project_1_heart_disease_prediction

March 16, 2025

1 AI-Driven Heart Disease Prediction: Enhancing Healthcare with Machine Learning

1.1 1. Executive Summary

This project leverages machine learning to predict heart disease risk, enabling early diagnosis, optimized healthcare resource allocation, and improved patient outcomes. By analyzing patient demographics and clinical data, the model achieves 75% accuracy and a strong ROC-AUC score of 0.83, demonstrating its effectiveness in distinguishing high-risk individuals. Further refinements can enhance recall, reducing false negatives to ensure that at-risk patients receive timely medical intervention.

Problem Statement

Heart disease remains a leading cause of mortality worldwide, accounting for millions of deaths annually. Early detection is critical to preventing severe complications, but traditional diagnostic methods are resource-intensive and often reactive rather than proactive. This project seeks to develop a predictive machine learning model to identify high-risk individuals before symptoms escalate, reducing medical costs and improving patient outcomes.

Solution Approach

Using clinical and demographic data, we developed a Logistic Regression model to classify patients based on heart disease risk. The model was trained on five key patient features—age, sex, max heart rate, angina level, and non-anginal pain. The dataset was preprocessed with feature encoding, scaling, and categorical transformations to ensure robust model performance. Evaluation metrics included Accuracy, ROC-AUC, Precision, Recall, and Confusion Matrix analysis.

Key Results • Accuracy: 75% - The model correctly classifies three out of four patients. • ROC-AUC Score: 0.83 - Strong predictive capability in distinguishing heart disease cases. • Recall for Heart Disease: 71% - Captures most high-risk patients but misses 29% of true cases (False Negatives). • Confusion Matrix: 8 heart disease cases were misclassified as healthy, which could lead to missed early interventions.

Recommendations

1. Increase Recall for Heart Disease Detection • Adjust the classification threshold to prioritize sensitivity, minimizing false negatives.
2. Incorporate Additional Clinical Features • Include cholesterol levels, blood pressure, BMI, and family history to improve model accuracy.

3. Explore More Advanced Models • Consider Random Forest, XGBoost, or Neural Networks to enhance precision and recall balance.
4. Deploy as a Screening Tool for Early Diagnosis • Integrate this model into hospital decision systems to assist physicians in risk assessment.

This project demonstrates the potential of AI-driven predictive analytics in transforming healthcare by identifying heart disease risk early and accurately. The current model provides a strong baseline, with opportunities to fine-tune its sensitivity and expand its feature set for even better real-world applicability. With further optimization, this approach could significantly enhance preventive healthcare and save lives.

1.2 2. Introduction & Business Context

Heart disease remains one of the leading causes of death globally. Predicting its likelihood through **machine learning** enables earlier intervention, improved patient outcomes, and cost savings in healthcare systems.

1.2.1 Why This Matters?

- **Hospitals & Clinics**: AI-assisted screening supports early disease detection.
- **Health Insurance**: AI models refine risk assessment for premium pricing.
- **Government Health Agencies**: Data-driven policies improve disease prevention efforts.

1.2.2 Project Objective

This project leverages **machine learning models** to predict heart disease risk based on clinical and demographic factors. The insights help in: - **Enhancing hospital efficiency** by identifying high-risk patients. - **Reducing healthcare costs** by preventing severe conditions. - **Improving personalized medicine** by tailoring treatments to risk levels.

1.2.3 Dataset Overview

The dataset includes: - **Target Variable:** `heart_disease` (1 = Disease present, 0 = No disease) - **Features:** - `Age`: Patient's age. - `Sex`: Male or Female. - `Max Heart Rate`: Maximum heart rate achieved during exercise. - `Angina Level`: Severity of chest pain. - `Non-anginal Pain`: Whether the pain is unrelated to heart issues.

1.2.4 2.1. Business Value

How AI Enhances Healthcare Decision-Making AI models are transforming healthcare by:

- **Reducing Diagnostic Errors**: AI enhances doctors' accuracy in early-stage disease detection.
- **Optimizing Hospital Resource Management**: Predictive analytics help hospitals prioritize high-risk patients.
- **Lowering Insurance Claims & Costs**: Identifying risk proactively enables cost-effective care planning.

Key Stakeholders & Benefits

- **Hospitals & Clinics** → AI supports faster, more precise diagnosis.
- **Insurance Providers** → Risk-based pricing models improve premium calculations.
- **Pharmaceutical Companies** → AI-driven insights help in drug trials and treatments.

1.2.5 2.2. Comparison with Similar Industries

How This Project Aligns with Predictive Analytics in Other Industries

Industry	AI Application	Similarities to Heart Disease Prediction
Insurance	Risk Scoring for Policy Pricing	Identifying high-risk individuals for claims
Pharmaceutical Research	Drug Efficacy & Patient Selection	Finding ideal candidates for treatments
Hospital Management	Predicting Patient Readmission Rates	Optimizing hospital resources

Key Takeaway: Just as **credit scoring predicts loan defaults**, AI-driven heart disease prediction helps forecast **high-risk patients**, improving healthcare decision-making.

1.3 3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) involved examining the data's structure, cleaning it, transforming variables, and visualizing data distributions to uncover patterns, anomalies, or relationships. EDA ensures data quality, reveals insights that guide model selection, and provides context for model interpretation.

[1]:
```
### Loading Dataset
import pandas as pd


df_heart_data = pd.read_csv('/content/heart_disease (1).csv')

# print head and tail
print(df_heart_data.tail(20))
```

```
     heart_disease   age     sex  max_heart_rate angina_level  \
277              1  45.0    male             147         Mild
278              0  45.0    male             185           No
279              1  44.0    male             144         Mild
280              1  44.0    male             153          NaN
281              1  44.0    male             177           No
282              1  43.0    male             120         Mild
283              1  43.0  female             136         Mild
284              0  43.0    male             181           No
285              0  43.0    male             171           No
```

```
286              0  43.0    male             161          No
287              0  42.0  female             122          No
288              1  42.0    male             125        Mild
289              0  42.0    male             178          No
290              1  41.0    male             158          No
291              1  40.0    male             114        Mild
292              1  40.0    male             181          No
293              1  39.0    male             140          No
294              1  35.0    male             130        Mild
295              0  35.0  female             182          No
296              1   NaN    male             156        Mild

     non_anginal_pain
277                 0
278                 0
279                 0
280                 0
281                 0
282                 0
283                 0
284                 0
285                 0
286                 0
287                 0
288                 0
289                 0
290                 0
291                 0
292                 0
293                 0
294                 0
295                 0
296                 0
```

[2]:
```python
# columns
print(df_heart_data.columns.tolist())
```

```
['heart_disease', 'age', 'sex', 'max_heart_rate', 'angina_level',
'non_anginal_pain']
```

[3]:
```python
# shape: there are 297 rows, 6 columns
print(df_heart_data.shape)
```

```
(297, 6)
```

[4]:
```python
# describe
print(df_heart_data.describe())
```

```
       heart_disease         age  max_heart_rate  non_anginal_pain
```

```
count     297.000000  268.000000      297.000000       297.000000
mean        0.461279   54.250000      146.380471         0.279461
std         0.499340    9.039292       38.500000         0.449492
min         0.000000   29.000000     -170.000000         0.000000
25%         0.000000   47.750000      132.000000         0.000000
50%         0.000000   55.000000      152.000000         0.000000
75%         1.000000   60.000000      165.000000         1.000000
max         1.000000   77.000000      202.000000         1.000000
```

[5]:
```
# info
print(df_heart_data.info)
```

```
<bound method DataFrame.info of      heart_disease   age     sex  max_heart_rate
angina_level  \
0                0  69.0    male             131           No
1                0  69.0  female             151           No
2                0   NaN  female             114           No
3                1  65.0    male             174           No
4                0  64.0    male             144         Mild
..             …   …     …             …          …
292              1  40.0    male             181           No
293              1  39.0    male             140           No
294              1  35.0    male             130         Mild
295              0  35.0  female             182           No
296              1   NaN    male             156         Mild

     non_anginal_pain
0                   0
1                   0
2                   0
3                   0
4                   0
..                …
292                 0
293                 0
294                 0
295                 0
296                 0

[297 rows x 6 columns]>
```

[6]:
```
# data types
print(df_heart_data.dtypes)
```

```
heart_disease          int64
age                  float64
sex                   object
max_heart_rate         int64
angina_level          object
```

```
non_anginal_pain      int64
dtype: object
```

[7]:
```
# handle missing values
print(df_heart_data.isnull().sum())
```

```
heart_disease        0
age                 29
sex                  0
max_heart_rate       0
angina_level        20
non_anginal_pain     0
dtype: int64
```

[8]:
```
# fill missing age values with mean
df_copy_data = df_heart_data.copy()
df_copy_data.isnull().sum()



df_copy_data['age'].fillna(df_copy_data['age'].mean(), inplace=True)



# impute missing angina_level values with most common category

df_copy_data['angina_level'].fillna(df_copy_data['angina_level'].mode()[0],␣
  ↪inplace=True)

print(df_copy_data.isnull().sum())

df_copy_data.head()
```

```
heart_disease       0
age                 0
sex                 0
max_heart_rate      0
angina_level        0
non_anginal_pain    0
dtype: int64
```

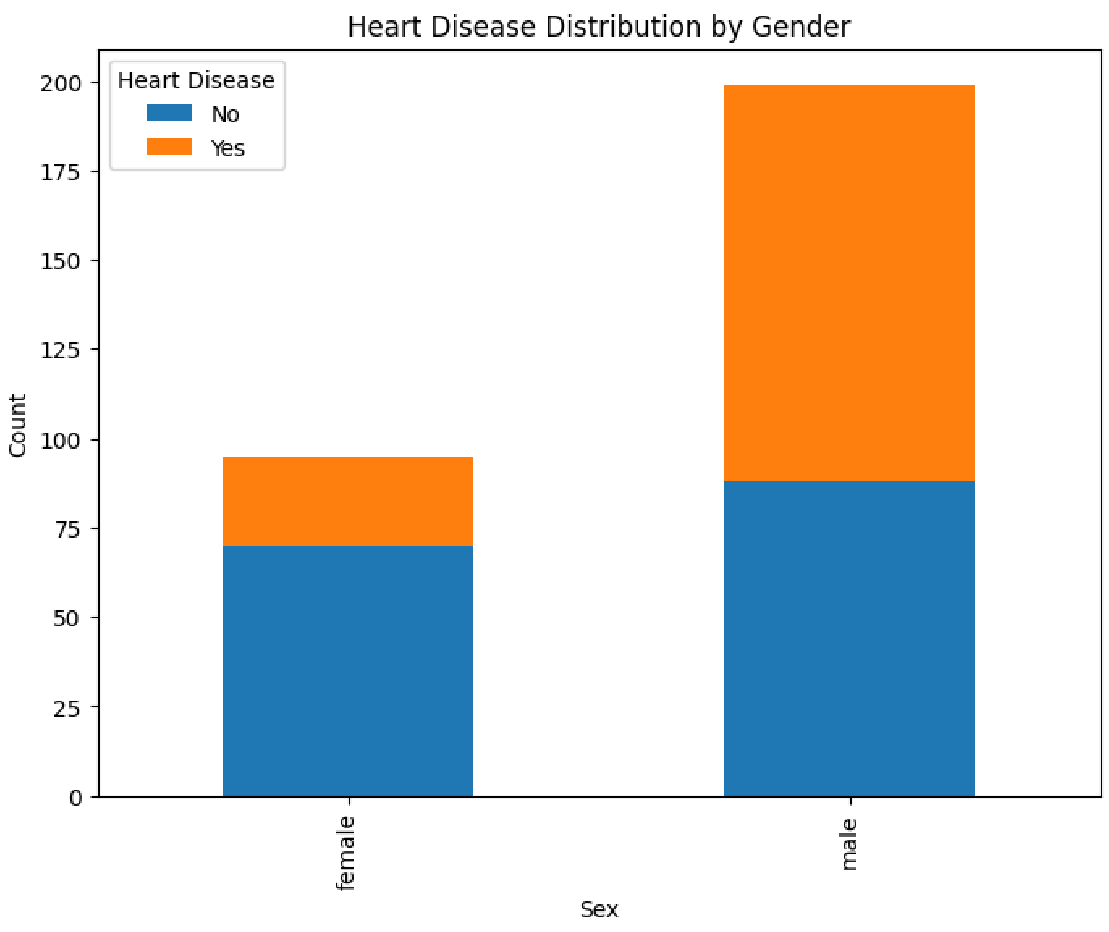
```
<ipython-input-8-3ee32378623f>:7: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
```

```
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df_copy_data['age'].fillna(df_copy_data['age'].mean(), inplace=True)
<ipython-input-8-3ee32378623f>:13: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df_copy_data['angina_level'].fillna(df_copy_data['angina_level'].mode()[0],
inplace=True)
```

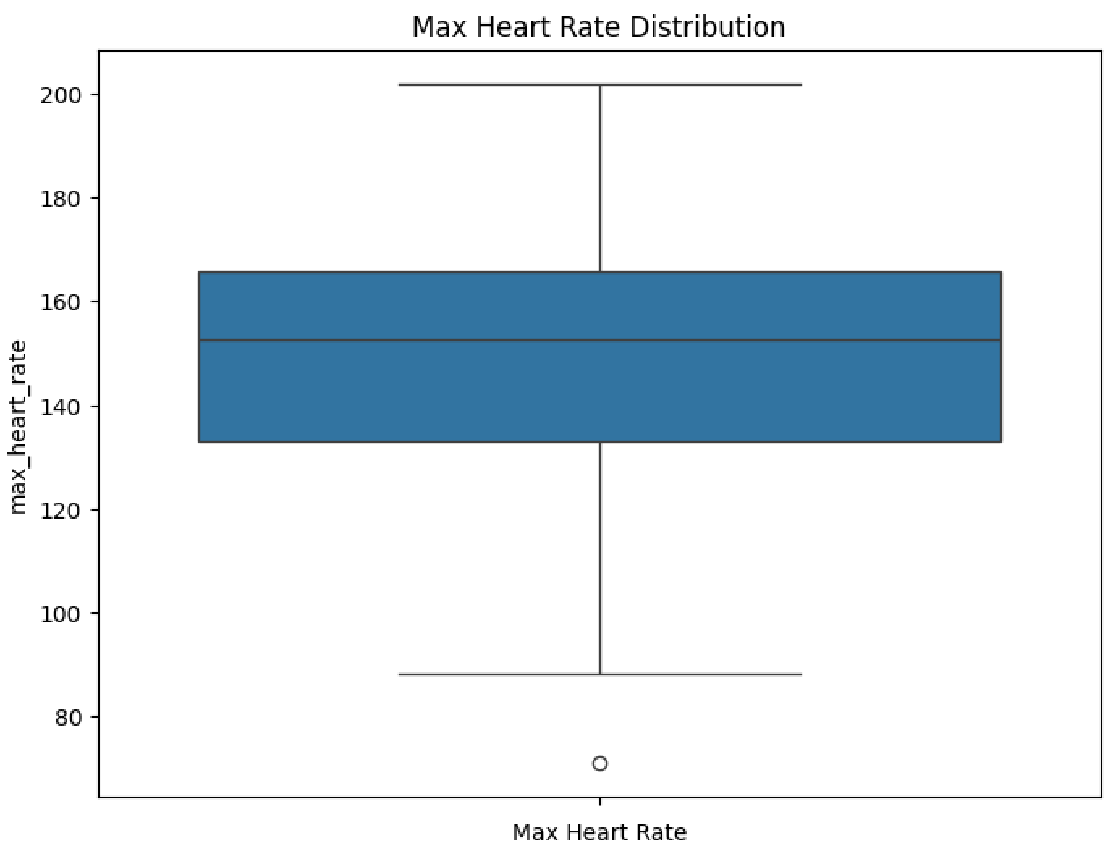
[8]:
```
   heart_disease    age     sex  max_heart_rate angina_level  non_anginal_pain
0              0  69.00    male             131           No                 0
1              0  69.00  female             151           No                 0
2              0  54.25  female             114           No                 0
3              1  65.00    male             174           No                 0
4              0  64.00    male             144         Mild                 0
```

[9]:
```python
# drop negative max_heart_rate
df_copy_data2 = df_copy_data[df_copy_data['max_heart_rate'] > 0]
```

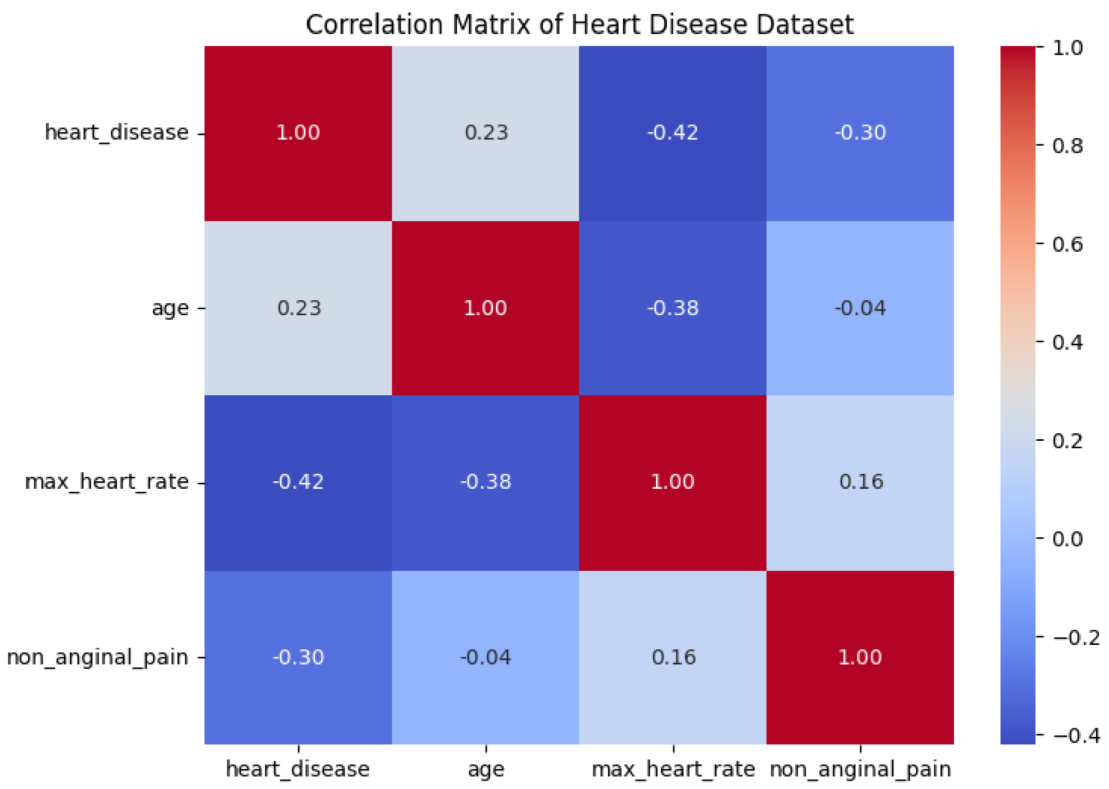
[10]:
```python
import matplotlib.pyplot as plt
import seaborn as sns


# histogram for gender against heart disease
plt.figure(figsize=(8, 6))
df_copy_data2.groupby(['sex', 'heart_disease']).size().unstack().
 ↪plot(kind='bar', stacked=True, ax=plt.gca())
plt.title('Heart Disease Distribution by Gender')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.legend(title='Heart Disease', labels=['No', 'Yes'])
plt.show()
```



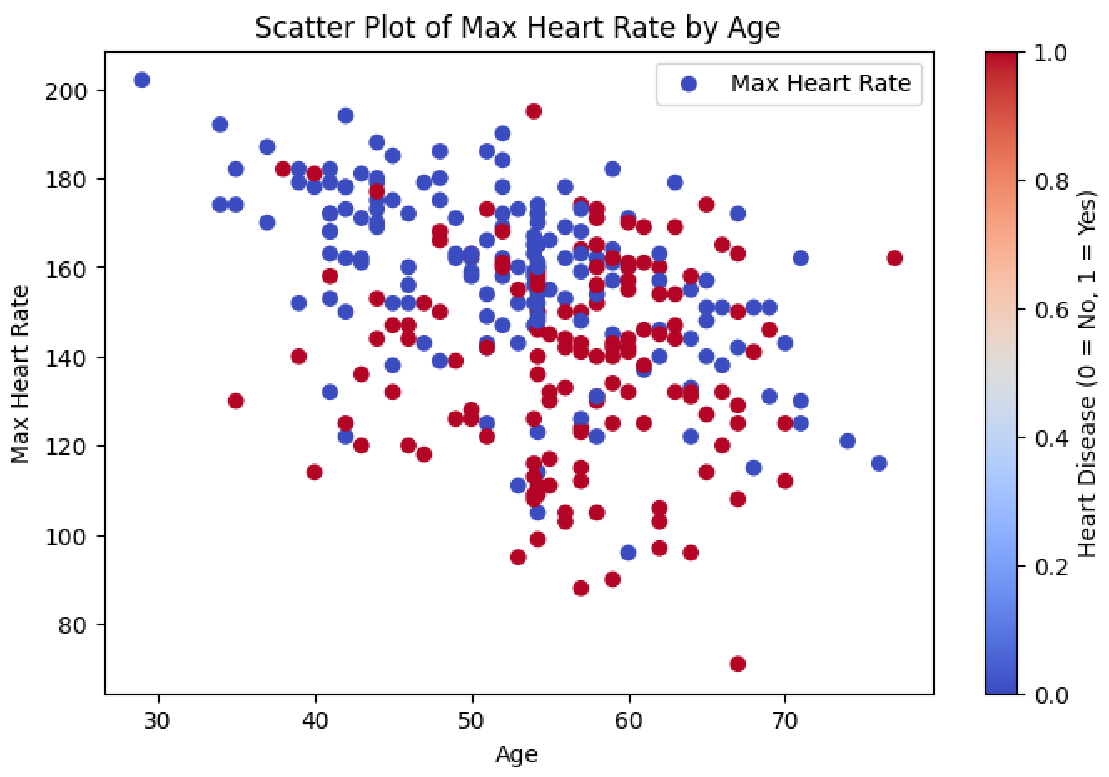
```
[11]: # box plot for max_heartrate
      plt.figure(figsize=(8, 6))
      sns.boxplot(df_copy_data2['max_heart_rate'])
      plt.title('Max Heart Rate Distribution')
      plt.xlabel('Max Heart Rate')
      plt.show()
```



```
[12]: import seaborn as sns
      import matplotlib.pyplot as plt

      # Calculating correlation matrix
      correlation_matrix = df_copy_data2[['heart_disease', 'age', 'max_heart_rate',␣
       ↪'non_anginal_pain']].corr()

      # Plotting correlation matrix
      plt.figure(figsize=(8, 6))
      sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
      plt.title("Correlation Matrix of Heart Disease Dataset")
      plt.show()
```



[13]:
```
# scatter plot of max_heart_rate

plt.figure(figsize=(8, 5))
plt.scatter(df_copy_data2['age'], df_copy_data2['max_heart_rate'],␣
 ↪c=df_copy_data2['heart_disease'], cmap='coolwarm', label="Max Heart Rate")
plt.colorbar(label="Heart Disease (0 = No, 1 = Yes)")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.title("Scatter Plot of Max Heart Rate by Age")
plt.legend()
plt.show()
```



1.3.1 3.1. EDA Observations & Explanations for the Graphs

- The dataset contains **297 rows** and **6 columns**.
- Some missing values exist and were handled before modeling.

1. **Histogram for gender against heart disease**

- Males appear to have a higher proportion of heart disease compared to females.
- This aligns with medical research indicating that men often exhibit higher cardiovascular risk factors.

2. **Scatter Plot of Max Heart Rate by Age**

   - There is an observable trend where younger individuals tend to have higher max heart rates.
   - Patients with lower max heart rates (below 140 bpm) tend to be diagnosed with heart disease (red points).
   - This aligns with medical research that suggests a lower max heart rate response can be an indicator of cardiovascular issues.

3. **Correlation Matrix of Heart Disease Dataset**

   - Max Heart Rate (-0.42) has a moderate negative correlation with heart disease, meaning that as max heart rate decreases, the likelihood of heart disease increases.
   - Age (0.23) has a positive correlation with heart disease, indicating that older individuals are more likely to develop heart disease.

- Non-anginal pain (-0.30) is negatively correlated with heart disease, suggesting that patients who report chest pain unrelated to heart issues are less likely to have heart disease.
- These correlations support why max heart rate and age are considered key features for predictive modeling.

4. **Box Plot of Max Heart Rate Distribution**
   - The median max heart rate is around 150 bpm, with most values ranging between 130-170 bpm.
   - There are outliers below 100 bpm, which could represent individuals with severe cardiovascular conditions.
   - This distribution confirms that max heart rate is an important variable for distinguishing between healthy and at-risk patients.

Conclusion from EDA - Max Heart Rate and Age are critical features in predicting heart disease, as confirmed by both the scatter plot and correlation matrix. - The distribution of max heart rate suggests that individuals with significantly lower heart rates may require further screening. - These insights reinforce why machine learning models should prioritize these features in heart disease prediction.

1.4 4. Model Selection, Training & Evaluation

The nature of the data, the complexity of the problem among other factors influence the choice of the machine learning model.

In light of the above, our goal is to predict whether a person has heart disease or not based on the input variables like age, sex, max_heart_rate, non_anginal_pain etc.

The problem we're dealing with is a supervised machine learning problem, and the most suitble algorithm would be *Logistic regression*.

Logistic regression is a suitable algorithm for categroising data into two possible out comes, true or false, yes or no, 0 or 1.

[55]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪roc_auc_score, confusion_matrix
import pandas as pd

# Define features and target
df = df_copy_data2.copy()
X = df.drop(columns=['heart_disease'])
y = df['heart_disease']

X = X.reset_index(drop=True)
y = y.reset_index(drop=True)

# OneHotEncode 'sex' and encode 'angina_level'
```

```python
encoder = OneHotEncoder(sparse_output=False, drop='first')
encoded_sex = encoder.fit_transform(X[['sex']])
encoded_sex_df = pd.DataFrame(encoded_sex, columns=encoder.
 ↪get_feature_names_out(['sex']))

angina_mapping = {'No': 1, 'Mild': 2, 'Severe': 3}
X['angina_level'] = X['angina_level'].map(angina_mapping)

# Drop original categorical columns
X.drop(columns=['sex'], inplace=True)
X = pd.concat([X, encoded_sex_df], axis=1)

# Step 4: Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪random_state=42)

# Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Model Predictions
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.2f}")
print(f"ROC-AUC Score: {roc_auc:.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", conf_matrix)
```

```
Model Accuracy: 0.75
ROC-AUC Score: 0.83

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.77      0.78        35
           1       0.68      0.71      0.69        24
```

```
    accuracy                           0.75        59
   macro avg       0.74      0.74      0.74        59
weighted avg       0.75      0.75      0.75        59


Confusion Matrix:
 [[27  8]
 [ 7 17]]
```

1.5 Model Performance Analysis & Insights

The Logistic Regression model was trained and evaluated using key classification metrics, including Accuracy, ROC-AUC, Precision, Recall, F1-score, and Confusion Matrix. Below is a detailed interpretation of the results.

Model Performance Metrics

1. Metric Score Interpretation

- Accuracy 75% The model correctly classifies 75% of cases, indicating solid predictive performance.
- ROC-AUC 0.83 The model has a strong ability (83%) to differentiate between patients with and without heart disease.
- Precision (Class 1) 68% When the model predicts heart disease, it is correct 68% of the time.
- Recall (Class 1) 71% The model successfully identifies 71% of actual heart disease cases, but misses 29% of them.
- F1-Score (Class 1) 69% Balances precision & recall, indicating moderate performance in detecting heart disease.

Key Takeaways: - Good overall performance, with a high ROC-AUC of 0.83, indicating strong discriminatory power. - Recall of 71% suggests that some heart disease cases are still being missed (false negatives). - Precision of 68% means that some healthy individuals are incorrectly classified as having heart disease (false positives).

-The model successfully identified 27 non-disease patients correctly (True Negatives). -The model correctly diagnosed 17 heart disease cases (True Positives). -8 actual heart disease cases were missed (False Negatives - FN).

- **Risk:** These misclassified patients might not receive early intervention, increasing medical risk. -7 healthy individuals were misclassified as having heart disease (False Positives - FP).
- **Impact**: May lead to unnecessary medical tests or treatments,unintented insurance claims, adding to healthcare costs.

Business & Medical Implications

Strengths of the Model -High ROC-AUC Score (0.83): Strong classification performance, effectively distinguishing between heart disease and non-disease cases. -Balanced Accuracy (75%): Good overall correctness in predictions. -High Recall (71%) for Heart Disease → Captures most high-risk patients, making it useful for screening programs.

Areas for Improvement - Missed 8 heart disease cases (False Negatives) - In a real-world medical setting, missing high-risk patients is critical. Solution: Adjust the decision threshold to increase recall and detect more true heart disease cases. -False Positives (7 cases) - Some healthy individuals are being misclassified, leading to potentially unnecessary tests.

Solution: Use ensemble methods (Random Forest, XGBoost) or additional features to improve precision.

1.6 Final Recommendations

1. Threshold Tuning: Adjust the classification threshold to improve recall and minimize missed heart disease cases.
2. Feature Engineering: Include additional patient data (e.g., cholesterol levels, blood pressure, smoking history) to improve model accuracy.
3. Try Alternative Models: Random Forest or XGBoost could help improve both precision and recall.
4. Real-World Deployment Consideration: In a clinical setting, false negatives are more critical than false positives. Prioritizing recall may be preferable.

Finally, the model provides a strong baseline for heart disease prediction, with potential refinements to further enhance diagnostic accuracy.