

AI Final Project - Heart Failure

A Detailed & Comprehensive Report

Kiana Ghamsari 400222079

- **Project Title:**
 - "Prediction and Diagnosis of Heart Failure in Individuals Using Machine Learning Techniques"
-

Introduction

- **Objective:**
 - The objective of this project is to develop a machine learning-based model for predicting and diagnosing heart failure in individuals based on various health-related features. The model will be trained using historical data containing patient health information and will aim to classify individuals into two categories: those with heart disease (represented by 1) and those without heart disease (represented by 0). The project will explore multiple machine learning algorithms, evaluate their performance, and select the best model for accurate diagnosis.

Data Review

Dataset Description

The dataset used in this project contains multiple health-related features of individuals, including demographic information, clinical test results, and lifestyle-related attributes. The target variable, **HeartDisease**, is a binary indicator where:

- 1 represents individuals diagnosed with heart disease
- 0 represents individuals without heart disease (normal)

Loading the Dataset

The dataset was loaded into a Pandas DataFrame, and the first step involved checking its structure and integrity. The `info()` function provided a summary of the dataset, revealing that it contains **no missing values**, ensuring the data is complete and suitable for analysis.

Features and Target Variable

The dataset consists of **11 features** categorized as **numerical and categorical** variables, along with one target variable (**HeartDisease**).

Numerical Features

These features represent continuous or discrete numerical values related to patient health indicators:

- **Age**: Age of the individual
- **RestingBP**: Resting blood pressure (in mmHg)
- **Cholesterol**: Serum cholesterol level (in mg/dL)
- **FastingBS**: Fasting blood sugar level (1 if fasting blood sugar > 120 mg/dL, 0 otherwise)
- **MaxHR**: Maximum heart rate achieved during exercise
- **Oldpeak**: ST depression induced by exercise relative to rest

Categorical Features

These features represent non-numeric health-related attributes:

- **Sex**: Gender of the individual (**M** for male, **F** for female)
- **ChestPainType**: Type of chest pain (**ATA**, **NAP**, **ASY**, **TA**)
- **RestingECG**: Resting electrocardiogram results (**Normal**, **ST**, **LVH**)
- **ExerciseAngina**: Presence of exercise-induced angina (**Y** for yes, **N** for no)
- **ST_Slope**: Slope of the peak exercise ST segment (**Up**, **Flat**, **Down**)

Target Variable

- **HeartDisease**: The dependent variable indicating whether the individual has heart disease (**1**) or not (**0**).

Checking for Missing Values

To ensure the dataset is complete and does not require imputation, the number of missing values was checked for each column. The results showed that there were **no missing values**, meaning the dataset is well-structured and ready for preprocessing without additional data cleaning.

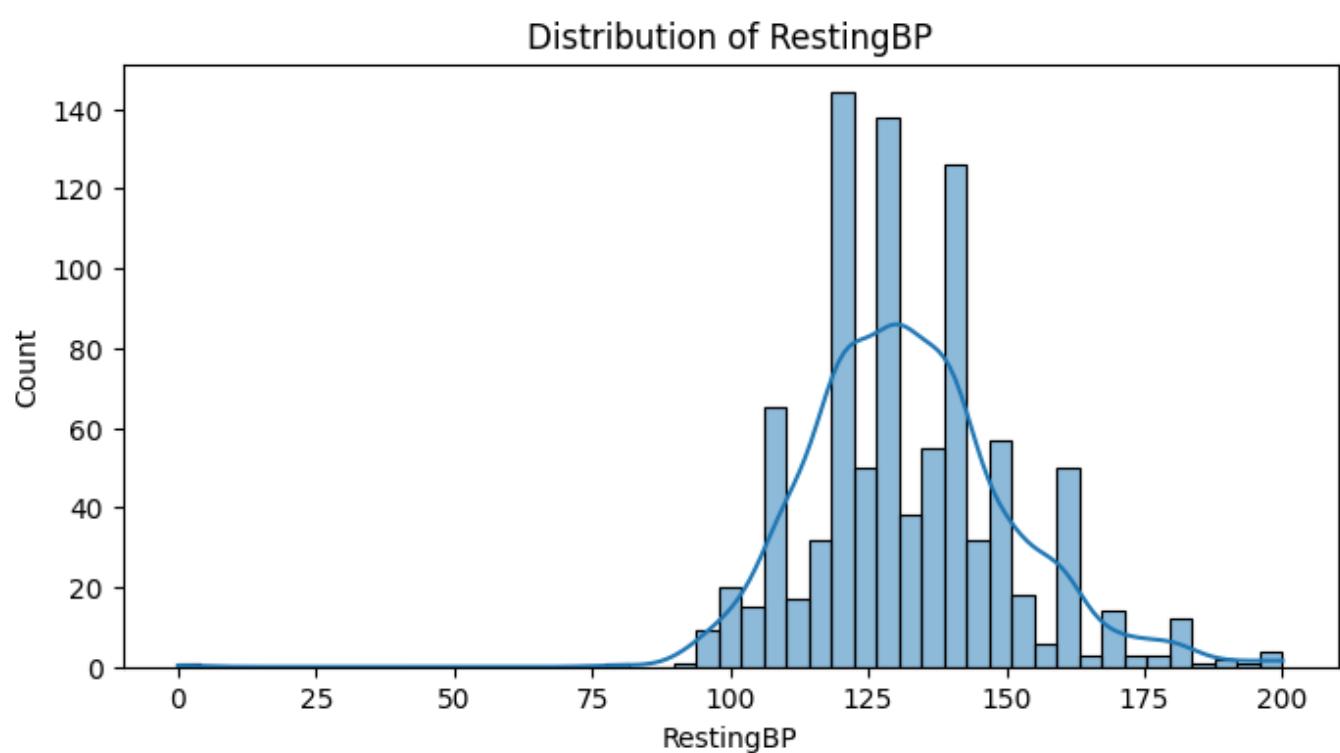
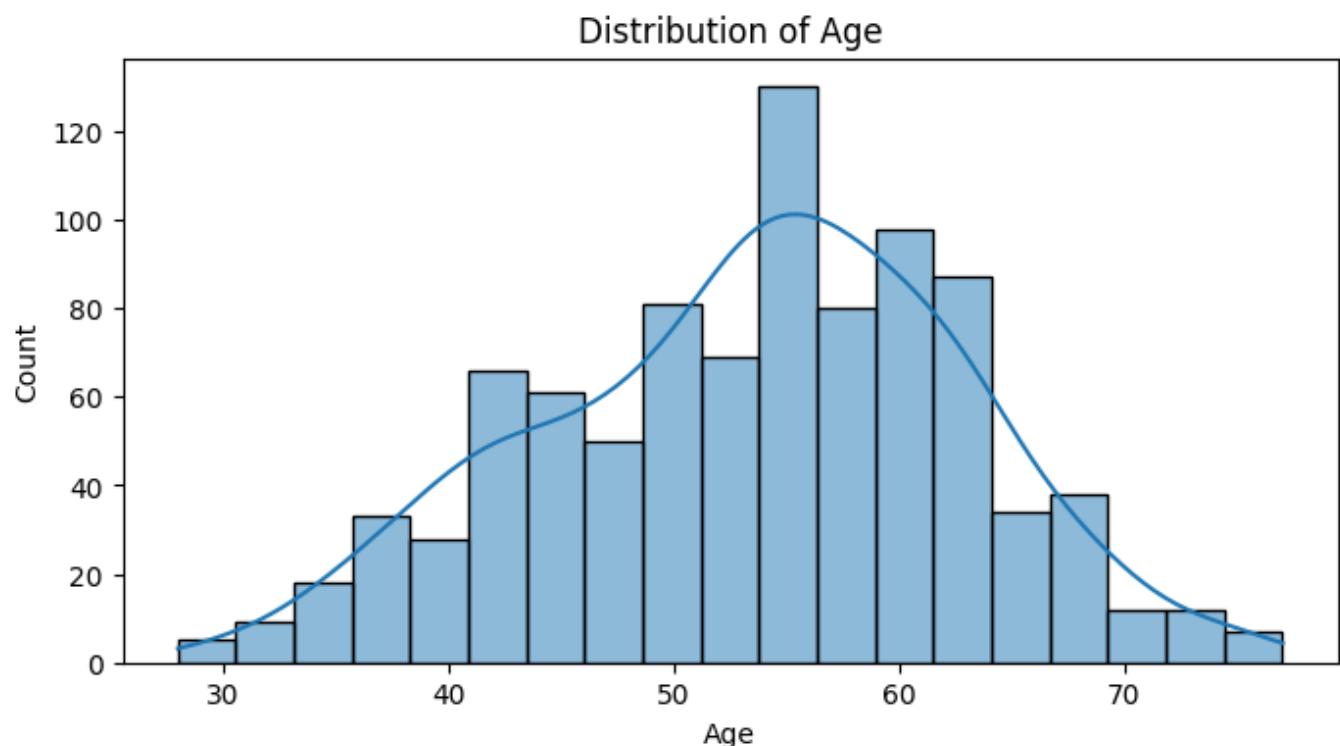
```
[ \begin{array}{|c|c|} \hline \textbf{Feature} & \textbf{Missing Values} \\ \hline Age & 0 \\ Sex & 0 \\ ChestPainType & 0 \\ RestingBP & 0 \\ Cholesterol & 0 \\ FastingBS & 0 \\ RestingECG & 0 \\ MaxHR & 0 \\ ExerciseAngina & 0 \\ Oldpeak & 0 \\ ST\_Slope & 0 \\ HeartDisease & 0 \\ \hline \end{array} ]
```

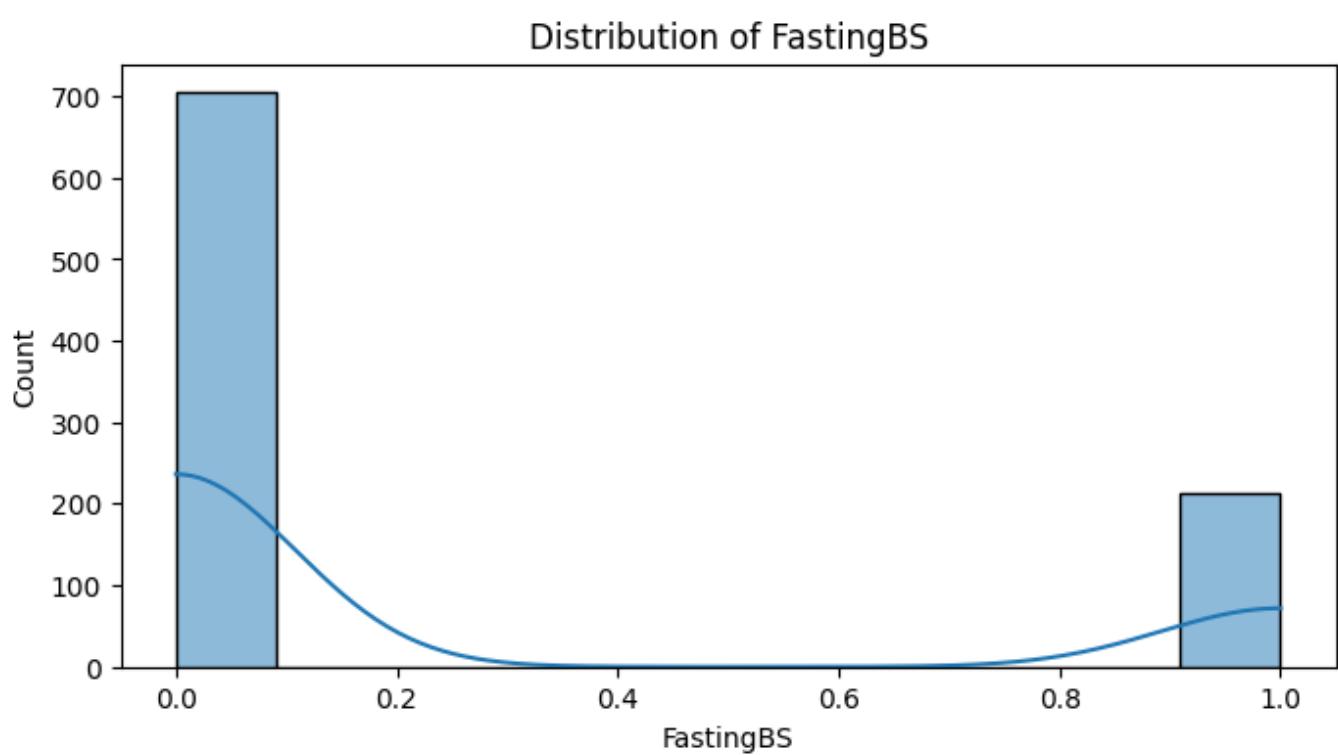
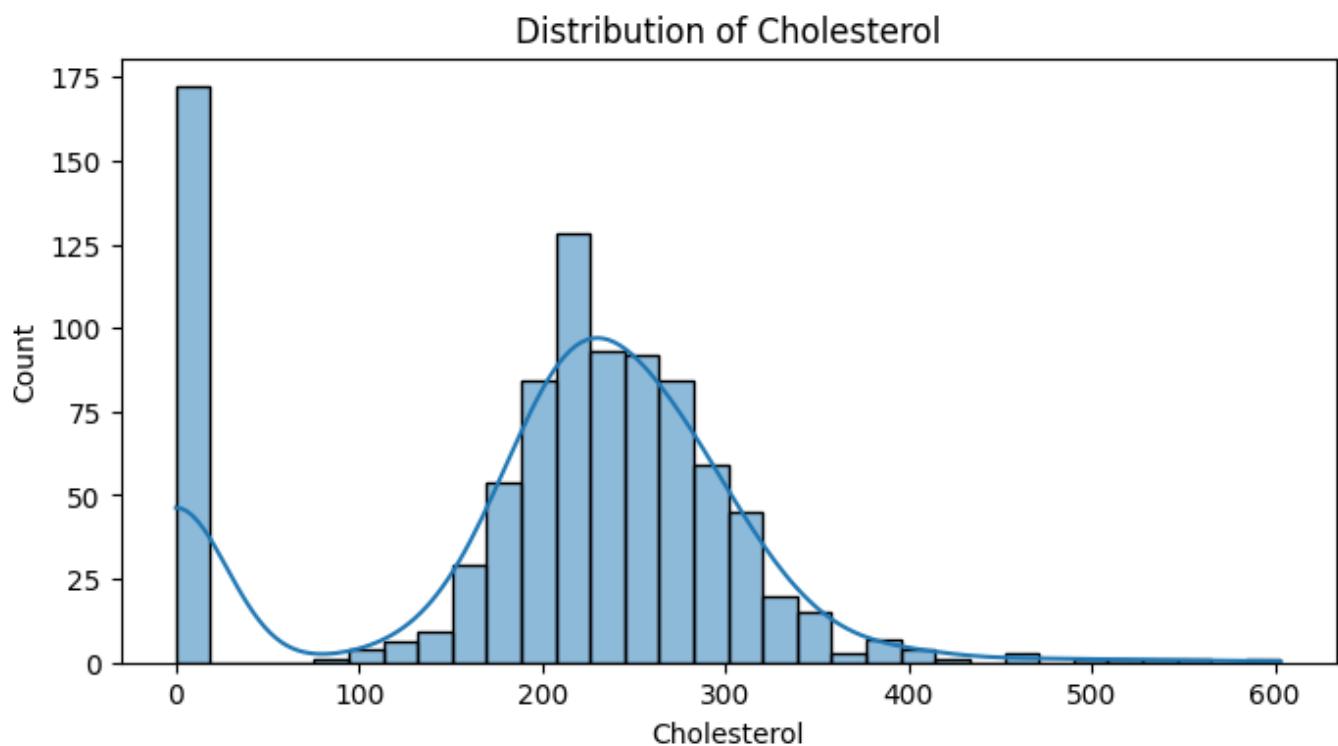
- **Fortunately, there is no missing value!**
- The dataset contains **no missing values**, ensuring a clean and complete dataset for analysis.
- The dataset consists of **11 input features** categorized into numerical and categorical attributes, along with the **binary target variable (HeartDisease)**.

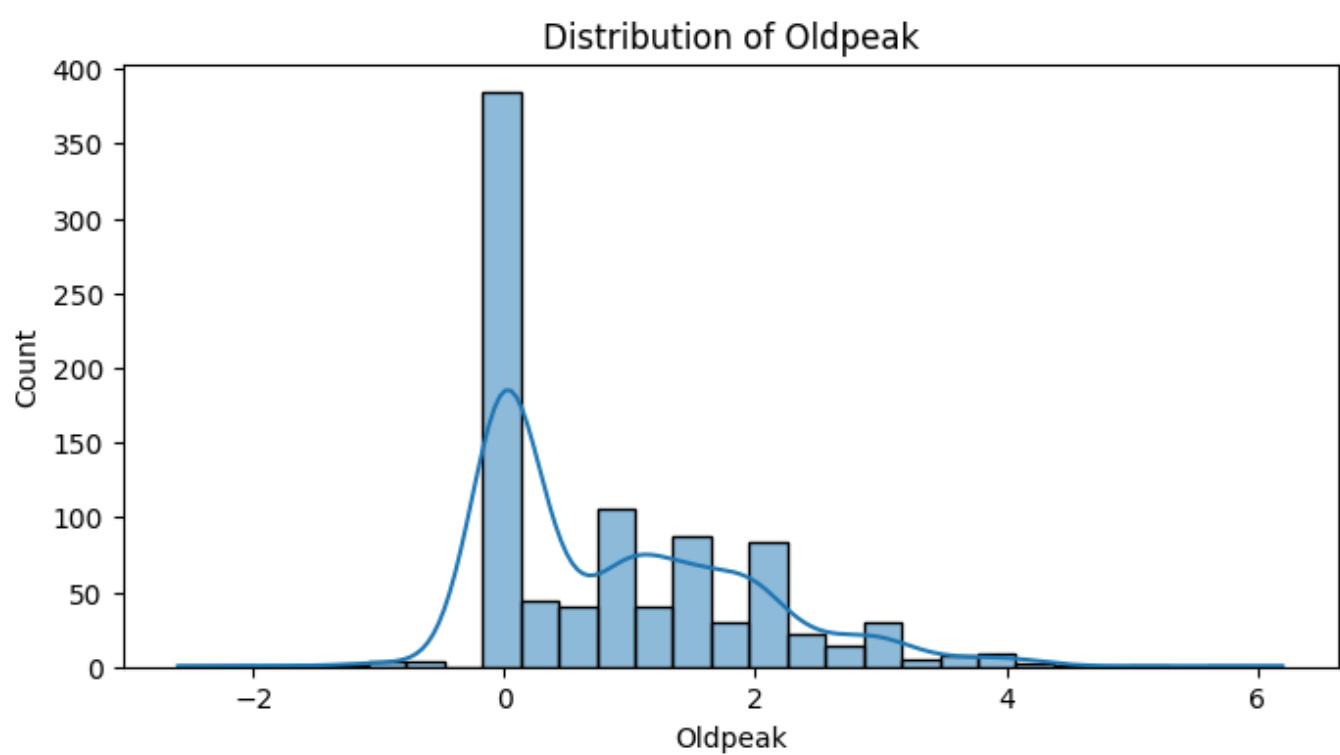
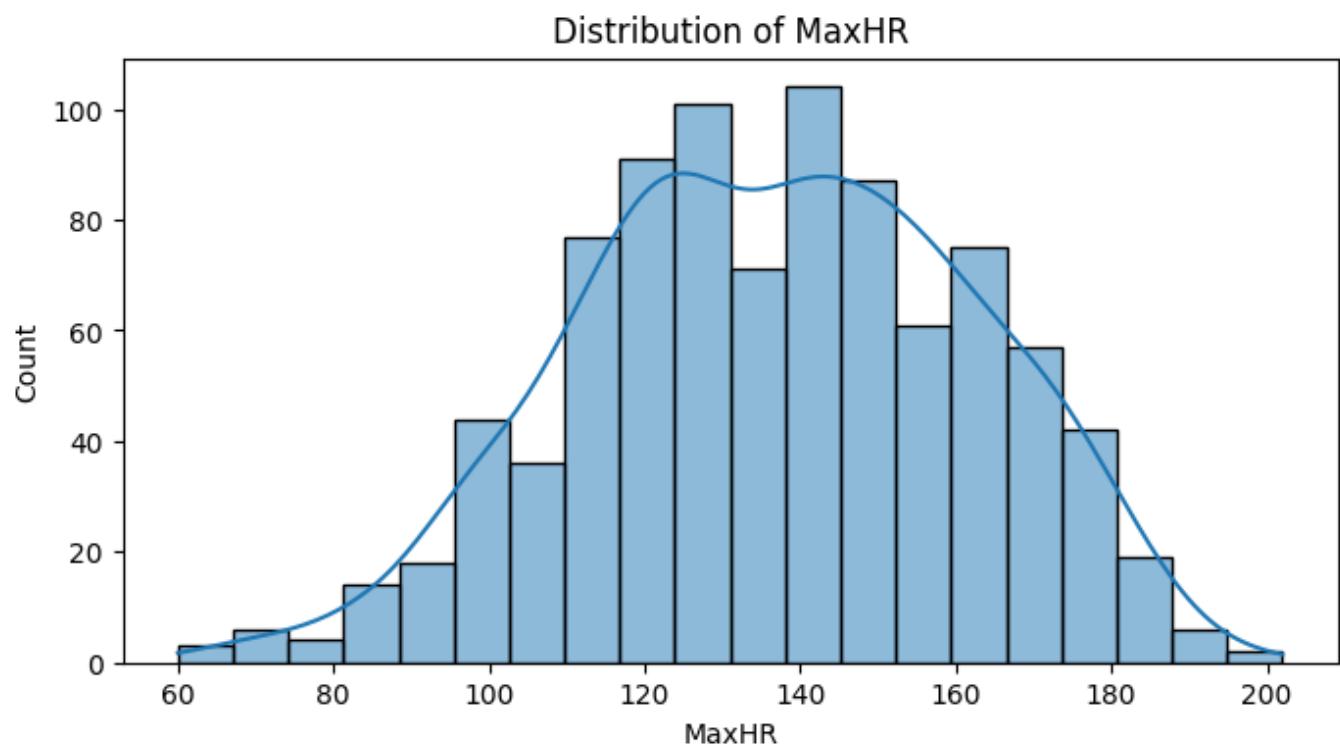
Data Distribution Analysis

Histogram Analysis for Numerical Features

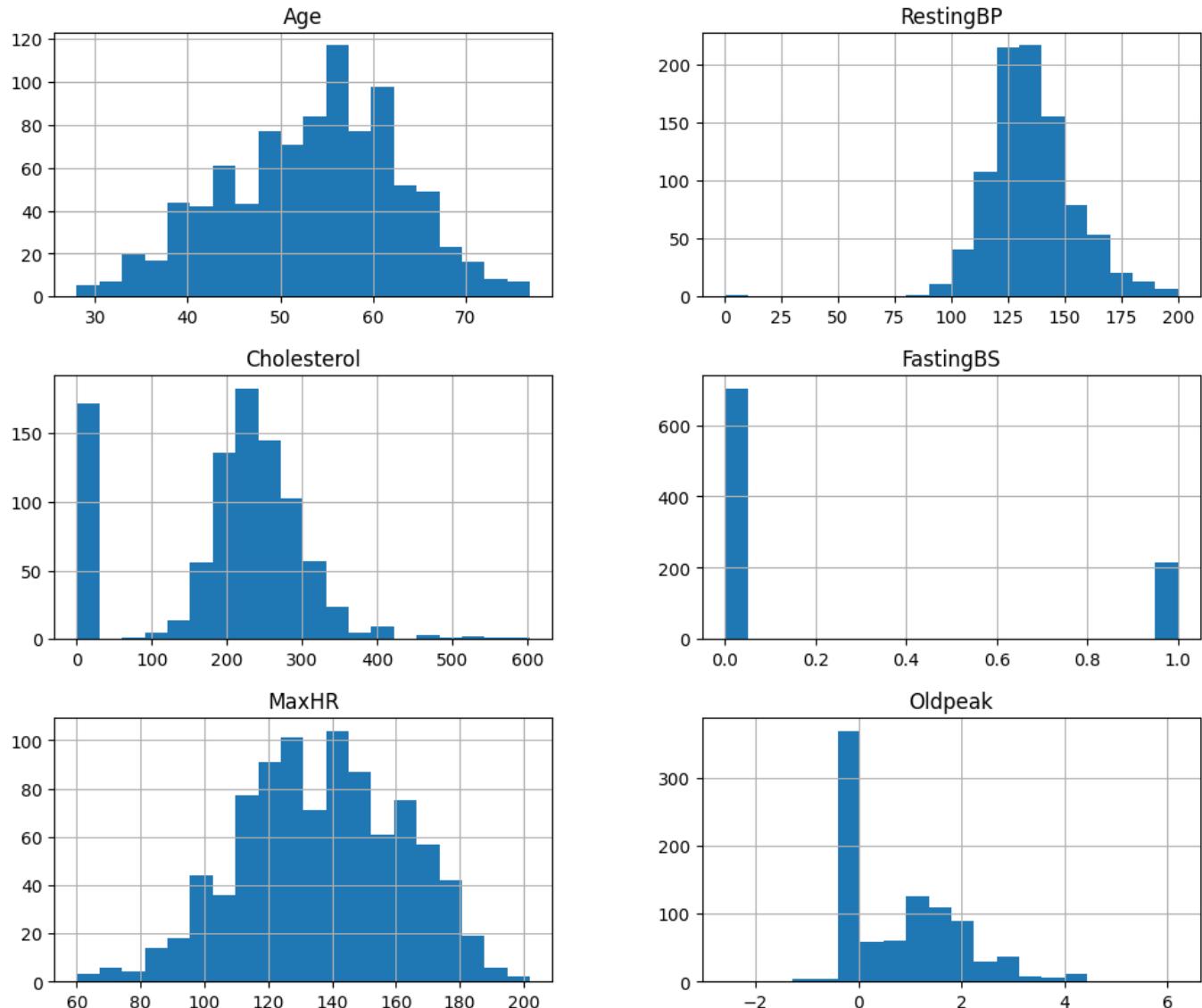
To visualize the distribution of numerical features, **histograms with Kernel Density Estimation (KDE)** have been plotted. This provides insights into the spread and skewness of each numerical variable.







Distribution of All Numerical Features

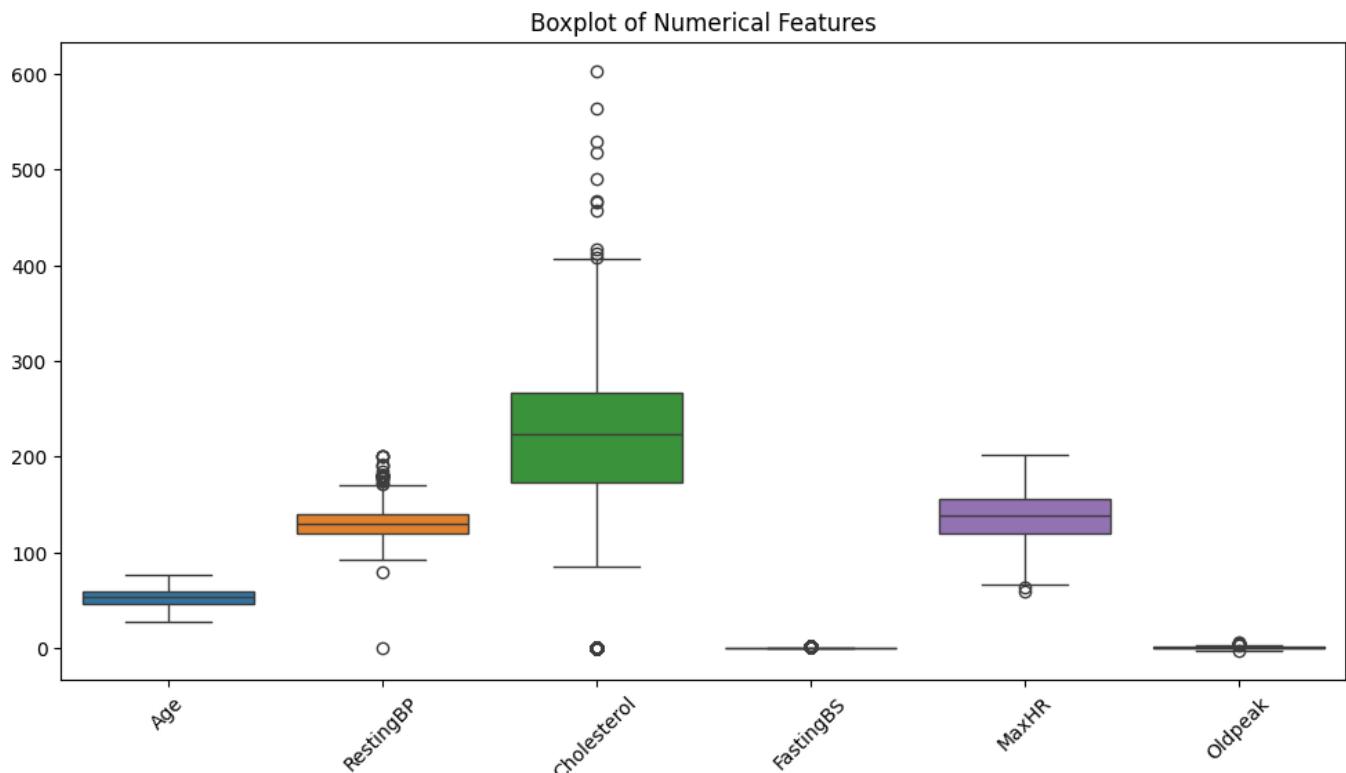


Findings from Histogram Analysis

- **Age:** The distribution appears fairly normal, with a higher concentration around middle-aged individuals.
- **RestingBP:** Shows a somewhat normal distribution with a few extreme values.
- **Cholesterol:** Right-skewed, indicating a few individuals with very high cholesterol levels.
- **FastingBS:** As a binary feature (0 or 1), it has two distinct peaks.
- **MaxHR:** Slightly left-skewed, meaning more individuals have a higher max heart rate.
- **Oldpeak:** Right-skewed with some extreme values, suggesting possible outliers.

Boxplot Analysis for Outliers

Boxplots have been used to detect outliers in numerical features before applying [StandardScaler](#). This is important to ensure that extreme values do not disproportionately influence the model.



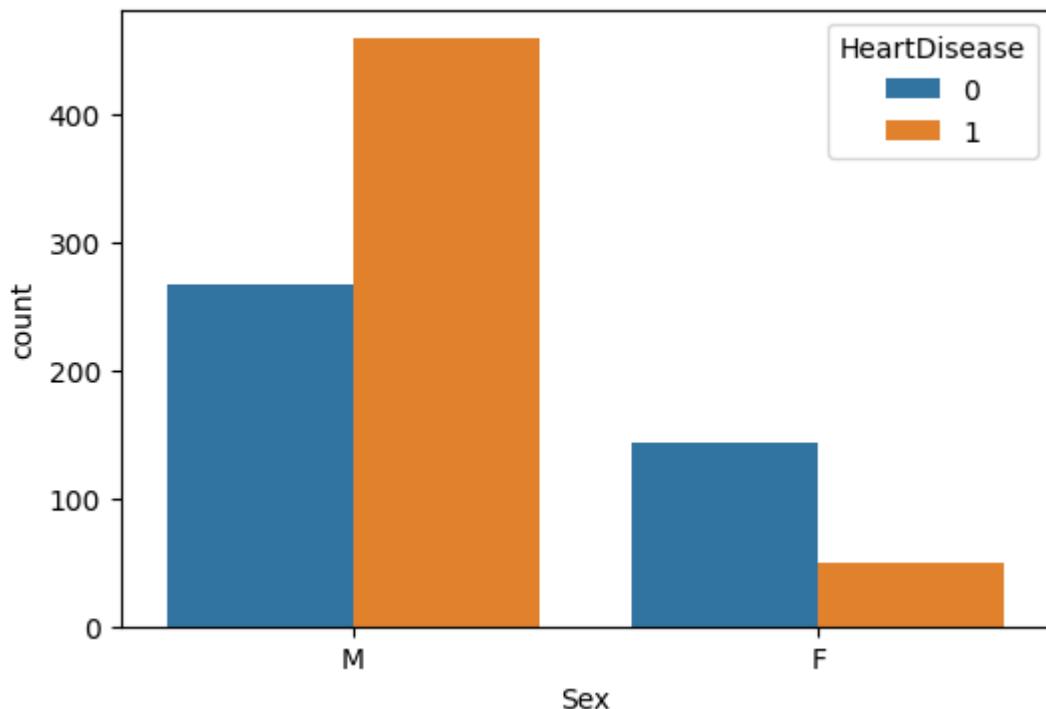
Findings from Boxplots

- **Cholesterol** and **Oldpeak** contain several outliers.
- **RestingBP** shows some extreme values, but most data points fall within an acceptable range.
- **MaxHR** has a few lower-end outliers, indicating individuals with unusually low heart rate responses.

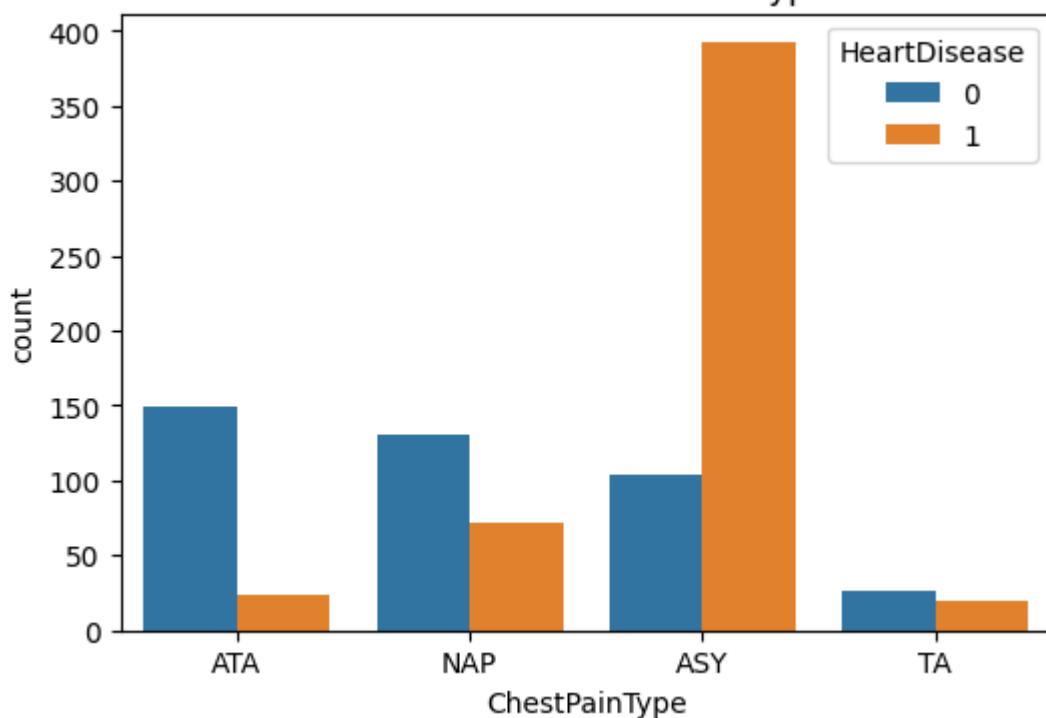
Count Plots for Categorical Features

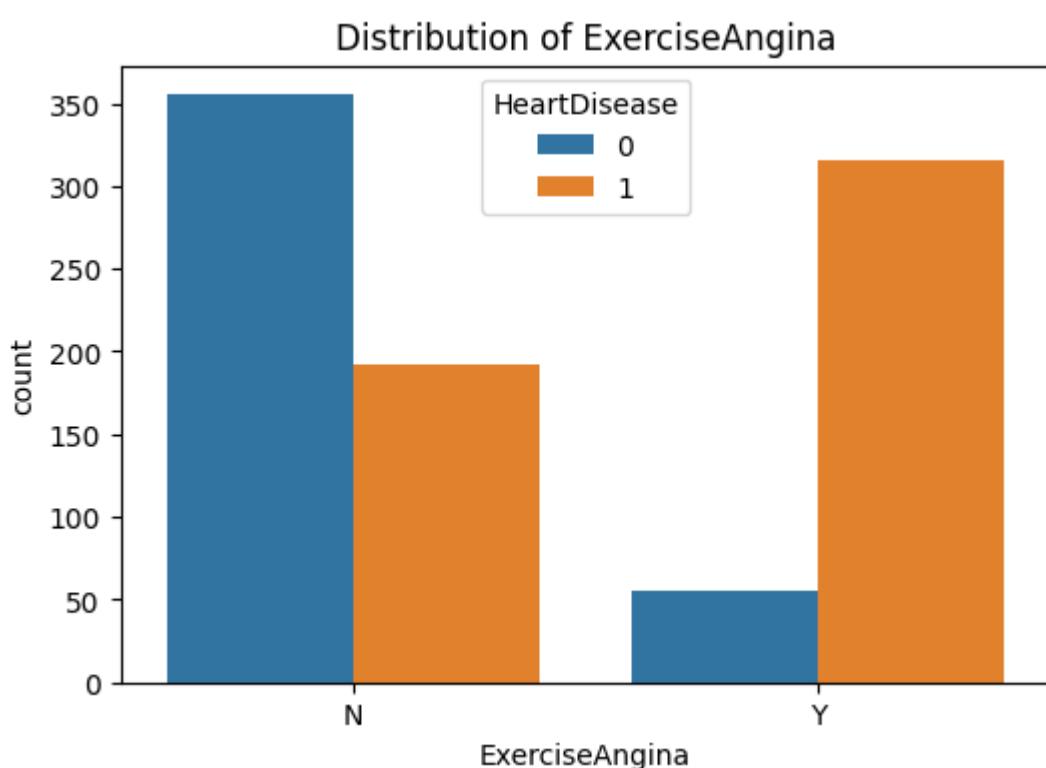
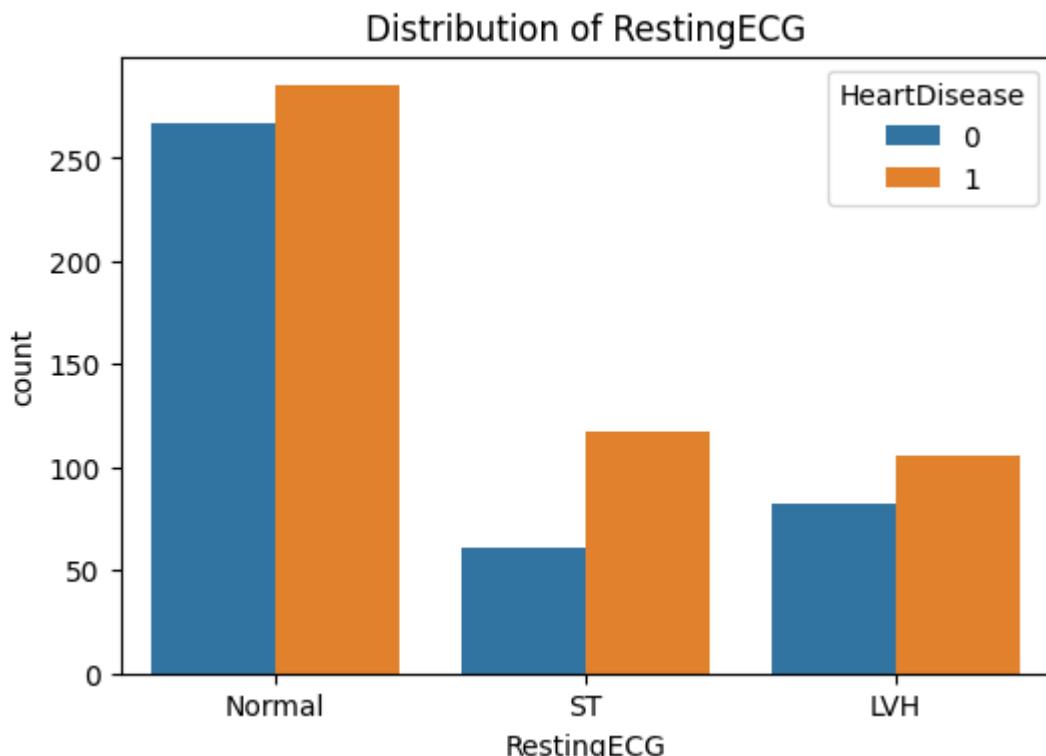
Count plots have been generated to visualize the distribution of categorical variables, helping to understand their frequency and balance.

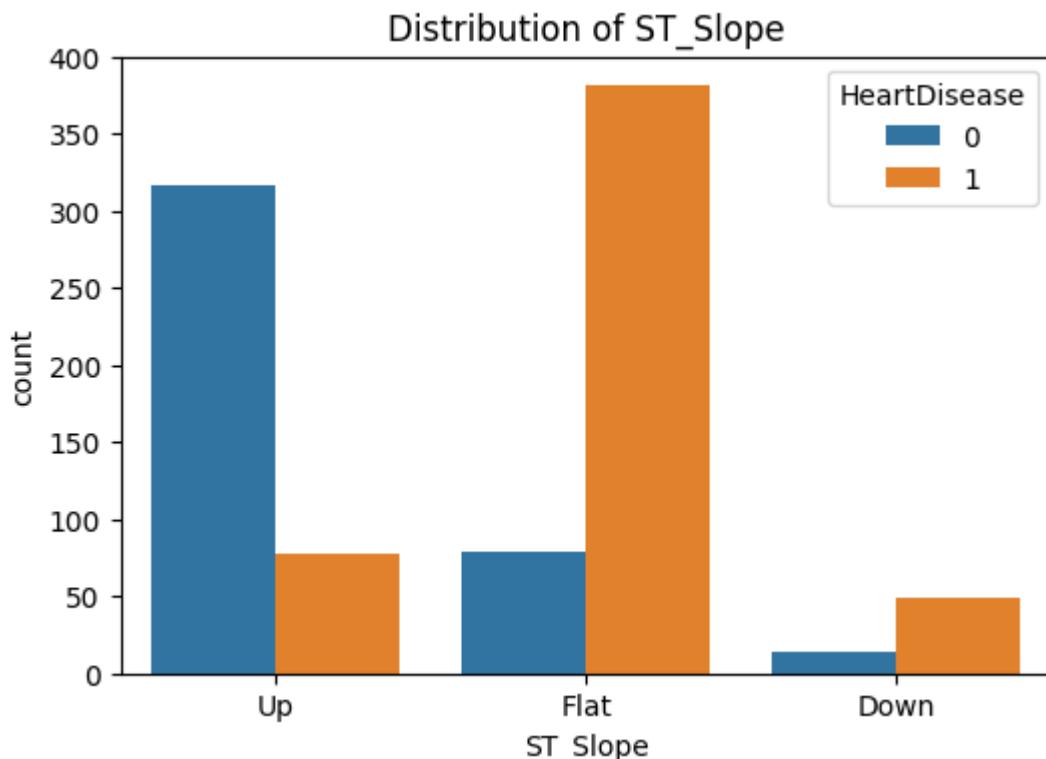
Distribution of Sex



Distribution of ChestPainType







Findings from Count Plots

- **Sex:** The dataset is **slightly imbalanced**, with more male (M) than female (F) participants.
- **ChestPainType:** The most frequent category is **ATA (Atypical Angina)**, while **TA (Typical Angina)** occurs less frequently.
- **RestingECG:** The **Normal** category dominates, while **ST** and **LVH** are less common.
- **ExerciseAngina:** Most individuals do not experience angina during exercise (N is more frequent than Y).
- **ST_Slope:** The **Up** slope is the most common, followed by **Flat** and **Down**.

Key Takeaways from Data Distribution Analysis

- **Histograms** revealed skewed distributions in **Cholesterol** and **Oldpeak**, suggesting potential transformations.
- **Boxplots** identified **outliers**, particularly in **Cholesterol**, **Oldpeak**, and **MaxHR**, which might require handling.
- **Count plots** showed **class imbalances** in some categorical features, which should be considered during modeling.

Data Preprocessing

To prepare the dataset for machine learning models, several preprocessing steps have been performed. These include encoding categorical variables, standardizing numerical features, detecting outliers post-scaling, and analyzing feature correlations.

Encoding Categorical Variables

Since the dataset contains both **binary** and **multi-category** categorical features, appropriate encoding techniques have been applied:

- **Binary Encoding:**
 - The binary categorical features **Sex** (M, F) and **ExerciseAngina** (Y, N) have been encoded using **Label Encoding**, where M → 1, F → 0, and Y → 1, N → 0.
- **One-Hot Encoding:**
 - Multi-category variables (**ChestPainType**, **RestingECG**, **ST_Slope**) have been encoded using **One-Hot Encoding**, creating separate binary columns for each category.

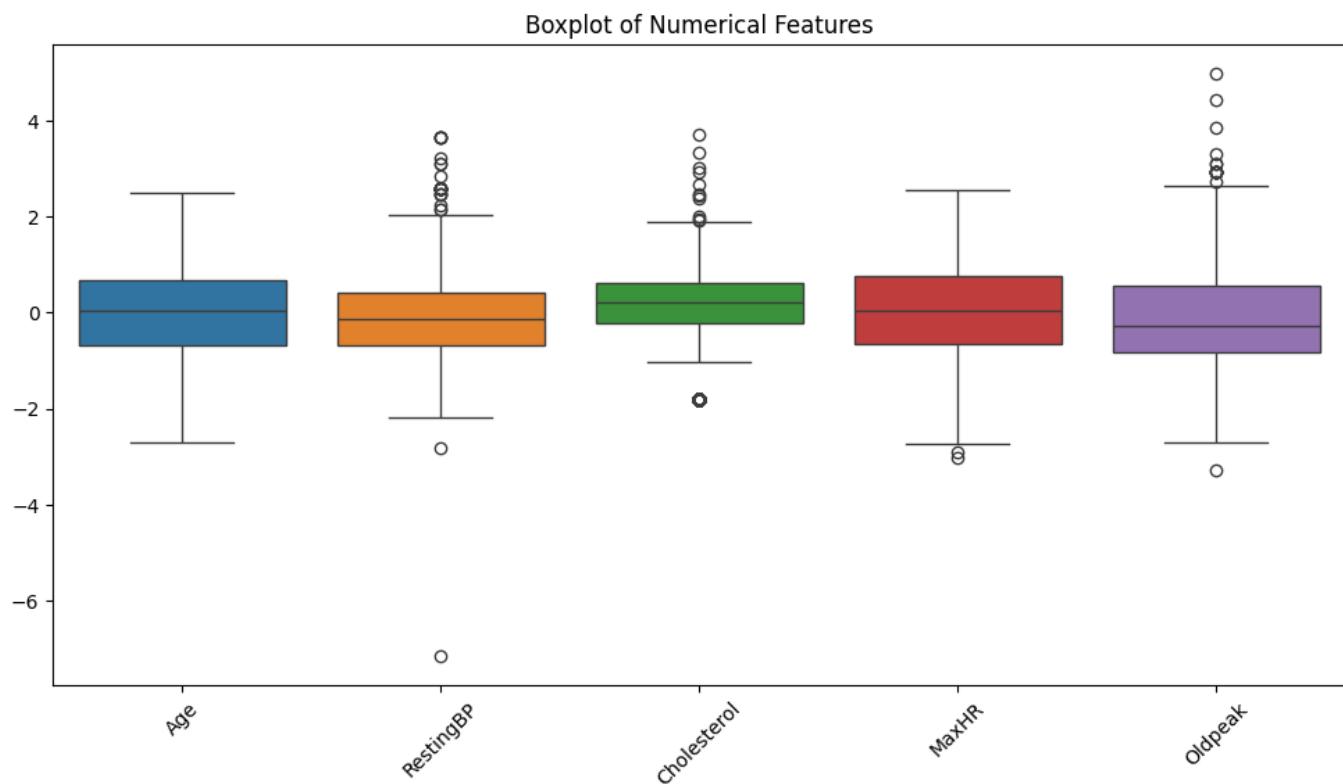
Standardizing Numerical Features

To ensure that all numerical features have the same scale and to improve model performance, **StandardScaler** has been applied to:

- Age
- RestingBP
- Cholesterol
- MaxHR
- Oldpeak

Boxplot Analysis (Post-Scaling)

A **boxplot** has been generated to visualize numerical features after standardization and to **check for remaining outliers**.



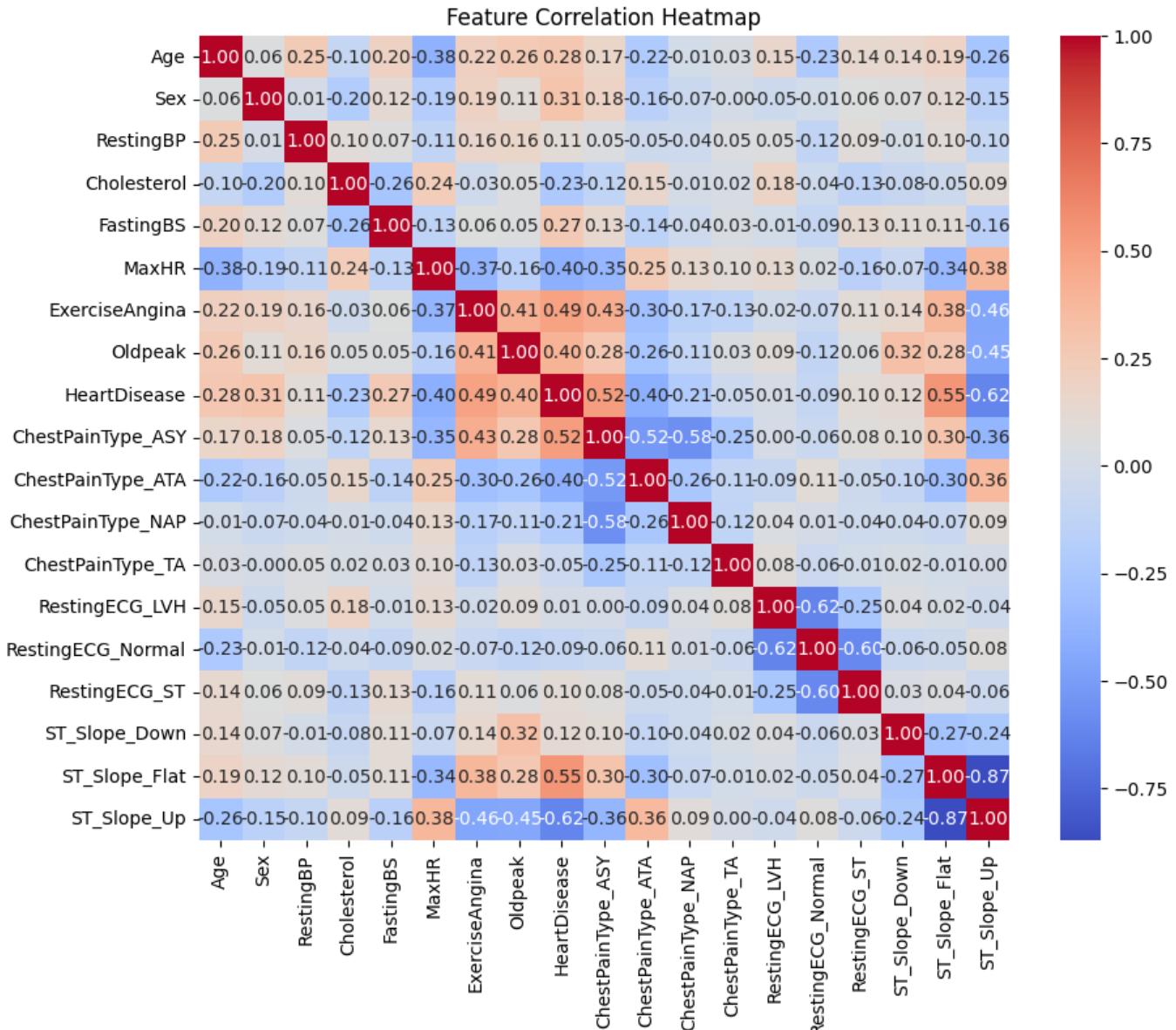
Findings from Boxplot Analysis

- Standardization **effectively reduced extreme value impact**, making distributions more uniform.
- Some features (**Cholesterol**, **Oldpeak**) still exhibit minor outliers, but they are **less extreme** after scaling.

- Outliers will be **handled in model training**, potentially using robust models that are less sensitive to them.

Correlation Analysis

A **heatmap of feature correlations** has been generated to explore relationships between features and the target variable (**HeartDisease**).



Key Insights from Correlation Analysis

- Strongest Positive Correlations with HeartDisease:**
 - Oldpeak** and **ST_Slope_Down** show a high positive correlation, meaning individuals with higher **Oldpeak values** and a **downward ST slope** are more likely to have heart disease.
- Strongest Negative Correlations with HeartDisease:**
 - MaxHR** and **ST_Slope_Up** have the strongest negative correlation, suggesting **higher heart rates and upward ST slopes** are linked to **lower risk** of heart disease.
- Moderate Correlations:**
 - ExerciseAngina** and **Age** show moderate correlations with **HeartDisease**, aligning with known medical risk factors.

- **Low Correlations:**

- Some variables like **RestingBP** and **Cholesterol** show relatively weak correlations with the target variable.

These insights will **guide feature selection and model development**, helping prioritize the most relevant features for prediction.

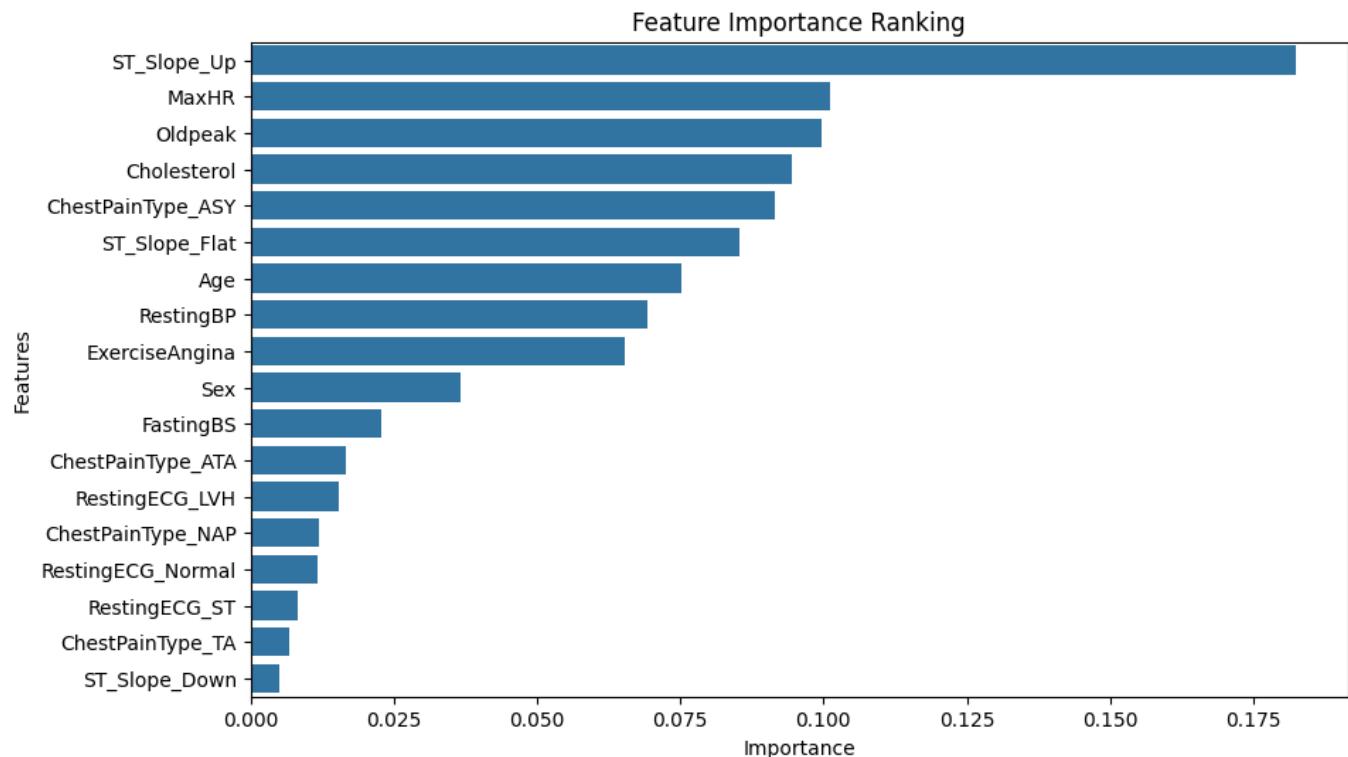
Feature Analysis

Feature Projection

To visualize high-dimensional data, **t-SNE** and **PCA** have been applied.

Feature Importance

Feature importance has been evaluated using **Random Forest** and **Decision Tree** classifiers.



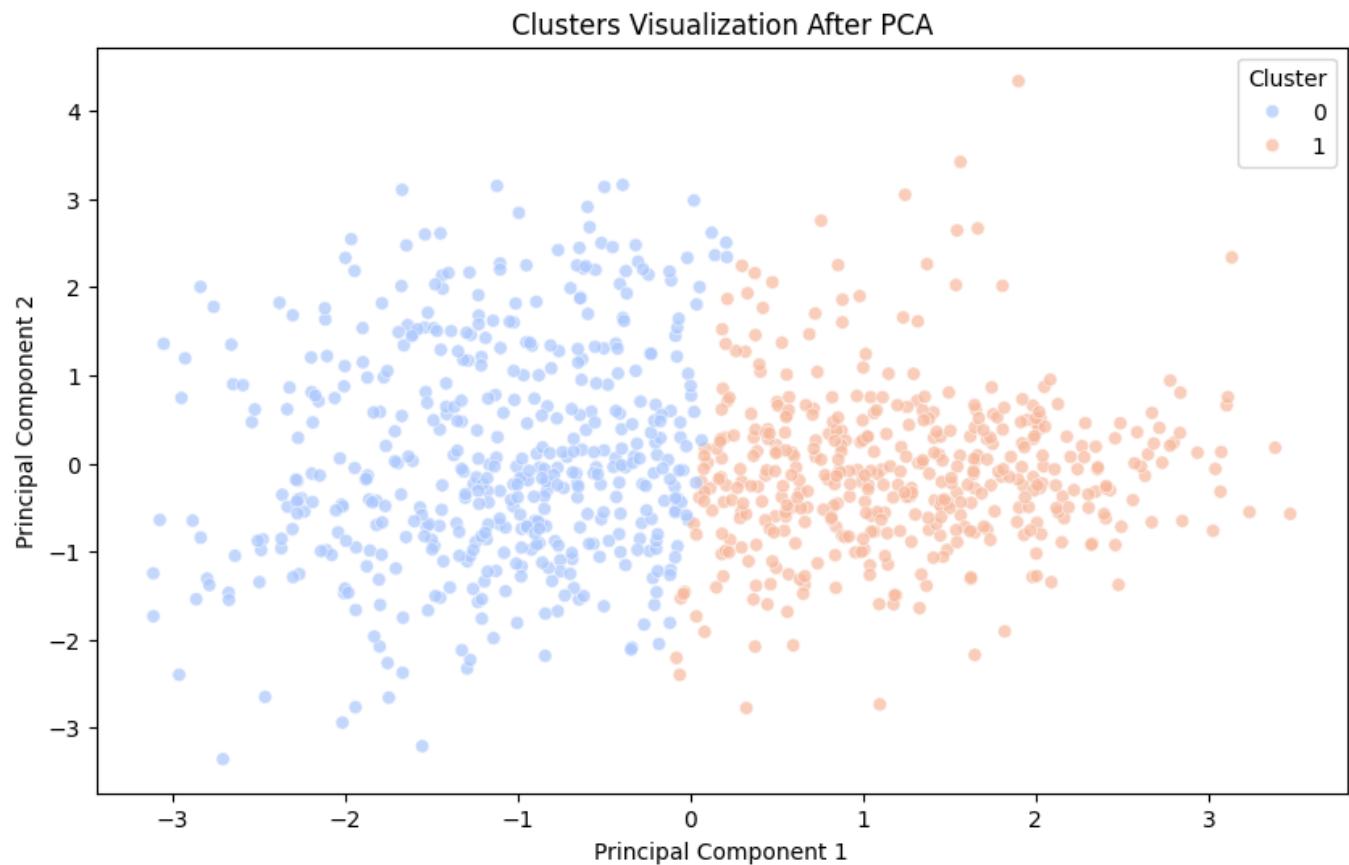
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Feature	Importance
17	ST_Slope_Up	0.387088

	Feature	Importance
3	Cholesterol	0.125370
5	MaxHR	0.089751
2	RestingBP	0.079652
8	ChestPainType_ASY	0.068116
0	Age	0.065223
7	Oldpeak	0.055972
1	Sex	0.041874
4	FastingBS	0.027799
6	ExerciseAngina	0.025193
13	RestingECG_Normal	0.010509
16	ST_Slope_Flat	0.008895
12	RestingECG_LVH	0.005877
10	ChestPainType_NAP	0.003896
15	ST_Slope_Down	0.002527
14	RestingECG_ST	0.002257
11	ChestPainType_TA	0.000000
9	ChestPainType_ATA	0.000000

PCA for Dimensionality Reduction



Model Training and Evaluation

What is the difference between `.fit()` and `.fit_transform()`?

The difference between `.fit()` and `.fit_transform()` mainly applies to **scikit-learn** preprocessing methods like `StandardScaler`, `MinMaxScaler`, `LabelEncoder`, etc.

1. `.fit()`

- **Learns the parameters** from the training data (e.g., mean and standard deviation for `StandardScaler`).
- **Does not transform** the data.
- Used when you **only want to learn** the transformation but will apply it later.

Example:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train) # Learns mean and std from X_train
```

2. `.fit_transform()`

- **Learns the parameters** (like `.fit()`) **and applies the transformation** to the data in one step.
- Used when you want to **train and transform in one line**.

Example:

```
X_train_scaled = scaler.fit_transform(X_train) # Learns and transforms X_train
```

Key Difference:

Function	Learns Parameters?	Transforms Data?	Use Case
.fit()	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No	Use before .transform() for consistency across datasets
.fit_transform()	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes	Use when training and transforming data in one step

When to Use Which?

- **Use .fit()** when you need to apply the same transformation to different datasets (e.g., fitting on X_train and transforming X_test separately).
- **Use .fit_transform()** for convenience when you are working with a single dataset and do not need to transform additional data later.

Example: Applying to Training and Test Data

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train) # Fit and transform training data
X_test_scaled = scaler.transform(X_test) # Only transform test data (without refitting)
```

Initial Model Performance

Several models were trained and evaluated on the dataset. Below are the initial accuracy results:

Model	Accuracy
Decision Tree	0.8098
Logistic Regression	0.8533
XGBoost	0.8859
SVM	0.8696

- **XGBoost achieved the highest accuracy (88.59%)** before hyperparameter tuning.
- **Logistic Regression and SVM** also performed well, while **Decision Tree had the lowest accuracy**.

Hyperparameter Tuning Results

Hyperparameter tuning was performed to optimize model performance. The best parameters and test accuracies are summarized below:

Model	Accuracy (Before)	Accuracy (After)	Improvement
Decision Tree	0.8098	0.8338	+2.40%
Logistic Regression	0.8533	0.8624	+0.91%
XGBoost	0.8859	0.8733	-1.26%
SVM	0.8696	0.8760	+0.64%

- **Decision Tree showed the most improvement** (+2.4%) after tuning.
- **SVM and Logistic Regression improved slightly**, while **XGBoost slightly decreased** due to reduced complexity.

Best Hyperparameters

Model	Best Parameters
Decision Tree	max_depth=5, max_features='auto', min_samples_leaf=2, min_samples_split=2
Logistic Regression	C=10, penalty='l2', solver='lbfgs'
XGBoost	learning_rate=0.2, max_depth=3, n_estimators=50, subsample=0.8
SVM	C=1, gamma=0.1, kernel='rbf'

Findings

- **XGBoost performed best initially, but hyperparameter tuning led to a slight drop in accuracy.**
- **Decision Tree benefited the most from tuning, improving its accuracy.**
- **SVM and Logistic Regression showed minor improvements.**
- **Tuning requires balancing model complexity and generalization to avoid overfitting.**

Model Interpretation and Evaluation

Performance Metrics Comparison

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.8098	0.8830	0.7757	0.8259
Logistic Regression	0.8533	0.9000	0.8411	0.8696
XGBoost	0.8804	0.9048	0.8879	0.8962
SVM	0.8696	0.8879	0.8879	0.8879

- **XGBoost achieved the highest F1-Score (0.8962)**, making it the best overall model.

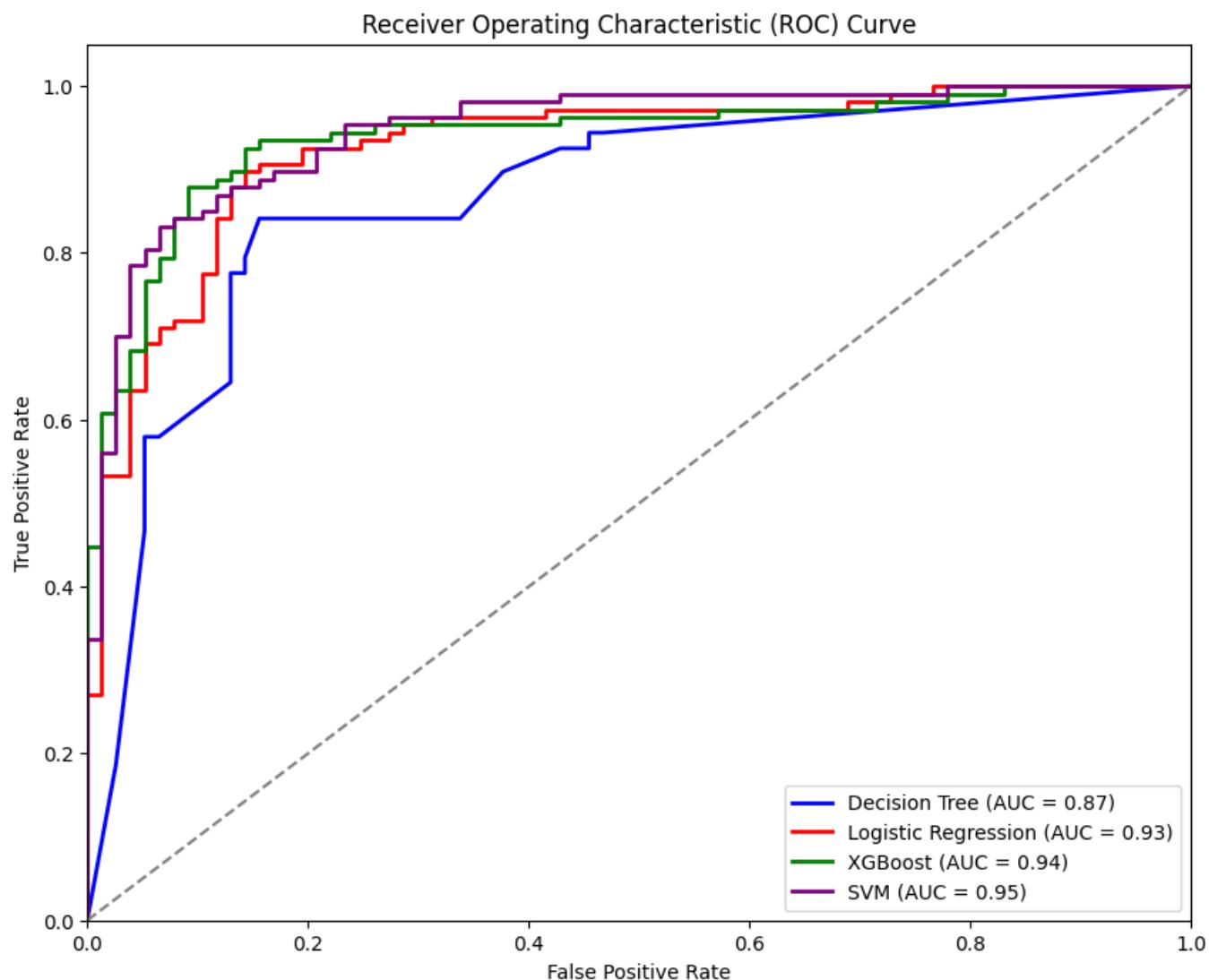
- **SVM had balanced Precision and Recall**, suggesting stable predictions.
- **Decision Tree had the lowest Recall (0.7757)**, indicating it missed more positive cases.

Training Time Analysis

Model	Accuracy	Precision	Recall	F1-Score	Training Time (s)
Decision Tree	0.8098	0.8830	0.7757	0.8259	0.0066
Logistic Regression	0.8533	0.9000	0.8411	0.8696	0.0154
XGBoost	0.8804	0.9048	0.8879	0.8962	0.0378
SVM	0.8696	0.8879	0.8879	0.8879	0.0725

- **Decision Tree was the fastest model**, training in just **0.0066s**.
- **XGBoost had the best performance but took longer to train (0.0378s)**.
- **SVM had the slowest training time (0.0725s)**, making it less efficient for large datasets.

ROC Curve and AUC Scores



Model	AUC Score
SVM	0.95

Model	AUC Score
Decision Tree	0.87
Logistic Regression	0.93
XGBoost	0.94
SVM	0.95

- **SVM had the highest AUC score (0.95)**, indicating strong separation between classes.
- **XGBoost and Logistic Regression also performed well, with AUC scores of 0.94 and 0.93.**
- **Decision Tree had the lowest AUC (0.87)**, suggesting weaker classification performance.

Findings

- **XGBoost offers the best balance between performance and efficiency.**
- **SVM performs slightly better in AUC but takes the longest to train.**
- **Decision Tree is the fastest but least accurate model.**
- **Logistic Regression provides a good trade-off between accuracy, AUC, and training time.**