A

PROJECT REPORT

ON

“AGE & GENDER IMPACT ON CARDIOVASCULAR DISEASE PREDICTION IN INDIAN POPULATION USING MACHINE LEARNING”

SUBMITTED

To

CENTRE FOR ONLINE LEARNING

Dr. D. Y. PATIL VIDYAPEETH, PUNE

IN PARTIAL FULFILMENT OF DEGREE OF

MASTER OF BUSINESS ADMISTRATION

BY

SURABHI PANDEY

PRN: 23050202595

BATCH 2023-2025

CERTIFICATE

This is to certify that Ms. Pandey Surabhi Sunil Kumar, PRN 23050202595 has completed her internship at Qollabb EduTech **s**tarting from 12.11.2024 to 16.12.2024. Her project work was a part of the MBA (ONLINE LEARNING). The project is on Age & Gender Impact on Cardiovascular Disease Prediction in Indian Population using Machine Learning which includes research as well as industry practices. She was very sincere and committed in all tasks.

Project Guide **Director**

Prof. Vikas Pawar Dr. Safia Farooqui

Date – 19/12/2024

16.12.2024

To Whomsoever It May Concern

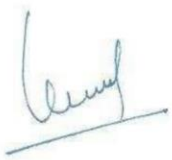
This is to certify that Ms. Pandey Surabhi Sunil Kumar, PRN 23050202595, has completed her project-based internship starting from 12.11.2024 to 16.12.2024.

Her project work was a part of the MBA (ONLINE LEARNING).

The project is Age & Gender impact on Cardiovascular disease prediction in Indian population using Machine learning, which includes research as well as industry practices.

She was very sincere and committed in all tasks.

For Qollabb EduTech Private Limited



Vipendra Singh
Chief Executive Officer

Qollabb EduTech Private Limited
231/4, SF II, Rashtrakavi Kuvempu Nagar, Behind Central Silk Board Building,BTM 2nd Stage Bengaluru, Karnataka-560068, India
http://www.qollabb.com support@qollab.com +918800047232

DECLARATION BY LEARNER

This is to declare that I have carried out this project work myself in part fulfillment of the M.B.A Program of Centre for Online Learning of Dr. D.Y. Patil Vidyapeeth's, Pune – 411018

The work is original, has not been copied from anywhere else, and has not been submitted to any other University / Institute for an award of any degree / diploma.

Date: 15/12/2024 Signature: 

Place: Pune Name: Surabhi Pandey

ACKNOWLEDGEMENT

It gives us great pleasure to present the project report on **Age & Gender Impact on Cardiovascular Disease Prediction In Indian Population Using Machine Learning.**

I would like to express my deep and sincere gratitude to my guide, **Dr. Vandana Sivaraj a domain expert at Qollabb Edutech**, for her unflagging support and continuous encouragement throughout the project work. Without her guidance and persistent help, this report would not have been possible.

I would like to express my gratitude towards the Head of Department for the kind cooperation and encouragement which helped me in the completion of this project. Furthermore, I would like to acknowledge with much appreciation the crucial role of the staff of DIT, Pimpri, who gave the permission to use all the required equipment and the necessary materials to complete my project.

Lastly, I am deeply grateful to my family for their constant love, support, and understanding during the project. Their encouragement and belief in my abilities have been a source of motivation throughout this journey, and I sincerely appreciate their unwavering presence every step of the way.

Surabhi Pandey

PRN: 23050202595

Table of Content

Executive Summary

Cardiovascular diseases (CVD) are the leading cause of death worldwide, accounting for millions of lives lost each year. Early detection and prevention of CVD are crucial for reducing mortality rates and improving patient outcomes. However, traditional diagnostic methods can be time-consuming, costly, and often fail to detect disease in its early stages. This project aims to address these challenges by developing a machine learning (ML)-based cardiovascular disease prediction system that can accurately identify individuals at risk of CVD before symptoms manifest.

The core objective of the project is to utilize machine learning algorithms to analyze critical health indicators such as age, gender, and other relevant risk factors. By training these models on historical data, we can predict the likelihood of cardiovascular events in patients with higher precision and speed than conventional methods. The ultimate goal is to enable healthcare professionals to make data-driven decisions, provide timely interventions, and improve patient care.

Key benefits of this ML-based prediction system include:

- **Early Detection:** Identifying high-risk individuals before the disease progresses, allowing for preventive measures like lifestyle changes or medication.
- **Improved Accuracy:** Machine learning models can process large datasets and uncover complex patterns, leading to more accurate predictions compared to traditional statistical methods.
- **Cost-Effectiveness:** Early intervention reduces the long-term healthcare costs associated with treating advanced CVD, benefiting both patients and healthcare systems.
- **Personalized Healthcare:** By analyzing individual patient data, ML models can offer personalized risk assessments, leading to tailored prevention and treatment plans.

This project seeks to harness the power of machine learning to revolutionize cardiovascular disease management. By improving the ability to predict and prevent CVD, this project has the potential to save lives, enhance the quality of care, and reduce the overall burden of cardiovascular diseases on healthcare systems globally.

CHAPTER 1: INTRODUCTION

1. Company Profile

Company Name: Qollabb Edutech Private Limited

Headquarters: Bangalore, Karnataka

Partner with Qollabb: Over 250 top employers

Collaborations With: Over 100 institutions and over 5000 students across India

Qollabb connects students, universities, and top companies by offering internships, fresher jobs, and live industry projects across various sectors such as IT, marketing, finance, and digital marketing. With projects and internships available in major cities like Noida, Bangalore, Indore, New Delhi, and Pune, students gain practical experience that enhances their employability.

Key Offering

Qollabb offers a wide range of opportunities for students to gain practical experience through various industry projects, internships, and fresher jobs. These industry projects span fields such as IT, marketing, finance, and digital marketing, providing real-world exposure. Internships are available in diverse areas like software, finance, and digital marketing, helping students enhance their skills. Fresher jobs are also offered across multiple industries, allowing graduates to kickstart their careers in major cities and sectors.

Top Skills & Programs

Qollabb offers a variety of top skills and programs designed to help students across different fields. These include specialized programs for degrees like B.Tech, B.Com, and MBA, providing valuable internship opportunities in high-demand areas such as Python programming, market research, data analysis, business development, and client relationship management. These internships equip students with practical experience, enhancing their academic knowledge and preparing them for future roles in the workforce.

Structure of the Organization

Team:

- **Vipendra Singh** CEO & Founder
- **Ayush Singhal** Chief Technical Officer
- **Pooja Banerjee** HR & Operations
- **Amisha Sinha** Strategic Partnerships
- **Prerna Singh** Digital Marketing
- **Pavan Gupta** Software Engineer
- **Rohan Gupta** Software Engineer

Contact

Email: info@qollabb.com

Phone: +91 742-809-9546

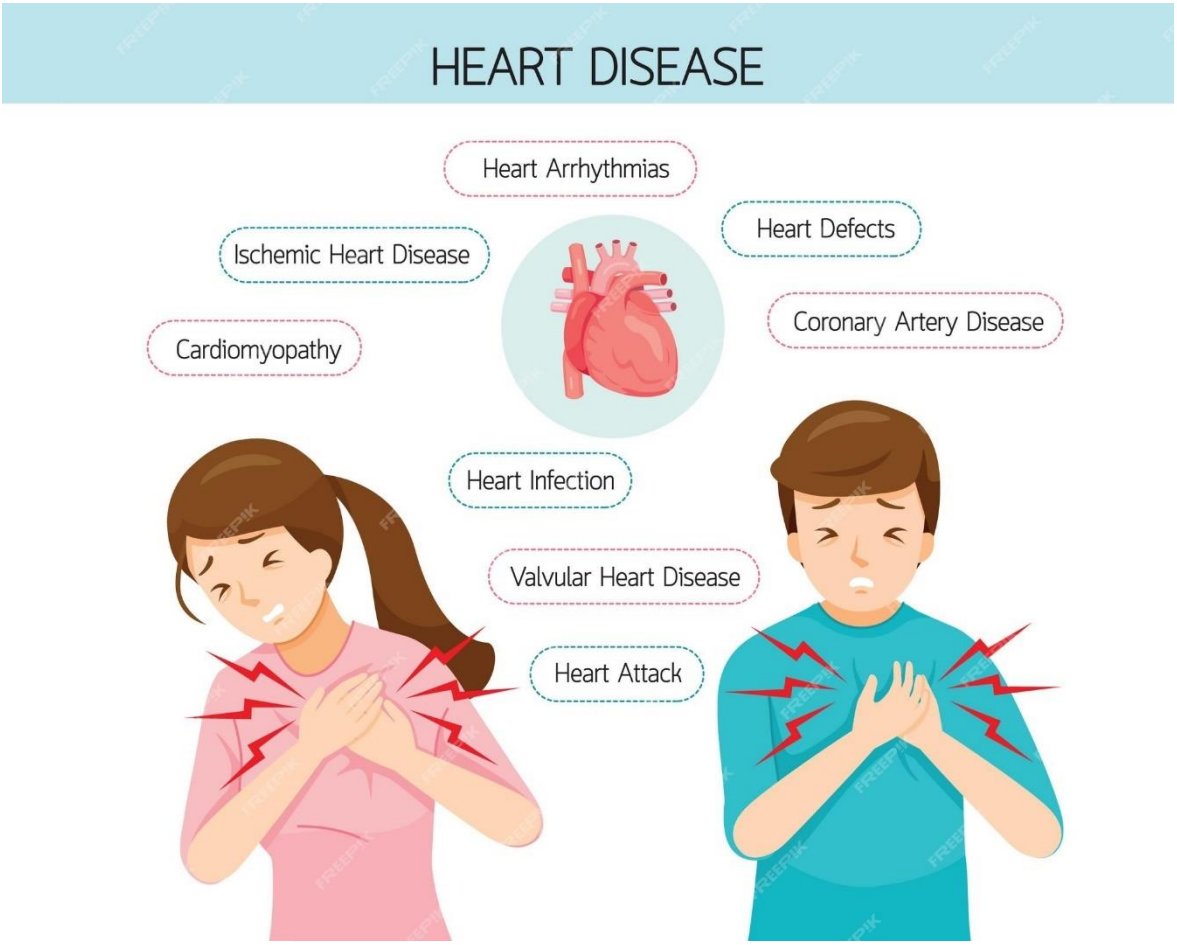
Addresses:

- **Corporate Office:** C-340, Tower-C, Bhutani Cyberpark, Sector-62, Noida, Uttar Pradesh- 201309, India
- **Registered Office:** 231/4 SF - II, Behind Silk Board Building, BTM Second Stage, Bangalore, Karnataka - 560068, India

2. General Introduction

Cardiovascular Disease: A Comprehensive Overview

Cardiovascular disease (CVD) refers to a group of disorders affecting the heart and blood vessels. These conditions include coronary artery disease, heart attacks, strokes, heart failure, and hypertension. CVD is one of the leading causes of death worldwide, often resulting from lifestyle factors like poor diet, lack of exercise, smoking, and excessive alcohol consumption. Other risk factors include high blood pressure, high cholesterol, diabetes, and a family history of heart disease.



Types of Cardiovascular Disease

CVD comprises various conditions, each affecting different parts of the cardiovascular system:

- **Coronary Artery Disease (CAD)**: The most common type of CVD, CAD occurs when the coronary arteries, which supply blood to the heart muscle, become narrowed or blocked by atherosclerosis (buildup of plaque). This can lead to chest pain (angina), heart attacks (myocardial infarctions), and potentially fatal complications.
- **Heart Failure**: This condition arises when the heart cannot pump blood effectively to meet the body's needs. It may be caused by CAD, hypertension, or other heart conditions. Heart failure can result in symptoms such as fatigue, shortness of breath, and fluid retention.
- **Arrhythmias**: These are irregularities in heart rhythm that may cause the heart to beat too fast (tachycardia), too slow (bradycardia), or erratically. Some arrhythmias can lead to sudden cardiac death.
- **Stroke**: A stroke occurs when the blood supply to part of the brain is interrupted or reduced, causing brain cells to die. This can result in disability or death. Strokes are classified into two types: ischemic (caused by a blood clot) and hemorrhagic (caused by a ruptured blood vessel).
- **Peripheral Artery Disease (PAD)**: This condition affects blood vessels outside the heart and brain, typically in the legs, and is caused by narrowed arteries. It increases the risk of heart attack and stroke.
- **Congenital Heart Disease**: These are structural defects in the heart present at birth. They can affect the heart walls, valves, and blood vessels.

Risk Factors for Cardiovascular Disease

Several risk factors contribute to the development of CVD. These factors can be categorized into modifiable and non-modifiable factors:

- **Non-modifiable risk factors**:
    - **Age**: The risk of CVD increases with age, particularly after 65.

    - **Gender**: Men are generally at higher risk of CVD than women, although the risk for women increases post-menopause.
    - **Family History**: A family history of heart disease increases the likelihood of developing CVD.
    - **Genetic Factors**: Some individuals may have a genetic predisposition to conditions like hypertension or high cholesterol.
- **Modifiable risk factors**:
    - **Smoking**: Tobacco use is one of the leading causes of cardiovascular disease, as it damages the heart and blood vessels.
    - **Unhealthy Diet**: Diets high in saturated fats, trans fats, cholesterol, salt, and sugar increase the risk of CVD.
    - **Physical Inactivity**: Lack of exercise leads to obesity, high blood pressure, and elevated cholesterol levels, contributing to CVD.
    - **Excessive Alcohol Consumption**: Drinking large amounts of alcohol can raise blood pressure and contribute to heart disease.
    - **Hypertension (High Blood Pressure)**: Chronic high blood pressure damages arteries and contributes to heart disease.
    - **High Cholesterol**: Elevated levels of LDL cholesterol contribute to plaque buildup in arteries, increasing the risk of heart attacks and strokes.
    - **Diabetes**: High blood sugar levels damage blood vessels and nerves controlling the heart, leading to a significantly higher risk of heart disease.
    - **Obesity**: Excess body weight is associated with increased blood pressure, high cholesterol, and insulin resistance.

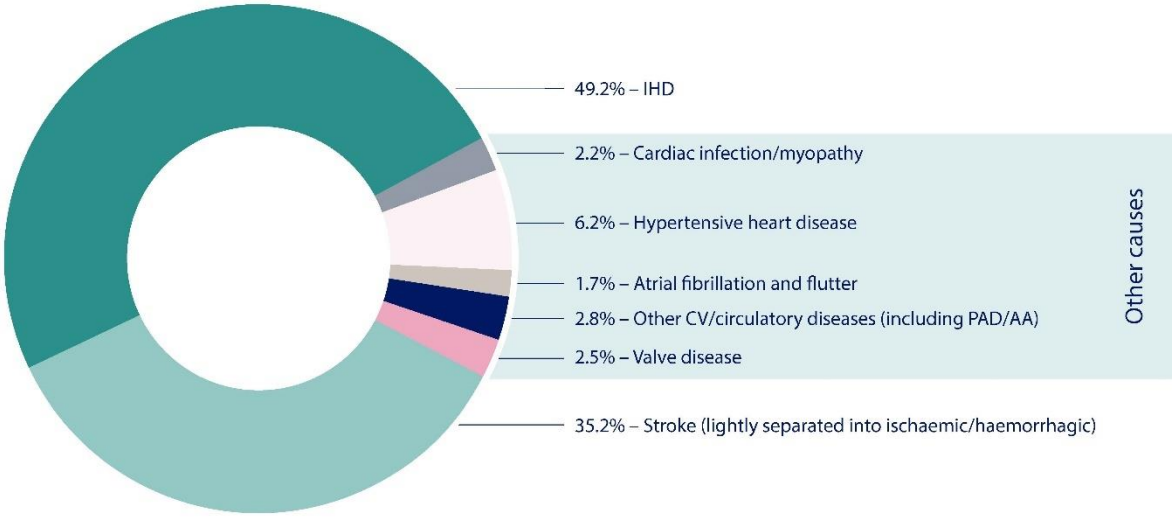
Global Burden of Cardiovascular Disease

CVD is the leading cause of mortality worldwide, responsible for more deaths than cancer or respiratory diseases. According to the Global Burden of Disease report, CVD accounted for approximately 18.6 million deaths in 2019. The burden of CVD has grown substantially over the last decade, with notable increases in years of life lost (YLL) in various countries. For instance, China experienced 81.6 million YLL, the United States 14.8 million YLL, and Indonesia 8.1 million YLL due to CVD.

In G20+ nations, CVD is the number one cause of death in 40% of countries and the second leading cause in 55%. Countries like Russia, the European Union, Indonesia, China, and India report some of the highest CVD mortality rates globally. After a decline in mortality from 1990 to 2010, many G20+ nations have observed rising CVD mortality rates again in recent years.

In 2019, cardiovascular diseases accounted for one-third of all global deaths, with 85% resulting from ischemic events like heart attacks and strokes. The remaining 15% of deaths were attributed to less common causes, such as hypertensive heart disease, atrial fibrillation, cardiomyopathy, and rheumatic heart disease. The chart below illustrates this breakdown.

Figure 1: Global Burden of Cardiovascular Diseases, by Type, 1990-2019 study



Infographic has been adapted from the global burden of cardiovascular diseases and risk factors 1990–2019 study[1] Global Burden of Cardiovascular Diseases and Risk Factors, 1990–2019. J Am Coll Cardiol. 2020 Dec.76(25)2982–3021. Doi: 10.1016/j.jacc.2020.11.010.

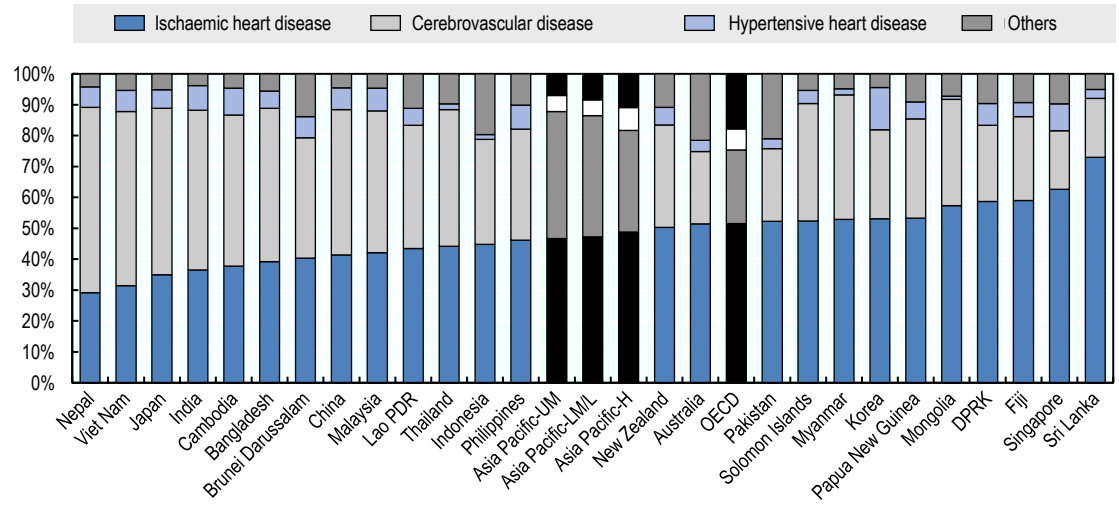
Burden of Cardiovascular Disease in India

Cardiovascular disease (CVD) has emerged as a significant public health challenge in India. Once considered a disease of affluence, it now affects populations across various socioeconomic strata, urban and rural alike. CVD is the leading cause of mortality in India, accounting for nearly 28% of all deaths. As the country continues to face rapid urbanization, lifestyle changes, and demographic shifts, the burden of CVD is expected to rise further, making it a critical concern for India's healthcare system.

Epidemiology of CVD in India

- **Prevalence**: The prevalence of CVD has seen a sharp increase over the past few decades. Estimates suggest that over 54.5 million people in India were living with CVD in 2016, and this number has been growing due to aging, urbanization, and lifestyle changes.
- **Mortality**: CVD has become the leading cause of death in India, responsible for approximately 2.5 million deaths annually. According to the Global Burden of Disease report, CVD mortality rates in India are higher than the global average, contributing to over 15% of the total global CVD deaths.
- **Years of Life Lost (YLL)**: In India, CVD contributes significantly to the overall disease burden, with a high number of years of life lost due to premature death. YLL from CVD in India is among the highest in the world, particularly affecting the working-age population (35-65 years).

Chart 1: Proportions of cardiovascular disease deaths, 2019



Source: Health at a Glance: Asia/Pacific 2022 - OECD 2022

Risk Factors Contributing to CVD in India

Several factors contribute to the growing burden of cardiovascular disease in India:

- **Hypertension**: High blood pressure is one of the most significant contributors to CVD in India. An estimated 30% of adults in urban areas and 25% in rural areas have hypertension, but many cases remain undiagnosed and untreated.
- **Diabetes**: India is often referred to as the "diabetes capital of the world," with over 77 million people living with diabetes. The co-occurrence of diabetes and heart disease significantly raises the risk of CVD, making diabetes a major factor in the increasing burden.
- **High Cholesterol**: Elevated cholesterol levels, particularly in urban populations, have contributed to the rise of ischemic heart disease in India. The consumption of unhealthy diets rich in fats, sugar, and processed foods exacerbates this issue.
- **Tobacco Use**: India has one of the highest rates of tobacco consumption globally, both in the form of smoking and smokeless tobacco. Tobacco use is a well-known risk factor for CVD, significantly increasing the likelihood of heart attacks and strokes.
- **Obesity**: Rising obesity rates, especially in urban areas, are contributing to increased risks of hypertension, diabetes, and CVD. A sedentary lifestyle, coupled with the increased intake of calorie-dense foods, has led to a surge in obesity rates across all age groups.
- **Physical Inactivity**: With urbanization and a shift towards more sedentary jobs and lifestyles, physical inactivity has become a significant risk factor for heart disease in India.
- **Air Pollution**: Air pollution, particularly in cities like Delhi, Mumbai, and Kolkata, is emerging as a growing concern for cardiovascular health. Studies have shown that exposure to high levels of particulate matter increases the risk of heart disease and stroke.

Economic Impact of CVD in India

The economic burden of cardiovascular disease in India is significant, both in terms of direct medical costs and indirect costs related to lost productivity. According to estimates by the World Economic Forum, India may lose up to $2.17 trillion in GDP between 2012 and 2030 due to CVD and other non-communicable diseases (NCDs).

- **Healthcare Costs**: CVD treatment in India is expensive, especially for the poorer sections of society. The cost of medication, hospitalization, surgeries like coronary artery bypass grafting (CABG) and angioplasty, and long-term care can be financially devastating for many families.
- **Productivity Loss**: CVD affects individuals in their prime working years, leading to premature death or disability. The loss of income due to illness or death further strains households and impacts the overall economy.

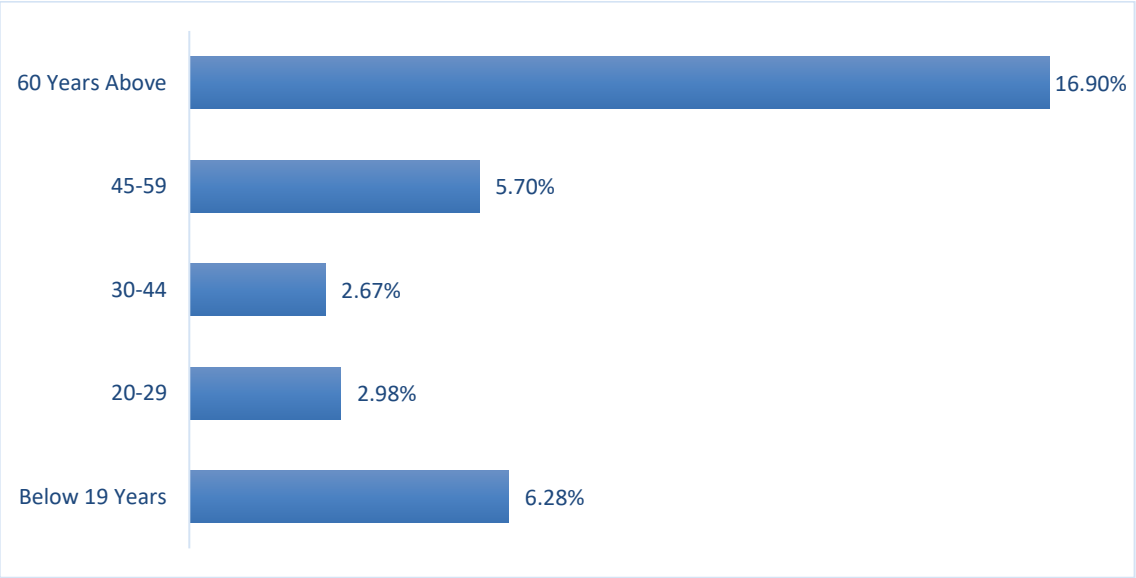
Importance of Early Detection of Cardiovascular Disease

Cardiovascular diseases (CVD), like atherosclerosis (artery hardening) and hypertension (high blood pressure), often progress gradually over time. Detecting them early allows for timely lifestyle changes and medical interventions, preventing the disease from worsening. Identifying risk factors such as high blood pressure, elevated cholesterol, or diabetes early on can significantly reduce the likelihood of heart attacks and strokes, which are often fatal or lead to long-term disability.

When CVD goes undiagnosed or untreated, it can result in debilitating conditions like heart failure, reduced physical activity, and a lower quality of life. Early detection enables timely treatment, potentially preventing or delaying such complications and allowing individuals to live healthier, more active lives. Moreover, managing CVD early through lifestyle changes, medications, and regular monitoring is far more cost-effective for both individuals and the healthcare system, reducing the economic burden on families and society.

The importance of early detection cannot be overstated. Identifying CVD risk factors and early signs enables healthcare professionals to implement preventive measures and treatments that can save lives, improve quality of life, and reduce healthcare costs. Traditional methods for diagnosing CVD, while effective, can be time-consuming and complex. The rise of machine learning (ML) technology offers a faster and more precise way to predict heart disease risk. By analyzing standard clinical information, electrocardiogram (ECG) signals, and X-ray images, ML models can help clinicians identify high-risk patients, leading to early interventions and better health outcomes.

Chart 2: Heart Issues Across India in 2020, By Age Group



Source: Statista

Introduction To Cardiovascular Disease Prediction

Recent advancements in technology, particularly in the field of machine learning (ML), have opened new avenues for predicting CVD more accurately and efficiently. By leveraging large datasets and analyzing various health parameters, such as age, cholesterol levels, blood pressure, and lifestyle factors, ML models can provide early warning signs of heart disease. These models not only enhance the ability to predict the risk of CVD but also enable healthcare professionals to tailor preventive interventions for high-risk individuals, significantly improving patient outcomes.

CVD prediction is becoming a vital tool in the battle against heart disease, as it shifts the focus from reactive treatment to proactive prevention. With continuous advancements in data science and artificial intelligence (AI), the integration of predictive analytics into healthcare systems promises to revolutionize the approach to managing cardiovascular health.

Problem Statement

According to the research published by Bisma Jan et al. (2024) and titled "Cardiovascular Diseases Among Indian Older Adults: A Comprehensive Review", in India, the risk of heart disease increases with age, from 22% in people aged 45-54 to 38% for those 70 and older. Women (32%) are more likely to have heart disease than men (26%), and people in urban areas (40%) are more affected than those in rural areas (25%). While there are ways to detect heart disease, they can be costly and not always accessible. Using machine learning, we can analyze health data to predict the risk of heart disease based on age and gender, helping with early detection in India.

Objective

To develop a machine learning model for predicting cardiovascular disease (CVD) using available clinical data from the Indian population, with a specific focus on evaluating the influence of age and gender as key risk factors. The project aims to offer preliminary insights into how these demographic variables contribute to CVD risk in India, within the constraints of the available data and time.

Scope

This project uses the "Cardiovascular Disease Dataset" from an Indian hospital, containing patient data on 12 features, including resting blood pressure, serum cholesterol, and electrocardiogram results. The scope includes data preprocessing, feature selection, model training, and evaluation using performance metrics such as accuracy, precision, recall, and F1-score. The goal is to create an accurate, deployable ML model capable of predicting heart disease risk.

Purpose of Study

The purpose of this study is to offer a more streamlined and efficient alternative to conventional CVD diagnostic methods through the application of machine learning. With the development of a user-friendly web application, this study aims to empower both healthcare providers and individuals to assess CVD risk in real time. This predictive tool can facilitate earlier diagnosis, encourage timely medical intervention, and ultimately help

reduce CVD-related mortality in regions like India, where the burden is disproportionately high.

Significance

Machine learning-based prediction models are particularly valuable in countries like India, where healthcare resources can be limited, and early intervention could significantly alleviate the burden of cardiovascular diseases. This project will help bridge the gap between clinical diagnosis and modern data science techniques, providing a scalable solution that can be applied across healthcare settings. The insights generated by this study could also inform public health initiatives aimed at reducing risk factors like hypertension and high cholesterol, further lowering the incidence of CVD.

Approach to Evaluate the Impact of Age and Gender on CVD Risk in the Indian Population

1. **Data Collection:** Use the dataset containing patient details such as age, gender, chest pain type, blood pressure, cholesterol levels, blood sugar, electrocardiogram results, and other clinical features from an Indian hospital.
2. **Data Preprocessing:** Clean the data, handle missing values, and appropriately encode the categorical variables (e.g., gender, chest pain type, electrocardiogram results).
3. **Exploratory Data Analysis (EDA):**
    - Visualize the relationship between age, gender, and CVD (target) using histograms and box plots.
    - Analyze the prevalence of CVD across different age groups and between genders.
4. **Feature Engineering:** Create new features based on age and gender to improve model prediction.
5. **Model Building:** Use a binary classification model like logistic regression to predict CVD presence based on age, gender, and other features.

6. **Assess Impact:** Evaluate the significance of age and gender in predicting CVD using model coefficients or feature importance.
7. **Interpret Results:** Analyze how age and gender contribute to CVD risk in the Indian population, supported by statistical tests and model predictions.

CHAPTER 2: LITERATURE REVIEW

Cardiovascular diseases (CVD) remain a leading cause of death globally. The integration of machine learning (ML) into CVD prediction has opened new avenues for early diagnosis and improved patient outcomes. This literature review provides a chronological overview of key research contributions to ML-based CVD prediction, highlighting advancements in the field, from traditional risk models to modern deep learning techniques.

1. Early Research on Predictors and Traditional Risk Models

Initial CVD prediction efforts focused on integrating traditional clinical risk factors with biomarkers. **Kaptoge et al. (2012)** provided a comprehensive analysis of the predictive value of inflammatory biomarkers, specifically C-reactive protein (CRP) and fibrinogen, for cardiovascular disease risk. Their study, involving data from 52 prospective studies with over 246,000 participants, demonstrated that adding CRP or fibrinogen to conventional risk factors such as age, cholesterol levels, and smoking status significantly improved cardiovascular risk prediction. The addition of these biomarkers enhanced the model's C-index by 0.0039 and 0.0027, respectively, indicating better risk discrimination. Furthermore, their inclusion yielded modest but statistically significant net reclassification improvements (1.52% for CRP and 0.83% for fibrinogen) in predicting the 10-year risk of cardiovascular events. Importantly, assessing CRP or fibrinogen in individuals at intermediate risk could prevent around 30 additional cardiovascular events per 100,000 adults over a decade, thus demonstrating the clinical value of integrating these biomarkers into risk assessment models. This study laid the groundwork for expanding cardiovascular risk models by incorporating a broader range of biochemical markers, enhancing early identification and prevention strategies.

Pocock et al. (2013) further made significant steps in predicting heart failure (HF) mortality by creating a risk score that utilized key predictors such as left-ventricular ejection fraction (EF) and comorbidities. Their analysis, based on data from 39,372 HF patients across 30 cohort studies, identified 13 crucial factors, including age, lower EF, serum creatinine, and diabetes, that influenced mortality. These factors were used to create an easy-to-use integer risk score, which showed a marked gradient in 3-year mortality rates, ranging from 10% to 70%. This score offers a practical tool for clinical application, providing an accessible means of identifying high-risk patients. These early works demonstrated the potential of using

clinical data to predict CVD outcomes more accurately, setting the stage for future ML applications.

2. The Emergence of Machine Learning in CVD Prediction

As machine learning techniques gained traction, researchers explored their capacity to improve upon traditional risk models. **Weng et al. (2017)** conducted a comparative analysis of machine learning (ML) algorithms in predicting cardiovascular disease (CVD) outcomes and found that ML significantly improved predictive accuracy over traditional models. In their study, four ML algorithms (random forest, logistic regression, gradient boosting, and neural networks) were tested on data from 378,256 patients, compared to an established algorithm (American College of Cardiology guidelines). The results showed that ML models, especially neural networks, outperformed traditional models, increasing predictive accuracy and correctly identifying more patients at risk for CVD, thereby enabling better-targeted interventions. Around the same time, **Shameer et al. (2018)** explored the fundamentals of machine learning (ML) algorithms and their potential to analyze complex, heterogeneous biomedical data. They highlighted the importance of selecting the appropriate ML algorithms, particularly when dealing with diverse data sources in cardiovascular medicine. The study emphasized that successful application of ML requires careful consideration of data depth, breadth, and heterogeneity, as well as the selection of algorithms and feature variables. Their work also discussed the promising use of AI for automated risk prediction, phenotype analysis, and augmenting healthcare decision-making.

3. Ensemble Models and Deep Learning Approaches

A significant shift occurred with the adoption of ensemble models and deep learning techniques for CVD prediction. **Dimopoulos et al. (2018)** compared machine learning (ML) methodologies with the established cardiovascular risk tool, HellenicSCORE, for predicting 10-year cardiovascular disease (CVD) incidence. Using data from the ATTICA prospective study, they trained ML classifiers such as k-NN, random forest, and decision tree, and compared their performance to HellenicSCORE. While the results varied depending on the classifier and dataset, the ML models showed comparable predictive capabilities to HellenicSCORE, with random forest providing the best results. This study highlights the potential of ML in CVD risk prediction. Similarly, **Habib et al. (2019)** utilized boosting-based ensemble machine learning techniques to predict coronary heart disease (CHD),

demonstrating the effectiveness of ensemble models in analyzing complex medical data. By incorporating medical factors such as blood pressure, hypertension, diabetes, and smoking habits, they trained several algorithms, including AdaBoost, Gradient Boosting Machine (GBM), Extreme Gradient Boosting (XGBoost), LightGBM (LGBM), and CatBoost. They evaluated the performance of these models and identified the one with the highest accuracy for CHD prediction, showcasing the potential of ML in improving early detection and risk assessment.

Mohan et al. (2019) introduced a hybrid model combining Random Forest and Linear Regression (HRFLM) to predict cardiovascular disease, achieving an impressive accuracy of 88.7%. This model leverages the strengths of both machine learning techniques to manage large datasets, identify significant features, and enhance predictive performance. By processing raw healthcare data, the HRFLM model demonstrates the potential to significantly improve heart disease prediction, offering early detection that could help reduce mortality rates. The study also suggests that future research could further optimize prediction accuracy by refining feature selection and applying the model to real-world datasets. These studies illustrate how ML, particularly ensemble models, can capture intricate relationships between variables, making them more effective than traditional models.

4. Practical Applications and Clinical Decision Support Tools

Advancements in ML models soon translated into practical tools for clinical use. **Terrada et al. (2020)** developed a Medical Decision Support System (MDSS) using supervised learning techniques such as Artificial Neural Networks (ANN) and K-nearest neighbors (KNN) to predict atherosclerosis. This system was one of the earliest applications of ML in real-time clinical decision-making, enabling earlier and more accurate interventions. Similarly, **Mathur et al. (2020)** explored AI applications in CVD treatment, including drug discovery and newer therapeutic strategies, broadening ML's impact beyond diagnosis to treatment optimization.

Lin et al. (2021) reviewed ML and deep learning's role in cardiovascular imaging, particularly for coronary artery disease risk stratification using non-invasive techniques. This expansion into imaging allowed ML to enhance diagnostic accuracy, particularly in identifying high-risk patients who could benefit from preventive measures. The growing use

of deep learning in cardiovascular imaging illustrates how these technologies can complement clinical expertise in complex cases.

5. Recent Advances in Prediction Models

Machine learning models have become increasingly refined, incorporating both ML and deep learning algorithms for higher predictive accuracy. In a study by **Xi Su et al. (2020)**, a random forest algorithm was applied to predict cardiovascular disease (CVD) using data from 498 subjects. The analysis revealed key predictive variables such as age, BMI, blood pressure, triglycerides, cholesterol, and blood glucose. Compared to a logistic regression model, the random forest model achieved an area under the curve (AUC) of 0.802. The logistic regression model included significant risk factors like age, BMI, and triglycerides, with an AUC of 0.843. This study underscores the effectiveness of random forests for early CVD prediction. **Alqahtani et al. (2022)** introduced an ensemble-based model that combined machine learning (ML) and deep learning (DL) techniques for cardiovascular disease (CVD) prediction. Using data from 70,000 patients, they employed six classification algorithms, with random forest (RF) identifying key features. The ensemble model achieved a high accuracy of 88.70% in predicting CVD. Their approach highlights the potential of artificial intelligence to improve early disease detection and intervention. Future work may involve using additional datasets and exploring deep learning and reinforcement learning for more precise predictions. These studies underscore how ensemble and deep learning approaches are being successfully implemented to predict cardiovascular outcomes with high precision.

6. Methodological Rigor and Validation Of Models

Damen et al. (2016) emphasized the importance of external validation and rigorous methodology in developing predictive models for cardiovascular disease (CVD) risk. They reviewed numerous models and found that many lacked adequate validation, making their practical application questionable. Machine learning can help address these challenges by providing more robust frameworks for validation and model comparison. This is essential for improving the reliability of CVD risk prediction tools and ensuring their utility in real-world clinical settings. Future research should focus on externally validating existing models rather than creating new ones. The need for innovative approaches is reiterated by **Alaa et al. (2019)**, demonstrated how leveraging comprehensive datasets through automated

machine learning (AutoML) can significantly improve cardiovascular disease (CVD) risk prediction. In their study involving 423,604 UK Biobank participants, the AutoPrognosis framework identified novel risk factors, such as walking pace and self-reported health, which enhanced prediction accuracy. The model outperformed traditional approaches, including the Framingham score and Cox proportional hazards models, highlighting the potential of AutoML to uncover complex risk interactions and boost prediction performance, particularly for underrepresented subgroups like individuals with diabetes. Their research demonstrated how automated machine learning frameworks could uncover novel risk predictors, thereby improving existing models' accuracy. However, the integration of non-traditional variables remains a potential area for future exploration.

7. Knowledge Gaps and Future Directions

Despite the advancements in machine learning applications for cardiovascular disease prediction, several knowledge gaps persist. The integration of novel biochemical markers, such as ceramides and phospholipids, into predictive models has been suggested by Hilvo et al. (2019), yet further research is needed to explore their full potential. Additionally, the application of machine learning in real-time risk assessments within medical information systems presents significant opportunities for enhancing patient management (Alotaibi, 2019).

Limitations Identified:

1. **Data Availability**: Limited access to large-scale, real-world datasets for training may hinder model generalization.
2. **Model Validation**: Insufficient validation across diverse populations raises concerns about clinical applicability.
3. **Complexity of Biological Interactions**: Machine learning models may not fully capture the complex biological mechanisms of CVD, especially in the presence of emerging biomarkers.

Future research should focus on the development of standardized validation frameworks for machine learning models in cardiovascular disease, ensuring their reliability and clinical applicability. Moreover, interdisciplinary collaborations between data scientists and

healthcare professionals could enhance the practical implementation of these predictive models in clinical settings.

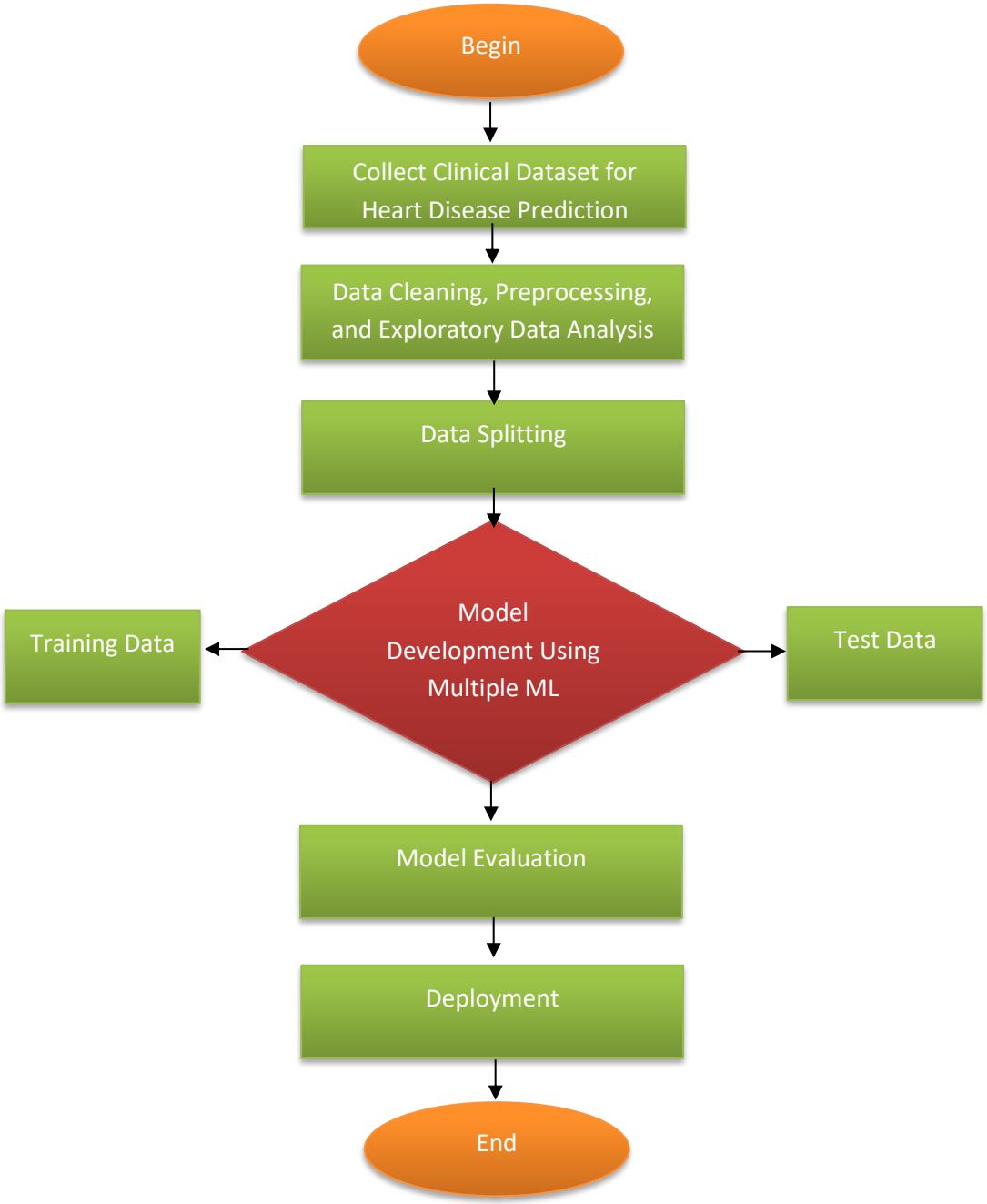
8. Conclusion

Machine learning has made significant strides in improving cardiovascular disease prediction by integrating traditional clinical data, biomarkers, and imaging with advanced algorithms. As research progresses, the focus must shift towards addressing current limitations, such as data availability, model validation, and understanding complex biological interactions. Standardized frameworks for model validation and interdisciplinary collaboration between data scientists and healthcare professionals will be essential for ensuring these predictive models are reliable and useful in clinical settings. Future research should prioritize the integration of novel biomarkers and deep learning techniques, ensuring that technological innovations continue to enhance patient outcomes.

CHAPTER 3: METHODOLOGY

1. System Architecture

The system architecture gives an overview of the workings of the system. The working of this system is shown below:



2. Dataset Details

Dataset Link: https://data.mendeley.com/datasets/dzz48mvjht/1

Published: 16 April 2021 | Version 1 | DOI: 10.17632/dzz48mvjht.1

Contributors: Bhanu Prakash Doppala, Debnath Bhattacharyya

Description: This heart disease dataset was obtained from a multispecialty hospital in India. It includes 12 key features collected from 1,000 subjects, focusing on 14 commonly observed factors in heart disease cases. This dataset is valuable for early-stage heart disease detection and for building predictive machine learning models to advance research in this area.

Dataset Attributes:

Table 1: Heart Disease Dataset from a Multispecialty Hospital in India, Attributes

Sr. No.	Attribute	Assigned Code	Unit	Type of the Data
1	Patient Identification Number	patientid	Number	Numeric
2	Age	age	In Years	Numeric
3	Gender	gender	1,0 (0= female, 1 = male)	Binary
4	Chest pain type	chestpain	0,1,2,3 (Value 0: typical angina, Value 1: atypical angina, Value 2: non-anginal pain, Value 3: asymptomatic)	Nominal
5	Resting blood pressure	restingBP	94-200 (in mm HG)	Numeric
6	Serum cholesterol	serumcholestrol	126-564 (in mg/dl)	Numeric
7	Fasting blood sugar	fastingbloodsugar	0,1 > 120 mg/dl (0 = false, 1 = true)	Binary

8	Resting electrocardiogram results	restingrelectro	0,1,2 (Value 0: normal, Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), Value 2: showing probable or definite left ventricular hypertrophy by Estes'criteria)	Nominal
9	Maximum heart rate achieved	maxheartrate	71-202	Numeric
10	Exercise induced angina	exerciseangia	0,1 (0 = no, 1 = yes)	Binary
11	Oldpeak =ST	oldpeak	0-6.2	Numeric
12	Slope of the peak exercise ST segment	slope	1,2,3 (1-upsloping, 2-flat, 3-downsloping)	Nominal
13	Number of major vessels	noofmajorvessels	0,1,2,3	Numeric
14	Classification	target	0,1 (0= Absence of Heart Disease, 1= Presence of Heart Disease)	Binary

3. Data Preprocessing and Exploratory Data Analysis

3.1. Data Preprocessing

Data preprocessing is a crucial step in machine learning where raw data is transformed into a clean and usable format. This process is essential to ensure that the machine learning model can properly interpret and learn from the data. It involves various techniques to handle issues like missing values, incorrect data types, or inconsistent entries. Let's break this down into the following steps:

a) Data Cleaning

- **Handling Missing Values:** Often datasets contain missing or incomplete values. These can be addressed in various ways, such as:
    - Removing rows or columns with missing data.
    - Replacing missing values with the mean, median, or mode of the column (imputation).
    - Using algorithms that can handle missing data natively.
- **Dealing with Outliers:** Outliers are data points that are significantly different from other points. They can negatively impact the performance of a machine learning model. Some ways to handle outliers include:
    - Removing outliers if they are due to errors.
    - Using robust algorithms that are less affected by outliers.
    - Applying transformations like logarithmic or scaling to reduce the effect of outliers.
- **Correcting Data Types:** Sometimes, numerical values might be stored as strings or dates need to be converted into usable formats. Converting these data types ensures accurate calculations.

b) Feature Encoding

- **Label Encoding or One-Hot Encoding:** Machine learning algorithms work with numerical data. If categorical data (such as 'male' and 'female' or 'yes' and 'no') is present, it needs to be converted into numerical form. Common techniques include:
    - **Label Encoding:** Assigns a unique integer to each category.
    - **One-Hot Encoding:** Creates binary columns for each category, indicating its presence with a 1 or 0.

c) Feature Scaling

- Different features in the dataset may have different units (e.g., age may range from 0 to 100, while income may range from 10,000 to 1,000,000). Scaling brings all features to a common range, making the training process faster and more stable. Common techniques include:
    - **Standardization (Z-score normalization):** Rescaling the data to have a mean of 0 and a standard deviation of 1.
    - **Min-Max Scaling:** Rescaling data to a fixed range, usually between 0 and 1.

d) Splitting the Dataset

- **Train-Test Split:** The dataset is typically divided into two parts: a training set (used to train the model) and a test set (used to evaluate the model). A common ratio is 80% training and 20% testing.
- **Cross-Validation:** In some cases, k-fold cross-validation is used where the data is split into 'k' parts, and the model is trained on 'k-1' parts and tested on the remaining part. This helps in ensuring the model generalizes well to unseen data.

3.2. Exploratory Data Analysis (EDA)

EDA is the process of analyzing the dataset in-depth to uncover patterns, trends, and relationships between the variables. It helps in understanding the data before feeding it into a model. EDA involves both graphical and statistical techniques.

a) Summary Statistics

- **Mean, Median, Mode**: These are measures of central tendency that give an idea of where the center of the data lies.
- **Standard Deviation and variance**: These metrics show the spread or dispersion of the data.
- **Percentiles, Quartiles:** These help in understanding the distribution of the data (e.g., the 25th percentile, 50th percentile (median), and 75th percentile).

b) Understanding Distributions

- **Histograms:** These graphs show the frequency of each value or range of values in the dataset, helping us understand the distribution of numerical features (normal distribution, skewness, etc.).
- **Box Plots:** Box plots help identify the spread of the data and spot potential outliers by showing the minimum, maximum, median, and interquartile range.
- **Density Plots:** These smooth plots show the probability density of a continuous variable.

c) Relationships Between Variables

- **Scatter Plots:** Scatter plots show the relationship between two variables. For example, a scatter plot between age and cholesterol might reveal if older people tend to have higher cholesterol levels.
- **Correlation Matrix:** The correlation matrix shows the linear relationship between multiple variables. A high correlation (close to 1 or -1) between two features indicates that they might provide similar information, which may be redundant for the model.

- **Heatmaps:** Heatmaps visualize the correlation matrix, using colors to represent the strength of the relationships between features.

d) Detecting Patterns and Trends

- **Pair Plots:** These are combinations of scatter plots for multiple feature pairs, helping visualize relationships between several features at once.
- **Group Comparisons:** EDA also involves comparing different groups within the data. For instance, comparing the average age, blood pressure, or cholesterol levels between those with and without heart disease.

e) Feature Selection

- **Identifying Important Features:** By analyzing the data, we can identify which features are most important for predicting the target variable. For example, in heart disease prediction, features like cholesterol levels, age, and blood pressure might have a strong relationship with heart disease.
- **Handling Multicollinearity:** If two features are highly correlated, it can lead to multicollinearity issues. This means the model might give undue importance to one of the correlated features. We can use techniques like **Principal Component Analysis (PCA)** or simply drop one of the correlated features to handle this.

Importance in Cardiovascular Disease Prediction

- In the context of predicting cardiovascular disease in the Indian population, **data preprocessing** ensures that the dataset, including factors like age, gender, blood pressure, cholesterol levels, and other relevant features, is clean and accurate. Proper handling of this data is essential, as errors or inconsistencies could lead to misleading predictions.
- **Exploratory Data Analysis (EDA)** plays a crucial role in uncovering trends and relationships, such as the impact of age and gender on the likelihood of developing cardiovascular disease. Visualizing these factors can provide valuable insights into their correlation with heart disease, helping to build a more accurate and reliable machine learning model for predicting cardiovascular disease based on these key features.

4. Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn patterns and make decisions from data without explicit programming. Instead of being manually coded for every possible decision or task, machine learning algorithms "train" on historical data to identify patterns and relationships within that data. Once trained, these models can make predictions or decisions based on new, unseen data. Machine learning has applications in numerous fields, including healthcare, where it's commonly used for tasks such as medical diagnosis, image recognition, and disease prediction.

Machine learning is generally categorized into three types:

1. **Supervised Learning**: The algorithm learns from labeled training data to make predictions or classifications.
2. **Unsupervised Learning**: The algorithm identifies patterns in data that is not labeled, often used for clustering or association tasks.
3. **Reinforcement Learning**: The model learns to make decisions by receiving rewards or penalties based on its actions.

Supervised Machine Learning

Supervised machine learning is a type of machine learning where the model is trained using labeled data. In this scenario, each input comes with an associated correct output (label), and the model's task is to learn the mapping from inputs to outputs. Once trained, the model can predict outputs for new, unseen data.

a) Classification Algorithm

Classification is a type of supervised learning where the output is a discrete category or label, such as predicting ‘0’ i.e. absence of heart disease, or ‘1’ i.e. presence of heart disease for a patient. The purpose of classification algorithms is to group data points into predefined categories based on the input features. In the context of heart disease prediction, classification algorithms assess a patient's risk by analyzing various input features like age, gender, blood pressure, cholesterol levels, and other health indicators, to determine whether they are likely to develop heart disease.

b) Logistic Regression

Logistic Regression is one of the simplest and most commonly used classification algorithms for binary classification problems. Logistic regression uses a **logistic function (sigmoid function)** to model the probability of a data point belonging to a certain class. The sigmoid function maps any real-valued number into a value between 0 and 1, representing the probability that an instance belongs to the positive class. It is particularly useful when the relationship between the features and the outcome is **linear**.

Key Concepts of Logistic Regression

1. **Binary Outcome (Classification)**: Logistic regression is used when the target variable is binary. The goal is to predict which of two classes an observation belongs to.

2. **Sigmoid Function**: The core idea of logistic regression is the use of the **sigmoid function**, also known as the **logistic function**, which maps any real-valued number into a value between 0 and 1. This is important because the output needs to be interpreted as a probability, which is bounded between 0 and 1.

   The **sigmoid function** is defined as:

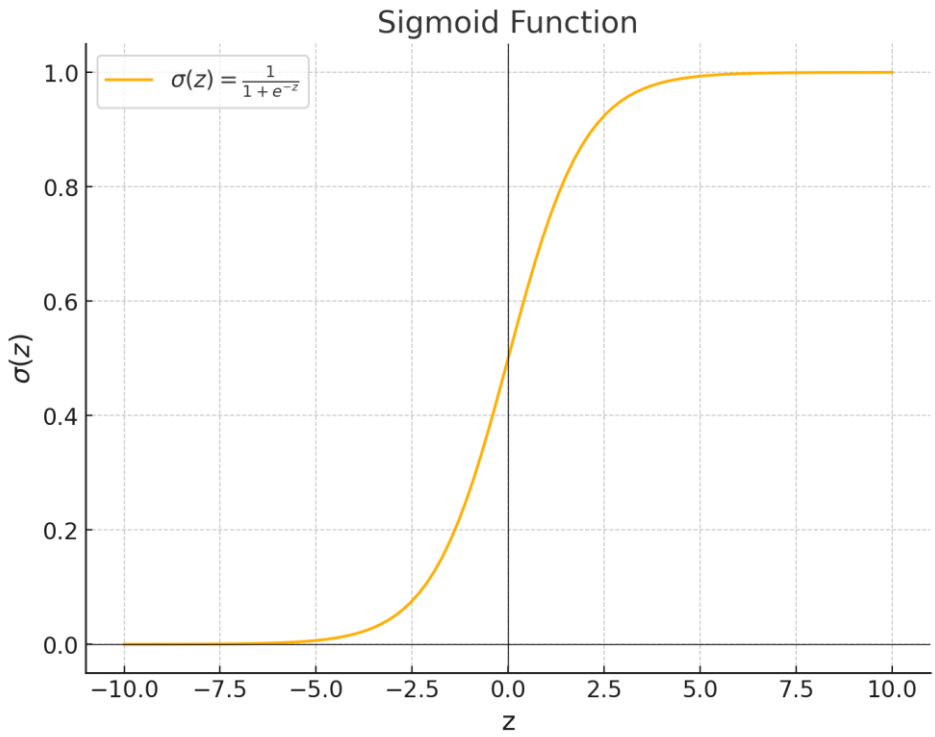
   $\sigma(z) = 1 \div (1 + e^{-z})$

   Where:

   - z is the linear combination of input features: $z = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + \cdots + \beta_n \times x_n$
   - e is the base of the natural logarithm.

   The output of the sigmoid function represents the probability that the given input point belongs to the positive class (e.g., class 1).

Chart 3: Sigmoid Function



3. **The Logistic Regression Equation:**

   $y = 1 \div (1 + e^{-(\beta_0+\beta_1\times x_1+\beta_2\times x_2+\cdots+\beta_n\times x_n)})$

   Where:

   - y is the predicted probability that the observation belongs to class 1.
   - β_0 is the intercept.
   - $\beta_1, \beta_2,\ldots, \beta_n$ are the weights or coefficients associated with each feature $x_1, x_2,\ldots,x_n$.

4. **Prediction:** Once we have the probability, we can classify an observation. If the probability y is greater than 0.5, we classify it as belonging to class 1 (positive class); otherwise, we classify it as belonging to class 0 (negative class).
5. **Model Training:**
   - Logistic regression is trained by adjusting the weights or coefficients and intercept to minimize the error between the predicted probabilities and the actual class labels (0 or 1).
   - This is done using an optimization technique such as Gradient Descent.

- The objective is to minimize the log loss (also called cross-entropy loss), which is a measure of the difference between the true labels and the predicted probabilities.

  The log loss function is:

  Log Loss = $-\frac{1}{m}\sum_{i=1}^{m}[y_i \times log(\hat{y}_i) + (1 - y_i) \times log(1 - \hat{y}_i)]$

  Where:

  - m is the number of data points.
  - y_i is the true label.
  - $\hat{y}_i$ is the predicted probability for the i-th observation.

Advantages of Logistic Regression in Predicting Cardiovascular Disease

1. **Interpretable Results**: Logistic regression provides clear and interpretable outputs. For heart disease prediction, it allows healthcare professionals to understand how different features (age, cholesterol levels, blood pressure, etc.) influence the likelihood of disease, making it easier to explain results to patients.
2. **Probabilistic Predictions**: Logistic regression outputs probabilities, which is valuable in heart disease prediction. It helps estimate the risk level (e.g., a patient has a 70% chance of having heart disease), allowing doctors to make informed decisions about further tests or treatments.
3. **Works Well with Binary Classification**: Since heart disease prediction is often a binary classification problem (disease or no disease), logistic regression is a natural fit, predicting whether a patient is likely to have heart disease or not.
4. **Efficient for Small and Medium Datasets**: Logistic regression is computationally efficient, especially with smaller datasets like patient records, making it a good choice for heart disease prediction without requiring high processing power.
5. **Handles Missing Data**: Logistic regression can handle missing or incomplete medical records by using techniques like data imputation, making it useful in real-world scenarios where not all patient data might be available.

Disadvantages of Logistic Regression in Predicting Cardiovascular Disease

1. **Assumes Linear Relationships**: Logistic regression assumes a linear relationship between the input features (like cholesterol, age, etc.) and the log-odds of the disease. However, heart disease may have more complex, non-linear patterns that logistic regression cannot easily capture without feature engineering.

2. **Sensitive to Outliers**: In medical data, outliers (extremely high or low values) can distort logistic regression models. For instance, an unusual blood pressure value could significantly impact the prediction, leading to unreliable results.

3. **Limited to Binary Outcomes**: Logistic regression is typically used for binary classification (heart disease vs. no heart disease). While it can be extended for multiclass classification, it is less flexible if predictions need to account for varying stages of heart disease.

4. **Requires Relevant Features**: Logistic regression works best when relevant, independent features are included. In heart disease prediction, this means careful feature selection and engineering are needed, as irrelevant features could lead to poor predictions.

5. **Struggles with Imbalanced Data**: If the dataset has more patients without heart disease than with it (class imbalance), logistic regression might predict the majority class more often, underperforming in identifying high-risk patients.

2. K-NEAREST NEIGHBORS (KNN)

K-Nearest Neighbors (KNN) is a simple, non-parametric model because it doesn't assume any underlying relationship between features. It simply stores all the data points and, when predicting, compares the new point with the closest k data points in the training set to determine its class or value.

Key Concepts of K-Nearest Neighbors:

- **Instance-Based Learning:** KNN is a lazy learning algorithm, meaning it does not create a generalized model during training. Instead, it memorizes the training data and makes predictions based on the data points closest to the input.

- **Distance Metrics:** KNN uses a distance metric (commonly Euclidean distance) to find the nearest data points to a given instance.

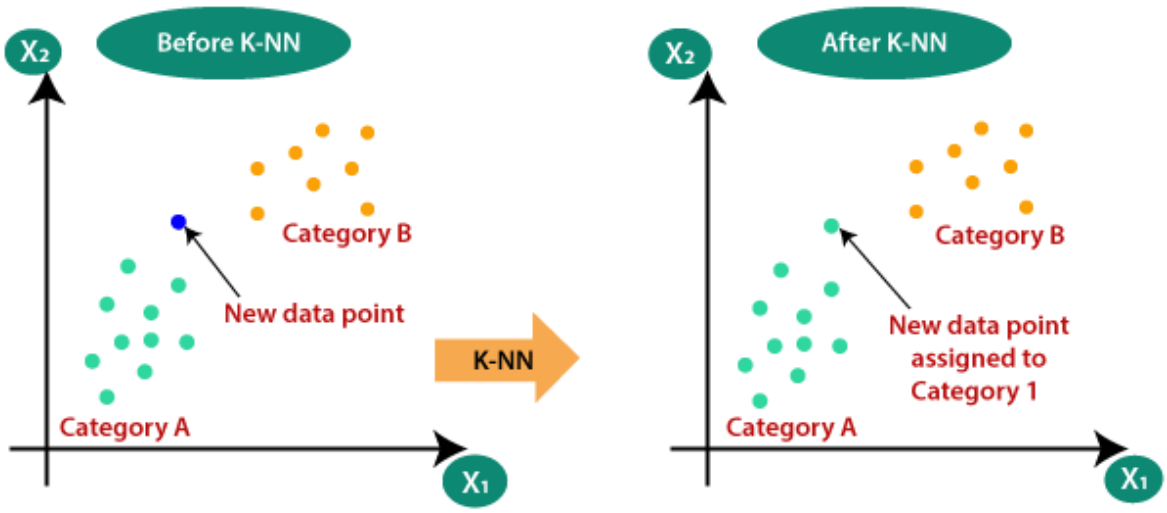
  The distance between two points:

  $X_i = x_1, x_2, \ldots, x_n$ and $X_j = x'_1, x'_2, \ldots, x'_n$ is computed as:

  $$d(X_i, X_j) = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \cdots + (x_n - x'_n)^2}$$

- **Voting Mechanism:** Once the nearest neighbors are identified, the algorithm predicts the class by taking a majority vote from the labels of the k-closest data points.

Figure 2: Classification of a New Data Point Using the K-Nearest Neighbors (K-NN) Algorithm



Source: javatpoint

c) KNN Algorithm:

1. Select the number of neighbors k.
2. Calculate the distance between the test instance and all training instances.
3. Select the k nearest neighbors based on the distance.
4. Assign the class label based on the majority class among the k nearest neighbors.

Prediction:

If more of the k neighbors belong to class 1 (presence of heart disease), the new instance is classified as class 1, otherwise class 0.

Model Training:

KNN requires no explicit training phase because it does not learn a model; rather, it uses the entire training dataset during prediction. The complexity comes in during prediction, as the algorithm computes the distance between the input and every training point.

Advantages of KNN Algorithm in Predicting Cardiovascular Disease:

1. **Simplicity and Interpretability**: KNN is easy to implement and understand, making it a useful tool in predicting cardiovascular disease based on age, gender, and other features. Its straightforward approach helps in easily explaining predictions, which is important for healthcare practitioners.
2. **No Assumptions about Data Distribution**: KNN does not require assumptions about the distribution of the data, which is particularly useful in heart disease prediction, as the relationships between age, gender, and disease risk may not follow a simple distribution.
3. **Flexibility for Complex Boundaries**: KNN can handle complex decision boundaries, especially when the relationship between age, gender, and other features is non-linear. It can identify patterns that other algorithms may miss.

4. **Works Well with Small Datasets**: In the context of Indian population-specific heart disease datasets, where sample sizes may not always be large, KNN can perform well without requiring a large amount of training data.

5. **No Training Phase**: Since KNN is a lazy learner, it doesn't require a lengthy training phase, making it an efficient choice for real-time predictions based on new patient data.

Disadvantages of KNN Algorithm in Predicting Cardiovascular Disease:

1. **Computationally Expensive with Large Datasets**: As KNN requires calculating distances between all data points during prediction, it can be slow when dealing with large datasets containing many features, such as multiple variables related to age, gender, cholesterol levels, etc.

2. **Memory Intensive**: Since KNN stores the entire dataset for classification, it requires a lot of memory, which can be a problem when dealing with large-scale datasets that include diverse patient information for cardiovascular disease prediction.

3. **Sensitivity to Noise and Outliers**: The algorithm can be heavily influenced by noisy data or outliers, such as incorrect age or gender information, which can distort the classification results and impact prediction accuracy, especially when working with imperfect medical data.

4. **Selection of K**: The performance of KNN depends on the value of 'k', which can significantly impact classification results. An inappropriate value of 'k' can lead to either overfitting (too small k) or underfitting (too large k), affecting heart disease prediction accuracy.

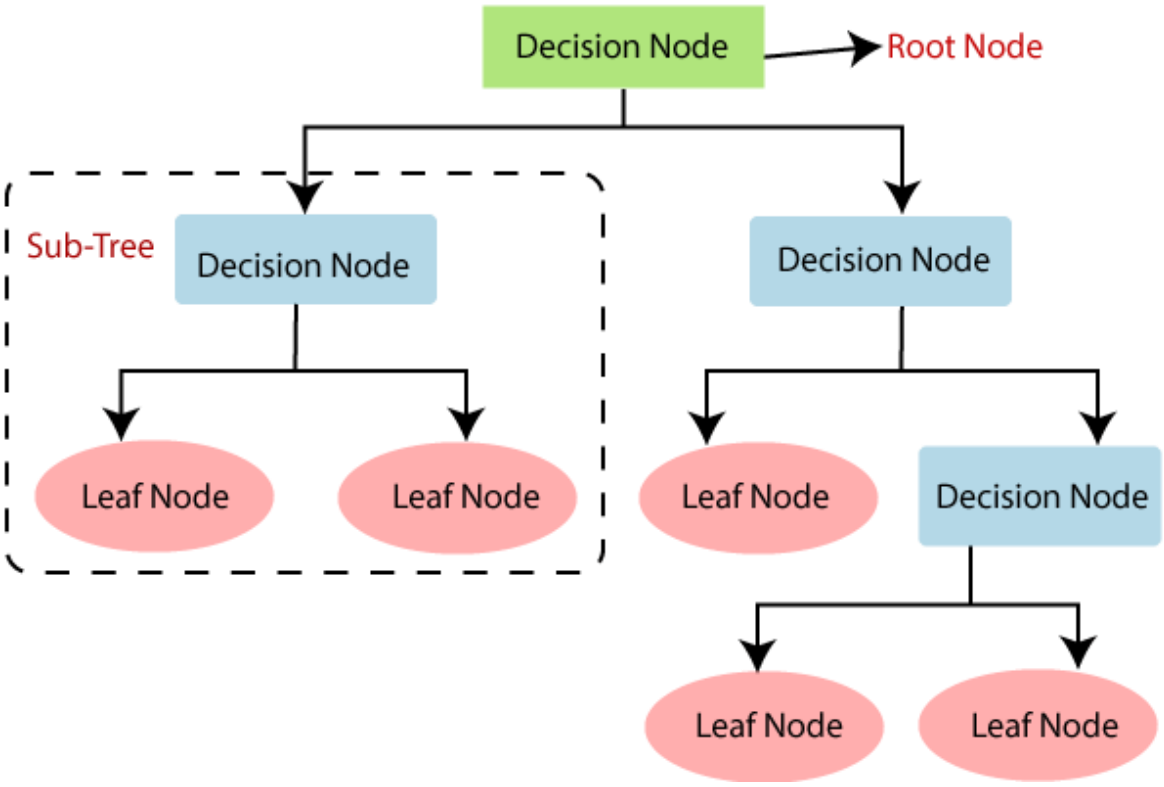
5. **Dimensionality Curse**: As the dataset grows with more features (such as adding additional medical tests or health-related parameters), KNN may struggle with high-dimensional data, as the distance metric becomes less effective in distinguishing meaningful neighbors, which can degrade performance.

6. **Feature Scaling**: KNN is sensitive to the scale of features. In the context of heart disease prediction, features like age and cholesterol levels are on different scales, requiring proper normalization or standardization to ensure accurate predictions.

d) Decision Tree

A decision tree is a popular and powerful supervised learning algorithm used for both classification and regression tasks. The model works by splitting the data into smaller subsets based on feature values, and this process is repeated recursively to build a tree-like structure. Each internal node of the tree represents a decision based on a specific feature, while each leaf node represents an outcome or prediction (e.g., a class label for classification or a continuous value for regression).

Figure 3: Structure of a Decision Tree



Source: javatpoint

1. **Root Node**: The root node is the topmost node of the tree and represents the entire dataset. The tree starts here by selecting the feature that provides the best split (using a metric like **Gini impurity**, **entropy**, or **variance**).

2. **Internal Nodes**: These nodes represent decisions or tests on features, and each node can have two or more child nodes. The decision is based on a feature value (e.g., "Is age > 50?").

3. **Leaf Nodes**: Leaf nodes represent the final outcome of the decision tree, such as a class label in classification or a predicted numerical value in regression.

4. **Branches**: Branches connect the internal nodes to the leaf nodes and represent the outcome of the decision or condition applied at the parent node.

How a Decision Tree Works

The decision tree algorithm recursively partitions the dataset into smaller subsets based on feature values, selecting the best feature at each step according to a specific criterion. The process can be described as follows:

1. **Feature Selection:** At each node of the tree, the algorithm selects the feature that best splits the dataset into meaningful subsets. For heart disease prediction, these features could include age, gender, cholesterol level, or blood pressure. The decision on which feature to select is based on a splitting criterion like Gini impurity, entropy (information gain), or mean squared error (MSE) for regression tasks.

2. **Splitting:** The dataset is divided into two or more subsets based on the selected feature. For example, a question like "Is age > 50?" can be used to separate data points into those with age greater than 50 and those less than or equal to 50. For categorical features, such as gender, the split might ask, "Is the patient male or female?" to separate the data accordingly.

3. **Recursive Partitioning:** The algorithm recursively splits the dataset at each node, aiming to create homogenous subsets where all data points have similar characteristics (such as all belonging to the same CVD risk category). The goal is to create pure nodes, where the data points in each node are highly consistent in terms of the target variable (CVD status).

4. **Stopping Criteria:** The recursive partitioning process stops when:

    - All data points at a node belong to the same class (pure node, e.g., no further distinction between healthy and at-risk patients).

    - A predefined maximum tree depth is reached to prevent overfitting.

    - The number of data points in a node becomes too small to make a valid split.

- No further information gain can be achieved, meaning that further splits don't improve predictions.

5. **Prediction:** Once the decision tree is built, new data points (e.g., a new patient's information like age, gender, cholesterol levels) are classified by traversing the tree from the root to a leaf node. The class label at the leaf node represents the prediction of whether the patient is at risk for cardiovascular disease or not.

Splitting Criteria

The decision tree algorithm can use several criteria to decide how to split the data at each node:

1. **Gini Impurity (for classification):** Gini impurity measures the likelihood that a randomly chosen data point will be incorrectly classified. The goal is to choose the feature that minimizes the Gini impurity.

   Formula:

   Gini = $1 - \sum_{i=1}^{c} p_i^2$

   Where, p_i is the proportion of instances of class i in the node, and C is the total number of classes.

   For heart disease prediction, the classes could be 1 as "At Risk" or 0 as "Not At Risk".

2. **Entropy and Information Gain (for classification):** Entropy measures the disorder or uncertainty in the dataset, while information gain quantifies how much information a feature provides in reducing that uncertainty.

   Formula:

   Entropy = $-\sum_{i=1}^{c} p_i \, log_2(p_i)$

   Where, p_i is the proportion of instances of class i in the node.

   Information gain is the reduction in entropy caused by splitting on a feature:

   Information Gain = Entropy(parent) - $\sum_{k=1}^{K} \frac{|D_k|}{|D|}$ Entropy(D_k)

   Where D_k is the subset of data after the split.

By using these criteria, decision trees can effectively model the impact of age, gender, and other factors on cardiovascular disease prediction, ensuring accurate and interpretable classifications for patients in the Indian population.

Advantages of Decision Trees

1. **Interpretability**: Decision trees are easy to interpret and understand. Each decision is based on simple conditions, making them highly transparent.
2. **No need for data normalization**: Decision trees don't require scaling or normalization of the data because they are based on splitting the data at specific feature values.
3. **Handles both categorical and numerical data**: Decision trees can work with both types of data, making them versatile.
4. **Non-parametric**: Decision trees are non-parametric, meaning they don’t assume any specific distribution of the data.

Disadvantages of Decision Trees

1. **Overfitting**: Decision trees can easily overfit the training data, especially if the tree is deep (many splits). This can lead to poor generalization of unseen data.
2. **Instability**: Small changes in the data can result in significantly different trees due to their greedy nature (choosing the best split at each node without considering global optimization).
3. **Bias towards features with more levels**: Decision trees can be biased toward features that have many unique values (high cardinality).

e) Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy and robustness. It's a supervised learning algorithm, commonly used for classification and regression tasks. In the context of cardiovascular disease (CVD) prediction, Random Forest can analyze multiple patient features (age, gender, blood pressure, cholesterol levels, etc.) to predict the likelihood of heart disease.

How Random Forest Works:

1. **Building Multiple Decision Trees**: Random Forest creates multiple decision trees from different subsets of the training data. These subsets are generated using **bootstrapping**, a technique where random samples are drawn with replacement.

2. **Random Feature Selection**: In addition to creating random data subsets, Random Forest selects a random subset of features at each split when building the decision trees. For instance, when splitting on patient data like age, gender, or cholesterol, it will randomly choose a few features to evaluate the best split.

3. **Voting Mechanism for Classification**: Once all decision trees are built, the Random Forest makes a final prediction by taking a **majority vote**. Each tree in the forest votes for a class (e.g., "Presence of Heart Disease" or "Absence of Heart Disease"), and the class with the most votes becomes the final prediction for that patient's CVD status.

4. **Final Prediction**: The final output is the **average** of all predictions for regression or the **majority class** for classification tasks. This voting mechanism reduces the likelihood of overfitting and ensures that predictions are more stable and accurate.

Advantages of Random Forest:

1. **Improved Accuracy**: Since random forest uses multiple decision trees, it significantly improves prediction accuracy compared to a single decision tree. For heart disease prediction, it ensures that a wide range of features (age, gender, blood pressure, etc.) are considered, which reduces the chance of missing critical risk factors.

2. **Robustness**: Random Forest is less sensitive to overfitting because it averages the results of multiple trees, each built on a different subset of data. This is particularly useful in healthcare data where noise or outliers (e.g., unusual patient profiles) may lead to overfitting in traditional models.

3. **Handles High-Dimensional Data**: In the case of CVD prediction, many features such as lifestyle, genetic markers, age, and gender may influence outcomes. Random Forest can efficiently handle large datasets with many features, identifying the most important ones for prediction.

4. **Feature Importance**: Random Forest ranks the importance of each feature in determining the outcome. This is valuable in understanding which features (age, gender, cholesterol, etc.) have the greatest impact on heart disease risk in the Indian population. This transparency is crucial for interpreting results in medical applications.

5. **Resilience to Missing Data**: Random Forest can handle missing values effectively by utilizing different subsets of data, ensuring that predictions are robust even if some patient data is missing or incomplete.

Disadvantages of Random Forest:

1. **Complexity**: Unlike a single decision tree, which is easy to interpret and visualize, random forests consist of hundreds or thousands of trees, making the model more complex and harder to interpret. For CVD prediction, it might be more difficult for clinicians to extract insights directly from the model.

2. **Slower Prediction Time**: Random Forests are computationally intensive, especially when there are many trees and many features. In real-time applications, such as predicting CVD risks for new patients, this could cause slower response times than simpler models like logistic regression.

3. **Bias Towards Majority Classes**: Random Forest may be biased toward majority classes if there is class imbalance in the dataset. For example, if the number of "Absence of Heart Disease" patients is much higher than the "Presence of Heart

Disease" patients, Random Forest might favour the majority class. Techniques like class weighting or undersampling may be required to address this imbalance.

4. **Difficulty in Interpretability**: Although random forest can identify important features, it is not as transparent as simpler models like decision trees. Clinicians may prefer simpler models for clinical decision-making, as they offer more straightforward explanations of how predictions are made.

Why Random Forest Is Suitable for This Project:

Random Forest is well-suited for predicting cardiovascular disease in the Indian population, especially when dealing with multiple factors like age, gender, and other health indicators. It provides high accuracy, handles both categorical (gender) and continuous features (age), and can deal with missing data effectively. However, its complexity and slower prediction time might pose challenges if interpretability or real-time application is essential.

In this project, Random Forest would help:

- Understand how **age and gender** impact CVD risk.
- Identify the most important features contributing to heart disease prediction.
- Improve prediction accuracy compared to simpler models while still allowing for flexibility in feature handling.

f) Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful classification algorithm that constructs a hyperplane (or decision boundary) or set of hyperplanes in a high-dimensional space that best separates data points of different classes in a high-dimensional space. This algorithm is highly effective in high-dimensional spaces and when the number of dimensions exceeds the number of samples.

Key Concepts in SVM:

1. **Hyperplane:** In SVM, a hyperplane is a decision boundary that separates the data into different classes. For example, in a 2D space, this is a line, and in a 3D space, it is a plane. SVM finds the hyperplane that maximizes the margin between two classes.

2. **Support Vectors:** Support vectors are the data points that are closest to the hyperplane. These points are crucial because they define the position and orientation of the hyperplane. The algorithm uses these support vectors to maximize the margin and decide the boundary that best separates the classes.

3. **Margin:** The margin is the distance between the hyperplane and the closest data point from either class. SVM tries to find the hyperplane that has the largest margin, meaning it maximizes the distance between the hyperplane and the support vectors. A larger margin leads to better generalization and robustness to new data.

4. **Linearly Separable Data:** When data is linearly separable, SVM can find a straight hyperplane that clearly divides the data into different classes. However, if the data is not linearly separable, SVM uses a kernel trick to transform the data into a higher-dimensional space where a hyperplane can separate the classes.

5. **Kernel Trick:** In cases where data is not linearly separable in its original space, SVM applies a kernel function to project the data into a higher-dimensional space where it becomes linearly separable. Common kernels include:

   Linear Kernel: Used for linearly separable data.

   Polynomial Kernel: Adds polynomial terms to account for non-linearity.

   Radial Basis Function (RBF) Kernel: Maps data into a high-dimensional space, often used when data has complex relationships.

How SVM Works:

1. **Training:** During the training phase, SVM takes labeled training data and tries to find the optimal hyperplane that separates the classes with the largest margin. It uses support vectors to calculate this margin.

2. **Classifying New Data:** Once the hyperplane is defined, new data points can be classified by determining which side of the hyperplane they fall on. The side corresponding to one class (e.g., "At Risk") will classify the point into that class, and the other side will classify the point into the second class (e.g., "Not At Risk").

Prediction:

SVM classifies a new data point based on which side of the hyperplane it lies on.

Model Training:

SVM is trained by solving the optimization problem to find the best hyperplane, using support vectors to define the margin.

g) Naive Bayes

Naive Bayes is a classification algorithm based on **Bayes' Theorem**. It assumes that the features (predictors) are independent of each other given the class label, which is a **naive** assumption, hence the name "Naive Bayes." Despite this assumption, Naive Bayes often performs well in practice, especially in applications like spam filtering, text classification, and medical diagnosis.

Bayes' Theorem

Bayes' Theorem describes the probability of an event, based on prior knowledge of conditions related to the event. It is expressed as:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Where:

- P(C|X): The probability of class C given the feature X (posterior probability)
- P(X|C): The probability of observing feature X given the class C (likelihood)
- P(C): The prior probability of class C
- P(X): The prior probability of the feature X

In simple terms, Naive Bayes calculates the probability of each class given the input features and predicts the class with the highest probability.

Naive Assumption

The "naive" part of the algorithm assumes that all features are **conditionally independent**, meaning that the presence or absence of one feature does not affect the presence of another,

given the class label. This is often not true in real-world data but allows for simplified computation.

For multiple features X= $\{x_1, x_2, x_3, \dots, x_n\}$, Bayes' Theorem becomes:

$$P(C|X) = \frac{P(C)\prod_{i=1}^{n} P(x_i|C)}{P(X)}$$

Where $P(x_i|C)$ is the conditional probability of each feature given the class.

Types of Naive Bayes Classifiers

There are several variations of Naive Bayes depending on how the feature probabilities $P(x_i|C)$ are computed:

1. **Gaussian Naive Bayes**: Assumes that the features follow a Gaussian (normal) distribution, making it suitable for continuous data.
2. **Multinomial Naive Bayes**: Commonly used for discrete data, especially in text classification where features represent the frequency of words.
3. **Bernoulli Naive Bayes**: Assumes that features are binary (1 if a feature is present, 0 if absent). It is often used for binary data.

Steps in Naive Bayes Classification

1. **Calculate Prior Probabilities**: Compute the prior probabilities for each class P(C).
2. **Calculate Conditional Probabilities**: For each feature x_i, calculate the likelihood $P(x_i|C)$ for each class.
3. **Calculate Posterior Probabilities**: Use Bayes' Theorem to calculate the posterior probability for each class P(C|X).
4. **Class Prediction**: Choose the class with the highest posterior probability as the predicted class.

Advantages of Naive Bayes

- **Fast and Efficient**: Requires less computational power and memory compared to other models, as it simplifies the computation of probabilities.

- **Works Well with High-Dimensional Data**: Performs well in cases with a large number of features.
- **Handles Categorical and Continuous Data**: Different versions (Multinomial, Gaussian, Bernoulli) can handle different types of data effectively.

Disadvantages of Naive Bayes

- **Naive Assumption of Independence**: The assumption that features are conditionally independent given the class is often unrealistic in real-world scenarios, which can reduce its performance.
- **Zero Probability Problem**: If a categorical variable has a category that wasn't present in the training dataset, the model may assign a probability of 0, which can be problematic. This issue can be resolved using **Laplace Smoothing**.
- **Limited Expressiveness**: Naive Bayes does not model interactions between features, which can limit its effectiveness when such interactions are important for prediction.

CHAPTER 4: DATA ANALYSIS

1. Import Modules

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

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

2. Loading Dataset

```
df = pd.read_csv('Cardiovascular_Disease_Dataset.csv')
```

```
df.head()
```

patientid	age	gender	chestpain	restingBP	serumcholestrol	fastingbloodsugar	restingrelectro	maxheartrate	exerciseangia	oldpeak	slope	noofmajorvessels	target
103368	53	1	2	171	0	0	1	147	0	5.3	3	3	1
119250	40	1	0	94	229	0	1	115	0	3.7	1	1	0
119372	49	1	2	133	142	0	0	202	1	5.0	1	0	0
132514	43	1	0	138	295	1	1	153	0	3.2	2	2	1
146211	31	1	1	199	0	0	2	136	0	5.3	3	2	1

3. Data Information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   patientid          1000 non-null   int64
 1   age                1000 non-null   int64
 2   gender             1000 non-null   int64
 3   chestpain          1000 non-null   int64
 4   restingBP          1000 non-null   int64
 5   serumcholestrol    1000 non-null   int64
 6   fastingbloodsugar  1000 non-null   int64
 7   restingrelectro    1000 non-null   int64
 8   maxheartrate       1000 non-null   int64
 9   exerciseangia      1000 non-null   int64
 10  oldpeak            1000 non-null   float64
 11  slope              1000 non-null   int64
 12  noofmajorvessels   1000 non-null   int64
 13  target             1000 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 109.5 KB
```

```
df.describe().T.style.background_gradient(axis=0)
```

	count	mean	std	min	25%	50%	75%	max
patientid	1000.000000	5048704.412000	2895904.500488	103368.000000	2536439.500000	4952508.500000	7681877.000000	9990855.000000
age	1000.000000	49.242000	17.864730	20.000000	34.000000	49.000000	64.250000	80.000000
gender	1000.000000	0.765000	0.424211	0.000000	1.000000	1.000000	1.000000	1.000000
chestpain	1000.000000	0.980000	0.953157	0.000000	0.000000	1.000000	2.000000	3.000000
restingBP	1000.000000	151.747000	29.965228	94.000000	129.000000	147.000000	181.000000	200.000000
serumcholestrol	1000.000000	311.447000	132.443801	0.000000	235.750000	318.000000	404.250000	602.000000
fastingbloodsugar	1000.000000	0.296000	0.456719	0.000000	0.000000	0.000000	1.000000	1.000000
restingrelectro	1000.000000	0.748000	0.770123	0.000000	0.000000	1.000000	1.000000	2.000000
maxheartrate	1000.000000	145.477000	34.190268	71.000000	119.750000	146.000000	175.000000	202.000000
exerciseangia	1000.000000	0.498000	0.500246	0.000000	0.000000	0.000000	1.000000	1.000000
oldpeak	1000.000000	2.707700	1.720753	0.000000	1.300000	2.400000	4.100000	6.200000
slope	1000.000000	1.540000	1.003697	0.000000	1.000000	2.000000	2.000000	3.000000
noofmajorvessels	1000.000000	1.222000	0.977585	0.000000	0.000000	1.000000	2.000000	3.000000
target	1000.000000	0.580000	0.493805	0.000000	0.000000	1.000000	1.000000	1.000000

4. Data Cleaning and Preprocessing Steps

1. Handling Missing Values

Conclusion: The output indicates that there are no missing values (0 missing values) for all columns in this dataset. This is a good sign because it means that there are no null or NaN values that need to be handled during preprocessing.

```
df.isna().sum()
```

```
patientid            0
age                  0
gender               0
chestpain            0
restingBP            0
serumcholestrol      0
fastingbloodsugar    0
restingrelectro      0
maxheartrate         0
exerciseangia        0
oldpeak              0
slope                0
noofmajorvessels     0
target               0
dtype: int64
```

2. Checking Data Type of Each Column

Conclusion:

1. Columns like gender, fastingbloodsugar, exerciseangia, and target are binary features that could be better represented as categorical data.
2. Columns like chestpain, restingrelectro, and slope are nominal features and should also be converted to categorical type (category dtype) to save memory and improve processing speed.

```
df.dtypes
```

```
patientid              int64
age                    int64
gender                 int64
chestpain              int64
restingBP              int64
serumcholestrol        int64
fastingbloodsugar      int64
restingrelectro        int64
maxheartrate           int64
exerciseangia          int64
oldpeak              float64
slope                  int64
noofmajorvessels       int64
target                 int64
dtype: object
```

3. Data Type Conversion

Why Convert to Category Type?

1. **Memory Efficiency:** The category dtype reduces memory usage when there are many repeated values in categorical columns, which is typical for binary and nominal features.
2. **Faster Computations:** Operations on categorical data can be faster since they are stored as integer codes internally rather than as strings.

```python
# Convert binary columns to categorical type
binary_cols = ['gender', 'fastingbloodsugar', 'exerciseangia', 'target']
df[binary_cols] = df[binary_cols].astype('category')

# Convert nominal columns to categorical type
nominal_cols = ['chestpain', 'restingrelectro', 'slope']
df[nominal_cols] = df[nominal_cols].astype('category')

# Check the updated data types
print(df.dtypes)
```

```
patientid               int64
age                     int64
gender               category
chestpain            category
restingBP               int64
serumcholestrol         int64
fastingbloodsugar    category
restingrelectro      category
maxheartrate            int64
exerciseangia        category
oldpeak               float64
slope                category
noofmajorvessels        int64
target               category
dtype: object
```

4. Handling Duplicated values

Conclusion: No duplicate values were found

```
df.duplicated().sum
```

```
<bound method Series.sum of 0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool>
```

```
# Remove duplicates
df.drop_duplicates(inplace=True)
```

```
df.shape
```

```
(1000, 14)
```

5. Handling Outliers

Conclusion:

1. No Outliers Detected: Both methods (IQR and Z-Score) show that there are no outliers in the numeric columns of the dataset. This indicates that the data points in the numeric features such as age, restingBP, serumcholestrol, maxheartrate, oldpeak, and noofmajorvessels fall within the expected range and there are no extreme values that deviate significantly from the rest of the data.
2. This is good for analysis and model-building, as outliers can sometimes skew the results of statistical tests and machine learning algorithms.

a) Outlier Detection Using IQR

```
# Define a function to detect outliers using IQR
def detect_outliers_iqr(df):
    outliers = {}
    for col in df.select_dtypes(include=['float64', 'int64']).columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers[col] = df[(df[col] < lower_bound) | (df[col] > upper_bound)].shape[0]  # Count of outliers
    return outliers

# Check for outliers in each numeric column using IQR
outliers_iqr = detect_outliers_iqr(df)
print("Outliers based on IQR:", outliers_iqr)
```

```
Outliers based on IQR: {'patientid': 0, 'age': 0, 'restingBP': 0, 'serumcholestrol': 0, 'maxheartrate': 0, 'oldpeak': 0, 'noofmajorvessels': 0}
```

b) Outlier Detection Using Z-Score

```
from scipy.stats import zscore

# Define a function to detect outliers using Z-Score
def detect_outliers_zscore(df):
    outliers = {}
    for col in df.select_dtypes(include=['float64', 'int64']).columns:
        z_scores = zscore(df[col].dropna())  # Remove NaN before calculating Z-score
        outliers[col] = (abs(z_scores) > 3).sum()  # Count of outliers (Z-score > 3 or < -3)
    return outliers

# Check for outliers in each numeric column using Z-Score
outliers_zscore = detect_outliers_zscore(df)
print("Outliers based on Z-Score:", outliers_zscore)
```

```
Outliers based on Z-Score: {'patientid': 0, 'age': 0, 'restingBP': 0, 'serumcholestrol': 0, 'maxheartrate': 0, 'oldpeak': 0, 'noofmajorvessels': 0}
```

6. Encode categorical variables

```
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Label Encoding for Binary columns (Gender, Fasting Blood Sugar, Exercise Angina, Target)
binary_cols = ['gender', 'fastingbloodsugar', 'exerciseangia', 'target']
label_encoder = LabelEncoder()

for col in binary_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

```
df.columns
```

```
Index(['patientid', 'age', 'gender', 'chestpain', 'restingBP',
       'serumcholestrol', 'fastingbloodsugar', 'restingrelectro',
       'maxheartrate', 'exerciseangia', 'oldpeak', 'slope', 'noofmajorvessels',
       'target'],
      dtype='object')
```

```
# One-Hot Encoding for Nominal columns (Chest Pain, Resting Electrocardiogram, Slope)
df = pd.get_dummies(df, columns=['chestpain', 'restingrelectro', 'slope'], drop_first=False)
```

```
df.columns
```

```
Index(['patientid', 'age', 'gender', 'restingBP', 'serumcholestrol',
       'fastingbloodsugar', 'maxheartrate', 'exerciseangia', 'oldpeak',
       'noofmajorvessels', 'target', 'chestpain_0', 'chestpain_1',
       'chestpain_2', 'chestpain_3', 'restingrelectro_0', 'restingrelectro_1',
       'restingrelectro_2', 'slope_0', 'slope_1', 'slope_2', 'slope_3'],
      dtype='object')
```

7. Feature Scaling

```
# Min-Max Scaling
from sklearn.preprocessing import MinMaxScaler

# Initialize the Min-Max scaler
scaler = MinMaxScaler()

# List of numeric columns
numeric_cols = ['age', 'restingBP', 'serumcholestrol', 'maxheartrate', 'oldpeak', 'noofmajorvessels']

# Apply Min-Max scaling to the selected columns
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Display first few rows after preprocessing
print("Cleaned and Preprocessed Data:")
print(df.head())
```

```
Cleaned and Preprocessed Data:
   patientid       age  gender  restingBP  serumcholestrol  fastingbloodsugar  \
0     103368  0.550000       1   0.726415         0.000000                  0
1     119250  0.333333       1   0.000000         0.380399                  0
2     119372  0.483333       1   0.367925         0.235880                  0
3     132514  0.383333       1   0.415094         0.490033                  1
4     146211  0.183333       1   0.990566         0.000000                  0

   maxheartrate  exerciseangia   oldpeak  noofmajorvessels  ...  chestpain_1  \
0      0.580153              0  0.854839          1.000000  ...        False
1      0.335878              0  0.596774          0.333333  ...        False
2      1.000000              1  0.806452          0.000000  ...        False
3      0.625954              0  0.516129          0.666667  ...        False
4      0.496183              0  0.854839          0.666667  ...         True

   chestpain_2  chestpain_3  restingrelectro_0  restingrelectro_1  \
0         True        False              False               True
1        False        False              False               True
2         True        False               True              False
3        False        False              False               True
4        False        False              False              False

   restingrelectro_2  slope_0  slope_1  slope_2  slope_3
0              False    False    False    False     True
1              False    False     True    False    False
2              False    False     True    False    False
3              False    False    False     True    False
4               True    False    False    False     True

[5 rows x 22 columns]
```

```python
# Convert all boolean columns to integers (0 and 1)
bool_cols = df.select_dtypes(include='bool').columns
df[bool_cols] = df[bool_cols].astype('int64')
```

```python
df.info()
```

```
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   patientid          1000 non-null   int64
 1   age                1000 non-null   float64
 2   gender             1000 non-null   int64
 3   restingBP          1000 non-null   float64
 4   serumcholestrol    1000 non-null   float64
 5   fastingbloodsugar  1000 non-null   int64
 6   maxheartrate       1000 non-null   float64
 7   exerciseangia      1000 non-null   int64
 8   oldpeak            1000 non-null   float64
 9   noofmajorvessels   1000 non-null   float64
 10  target             1000 non-null   int64
 11  chestpain_0        1000 non-null   int64
 12  chestpain_1        1000 non-null   int64
 13  chestpain_2        1000 non-null   int64
 14  chestpain_3        1000 non-null   int64
 15  restingrelectro_0  1000 non-null   int64
 16  restingrelectro_1  1000 non-null   int64
 17  restingrelectro_2  1000 non-null   int64
 18  slope_0            1000 non-null   int64
 19  slope_1            1000 non-null   int64
 20  slope_2            1000 non-null   int64
 21  slope_3            1000 non-null   int64
dtypes: float64(6), int64(16)
memory usage: 172.0 KB
```

8. Feature Selection

1. Check Correlation

Based on the output of the correlation matrix, here's a conclusion regarding the relationship between features and the target variable (heart disease presence):

Strongest Positive Correlations with Target:

1. slope_2 (0.534405): The variable slope_2 (flat slope in the peak exercise ST segment) has the highest positive correlation with the target variable, indicating that a flat slope is strongly associated with the presence of heart disease.
2. noofmajorvessels (0.489866): The number of major vessels is also positively correlated with the target, meaning individuals with more major vessels are more likely to have heart disease.

3. restingBP (0.482387): Resting blood pressure has a moderate positive correlation with heart disease, suggesting higher resting blood pressure is associated with an increased likelihood of having heart disease.
4. chestpain_2 (0.428739): Chest pain type 2 (non-anginal pain) also shows a moderate positive correlation with the presence of heart disease.

Moderate Positive Correlations:

1. slope_3 (0.424151): Downsloping of the ST segment during peak exercise is moderately positively correlated with heart disease.
2. restingrelectro_2 (0.387768): Resting electrocardiogram result showing probable or definite left ventricular hypertrophy is moderately correlated with the target.
3. fastingbloodsugar (0.303233): Fasting blood sugar (value of 1 indicating higher than 120 mg/dl) has a moderate positive correlation, implying that elevated fasting blood sugar levels are linked to heart disease.

Weak to Very Weak Positive Correlations:

1. maxheartrate (0.228343): A positive but weak correlation with max heart rate achieved during exercise.
2. serumcholestrol (0.195340): A weak positive correlation with serum cholesterol, indicating that higher cholesterol might be slightly related to heart disease.
3. chestpain_3 (0.133167): Chest pain type 3 (asymptomatic pain) has a very weak positive correlation with the target.
4. chestpain_1 (0.117021): Chest pain type 1 (typical angina) also has a weak positive correlation with heart disease.

Negative Correlations:

1. restingrelectro_0 (-0.347207): A negative correlation with resting electrocardiogram result (normal), suggesting that a normal ECG result is associated with a lower likelihood of heart disease.

2. slope_1 (-0.453264): The upsloping of the ST segment during peak exercise has a negative correlation, meaning it is less likely to indicate heart disease.
3. slope_0 (-0.550578): A stronger negative correlation with an upsloping ST segment, further suggesting that a normal or upsloping ST segment during exercise is not strongly associated with heart disease.
4. chestpain_0 (-0.556650): Chest pain type 0 (typical angina) has the strongest negative correlation with the presence of heart disease in this dataset.

Weak or No Correlation:

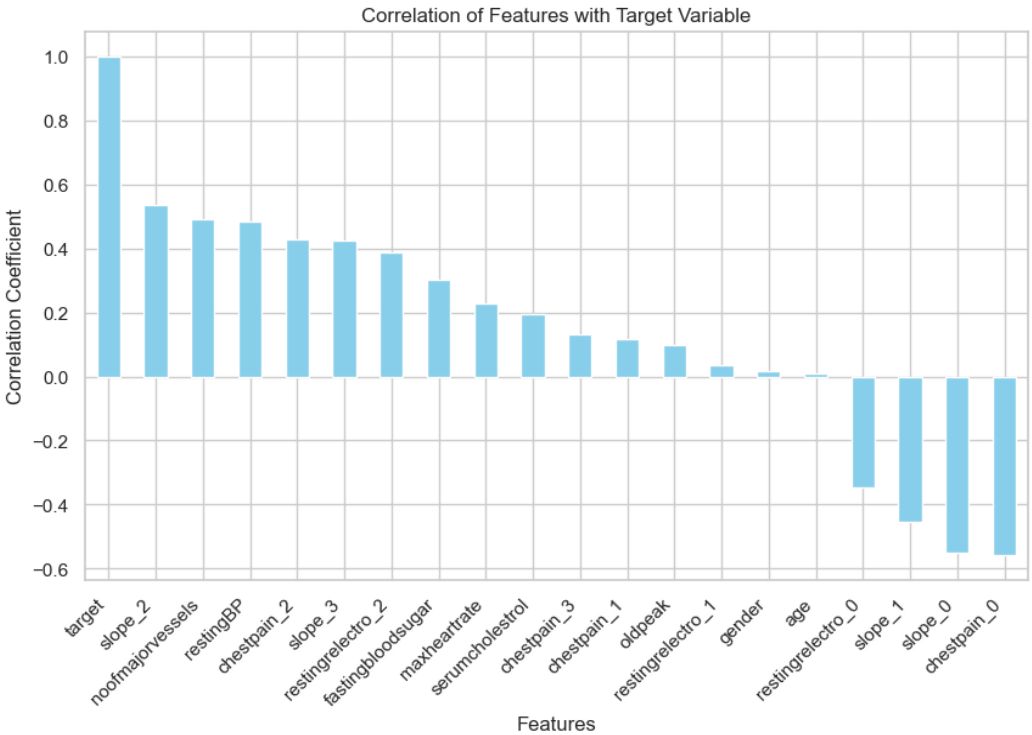
1. gender (0.015769): Gender has almost no correlation with the target variable.
2. age (0.008356): Age shows a very weak correlation with heart disease in this dataset, implying age alone might not be a significant predictor in this case.
3. exerciseangia (-0.039874): Exercise-induced angina shows a very weak negative correlation with heart disease, suggesting it is not a strong predictor in this dataset.

Conclusion: Important Features: The most important features based on the correlation matrix are slope_2, noofmajorvessels, restingBP, chestpain_2, and slope_3 since they show strong to moderate positive correlations with the target variable. **Less Useful Features:** Features like gender, age, and exerciseangia have very weak or negligible correlations with the target, making them less relevant for prediction. **Negative Correlations:** Variables like slope_0, slope_1, and chestpain_0 have negative correlations with heart disease, indicating that they are associated with a lower likelihood of heart disease.

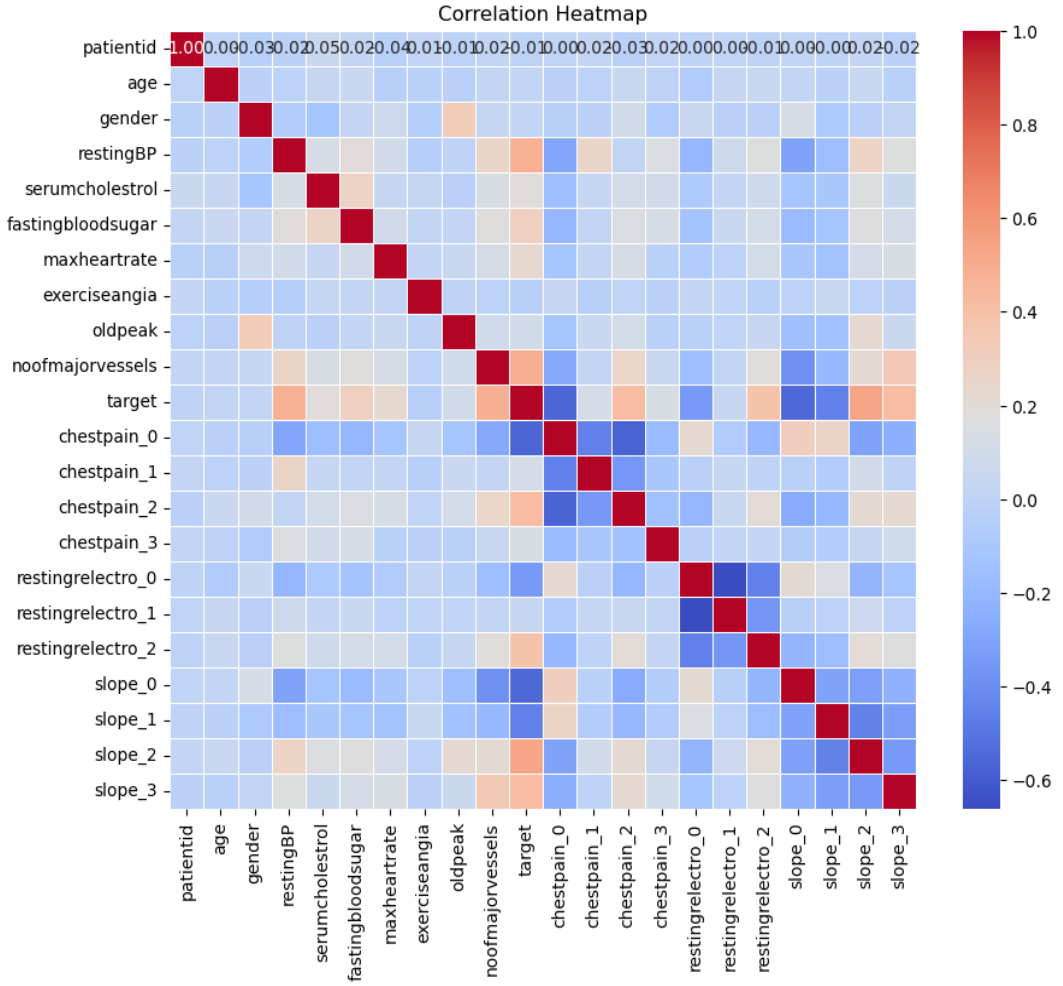
This analysis can guide us in selecting features for your model, focusing on those with higher correlation and excluding those that have little to no impact.

```
# Calculate correlation matrix
correlation_matrix = df.corr()

# Display correlation with the target variable
print(correlation_matrix['target'].sort_values(ascending=False))
```

```
target                1.000000
slope_2               0.534405
noofmajorvessels      0.489866
restingBP             0.482387
chestpain_2           0.428739
slope_3               0.424151
restingrelectro_2     0.387768
fastingbloodsugar     0.303233
maxheartrate          0.228343
serumcholestrol       0.195340
chestpain_3           0.133167
chestpain_1           0.117021
oldpeak               0.098053
restingrelectro_1     0.036168
gender                0.015769
age                   0.008356
patientid            -0.005637
exerciseangia        -0.039874
restingrelectro_0    -0.347207
slope_1              -0.453264
slope_0              -0.550578
chestpain_0          -0.556650
Name: target, dtype: float64
```



```
# Correlation Heatmap
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



2. Drop Irrelevant Columns

```
# Drop 'patientid'
df = df.drop(columns=['patientid'])
```

```
# Drop 'exerciseangia'
df = df.drop(columns=['exerciseangia'])
```

Keeping the gender and age features in the dataset, even though they have a weak correlation with the target variable. Since my research focuses specifically on the

impact of age and gender on cardiovascular disease prediction, it's important to include these variables in the analysis.

Including age and gender will allow us to assess whether they have any meaningful impact on the prediction model and to draw conclusions about their role in predicting cardiovascular disease within the context of my research. After model training, we can evaluate the feature importance and assess the predictive power of these variables, even if their correlation with the target variable is weak.

5. Exploratory Data Analysis (Eda)

Summary statistics for each feature

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	1000.0	0.487367	0.297746	0.0	0.233333	0.483333	0.737500	1.0
gender	1000.0	0.765000	0.424211	0.0	1.000000	1.000000	1.000000	1.0
restingBP	1000.0	0.544783	0.282691	0.0	0.330189	0.500000	0.820755	1.0
serumcholestrol	1000.0	0.517354	0.220006	0.0	0.391611	0.528239	0.671512	1.0
fastingbloodsugar	1000.0	0.296000	0.456719	0.0	0.000000	0.000000	1.000000	1.0
maxheartrate	1000.0	0.568527	0.260994	0.0	0.372137	0.572519	0.793893	1.0
oldpeak	1000.0	0.436726	0.277541	0.0	0.209677	0.387097	0.661290	1.0
noofmajorvessels	1000.0	0.407333	0.325862	0.0	0.000000	0.333333	0.666667	1.0
target	1000.0	0.580000	0.493805	0.0	0.000000	1.000000	1.000000	1.0
chestpain_0	1000.0	0.420000	0.493805	0.0	0.000000	0.000000	1.000000	1.0
chestpain_1	1000.0	0.224000	0.417131	0.0	0.000000	0.000000	0.000000	1.0
chestpain_2	1000.0	0.312000	0.463542	0.0	0.000000	0.000000	1.000000	1.0
chestpain_3	1000.0	0.044000	0.205198	0.0	0.000000	0.000000	0.000000	1.0
restingrelectro_0	1000.0	0.454000	0.498129	0.0	0.000000	0.000000	1.000000	1.0
restingrelectro_1	1000.0	0.344000	0.475279	0.0	0.000000	0.000000	1.000000	1.0
restingrelectro_2	1000.0	0.202000	0.401693	0.0	0.000000	0.000000	0.000000	1.0
slope_0	1000.0	0.180000	0.384380	0.0	0.000000	0.000000	0.000000	1.0
slope_1	1000.0	0.299000	0.458049	0.0	0.000000	0.000000	1.000000	1.0
slope_2	1000.0	0.322000	0.467477	0.0	0.000000	0.000000	1.000000	1.0
slope_3	1000.0	0.199000	0.399448	0.0	0.000000	0.000000	0.000000	1.0

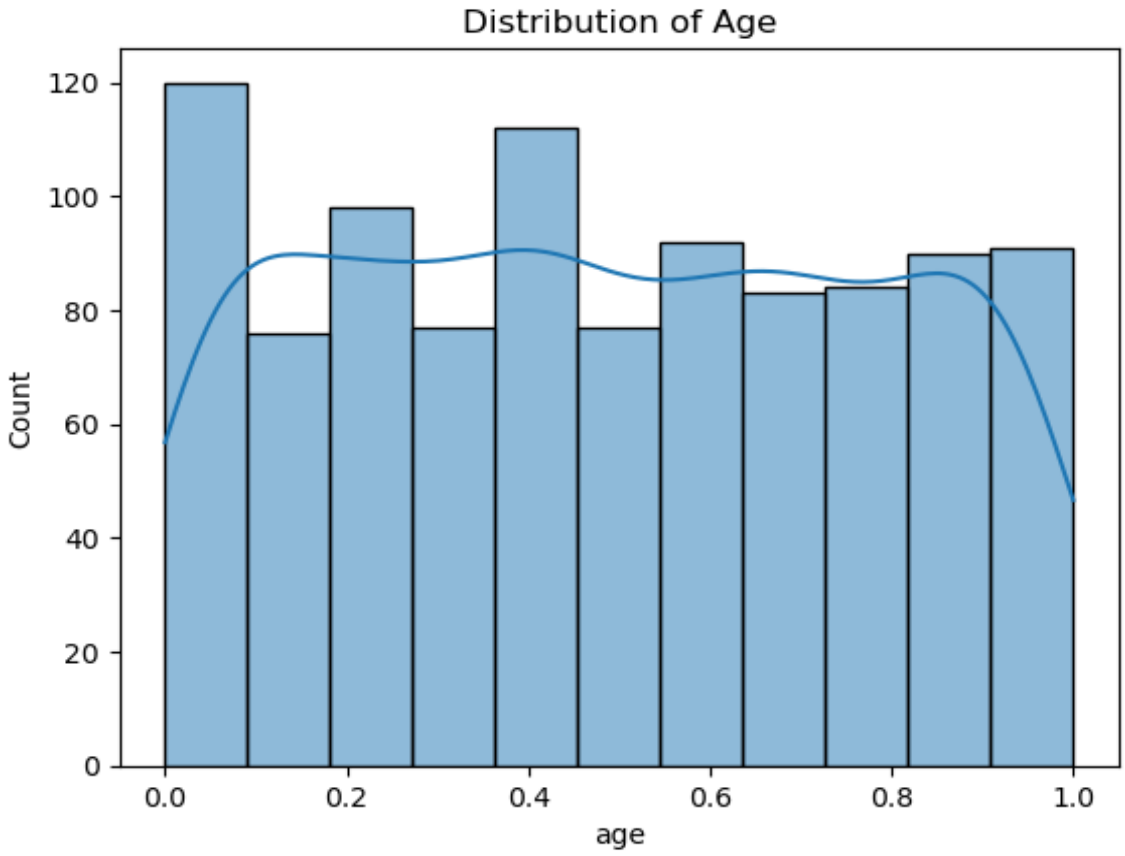
Distribution of each numeric feature

Numeric features like:

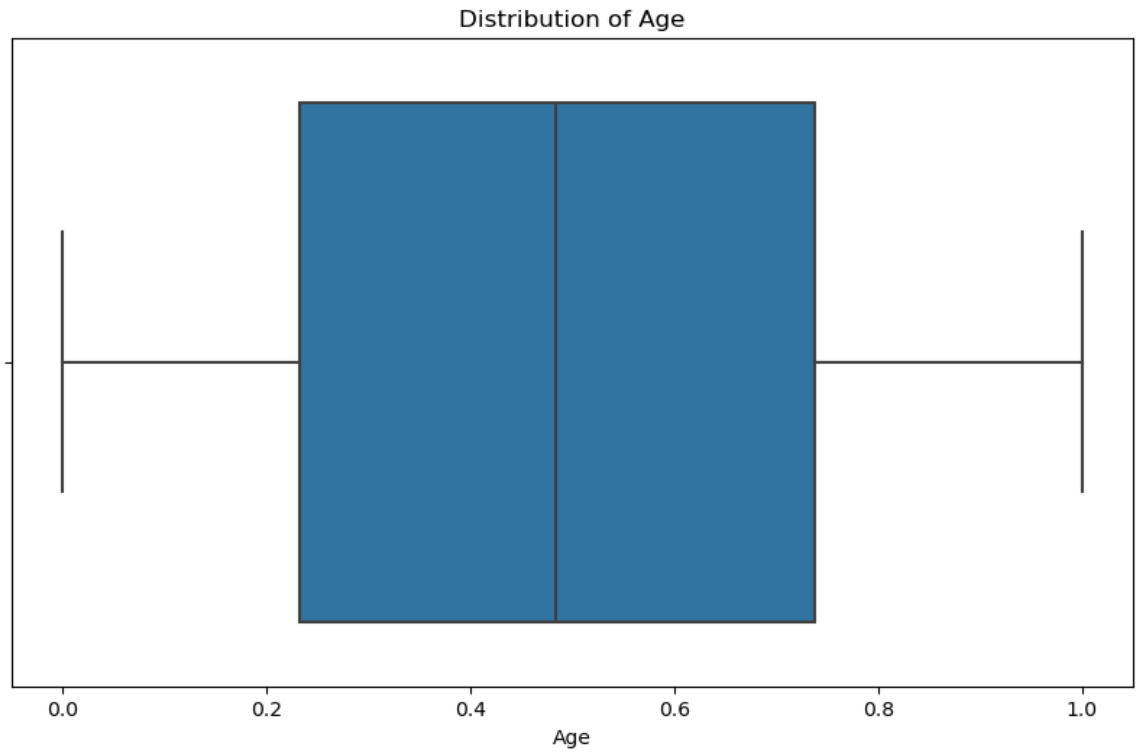
- age
- restingBP
- serumcholestrol
- maxheartrate
- oldpeak
- noofmajorvessels

1. Age

```
# 'age'
sns.histplot(df['age'], kde=True)
plt.title('Distribution of Age')
plt.show()
```

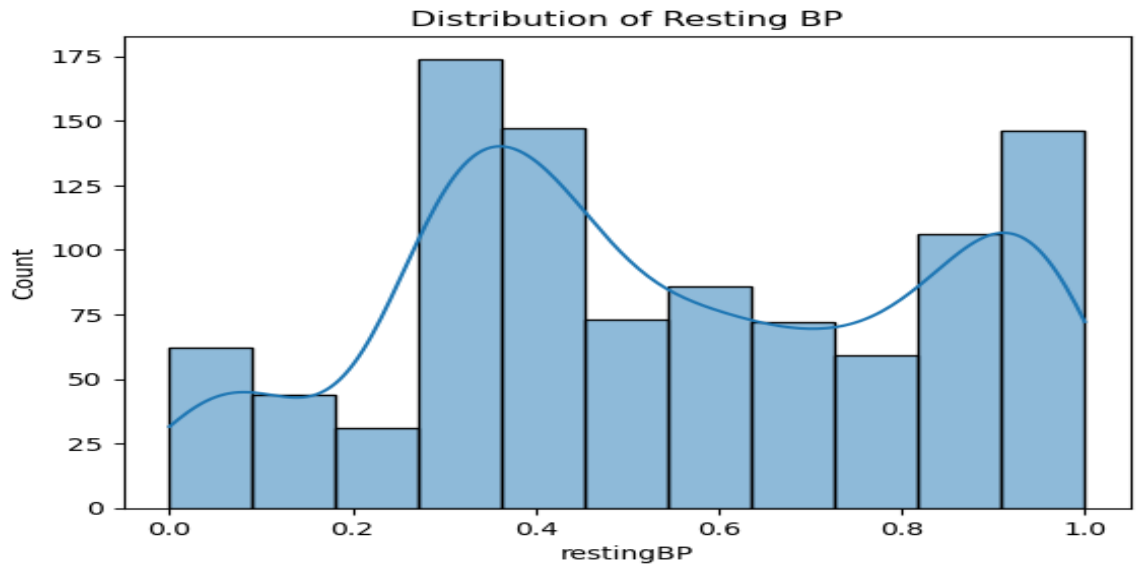


```
plt.figure(figsize=(10, 6))
sns.boxplot(x='age', data=df)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.show()
```



2. Resting blood pressure

```
# 'restingBP'
sns.histplot(df['restingBP'], kde=True)
plt.title('Distribution of Resting BP')
plt.show()
```

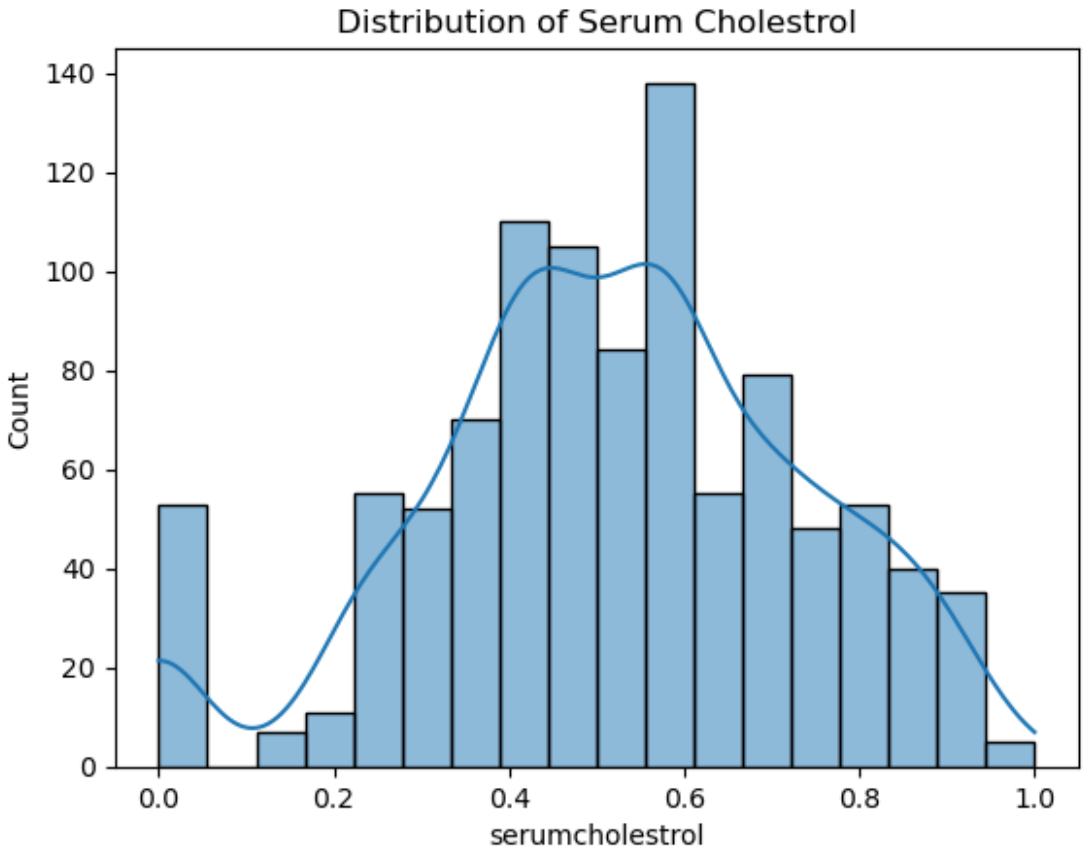


```
# Average resting blood pressure among the patients
average_resting_bp = df['restingBP'].mean()
print(f"Average Resting Blood Pressure: {average_resting_bp:.2f} mm Hg")
```

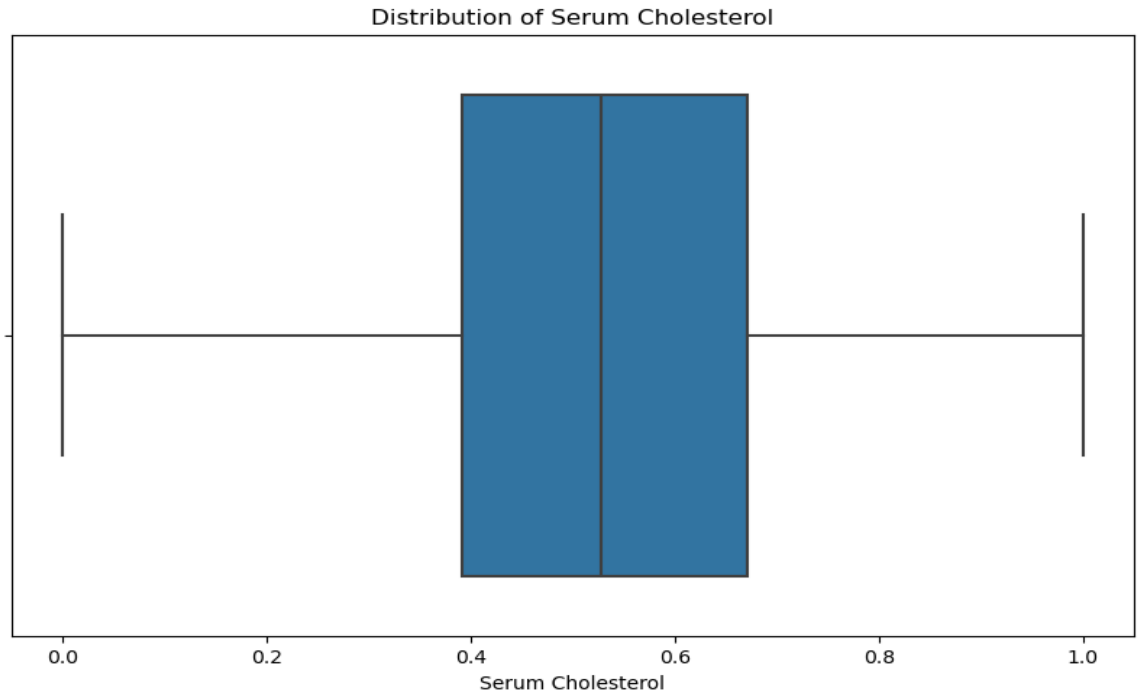
```
Average Resting Blood Pressure: 0.54 mm Hg
```

3. Serum cholesterol

```
# 'serumcholestrol'
sns.histplot(df['serumcholestrol'], kde=True)
plt.title('Distribution of Serum Cholestrol')
plt.show()
```

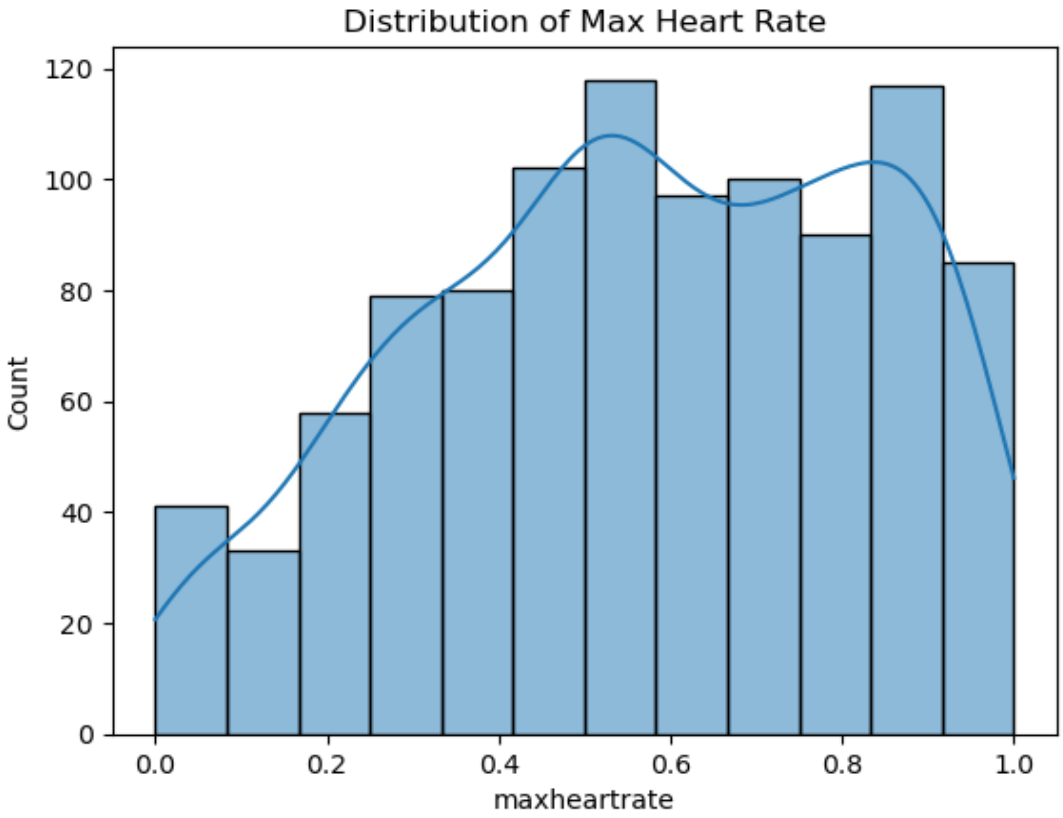


```
plt.figure(figsize=(10, 6))
sns.boxplot(x='serumcholestrol', data=df)
plt.title('Distribution of Serum Cholesterol')
plt.xlabel('Serum Cholesterol')
plt.show()
```



4. Maximum heart rate achieved

```
# 'maxheartrate'
sns.histplot(df['maxheartrate'], kde=True)
plt.title('Distribution of Max Heart Rate')
plt.show()
```

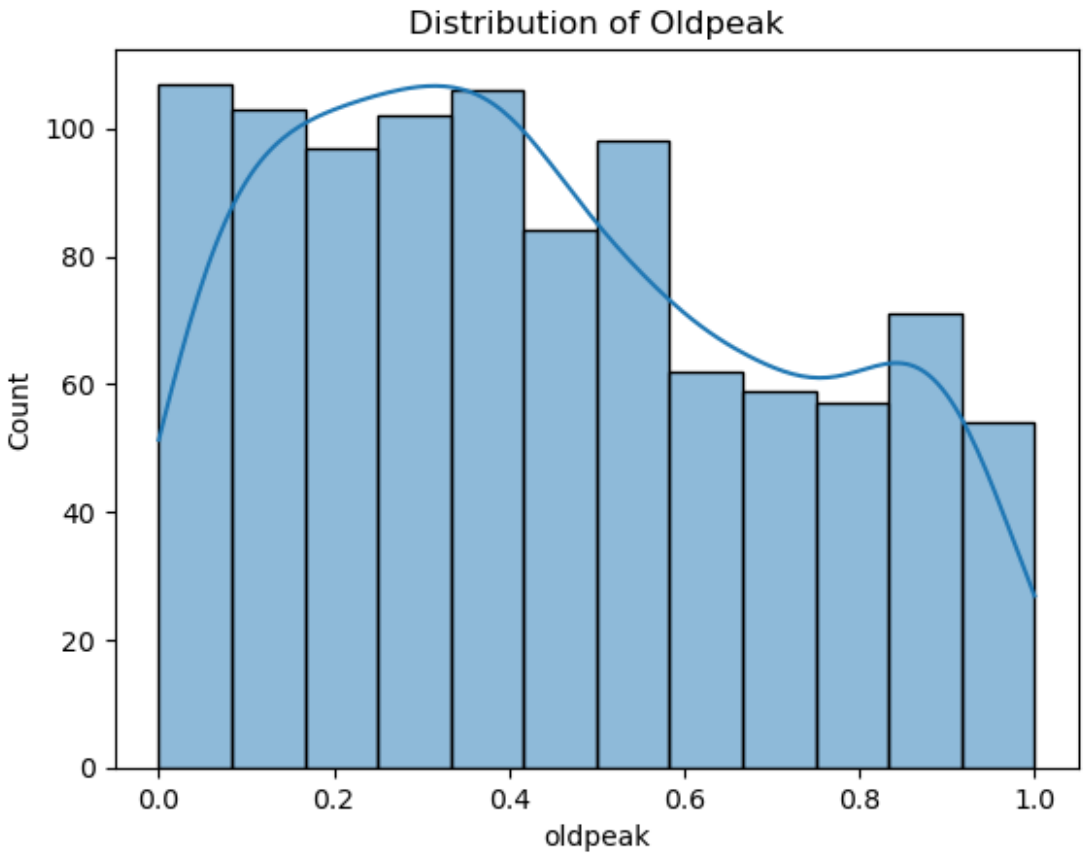



```
# What is the average maximum heart rate achieved by the patients on average?
average_max_heart_rate = df['maxheartrate'].mean()
print(f"Average Maximum Heart Rate: {average_max_heart_rate:.2f}")
```

```
Average Maximum Heart Rate: 0.57
```

5. Oldpeak =ST

```
# 'oldpeak'
sns.histplot(df['oldpeak'], kde=True)
plt.title('Distribution of Oldpeak')
plt.show()
```

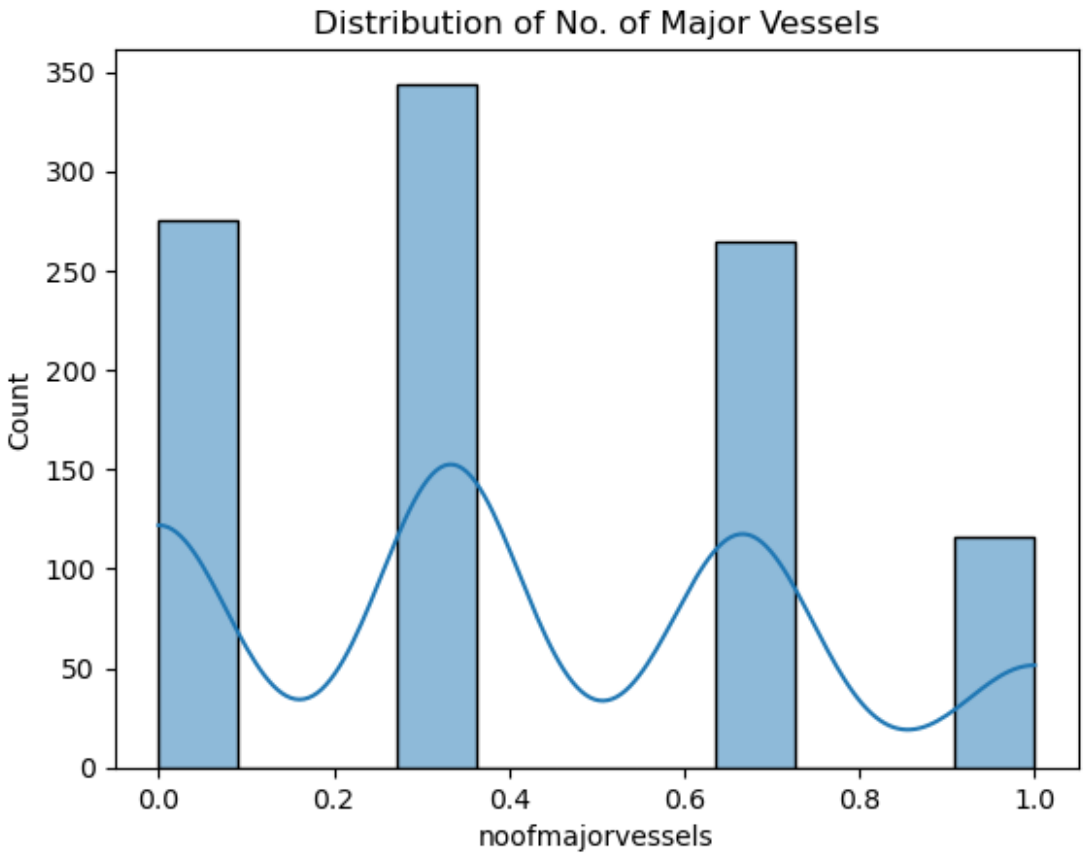



```
# What is the average oldpeak (ST depression induced by exercise relative to rest) among the patients?
average_oldpeak = df['oldpeak'].mean()
print(f"Average Oldpeak: {average_oldpeak:.2f}")
```

```
Average Oldpeak: 0.44
```

6. Number of major vessels

```
# 'noofmajorvessels'
sns.histplot(df['noofmajorvessels'], kde=True)
plt.title('Distribution of No. of Major Vessels')
plt.show()
```




```
# What is the range of the number of major vessels in the patients?
vessels_range = f"Number of Major Vessels Range: {df['noofmajorvessels'].min()} - {df['noofmajorvessels'].max()}"
print(vessels_range)
```

```
Number of Major Vessels Range: 0.0 - 1.0
```

```
# How many patients have all three major vessels showing defects?
defects_count = len(df[df['noofmajorvessels'] == 3])
print(f"Number of Patients with Defects in All Three Major Vessels: {defects_count}")
```

```
Number of Patients with Defects in All Three Major Vessels: 0
```

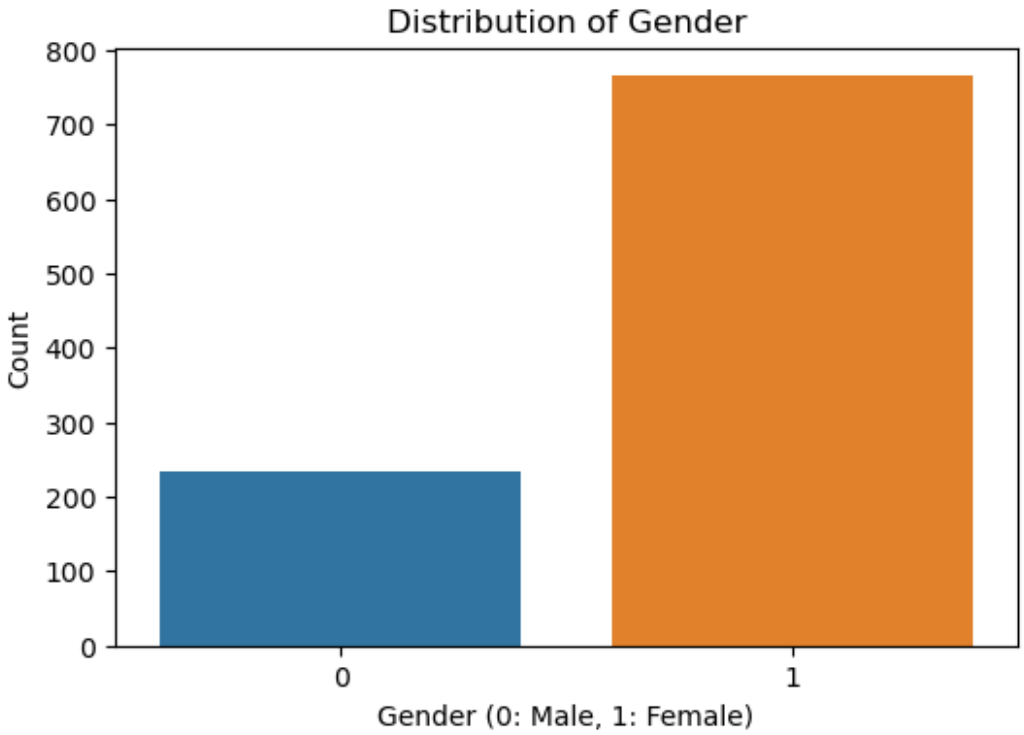
Distribution of each binary or nominal feature

Binary or nominal features like:

- gender
- chestpain
- fastingbloodsugar
- exerciseangia
- restingrelectro
- slope

7. Gender

```
# Count plot for gender
plt.figure(figsize=(6, 4))
sns.countplot(x='gender', data=df)
plt.title('Distribution of Gender')
plt.xlabel('Gender (0: Male, 1: Female)')
plt.ylabel('Count')
plt.show()
```

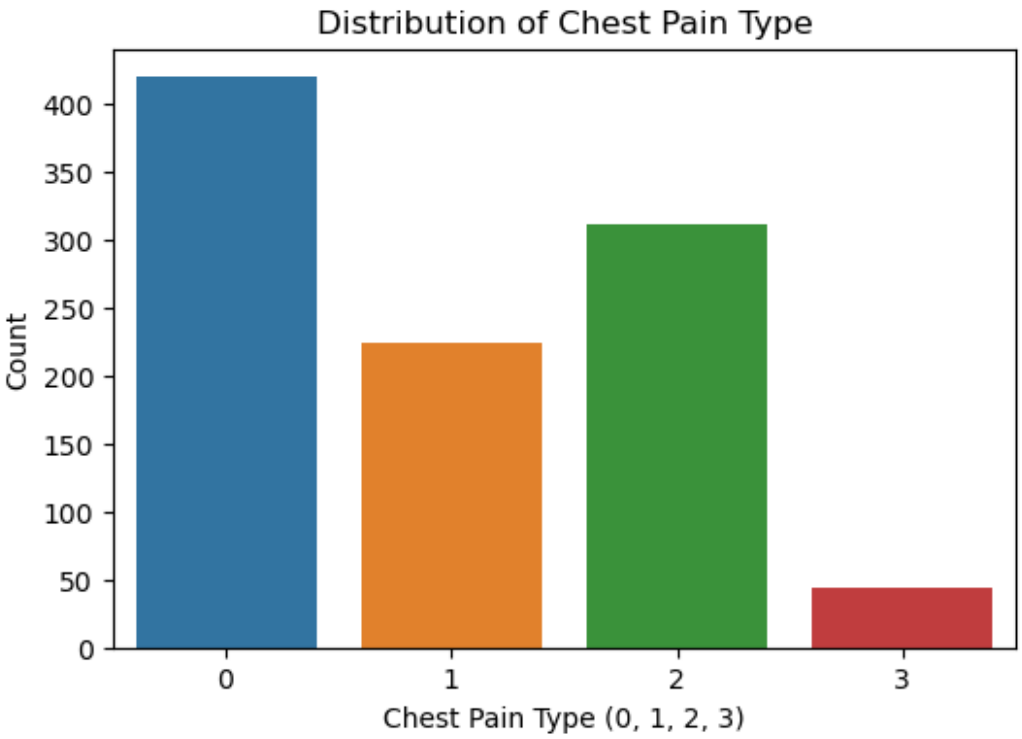



```
gender_count = df['gender'].value_counts()
print(gender_count)
```

```
gender
1    765
0    235
Name: count, dtype: int64
```

8. Chest pain type

```
# Count plot for chestpain
''' Value 0: typical angina
    Value 1: atypical angina
    Value 2: non-anginal pain
    Value 3: asymptomatic '''

# Create a new column for chest pain type by taking the argmax of the one-hot encoded columns
df['chestpain'] = df[['chestpain_0', 'chestpain_1', 'chestpain_2', 'chestpain_3']].idxmax(axis=1).apply(lambda x: int(x.split('_')[1]))

# Now plot the count plot using the new 'chestpain' column
plt.figure(figsize=(6, 4))
sns.countplot(x='chestpain', data=df)
plt.title('Distribution of Chest Pain Type')
plt.xlabel('Chest Pain Type (0, 1, 2, 3)')
plt.ylabel('Count')
plt.show()
```

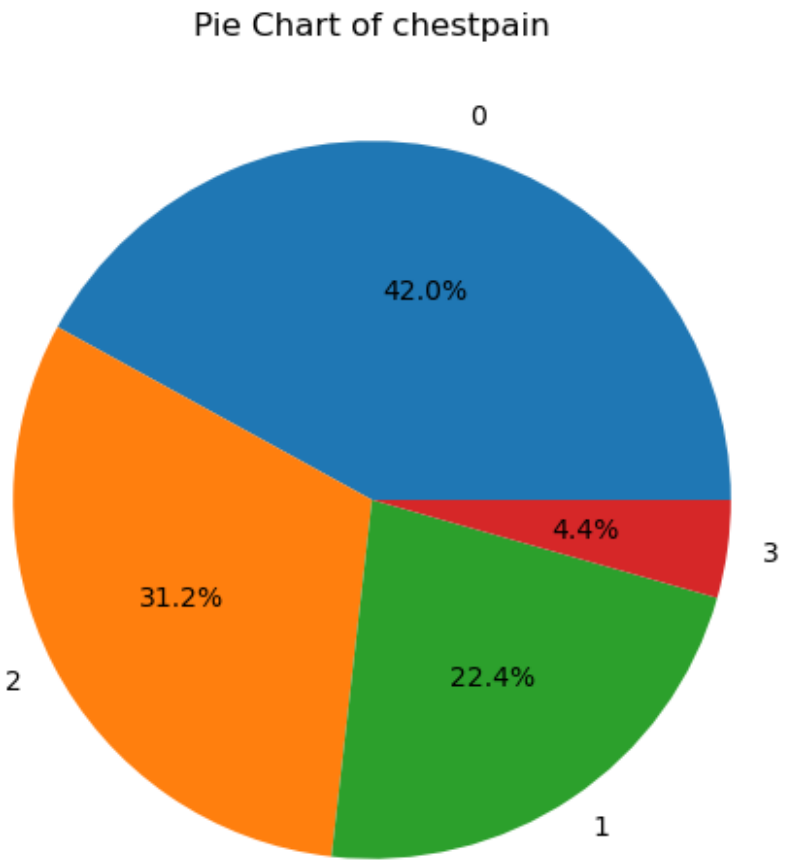



```
print(df['chestpain'].unique())
print(df['chestpain'].value_counts())
chest = {
    0: "typical angina",
    1: "atypical angina",
    2: "non-anginal pain",
    3: "asymptomatic"
}
```

```
[2 0 1 3]
chestpain
0    420
2    312
1    224
3     44
Name: count, dtype: int64
```

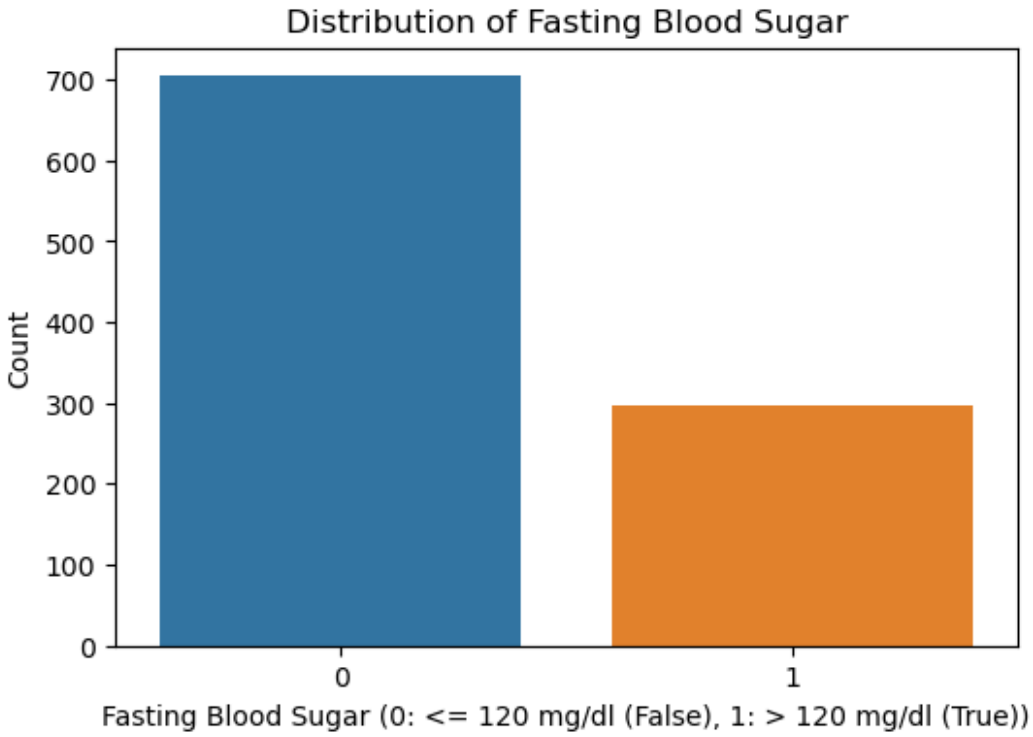
```
def PiePlot(column):
    '''
    Parameters:
    -column(Optional): column in df to be visualized
    -shadow(Optional): Apply shadow effect on pie chart

    '''
    plt.figure(figsize=(12,6))
    plt.pie(df[column].value_counts(), labels = df[column].value_counts().index, autopct = "%1.1f%%")
    plt.title(f'Pie Chart of {column}')
    plt.show()
```

```
chest = {
    0: "typical angina",
    1: "atypical angina",
    2: "non-anginal pain",
    3: "asymptomatic"
}
PiePlot('chestpain')
for i in np.sort(df['chestpain'].unique()):
    print(f'Type {i} is {chest[i]}')
```



9. Fasting Blood Sugar

```
# Count plot for fastingbloodsugar
plt.figure(figsize=(6, 4))
sns.countplot(x='fastingbloodsugar', data=df)
plt.title('Distribution of Fasting Blood Sugar')
plt.xlabel('Fasting Blood Sugar (0: <= 120 mg/dl (False), 1: > 120 mg/dl (True))') # 0,1 > 120 mg/dl (0 = false , 1 = true)
plt.ylabel('Count')
plt.show()
```




```
fastingbloodsugar = df['fastingbloodsugar'].value_counts()
print(fastingbloodsugar)
```

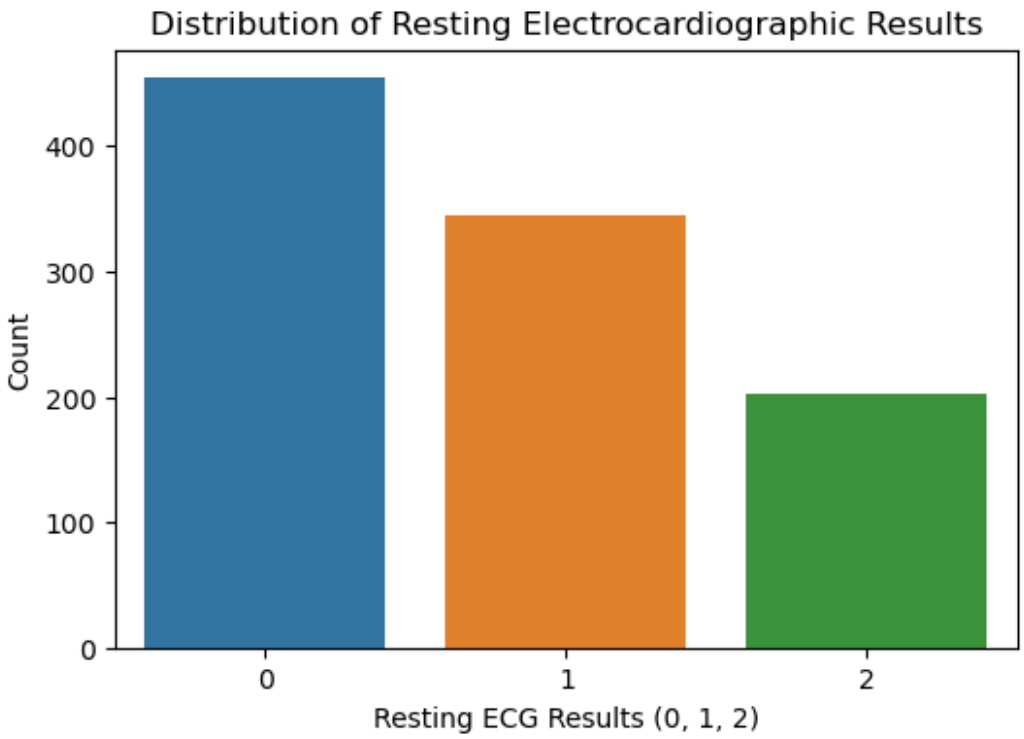
```
fastingbloodsugar
0    704
1    296
Name: count, dtype: int64
```

```
# What percentage of patients have fasting blood sugar greater than 120 mg/dl?
percentage_high_fasting_sugar = (df['fastingbloodsugar'].sum() / len(df)) * 100
print(f"Percentage of patients with fasting blood sugar > 120 mg/dl: {percentage_high_fasting_sugar:.2f}%")
```

```
Percentage of patients with fasting blood sugar > 120 mg/dl: 29.60%
```

10. Resting Electrocardiographic Results

```
# Count plot for restingrelectro
'''
Value 0: normal,
Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV),
Value 2: showing probable or definite left ventricular hypertrophy by Estes'criteria
'''
# 'restingrelectro_0', 'restingrelectro_1', 'restingrelectro_2' are the one-hot encoded columns
# Combine these columns into a single 'restingrelectro' column with their respective values

df['restingrelectro'] = df[['restingrelectro_0', 'restingrelectro_1', 'restingrelectro_2']].idxmax(axis=1).apply(lambda x: int(x.split('_')[1]))

# Now plot the count plot using the new 'restingrelectro' column
plt.figure(figsize=(6, 4))
sns.countplot(x='restingrelectro', data=df)
plt.title('Distribution of Resting Electrocardiographic Results')
plt.xlabel('Resting ECG Results (0, 1, 2)')
plt.ylabel('Count')
plt.show()
```

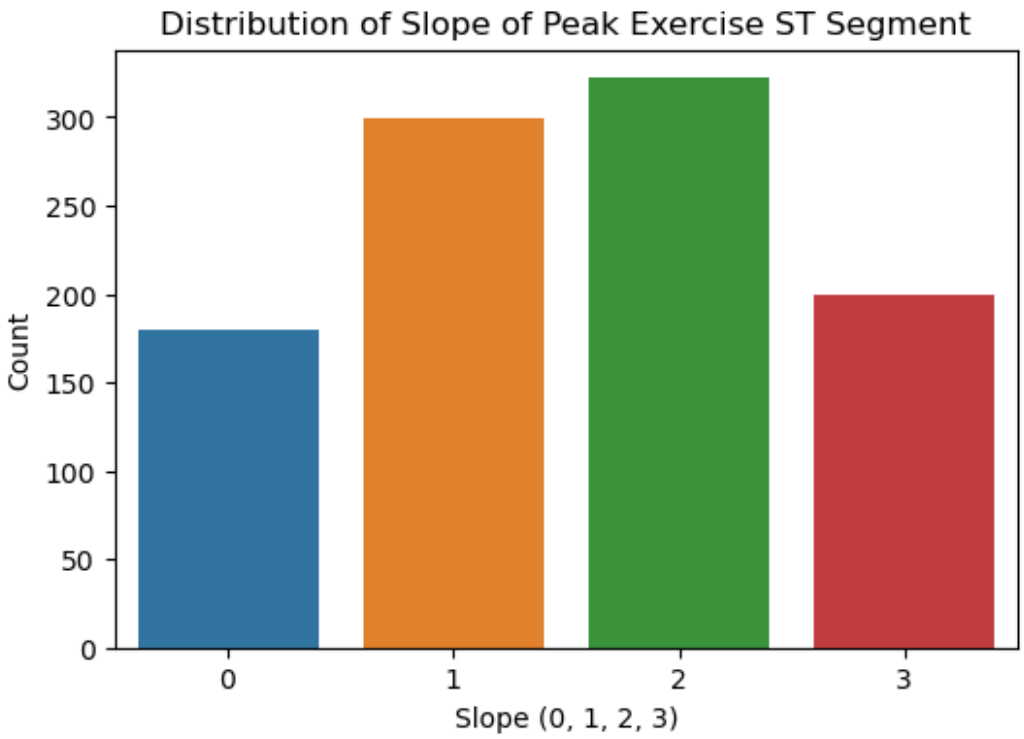



```
# What are the predominant resting electrocardiogram results in the dataset?
resting_electro_counts = df['restingrelectro'].value_counts()
print(resting_electro_counts)
```

```
restingrelectro
0    454
1    344
2    202
Name: count, dtype: int64
```

12. Slope of Peak Exercise ST Segment

```
# Count plot for slope
# Combine the one-hot encoded slope columns into a single 'slope' column
df['slope'] = df[['slope_0', 'slope_1', 'slope_2', 'slope_3']].idxmax(axis=1).apply(lambda x: int(x.split('_')[1]))

# Now plot the count plot using the new 'slope' column
plt.figure(figsize=(6, 4))
sns.countplot(x='slope', data=df)
plt.title('Distribution of Slope of Peak Exercise ST Segment')
plt.xlabel('Slope (0, 1, 2, 3)') # 1, 2, 3 (1-upsloping, 2-flat, 3-downsloping)
plt.ylabel('Count')
plt.show()
```



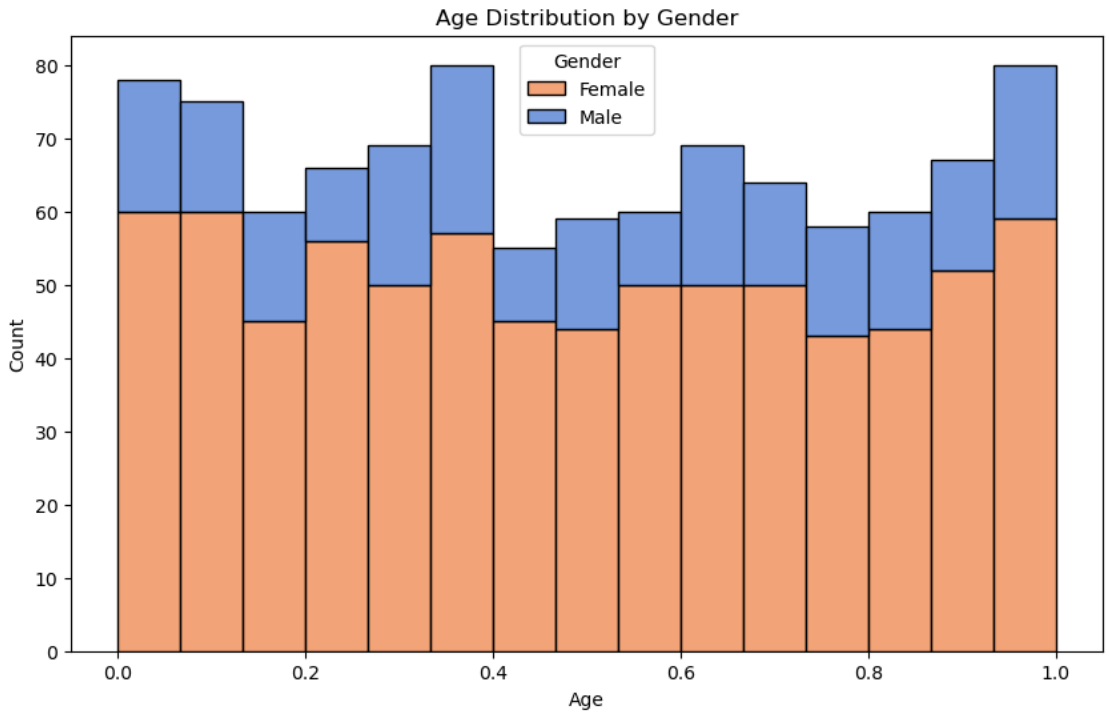

```
# What percentage of patients in the dataset have heart disease (target = 1)?
percentage_heart_disease = (df['target'].sum() / len(df)) * 100
print(f"Percentage of Patients with Heart Disease: {percentage_heart_disease:.2f}%")
```

```
Percentage of Patients with Heart Disease: 58.00%
```

Bivariate & Multivariate Analysis

1. Age Distribution by Gender

```
# Age Distribution by Gender:
plt.figure(figsize=(10, 6))
sns.histplot(x='age', hue='gender', data=df, palette='muted', multiple='stack', bins=15)
plt.title('Age Distribution by Gender')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Gender', labels=['Female', 'Male'])
plt.show()
```



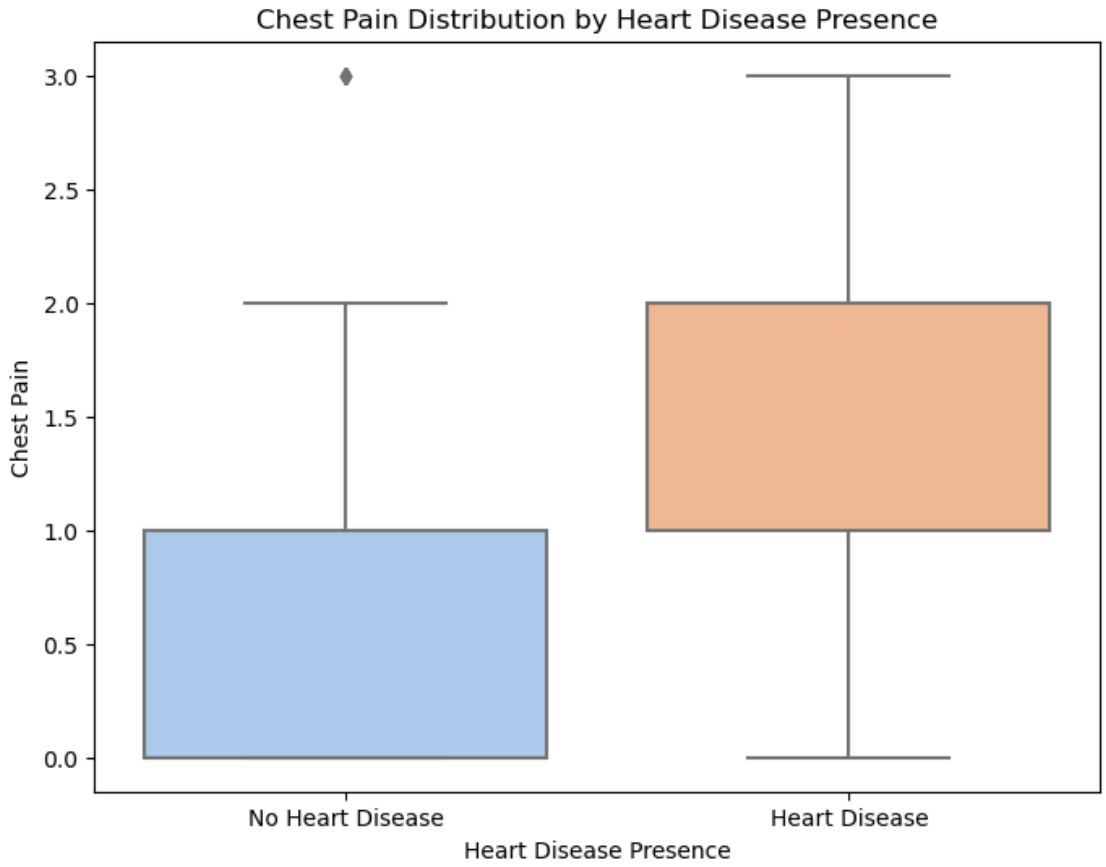
2. Correlation between age and maximum heart rate

```
correlation_age_maxheartrate = df['age'].corr(df['maxheartrate'])
print(f"Correlation between Age and Maximum Heart Rate: {correlation_age_maxheartrate:.2f}")
```

```
Correlation between Age and Maximum Heart Rate: -0.04
```

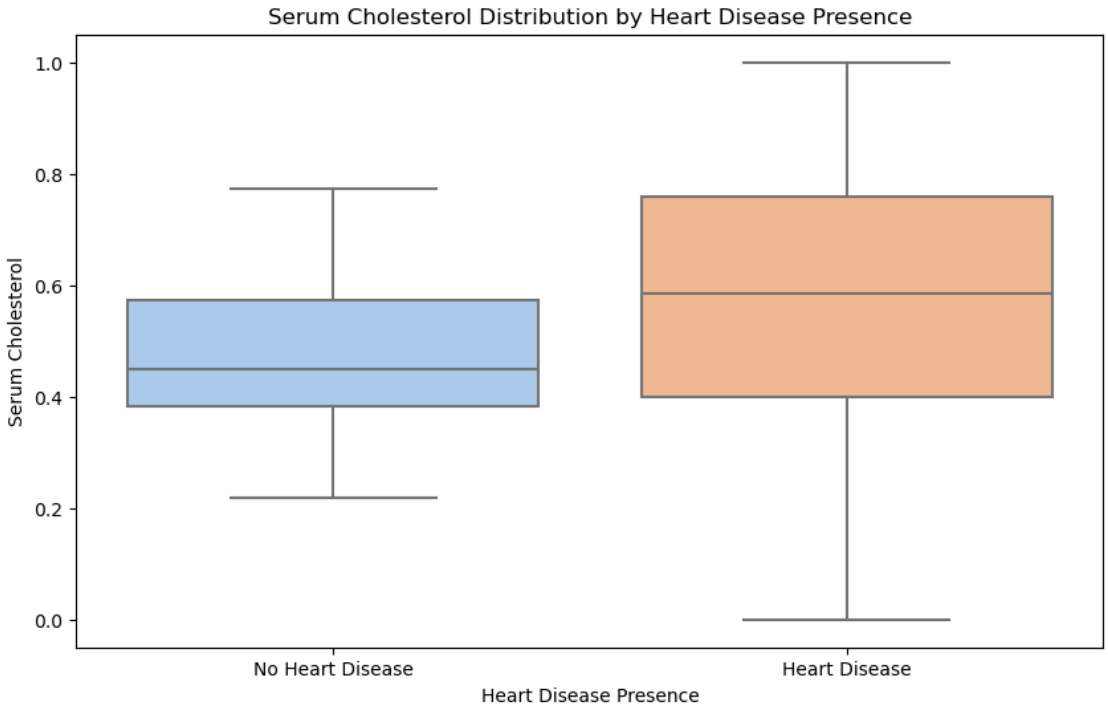
3. Relationship between chest pain type and the presence of heart disease

```
# box plot for chest pain type vs heart disease
plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='chestpain', data=df, palette='pastel')
plt.title('Chest Pain Distribution by Heart Disease Presence')
plt.xlabel('Heart Disease Presence')
plt.ylabel('Chest Pain')
plt.xticks(ticks=[0, 1], labels=['No Heart Disease', 'Heart Disease'])
plt.show()
```



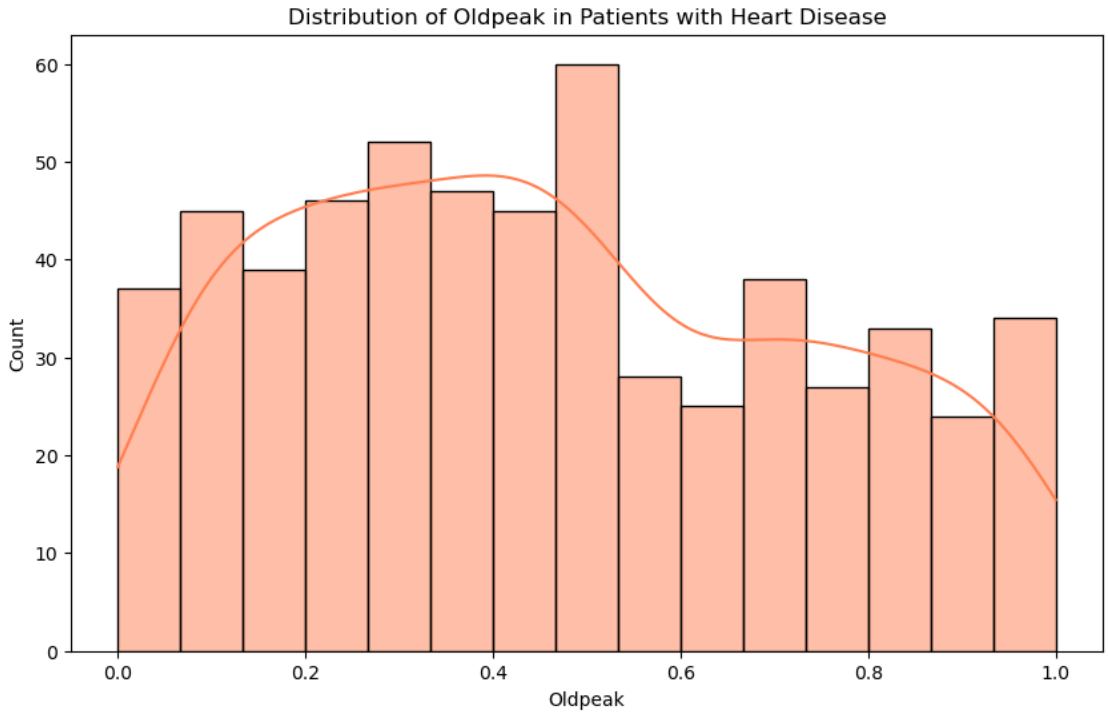
4. How does serum cholesterol differ between patients with and without heart disease?

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='target', y='serumcholestrol', data=df, palette='pastel')
plt.title('Serum Cholesterol Distribution by Heart Disease Presence')
plt.xlabel('Heart Disease Presence')
plt.ylabel('Serum Cholesterol')
plt.xticks(ticks=[0, 1], labels=['No Heart Disease', 'Heart Disease'])
plt.show()
```



5. What is the distribution of oldpeak values for patients with heart disease?

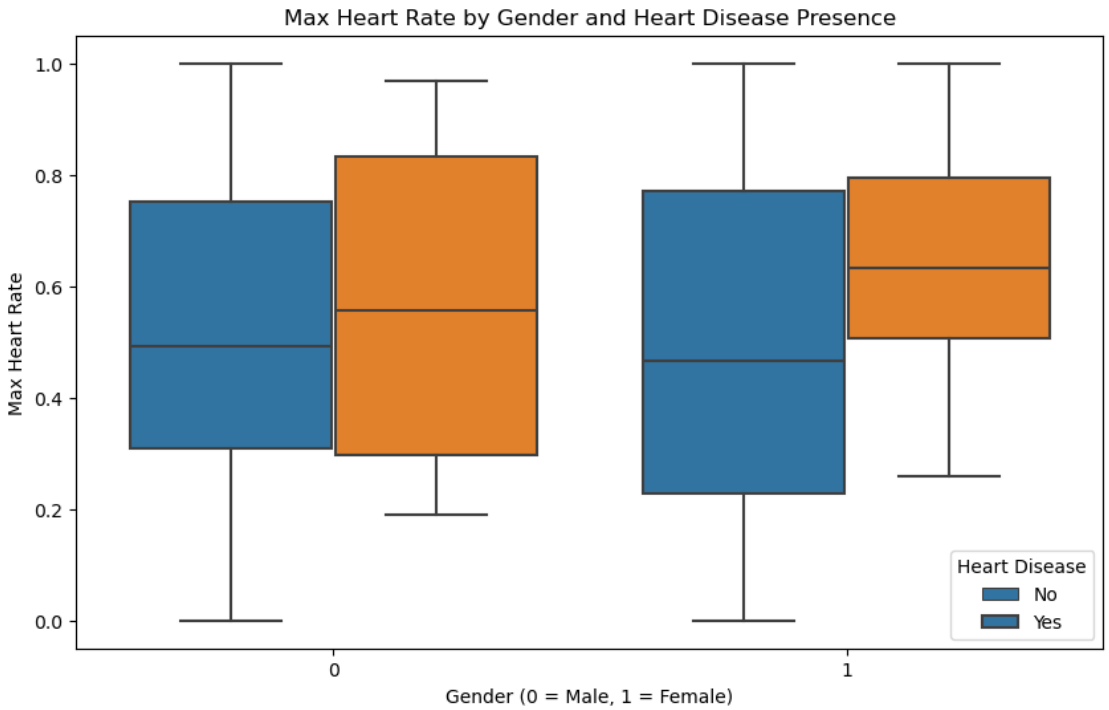
```
plt.figure(figsize=(10, 6))
sns.histplot(x='oldpeak', data=df[df['target'] == 1], bins=15, kde=True, color='coral')
plt.title('Distribution of Oldpeak in Patients with Heart Disease')
plt.xlabel('Oldpeak')
plt.ylabel('Count')
plt.show()
```



6. Max Heart Rate by Gender and Heart Disease Presence

```
# Ensure 'gender' and 'target' (Heart Disease) are categorical for analysis
df['gender'] = df['gender'].astype('category')
df['target'] = df['target'].astype('category')

# Group by Gender and Heart Disease presence and calculate mean, median, and max heart rate statistics
max_hr_stats = df.groupby(['gender', 'target'])['maxheartrate'].describe()
print(max_hr_stats)

# Boxplot to visualize Max Heart Rate by Gender and Heart Disease presence
plt.figure(figsize=(10, 6))
sns.boxplot(x='gender', y='maxheartrate', hue='target', data=df)
plt.title('Max Heart Rate by Gender and Heart Disease Presence')
plt.xlabel('Gender (0 = Male, 1 = Female)')
plt.ylabel('Max Heart Rate')
plt.legend(title='Heart Disease', labels=['No', 'Yes'])
plt.show()
```

```
                count      mean       std       min       25%       50%  \
gender target
0      0        102.0  0.515716  0.282708  0.000000  0.309160  0.492366
       1        133.0  0.555300  0.264015  0.190840  0.297710  0.557252
1      0        318.0  0.493015  0.305158  0.000000  0.229008  0.465649
       1        447.0  0.638233  0.195029  0.259542  0.507634  0.633588

                    75%       max
gender target
0      0       0.751908  1.000000
       1       0.832061  0.969466
1      0       0.770992  1.000000
       1       0.793893  1.000000
```

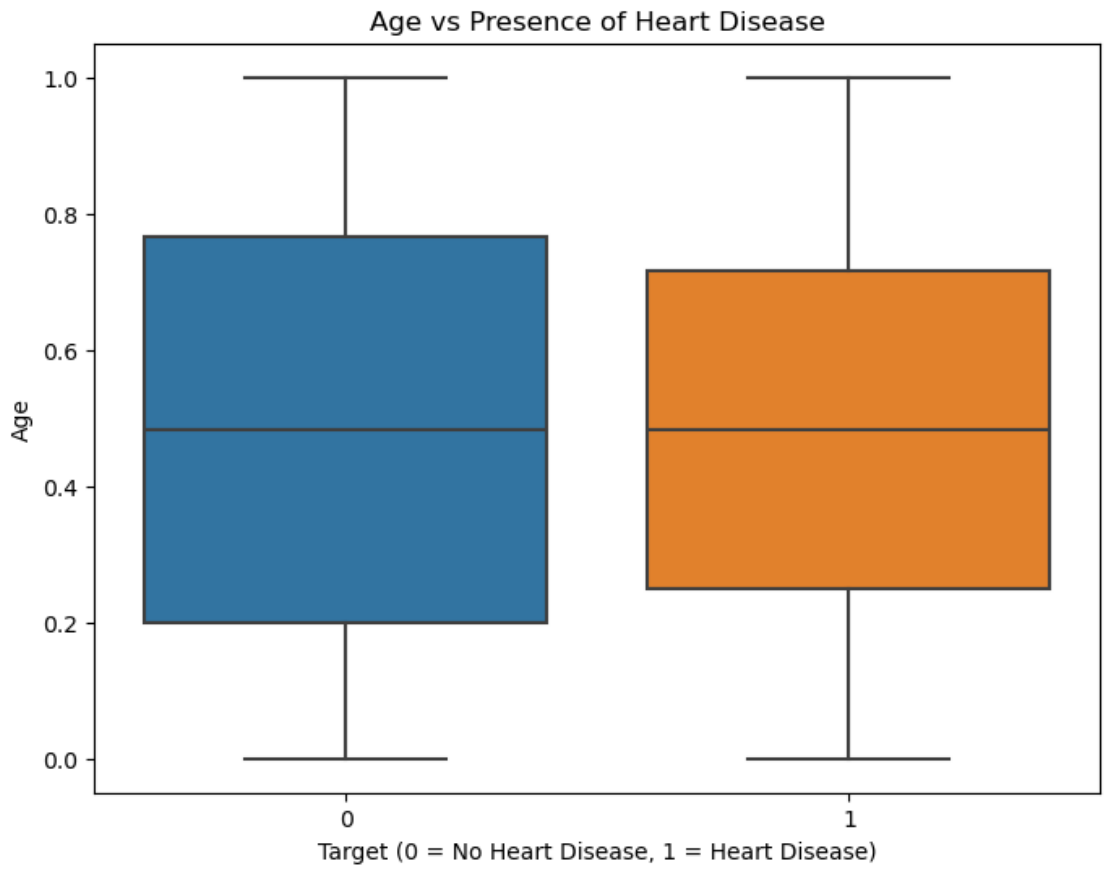


7. Age vs Target (Heart Disease Presence)

```
# Calculate summary statistics for Age based on Heart Disease (Target)
age_stats = df.groupby('target')['age'].describe()
print(age_stats)
```

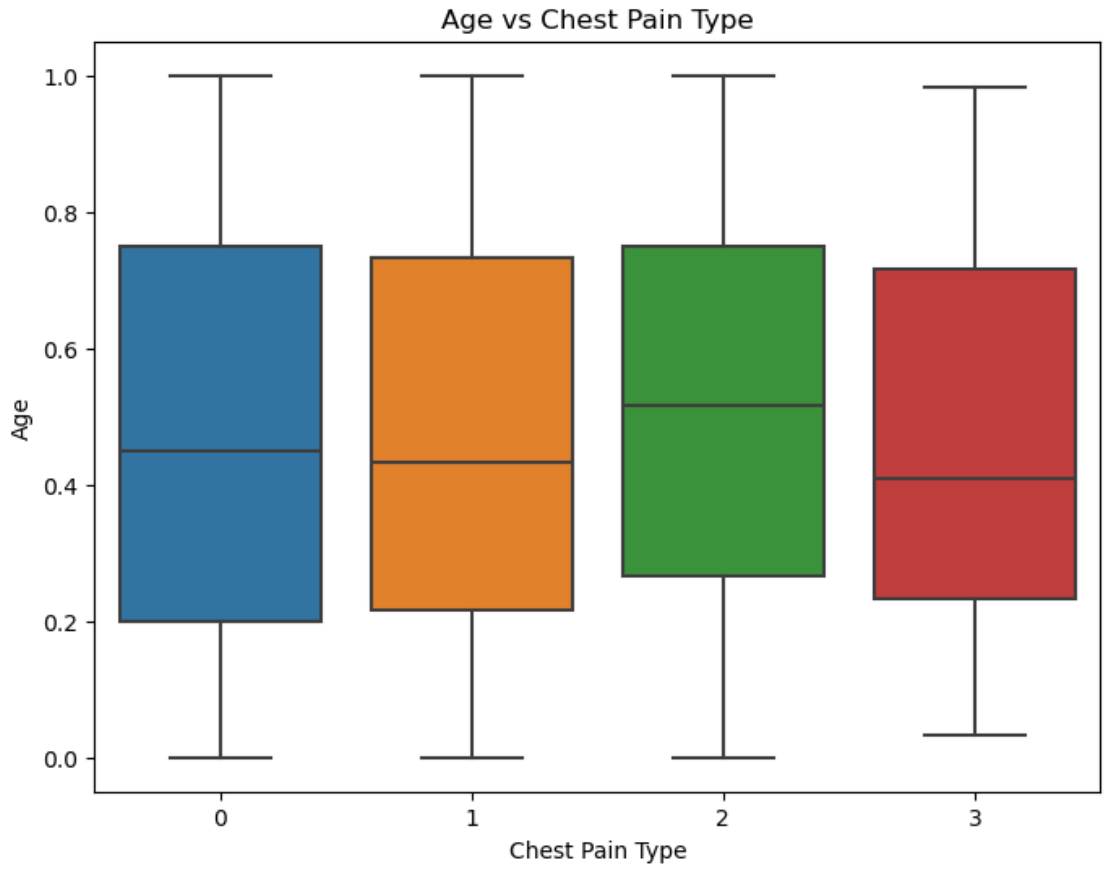
```
        count      mean       std  min   25%       50%       75%  max
target
0       420.0  0.484444  0.311743  0.0  0.20  0.483333  0.766667  1.0
1       580.0  0.489483  0.287439  0.0  0.25  0.483333  0.716667  1.0
```

```
# Boxplot to compare Age distribution between Heart Disease presence (0 = No, 1 = Yes)
plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='age', data=df)
plt.title('Age vs Presence of Heart Disease')
plt.xlabel('Target (0 = No Heart Disease, 1 = Heart Disease)')
plt.ylabel('Age')
plt.show()
```



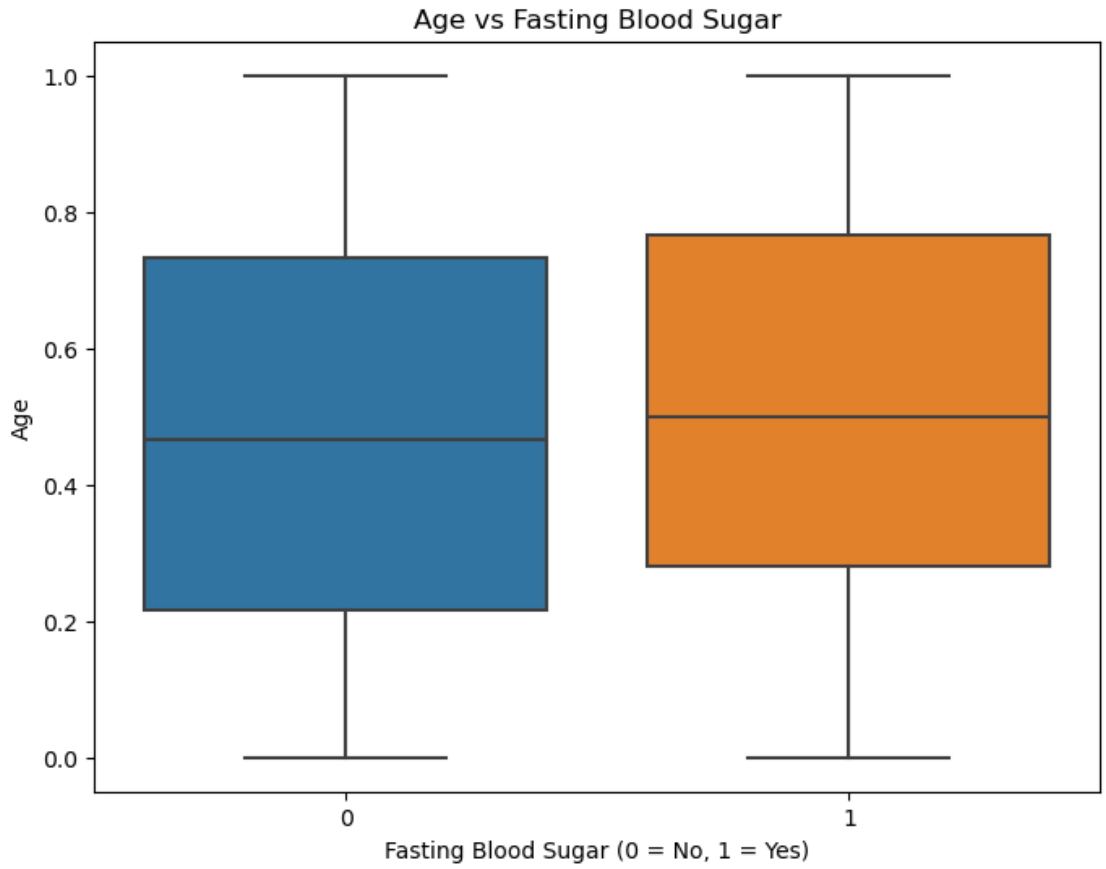
8. Age vs Chest Pain Type

```
# Boxplot to show the relationship between Age and Chest Pain Type
plt.figure(figsize=(8, 6))
sns.boxplot(x='chestpain', y='age', data=df)
plt.title('Age vs Chest Pain Type')
plt.xlabel('Chest Pain Type')
plt.ylabel('Age')
plt.show()
```



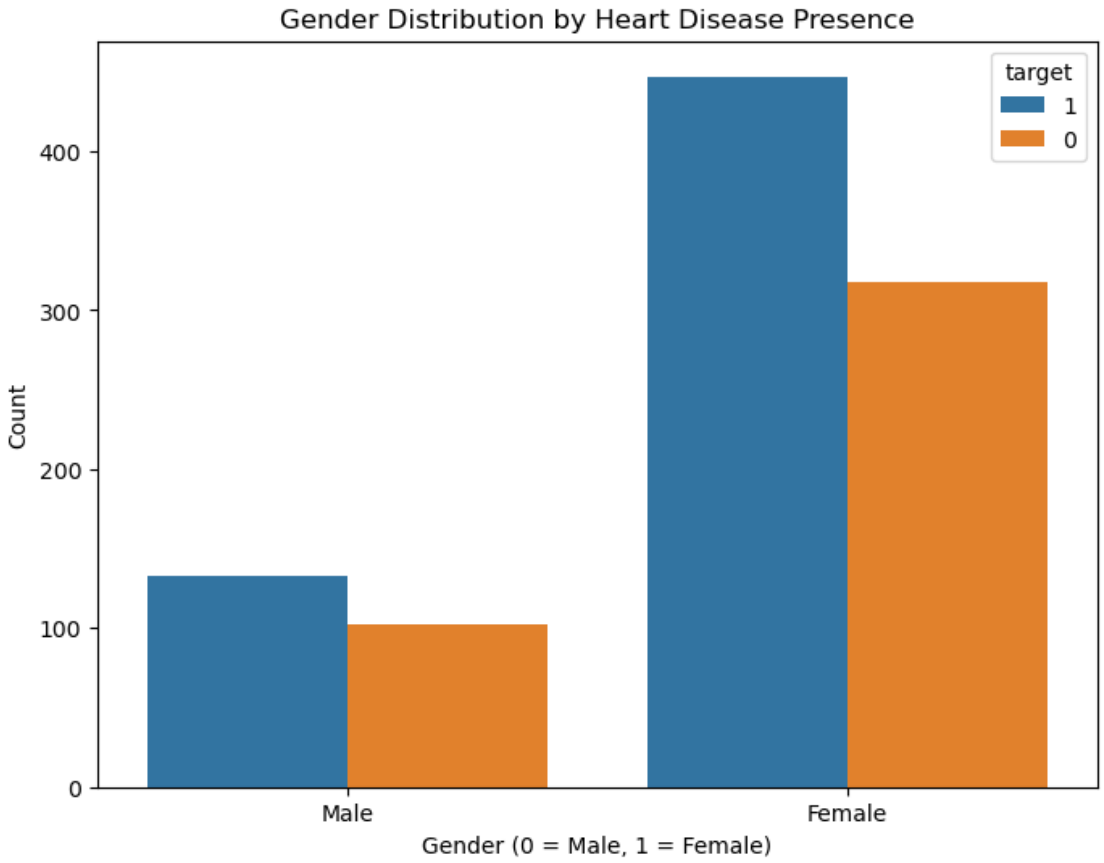
9. Age vs Fasting Blood Sugar

```
# Boxplot to show the relationship between Age and Fasting Blood Sugar
plt.figure(figsize=(8, 6))
sns.boxplot(x='fastingbloodsugar', y='age', data=df)
plt.title('Age vs Fasting Blood Sugar')
plt.xlabel('Fasting Blood Sugar (0 = No, 1 = Yes)')
plt.ylabel('Age')
plt.show()
```



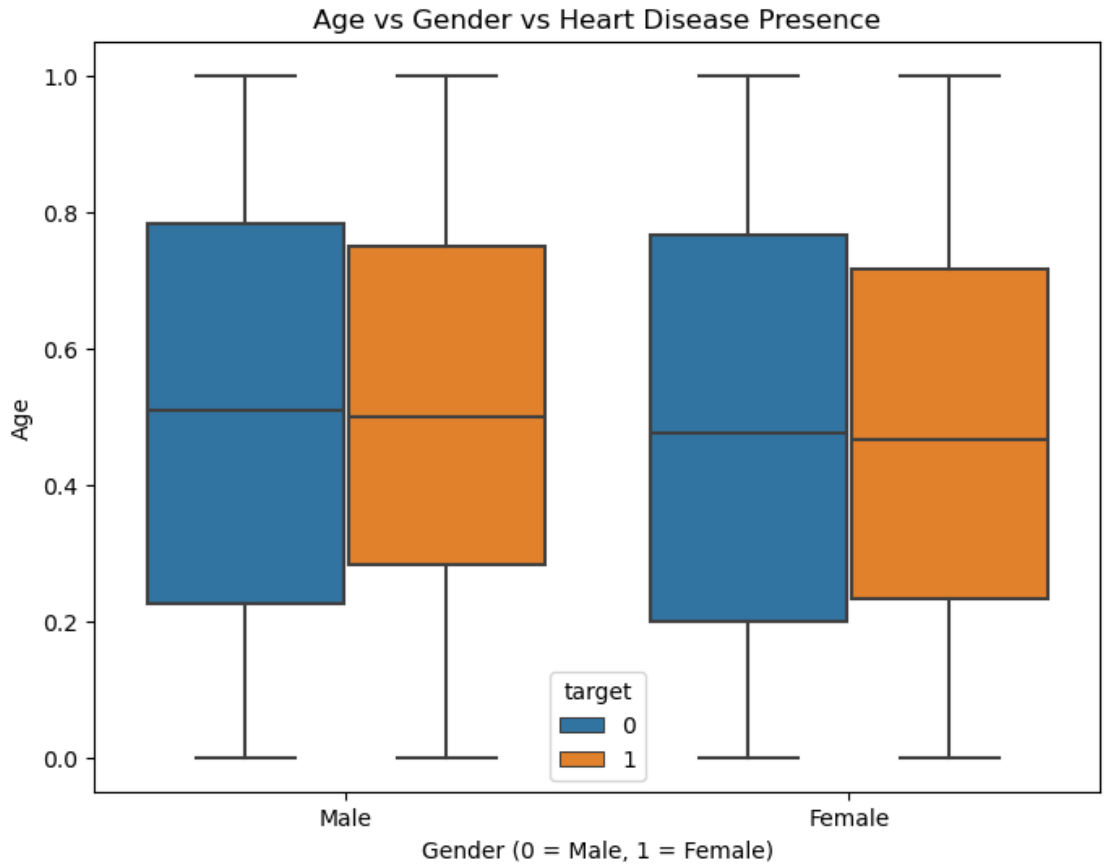
10. Gender Distribution for Heart Disease Presence

```
# convert to string if you'd like to see '0' and '1' as labels
df1 = df.copy()
df1['target'] = df1['target'].astype(str)

# Countplot to show the relationship between Gender and Heart Disease Presence
plt.figure(figsize=(8, 6))
sns.countplot(x='gender', hue='target', data=df1)
plt.title('Gender Distribution by Heart Disease Presence')
plt.xlabel('Gender (0 = Male, 1 = Female)')
plt.ylabel('Count')
plt.xticks([0, 1], ['Male', 'Female'])
plt.show()
```



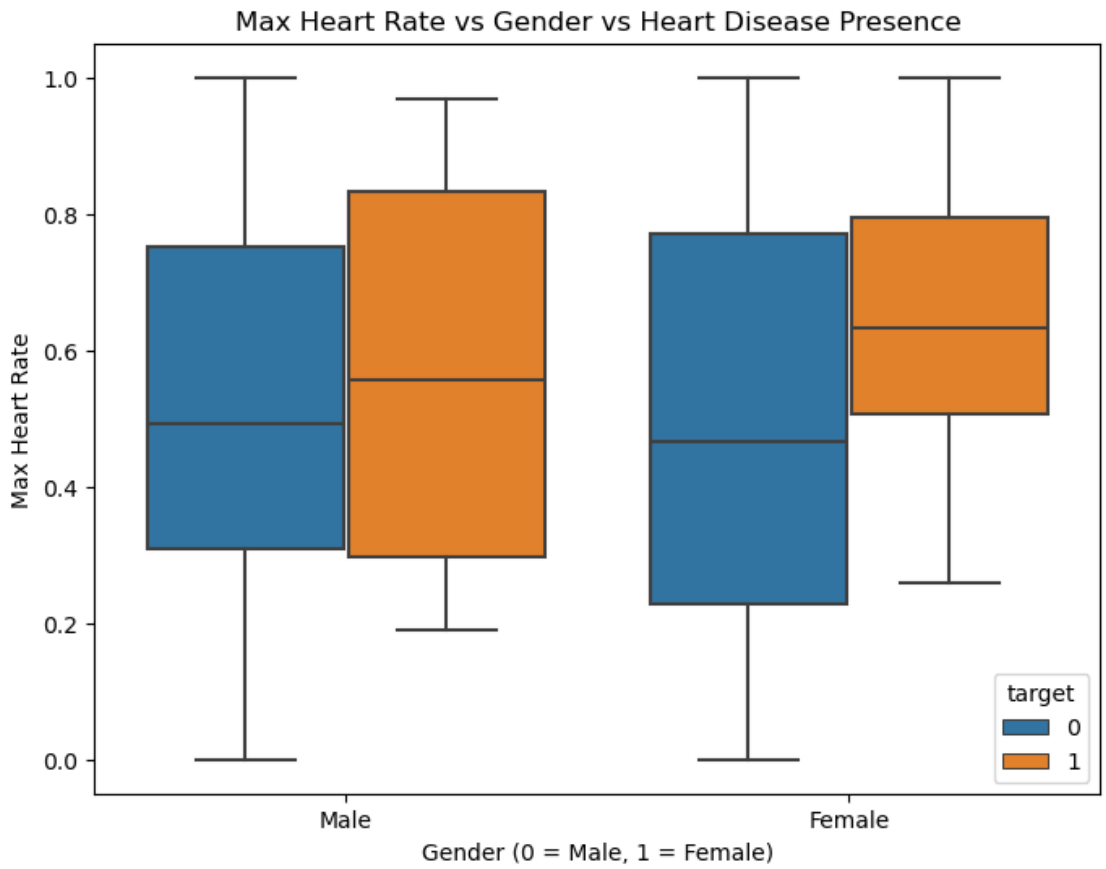
11. Age vs Gender and Heart Disease Presence

```
# Boxplot to show the relationship between Age, Gender, and Heart Disease Presence
plt.figure(figsize=(8, 6))
sns.boxplot(x='gender', y='age', hue='target', data=df)
plt.title('Age vs Gender vs Heart Disease Presence')
plt.xlabel('Gender (0 = Male, 1 = Female)')
plt.ylabel('Age')
plt.xticks([0, 1], ['Male', 'Female'])
plt.show()
```



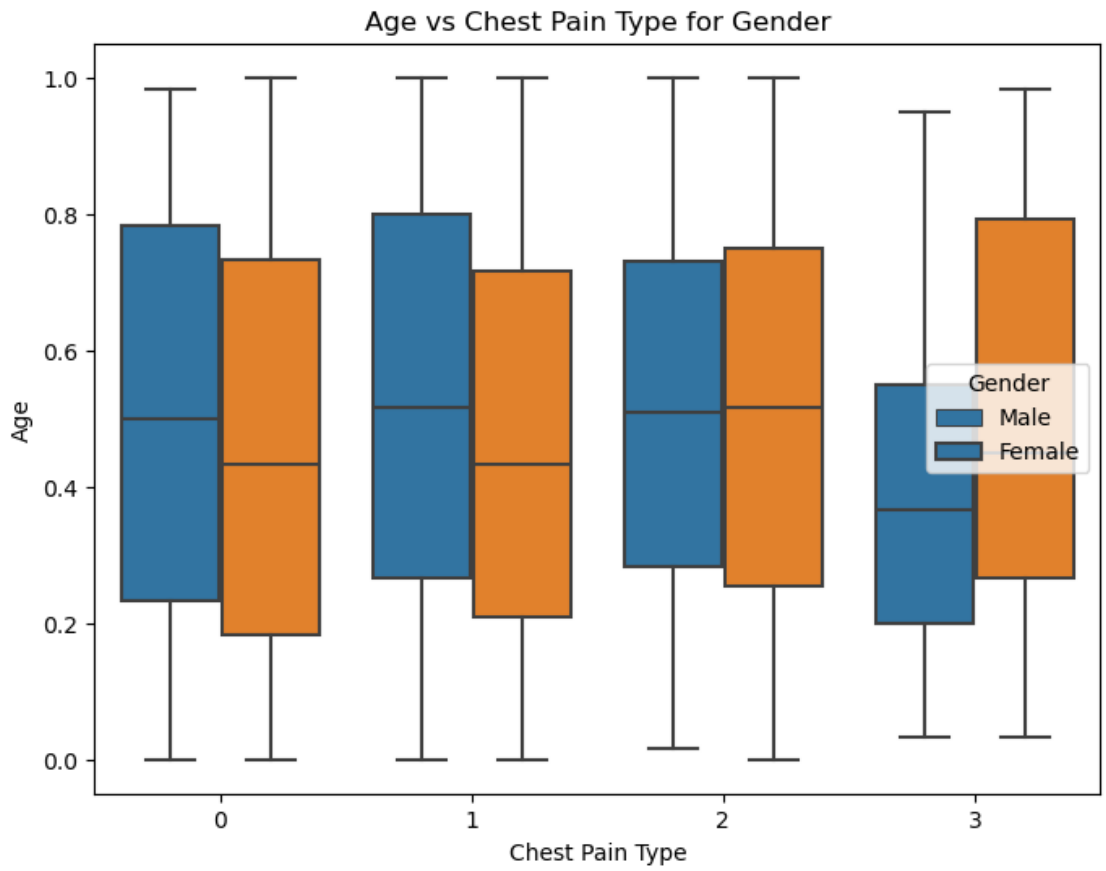
12. Max Heart Rate by Gender and Heart Disease Presence

```
# Boxplot to show the relationship between Max Heart Rate, Gender, and Heart Disease
plt.figure(figsize=(8, 6))
sns.boxplot(x='gender', y='maxheartrate', hue='target', data=df)
plt.title('Max Heart Rate vs Gender vs Heart Disease Presence')
plt.xlabel('Gender (0 = Male, 1 = Female)')
plt.ylabel('Max Heart Rate')
plt.xticks([0, 1], ['Male', 'Female'])
plt.show()
```



13. Age vs Gender Distribution for Different Chest Pain Types

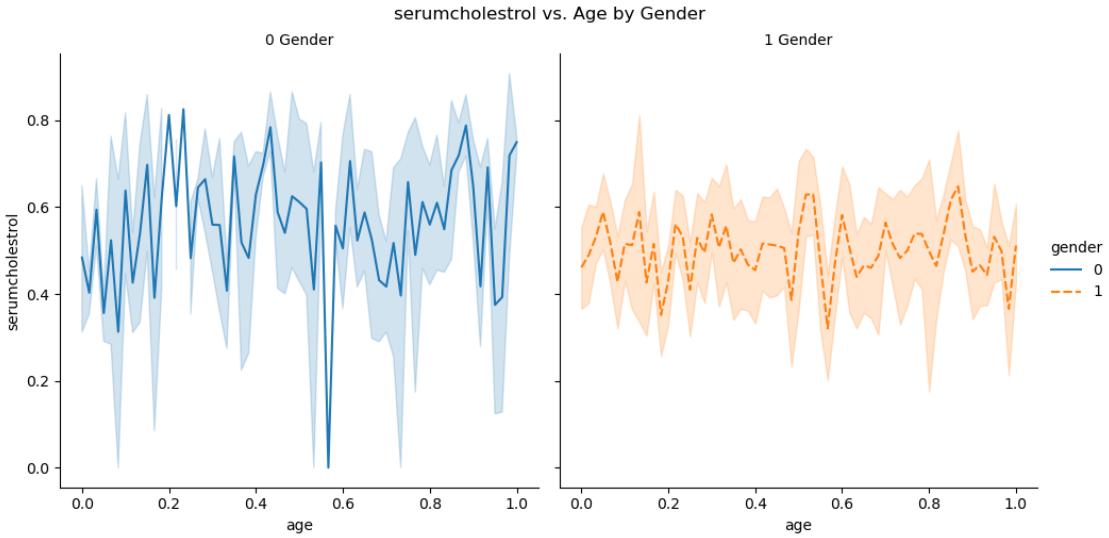
```
# Boxplot to show Age vs Chest Pain Type for Males and Females
plt.figure(figsize=(8, 6))
sns.boxplot(x='chestpain', y='age', hue='gender', data=df)
plt.title('Age vs Chest Pain Type for Gender')
plt.xlabel('Chest Pain Type')
plt.ylabel('Age')
plt.legend(title='Gender', labels=['Male', 'Female'])
plt.show()
```

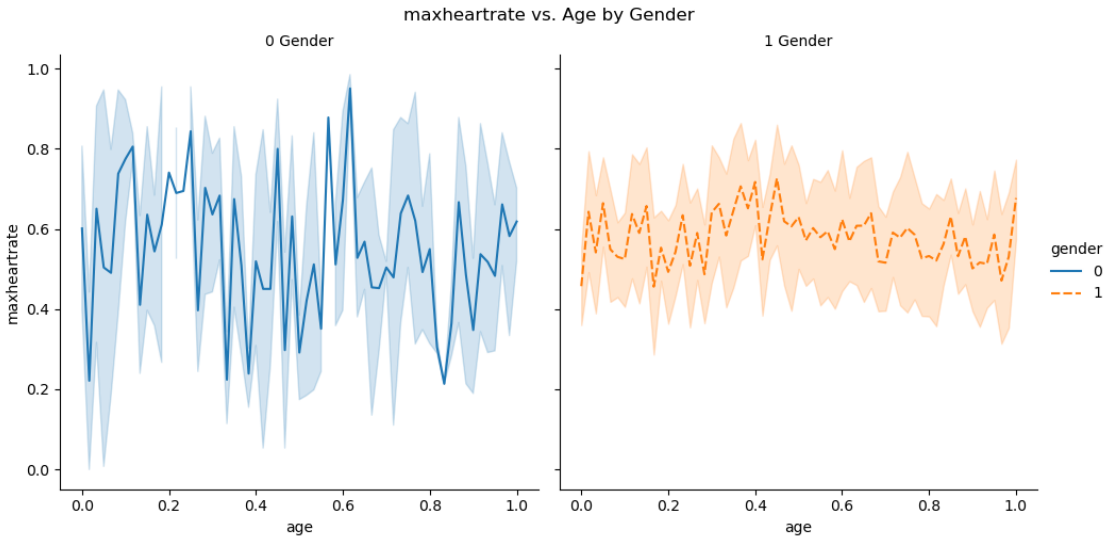


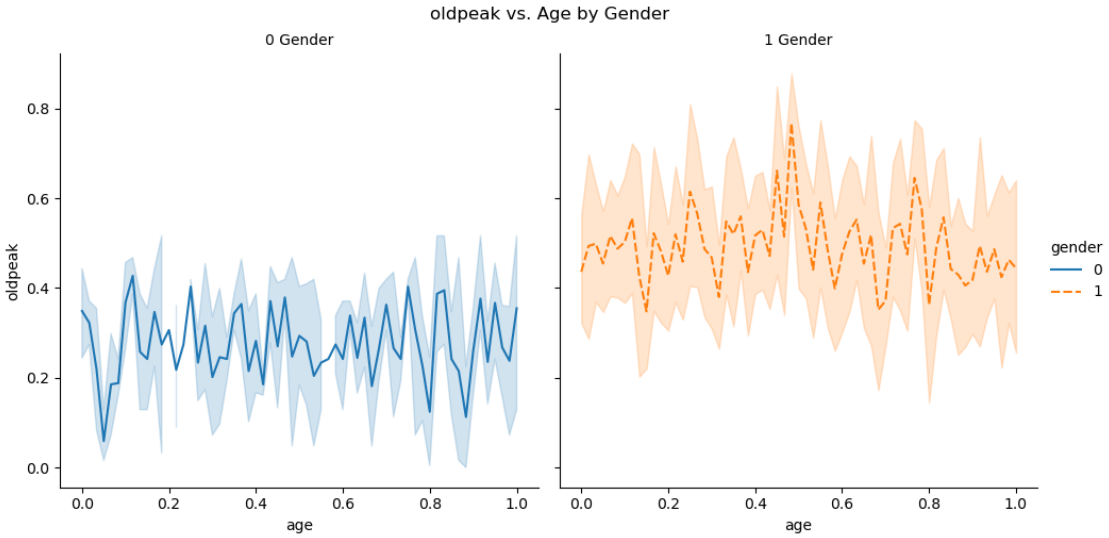
14. Visualization for Resting blood pressure, Serum cholesterol, Maximum heart rate achieved and Oldpeak (ST)

```
numeric_cols = ['restingBP', 'serumcholestrol', 'maxheartrate', 'oldpeak']
gender = {0: 'Male', 1: 'Female'}  # Gender dictionary (needed for mapping)

for i in numeric_cols:
    plt.figure(figsize=(20, 12))

    # Using sns.relplot with 'gender' as hue
    plot = sns.relplot(data=df, x='age', y=i, kind='line', hue='gender', style='gender', col='gender')

    # Set the title for each subplot
    plot.set_titles("{col_name} Gender")

    # Add a main title outside the plot
    plt.suptitle(f"{i} vs. Age by Gender", y=1.02)

    # Show the plot
    plt.show()
```


restingBP vs. Age by Gender
0 Gender
1 Gender
restingBP
age
gender
0
1


serumcholestrol vs. Age by Gender
0 Gender
1 Gender
serumcholestrol
age
gender
0
1





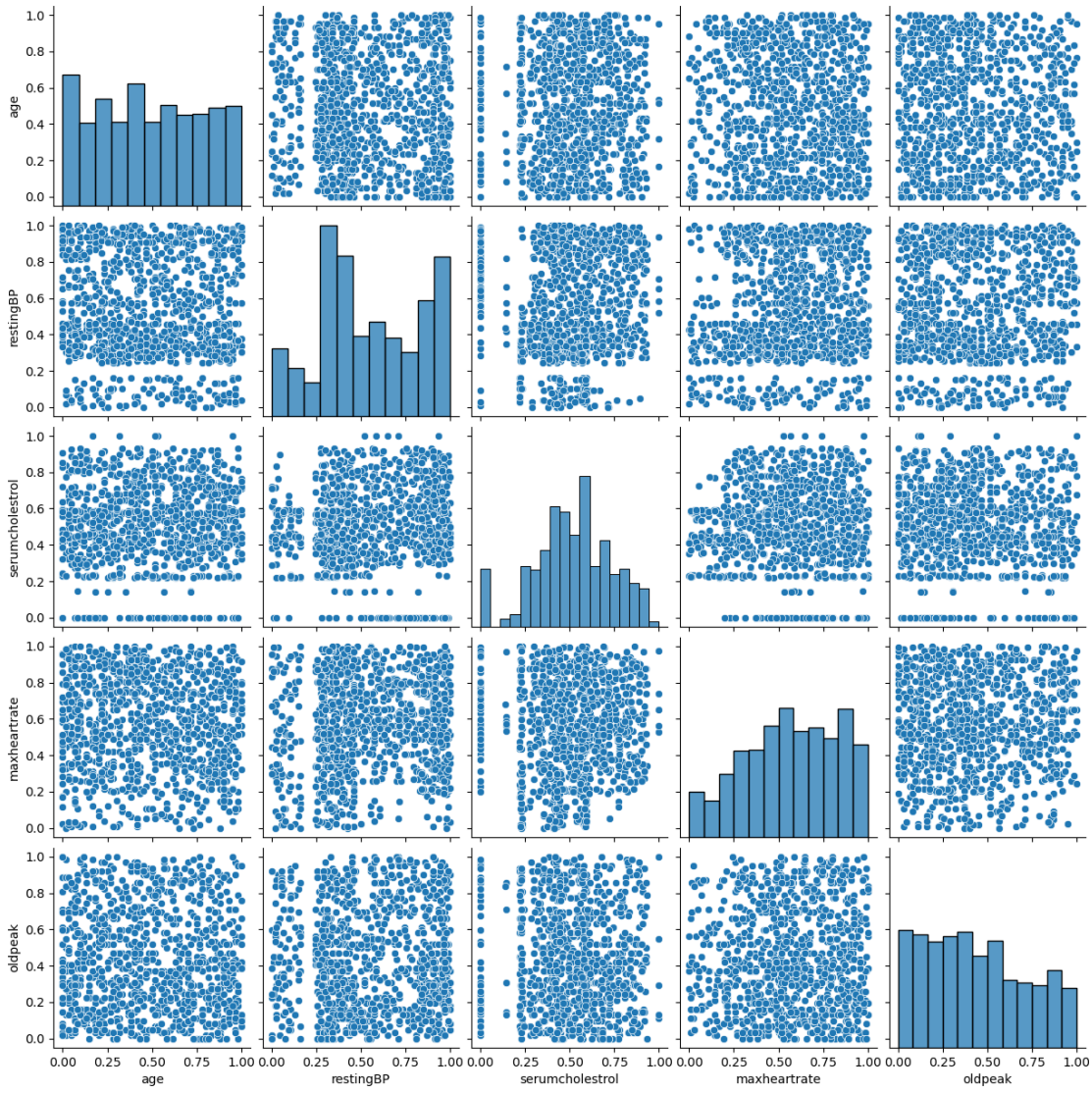

```
df['gender'].value_counts()
```

```
gender
1    765
0    235
Name: count, dtype: int64
```

15. Pair Plot (Pairwise relationships)

A matrix of scatter plots for all pairs of numerical variables.

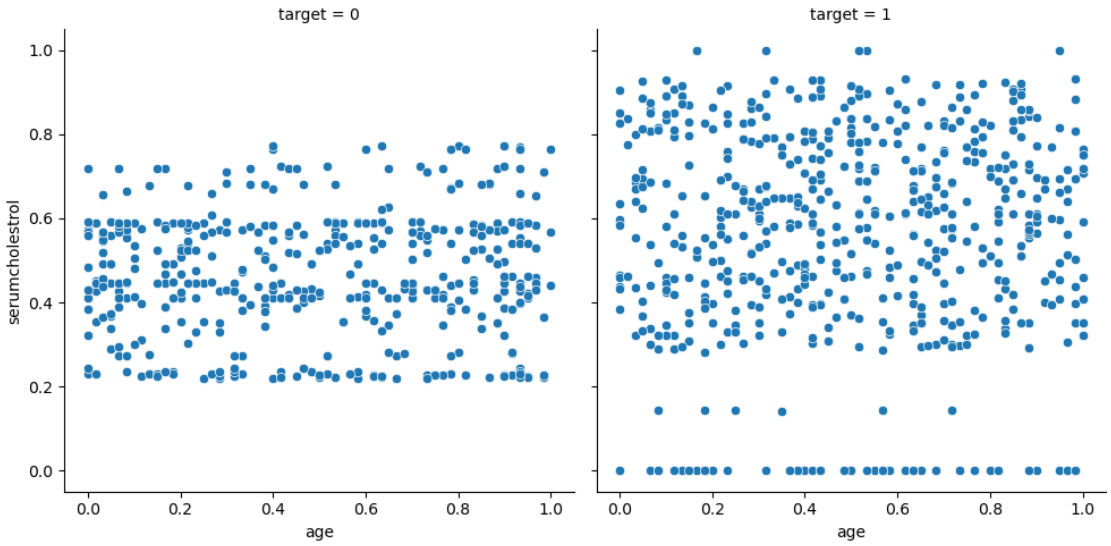
```
sns.pairplot(df[['age', 'restingBP', 'serumcholestrol', 'maxheartrate', 'oldpeak', 'target']])
plt.show()
```



16. FacetGrid (Pair of plots by a categorical variable)

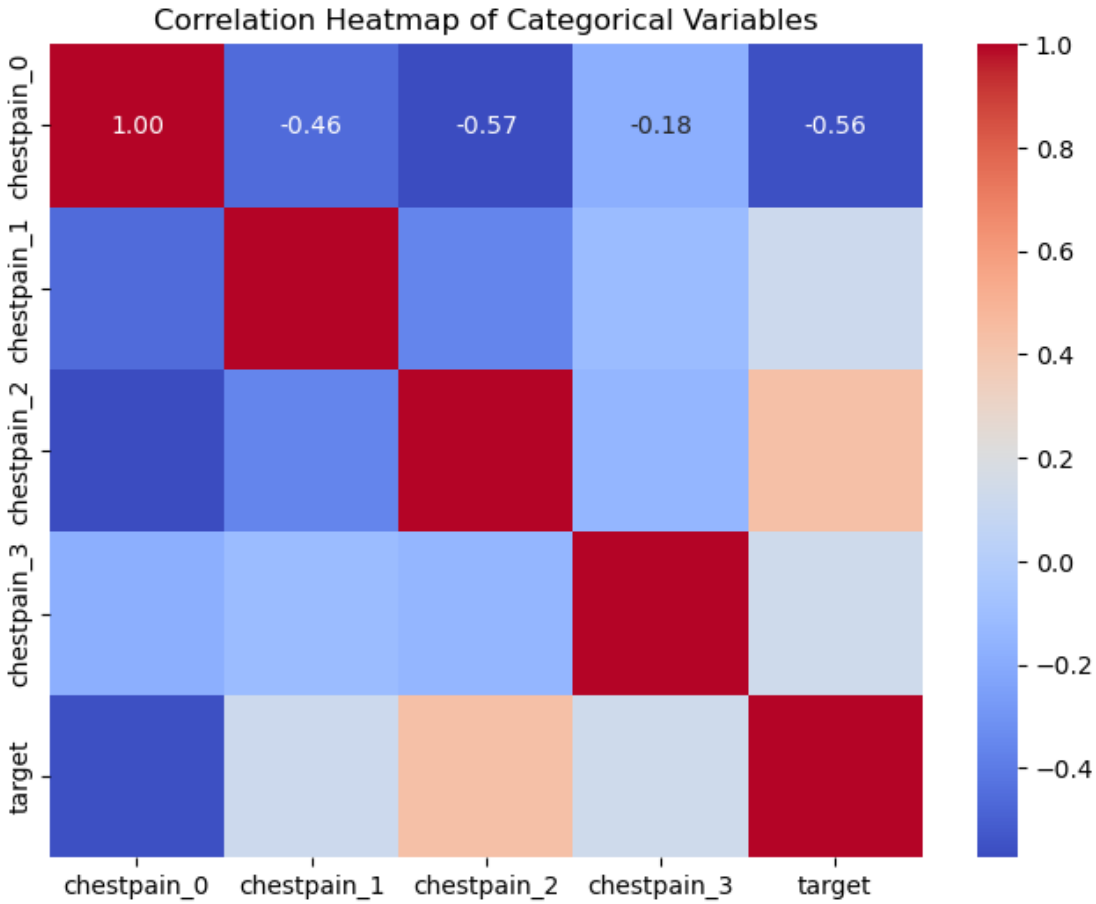
age vs. serumcholestrol, split by target

```
g = sns.FacetGrid(df, col="target", height=5)
g.map(sns.scatterplot, "age", "serumcholestrol")
plt.show()
```

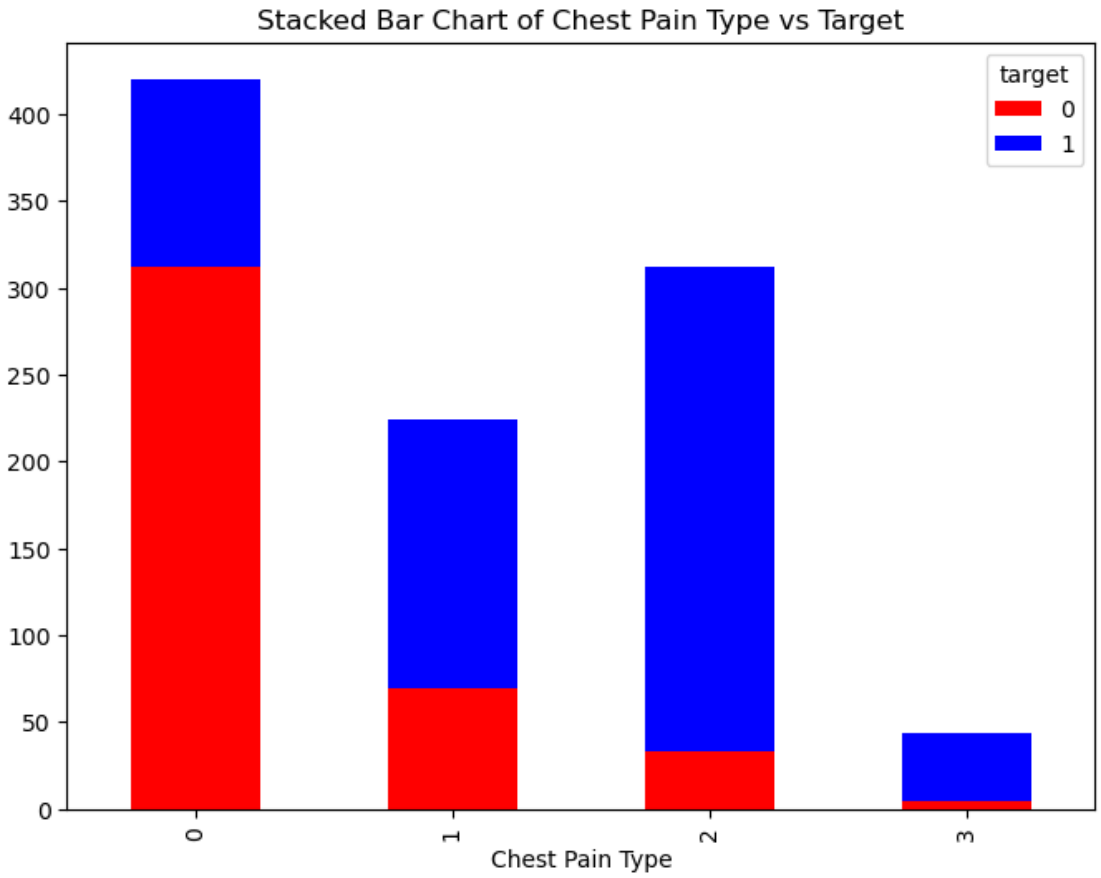


17. Heatmap of Categorical Variables

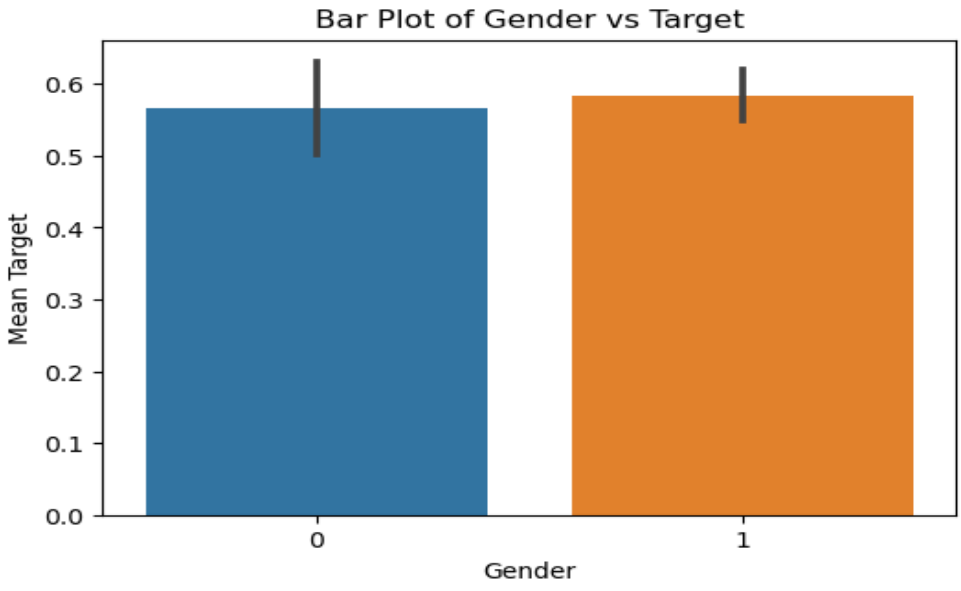
```
# correlation between categorical variables like chestpain and target
cat_corr = df[['chestpain_0', 'chestpain_1', 'chestpain_2', 'chestpain_3', 'target']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(cat_corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap of Categorical Variables')
plt.show()
```




```
# Stacked Bar Chart for chestpain types vs. target
cross_tab = pd.crosstab(df['chestpain'], df['target'])
cross_tab.plot(kind='bar', stacked=True, figsize=(8, 6), color=['red', 'blue'])
plt.title('Stacked Bar Chart of Chest Pain Type vs Target')
plt.xlabel('Chest Pain Type')
plt.ylabel
```

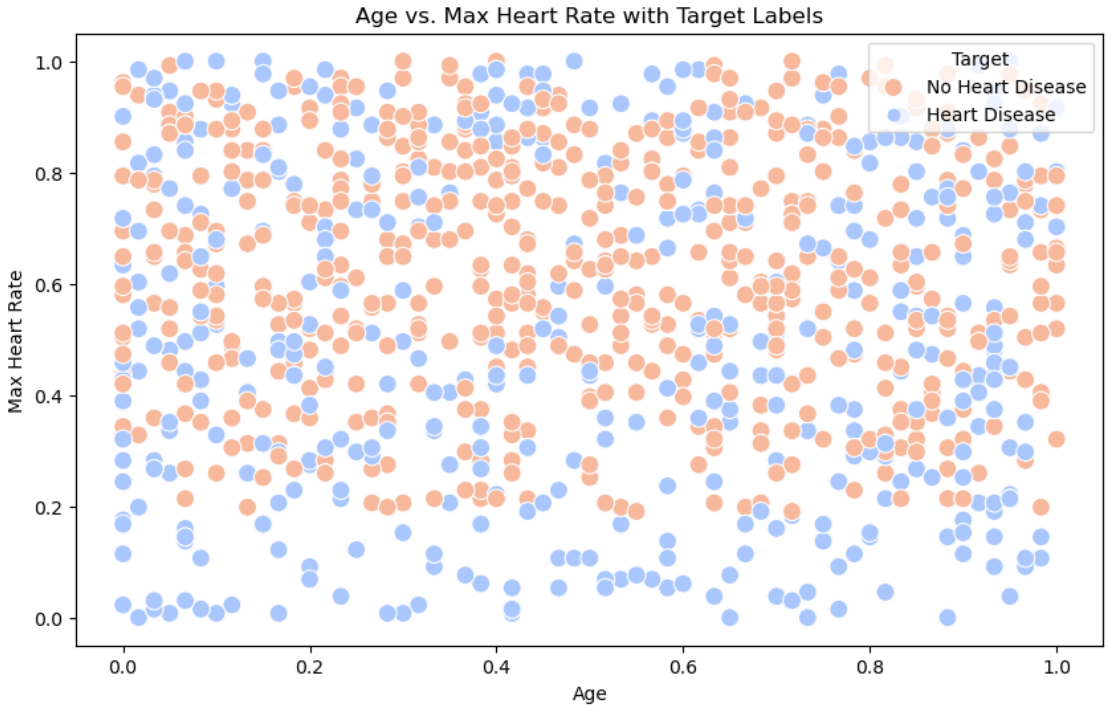



```
plt.figure(figsize=(6, 4))
sns.barplot(x='gender', y='target', data=df)
plt.title('Bar Plot of Gender vs Target')
plt.xlabel('Gender')
plt.ylabel('Mean Target')
plt.show()
```



Age vs. Max Heart Rate with Target Labels

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='maxheartrate', hue='target', data=df, palette='coolwarm', s=100)
plt.title('Age vs. Max Heart Rate with Target Labels')
plt.xlabel('Age')
plt.ylabel('Max Heart Rate')
plt.legend(title='Target', loc='upper right', labels=['No Heart Disease', 'Heart Disease'])
plt.show()
```



CHAPTER 5: MODEL DEVELOPMENT & EVALUATION

1. Split The Data Into Training And Testing Sets

```
from sklearn.model_selection import train_test_split

# Features (X) and target (y)
X = df.drop(columns=['target'])  # Drop the target variable from features
y = df['target']  # Target variable

# Split the data into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Model Selection

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Support Vector Machine (SVM)
5. K-Nearest Neighbors (KNN)
6. Gradient Boosting (XGBoost)

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Initialize models
logreg = LogisticRegression()
dtree = DecisionTreeClassifier()
rf = RandomForestClassifier()
svm = SVC()
knn = KNeighborsClassifier()
gb = GradientBoostingClassifier()

# List of models to evaluate
models = [logreg, dtree, rf, svm, knn, gb]
```

3. Train The Models

Each model will be trained on the training data

```
# Train each model on the training set
for model in models:
    model.fit(X_train, y_train)
```

4. Evaluate The Models

Now that the models are trained, we can evaluate them using various metrics such as accuracy, precision, recall, F1-score, and ROC AUC score.

We'll use cross-validation and confusion matrices to get a better understanding of the models' performance.

from sklearn.metrics **import** accuracy_score, confusion_matrix, classification_report

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Evaluate the models on the testing set
for model in models:
    y_pred = model.predict(X_test)
    print(f"Model: {model.__class__.__name__}")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
    print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
    print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
    print("\n" + "-"*50 + "\n")
```

```
Model: LogisticRegression
Accuracy: 0.9700
Confusion Matrix:
[[ 81   2]
 [  4 113]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96        83
           1       0.98      0.97      0.97       117

    accuracy                           0.97       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.97      0.97      0.97       200


--------------------------------------------------
```

```
Model: DecisionTreeClassifier
Accuracy: 0.9750
Confusion Matrix:
[[ 79   4]
 [  1 116]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.95      0.97        83
           1       0.97      0.99      0.98       117

    accuracy                           0.97       200
   macro avg       0.98      0.97      0.97       200
weighted avg       0.98      0.97      0.97       200


--------------------------------------------------
```

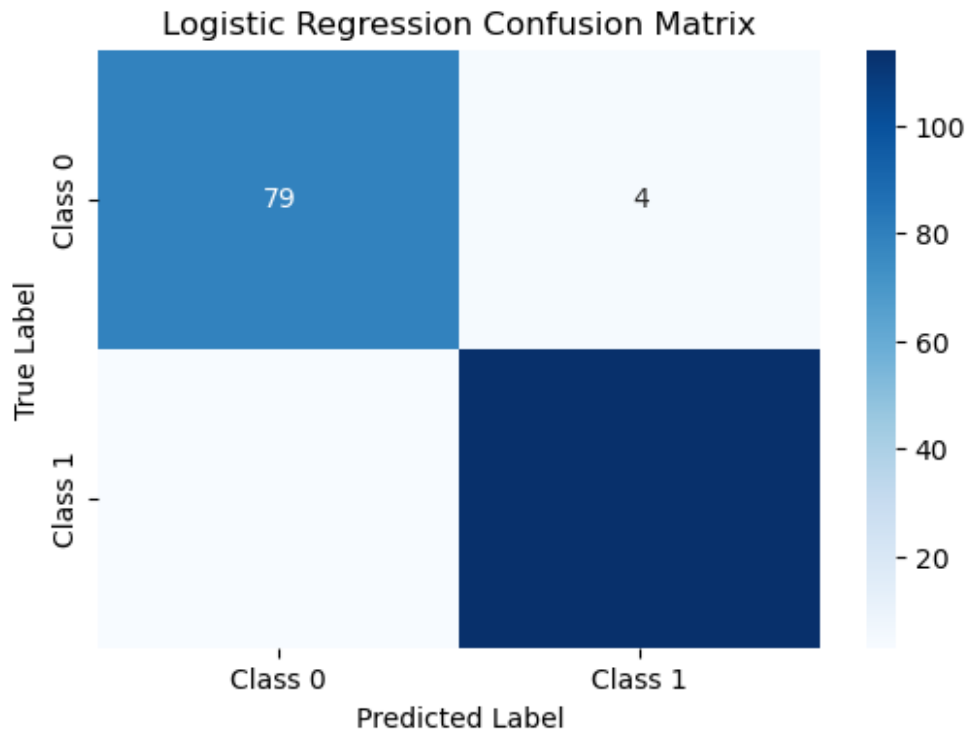
```
Model: RandomForestClassifier
Accuracy: 0.9800
Confusion Matrix:
[[ 81   2]
 [  2 115]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        83
           1       0.98      0.98      0.98       117

    accuracy                           0.98       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.98      0.98      0.98       200


--------------------------------------------------


Model: SVC
Accuracy: 0.9600
Confusion Matrix:
[[ 79   4]
 [  4 113]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        83
           1       0.97      0.97      0.97       117

    accuracy                           0.96       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.96      0.96      0.96       200


--------------------------------------------------


Model: KNeighborsClassifier
Accuracy: 0.9650
Confusion Matrix:
[[ 78   5]
 [  2 115]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.96        83
           1       0.96      0.98      0.97       117

    accuracy                           0.96       200
   macro avg       0.97      0.96      0.96       200
weighted avg       0.97      0.96      0.96       200


--------------------------------------------------
```

```
Model: GradientBoostingClassifier
Accuracy: 0.9950
Confusion Matrix:
[[ 82   1]
 [  0 117]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99        83
           1       0.99      1.00      1.00       117

    accuracy                           0.99       200
   macro avg       1.00      0.99      0.99       200
weighted avg       1.00      0.99      0.99       200


--------------------------------------------------
```

5. Hyperparameter Tuning

1. Logistic Regression Hyperparameter Tuning and Evaluation

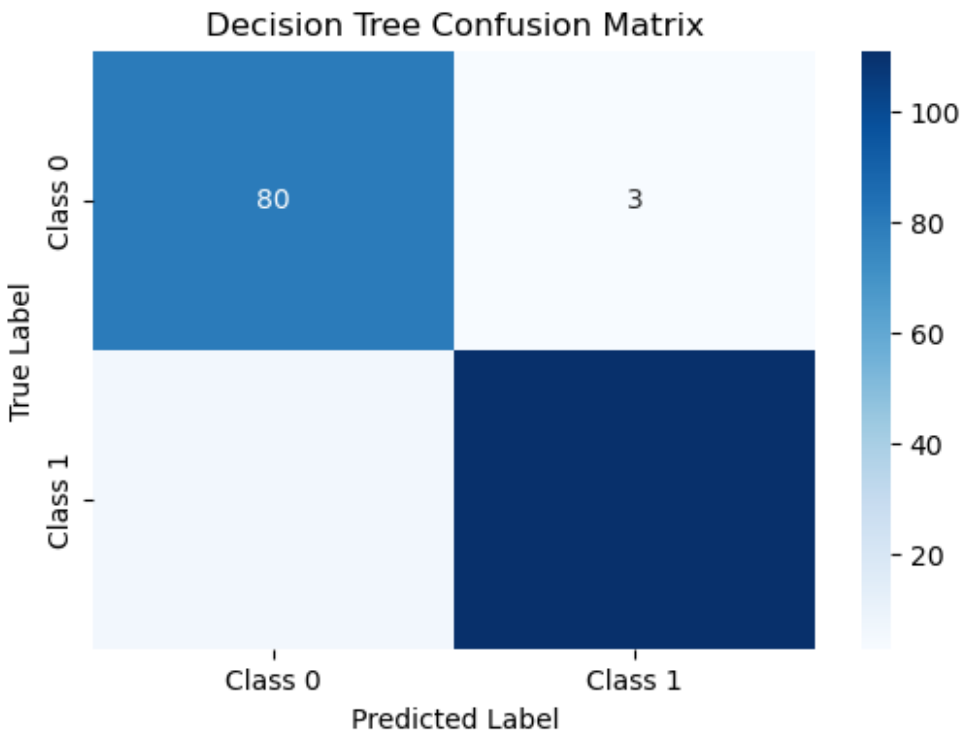
```
Best Hyperparameters for Logistic Regression: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
Best Cross-Validation Score for Logistic Regression: 0.9675
Logistic Regression Test Accuracy: 0.9650
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.96        83
           1       0.97      0.97      0.97       117

    accuracy                           0.96       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.96      0.96      0.96       200
```




```
Logistic Regression ROC-AUC Score: 0.9975
```

2. Decision Tree Hyperparameter Tuning and Evaluation

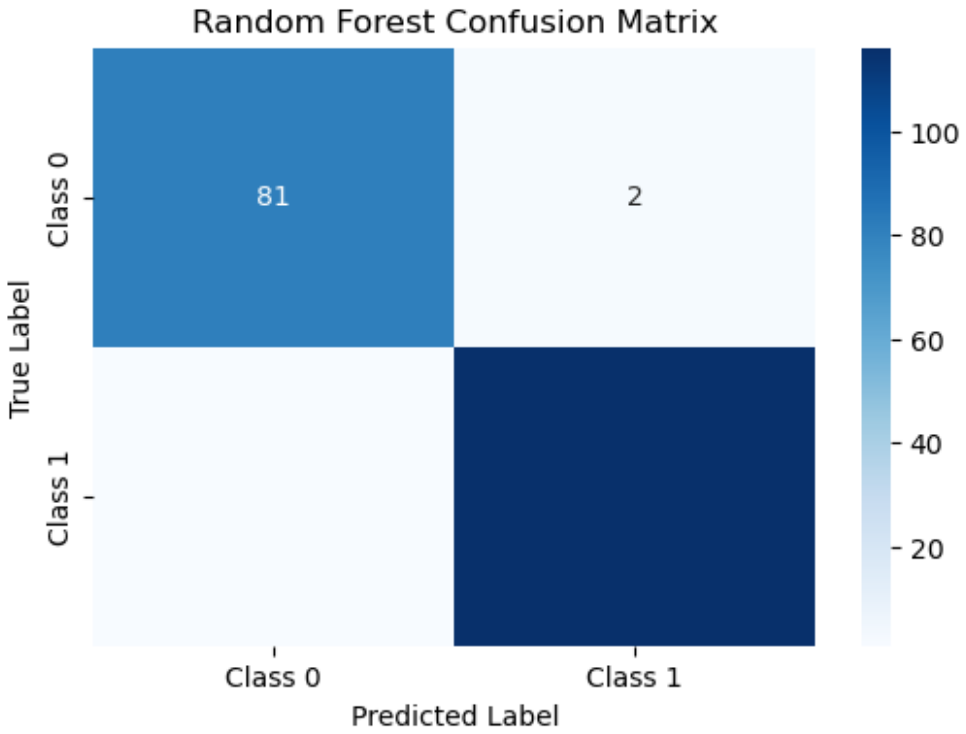
```
Best Hyperparameters for Decision Tree: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Best Cross-Validation Score for Decision Tree: 0.9762
Decision Tree Test Accuracy: 0.9550
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.95        83
           1       0.97      0.95      0.96       117

    accuracy                           0.95       200
   macro avg       0.95      0.96      0.95       200
weighted avg       0.96      0.95      0.96       200
```




```
Decision Tree ROC-AUC Score: 0.9734
```

3. Random Forest Hyperparameter Tuning and Evaluation

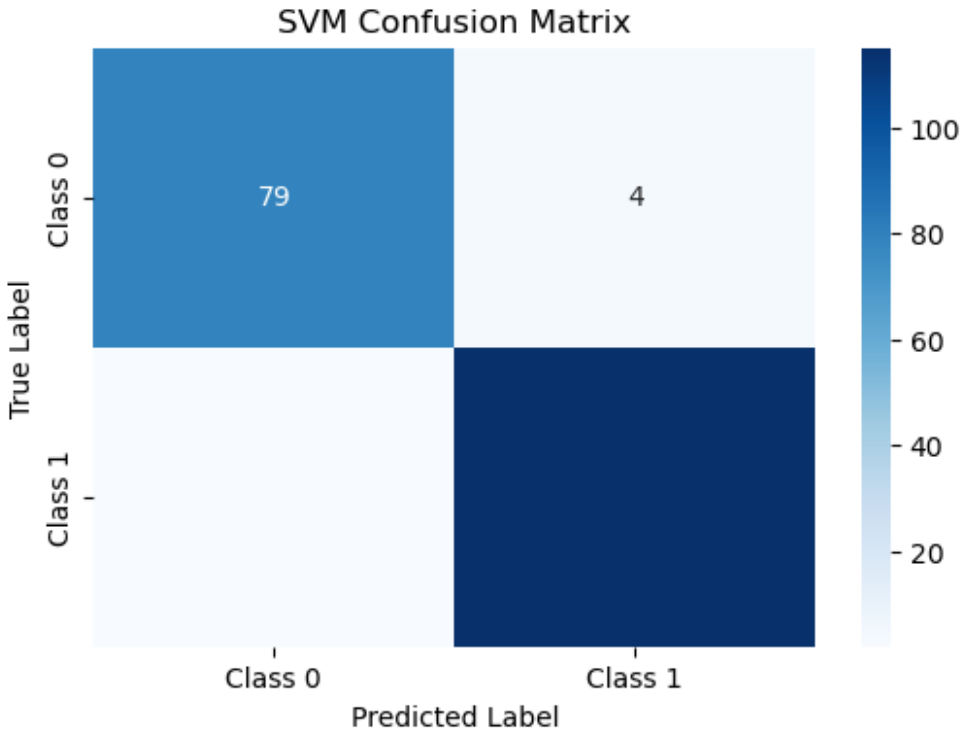
```
Best Hyperparameters for Random Forest: {'bootstrap': False, 'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}
Best Cross-Validation Score for Random Forest: 0.9800
Random Forest Test Accuracy: 0.9900
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        83
           1       0.99      0.99      0.99       117

    accuracy                           0.99       200
   macro avg       0.99      0.99      0.99       200
weighted avg       0.99      0.99      0.99       200
```




```
Random Forest ROC-AUC Score: 0.9995
```

4. Support Vector Machine (SVM) Hyperparameter Tuning and Evaluation

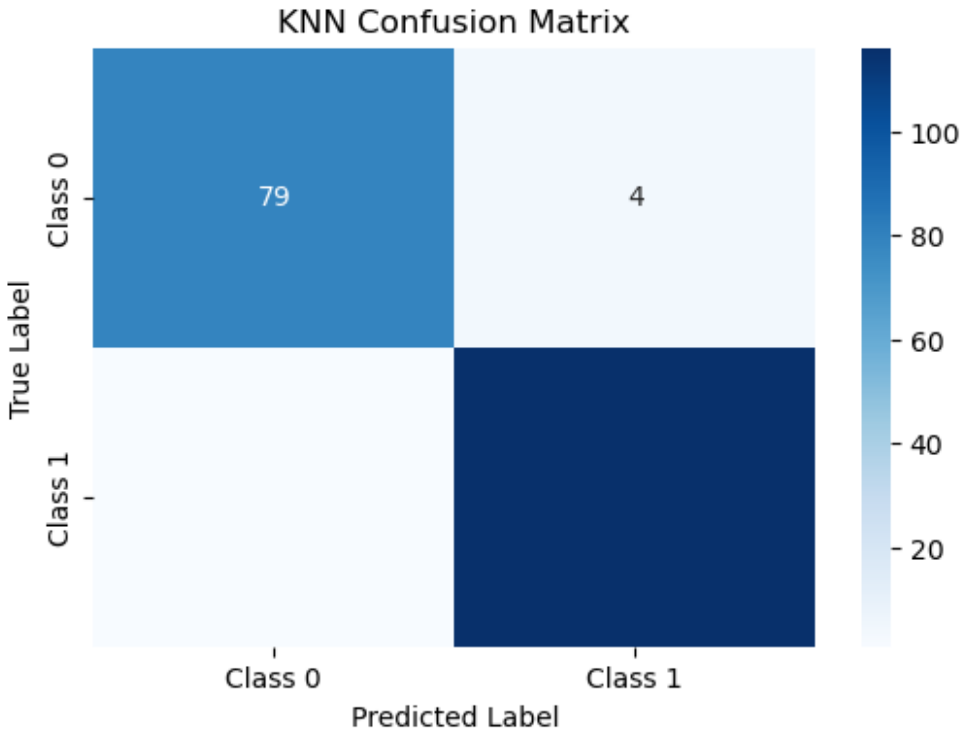
```
Best Hyperparameters for SVM: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
Best Cross-Validation Score for SVM: 0.9650
SVM Test Accuracy: 0.9700
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        83
           1       0.97      0.98      0.97       117

    accuracy                           0.97       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.97      0.97      0.97       200
```




```
SVM ROC-AUC Score: 0.9968
```

5. K-Nearest Neighbors (KNN) Hyperparameter Tuning and Evaluation

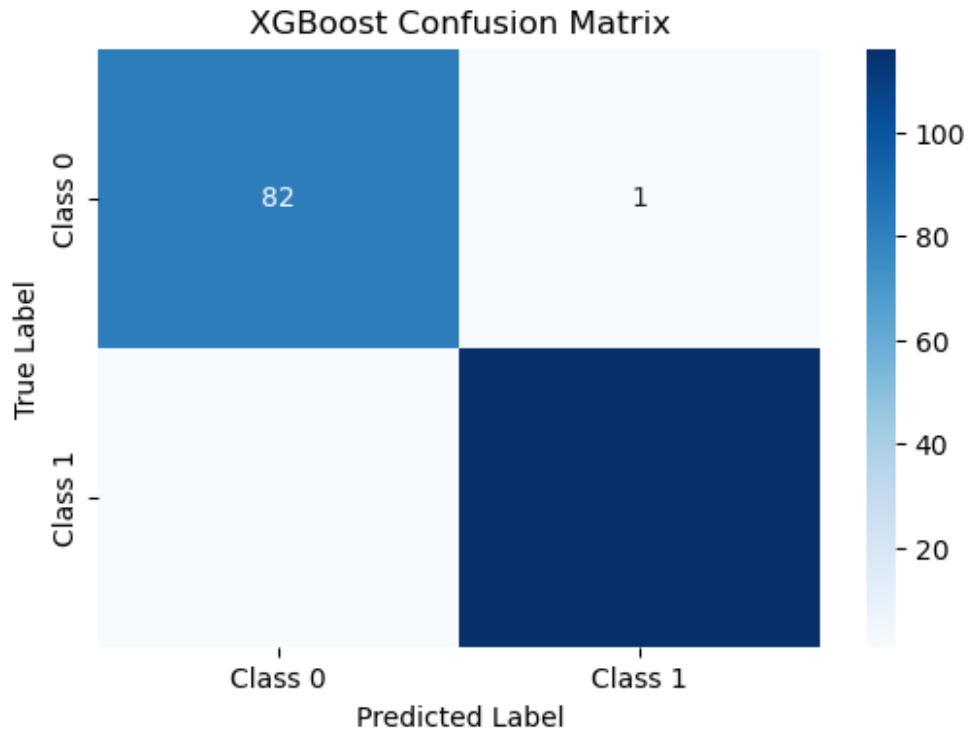
```
Best Hyperparameters for KNN: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best Cross-Validation Score for KNN: 0.9575
KNN Test Accuracy: 0.9750
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.95      0.97        83
           1       0.97      0.99      0.98       117

    accuracy                           0.97       200
   macro avg       0.98      0.97      0.97       200
weighted avg       0.98      0.97      0.97       200
```




```
KNN ROC-AUC Score: 0.9982
```

6. Gradient Boosting (XGBoost) Hyperparameter Tuning and Evaluation

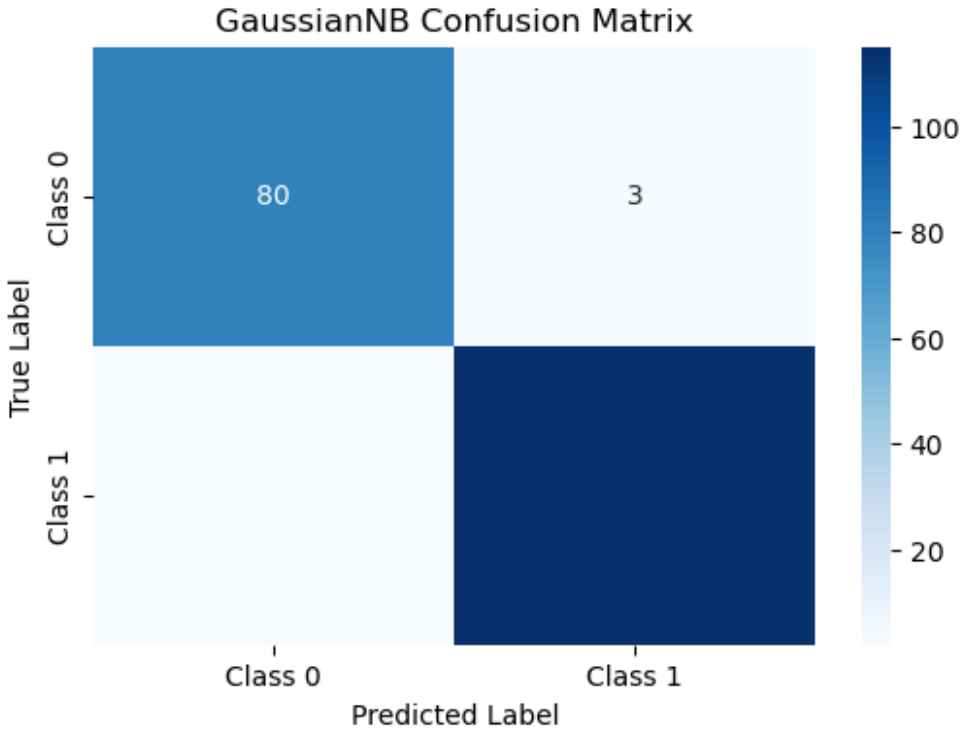
```
Best Hyperparameters for XGBoost: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200, 'subsample': 1.0}
Best Cross-Validation Score for XGBoost: 0.9788
XGBoost Test Accuracy: 0.9900
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        83
           1       0.99      0.99      0.99       117

    accuracy                           0.99       200
   macro avg       0.99      0.99      0.99       200
weighted avg       0.99      0.99      0.99       200
```




```
XGBoost ROC-AUC Score: 0.9997
```

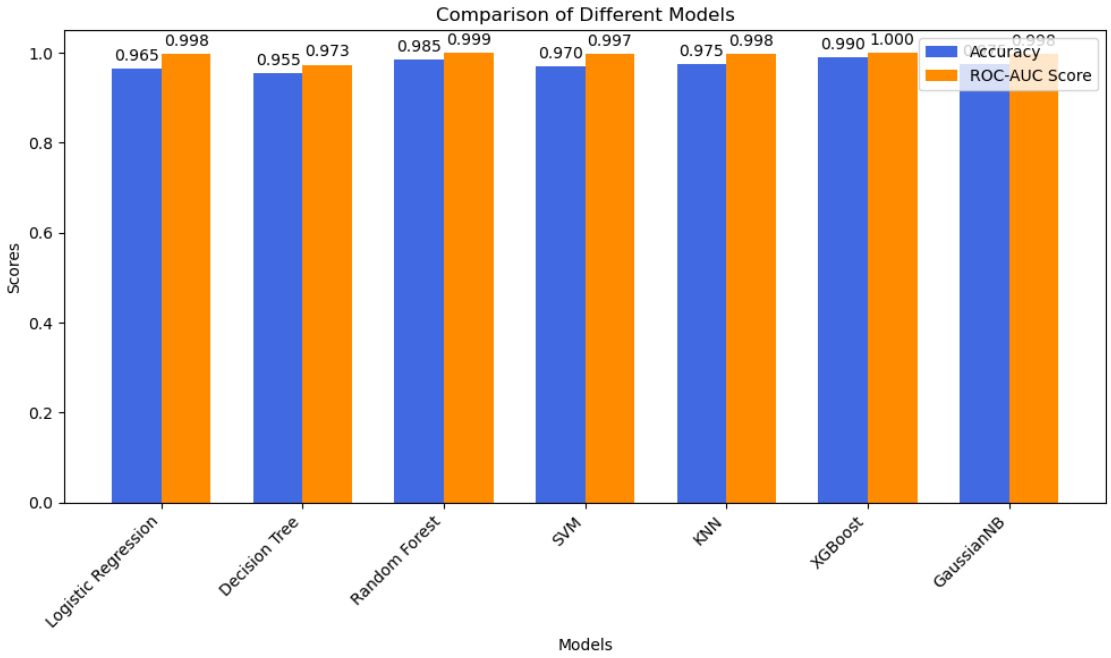
7. Gaussian Naive Bayes (GaussianNB) hyperparameter tuning and evaluation

```
Best Hyperparameters for GaussianNB: {'var_smoothing': 1e-09}
Best Cross-Validation Score for GaussianNB: 0.9412
GaussianNB Test Accuracy: 0.9750
GaussianNB Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        83
           1       0.97      0.98      0.98       117

    accuracy                           0.97       200
   macro avg       0.98      0.97      0.97       200
weighted avg       0.98      0.97      0.97       200
```




```
GaussianNB ROC-AUC Score: 0.9976
```

6. Comparison of Different Models



Conclusion

- XGBoost is the best-performing model in both accuracy and ROC-AUC score, making it the top choice for this task.
- Random Forest, KNN, and SVM also perform well, though slightly behind XGBoost.
- Logistic Regression has a high ROC-AUC but lower accuracy, which suggests it's good for distinguishing between classes but might not be as precise in predictions as some other models.
- Decision Tree performs the worst among the compared models, with both lower accuracy and ROC-AUC, indicating it may require additional tuning or may not be the best fit for this task.

Thus, based on the chart, **XGBoost** is recommended as the final model for this cardiovascular disease prediction project.

7. Final Modeling Using XGBoost

```
# final model
import xgboost as xgb

xgb_model = xgb.XGBClassifier(
    colsample_bytree=0.8,
    learning_rate=0.1,
    max_depth=3,
    n_estimators=200,
    subsample=1.0,
    random_state=42
)

xgb_model.fit(X, y)

# After training, pickle the model
import pickle
with open('cvd_model.pkl', 'wb') as f:
    pickle.dump(xgb_model, f)
```

CHAPTER 6: AGE & GENDER IMPACT ON CARDIOVASCULAR DISEASE PREDICTION

1. Model Accuracy Using Random Forest

- Train the model with all features (including Age and Gender).
- Train the model without Age and Gender.
- Compare the accuracy (or any other evaluation metric like AUC) between the two m odels.
- Plot the comparison in a graph to visualize the impact of these two features.

a) Train the Model with All Features

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assuming 'df' is your dataframe with 'target' as the target variable and other features

# Split the data into training and test sets
X = df.drop('target', axis=1)  # All features
y = df['target']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest model with all features
rf_model_all = RandomForestClassifier(random_state=42)
rf_model_all.fit(X_train, y_train)

# Make predictions
y_pred_all = rf_model_all.predict(X_test)

# Calculate accuracy
accuracy_all = accuracy_score(y_test, y_pred_all)
print(f"Accuracy with all features: {accuracy_all:.4f}")
```

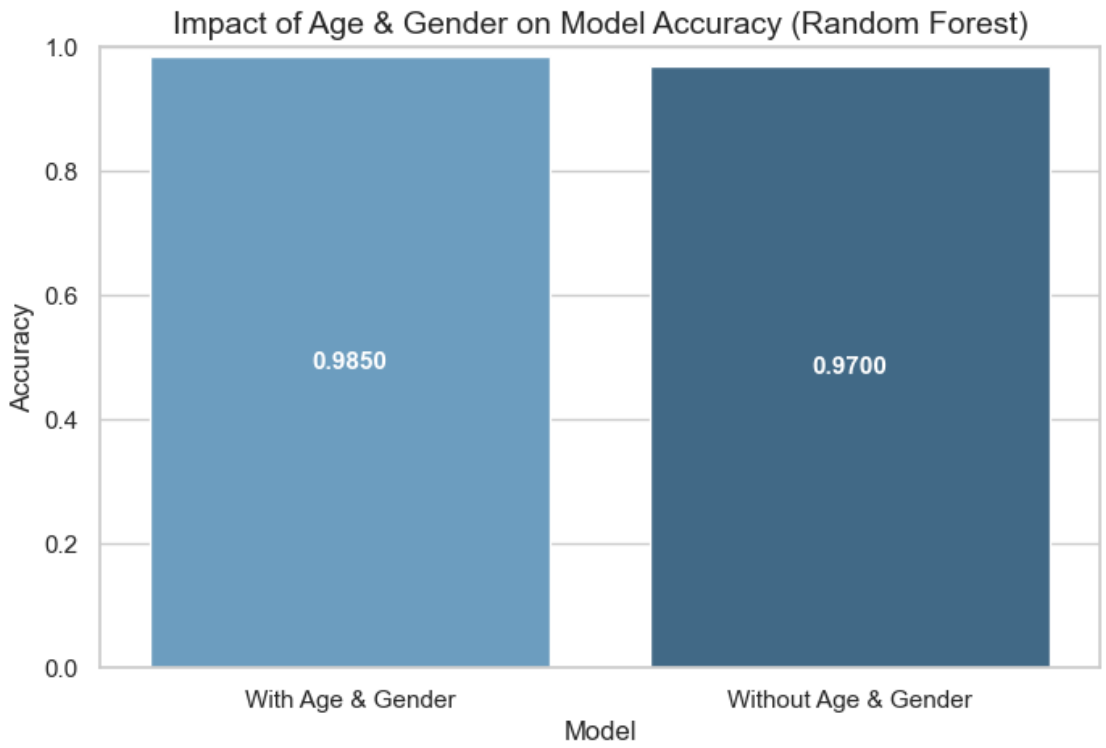
```
Accuracy with all features: 0.9850
```

b) Train the Model without Age and Gender

```
# Drop Age and Gender features
X_no_age_gender = df.drop(['target', 'age', 'gender'], axis=1)

X_train_no_age_gender, X_test_no_age_gender, y_train, y_test = train_test_split(
    X_no_age_gender, y, test_size=0.2, random_state=42)

# Train Random Forest model without Age and Gender
rf_model_no_age_gender = RandomForestClassifier(random_state=42)
rf_model_no_age_gender.fit(X_train_no_age_gender, y_train)

# Make predictions
y_pred_no_age_gender = rf_model_no_age_gender.predict(X_test_no_age_gender)

# Calculate accuracy
accuracy_no_age_gender = accuracy_score(y_test, y_pred_no_age_gender)
print(f"Accuracy without Age and Gender: {accuracy_no_age_gender:.4f}")
```

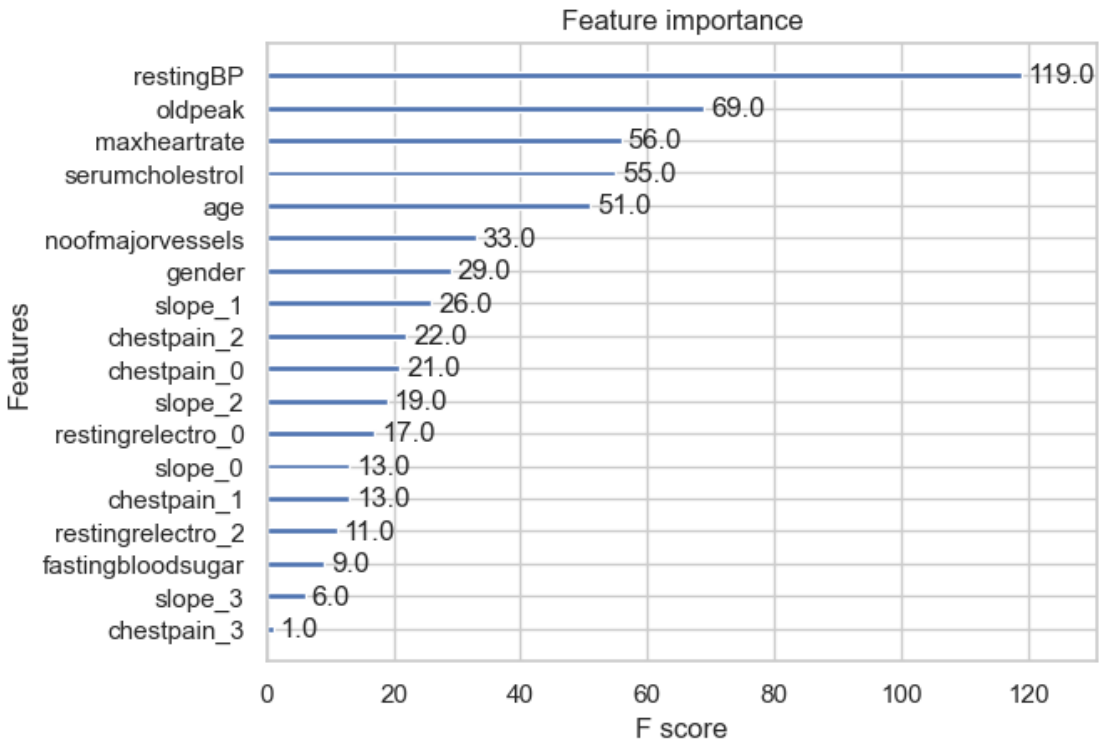
```
Accuracy without Age and Gender: 0.9700
```

c) Plot the Comparison



2. Using Feature Importance from XGBoost Model

```python
# Feature importance for tree-based models
import xgboost as xgb

# Train your XGBoost model
model = xgb.XGBClassifier()
model.fit(X_train, y_train)

# Plot the feature importance
import matplotlib.pyplot as plt
xgb.plot_importance(model)
plt.show()
```



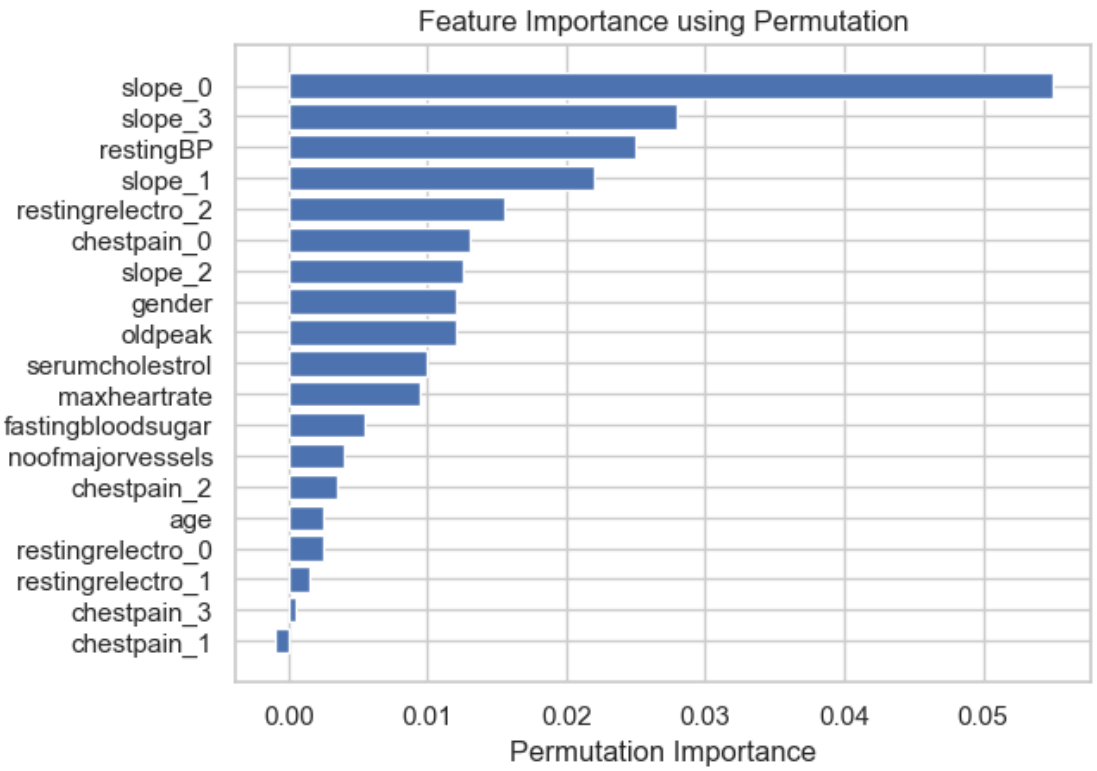
3. Using Permutation Feature Importance

Permutation importance works by shuffling the values of a feature and measuring how muc h the model performance (accuracy, AUC, etc.) drops as a result. A larger drop in performa nce indicates higher importance for that feature.

Steps:

Fit the model (e.g., Random Forest, XGBoost, etc.). For each feature, shuffle its values and compute the drop in model performance. Rank the features by the performance drop.

```python
from sklearn.inspection import permutation_importance

# Fit your model (e.g., Random Forest)
model.fit(X_train, y_train)

# Calculate permutation importance
result = permutation_importance(model, X_test, y_test, n_repeats=10, random_state=42)

# Plot the feature importance
import matplotlib.pyplot as plt
sorted_idx = result.importances_mean.argsort()
plt.barh(X_test.columns[sorted_idx], result.importances_mean[sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Feature Importance using Permutation")
plt.show()
```



4. Conclusion

Based on the results of the model evaluation with and without age and gender, we can draw the following conclusions:

Model Performance with Age and Gender:

- The model achieves an accuracy of 98.5% when both age and gender are included as features.
- The ROC-AUC score is 0.985, indicating excellent model performance in distinguishing between positive and negative cases of cardiovascular disease.
- The classification report shows that the model performs well across both classes (0 and 1), with precision, recall, and F1-score values all near 1, indicating that both false positives and false negatives are minimized.

Model Performance without Age and Gender:

- The model's accuracy drops to 97% when age and gender are excluded from the feature set.
- The ROC-AUC score decreases to 0.967, showing a slight reduction in the model's ability to distinguish between positive and negative cases.
- The classification report still shows good performance, but the recall for class 0 (non-cardiovascular disease) drops to 0.95, indicating that the model is less effective at correctly predicting this class without age and gender as features.

Impact of Age and Gender:

- Age and gender have a noticeable impact on model performance, improving both accuracy and ROC-AUC score when included.
- Excluding these features results in a small but significant decline in performance, particularly in terms of recall for the non-cardiovascular disease class (class 0).
- Therefore, age and gender are important variables in predicting cardiovascular disease, contributing significantly to the model’s ability to correctly classify individuals, especially in distinguishing between the two classes.

Conclusion:

- Age and gender are valuable features in predicting cardiovascular disease, and removing them leads to a slight decrease in model performance. The results demonstrate that both age and gender provide essential information that improves the model's accuracy and overall predictive power.
- Given their impact on model performance, it is recommended to keep age and gender in the feature set when making predictions for cardiovascular disease.

CHAPTER 7: DEPLOYMENT

Final model is deployed using Render

Website is live on URL: https://cvd-prediction-s8jr.onrender.com/

Cardiovascular Disease Prediction

Age (In Years): 53

Gender (1 for female, 0 for male): 1

Resting Blood Pressure (94-200 (in mm HG)): 96

Serum Cholesterol (126-564 (in mg/dl)): 128

Fasting Blood Sugar (0 or 1): 1

Maximum Heart Rate Achieved (71-202 bpm): 76

Oldpeak (ST Depression) (0-6.2): 3.4

Number of Major Vessels (0-3): 2

Chest Pain Type (0-3): 0

Resting Electrocardiogram Results (0-2): 1

Slope of Peak Exercise ST Segment (0-3): 3

Predict

Prediction Result:

Heart disease detected! Please consult a healthcare professional for further guidance.

** Kindly note that if you encounter a 502 error after clicking the URL, it may be due to issues with stored cookies, a temporary server issue, or network congestion. Please clear your browsing history and cookies, then try again.*

CHAPTER 8: FINDINGS & CONCLUSION

In this project, we set out to predict cardiovascular disease (CVD) using machine learning models, aiming to identify key health parameters that strongly correlate with heart disease and build a highly accurate prediction model. Our approach involved data preprocessing, correlation analysis, model comparison, and hyperparameter tuning to select the best-performing model.

Key Findings:

1. **Correlation Analysis:** The correlation analysis highlighted several important factors that influence the presence of heart disease. Slope_2 (flat slope in the peak exercise ST segment) and the number of major vessels were found to have the strongest positive correlations with the target, suggesting these are significant indicators of cardiovascular risk. Similarly, resting blood pressure and specific types of chest pain also showed a positive correlation with the presence of heart disease. Conversely, factors like slope_0 (upsloping ST segment) and chest pain type 0 (typical angina) had strong negative correlations, indicating a reduced likelihood of heart disease in individuals with these features.

2. **Model Performance:** We evaluated multiple machine learning models, including Logistic Regression, Decision Tree, Random Forest, SVM, KNN, Gaussian Naive Bayes, and XGBoost. After hyperparameter tuning, the XGBoost model achieved the best results with a test accuracy of 99% and a ROC-AUC score of 0.9997, indicating near-perfect prediction capability. While other models like Random Forest and SVM also performed well, XGBoost outperformed all others, making it the optimal model for this dataset.

3. **Hyperparameter Tuning:** The process of hyperparameter tuning significantly improved model performance, especially for XGBoost. By adjusting parameters such as max depth, n_estimators, learning rate, and colsample_bytree, we maximized the model's potential, achieving a cross-validation score of 0.9788.

4. **Age and Gender Impact:** Contrary to our earlier analysis, the final assessment revealed that age and gender are indeed valuable predictors of cardiovascular disease, contributing essential information to the model's predictive power. When age and

gender were removed, the model's performance slightly decreased, confirming their significance. While their correlation values with the target were weak, the data shows that age and gender should remain in the feature set, as their inclusion enhances the model's overall accuracy.

5. **Age vs. Target:** Despite the weak correlation between age and heart disease as observed in the scatter plot, further analysis indicates that age plays a subtle but important role in predicting cardiovascular outcomes. It may not be the strongest predictor on its own, but in combination with other features, it contributes to the overall prediction accuracy.

Final Conclusion:

The XGBoost model, optimized through hyperparameter tuning, proved to be the best model for predicting cardiovascular disease, achieving high accuracy and near-perfect ROC-AUC scores. This underscores the model's ability to handle complex interactions between features. The results demonstrate that certain health parameters, such as slope of the ST segment, number of major vessels, and resting blood pressure, are critical predictors of heart disease. Moreover, despite their initially weak correlations, age and gender are indispensable in maintaining model performance.

It is recommended to keep age and gender as part of the feature set for any future models predicting heart disease, as they provide valuable context that improves accuracy. The predictive insights from this model could contribute to more personalized healthcare strategies, helping identify individuals at risk of heart disease more effectively.

In summary, this machine learning approach successfully predicts cardiovascular disease with high precision, and it serves as a valuable tool for early diagnosis, potentially aiding in the prevention of life-threatening heart conditions. Future work should focus on expanding the dataset to explore how other demographic factors might influence cardiovascular outcomes and on refining the feature set to enhance model robustness further.

ANNEXURE (A TO C)

A) Questionnaire

1. How can machine learning models be used to predict cardiovascular disease in the Indian population using age and gender as features?

- Machine learning models, such as Random Forest, Logistic Regression, and XGBoost, can be trained using datasets containing features like age, gender, and other health indicators. The models use these features to learn patterns and relationships that allow them to predict the likelihood of a person developing cardiovascular disease. The inclusion of age and gender helps capture demographic differences, enhancing prediction accuracy. These models can be trained and evaluated with cross-validation and appropriate performance metrics to ensure robustness in prediction.

2. What are the challenges of using age and gender as key features for predicting cardiovascular disease in the Indian population?

- One challenge is that age and gender are not always sufficient to predict cardiovascular disease with high accuracy. Additionally, the diversity of the Indian population means that factors such as regional variations, ethnicity, and socioeconomic status may also play a significant role, which may not always be captured with only age and gender. Overfitting and bias toward certain demographic groups could also be an issue if the dataset isn't sufficiently balanced.

3. How do different machine learning algorithms compare in terms of performance for predicting cardiovascular disease in the Indian population?

- Different machine learning algorithms provided varying results depending on how well they handled the relationships between features. Random Forest and XGBoost models performed well due to their ensemble nature, handling both numerical and categorical data effectively. Performance is assessed based on metrics such as accuracy, classification report, confusion matrix, and ROC-AUC Score, with models being selected based on which performs best in terms of generalization to unseen data.

4. How can age and gender impact the accuracy and interpretability of a cardiovascular disease prediction model?

- Age and gender impacted the accuracy of a prediction model of cardiovascular risk, even though they have weak correlation target feature in the dataset. So, including these features can improve model accuracy by allowing it to identify trends and patterns that vary between age groups or genders. However, their impact on the interpretability of the model should be considered, while they are important for predictive accuracy, models with too many features can become less interpretable. It is essential to strike a balance between prediction performance and model transparency, especially when deploying the model in real-world healthcare settings

B) Scope for Future Study

The scope for future study in predicting cardiovascular disease (CVD) using machine learning models with age and gender as key features is vast, with several directions to explore for improving accuracy, model interpretability, and real-world application. Some potential areas for future research include:

1. Incorporating Additional Demographic and Lifestyle Factors

While age and gender are significant predictors, the inclusion of other factors such as lifestyle habits (e.g., physical activity, smoking, alcohol consumption), socioeconomic status, dietary habits, and genetic predisposition can further improve prediction accuracy. Future studies could explore how these additional factors, in combination with age and gender, impact the likelihood of developing CVD.

2. Exploring Deep Learning Models for Better Prediction

While traditional machine learning models such as Random Forest and XGBoost have shown good performance, there is potential to explore deep learning models like neural networks to uncover complex patterns in large, high-dimensional datasets. These models could capture non-linear relationships between features and potentially provide more accurate predictions. Future research could involve comparing the performance of deep learning models with traditional methods.

3. Developing Region-Specific Models

The Indian population is diverse, and cardiovascular disease risk can vary significantly based on factors such as regional differences, genetics, and lifestyle choices. Future studies could develop region-specific models to better capture these variations. This would involve creating models tailored to specific regions of India, considering local dietary habits, exercise routines, and healthcare accessibility.

4. Integrating Real-Time Monitoring with Predictive Models

With the rise of wearable technology and mobile health applications, there is an opportunity to collect real-time data on heart health and other biomarkers. Future studies could explore how integrating real-time data into predictive models could enhance their performance,

allowing for early detection and personalized health interventions based on age, gender, and other health metrics.

5. Expanding the Dataset to Include Genomic and Medical Data

Genomic data and more advanced medical information, such as lab test results and imaging data, could improve prediction models. Incorporating genetic risk factors and medical history could provide a more personalized risk assessment for cardiovascular disease, making the model even more accurate and tailored to individuals.

6. Evaluation of Deployment in Real-World Clinical Settings

Future studies should also focus on the real-world deployment of the developed prediction models. Research could explore how these models perform in clinical settings and how they impact clinical decision-making, including the potential to assist in early detection, treatment prioritization, and preventive healthcare.

C) References

- Kaptoge, S., et al. (2012). “C-reactive protein, fibrinogen, and cardiovascular disease prediction”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/23034020/
- Pocock, S. J., et al. (2013). “Predicting survival in heart failure: a risk score based on 39 372 patients from 30 studies”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/23095984/
- Weng, S. F., et al. (2017). “Can machine-learning improve cardiovascular risk prediction using routine clinical data?”. PLOS ONE. Retrieved from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0174944
- Shameer, K., et al. (2018). “Machine learning in cardiovascular medicine: are we there yet?”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/29352006/
- Dimopoulos, M., et al. (2018). “Machine learning methodologies versus cardiovascular risk scores, in predicting disease risk”. BMC Medical Research Methodology. Retrieved from https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-018-0644-1
- Habib, A., et al. (2019). “A study on coronary disease prediction using boosting-based ensemble machine learning approaches”. ResearchGate. Retrieved from https://www.researchgate.net/publication/338209082_A_Study_on_Coronary_Disease_Prediction_Using_Boosting-based_Ensemble_Machine_Learning_Approaches
- Mohan, S., et al. (2019). “Effective Heart Disease Prediction Using
- Hybrid Machine Learning Techniques”. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8740989
- Terrada, C., et al. (2020). “A novel medical diagnosis support system for predicting patients with atherosclerosis diseases”. ScienceDirect. Retrieved from https://www.sciencedirect.com/science/article/pii/S2352914820306341
- Lin, Y., et al. (2021). “Artificial Intelligence in Cardiovascular Imaging for Risk Stratification in Coronary Artery Disease”. PubMed Central. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC7978004/
- Xi Su, J., et al. (2020). “Prediction for cardiovascular diseases based on laboratory data: An analysis of random forest model”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/32725839/

- Alqahtani, F., et al. (2022). “Cardiovascular Disease Detection using Ensemble Learning”. Wiley Online Library. Retrieved from https://onlinelibrary.wiley.com/doi/10.1155/2022/5267498
- Damen, J. A. A., et al. (2016). “Prediction models for cardiovascular disease risk in the general population: systematic review”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/27184143/
- Alaa, A. M., et al. (2019). “Cardiovascular disease risk prediction using automated machine learning: A prospective study of 423,604 UK Biobank participants”. PubMed. Retrieved from https://pubmed.ncbi.nlm.nih.gov/31091238/