University of Science and Technology of Hanoi

Fundamental of Data Science

Lecturer: Prof. Nguyen Duc Dung

FINAL PROJECT
HEART FAILURE PREDICTION

Student Name	Student ID
Nguyen Hoang Tung	22BA13388
Pham Cong Duyet	23BI14136

Hanoi, September 2025

Abstract

Heart disease is one of the most common and serious health problems worldwide. This project analyzes a dataset of 918 patients, which includes demographic and clinical information such as age, gender, blood pressure, cholesterol, chest pain type, and medical test results.

The dataset was cleaned by correcting invalid or missing values and explored through various visualizations, including histograms, boxplots, and heatmaps, to identify key patterns and relationships among features.

Several machine learning modelsLogistic Regression, Random Forest, SVM, and XGBoostwere then trained and evaluated to predict the presence of heart disease. Among them, the XGBoost model achieved the highest accuracy and stability.

Finally, a web-based application was developed using Flask to allow users to input their health indicators and receive real-time predictions. The project demonstrates how basic data analysis and machine learning techniques can be applied to support early detection and prevention of heart disease.

Contents

1 Introduction **4**
- 1.1 Motivation . . . 4
- 1.2 Business Analysis . . . 4
- 1.3 Goals . . . 5
- 1.4 Research Scope and Object of Study . . . 5
- 1.5 Contributions of the Study . . . 6

2 Data collection and management **8**
- 2.1 Data collection . . . 8
- 2.2 Data management . . . 10
  - 2.2.1 Data Cleaning . . . 10
  - 2.2.2 Data Processing . . . 12

3 Visualization **14**
- 3.1 Histogram . . . 14
- 3.2 Countplot . . . 16
- 3.3 Boxplot . . . 18
- 3.4 Distribution . . . 20
- 3.5 Heatmap . . . 22

4 Model Development **24**
- 4.1 Tools and Libraries . . . 24
- 4.2 Data Preparation . . . 25
- 4.3 Logistic Regression . . . 26
  - 4.3.1 How Logistic Regression works . . . 26
  - 4.3.2 Implementation of Logistic Regression . . . 26
  - 4.3.3 Results of Logistic Regression . . . 27
- 4.4 Support Vector Classification . . . 28
  - 4.4.1 How Support Vector Machine (SVM) works . . . 28
  - 4.4.2 Implementation of SVM . . . 28
  - 4.4.3 Results of SVC . . . 29
- 4.5 Random Forest . . . 30
  - 4.5.1 How Random Forest works . . . 30
  - 4.5.2 Implementation of Random Forest . . . 30
  - 4.5.3 Results of Random Forest . . . 31
- 4.6 Extreme Gradient Boosting (XGBoost) . . . 31
  - 4.6.1 How XGBoost works . . . 32
  - 4.6.2 Implementation of XGBoost . . . 32
  - 4.6.3 Results of XGBoost . . . 32

4.7 Summary of results . . . 33
4.7.1 SHapley Additive exPlanations . . . 34
4.7.2 Permutation Importance . . . 36

5 User Interface Implementation **37**
5.1 Model Persistence with Joblib . . . 37
5.2 Create app.py file . . . 37
5.3 Create index.html file . . . 38

6 Conclusion and References **39**
6.1 Conclusion . . . 39
6.2 References . . . 40

References **40**

List of Figures

3.1 Histogram of continuous features. . . . 14
3.2 Countplots of categorical variables. . . . 16
3.3 Boxplots of selected features by HeartDisease. . . . 18
3.4 Class-conditional KDE distributions of continuous features by *HeartDisease*. 20
3.5 Correlation heatmap of features. . . . 22

4.1 Top 10 most important features based on mean |SHAP| values. . . . 35
4.2 Top 10 features based on permutation importance (mean decrease in ROC-AUC). . . . 36

Chapter 1

Introduction

1.1 Motivation

Heart disease is still one of the leading causes of death in the world today. According to the World Health Organization (WHO), cardiovascular diseases cause about 17.9 million deaths every year nearly one-third of all global deaths and this number may continue to rise by 2030 if prevention and early diagnosis are not improved. Most of these cases occur in developing countries, where healthcare systems still face many challenges in early detection and long-term treatment.

In addition to health risks, heart disease also creates a heavy economic burden. The World Economic Forum estimates that cardiovascular and other chronic diseases could cost the global economy over one trillion USD per year by 2030 due to medical expenses and loss of productivity.

In Vietnam, the situation is also worrying. Around one in four adults has high blood pressure, and heart disease ranks among the top causes of death and hospitalization, especially in major cities like Hanoi and Ho Chi Minh City. Many patients only discover the problem at a late stage, when treatment becomes more difficult and expensive.

Therefore, finding new and effective ways to detect heart disease early is very important. In recent years, machine learning has become a promising solution because it can analyze large amounts of medical data and discover patterns that traditional methods might overlook. By examining variables such as age, blood pressure, cholesterol, ECG results, and exercise indicators, machine learning models can provide faster and more objective risk predictions.

This project aims to develop a practical and reliable machine learning model for predicting heart disease. The model is expected to support early diagnosis, reduce treatment costs, and contribute to improving the quality of healthcare services in Vietnam and beyond.

1.2 Business Analysis

The project uses the Heart Failure Prediction Dataset (Fedesoriano, 2021), which includes demographic and clinical information such as age, sex, cholesterol level, blood pressure, chest pain type, and ECG results. This dataset provides a good foundation for developing predictive models that can estimate the risk of heart disease at an early stage, even before serious symptoms appear.

From a practical perspective, predictive models like this can bring several benefits to both patients and healthcare organizations.

Early detection: When the model can identify high-risk patients early, doctors can provide timely treatment or recommend lifestyle changes before the condition becomes severe.

Cost reduction: Early intervention helps reduce expensive emergency care and long-term treatment, which lowers the overall financial burden for both patients and hospitals.

Improved efficiency: Medical staff can focus their attention and resources on patients who need care the most, making the diagnosis and treatment process more efficient.

Better patient experience: When patients receive early warnings and clear explanations about their risks, they are more likely to follow medical advice and feel more confident in their treatment.

In countries like Vietnam, where medical resources are still limited, using data-driven approaches can help hospitals allocate equipment and staff more effectively. The insights generated from such models can also support evidence-based decision-making for doctors and healthcare managers.

Therefore, this project is not only a technical exercise in data analysis but also a practical example of how machine learning can be applied in the healthcare sector to save costs, improve service quality, and support better health outcomes for the community.

1.3 Goals

The main goal of this project is to build a machine learning model that can predict whether a patient is likely to have heart disease based on clinical data. More specifically, we aim to:

1. Collect and preprocess patient data such as age, sex, blood pressure, cholesterol, and other health indicators.
2. Explore and analyze the dataset to find patterns and important risk factors.
3. Train and evaluate different machine learning models to classify the presence of heart disease.
4. Compare the performance of models and select the one with the best accuracy and reliability.
5. Provide insights and a practical tool that can support doctors in early diagnosis and decision-making.

1.4 Research Scope and Object of Study

Research Object:

- This study focuses on predicting the risk of heart disease in patients based on their clinical and demographic information.

- The dataset includes both demographic features (age, sex) and clinical indicators such as chest pain type, resting blood pressure, cholesterol level, fasting blood sugar, resting electrocardiogram (ECG), maximum heart rate, and ST-related variables (Oldpeak, ST Slope).
- The goal is to develop a model that classifies patients into two categories: those at risk of heart disease and those not at risk.
- Through this, the research aims to support early detection and assist doctors in identifying high-risk individuals more efficiently.

Research Scope:

- The dataset consists of 918 patient records with 12 attributes, collected from the Heart Failure Prediction dataset on Kaggle.
- The scope of this study is limited to structured tabular data, without involving other types such as medical imaging or time-series data.
- The analysis focuses on data preprocessing, exploratory data analysis, machine learning model training, performance evaluation, and the development of a simple web-based interface for prediction.
- Broader applications like integration with hospital systems, real-time clinical deployment, or large-scale medical databases are considered beyond the current research scope due to time and resource limitations.

1.5 Contributions of the Study

Theoretical Contributions:

- Summarize basic theories and key risk factors related to cardiovascular diseases based on structured medical data.
- Compare the performance of different machine learning algorithms (Logistic Regression, Random Forest, SVM, XGBoost) in predicting heart disease.
- Emphasize the importance of preprocessing techniques such as handling missing data, feature scaling, and encoding categorical variables to improve prediction accuracy.
- Propose a combined approach that balances interpretability and performance by using both linear models and ensemble models in heart disease prediction.

Practical Contributions:

- Provide a simple predictive tool that can help doctors and medical staff identify high-risk patients more quickly and prioritize treatment.
- Support hospitals and clinics in making data-driven decisions to improve diagnostic efficiency and reduce costs.

- Increase public awareness of how data science and artificial intelligence can be applied in healthcare and early disease prevention.
- Offer useful references and a practical framework for future research on applying machine learning to predict and manage other chronic diseases.

Chapter 2

Data collection and management

2.1 Data collection

The dataset used in this project was collected from Kaggle: Heart Failure Prediction by fedesoriano. It contains clinical records of patients with and without heart disease, making it highly suitable for building classification models.

The dataset provides a total of 918 samples and 12 features, including:

- **Age:** age of the patient [years]
- **Sex:** sex of the patient [M: Male, F: Female]
- **ChestPainType:** chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- **RestingBP:** resting blood pressure [mm Hg]
- **Cholesterol:** serum cholesterol [mm/dl]
- **FastingBS:** fasting blood sugar [1: if FastingBS $>$ 120 mg/dl, 0: otherwise]
- **RestingECG:** resting electrocardiogram results [Normal: Normal, ST: ST-T wave abnormality (T wave inversions and/or ST elevation or depression $>$ 0.05 mV), LVH: probable or definite left ventricular hypertrophy by Estes' criteria]
- **MaxHR:** maximum heart rate achieved [numeric value between 60 and 202]
- **ExerciseAngina:** exercise-induced angina [Y: Yes, N: No]
- **Oldpeak:** ST depression induced by exercise relative to rest [numeric value]
- **ST_Slope:** the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- **HeartDisease:** output class [1: heart disease, 0: normal]

The target variable is HeartDisease, which indicates 1 for heart disease and 0 for Normal.

Table 2.1: Sample records from the Heart Disease Dataset (abbreviated column names)

Age	**Sex**	**CP**	**RBP**	**Chol**	**FBS**	**ECG**	**HR**	**ExAng**	**OP**	**Slope**	**HD**
40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Overall, the dataset is well-organized and easy to understand. It includes 918 patient records, which is a moderate amount of data but still sufficient to identify meaningful patterns and relationships. The variables are clearly defined, covering both demographic information (such as age and sex) and clinical measurements (including blood pressure, cholesterol, heart rate, and ECG results). At first check, there are no missing values (NaN), making it convenient to start analysis without complex preprocessing steps.

In addition, the dataset contains a balanced combination of categorical and numerical variables. Categorical features like chest pain type or exercise-induced angina describe patient conditions, while numerical features such as age, cholesterol, and maximum heart rate provide quantitative indicators. The target variable (*HeartDisease*) is also relatively balanced, with roughly half of the patients diagnosed with heart disease and half not. This balance helps ensure that the analysis and model training remain fair and unbiased. With these characteristics, the dataset is suitable for exploratory data analysis and for developing reliable predictive models.

2.2 Data management

2.2.1 Data Cleaning

When examining the dataset more carefully, it becomes clear that some columns contain values that are not realistic in a medical context. These numbers are not valid measurements but rather indicate missing or incorrectly recorded data.

- **RestingBP (Resting Blood Pressure):** There is one record with a value of 0. Since a blood pressure of 0 cannot occur in a living person, this value is considered invalid. It is therefore treated as missing data and replaced with a more reasonable estimate, specifically the median blood pressure from the dataset.
- **Cholesterol:** This represents the most significant issue in the dataset. A total of 172 records have cholesterol values equal to 0. Such values are not medically possible and are more likely the result of unrecorded or incorrectly entered data. Because cholesterol is an important factor in the study of heart disease, simply removing these rows would lead to the loss of too much information. Therefore, the missing values were imputed to preserve the dataset and maintain its reliability
- **Other features (Age, Sex, ChestPainType, RestingECG, MaxHR, ExerciseAngina, ST_Slope):** No zero or missing values were found here. These columns looked consistent and did not require cleaning.

In short, the main data quality problems came from **RestingBP** and **Cholesterol**. Values of 0 were considered as missing and replaced with reasonable estimates, which made the dataset cleaner and more reliable. This step is important to ensure that the analysis does not rely on "impossible" numbers, allowing the results to be more accurate and meaningful.

Table 2.2: Number of zero of NaN values in each column

Column	**Zero Values**
Age	0
Sex	0
ChestPainType	0
RestingBP	1
Cholesterol	172
RestingECG	0
MaxHR	0
ExerciseAngina	0
ST_Slope	0

Missing value in RestingBP

During the data cleaning stage, it was observed that the **RestingBP** (resting blood pressure) column contained 1 record with a value of 0. Such a value is not medically possible and was therefore considered invalid. This type of entry likely represents a missing measurement or a data entry error.

To address this issue, the invalid value was replaced with the **median RestingBP** from the dataset. The median was chosen over the mean because it is less sensitive to outliers and better represents the central tendency of blood pressure values.

What is the Median?

The **median** is a statistical measure that represents the middle value of a dataset when all observations are arranged in ascending or descending order. Unlike the mean, which can be strongly influenced by extreme values (outliers), the median is more robust and provides a better representation of the central tendency in skewed distributions.

Calculation of the Median:

1. Arrange all data points in numerical order.
2. If the number of observations (n) is odd, the median is the value at position $\frac{n+1}{2}$.
3. If n is even, the median is the average of the two middle values at positions $\frac{n}{2}$ and $\frac{n}{2} + 1$.

Application in Data Cleaning: In the context of missing or invalid medical data, replacing values with the median helps preserve the size of the dataset while reducing the risk of distortion caused by outliers. For the **RestingBP** feature, where only one invalid value (0) was found, the median of the valid blood pressure values was used as a substitute. This ensures that the imputed value remains within a realistic clinical range and maintains the reliability of the dataset.

Missing value in Cholesterol

One of the biggest problems in the dataset was found in the **Cholesterol** column. There were 172 records (about 19% of all data) where cholesterol was recorded as 0. In reality, a cholesterol level of zero cannot happen, so these entries clearly represent missing or unrecorded values.

If all these rows were deleted, almost one-fifth of the dataset would be lost, which would weaken the model and reduce its ability to learn. Instead, the 0 values were treated as missing data and filled in using imputation methods.

Two main approaches were considered:

- **Mean/Median filling:** Simple and fast, but it makes many patients have the same cholesterol value, which reduces the natural variation in the data.
- **KNN Imputer:** A smarter method that looks at patients with similar health profiles (neighbors) and uses their cholesterol levels to estimate the missing ones. This way, the filled values are more realistic and keep the relationships between features.

For this project, the **KNN Imputer** method was used, as it produces more reliable values than just plugging in the mean or median. With this approach, the cholesterol column becomes more accurate and continues to play its role as an important factor in predicting heart disease.

What is the KNN Imputer

The K-Nearest Neighbors (KNN) Imputer is a method that estimates missing values by looking at the values of similar records, also called neighbors. Instead of assigning a global mean or median, this method uses information from patients with similar clinical features to generate a more realistic replacement.

How it works:

1. For a record with a missing value, calculate the distance between this record and all other records using available features (e.g., Age, RestingBP, MaxHR).
2. The most common distance metric is Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

   where x_i and y_i are the feature values of two patients.
3. Identify the k nearest neighbors (the k most similar patients).
4. Estimate the missing value by taking the average (or weighted average) of the corresponding feature from these neighbors. If weighted, closer neighbors have more influence on the final value.

2.2.2 Data Processing

The dataset contains several categorical features that cannot be directly processed by most machine learning algorithms, including **Sex**, **ChestPainType**, **RestingECG**, **ExerciseAngina**, and **ST_Slope** . Since these variables are non-numeric, they must be transformed into numerical form before being used in modeling.

For this project, two different encoding strategies were applied depending on the type of categorical feature:

- **Binary Encoding (0/1)** for features with only two categories:
  - Sex: "M" $\rightarrow$ 1, "F" $\rightarrow$ 0
  - ExerciseAngina: "Y" $\rightarrow$ 1, "N" $\rightarrow$ 0
- **One-Hot Encoding** for features with more than two categories:
  - ChestPainType: four binary columns created for "ATA", "NAP", "ASY", and "TA"
  - RestingECG: three binary columns created for "Normal", "ST", and "LVH"
  - ST_Slope: three binary columns created for "Up","Flat", and "Down"

Advantages of this approach:

- Binary features (Sex, ExerciseAngina) are represented compactly without creating extra columns.
- One-Hot Encoding avoids introducing artificial ordinal relationships among nominal categories, ensuring that models do not misinterpret them as ranked values.
- This representation makes the dataset compatible with both linear models (e.g., Logistic Regression) and non-linear models (e.g., Decision Trees, Random Forests).

Limitations:

- One-Hot Encoding increases dataset dimensionality by creating additional columns. However, in this dataset the number of categories is small, so the dimensionality increase remains manageable.
- Binary encoding cannot capture any potential imbalance if one category is significantly underrepresented (e.g., very few females compared to males).

Reason for choosing this encoding strategy:

- It ensures that categorical variables are represented numerically without introducing misleading ordinal relationships.
- The dataset has relatively few categorical features and limited categories, so the increase in dimensionality from One-Hot Encoding is minimal.
- This hybrid approach combines the efficiency of binary encoding with the interpretability and correctness of One-Hot Encoding for nominal features.

By applying this combination of Binary Encoding and One-Hot Encoding, categorical features were successfully transformed into numeric form while preserving their true nature, ensuring compatibility with machine learning models and preparing the dataset for subsequent preprocessing and modeling.

Chapter 3

Visualization

3.1 Histogram

The figure below presents **histograms** that describe the distribution of key variables in the heart disease dataset, including: Age, Resting Blood Pressure (RestingBP), Cholesterol, Maximum Heart Rate achieved (MaxHR), and ST depression induced by exercise (Oldpeak).

These plots provide a clear overview of how the data is distributed:

- Some variables are fairly centered around typical values, while others show greater spread and variability.
- Certain variables exhibit skewness and contain clear outliers.

Examining these distributions helps us better understand patient characteristics and provides an important foundation for data preprocessing and predictive modeling.

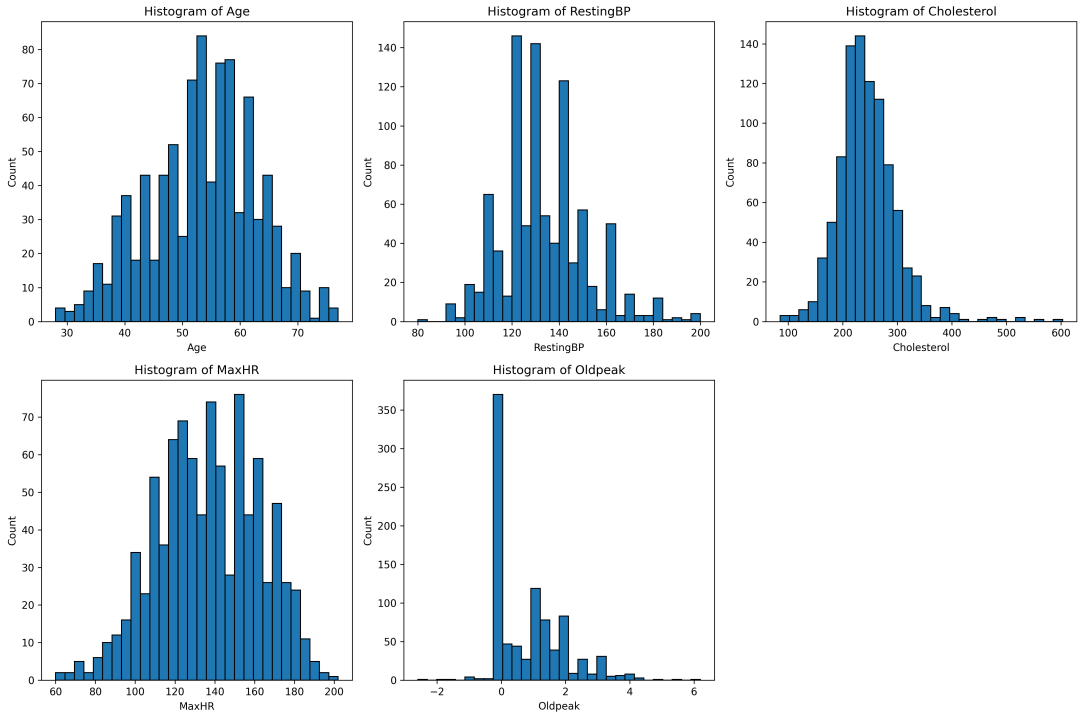

Figure 3.1: Histogram of continuous features.

- **Age**
  - *Distribution:* Concentrated mainly between 45 and 60 years, with fewer patients above 65.
  - *Insight:* The dataset is dominated by middle-aged and older adults, which aligns with the higher prevalence of heart disease in these age groups.
- **RestingBP**
  - *Distribution:* Most values fall between 120 and 140 mmHg.
  - *Insight:* Slightly elevated resting blood pressure is frequent, reflecting hypertension as a common comorbidity of heart disease.
- **Cholesterol**
  - *Distribution:* Right-skewed, with the majority of patients below 300 mg/dl and a few extreme cases above 400 mg/dl.
  - *Insight:* While most individuals have moderate cholesterol, a subset shows abnormally high values, likely due to genetic or lifestyle factors.
- **MaxHR**
  - *Distribution:* Approximately normal, centered between 120–160 bpm.
  - *Insight:* This range is typical for adults, but patients with heart disease tend to have lower maximum heart rates, consistent with reduced exercise tolerance.
- **Oldpeak**
  - *Distribution:* Strongly concentrated at 0, with a long tail extending above 4.
  - *Insight:* Most healthy patients show minimal ST depression, while ischemic patients present higher Oldpeak values, capturing their clinical severity.

3.2 Countplot

The figure below shows **count plots** for key categorical variables in the heart disease dataset, including: Sex, Fasting Blood Sugar (FastingBS), Exercise-induced Angina (ExerciseAngina), Chest Pain Type (ChestPainType), Resting Electrocardiogram (RestingECG), ST Slope (ST_Slope), and the target variable HeartDisease.

These plots highlight:

- The distribution of patients across different categories.
- Some variables are imbalanced (e.g., males are the majority in Sex).
- The target label (HeartDisease) shows that positive cases are slightly more common than negatives.

Such categorical insights are important for understanding potential relationships between features and the disease, providing useful guidance for feature engineering and model interpretation.

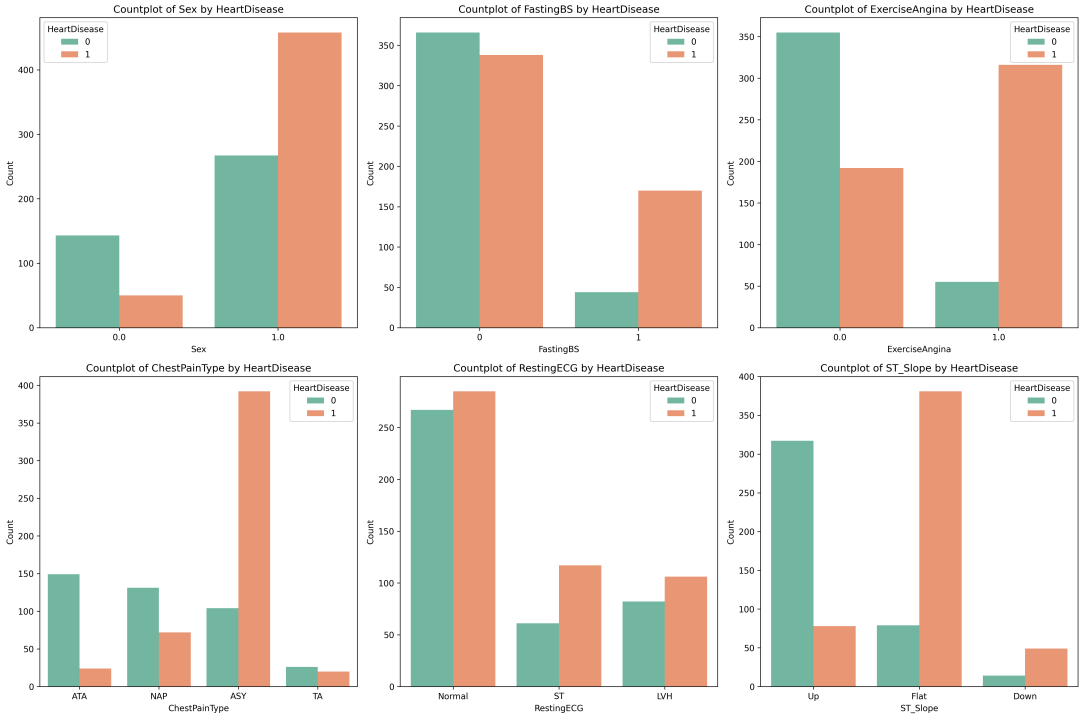

Figure 3.2: Countplots of categorical variables.

- **Sex**
  Distribution: Male patients dominate the dataset (∼79%), while females account for only ∼21%.
  Insight: Men are more likely to develop heart disease, consistent with epidemiological findings that male sex is an independent risk factor, especially in middle age.

- **FastingBS**
  Distribution: The majority of patients (∼77%) have normal fasting blood sugar

(< 120 mg/dl), while $\sim$23% exceed this threshold.
Insight: Elevated fasting blood sugar indicates potential diabetes, which increases cardiovascular risk, though most patients in this dataset do not present abnormal glucose.

- **ExerciseAngina**
  Distribution: Around 60% of patients report no angina during exercise, while 40% experience it.
  Insight: Exercise-induced angina is a clear clinical marker of ischemia, and its presence strongly associates with heart disease.

- **ChestPainType**
  Distribution: Asymptomatic (ASY) chest pain is the most common category, followed by ATA, NAP, and TA.
  Insight: The dominance of asymptomatic chest pain reflects that many patients with heart disease may not show typical angina symptoms, emphasizing the importance of diagnostic testing.

- **RestingECG**
  Distribution: Most patients show a Normal ECG, with LVH and ST-T abnormalities less frequent.
  Insight: Normal resting ECGs are common even among heart patients, highlighting that ECG alone may not be sufficient for early detection.

- **ST_Slope**
  Distribution: Flat and Up slopes dominate, while Down slope is rare.
  Insight: A Flat slope is highly associated with disease, while an Up slope is often protective, aligning with clinical cardiology knowledge.

- **HeartDisease (Target)**
  Distribution: 508 patients (55%) have heart disease, while 410 (45%) do not.
  Insight: The dataset is relatively balanced but slightly enriched with positive cases, ensuring the model can learn patterns of diseased patients effectively.

3.3 Boxplot

The figure below presents **boxplots** comparing patients with heart disease (HeartDisease = 1) and those without (HeartDisease = 0) across continuous variables: Age, RestingBP, Cholesterol, MaxHR, and Oldpeak.

The plots indicate that:

- Patients with heart disease tend to be older and exhibit significantly higher Oldpeak values.
- In contrast, their MaxHR is generally lower compared to those without the disease.
- RestingBP and Cholesterol show much overlap between groups and contain many outliers, making them weaker discriminators.

These patterns suggest that some continuous variables, particularly MaxHR and Oldpeak, play a more significant role in distinguishing between heart disease and non-heart disease patients.

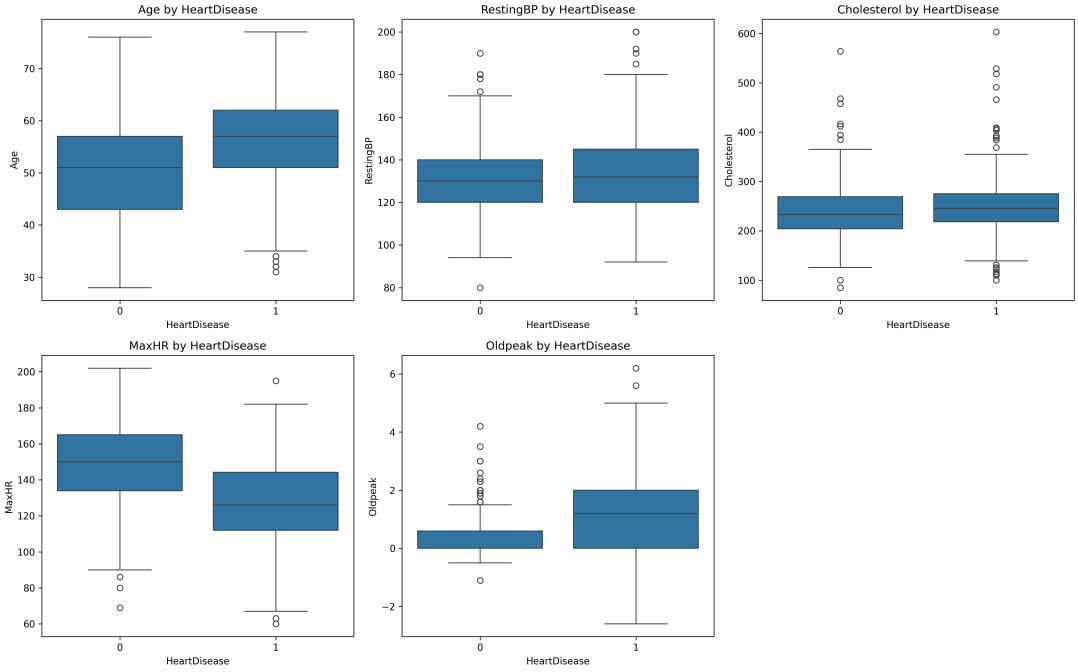

Figure 3.3: Boxplots of selected features by HeartDisease.

- **Age**
  Distribution: Patients with heart disease show a higher median age and narrower variability.
  Insight: Cardiovascular risk increases with age due to vascular stiffening and cumulative exposure to risk factors.
- **RestingBP**
  Distribution: Medians are similar between groups, with wide variability and numerous outliers.
  Insight: Although hypertension is a known risk factor, in this dataset RestingBP does not strongly differentiate the groups.

- **Cholesterol**
  Distribution: Both groups show wide spread and overlapping medians, with several extreme high values.
  Insight: Elevated cholesterol is clinically important, but the weak separation here may be influenced by diet, genetics, or medical treatment prior to data collection.

- **MaxHR**
  Distribution: Non-diseased patients reach higher MaxHR, while diseased patients cluster at lower values.
  Insight: Diseased patients demonstrate reduced exercise tolerance because impaired coronary circulation prevents achieving high heart rates.

- **Oldpeak**
  Distribution: Diseased patients show higher median Oldpeak and a longer upper tail, with many extreme outliers.
  Insight: Increased ST depression during exercise is a classic clinical marker of myocardial ischemia, aligning with elevated Oldpeak in heart disease cases.

3.4 Distribution

The figure below shows **density plots** comparing the distributions of continuous variables (Age, RestingBP, Cholesterol, MaxHR, and Oldpeak) across patients with heart disease (HeartDisease = 1) and those without (HeartDisease = 0).

Key observations include:

- Patients with heart disease tend to be older and present higher Oldpeak values.
- Non-diseased patients typically achieve higher MaxHR.
- RestingBP and Cholesterol distributions show heavy overlap between groups, providing limited discriminatory power.

These findings suggest that variables like **Age**, **MaxHR**, and **Oldpeak** offer stronger separation between diseased and non-diseased patients, making them potentially more informative for predictive modeling.

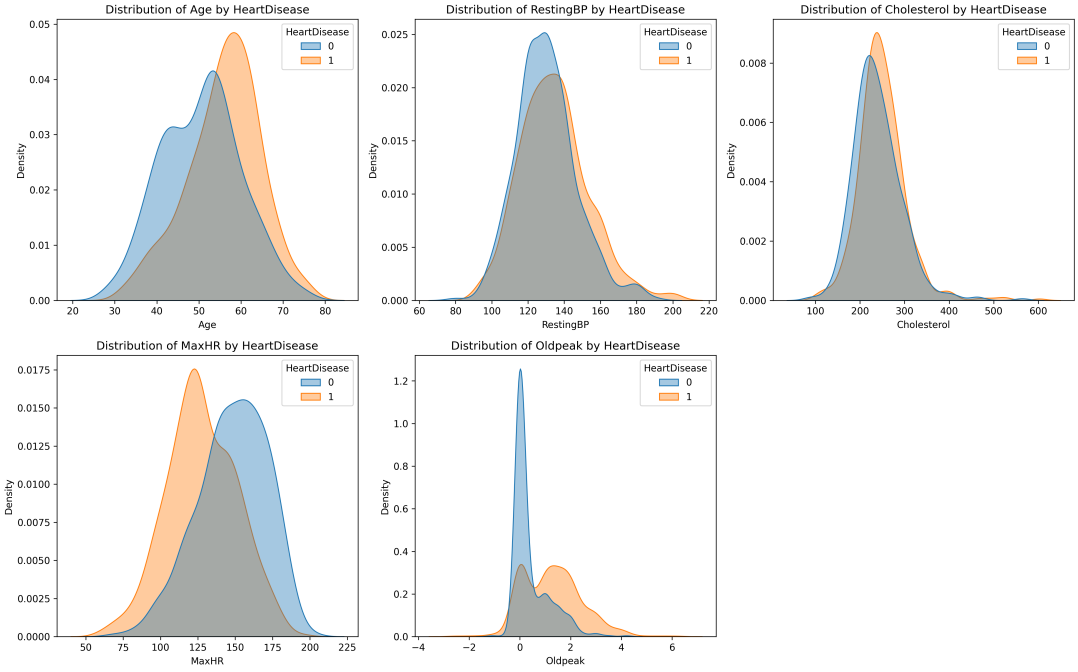

Figure 3.4: Class-conditional KDE distributions of continuous features by *HeartDisease*.

- **Age**
  Distribution: The density curve for diseased patients shifts rightward, peaking in the 55–65 age range.
  Insight: Accumulated exposure to risk factors and vascular aging contribute to the higher prevalence of disease in older individuals.

- **RestingBP**
  Distribution: Nearly identical curves for both groups, centered around 120–140 mmHg.
  Insight: Although hypertension is clinically important, in this dataset RestingBP does not effectively separate groups.

- **Cholesterol**
  Distribution: Heavily overlapping across diseased and non-diseased patients, with no clear shift.
  Insight: Cholesterol may be confounded by prior treatment or lifestyle factors, reducing its discriminative value here.

- **MaxHR**
  Distribution: Non-diseased patients show strong peaks at 150–170 bpm, while diseased patients cluster more below 140 bpm.
  Insight: Lower achievable heart rate indicates impaired cardiac reserve, a strong clinical marker of coronary disease.

- **Oldpeak**
  Distribution: Diseased patients exhibit a broader curve shifted toward higher values, while non-diseased patients are concentrated around zero.
  Insight: ST depression is a hallmark of myocardial ischemia, making Oldpeak one of the most powerful continuous predictors in this dataset.

3.5 Heatmap

The correlation heatmap (Figure 3.5) highlights relationships between features in the dataset. Heart disease shows strong positive correlation with Oldpeak, exercise-induced angina, and flat ST slope, while it is negatively correlated with MaxHR and upsloping ST slope. Age also correlates moderately with disease status. By contrast, resting blood pressure and cholesterol exhibit only weak correlations with the target, suggesting they may not be strong predictors in this dataset.

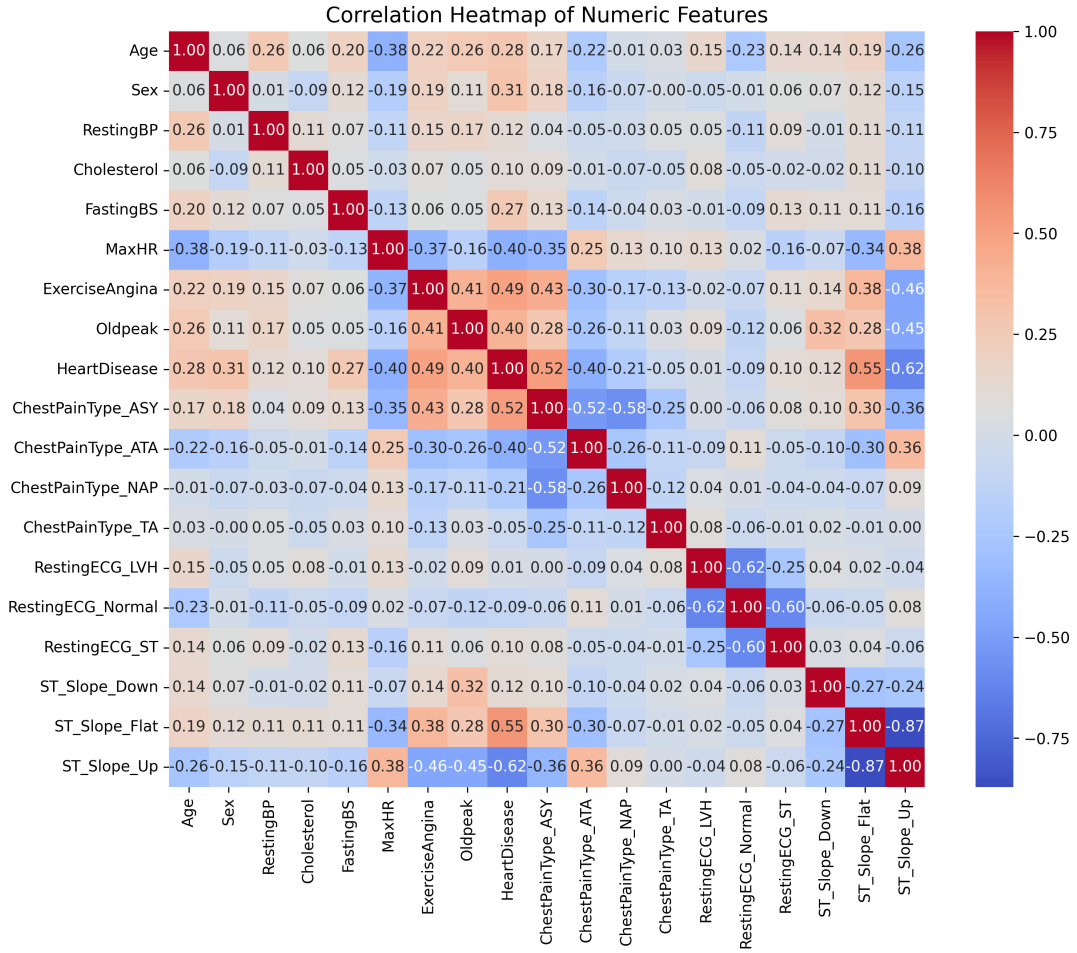

Figure 3.5: Correlation heatmap of features.

- **Age**
  Correlation: Moderately positive correlation with heart disease.
  Insight: Older age is a consistent cardiovascular risk factor, aligning with observed higher prevalence in older patients.

- **Oldpeak**
  Correlation: Strong positive correlation with heart disease.
  Insight: Greater ST depression indicates ischemia, making Oldpeak a powerful continuous predictor.

- **ExerciseAngina**
  Correlation: Strong positive correlation with heart disease.

Insight: Angina triggered by exercise signals restricted blood flow in coronary arteries.

- **MaxHR**
  Correlation: Strong negative correlation with heart disease.
  Insight: Patients unable to achieve high heart rates during exertion are more likely to have compromised cardiac function.

- **ST_Slope**
  Correlation: Flat ST slope is positively correlated, while upsloping is strongly negatively correlated with disease.
  Insight: Flat or downsloping ST segments are clinical indicators of ischemia, whereas upsloping is generally protective.

- **RestingBP & Cholesterol**
  Correlation: Very weak correlation with heart disease.
  Insight: Although both are traditional cardiovascular risk factors, within this dataset they do not provide strong separation between diseased and non-diseased groups.

Chapter 4

Model Development

4.1 Tools and Libraries

In this study, we utilized **Python** with a comprehensive set of data science and machine learning libraries to implement and evaluate predictive models:

- **Data handling:**
  - `pandas`: data manipulation.
  - `numpy`: numerical computation and matrix operations.
- **Data splitting & hyperparameter optimization:**
  - `train_test_split` from `sklearn.model_selection`: dataset partitioning.
  - `StratifiedKFold` from `sklearn.model_selection`: stratified sampling.
  - `GridSearchCV` from `sklearn.model_selection`: hyperparameter tuning.
- **Data preprocessing:**
  - `ColumnTransformer` from `sklearn.compose`.
  - `StandardScaler` from `sklearn.preprocessing`.
- **Pipeline:**
  - `Pipeline` from `sklearn.pipeline`: reproducible workflows.
- **Model evaluation:**
  - `roc_auc_score`, `classification_report`, `confusion_matrix`,`make_scorer`, `average_precision_score` from `sklearn.metrics`.
- **Machine learning models:**
  - `LogisticRegression` from `sklearn`.
  - `SVC` (Support Vector Classifier) from `sklearn`.
  - `RandomForestClassifier` from `sklearn`.
  - `XGBClassifier` from `xgboost`.

- **Model interpretability & feature analysis:**
  - `permutation_importance` from `sklearn.inspection`.
  - **SHAP** (SHapley Additive exPlanations): model interpretation, summary plots, force plots.

This toolkit supported the end-to-end workflow from **preprocessing → model training → hyperparameter tuning → evaluation → interpretability**.

4.2 Data Preparation

- **Train/Test Split:**
  The dataset is divided into training and testing sets with an 80/20 ratio. Stratification by the target label (`HeartDisease`) is applied to preserve the 0/1 distribution across both sets, ensuring more stable evaluation in the case of class imbalance. A fixed random seed is used to guarantee reproducibility.

- **Feature Grouping:**
  Features are organized into two groups for preprocessing:
  - **Continuous variables:** Age, RestingBP, Cholesterol, MaxHR, Oldpeak.
  - **Binary variables:** Sex, FastingBS, ExerciseAngina (values 0/1).

  In addition, some columns are already one-hot encoded; these are retained as-is to leverage pre-existing encoding.

- **Preprocessing with ColumnTransformer:**
  - Continuous variables are standardized using `StandardScaler` (zero mean, unit variance) to improve convergence and ensure consistent distance measures.
  - Binary variables are kept unchanged (`passthrough`) to preserve their interpretability.
  - Pre-encoded one-hot columns are preserved via the "remainder passthrough" option.

  This design treats each group of features appropriately and avoids unnecessary scaling of indicator variables.

- **Pipeline Organization:**
  All preprocessing steps are wrapped in a `Pipeline`, ensuring:
  - No data leakage: scaling parameters are estimated only on the training set and then applied to the test set.
  - Reproducibility and automation: transformations are applied consistently across runs and models.
  - Extendability: models such as Logistic Regression, SVM, Random Forest or XGBoost can be added after preprocessing.

- **Cross-Validation Scheme:**
  A `Stratified K-Fold` procedure (k=5) is used for tuning and evaluation. Stratification ensures robust performance estimation by maintaining label distribution within each fold.

- **Column Name Restoration:**
  After transformation, feature names (continuous, binary, and pre-encoded one-hot) are reconstructed into a new DataFrame. This enables:

  – Monitoring of data structure (dimensions, column order).

  – Interpretability in post-hoc analyses, e.g., Permutation Importance and SHAP.

4.3 Logistic Regression

Logistic Regression is a machine learning model that belongs to the family of linear models and is commonly used for binary classification problems, such as predicting whether a customer will make a purchase or whether a patient has heart disease.

Although its name includes the word "regression", Logistic Regression is actually applied for classification rather than predicting continuous values. The key difference lies in the fact that instead of directly predicting a numerical outcome, Logistic Regression estimates the probability that an observation belongs to a specific class (e.g., disease/no disease).

4.3.1 How Logistic Regression works

Linear relationship. The model assumes that the target variable y can be expressed as a linear combination of the features X:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Logistic (sigmoid) function. To transform the value z, which can range from $(-\infty, +\infty)$, into a probability within $[0, 1]$, Logistic Regression uses the sigmoid function:

$$p = \frac{1}{1 + e^{-z}}$$

Here, p represents the probability that the sample belongs to the positive class (e.g., having the disease).

Classification rule. If $p \geq 0.5$, the model predicts class 1. If $p < 0.5$, the model predicts class 0. The threshold of 0.5 can be adjusted to balance precision and recall depending on the problem.

Parameter estimation. The coefficients β are estimated by maximizing the likelihood function (Maximum Likelihood Estimation – MLE), rather than by minimizing squared error as in Linear Regression.

4.3.2 Implementation of Logistic Regression

In this study, the Logistic Regression model was implemented within a machine learning pipeline that included preprocessing and hyperparameter tuning. The model was

trained using the training dataset and evaluated with cross-validation and a separate test set.

First, the pipeline was constructed with two main components:

- **Preprocessing (`prep`)**: numerical features were standardized using `StandardScaler`, while binary features were passed through without scaling.
- **Classifier (`clf`)**: a Logistic Regression model with maximum iterations set to 1000 and a fixed random seed for reproducibility.

A grid search with cross-validation (`GridSearchCV`) was applied to optimize the hyperparameters of the Logistic Regression model. The parameter grid included:

- `penalty`: the type of regularization applied, set to L2 regularization in this case.
- `C`: the inverse of the regularization strength, tested with values $\{0.01, 0.1, 1, 10\}$. A smaller value of `C` indicates stronger regularization.
- `solver`: the optimization algorithm used to fit the model, tested with `liblinear` (suitable for smaller datasets and L1/L2 penalties) and `lbfgs` (efficient for larger datasets and L2 penalty).
- `class_weight`: either `None` (no adjustment) or `balanced` (adjusts weights inversely proportional to class frequencies to handle class imbalance).

The grid search was performed with stratified cross-validation using the ROC-AUC score as the evaluation metric. After identifying the best hyperparameter combination, the model was refitted on the full training set. The performance was then assessed on the test set using ROC-AUC, as well as standard classification metrics such as precision, recall, and F1-score.

4.3.3 Results of Logistic Regression

The best hyperparameters were:

```
C = 1, class_weight = balanced, penalty = L2, solver = lbfgs
```

The model achieved a cross-validation ROC-AUC of **0.9221** and a test ROC-AUC of **0.9320**.

Table 4.1: Classification report for Logistic Regression on the test set

Class / Metric	Precision	Recall	F1-score	Support
Class 0	0.85	0.88	0.86	82
Class 1	0.90	0.87	0.89	102
Accuracy		0.88		184
Macro avg	0.87	0.88	0.87	184
Weighted avg	0.88	0.88	0.88	184

Discussion.

- For Class 0 (negative class): precision = 0.85, recall = 0.88, F1-score = 0.86. The model correctly identifies most negative cases, with a relatively high recall but slightly lower precision.
- For Class 1 (positive class): precision = 0.90, recall = 0.87, F1-score = 0.89. The model is effective in detecting positive cases with both high precision and recall, indicating few false alarms.
- Overall accuracy is 0.88 across 184 samples, showing stable performance.
- Macro-averaged metrics (precision = 0.87, recall = 0.88, F1 = 0.87) indicate balanced performance between the two classes.
- Weighted averages (all 0.88) confirm that the use of `class_weight=balanced` effectively addressed class imbalance in the dataset.

In summary, the Logistic Regression model demonstrates strong predictive ability, balanced class performance, and robustness, making it a reliable baseline for binary classification in this study.

4.4 Support Vector Classification

Support Vector Machine (SVM) is a supervised learning algorithm that is widely used for classification tasks. The core idea of SVM is to find the optimal hyperplane that separates data points of different classes with the maximum margin. The margin is defined as the distance between the separating hyperplane and the nearest data points from each class, known as support vectors. By maximizing this margin, SVM aims to achieve better generalization on unseen data.

4.4.1 How Support Vector Machine (SVM) works

Mathematically, for linearly separable data, the decision function can be written as:

$$f(x) = w^T x + b,$$

where w is the weight vector and b is the bias. A sample is classified based on the sign of $f(x)$. For non-linear cases, SVM employs kernel functions (such as the radial basis function, RBF) to map the original input space into a higher-dimensional space where a linear separation becomes possible.

In addition, SVM can output class probabilities by applying methods such as Platt scaling, which calibrates decision values into probabilities. This makes SVM not only a powerful classifier but also suitable for evaluation with metrics like ROC-AUC.

4.4.2 Implementation of SVM

In this study, the Support Vector Classifier (SVC) was implemented using a machine learning pipeline consisting of preprocessing and hyperparameter tuning. Numerical features were standardized while binary features were kept as is. The SVC model was trained with probability estimation enabled (`probability=True`) to allow ROC-AUC evaluation.

The hyperparameters were optimized using grid search with cross-validation (`GridSearchCV`). The parameter grid included:

- `kernel`: type of kernel function, tested with `linear` and `rbf` (Radial Basis Function).
- `C`: regularization parameter with values $\{0.1, 1, 10\}$, controlling the trade-off between maximizing margin and minimizing classification error.
- `gamma`: kernel coefficient for RBF kernel, tested with `scale` (default) and `auto`.
- `class_weight`: either `None` or `balanced`, to address class imbalance by adjusting weights inversely proportional to class frequencies.

The grid search used ROC-AUC as the scoring metric with stratified cross-validation. The best parameter combination was then refitted on the full training set and evaluated on the test set.

4.4.3 Results of SVC

The best hyperparameters were:

```
C = 1, class_weight = balanced, gamma = auto, kernel = rbf
```

The model achieved a cross-validation ROC-AUC of **0.9201** and a test ROC-AUC of **0.9366**.

Table 4.2: Classification report for Support Vector Machine on the test set

Class / Metric	Precision	Recall	F1-score	Support
Class 0	0.83	0.87	0.85	82
Class 1	0.89	0.85	0.87	102
Accuracy		0.86		184
Macro avg	0.86	0.86	0.86	184
Weighted avg	0.86	0.86	0.86	184

Discussion.

- For Class 0 (negative class): precision = 0.83, recall = 0.87, F1-score = 0.85. The model detects most negative cases correctly, but precision is slightly lower, meaning some positive samples are misclassified as negative.
- For Class 1 (positive class): precision = 0.89, recall = 0.85, F1-score = 0.87. This indicates that the model identifies the majority of positive cases with relatively high precision.
- The overall accuracy of 0.86 shows stable and consistent performance across the dataset.
- Macro-averaged metrics (all equal to 0.86) demonstrate that performance between the two classes is balanced.
- Weighted averages (all 0.86) confirm that the use of `class_weight=balanced` helped mitigate class imbalance effectively.

4.5 Random Forest

Random Forest (RF) is an ensemble learning method based on decision trees, designed to improve predictive accuracy and control overfitting. It constructs multiple decision trees during training and outputs the final prediction by aggregating the results of individual trees (majority voting for classification, averaging for regression).

4.5.1 How Random Forest works

The key idea behind Random Forest is to introduce randomness in two main ways:

- **Bootstrap sampling (bagging):** Instead of training each tree on the entire dataset, Random Forest generates different training subsets by sampling the data with replacement. This means that each tree is exposed to slightly different training data, which increases diversity among the trees.
- **Random feature selection:** When a tree splits at a node, it does not evaluate all available features but only considers a random subset. This ensures that not all trees rely on the same dominant predictors, further reducing correlation between them.

After training, the predictions of all trees are aggregated. For classification problems, Random Forest applies majority voting across the trees, while for regression it takes the average of all tree outputs. This aggregation process leads to more stable and accurate predictions compared to individual decision trees.

Random Forest is particularly effective for tabular datasets that include complex feature interactions, heterogeneous feature types, and even noisy data. It requires less hyperparameter tuning than boosting methods, is less prone to overfitting than single decision trees, and can also provide estimates of feature importance, making it both powerful and interpretable.

4.5.2 Implementation of Random Forest

The Random Forest model was implemented using a pipeline combining preprocessing and classification. A grid search with cross-validation (`GridSearchCV`) was performed to tune the following hyperparameters:

- `n_estimators`: number of trees in the forest, tested with 200 and 500.
- `max_depth`: maximum depth of each tree, tested with `None`, 5, 10, and 20.
- `min_samples_split`: minimum number of samples required to split an internal node, tested with 2, 5, and 10.
- `max_features`: number of features considered when looking for the best split, tested with `sqrt` and `log2`.

4.5.3 Results of Random Forest

The best hyperparameters were:

```
max_depth = 5, max_features = sqrt, min_samples_split = 5, n_estimators = 500
```

The model achieved a cross-validation ROC-AUC of **0.9290** and a test ROC-AUC of **0.9229**.

Table 4.3: Classification report for Random Forest on the test set

Class / Metric	Precision	Recall	F1-score	Support
Class 0	0.86	0.82	0.84	82
Class 1	0.86	0.89	0.88	102
Accuracy		0.86		184
Macro avg	0.86	0.85	0.86	184
Weighted avg	0.86	0.86	0.86	184

Discussion.

- For Class 0 (negative class): precision = 0.86, recall = 0.82, F1-score = 0.84. The model correctly identifies most negative cases, but recall is slightly lower, meaning some negatives are misclassified as positives.
- For Class 1 (positive class): precision = 0.86, recall = 0.89, F1-score = 0.88. This shows strong performance in detecting positive cases with good balance between precision and recall.
- The overall accuracy is 0.86 across 184 samples, consistent with the performance of other models tested.
- Macro-averaged metrics (precision = 0.86, recall = 0.85, F1 = 0.86) show balanced performance across both classes.
- Weighted averages (all 0.86) confirm stable model behavior, even under class imbalance.

4.6 Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is a powerful ensemble learning method based on the gradient boosting framework. It builds an additive model in a forward stage-wise manner, where new decision trees are iteratively added to correct the errors made by previous trees. At each iteration, the model fits a new weak learner (usually a shallow decision tree) to the negative gradient of the loss function, effectively minimizing the overall error.

XGBoost introduces several enhancements over traditional gradient boosting, such as regularization to prevent overfitting, efficient handling of missing values, parallelized tree construction, and shrinkage via the learning rate. These improvements make XGBoost highly effective for both classification and regression tasks, and it has become one of the most widely used algorithms in data science competitions and real-world applications.

4.6.1 How XGBoost works

XGBoost (Extreme Gradient Boosting) is an ensemble learning method that builds on the principle of gradient boosting. Instead of training trees independently as in Random Forest, XGBoost trains decision trees sequentially, where each new tree is added to correct the errors of the previous ensemble. By combining many weak learners in this way, the algorithm gradually improves overall prediction accuracy.

At each step, XGBoost focuses more on the instances that were misclassified in previous iterations, allowing the model to progressively reduce errors. A learning rate is applied to control how much each new tree contributes, which helps prevent overfitting. In addition, XGBoost introduces regularization techniques to penalize overly complex trees, making the model more robust and generalizable.

Compared to traditional gradient boosting, XGBoost offers several enhancements:

- It supports both L_1 and L_2 regularization to control complexity.
- It applies shrinkage (learning rate) and column subsampling, reducing variance and improving generalization.
- It can efficiently handle missing values and is optimized for speed with parallelized tree construction.

These improvements make XGBoost highly effective for tabular data and one of the most widely used algorithms in data science competitions and real-world applications.

4.6.2 Implementation of XGBoost

In this study, the XGBoost classifier (`XGBClassifier`) was implemented within a machine learning pipeline. The model was trained with the `binary:logistic` objective function for binary classification and evaluated using the `logloss` metric.

The hyperparameters were optimized using grid search with cross-validation (`GridSearchCV`). The parameter grid included:

- `n_estimators`: number of boosting rounds, tested with 200 and 500.
- `max_depth`: maximum depth of trees, tested with 3, 5, and 7.
- `learning_rate`: shrinkage step size, tested with 0.01 and 0.1.
- `subsample`: fraction of samples used per boosting round, tested with 0.8 and 1.0.
- `colsample_bytree`: fraction of features used per tree, tested with 0.8 and 1.0.
- `reg_lambda`: L2 regularization parameter, tested with 1 and 5.

4.6.3 Results of XGBoost

The best hyperparameters were:

```
colsample_bytree = 0.8, learning_rate = 0.01, max_depth = 3

n_estimators = 500, reg_lambda = 5, subsample = 0.8
```

The model achieved a cross-validation ROC-AUC of **0.9328** and a test ROC-AUC of **0.9318**.

Table 4.4: Classification report for XGBoost on the test set

Class / Metric	Precision	Recall	F1-score	Support
Class 0	0.90	0.84	0.87	82
Class 1	0.88	0.92	0.90	102
Accuracy		0.89		184
Macro avg	0.89	0.88	0.88	184
Weighted avg	0.89	0.89	0.89	184

Discussion.

- For Class 0 (negative class): precision = 0.90, recall = 0.84, F1-score = 0.87. The model performs well in identifying negative cases, though the slightly lower recall indicates that some negatives were misclassified as positives.
- For Class 1 (positive class): precision = 0.88, recall = 0.92, F1-score = 0.90. This shows that the model is highly effective at capturing positive instances with balanced precision and recall.
- The overall accuracy is 0.89, reflecting strong predictive performance and improvement compared with baseline models.
- The macro-averaged metrics (precision = 0.89, recall = 0.88, F1-score = 0.88) demonstrate that the classifier maintains balanced performance across both classes.
- Weighted averages (all around 0.89) confirm that XGBoost provides stable and robust results even under moderate class imbalance.

4.7 Summary of results

To evaluate the performance of different machine learning algorithms on the heart disease prediction task, four models were implemented and tuned: Logistic Regression, Support Vector Machine (SVC), Random Forest, and XGBoost. Each model was optimized using cross-validation with ROC-AUC as the scoring metric, and the best hyperparameters were selected.

The final performance of each model was then assessed on the test set using two main evaluation metrics:

Table 4.5: Detailed performance comparison of models on the test set

Model	CV ROC-AUC	Test ROC-AUC	Accuracy
LR	0.9221	0.9320	0.88
SVC	0.9201	0.9366	0.86
RF	0.9290	0.9229	0.86
XGB	0.9328	0.9318	0.89

- **Logistic Regression (LR):** achieved an accuracy of 0.88 and a Test ROC-AUC of 0.9320, indicating both high predictive power and consistency between cross-validation (0.9221) and test results. Despite its simplicity, it remains a strong and interpretable baseline.
- **Support Vector Machine (SVC):** obtained the highest Test ROC-AUC (0.9366), showing the best discriminative ability among all models. However, its accuracy (0.86) is slightly lower than Logistic Regression, and the difference between CV (0.9201) and test results suggests less stability.
- **Random Forest (RF):** produced a relatively high CV ROC-AUC (0.9290) but the lowest Test ROC-AUC (0.9229), suggesting that it may overfit slightly and generalize less effectively compared to the other models.
- **XGBoost (XGB):** delivered the most balanced performance, with the highest CV ROC-AUC (0.9328), a strong Test ROC-AUC (0.9318), and the highest accuracy (0.89). The close agreement between CV and test scores demonstrates strong stability and robustness across datasets.

Conclusion. If the objective is to use a simple and interpretable model, Logistic Regression is the most appropriate choice due to its strong accuracy and robustness. If the goal is to maximize discriminative power, SVC is the best performer with the highest Test ROC-AUC. However, considering both stability and overall performance, **XGBoost** can be regarded as the most suitable model. It offers a good trade-off between accuracy and ROC-AUC, while maintaining strong generalization across training and test sets.

4.7.1 SHapley Additive exPlanations

In order to better understand the internal decision-making mechanism of machine learning models, the SHapley Additive exPlanations (**SHAP**) method is employed. SHAP is a game-theoretic approach to explain the output of complex models by assigning each feature an importance value that reflects its contribution to the final prediction. Unlike traditional feature importance methods, SHAP provides consistent and locally accurate explanations, allowing researchers to quantify both the direction (positive or negative effect) and the magnitude of each features impact. This makes SHAP particularly valuable in sensitive domains such as healthcare, where interpretability and trust in model predictions are crucial.

The following figure illustrates the top 10 features with the highest mean |SHAP| values, indicating their average contribution to the prediction of heart disease across the dataset.

The SHAP summary bar chart shows the top 10 most influential features in predicting the target variable (Heart Disease). Among them, **ST_Slope_Up** and **ChestPainType_ASY** stand out with significantly higher mean |SHAP| values compared to other features, meaning they contribute the most to the models predictions.

- **ST_Slope_Up**: This feature dominates the models decision-making, indicating that patients with an "up-sloping" ST segment in their ECG strongly influence the prediction.

Model/shap1.png Model/shap1.bb

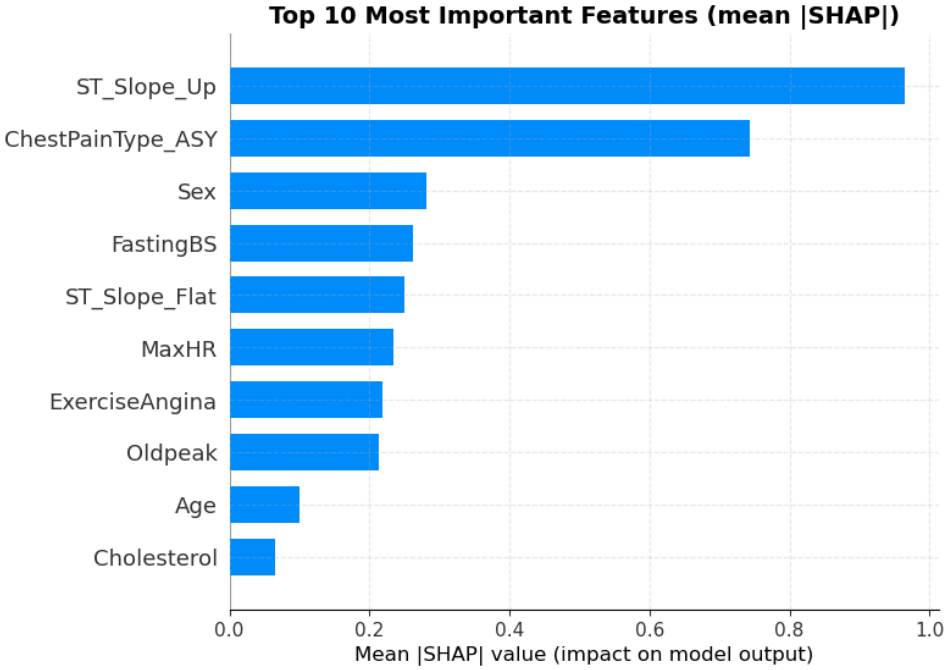

Figure 4.1: Top 10 most important features based on mean |SHAP| values.

- **ChestPainType_ASY**: Asymptomatic chest pain is also a highly impactful variable, aligning with medical understanding that silent or atypical chest pain often signals underlying heart issues.
- **Sex and FastingBS**: These features play a moderate role, suggesting biological and metabolic differences are still important risk factors.
- **ST_Slope_Flat, MaxHR, ExerciseAngina, Oldpeak**: These features cluster in the mid-tier, showing that exercise-related ECG patterns and heart stress measurements remain relevant.
- **Age and Cholesterol**: Surprisingly, they appear at the bottom, implying that for this dataset, age and cholesterol levels contribute less to predictions compared to ECG-based and symptom-based features.

1. The dominance of ECG-related features (*ST_Slope, ExerciseAngina, Oldpeak*) indicates that heart function under stress and ECG abnormalities are more predictive than traditional risk factors like age or cholesterol.
2. **ChestPainType_ASY** suggests that asymptomatic patients should not be overlooked—screening silent cases is critical for early intervention.
3. The relatively low contribution of **Age** and **Cholesterol** might be dataset-specific (sample bias) or reflect that modern diagnostic tools capture risk more effectively than traditional markers.

4.7.2 Permutation Importance

Permutation Importance is a model-agnostic method to evaluate feature importance by measuring the decrease in model performance (e.g., ROC-AUC) when the values of a feature are randomly shuffled. If shuffling a feature leads to a substantial drop in performance, that feature is considered important. Compared to SHAP, permutation importance provides a more direct, performance-based perspective, highlighting how much each feature contributes to predictive accuracy.

Model/perm1.png Model/perm1.bb

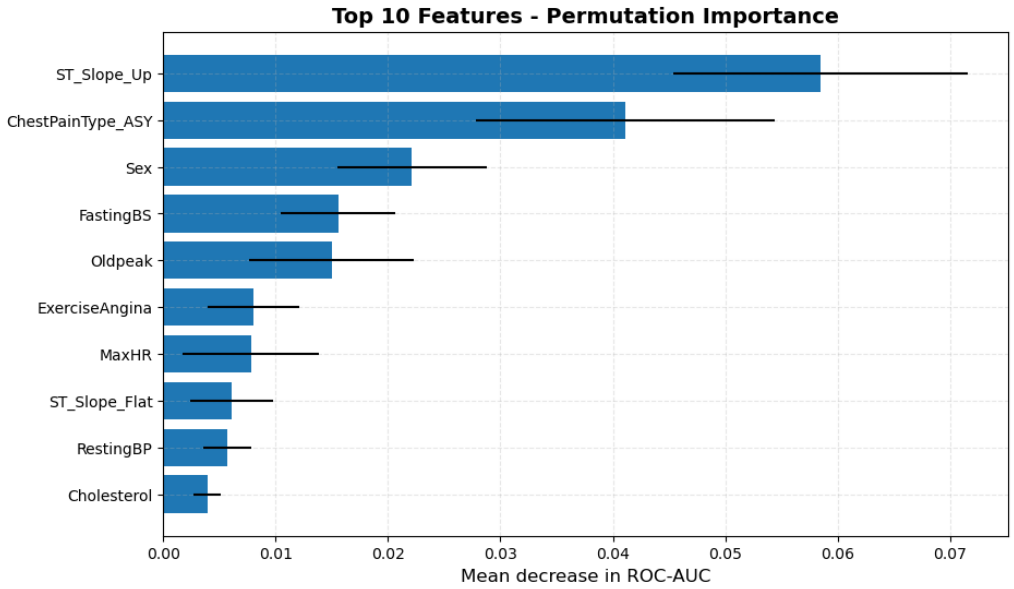

Figure 4.2: Top 10 features based on permutation importance (mean decrease in ROC-AUC).

- The permutation importance results confirm the dominance of **ST_Slope_Up** and **ChestPainType_ASY**, which cause the largest decrease in ROC-AUC when shuffled, indicating that the model heavily relies on them.
- Features such as **Sex, FastingBS, and Oldpeak** also show moderate importance.
- Traditional clinical indicators like **Cholesterol and RestingBP** contribute minimally.
- Compared to SHAP, the ranking is broadly consistent, though permutation importance directly quantifies predictive reliance.

1. ECG-related features remain the strongest predictors, reinforcing their clinical value in automated heart disease detection.
2. The alignment between SHAP and permutation importance strengthens confidence in the reliability of these top predictors.
3. The weak contribution of traditional markers such as cholesterol suggests that modern predictive models prioritize functional and symptomatic signals over static baseline measures.

Chapter 5

User Interface Implementation

5.1 Model Persistence with Joblib

To facilitate deployment, the trained model was saved using the `Joblib` library. Joblib is a Python package optimized for serializing and deserializing large objects, especially those containing NumPy arrays such as machine learning models. It provides a more efficient alternative to the standard `pickle` module when handling models and pipelines. In this project, the best-performing XGBoost pipeline obtained from GridSearchCV was exported and stored as a `xgb_pipeline.pkl` file. This approach allows the web application to directly load the pre-trained model without retraining, ensuring faster response time and smoother integration into the user interface.

5.2 Create app.py file

The application is a **web app for heart disease prediction**. Users input their medical information (such as age, sex, blood pressure, cholesterol, chest pain type, ECG results, etc.). The system then transforms the data into the required model format (via one-hot encoding) and feeds it into a pre-trained XGBoost model (`xgb_pipeline.pkl`) to predict the probability of having heart disease.

- **Application Initialization**: The application is implemented using the Flask framework to build a simple web-based system. Upon execution, it loads the pre-trained model (stored in `xgb_pipeline.pkl`) through the Joblib library. This model is based on XGBoost and is used to predict the likelihood of heart disease.
- **Communication Between User and Server**: The application defines a main route ("/") to handle two types of requests:
  - **GET**: displays the web interface and input form.
  - **POST**: receives the submitted data from the form when the user requests a prediction.
- **Input Data Processing**: When a POST request is received, the system performs the following steps:
  - Retrieves user inputs (age, sex, blood pressure, cholesterol, maximum heart rate, fasting blood sugar, chest pain type, ECG results, etc.).

  - Validates the inputs:
    * Age must be between 0 and 120.
    * Variables must match the required format (e.g., sex $\in \{M, F\}$, FastingBS $\in \{0, 1\}$).
    * Invalid inputs return an error message to the interface.
  - Transforms the data:
    * Converts some variables into numerical values (e.g., sex $\rightarrow$ 0/1).
    * Converts categorical variables (ChestPainType, ST_Slope, RestingECG) into one-hot encoded features.

- **Data Preparation for the Model**: After transformation, the input is organized into a Pandas DataFrame, ensuring the columns match those required by the trained model.

- **Prediction Using the Model**: The system applies the `predict_proba` method of the XGBoost model to calculate the probability of heart disease. If the probability is greater than or equal to 50%, the prediction is labeled as **Positive**; otherwise, it is labeled as **Negative**. The result includes both the probability value and the prediction label.

- **Returning Results to the Interface**: The prediction result is then sent back to `index.html` to be displayed for the user. If any error occurs during input processing, an error message is returned and shown in the result box on the interface.

5.3 Create index.html file

- This is the **frontend interface** of the application, developed using HTML and Bootstrap 5.

- **Design**: blue gradient background, white rounded form with shadow, and blue buttons.

- The input form consists of **11 main fields**: age, sex, resting blood pressure, cholesterol, maximum heart rate, oldpeak, fasting blood sugar, exercise angina, chest pain type, ST slope, and resting ECG.

- Tooltips () are provided to explain the meaning of each field for better user guidance.

- Data is submitted via **POST** to the Flask backend when the Predict button is pressed.

- A **loading spinner** is displayed while the prediction is being processed.

- The prediction result (probability percentage and Positive/Negative label) is displayed in a dedicated result box.

Chapter 6

Conclusion and References

6.1 Conclusion

This project set out to design and implement a machine learning-based system for predicting the probability of heart disease. The core contribution of this work lies in the successful integration of a robust classification model with a user-friendly web interface, providing an accessible tool that demonstrates the potential of data-driven methods in healthcare applications.

The XGBoost algorithm, selected for its strong performance in handling structured tabular data, was trained and tuned using the given dataset. Through preprocessing steps such as normalization and one-hot encoding, the data was transformed into a form suitable for the model, ensuring accurate handling of both numerical and categorical features. The models performance was validated with cross-validation, confirming its ability to generalize to unseen cases. Once trained, the optimal pipeline was exported using Joblib, which enabled seamless deployment without the need to retrain the model each time the application is launched.

The backend of the system was developed using Flask, which provided the framework to connect the trained model with the user-facing interface. The web application accepts user inputs such as age, sex, blood pressure, cholesterol, chest pain type, ECG results, and other relevant health parameters. The application validates the inputs, converts them into the correct numerical and categorical representations, and sends them to the prediction model. The system then returns both the probability of heart disease and a clear classification outcome (Positive/Negative). This real-time interaction between the user and the predictive model demonstrates how machine learning can be deployed in a practical and efficient manner.

On the frontend side, a responsive and visually appealing user interface was built using HTML and Bootstrap. The design focused on clarity, accessibility, and usability, incorporating tooltips to explain the meaning of each input field and a loading spinner to enhance the user experience. The result presentation is concise and easy to interpret, showing both percentage probability and categorical prediction, ensuring that the output is understandable even for non-technical users.

Overall, the project demonstrates that predictive analytics, when carefully combined with modern web technologies, can be transformed into practical tools that have real-world impact. In this case, the system highlights how machine learning can support the early detection of heart disease by providing rapid, data-driven risk assessments. Although it is not intended to replace medical expertise, the application illustrates the

potential of such systems to act as decision-support tools for both patients and healthcare providers.

Future work can extend this project in multiple directions. First, expanding the dataset with more samples and additional clinical features would likely improve the models accuracy and generalizability. Second, integrating explainability techniques such as SHAP (SHapley Additive exPlanations) could help users and clinicians better understand the reasoning behind predictions, thereby increasing trust in the system. Third, deploying the system to a cloud environment could make the application accessible to a wider audience and allow for scalable performance. Finally, integrating the predictive tool with electronic health record (EHR) systems could create a more comprehensive platform to support preventive care and personalized medicine.

In conclusion, this project not only achieved its initial objective of building a heart disease prediction system but also laid the groundwork for further research and development. It highlights the synergy between machine learning, software engineering, and user interface design in creating innovative solutions to healthcare challenges. By bridging the gap between data science and practical application, the project demonstrates the real-world potential of predictive analytics in improving health outcomes and supporting medical decision-making.

6.2 References

1. Fedesoriano. (2021). *Heart Failure Prediction Dataset*. Kaggle. Available at: `https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction`
2. Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30. Available at: `https://arxiv.org/abs/1705.07874`
3. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785794. Available at: `https://arxiv.org/abs/1603.02754`
4. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 532. Springer.
5. Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2), 215242.
6. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3), 273297. Springer.
7. Little, R. J., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data* (3rd ed.). Wiley.
8. Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.
9. Kanchitank. (2022). *Medibuddy Smart Disease Predictor*. GitHub repository. Available at: `https://github.com/kanchitank/Medibuddy-Smart-Disease-Predictor`
10. Projects Developer. (2021). *Heart Diseases Prediction Project*. GitHub repository. Available at: `https://github.com/Projects-Developer/Heart-Diseases-Prediction-Project`
11. Sharma, A. (2021). *Heart Disease Prediction Deployment*. GitHub repository. Available at: `https://github.com/asthasharma98/Heart-Disease-Prediction-Deployment`