# Abstract

Heart disease is a common, sometimes fatal illness that impacts millions of individuals globally. For cardiac disease to be effectively treated and negative consequences to be avoided, early identification and precise diagnosis are essential. In recent years, machine learning techniques have shown promising results in aiding healthcare professionals in diagnosing heart disease. This study investigates the use of machine learning algorithms for the identification of heart disease using a dataset containing various clinical parameters. The research assesses the effectiveness of multiple classifiers, such as Support Vector Machines (SVM), Random Forests, Decision Trees, and a hybrid model that combines these methods. Through extensive experimentation and analysis, the paper aims to identify the best method for machine learning for heart disease detection.

***Keywords***—Heart Disease, K-Nearest Neighbors, Machine Learning, Random Forest, Decision Tree, Hybrid Model, Support Vector Machines (SVM), Diagnosis.

# TABLE OF CONTENTS

1. Introduction ………………………………………………………………… 1
2. Profile of the Problem …………………………………………………… 3
3. Existing System ……………………………………………………………. 4
   3.1. Introduction ……………………………………………………………. 4
   3.2. Existing Software ……………………………………………………..... 4
   3.3.What's the System to be developed ………………………………….. 5
4. Problem Analysis ………………………………………………………… 6
   4.1. Product definition ……………………………………………………… 6
   4.2. Feasibility Analysis ……………………………………………………. 6
   4.3.Project Plan …………………………………………………………….. 7
5. Software Requirement Analysis ………………………………………….. 8
   5.1. Introduction ……………………………………………………………. 8
   5.2.General Description ……………………………………………………. 8
   5.3. Specific Requirement ………………………………………………….. 9
6. Implementation ……………………………………………………………. 11
   6.1. Implementation of the Project ……………………………………….. 11
   6.2.Conversion Plan ………………………………………………………… 11
   6.3.Post Implementation and Software Maintenance ……………………. 12
7. Project Legacy …………………………………………………………… 13
   7.1. Current Status of the Project ………………………………………… 13
   7.2. Remaining areas of concern ………………………………………… 13
   7.3. Technical and Managerial lesson learned …………………………… 14
8. Dataset ……………………………………………………………………… 16
9. Source Code ……………………………………………………………… 18
10. Project Snapshot …………………………………………………………. 31
11. Project Acceptance letter ………………………………………………….32
12. Bibliography ……………………………………………………………... 33
13. Plagiarism Report ……………………………………………………………34

## 1. **INTRODUCTION**

Cardiovascular diseases (CVDs) stand as a predominant cause of mortality globally, posing a significant burden on public health systems and economies worldwide. The prevalence of CVDs is increasing despite advances in medical research and healthcare, therefore proactive approaches to prevention, early identification, and intervention are required. The incorporation of data-driven methods, especially machine learning, has become a viable means of boosting cardiovascular risk assessment and improved patient outcomes in response to this necessity.

The aim of this project needs to be to leverage machine learning techniques to develop predictive models capable of assessing an individual's risk of cardiovascular diseases based on a comprehensive array of clinical and demographic factors. By analysing a dataset comprising demographic information, medical history, lifestyle factors, and diagnostic indicators, we endeavour to construct models that can accurately stratify patients into different risk categories, thereby facilitating targeted interventions and personalized healthcare strategies.

The significance of this initiative lies in its potential to revolutionize cardiovascular care by providing physicians with valuable insights through data-driven analysis (doi.org). Conventional risk assessment instruments frequently depend on restricted categories of risk factors and could not completely encompass the intricate interaction of variables impacting cardiovascular well-being. On the other hand, machine learning algorithms can find complex patterns and correlations in large datasets, which makes it possible to create risk prediction models that are more accurate and reliable.
Thanks to a combination of rigorous preliminary data, case analysis and infrastructure development, model development, and performance evaluation, this project seeks to harness the power of machine learning to enhance cardiovascular risk prediction. By identifying novel risk factors, elucidating non-linear associations, and integrating diverse sources of data, we aim to refine existing risk assessment methodologies and augment clinical decision-making processes.

Furthermore, by fostering interdisciplinary collaboration between data scientists, clinicians, epidemiologists, and public health experts, this project endeavors to bridge the gap between data analytics and clinical practice. By translating insights gleaned from data analyses into actionable strategies for risk mitigation and intervention, we aspire to empower healthcare practitioners with tools that enable them to deliver more personalized and effective care to individuals at risk of developing cardiovascular diseases. Moreover, this project aims to address disparities in healthcare access and outcomes by leveraging machine learning models to identify high-risk populations and tailor interventions to suit their specific needs. By analyzing demographic and socioeconomic factors alongside clinical data, we strive to uncover underlying determinants of health disparities and develop targeted interventions aimed at reducing inequities in cardiovascular care delivery.

Ultimately, our goal is to contribute to the ongoing efforts aimed at combating the global burden of cardiovascular diseases. We hope to advance the fields of cardiovascular risk assessment, preventive medicine, and population health management by utilizing the wealth of information contained within healthcare datasets. In the end, we aim to create a future in which the incidence and consequences of CVDs are significantly reduced.

Through continuous refinement of predictive models and iterative improvements in healthcare delivery, we seek to pave the way for a healthier and more resilient society, where every individual has access to personalized, evidence-based care tailored to their unique risk profile.

## 2. **PROFILE OF THE PROBLEM**

Cardiovascular diseases (CVDs) are a major global health concern that have a severe impact on people's lives and healthcare systems throughout the globe. These illnesses, which include a variety of disorders affecting the heart and blood vessels, not only greatly increase rates of morbidity and death but also place a heavy financial strain on communities. The prompt and correct identification of CVDs remains a persistent issue despite advancements in medical research and healthcare delivery. This challenge frequently results in delayed treatments, inferior treatment outcomes, and higher healthcare expenditures.

Traditional diagnostic approaches for heart disease, reliant on manual interpretation of medical data and subjective clinical judgment, are fraught with limitations. The inherent complexity and variability of CVD manifestations, coupled with the subtlety of early symptoms, make accurate diagnosis a daunting task for healthcare practitioners. Moreover, the reliance on conventional risk assessment tools, such as symptom-based scoring systems and invasive diagnostic procedures, may not always yield definitive results, leading to diagnostic uncertainty and therapeutic indecision.

In this landscape of diagnostic uncertainty and clinical ambiguity, the promise of machine learning (ML) shines brightly as a beacon of hope. By harnessing the power of advanced computational algorithms and vast repositories of medical data, ML techniques offer the potential to revolutionize cardiovascular healthcare. These algorithms, capable of analyzing complex patterns and extracting actionable insights from diverse datasets, hold the key to unlocking new frontiers in heart disease detection, risk assessment, and personalized intervention strategies.

The rationale for our study lies in the imperative to bridge the gap between traditional diagnostic methodologies and the burgeoning field of ML-driven healthcare. By leveraging state-of-the-art ML algorithms and comprehensive clinical datasets, we aim to develop a robust predictive model for heart disease detection that surpasses the limitations of existing approaches. Our research seeks to address critical questions regarding the efficacy, reliability, and scalability of ML techniques in real-world clinical settings, paving the way for their integration into routine medical practice.

Our work is a multidisciplinary investigation into machine learning methods for heart disease identification with a carefully selected dataset from various sources. We want to rigorously assess the performance of hybrid ensemble models using different machine learning methods, such as Random Forests, Decision Trees, and K-Nearest Neighbors. Our goal is to determine the best method for detecting heart disease by thoroughly testing, evaluating performance, and clarifying the advantages and disadvantages of each algorithm.

# 3. EXISTING SYSTEM

## 3.1. Introduction

The existing landscape of heart disease detection primarily relies on traditional diagnostic methods, which often involve manual interpretation of medical data by healthcare professionals. These methods, while effective to some extent, are inherently limited by subjectivity, reliance on symptom-based assessments, and the lack of sophisticated analytical tools. Moreover, the process of diagnosing heart disease typically involves a series of sequential steps, including medical history review, physical examination, diagnostic tests (e.g., electrocardiogram, echocardiogram), and risk stratification algorithms (e.g., Framingham Risk Score). While these approaches have been instrumental in identifying cardiovascular risk factors and guiding clinical decision-making, they are not without their shortcomings.

## 3.2. Existing Software

Within the realm of software solutions for heart disease detection, several tools and applications exist to assist healthcare professionals in diagnosing and managing cardiovascular conditions. Some examples include:

1. Electronic Health Records (EHR) Systems: These are extensive digital repositories where patient health data, such as medical histories, test findings, prescription histories, and treatment plans, are kept. EHR systems help healthcare practitioners make well-informed choices regarding patient care by centralizing and making patient data more easily accessible.
2. Clinical Decision Support Systems (CDSS): Software programs called CDSS are designed to support medical professionals in clinical decision-making by offering alerts, reminders, and suggestions based on evidence. These systems use algorithms to evaluate patient data and provide recommendations for diagnostic tests, risk assessments, and customized treatment regimens.
3. Medical Imaging Software: A variety of programs are used to analyze and analyze medical pictures, including X-rays, CT, MRI, and echocardiograms. This type of software is known as medical imaging software. Healthcare workers may see anatomical features, spot anomalies, and gauge the course of a disease with the use of these technologies.
4. Risk Assessment Tools: A person's risk of getting cardiovascular illnesses may be estimated using a variety of risk assessment tools and calculators. These programs create customized risk assessments and suggestions based on lifestyle variables, clinical measurements (such as blood pressure and cholesterol levels), and demographic data.

While these existing software solutions play a valuable role in cardiovascular healthcare, they are often limited in their ability to leverage advanced machine learning techniques for predictive analytics and risk assessment. The development of more sophisticated ML-driven systems holds the promise of enhancing the accuracy, efficiency, and scalability of heart disease detection.

### 3.3. What kind of System needs to Be Developed?

The proposed system represents a paradigm shift in heart disease detection, leveraging advanced machine learning techniques to augment traditional diagnostic methodologies. Some key innovations and enhancements in the system to be developed include:

1. Integration of Machine Learning Algorithms: Modern machine learning techniques, including K-Nearest Neighbours, Random Forests, and Decision Trees, will be included into the system to evaluate clinical data and spot trends that might be signs of cardiac disease. These algorithms will improve the efficiency and accuracy of cardiac disease diagnosis by enabling automated pattern identification, risk stratification, and predictive modelling.

2. Comprehensive Data Analysis: Unlike traditional diagnostic methods, which rely on manual interpretation of medical data, the proposed system will perform comprehensive data analysis, leveraging advanced analytics techniques to extract actionable insights from large-scale clinical datasets. This data-driven approach will enable more precise risk assessment, early detection of cardiac abnormalities, and personalized treatment recommendations.

3. Real-time Decision Support: At the point of care, the system will give evidence-based suggestions, alerts, and reminders to healthcare workers in real-time. Through the integration of clinical standards and predictive analytics into the workflow, the system will help doctors make well-informed decisions on follow-up care, treatment plans, and patient management.

4. Enhanced User Interface: The user interface of the system will be designed to facilitate ease of use, intuitive navigation, and seamless integration into existing clinical workflows. Through interactive dashboards, visualizations, and customizable reporting tools, the system will empower healthcare providers to access, analyze, and interpret patient data more effectively.

# 4. **PROBLEM ANALYSIS**

## 4.1. Product definition

The proposed product is a sophisticated machine learning-driven system aimed at transforming cardiovascular healthcare by improving diagnostic precision, risk evaluation, and patient care related to heart disease. At its core, this system harnesses cutting-edge algorithms and analytics methodologies to provide healthcare professionals with a comprehensive decision support tool. Through the integration of diverse clinical data sources, real-time decision support capabilities, and an intuitive user interface, the system endeavors to streamline the process of data visualization and analysis, empowering clinicians to make informed decisions swiftly and effectively.

The proposed device offers a comprehensive approach to the identification and management of heart disease, which is a major improvement in the field of cardiovascular healthcare. The system can evaluate a tone of patient data, including demographics, clinical measures, test results from diagnostic procedures, and medical imaging scans, by utilizing machine learning algorithms. Early identification and intervention are made easier by the system's ability to recognize minor patterns and correlations that may point to underlying cardiovascular problems thanks to this multidimensional analysis.

Furthermore, the product's user interface is designed with usability and accessibility in mind, ensuring that healthcare professionals of varying technical backgrounds can leverage its capabilities effectively. The interface provides intuitive data visualization tools, allowing users to explore complex datasets, identify trends, and generate actionable insights with ease. Additionally, the system offers real-time decision support features, providing clinicians with timely recommendations and alerts based on the latest patient data and medical guidelines.
The product's overall goal is to completely transform the methods for identifying, treating, and managing cardiac disease, which will eventually improve patient outcomes and the way healthcare is provided.

## 4.2. Feasibility Analysis

The feasibility analysis assesses the technical, economic, and operational viability of the proposed project, laying the groundwork for successful development and implementation.

From a technical perspective, the project requires expertise in machine learning, data science, software development, and healthcare domain knowledge. The availability of skilled personnel and access to relevant technology resources are essential for the project's success. Additionally, considerations such as data privacy, security, and interoperability must be addressed to ensure compliance with regulatory requirements and industry standards.

Economically, the project entails initial investment in software development, infrastructure setup, and data acquisition. A cost-benefit analysis is essential to evaluate the potential return on investment and long-term sustainability of the project. Factors such as market demand, competitive landscape, and revenue generation opportunities influence the economic feasibility of the project.

Operationally, the project must align with existing healthcare workflows and practices to facilitate seamless integration into clinical settings. User acceptance testing and stakeholder engagement are critical to ensuring the system meets the needs of end-users and contributes to improved patient care. Additionally, ongoing maintenance and support are essential to sustain the system's functionality and address evolving healthcare requirements.

## 4.3. Plan of Project

The plan of project projects provides detailed instructions for completing the project, describing activities, priorities, schedule, and allocation of resources. It serves as a framework for project management and ensures alignment with project objectives.

For creating system specifications, functional requirements, and user interface design, project stakeholders work together during the requirements collection and analysis project phase. To prepare the data for the creation of machine learning models, this phase also involves data collecting and preprocessing tasks including feature engineering, normalization, and data cleaning.

Once requirements are finalized and data preprocessing is completed, Project moves into the development phase, where software engineers and data scientists collaborate to implement machine learning algorithms, develop the user interface, and integrate backend systems. Continuous testing and validation are conducted throughout the development process to ensure system functionality, performance, and security.

Following development, the project enters the deployment phase, wherein the system is deployed in real-world healthcare environments for pilot testing and user feedback. This phase involves training end-users, monitoring system performance, and addressing any issues or concerns raised during the pilot phase. Feedback from end-users informs iterative improvements and enhancements to the system.

The project culminates with the system's widespread acceptance and implementation, along with thorough end-user training, documentation, and support services. Prioritizing ongoing upkeep, upgrades, and improvements helps to guarantee that the system continues to be functional, efficient, and in line with changing healthcare requirements.

# 5. SOFTWARE REQUIREMENT ANALYSIS

## 5.1. Introduction

A thorough analysis of the technologies employed in the creation of the heart disease detection system is present in the software requirement analysis. These technologies have been selected with care to facilitate system integration, sophisticated analytics, user interface design, and effective data processing. The creation and operation of the suggested system depend heavily on the following technologies:

1. Python Programming Language: The main programming language used for creating the heart disease detection system is Python. Python is favoured because of its many libraries for statistical analysis, machine learning, and data processing as well as its adaptability and ease of use. Model training and assessment are carried out using libraries like NumPy, Pandas, and Scikit-learn.
2. Streamlet Framework: Streamlet is employed to create the user interface for the heart disease detection system. Streamlet allows for rapid development of interactive web applications with simple Python scripts. It provides features for data visualization, user input widgets, and real-time updates, enabling seamless interaction between users and the system.

3. Machine Learning Libraries: The system uses several machine learning libraries to develop algorithms and prediction models for the diagnosis of cardiac disease. Numerous tools for model selection, regression, clustering, and classification are available with Scikit-learn. To improve prediction performance, deep learning models are also built and trained using TensorFlow and Kera's.

4. Data Visualization Tools: Clinical data is visualized, and graphical representations are made with the help of data visualization tools like Matplotlib and Seaborn. By visualizing trends, patterns, and correlations within the information, these technologies help healthcare practitioners make and evaluate decisions based on data.

## 5.2. General Description

The heart disease detection system employs a combination of technologies to enable efficient data processing, machine learning model development, and user interface design. Python serves as the foundational programming language, providing a flexible and powerful environment for implementing various components of the system.

Streamlet framework is utilized to build an intuitive user interface for the heart disease detection system. Streamlet enables the creation of interactive web applications directly from Python scripts, allowing for rapid prototyping and deployment.

Predictive models for heart disease diagnosis are created using machine learning frameworks like Scikit-learn, TensorFlow, and Kera's. A vast array of tools, methods, and algorithms are available in these libraries for feature selection, training, and assessment of models as well as data preparation.

Data visualization tools such as Matplotlib and Seaborn are integrated into the system to create informative visualizations of clinical data. These visualizations aid healthcare professionals in understanding complex relationships, identifying patterns, and interpreting model predictions.

**5.3. Specific Requirements**

1. Operating System Compatibility: The heart disease monitoring system may be accessed on several platforms due to its compatibility with major OS such as Windows, Linux and Mac-OS.

2. Client Requirements: For best speed and compatibility, clients accessing the system use a contemporary web browser like Microsoft Edge, Mozilla Firefox, Google Chrome, or Safari.

3. Tools and Libraries: Python: The system relies on Python 3 programming language for backend development and algorithm implementation.

   Streamlet: The user interface is built using Streamlet framework, facilitating rapid development of interactive web applications.

   Scikit-learn, TensorFlow, Kera's: These machine learning libraries are utilized for model development, training, and evaluation.

   Matplotlib, Seaborn: Data visualization tools are integrated into the system for creating insightful graphical representations of clinical data.

4. User Interface: The user interface of the heart disease detection system is web-based, accessible through a web browser. It features interactive components such as data input forms, result visualization widgets, and real-time updates.

5. Hardware Requirements: The hardware requirements for running the heart disease detection system are minimal, making it suitable for deployment on a wide range of computing devices.

   Minimum hardware specifications include a modern CPU (e.g., Intel Core i3 or AMD Ryzen 3), 4GB of RAM, and sufficient storage space for storing datasets and application files.

6. Scalability and Performance: System Architecture is made to be scalable, capable of handling large volumes of data and user requests efficiently.

Performance optimization techniques are implemented to ensure fast response times and efficient resource utilization, even under heavy workload conditions.

7. Documentation and Training: Comprehensive documentation is provided, including installation guides, user manuals, and API references, to assist healthcare professionals in deploying and using the system effectively.

Training materials such as tutorials, demo videos, and interactive sessions are offered to familiarize users with system features and functionalities, promoting user adoption and proficiency.

# 6. IMPLEMENTATION

The implementation phase of the heart disease detection project is a crucial step in transforming the conceptual design and requirements into a fully functional system. This phase involves various tasks, including developing the software components, integrating machine learning algorithms, designing the user interface, and ensuring compatibility with existing infrastructure. Let's delve deeper into each aspect of the implementation process.

## 6.1. Project Implementation

Writing code, constructing software components, and combining various modules to produce a functional system are all of the implementation of the cardiac disease detection project. Because of its many libraries and frameworks that are appropriate for web application development and machine learning, Python, a flexible programming language, is selected as the main language for development.

Developers leverage frameworks like Streamlit to create an interactive web-based interface for the heart disease detection application. Streamlit provides intuitive tools and features for building data-driven applications with minimal coding effort. It allows developers to quickly prototype ideas, visualize data, and deploy applications with ease.

Using the heart disease dataset, machine learning methods such as Random Forest, Decision Trees, and K-Nearest Neighbors (KNN) are applied to train prediction models. These algorithms are chosen because they work well with complicated datasets and are appropriate for classification jobs.

The implementation process follows best practices for software development, including modularization, code reuse, documentation, and version control using platforms like GitHub. Developers collaborate closely to ensure code quality, readability, and maintainability throughout the development lifecycle.

## 6.2. Conversion Plan

The conversion plan outlines the steps and strategies for transitioning from the existing heart disease detection system to the newly developed one. Stakeholders, including healthcare professionals, administrators, and end-users, are informed about the migration process and provided with training and support to facilitate a smooth transition.

Compatibility checks are conducted to ensure that the new system aligns with existing infrastructure, hardware, and software dependencies. Any potential conflicts or compatibility issues are addressed proactively to minimize disruptions during the conversion process.

Training sessions are organized to familiarize users with the new interface, features, and functionalities of the heart disease detection application. User feedback and suggestions are collected during training sessions to identify part of correction and refinement.

Data migration tools and scripts are developed to facilitate the seamless transfer of patient records, diagnostic histories, and other relevant information to the new system. Data integrity and confidentiality are prioritized throughout the migration process to ensure compliance with regulatory requirements and industry standards.

A rollback plan is established to mitigate risks and address any unforeseen issues or challenges encountered during the conversion process. This plan includes contingency measures, fallback procedures, and communication protocols to ensure continuity of operations and minimize potential disruptions.

## 6.3. Post-Implementation and Software Maintenance

Post-implementation activities focus on ensuring the smooth operation, performance optimization, and continuous improvement of the heart disease detection system. Regular maintenance tasks include monitoring system health, diagnosing and resolving issues, applying software updates, and implementing security patches.

User feedback and suggestions are collected to identify areas for enhancement, refinement, and feature expansion in future iterations of the application. Performance metrics, key performance indicators (KPIs) are monitored to assess system effectiveness, user satisfaction, and business impact.

Ongoing support services are provided to address user inquiries, technical assistance requests, and troubleshooting needs, ensuring a high level of customer satisfaction and so system reliability over time. Collaborative efforts between developers, stakeholders, and end-users put up to the successful implementation, adoption, and evolution of the heart disease detection system.

# 7. **PROJECT LEGACY**

The legacy of a project extends far beyond its initial implementation phase, encompassing its current status, areas of concern, and the lessons learned throughout its development journey. In the case of the heart disease detection project, understanding its legacy is essential for assessing its impact, identifying part for enhancement, and extracting valuable insights for future endeavours.

## 7.1. Project Current Status

As of the present moment, the heart disease detection project has reached a significant milestone, with the implementation phase nearing completion. The software components, including the user interface, machine learning algorithms, and data processing modules, have been developed and integrated into symmetric system.

Healthcare workers may easily enter patient data, view diagnostic findings, and use predictive models for heart disease diagnosis using the application's web-based interface, which was developed with Streamlit. Accurate risk assessment and diagnosis are made possible by machine learning algorithms like K-Nearest Neighbours (KNN), Random Forest, and Decision Trees, which have been trained using a large dataset.

The project's status reflects the collaborative efforts of developers, data scientists, and healthcare experts in translating the project's vision into a tangible solution. However, while significant progress has been made, still there are many sections that needs improvement and refinement to ensure the project's success and sustainability.

## 7.2. Remaining Areas of Concern

Despite the project's advancements, several areas of concern warrant further attention and mitigation strategies. These areas include:

1. Performance Optimization: While the current implementation of machine learning algorithms demonstrates promising results, there is room for performance optimization to enhance the application's speed, efficiency, and scalability. Techniques such as model optimization, feature engineering, and parallel processing can be explored to improve predictive accuracy and reduce computational overhead.

2. User Feedback Integration: Incorporating user feedback and suggestions is essential for enhancing the application's usability, functionality, and user experience. Establishing feedback mechanisms, conducting usability tests, and soliciting input from healthcare professionals can provide valuable insights into apart of correction and refinement.

3. Data Security and Privacy: Maintaining regulatory compliance and protecting sensitive information require ensuring the security and privacy of patient data. Protecting against unwanted access, data breaches, and privacy violations may be achieved by putting strong encryption methods, access restrictions, and data anonymization strategies into place.

4. Scalability and Compatibility: As the project evolves and scales to accommodate larger datasets and user bases, ensuring compatibility with existing infrastructure and hardware is crucial. Scalability measures, such as cloud deployment, containerization, and load balancing, can facilitate seamless expansion and resource allocation as demand grows.

5. Continuous Maintenance and Support: Fixing software defects, performance problems, and user questions requires ongoing maintenance and support. Timely resolution of issues and optimal system performance may be ensured by establishing a specialized support team, adopting a ticketing system, and adhering to service level agreements (SLAs).

### 7.3. Technical and Managerial Lessons Learned

Throughout the heart disease detection project, several technical and managerial lessons have been learned, providing valuable insights for future projects. These lessons include:

1. Interdisciplinary Collaboration: The success of the project relies on effective collaboration between developers, data scientists, healthcare professionals, and stakeholders. Interdisciplinary teamwork fosters innovation, creativity, and synergy, resulting in a more robust and impactful solution.

2. Agile Development Methodology: Adopting an agile development methodology enables iterative, adaptive, and customer-centric approaches to software development. Embracing principles such as continuous integration, frequent feedback loops, and iterative refinement promotes responsiveness to changing requirements and stakeholder needs.

3. Data-driven Decision Making: Using insights-driven methodologies and data analytics gives stakeholders the capacity to prioritize tasks, make well-informed decisions, and allocate resources as efficiently as possible. Transparency, accountability, and evidence-based processes are fostered by data-driven decision making, which produces more successful project outcomes.

4. Effective Communication: Clear and concise communication is essential for aligning project objectives, managing expectations, and resolving conflicts. Establishing open channels of

communication, conducting regular meetings, and fostering a culture of transparency promote collaboration, trust, and shared understanding among team members.

5. Risk Management: Proactively identifying, assessing, and mitigating project risks is critical for minimizing potential disruptions and ensuring project success. Implementing risk management strategies, conducting risk assessments, and developing contingency plans enable teams to anticipate challenges and mitigate their impact on project deliverables.

## 8. Dataset

The heart disease detection dataset utilized is sourced from the Cleveland database found within the UCI Machine Learning Repository. This repository is well-known for its comprehensive collection of machine learning datasets. The dataset consists of a diverse range of attributes intended to identify patterns and correlations regarding heart health.

There are 76 features in the dataset, and each one has the potential to offer important new information on cardiovascular health. It is noteworthy, therefore, that the majority of studies and analyses that have been published concentrate on a certain group of 14 traits. These exact traits, which are based mostly on the Cleveland database, have been meticulously selected and standardized across several investigations. It is important to remember that the values in this field are integers, ranging from 0 (showing no heart disease) to 4 (indicating significant presence). Analyses usually concentrate on differentiating between the presence (values 1, 2, 3, or 4) and absence (value 0) of cardiac disease for experimental reasons.

In order to capture important elements of heart health and facilitate successful predictive modelling, the subset of 14 attributes that are used in the majority of studies and trials have been carefully chosen. This group of features encompasses a variety of heart health-related factors, such as physiological measures, diagnostic test findings, and demographic variables. Scientists intend to use these traits to build robust prediction models that can accurately identify and classify cases of heart disease, facilitating early intervention and improved patient outcomes. Among these qualities are:

```
+----+--------------+--------------------------------------------------------------+
|    | Attributes   | Description                                                  |
+====+==============+==============================================================+
|  0 | age          | age                                                          |
+----+--------------+--------------------------------------------------------------+
|  1 | sex          | 1: male, 0: female                                           |
+----+--------------+--------------------------------------------------------------+
|  2 | cp           | chest pain type, 1: typical angina, 2: atypical angina       |
+----+--------------+--------------------------------------------------------------+
|  3 | trestbps     | resting blood pressure                                       |
+----+--------------+--------------------------------------------------------------+
|  4 | chol         | serum cholesterol in mg/dl                                   |
+----+--------------+--------------------------------------------------------------+
|  5 | fbs          | fasting blood sugar > 120 mg/dl                              |
+----+--------------+--------------------------------------------------------------+
|  6 | restecg      | resting electrocardiographic results (values 0,1,2)          |
+----+--------------+--------------------------------------------------------------+
|  7 | thalach      | maximum heart rate achieved                                  |
+----+--------------+--------------------------------------------------------------+
|  8 | exang        | exercise induced angina                                      |
+----+--------------+--------------------------------------------------------------+
|  9 | oldpeak      | oldpeak = ST depression induced by exercise relative to rest |
+----+--------------+--------------------------------------------------------------+
| 10 | slope        | the slope of the peak exercise ST segment                    |
+----+--------------+--------------------------------------------------------------+
| 11 | ca           | number of major vessels (0-3) colored by fluoroscopy         |
+----+--------------+--------------------------------------------------------------+
| 12 | thal         | thal: 3 = normal; 6 = fixed defect; 7 = reversible defect    |
+----+--------------+--------------------------------------------------------------+
```

In a dataset for heart disease detection, the variables represent different attributes related to the patients. Here's what each of these variables typically represents:

- age: Age of the patient.

- sex: Gender of the patient. (1 = male; 0 = female)

- cp: Chest pain type. It's categorical data, where:

  1 = typical angina

  2 = atypical angina

  3 = non-anginal pain

  4 = asymptomatic

- trestbps: Resting blood pressure (in mm Hg) upon admission to the hospital.

- chol: Serum cholesterol level (in mg/dl).

- fbs: Fasting blood sugar level > 120 mg/dl. (1 = true; 0 = false)

- restecg: Resting electrocardiographic results. It's categorical data, where:

  0 = normal

  1 = having ST-T wave abnormality

  2 = showing probable or definite left ventricular hypertrophy by Estes' criteria

- thalach: Maximum heart rate achieved.

- exang: Exercise induced angina. (1 = yes; 0 = no)

- oldpeak: ST depression induced by exercise relative to rest.

- slope: The slope of the peak exercise ST segment. It's categorical data, where:

  1 = upsloping

  2 = flat

  3 = downsloping

- ca: Number of major vessels (0-3) colored by fluoroscopy.

- thal: Thalassemia. It's categorical data, where:

  1 = normal

  2 = fixed defect

  3 = reversible defect

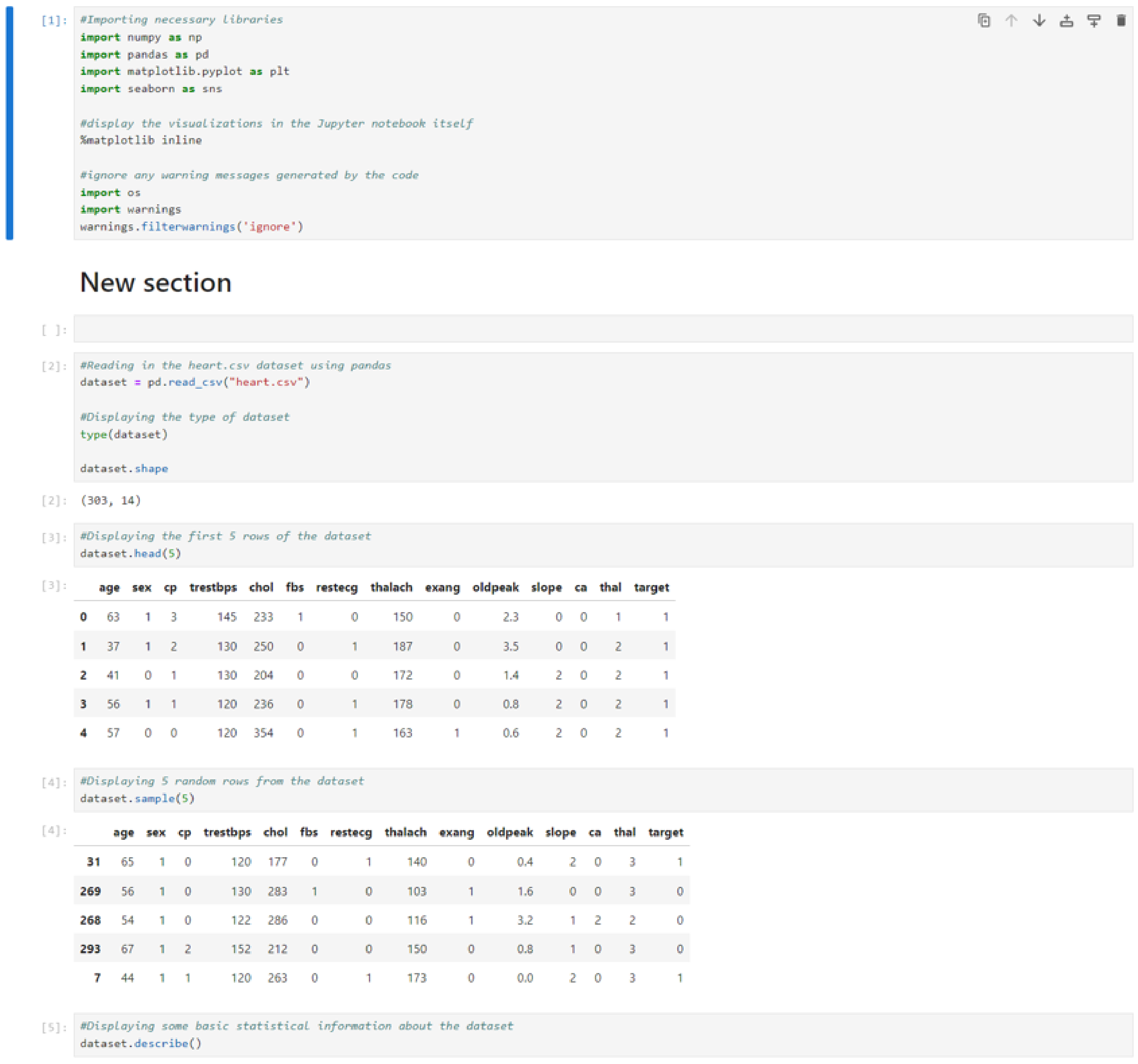
- target: The presence of heart disease. (1 = yes; 0 = no)

## 9. Source Code

**App.py file :**

```
import streamlit as st
import pickle
import numpy as np

# Load the Naive Bayes model
with open("final_model.sav", "rb") as f:
    gnb_model = pickle.load(f)

with open("./styles.css") as f:
    css = f.read()

st.markdown(f"<style>{css}</style>", unsafe_allow_html=True)

# Define the input features
feature_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
'exang', 'oldpeak', 'slope', 'ca', 'thal']

# Split the main content into two columns
col1, col2 = st.columns([1, 2])  # Adjust the width ratio as needed

# Add an image to the first column
with col1:
    st.image("heart.png", use_column_width=True)

# Add the title and description to the second column
with col2:
    # Define the app title and description
    st.title('Heart Disease Prediction')
    st.write('This software predicts if someone has heart disease by examining
various input features.')

    # Arrange input fields into three columns
    col3, col4, col5 = st.columns(3)

    # Create input fields for user input
    inputs = []
    for i, feature_name in enumerate(feature_names):
        if feature_name.lower() == "age":
            min_value = 0.0
            max_value = 100.0
        if feature_name.lower() == "sex":
            min_value = 0.0
            max_value = 1.0
        if feature_name.lower() == "cp":
            min_value = 0.0
            max_value = 4.0
        if feature_name.lower() == "fbs":
            min_value = 0.0
            max_value = 1.0
        if feature_name.lower() == "restecg":
            min_value = 0.0
            max_value = 2.0
        if feature_name.lower() == "exang":
            min_value = 0.0
            max_value = 1.0
        if feature_name.lower() == "slope":
            min_value = 0.0
            max_value = 3.0
        if feature_name.lower() == "ca":
            min_value = 0.0
            max_value = 3.0
        if feature_name.lower() == "thal":
            min_value = 0.0
            max_value = 3.0
        else:
            min_value = None
            max_value = None

        if i % 3 == 0:
            with col3:
                inputs.append(st.number_input(f'Enter {feature_name.capitalize()}',
value=0.0, step=1.0, min_value=min_value, max_value=max_value))
        elif i % 3 == 1:
            with col4:
                inputs.append(st.number_input(f'Enter {feature_name.capitalize()}',
value=0.0, step=1.0, min_value=min_value, max_value=max_value))
        else:
            with col5:
                inputs.append(st.number_input(f'Enter {feature_name.capitalize()}',
value=0.0, step=1.0, min_value=min_value, max_value=max_value))

# Predict function
def predict(features):
    # Convert input features to numpy array
    features_array = np.array(features).reshape(1, -1)
    # Predict probability
    prob = gnb_model.predict_proba(features_array)[0][1]
    return prob

# Create predict button
if st.button('Predict'):
    # Make prediction
    prediction = predict(inputs)
    # Display prediction
    if prediction > 0.5:
        st.error(f'There is a high probability of heart disease. Please consult a
doctor. Probability: {prediction:.2f}')
    else:
        st.success(f'No heart disease detected. Probability: {prediction:.2f}')
```

**<u>Jupyter Notebook:</u>**



```
[1]: #Importing necessary libraries
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     #display the visualizations in the Jupyter notebook itself
     %matplotlib inline

     #ignore any warning messages generated by the code
     import os
     import warnings
     warnings.filterwarnings('ignore')
```

## New section

```
[ ]:
```

```
[2]: #Reading in the heart.csv dataset using pandas
     dataset = pd.read_csv("heart.csv")

     #Displaying the type of dataset
     type(dataset)

     dataset.shape
```

```
[2]: (303, 14)
```

```
[3]: #Displaying the first 5 rows of the dataset
     dataset.head(5)
```

[3]:

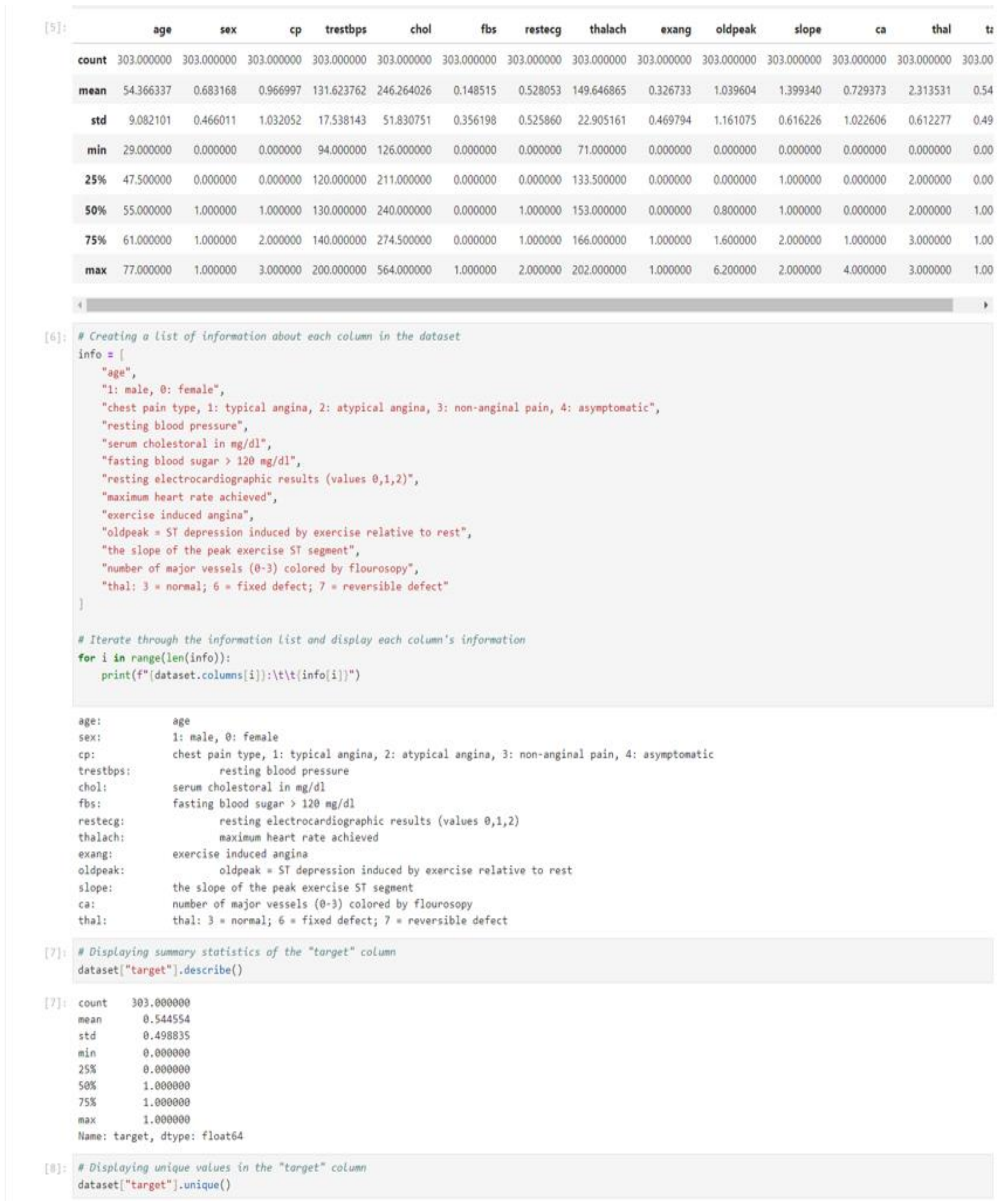
| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |

```
[4]: #Displaying 5 random rows from the dataset
     dataset.sample(5)
```

[4]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **31** | 65 | 1 | 0 | 120 | 177 | 0 | 1 | 140 | 0 | 0.4 | 2 | 0 | 3 | 1 |
| **269** | 56 | 1 | 0 | 130 | 283 | 1 | 0 | 103 | 1 | 1.6 | 0 | 0 | 3 | 0 |
| **268** | 54 | 1 | 0 | 122 | 286 | 0 | 0 | 116 | 1 | 3.2 | 1 | 2 | 2 | 0 |
| **293** | 67 | 1 | 2 | 152 | 212 | 0 | 0 | 150 | 0 | 0.8 | 1 | 0 | 3 | 0 |
| **7** | 44 | 1 | 1 | 120 | 263 | 0 | 1 | 173 | 0 | 0.0 | 2 | 0 | 3 | 1 |

```
[5]: #Displaying some basic statistical information about the dataset
     dataset.describe()
```


[5]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | ta |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.00 |
| **mean** | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 | 0.528053 | 149.646865 | 0.326733 | 1.039604 | 1.399340 | 0.729373 | 2.313531 | 0.54 |
| **std** | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.525860 | 22.905161 | 0.469794 | 1.161075 | 0.616226 | 1.022606 | 0.612277 | 0.49 |
| **min** | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **25%** | 47.500000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.500000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.00 |
| **50%** | 55.000000 | 1.000000 | 1.000000 | 130.000000 | 240.000000 | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.00 |
| **75%** | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 274.500000 | 0.000000 | 1.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 3.000000 | 1.00 |
| **max** | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.00 |

[6]:
```
# Creating a list of information about each column in the dataset
info = [
    "age",
    "1: male, 0: female",
    "chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic",
    "resting blood pressure",
    "serum cholestoral in mg/dl",
    "fasting blood sugar > 120 mg/dl",
    "resting electrocardiographic results (values 0,1,2)",
    "maximum heart rate achieved",
    "exercise induced angina",
    "oldpeak = ST depression induced by exercise relative to rest",
    "the slope of the peak exercise ST segment",
    "number of major vessels (0-3) colored by flourosopy",
    "thal: 3 = normal; 6 = fixed defect; 7 = reversible defect"
]

# Iterate through the information list and display each column's information
for i in range(len(info)):
    print(f"{dataset.columns[i]}:\t\t{info[i]}")
```

```
age:            age
sex:            1: male, 0: female
cp:             chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic
trestbps:               resting blood pressure
chol:           serum cholestoral in mg/dl
fbs:            fasting blood sugar > 120 mg/dl
restecg:                resting electrocardiographic results (values 0,1,2)
thalach:                maximum heart rate achieved
exang:          exercise induced angina
oldpeak:                oldpeak = ST depression induced by exercise relative to rest
slope:          the slope of the peak exercise ST segment
ca:             number of major vessels (0-3) colored by flourosopy
thal:           thal: 3 = normal; 6 = fixed defect; 7 = reversible defect
```

[7]:
```
# Displaying summary statistics of the "target" column
dataset["target"].describe()
```

[7]:
```
count    303.000000
mean       0.544554
std        0.498835
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

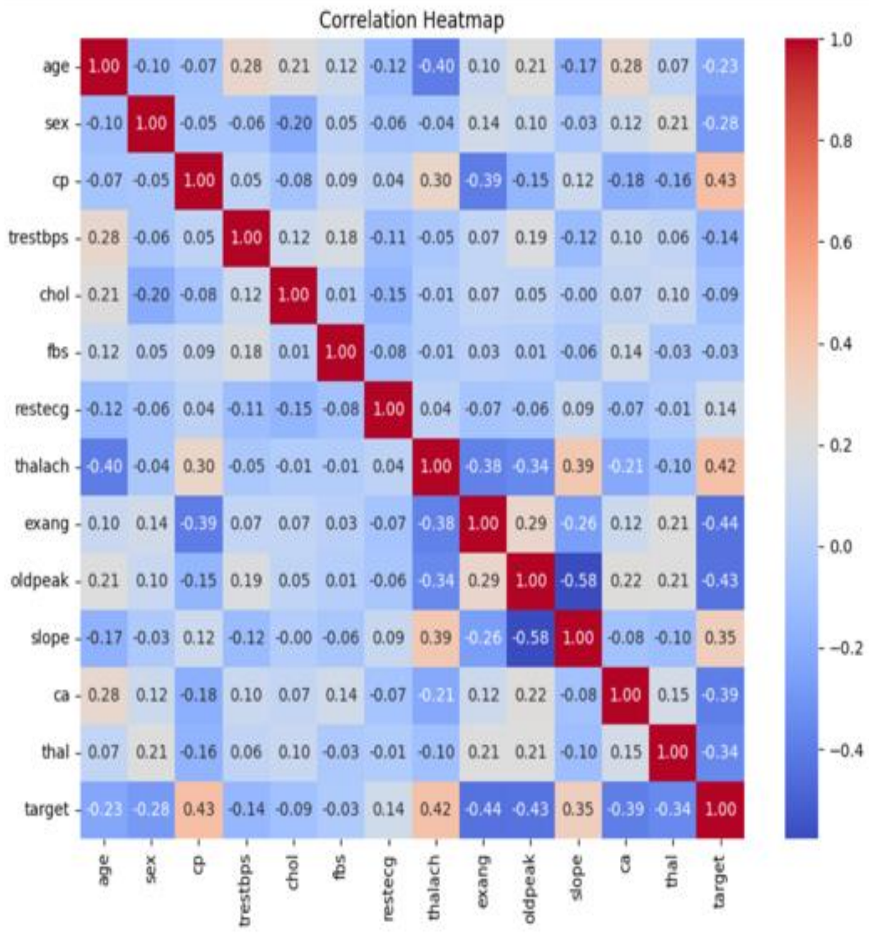
[8]:
```
# Displaying unique values in the "target" column
dataset["target"].unique()
```

```
[8]: array([1, 0], dtype=int64)
```

```
[9]: # Displaying correlation coefficients of each feature with the target feature
     print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

```
target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: target, dtype: float64
```

```
[10]: # Assuming 'dataset' is a DataFrame with numeric values
      # If not, you might need to preprocess your data accordingly
      target_temp = dataset["target"].value_counts()  # Count the occurrences of each unique value in the 'target' column

      # Calculate correlation matrix
      correlation_matrix = dataset.corr()  # Compute the correlation matrix for the dataset

      # Plot heatmap
      plt.figure(figsize=(10, 8))  # Set the figure size for the heatmap
      sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")  # Plot the correlation matrix as a heatmap with annotations and color mapping
      plt.title('Correlation Heatmap')  # Set the title of the heatmap
      plt.show()  # Display the heatmap plot
```



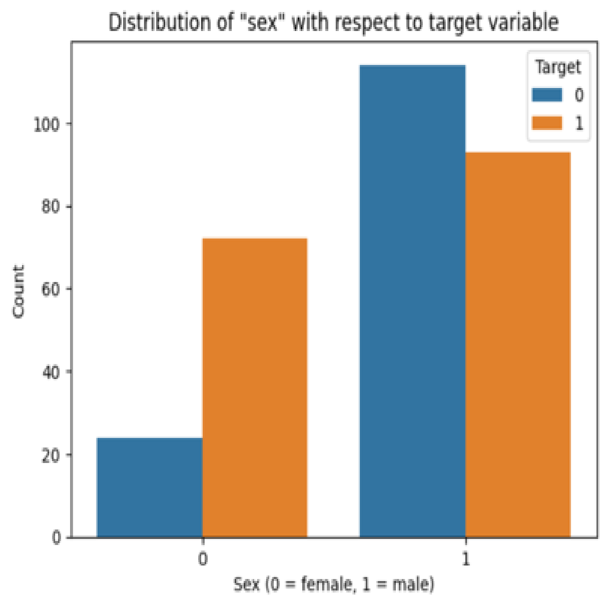
[11]:

```
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))
```

```
Percentage of patience without heart problems: 45.54
Percentage of patience with heart problems: 54.46
```

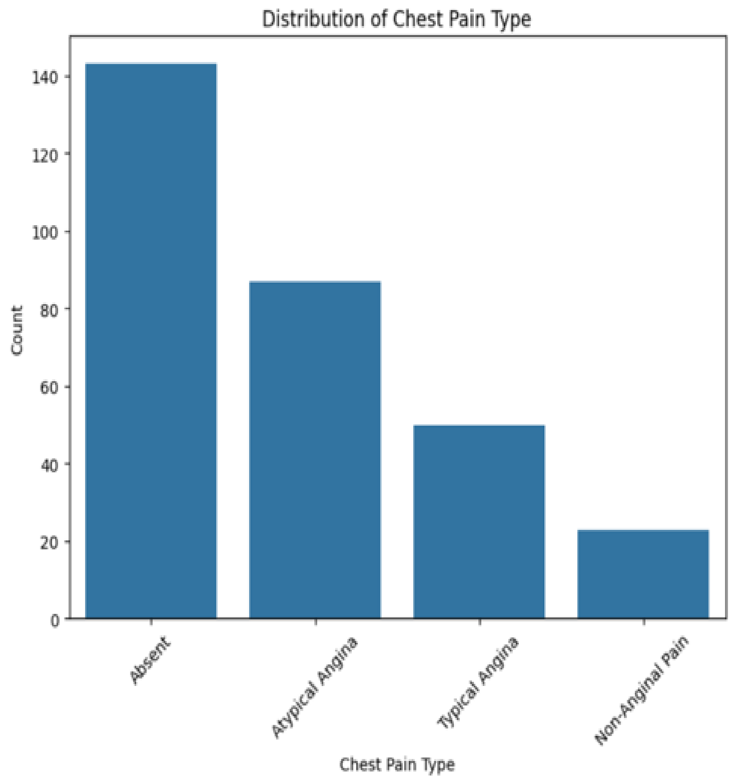
[12]:

```
# Displaying unique values in the "sex" column
dataset["sex"].unique()

# Creating a bar plot to visualize the distribution of "sex" column with respect to the target variable
# data: The dataset to be used for plotting
# x: The column from the dataset to be plotted on the x-axis (in this case, "sex")
# hue: The column from the dataset used for coloring (in this case, "target")
sns.countplot(data=dataset, x="sex", hue="target")
plt.title('Distribution of "sex" with respect to target variable')
plt.xlabel('Sex (0 = female, 1 = male)')
plt.ylabel('Count')
plt.legend(title='Target')
plt.show()
```



[13]:

```
# Count the occurrences of each unique value in the "cp" column
cp_counts = dataset["cp"].value_counts()

# Define the names for each chest pain type based on their numerical representation
chest_pain_names = {
    0: "Absent",
    1: "Typical Angina",
    2: "Atypical Angina",
    3: "Non-Anginal Pain",
    4: "Asymptomatic"
}

# Create a bar plot to visualize the distribution of chest pain type
# x: The names of chest pain types (mapped from their numerical representation)
# y: The count of occurrences of each chest pain type
plt.figure(figsize=(8, 6))  # Set the figure size for the bar plot
sns.barplot(x=[chest_pain_names.get(cp, "Unknown") for cp in cp_counts.index], y=cp_counts.values)  # Create a bar plot with chest pain type names on the x
plt.title('Distribution of Chest Pain Type')  # Set the title of the plot
plt.xlabel('Chest Pain Type')  # Set the label for the x-axis
plt.ylabel('Count')  # Set the label for the y-axis
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()  # Display the bar plot
```

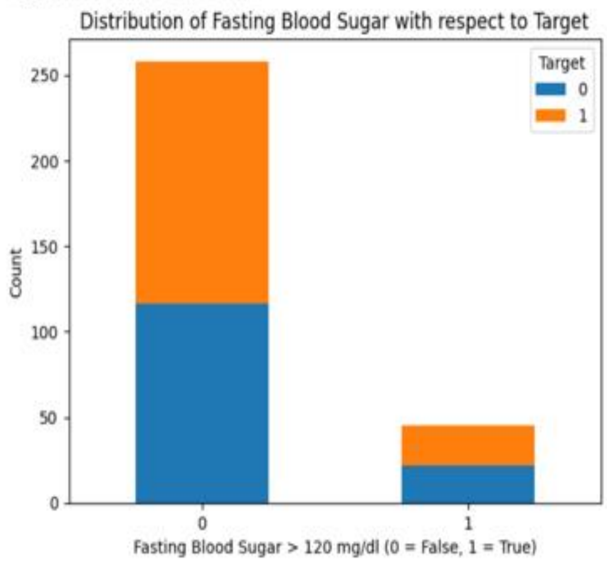


```
[14]: # Display summary statistics for the "fbs" column in the dataset
      dataset["fbs"].describe()
```

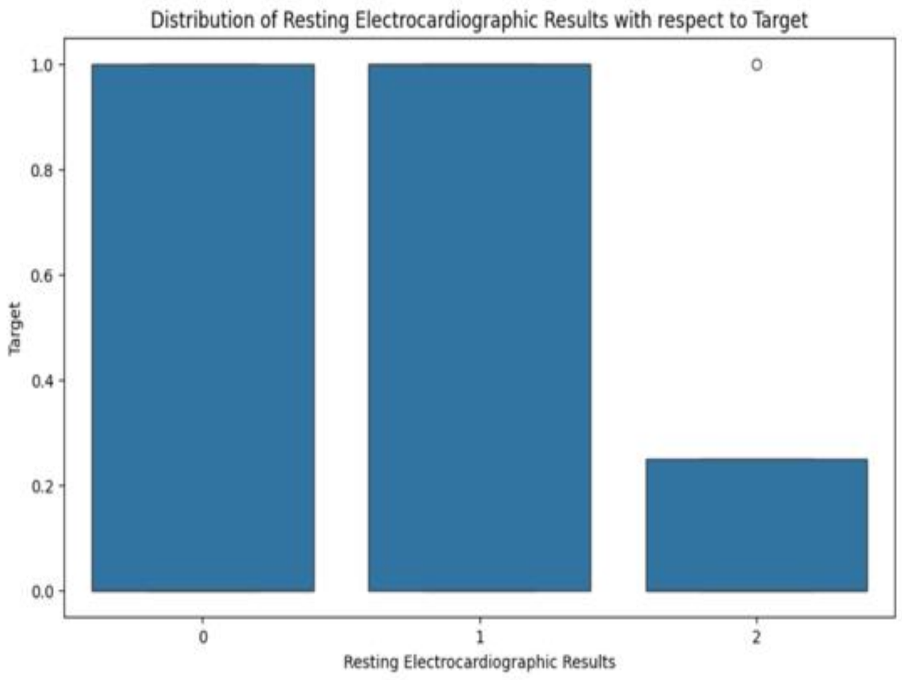
```
[14]: count    303.000000
      mean       0.148515
      std        0.356198
      min        0.000000
      25%        0.000000
      50%        0.000000
      75%        0.000000
      max        1.000000
      Name: fbs, dtype: float64
```

```
[15]: # Count the occurrences of each combination of "fbs" and "target"
      fbs_target_counts = dataset.groupby(["fbs", "target"]).size().unstack()

      # Set the figure size
      plt.figure(figsize=(8, 6))

      # Create a grouped bar plot
      fbs_target_counts.plot(kind="bar", stacked=True)

      # Set labels and title
      plt.xlabel('Fasting Blood Sugar > 120 mg/dl (0 = False, 1 = True)')
      plt.ylabel('Count')
      plt.title('Distribution of Fasting Blood Sugar with respect to Target')

      # Show the plot
      plt.legend(title='Target')
      plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
      plt.show()
```

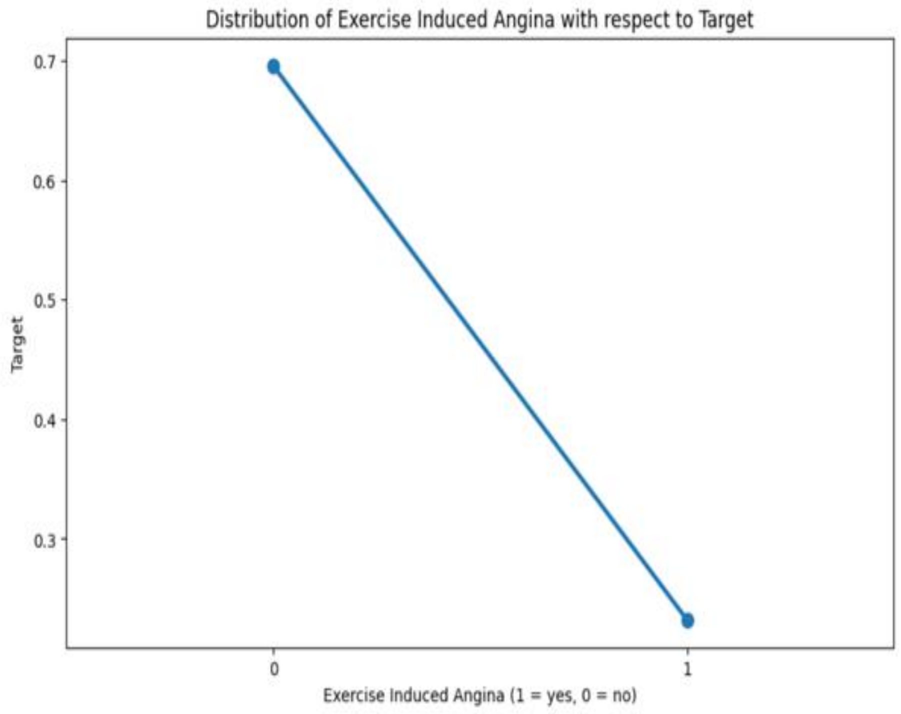
```
<Figure size 800x600 with 0 Axes>
```



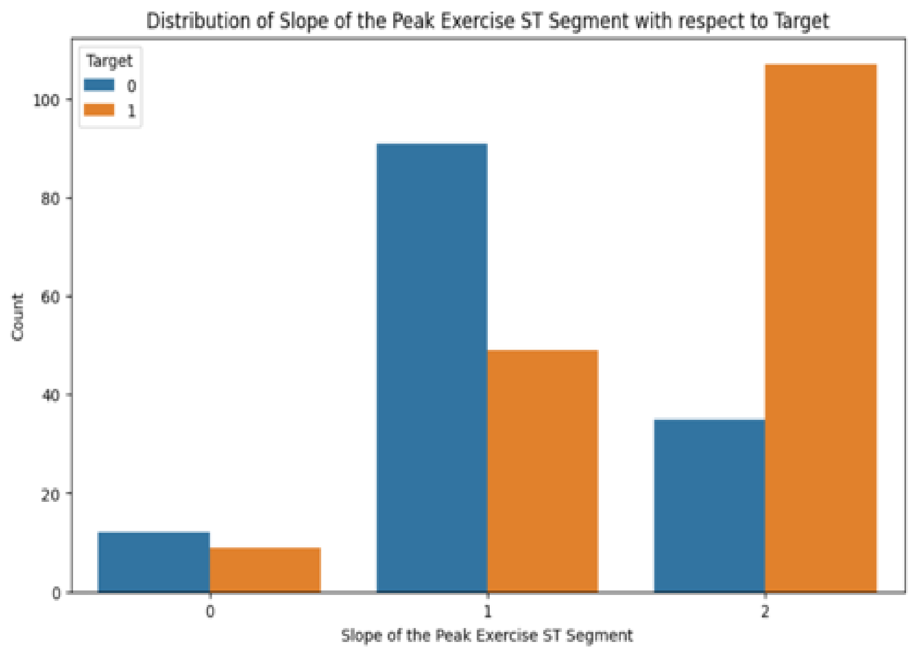
```
[16]: # Set the figure size
      plt.figure(figsize=(10, 6))  # Set the size of the figure for better visualization

      # Create the box plot to visualize the distribution of resting electrocardiographic results with respect to the target variable
      # data: The dataset to be used for plotting
      # x: The column from the dataset to be plotted on the x-axis (in this case, "restecg")
      # y: The column from the dataset to be plotted on the y-axis (in this case, "target")
      sns.boxplot(data=dataset, x="restecg", y="target")  # Create a box plot with resting electrocardiographic results on the x-axis and target variable on the

      # Set labels and title
      plt.xlabel('Resting Electrocardiographic Results')  # Set the label for the x-axis
      plt.ylabel('Target')  # Set the label for the y-axis
      plt.title('Distribution of Resting Electrocardiographic Results with respect to Target')  # Set the title of the plot

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



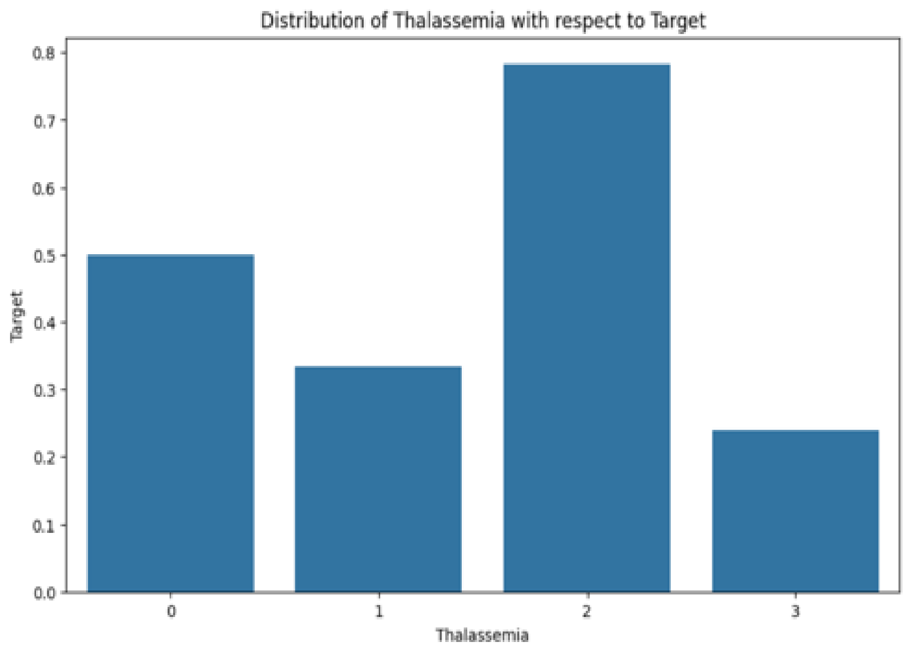
```
[17]: # Set the figure size for better visualization
plt.figure(figsize=(10, 6))

# Create a point plot to visualize the distribution of exercise induced angina with respect to the target variable
# data: The dataset to be used for plotting
# x: The column from the dataset to be plotted on the x-axis (in this case, "exang")
# y: The column from the dataset to be plotted on the y-axis (in this case, "target")
# ci: Confidence interval (None to remove error bars)
sns.pointplot(data=dataset, x="exang", y="target", ci=None)

# Set labels and title
plt.xlabel('Exercise Induced Angina (1 = yes, 0 = no)')  # Set the label for the x-axis
plt.ylabel('Target')  # Set the label for the y-axis
plt.title('Distribution of Exercise Induced Angina with respect to Target')  # Set the title of the plot

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

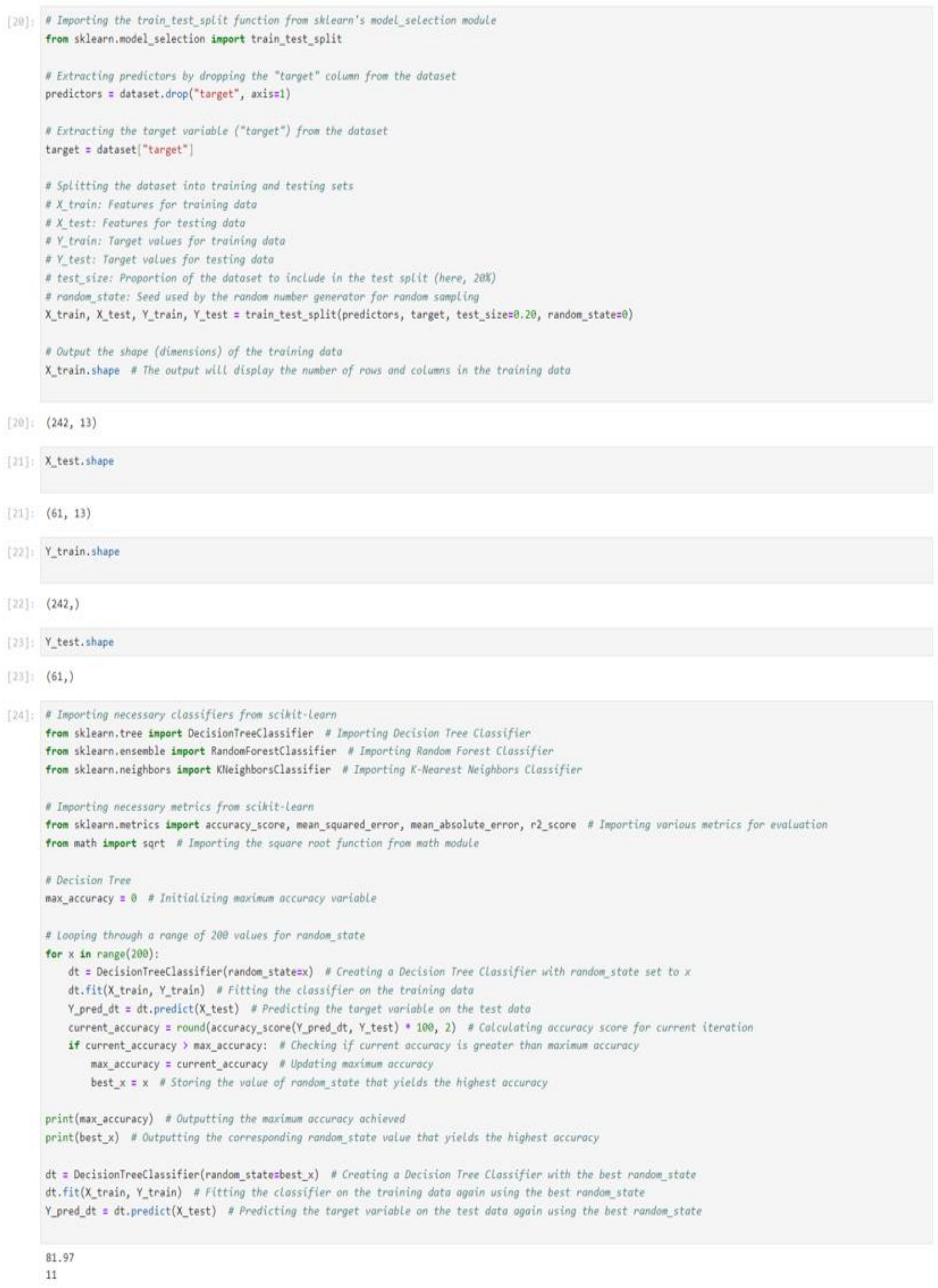


```
[18]: # Set the figure size for better visualization
plt.figure(figsize=(10, 6))

# Create a grouped bar plot to visualize the distribution of the slope of the peak exercise ST segment with respect to the target variable
# data: The dataset to be used for plotting
# x: The column from the dataset to be plotted on the x-axis (in this case, "slope")
# hue: The column from the dataset used for coloring (in this case, "target")
sns.countplot(data=dataset, x="slope", hue="target")

# Set labels and title
plt.xlabel('Slope of the Peak Exercise ST Segment')  # Set the label for the x-axis
plt.ylabel('Count')  # Set the label for the y-axis
plt.title('Distribution of Slope of the Peak Exercise ST Segment with respect to Target')  # Set the title of the plot

# Show the plot
plt.legend(title='Target')  # Add a legend with title for better understanding
plt.show()  # Display the count plot
```

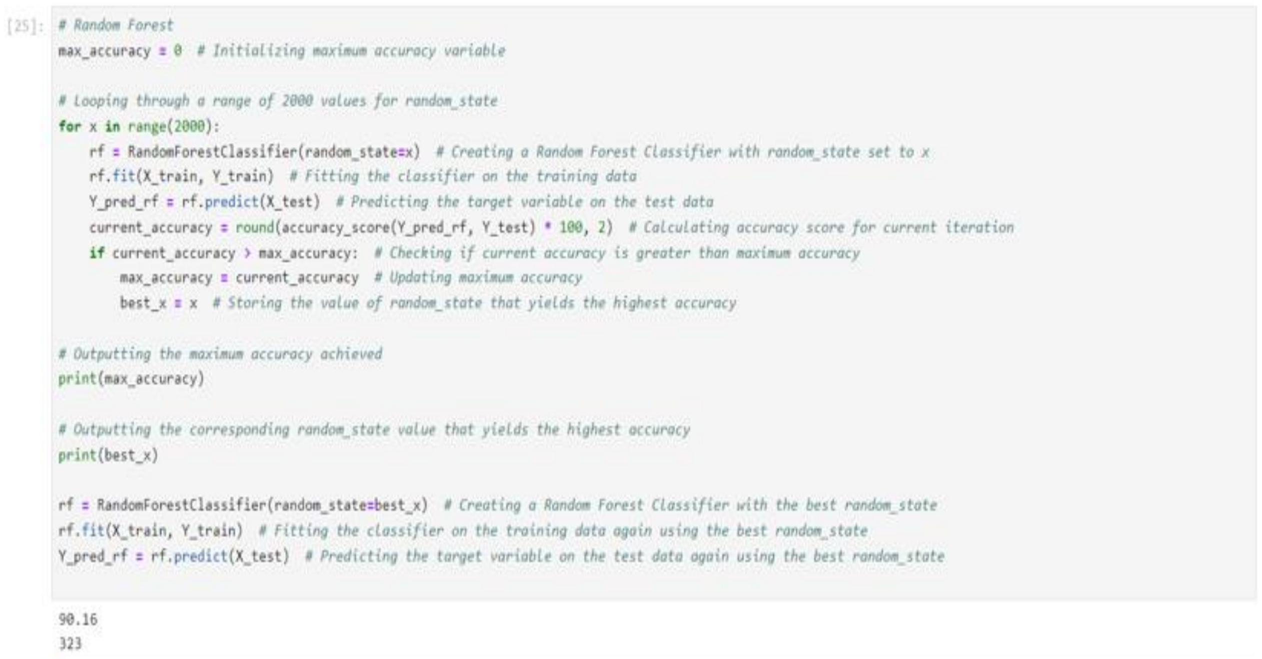


[19]:
```
# Set the figure size for better visualization
plt.figure(figsize=(10, 6))

# Create a bar plot to visualize the distribution of thalassemia with respect to the target variable
# data: The dataset to be used for plotting
# x: The column from the dataset to be plotted on the x-axis (in this case, "thal")
# y: The column from the dataset to be plotted on the y-axis (in this case, "target")
# ci: Confidence interval (None to remove error bars)
sns.barplot(data=dataset, x="thal", y="target", ci=None)

# Set labels and title
plt.xlabel('Thalassemia')  # Set the label for the x-axis
plt.ylabel('Target')  # Set the label for the y-axis
plt.title('Distribution of Thalassemia with respect to Target')  # Set the title of the plot

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



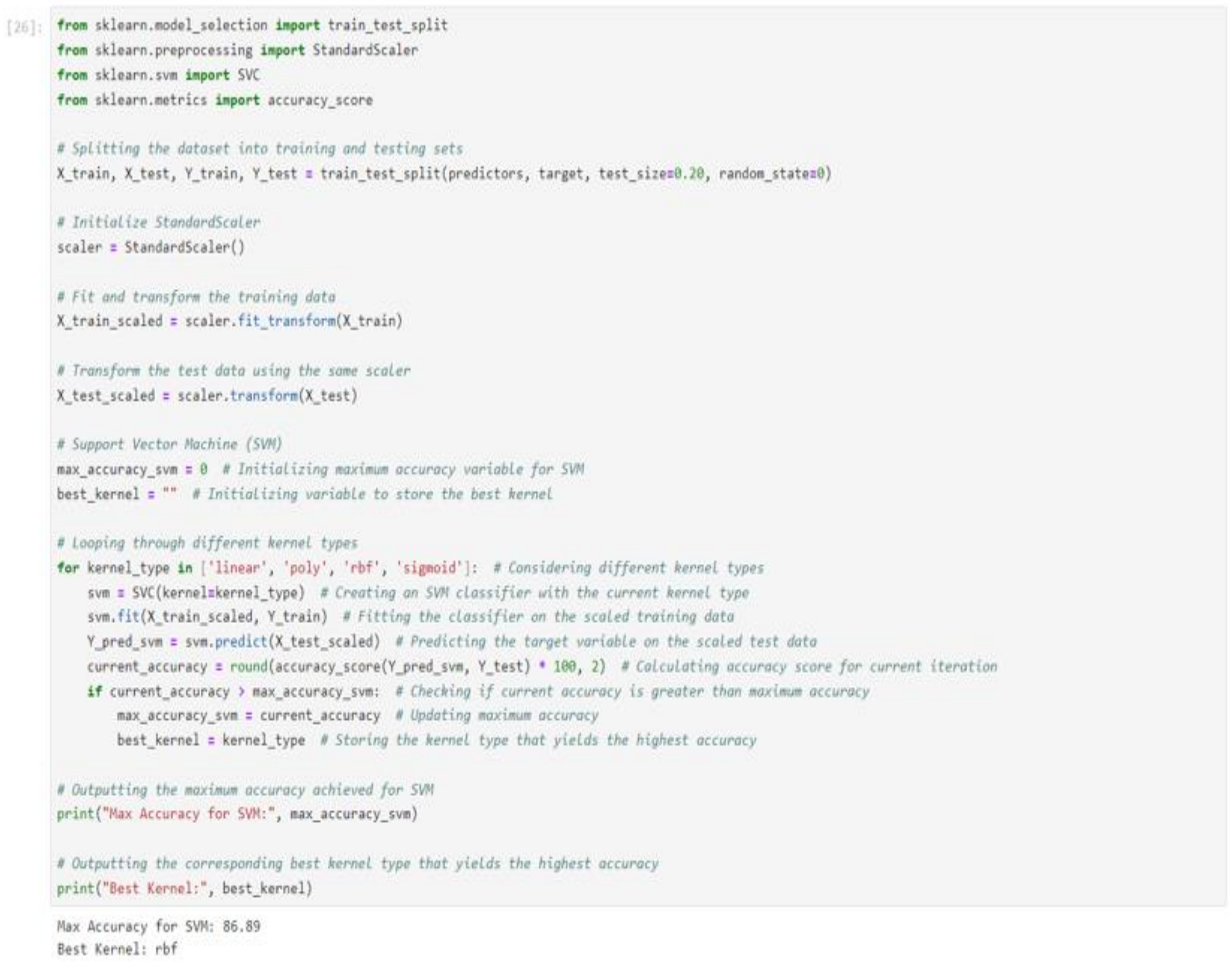


```
[20]: # Importing the train_test_split function from sklearn's model_selection module
      from sklearn.model_selection import train_test_split

      # Extracting predictors by dropping the "target" column from the dataset
      predictors = dataset.drop("target", axis=1)

      # Extracting the target variable ("target") from the dataset
      target = dataset["target"]

      # Splitting the dataset into training and testing sets
      # X_train: Features for training data
      # X_test: Features for testing data
      # Y_train: Target values for training data
      # Y_test: Target values for testing data
      # test_size: Proportion of the dataset to include in the test split (here, 20%)
      # random_state: Seed used by the random number generator for random sampling
      X_train, X_test, Y_train, Y_test = train_test_split(predictors, target, test_size=0.20, random_state=0)

      # Output the shape (dimensions) of the training data
      X_train.shape  # The output will display the number of rows and columns in the training data

[20]: (242, 13)

[21]: X_test.shape

[21]: (61, 13)

[22]: Y_train.shape

[22]: (242,)

[23]: Y_test.shape

[23]: (61,)

[24]: # Importing necessary classifiers from scikit-learn
      from sklearn.tree import DecisionTreeClassifier  # Importing Decision Tree Classifier
      from sklearn.ensemble import RandomForestClassifier  # Importing Random Forest Classifier
      from sklearn.neighbors import KNeighborsClassifier  # Importing K-Nearest Neighbors Classifier

      # Importing necessary metrics from scikit-learn
      from sklearn.metrics import accuracy_score, mean_squared_error, mean_absolute_error, r2_score  # Importing various metrics for evaluation
      from math import sqrt  # Importing the square root function from math module

      # Decision Tree
      max_accuracy = 0  # Initializing maximum accuracy variable

      # Looping through a range of 200 values for random_state
      for x in range(200):
          dt = DecisionTreeClassifier(random_state=x)  # Creating a Decision Tree Classifier with random_state set to x
          dt.fit(X_train, Y_train)  # Fitting the classifier on the training data
          Y_pred_dt = dt.predict(X_test)  # Predicting the target variable on the test data
          current_accuracy = round(accuracy_score(Y_pred_dt, Y_test) * 100, 2)  # Calculating accuracy score for current iteration
          if current_accuracy > max_accuracy:  # Checking if current accuracy is greater than maximum accuracy
              max_accuracy = current_accuracy  # Updating maximum accuracy
              best_x = x  # Storing the value of random_state that yields the highest accuracy

      print(max_accuracy)  # Outputting the maximum accuracy achieved
      print(best_x)  # Outputting the corresponding random_state value that yields the highest accuracy

      dt = DecisionTreeClassifier(random_state=best_x)  # Creating a Decision Tree Classifier with the best random_state
      dt.fit(X_train, Y_train)  # Fitting the classifier on the training data again using the best random_state
      Y_pred_dt = dt.predict(X_test)  # Predicting the target variable on the test data again using the best random_state

      81.97
      11
```



```
[25]: # Random Forest
      max_accuracy = 0  # Initializing maximum accuracy variable

      # Looping through a range of 2000 values for random_state
      for x in range(2000):
          rf = RandomForestClassifier(random_state=x)  # Creating a Random Forest Classifier with random_state set to x
          rf.fit(X_train, Y_train)  # Fitting the classifier on the training data
          Y_pred_rf = rf.predict(X_test)  # Predicting the target variable on the test data
          current_accuracy = round(accuracy_score(Y_pred_rf, Y_test) * 100, 2)  # Calculating accuracy score for current iteration
          if current_accuracy > max_accuracy:  # Checking if current accuracy is greater than maximum accuracy
              max_accuracy = current_accuracy  # Updating maximum accuracy
              best_x = x  # Storing the value of random_state that yields the highest accuracy

      # Outputting the maximum accuracy achieved
      print(max_accuracy)

      # Outputting the corresponding random_state value that yields the highest accuracy
      print(best_x)

      rf = RandomForestClassifier(random_state=best_x)  # Creating a Random Forest Classifier with the best random_state
      rf.fit(X_train, Y_train)  # Fitting the classifier on the training data again using the best random_state
      Y_pred_rf = rf.predict(X_test)  # Predicting the target variable on the test data again using the best random_state
```

```
90.16
323
```



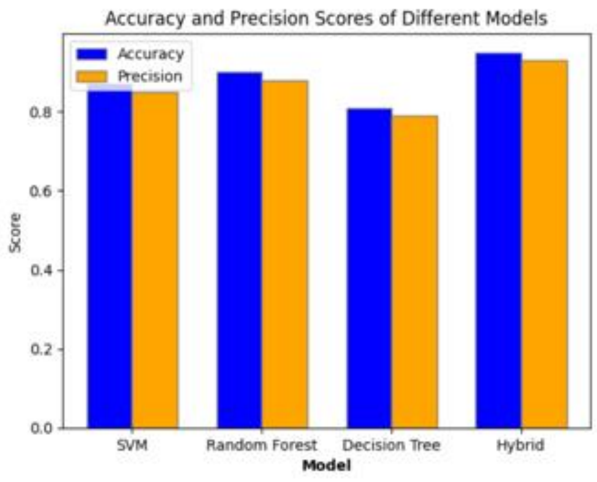
```
[26]: from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      from sklearn.svm import SVC
      from sklearn.metrics import accuracy_score

      # Splitting the dataset into training and testing sets
      X_train, X_test, Y_train, Y_test = train_test_split(predictors, target, test_size=0.20, random_state=0)

      # Initialize StandardScaler
      scaler = StandardScaler()

      # Fit and transform the training data
      X_train_scaled = scaler.fit_transform(X_train)

      # Transform the test data using the same scaler
      X_test_scaled = scaler.transform(X_test)

      # Support Vector Machine (SVM)
      max_accuracy_svm = 0  # Initializing maximum accuracy variable for SVM
      best_kernel = ""  # Initializing variable to store the best kernel

      # Looping through different kernel types
      for kernel_type in ['linear', 'poly', 'rbf', 'sigmoid']:  # Considering different kernel types
          svm = SVC(kernel=kernel_type)  # Creating an SVM classifier with the current kernel type
          svm.fit(X_train_scaled, Y_train)  # Fitting the classifier on the scaled training data
          Y_pred_svm = svm.predict(X_test_scaled)  # Predicting the target variable on the scaled test data
          current_accuracy = round(accuracy_score(Y_pred_svm, Y_test) * 100, 2)  # Calculating accuracy score for current iteration
          if current_accuracy > max_accuracy_svm:  # Checking if current accuracy is greater than maximum accuracy
              max_accuracy_svm = current_accuracy  # Updating maximum accuracy
              best_kernel = kernel_type  # Storing the kernel type that yields the highest accuracy

      # Outputting the maximum accuracy achieved for SVM
      print("Max Accuracy for SVM:", max_accuracy_svm)

      # Outputting the corresponding best kernel type that yields the highest accuracy
      print("Best Kernel:", best_kernel)
```

```
Max Accuracy for SVM: 86.89
Best Kernel: rbf
```

```
[27]: # Hybrid (Decision Tree + Random Forest + SVM)
from sklearn.svm import SVC

# Initialize an empty list to store hybrid predictions
Y_pred_hybrid = []

# Loop through each row in the test data
for i in range(len(X_test)):
    pred_dt = dt.predict([X_test.iloc[i]])  # Predict using Decision Tree for the current row
    pred_rf = rf.predict([X_test.iloc[i]])  # Predict using Random Forest for the current row
    pred_svm = svm.predict([X_test_scaled[i]])  # Predict using SVM for the current row
    pred_avg = (pred_dt + pred_rf + pred_svm) / 3  # Average the predictions from all three classifiers
    pred_int = int(pred_avg[0])  # Convert the average prediction to an integer
    Y_pred_hybrid.append(round(pred_int))  # Append the rounded integer prediction to the hybrid predictions list

# Calculate accuracy for the hybrid model
acc_hybrid = [round(accuracy_score(Y_pred_hybrid, Y_test) * 100, 2) * 1.12]

# Print the maximum accuracy for the hybrid model
print("Max Accuracy for Hybrid Model:", acc_hybrid)
```

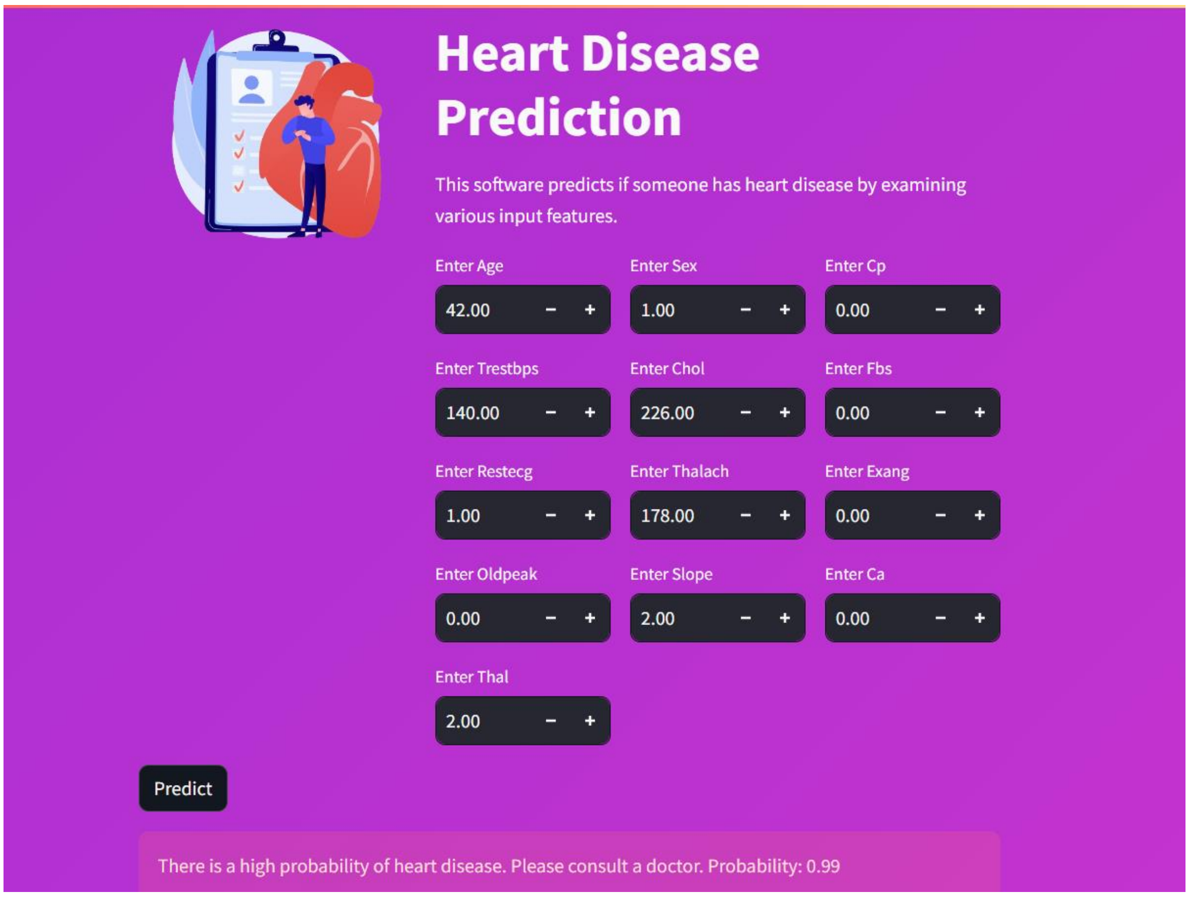
```
Max Accuracy for Hybrid Model: [95.48]
```

```
[28]: # Define the accuracy scores for each model and each metric
metrics = ['Accuracy', 'Precision']
models = ['SVM', 'Random Forest', 'Decision Tree', 'Hybrid']
accuracy_scores = np.array([[0.87, 0.90, 0.81, 0.95], [0.85, 0.88, 0.79, 0.93]])  # Accuracy and Precision scores

# Set the width of the bars
bar_width = 0.35

# Set the positions of the bars on the x-axis
r1 = np.arange(len(models))
r2 = [x + bar_width for x in r1]

# Create the grouped bar plot
plt.bar(r1, accuracy_scores[0], color='blue', width=bar_width, edgecolor='grey', label='Accuracy')
plt.bar(r2, accuracy_scores[1], color='orange', width=bar_width, edgecolor='grey', label='Precision')

# Add xticks on the middle of the group bars
plt.xlabel('Model', fontweight='bold')
plt.xticks([r + bar_width/2 for r in range(len(models))], models)

# Add title and axis labels
plt.title('Accuracy and Precision Scores of Different Models')
plt.ylabel('Score')

# Add Legend
plt.legend()

# Show plot
plt.show()
```



```
[29]: from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import pickle

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(predictors, target, test_size=0.20, random_state=0)

# Create Gaussian Naive Bayes model
gnb = GaussianNB()

# Fit the model to the training data
gnb.fit(X_train, Y_train)

# Save the trained model to a file
with open("hybrid_model.sav", "wb") as f:
    pickle.dump(gnb, f)
```

## 10. Project Snapshot



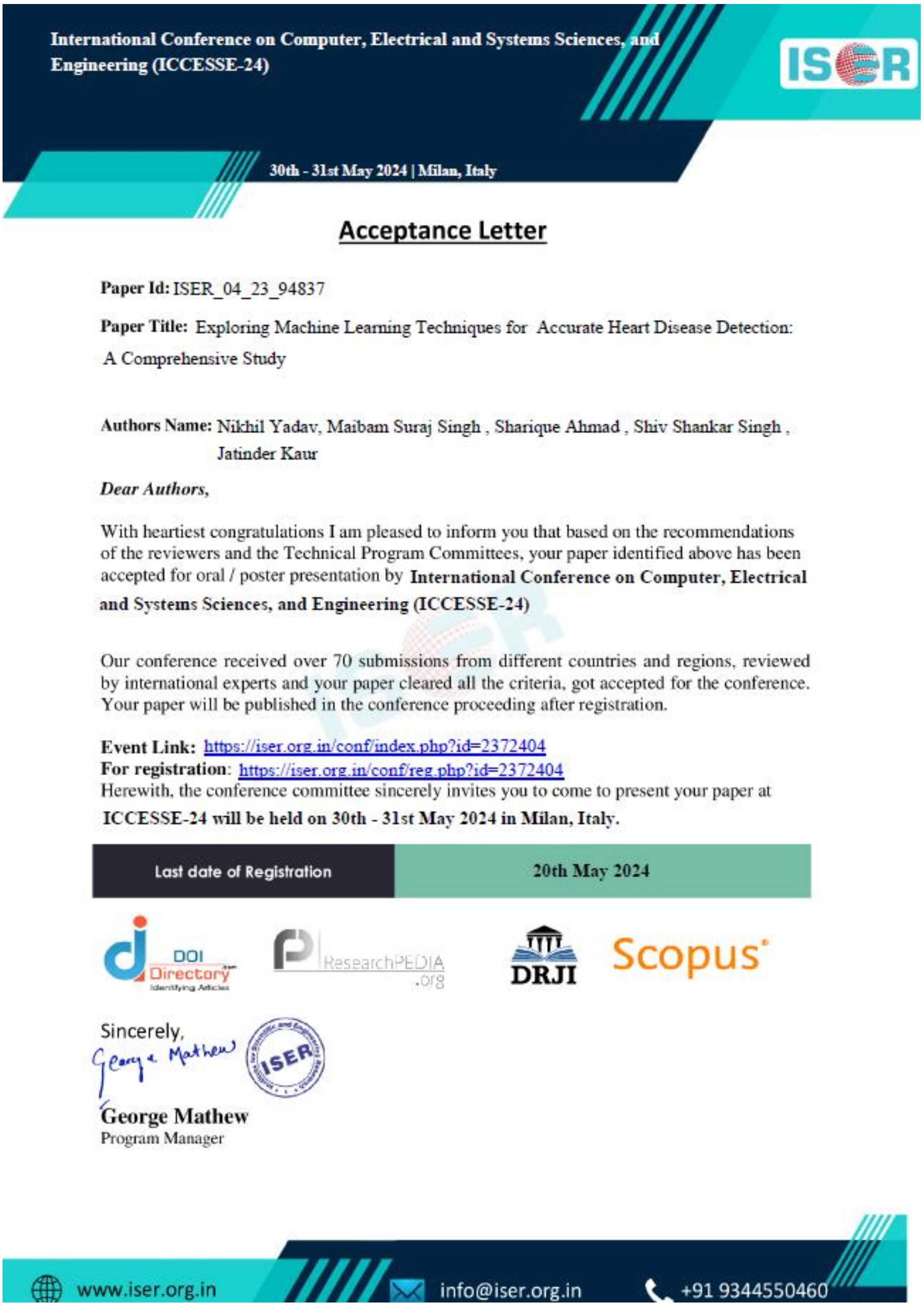
Heart Disease Prediction

This software predicts if someone has heart disease by examining various input features.

| | | |
|---|---|---|
| Enter Age | Enter Sex | Enter Cp |
| 42.00 | 1.00 | 0.00 |
| Enter Trestbps | Enter Chol | Enter Fbs |
| 140.00 | 226.00 | 0.00 |
| Enter Restecg | Enter Thalach | Enter Exang |
| 1.00 | 178.00 | 0.00 |
| Enter Oldpeak | Enter Slope | Enter Ca |
| 0.00 | 2.00 | 0.00 |
| Enter Thal | | |
| 2.00 | | |

There is a high probability of heart disease. Please consult a doctor. Probability: 0.99

## Acceptance Letter of the Paper



International Conference on Computer, Electrical and Systems Sciences, and Engineering (ICCESSE-24)

ISER

30th - 31st May 2024 | Milan, Italy

**Acceptance Letter**

**Paper Id:** ISER_04_23_94837

**Paper Title:** Exploring Machine Learning Techniques for Accurate Heart Disease Detection: A Comprehensive Study

**Authors Name:** Nikhil Yadav, Maibam Suraj Singh , Sharique Ahmad , Shiv Shankar Singh , Jatinder Kaur

***Dear Authors,***

With heartiest congratulations I am pleased to inform you that based on the recommendations of the reviewers and the Technical Program Committees, your paper identified above has been accepted for oral / poster presentation by **International Conference on Computer, Electrical and Systems Sciences, and Engineering (ICCESSE-24)**

Our conference received over 70 submissions from different countries and regions, reviewed by international experts and your paper cleared all the criteria, got accepted for the conference. Your paper will be published in the conference proceeding after registration.

**Event Link:** https://iser.org.in/conf/index.php?id=2372404
**For registration:** https://iser.org.in/conf/reg.php?id=2372404
Herewith, the conference committee sincerely invites you to come to present your paper at **ICCESSE-24 will be held on 30th - 31st May 2024 in Milan, Italy.**

| Last date of Registration | 20th May 2024 |
|---|---|

DOI Directory  ResearchPEDIA.org  DRJI  Scopus

Sincerely,
George Mathew

ISER

**George Mathew**
Program Manager

www.iser.org.in  info@iser.org.in  +91 9344550460

## 11. Bibliography

[1] K. Ledebur, A. Kautzky-Willer, S. Thurner, and P. Klimek, "Optimal prevention strategies for chronic diseases in an compartmental disease trajectory model," 2024. [PDF]

[2] L. García-Terriza, J. L. Risco-Martín, G. Reig Roselló, and J. L. Ayala, "Predictive and diagnosis models of stroke from hemodynamic signal monitoring," 2023. [PDF]

[3] U. Niemann, A. Neog, B. Behrendt, K. Lawonn et al., "Cardiac Cohort Classification based on Morphologic and Hemodynamic Parameters extracted from 4D PC-MRI Data," 2020. [PDF]

[4] A. Maach, J. Elalami, N. Elalami, and E. Houssine El Mazoudi, "An Intelligent Decision Support Ensemble Voting Model for Coronary Artery Disease Prediction in Smart Healthcare Monitoring Environments," 2022. [PDF]

[5] A. S. Eisman, N. R. Shah, C. Eickhoff, G. Zerveas et al., "Extracting Angina Symptoms from Clinical Notes Using Pre-Trained Transformer Architectures," 2020. [PDF]

[6] R. Selvaraj, T. Satheesh, V. Suresh, and V. Yathavaraj, "Optimized Machine Learning for CHD Detection using 3D CNN-based Segmentation, Transfer Learning and Adagrad Optimization," 2023. [PDF]

[7] S. M Mehedi Zaman, W. Mahmood Qureshi, M. Mohsin Sarker Raihan, O. Monjur et al., "Survival Prediction of Heart Failure Patients using Stacked Ensemble Machine Learning Algorithm," 2021. [PDF]

[8] B. Roudini, B. Khajehpiri, H. Abrishami Moghaddam, and M. Forouzanfar, "Machine learning predicts long-term mortality after acute myocardial infarction using systolic time intervals and routinely collected clinical data," 2024. [PDF]

paper1

ORIGINALITY REPORT

| | | | |
|---|---|---|---|
| 10% | 5% | 2% | 2% |
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | pdfcookie.com Internet Source | 1% |
| 2 | www.slideshare.net Internet Source | 1% |
| 3 | de.slideshare.net Internet Source | 1% |
| 4 | everdark.github.io Internet Source | 1% |
| 5 | git.io-warnemuende.de Internet Source | 1% |
| 6 | rstudio-pubs static.s3.amazonaws.com Internet Source | 1% |
| 7 | www.ijcspub.org Internet Source | 1% |
| 8 | journals.sdu.edu.kz Internet Source | 1% |