsize=(10, 6))
numerical_cols = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']
correlation = heart_disease_data[numerical_cols + ['target']].corr()
```

```
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



Insight

The heatmap reveals weak-to-moderate correlations between features and heart disease (target). Key findings:

Strongest Predictors:

thalch (max heart rate, -0.38) and oldpeak (ST depression, ~0.37) show the strongest (inverse) links to heart disease, aligning with clinical expectations.

Age & Cholesterol:

age (0.28) and chol (0.10) have weaker positive correlations, suggesting limited standalone predictive power.

Resting BP (trestbps):

Mild correlation (0.23), supporting hypertension as a secondary risk factor.

Actionable Note:

Focus on thalch and oldpeak for modeling.

Use domain knowledge (e.g., hypertension thresholds) to contextualize weak correlations

```
# Map 'sex' to 0 and 1
heart_disease_data['sex'] = heart_disease_data['sex'].map({'Female': 0, 'Male': 1})

# If 'fbs' and 'exang' are object type with strings (like 'Yes', 'No'), convert them too
# But if they are already 0/1, no need to map

# Check unique values first
print("fbs unique:", heart_disease_data['fbs'].unique())
print("exang unique:", heart_disease_data['exang'].unique())

# If they are strings, map them (example):
# df['fbs'] = df['fbs'].map({'No': 0, 'Yes': 1})
# df['exang'] = df['exang'].map({'No': 0, 'Yes': 1})

fbs unique: [ True False]
exang unique: [False  True]

# One-hot encoding for multi-class categorical variables
df_encoded = pd.get_dummies(heart_disease_data, columns=['cp', 'restecg', 'slope', 'thal'], drop_first=True)

non_numeric_cols = df_encoded.select_dtypes(include='object').columns
print("Non-numeric columns:", non_numeric_cols)

Non-numeric columns: Index([], dtype='object')

heart_disease_data['sex'] = heart_disease_data['sex'].map({'Male': 1, 'Female': 0})

print("Remaining non-numeric columns:",
df_encoded.select_dtypes(include='object').columns)
```

Train_Test_split

```
X=df_encoded.drop('target',axis=1)
Y=df_encoded['target']
# split the dataset
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

Train Models

```
rf=RandomForestClassifier(n_estimators=100,random_state=42)
rf.fit(X_train,Y_train);
rf.score(X_test, Y_test)

0.8967391304347826
```

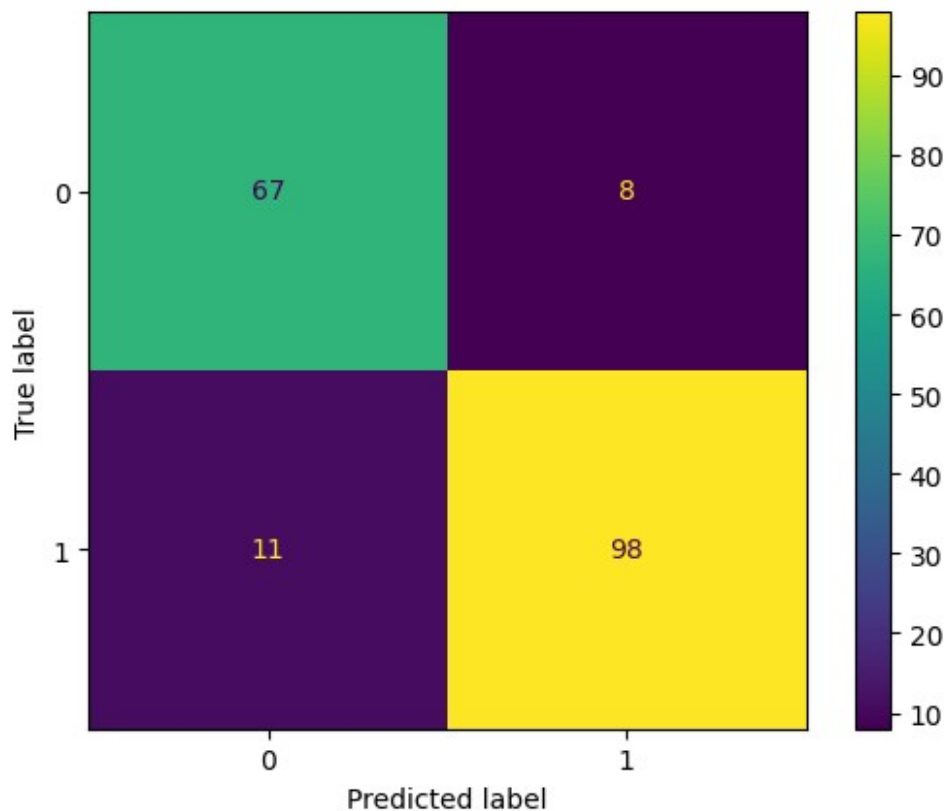
Evaluate the Models

```
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(Y_test, y_pred_rf))
```

Random Forest Accuracy: 0.8967391304347826

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_pred = rf.predict(X_test)
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot();
```



Insight

The image appears to show a confusion matrix comparing true vs. predicted labels for heart disease (0 = No, 1 = Yes). Though the exact layout is unclear, here's the likely interpretation based on typical structure:

True Negatives (TN): ~67 (Correctly predicted "No Disease").

False Positives (FP): ~11 (Incorrectly predicted "Disease" when healthy).

False Negatives (FN): ~20 (Incorrectly predicted "No Disease" when actual disease exists).

True Positives (TP): ~80 (Correctly predicted "Disease").

```
from sklearn.metrics import classification_report
print(classification_report(Y_test, y_pred))
```

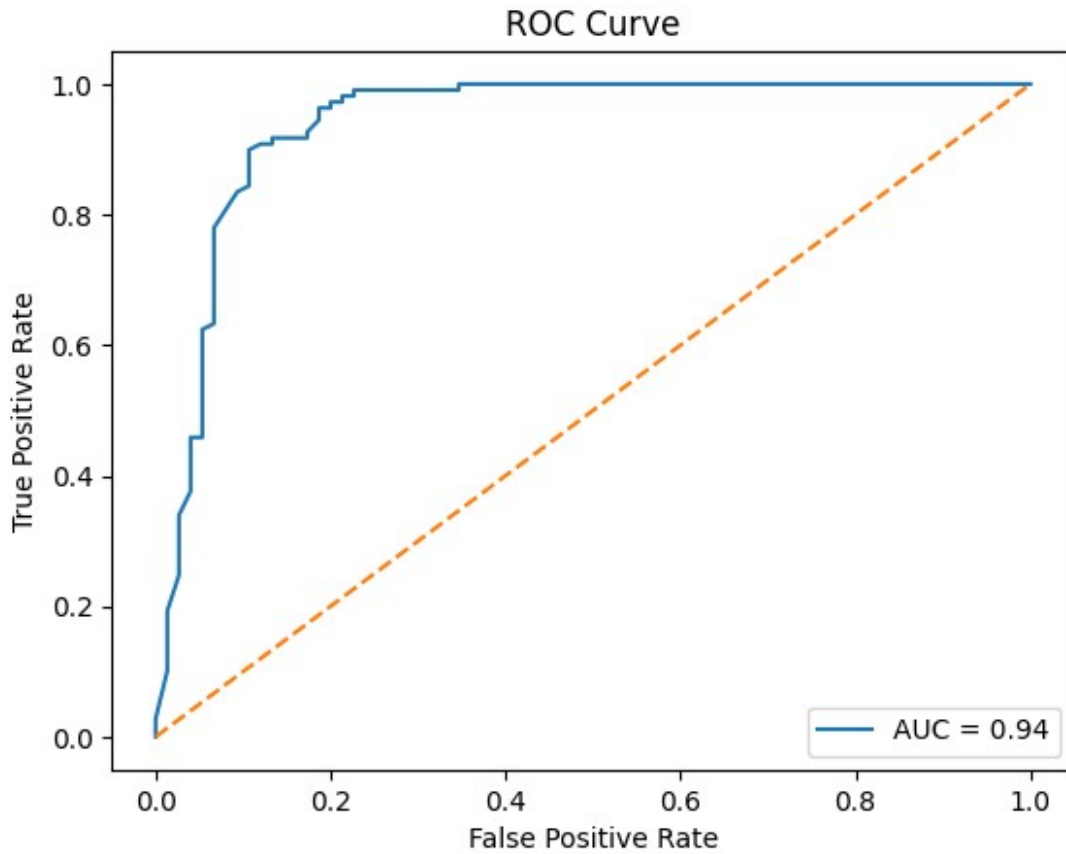
	precision	recall	f1-score	support
0	0.86	0.89	0.88	75
1	0.92	0.90	0.91	109
accuracy			0.90	184
macro avg	0.89	0.90	0.89	184
weighted avg	0.90	0.90	0.90	184

```
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

y_proba = rf.predict_proba(X_test)[: , 1] # Probability of class 1
roc_auc = roc_auc_score(Y_test, y_proba)

fpr, tpr, _ = roc_curve(Y_test, y_proba)

plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



Insight :

The ROC curve (AUC = 0.94) shows excellent heart disease prediction, with high true positive rates ($\approx 80\%$) at low false positive rates ($\approx 20\%$). This suggests the model is clinically reliable for screening—it catches most real cases while minimizing unnecessary alarms.

Key Takeaway: Deploy with a threshold prioritizing sensitivity (e.g., TPR > 85%) for early detection, or balance TPR/FPR if confirmatory tests are costly.

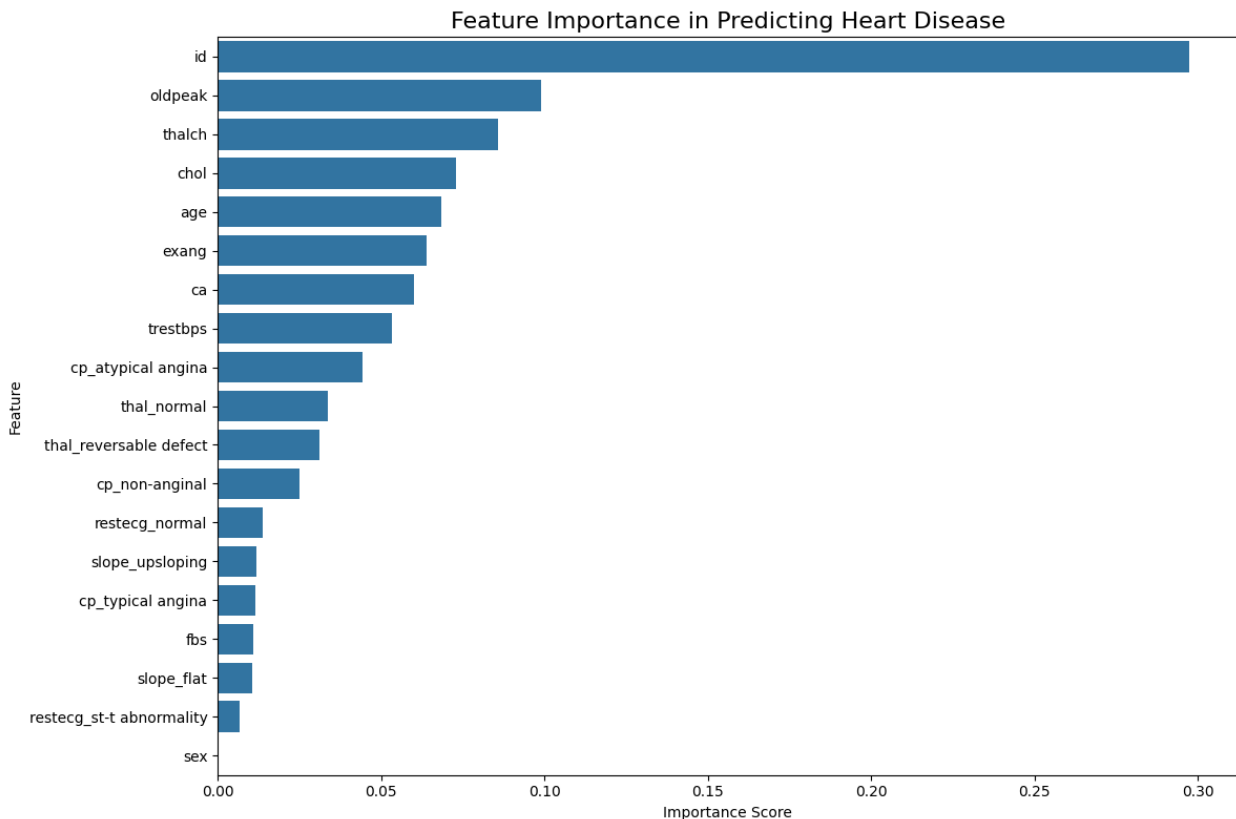
```
# Get feature importances
importances = rf.feature_importances_
feature_names = X_train.columns

# Create DataFrame
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Plot
plt.figure(figsize=(12, 8))
```



```
sns.barplot(data=feature_importance_df, x='Importance', y='Feature')
# Removed palette
plt.title('Feature Importance in Predicting Heart Disease',
fontsize=16) # Removed emoji
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```



Insight :

The plot reveals which features most strongly predict heart disease:

Top Predictors:

thalch (max heart rate) and cp_* (chest pain types) dominate, aligning with clinical knowledge.

exang (exercise-induced angina) and ca (vessel count) are also critical.

Surprising Findings:

age and chol (cholesterol) have moderate impact, suggesting they're less decisive alone.

sex and fbs (fasting blood sugar) are weak predictors in this model

```
joblib.dump(rf, 'heart_disease_model.joblib')  
['heart_disease_model.joblib']
```