# paper1

*by* Jatinder Kaur

**Submission date:** 02-May-2024 08:56PM (UTC+0530)
**Submission ID:** 2368903746
**File name:** KC341.pdf (3.37M)
**Word count:** 6278
**Character count:** 40069

**CAPSTONE PROJECT REPORT**

(Project Term January-May 2024)

# Exploring Machine Learning Techniques for Accurate Heart Disease Detection

Submitted by


| | |
|---|---|
| **Nikhil Yadav** | **12015390** |
| **Maibam Suraj Singh** | **12002241** |
| **Sharique Ahmad** | **12105943** |
| **Shiv Shankar Singh** | **12018635** |


**Project Group Number: CSERGC0341**

**Course Code: CSE445**

Under the Guidance of

**Jatinder Kaur (Assistant Professor)**

School of Computer Science and Engineering

**TOPIC APPROVAL PERFORMA**

School of Computer Science and Engineering (SCSE)

**Program :** P132::B.Tech. (Computer Science and Engineering)

**COURSE CODE :** CSE445 **REGULAR/BACKLOG :** Regular **GROUP NUMBER :** CSERGC0341

**Supervisor Name :** Jatinder Kaur **UID :** 29755 **Designation :** Assistant Professor

**Qualification :** _____ **Research Experience :** _____

| SR.NO. | NAME OF STUDENT | Prov. Regd. No. | BATCH | SECTION | CONTACT NUMBER |
|---|---|---|---|---|---|
| 1 | Nikhil Yadav | 12015390 | 2020 | K20AR | 9050543344 |
| 2 | Shiv Shankar Singh | 12018635 | 2020 | K20CT | 6392592680 |
| 3 | Sharique Ahmad | 12105943 | 2021 | K20MD | 8092476009 |
| 4 | Maibam Suraj Singh | 12002241 | 2020 | K20CT | 9592113464 |

**SPECIALIZATION AREA :** Networking and Security-I **Supervisor Signature:** _____

**PROPOSED TOPIC :** Heart Disease Detection

| Qualitative Assessment of Proposed Topic by PAC | | |
|---|---|---|
| **Sr.No.** | **Parameter** | **Rating (out of 10)** |
| 1 | Project Novelty: Potential of the project to create new knowledge | 6.60 |
| 2 | Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students. | 6.67 |
| 3 | Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program. | 6.53 |
| 4 | Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills. | 6.87 |
| 5 | Social Applicability: Project work intends to solve a practical problem. | 6.20 |
| 6 | Future Scope: Project has potential to become basis of future research work, publication or patent. | 6.20 |

| PAC Committee Members | | |
|---|---|---|
| PAC Member (HOD/Chairperson) Name: Dr. Harwant Singh Arri | UID: 12975 | Recommended (Y/N): Yes |
| PAC Member (Allied) Name: Dr. Vishu | UID: 18807 | Recommended (Y/N): Yes |
| PAC Member 3 Name: Dr. Balraj Singh | UID: 13075 | Recommended (Y/N): Yes |

**Final Topic Approved by PAC:** **Heart Disease Detection**

**Overall Remarks:** Approved

**PAC CHAIRPERSON Name:** 13897::Dr. Deepak Prashar **Approval Date:** 12 Apr 2024

# DECLARATION

We hereby declare that the project work entitled "Exploring Predictive Models for Heart Disease Detection" is an authentic record of our own work conducted as a requirement of the Capstone Project for the award of a B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara. This project was carried out under the guidance of Jatinder Kaur, during the period from January to May 2024.

Project Group Number: CSERGC0341

Name of Student 1: Nikhil Yadav Signature:
Registration Number: 12015390 Date:

Name of Student 2: Shiv Shankar Singh Signature:
Registration Number: 12018635 Date:

Name of Student 3: Sharique Ahmad Signature:
Registration Number: 12105943 Date:

Name of Student 4: Maibam Suraj Singh Signature:
Registration Number: 12002241 Date:

Date : 20-April-2024

# CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Signature

**Name of Mentor: Jatinder Kaur**

**School of Computer Science & Engineering**

Lovely Professional University,

Phagwara, Punjab

Date: 20-April-2024

## ACKNOWLEDGEMENT

We extend our sincere appreciation to Ms. Jatinder Kaur, our esteemed teacher, for entrusting us with the opportunity to undertake this insightful project on " Exploring Predictive Models for Heart Disease Detection " Ms. Jatinder's guidance and encouragement were invaluable throughout the project, motivating us to achieve excellence.

Undertaking this project enabled us to delve into extensive research, broadening our understanding of various facets of the subject matter. We are grateful to everyone who contributed to the successful completion of this endeavor within the given time constraints.

Our heartfelt thanks go to all the faculty members whose unwavering support and assistance played a significant role in shaping this project. Their expertise and willingness to share knowledge enriched our learning experience and facilitated the accomplishment of our tasks.

We express our gratitude to the Head of the Department for their continuous encouragement and support throughout this journey.

Furthermore, we would like to extend our appreciation to Lovely Professional University for providing us with the platform to pursue our academic endeavors.

Thank you to all who provided encouragement and support, making this project a rewarding and enriching experience for us.

Nikhil Yadav

Shiv Shankar Singh

Sharique Ahmad

Maibam Suraj Singh

## 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 Identifying Articles · ResearchPEDIA .org · DRJI · Scopus

Sincerely,

**George Mathew**
Program Manager

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

# TABLE OF CONTENTS

# 1. INTRODUCTION

Cardiovascular diseases (CVDs) stand as a predominant cause of mortality globally, posing a significant burden on public health systems and economies worldwide. Despite advancements in medical science and healthcare, the prevalence of CVDs continues to rise, necessitating proactive measures for prevention, early detection, and intervention. In response to this imperative, the integration of data-driven approaches, particularly machine learning, has emerged as a promising avenue for enhancing cardiovascular risk assessment and improving patient outcomes.

The aim of this project is 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 analyzing a dataset comprising demographic information, medical history, lifestyle factors, and diagnostic indicators, we endeavor to construct models that can accurately stratify patients into different risk categories, thereby facilitating targeted interventions and personalized healthcare strategies.

The significance of this endeavor lies in its potential to revolutionize the landscape of cardiovascular care by providing clinicians with actionable insights derived from data-driven analyses. Traditional risk assessment tools often rely on limited sets of risk factors and may not fully capture the complex interplay of variables influencing cardiovascular health. In contrast, machine learning algorithms have the capacity to uncover intricate patterns and relationships within vast datasets, enabling the development of more robust and precise risk prediction models.

Through a combination of rigorous data preprocessing, exploratory data analysis, feature engineering, 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. By leveraging the wealth of information embedded within healthcare datasets, we aspire to advance the frontiers of cardiovascular risk assessment, preventive medicine, and population health management, ultimately striving towards a future where the incidence and impact of CVDs are significantly mitigated. 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) stand as a formidable global health challenge, exacting a heavy toll on individuals and healthcare systems worldwide. These diseases, encompassing a range of conditions affecting the heart and blood vessels, not only contribute significantly to morbidity and mortality rates but also impose substantial economic burdens on societies. Despite advances in medical science and healthcare delivery, the timely and accurate detection of CVDs remains a persistent challenge, often leading to delayed interventions, suboptimal treatment outcomes, and increased healthcare costs.

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.

The scope of our study encompasses a multidisciplinary exploration of machine learning techniques for heart disease detection, utilizing a curated dataset derived from diverse sources. We endeavor to systematically evaluate the performance of various ML algorithms, including Decision Trees, Random Forests, and K-Nearest Neighbors, as well as hybrid ensemble models. Through rigorous experimentation, cross-validation, and performance evaluation, we aspire to elucidate the strengths and

limitations of each algorithm and identify the optimal approach for heart disease detection.

# 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 comprehensive digital repositories that store patient health information, including medical history, diagnostic test results, medication records, and treatment plans. EHR systems facilitate the centralization and accessibility of patient data, enabling healthcare providers to make informed decisions about patient care.
2. Clinical Decision Support Systems (CDSS): CDSS are software applications designed to assist healthcare providers in clinical decision-making by providing evidence-based recommendations, alerts, and reminders. These systems leverage algorithms to analyze patient data and generate personalized treatment plans, risk assessments, and diagnostic suggestions.
3. Medical Imaging Software: Medical imaging software encompasses a range of applications used for interpreting and analyzing medical images, such as X-rays, CT scans, MRI scans, and echocardiograms. These tools enable healthcare professionals to visualize anatomical structures, identify abnormalities, and assess disease progression.
4. Risk Assessment Tools: Various risk assessment tools and calculators are available to estimate an individual's risk of developing cardiovascular diseases. These tools incorporate demographic information, clinical parameters (e.g., blood pressure, cholesterol levels), and lifestyle factors to generate personalized risk scores and recommendations.

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's the System 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: The system will incorporate state-of-the-art machine learning algorithms, such as Decision Trees, Random Forests, and K-Nearest Neighbors, to analyze clinical data and identify patterns indicative of heart disease. These algorithms will enable automated pattern recognition, risk stratification, and predictive modeling, enhancing the accuracy and efficiency of heart disease detection.

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: The system will provide real-time decision support to healthcare professionals, offering evidence-based recommendations, alerts, and reminders at the point of care. By integrating predictive analytics and clinical guidelines into the workflow, the system will assist providers in making informed decisions about patient management, treatment strategies, and follow-up care.

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 envisioned product represents a significant advancement in the field of cardiovascular healthcare, offering a holistic approach to heart disease detection and management. By leveraging machine learning techniques, the system can analyze vast amounts of patient data, including demographic information, clinical measurements, diagnostic test results, and medical imaging scans. This multidimensional analysis enables the system to identify subtle patterns and correlations indicative of underlying cardiovascular conditions, facilitating early detection and intervention.

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.

Overall, the product aims to revolutionize the way heart disease is detected, diagnosed, and managed, ultimately leading to improved patient outcomes and healthcare delivery.

## 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. Project Plan

The project plan provides a detailed roadmap for project execution, outlining activities, milestones, timelines, and resource allocation. It serves as a guiding framework for project management and ensures alignment with project goals and objectives.

The first phase of the project involves requirements gathering and analysis, wherein project stakeholders collaborate to define system specifications, functional requirements, and user interface design. This phase also includes data collection and preprocessing activities, such as data cleaning, normalization, and feature engineering, to prepare the data for machine learning model development.

Once requirements are finalized and data preprocessing is complete, the 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.

Finally, the project concludes with the system's full-scale deployment and adoption, accompanied by comprehensive training, documentation, and support services for end-users. Ongoing maintenance, updates, and enhancements are prioritized to ensure the system remains effective, efficient, and aligned with evolving healthcare needs.

# 5. SOFTWARE REQUIREMENT ANALYSIS

## 5.1. Introduction

The software requirement analysis encompasses a comprehensive examination of the technologies utilized in the development of the heart disease detection system. These technologies are carefully chosen to enable efficient data processing, advanced analytics, user interface design, and system integration. The following technologies play a pivotal role in the development and functionality of the proposed system:

1. Python Programming Language: Python serves as the primary programming language for developing the heart disease detection system. Python is chosen for its versatility, ease of use, and extensive libraries for data manipulation, statistical analysis, and machine learning. Libraries such as Pandas, NumPy, and Scikit-learn are utilized for data preprocessing, model training, and evaluation.

2. Streamlit Framework: Streamlit is employed to build the user interface for the heart disease detection system. Streamlit 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 leverages various machine learning libraries to implement predictive models and algorithms for heart disease detection. Scikit-learn provides a wide range of tools for classification, regression, clustering, and model selection. Additionally, TensorFlow and Keras are utilized for building and training deep learning models for enhanced predictive performance.

4. Data Visualization Tools: Data visualization tools such as Matplotlib and Seaborn are employed to create insightful visualizations and graphical representations of clinical data. These tools enable healthcare professionals to visualize trends, patterns, and correlations within the dataset, facilitating data-driven decision-making and interpretation.

## 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.

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

Machine learning libraries such as Scikit-learn, TensorFlow, and Keras are utilized to develop predictive models for heart disease detection. These libraries offer a wide range of algorithms, techniques, and tools for data preprocessing, feature selection, model training, and evaluation.

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 detection system is compatible with major operating systems, including Windows, macOS, and Linux distributions, ensuring accessibility across different platforms.

2. Client Requirements: Clients accessing the system require a modern web browser such as Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge for optimal performance and compatibility.

3. Tools and Libraries: Python: The system relies on Python 3 programming language for backend development and algorithm implementation.
   Streamlit: The user interface is built using Streamlit framework, facilitating rapid development of interactive web applications.
   Scikit-learn, TensorFlow, Keras: 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: The system architecture is designed 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. Implementation of the Project

The implementation of the heart disease detection project entails writing code, building software components, and integrating different modules to create a cohesive system. Python, a versatile programming language, is chosen as the primary language for development due to its extensive libraries and frameworks suitable for machine learning and web application 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.

Machine learning algorithms, including Decision Trees, Random Forest, and K-Nearest Neighbors (KNN), are implemented to train predictive models using the heart disease dataset. These algorithms are selected based on their suitability for classification tasks and their ability to handle complex datasets.

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 areas for improvement 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 and 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 system reliability over time. Collaborative efforts between developers, stakeholders, and end-users contribute 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 areas for improvement, and extracting valuable insights for future endeavors.

## 7.1. Current Status of the Project

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 a cohesive system.

The application's web-based interface, built using Streamlit, provides healthcare professionals with intuitive tools for inputting patient data, visualizing diagnostic results, and accessing predictive models for heart disease detection. Machine learning algorithms, such as Decision Trees, Random Forest, and K-Nearest Neighbors (KNN), have been trained using a comprehensive dataset, enabling accurate risk assessment and diagnosis.

The project's current 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, there are still areas that require attention 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 areas for improvement and refinement.

3. Data Security and Privacy: Ensuring the security and privacy of patient data is paramount to maintaining regulatory compliance and safeguarding sensitive information. Implementing robust encryption protocols, access controls, and data anonymization techniques can mitigate the risk of unauthorized access, data breaches, and privacy violations.

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: Ongoing maintenance and support are essential for addressing software bugs, performance issues, and user inquiries. Establishing a dedicated support team, implementing a ticketing system, and adhering to service level agreements (SLAs) can ensure timely resolution of issues and optimal system performance.

## 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: Leveraging data analytics and insights-driven approaches empowers stakeholders to make informed decisions, prioritize

initiatives, and optimize resource allocation. Data-driven decision making fosters transparency, accountability, and evidence-based practices, leading to more effective 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.

The dataset comprises of a total of 76 characteristics, each of which has the potential to provide valuable insights into cardiovascular health. However, it is important to mention that most published experiments and analyses focus on a specific set of 14 characteristics. These specific characteristics have been carefully chosen and standardized across various studies, with a particular emphasis on the Cleveland database. It is worth noting that this field consists of integer values, ranging from 0 (indicating no presence of heart disease) to 4 (indicating severe presence). For experimental purposes, analyses typically focus on distinguishing between heart disease's presence (values 1, 2, 3, or 4) and absence (value 0).

The subset of 14 attributes utilized in most analyses and experiments are carefully curated to capture essential aspects of heart health and aid in effective predictive modeling. This subset of attributes encapsulates diverse aspects of heart health, ranging from demographic characteristics to physiological parameters and diagnostic test results. By utilizing these characteristics, scientists hope to create strong prediction models that can reliably detect and categorize heart disease cases, enabling early intervention and better patient outcomes. These attributes include:

| | 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

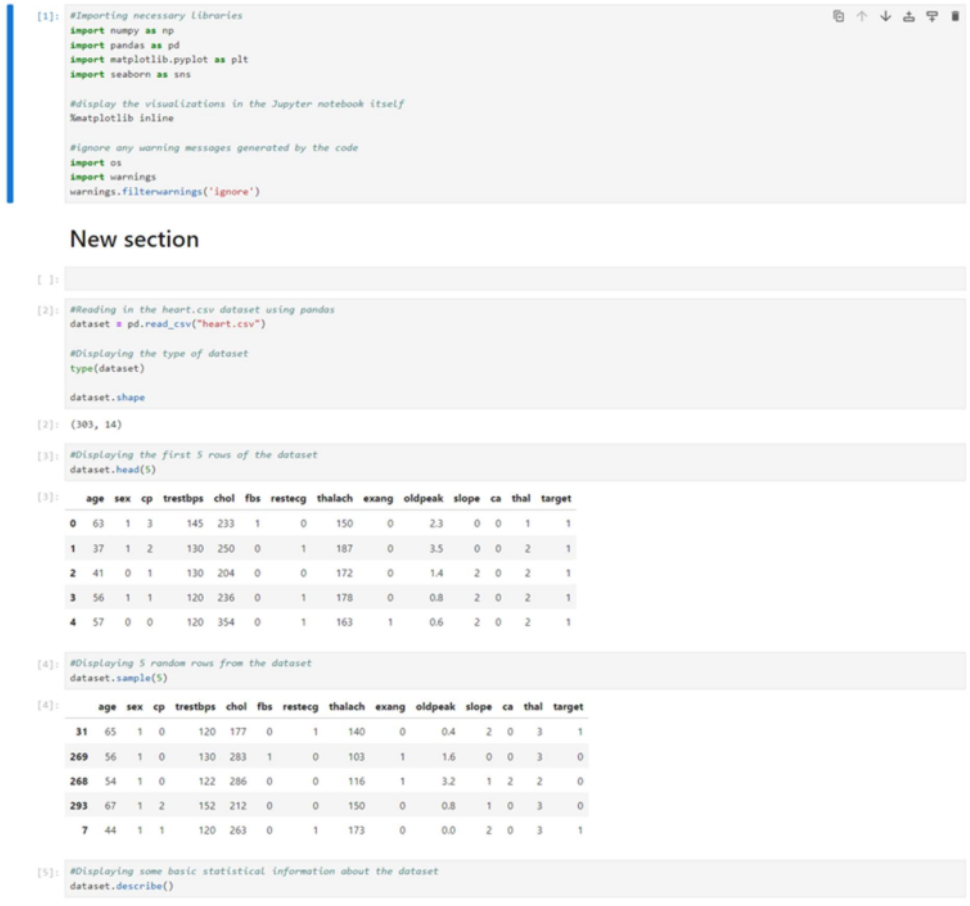
**App.py file :**

```
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}')
```

**Jupyter Notebook:**



```
[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 | t |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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
```

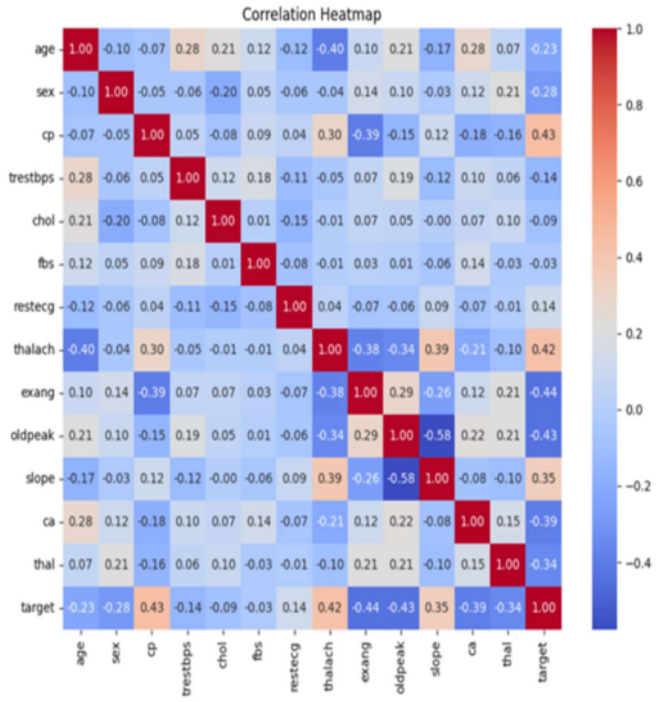
[8]:
```
# Displaying unique values in the "target" column
dataset["target"].unique()
```

[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
```

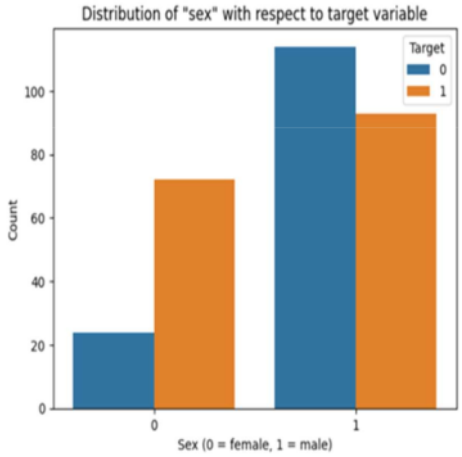
[10]:
```
# Assuming 'dataset' is a DataFrame with numeric values
# If not, you might need to preprocess your data accordingly
target_temp = dataset["target"].value_counts()  # Count the occurrences of each unique value in the 'target' column

# 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
```



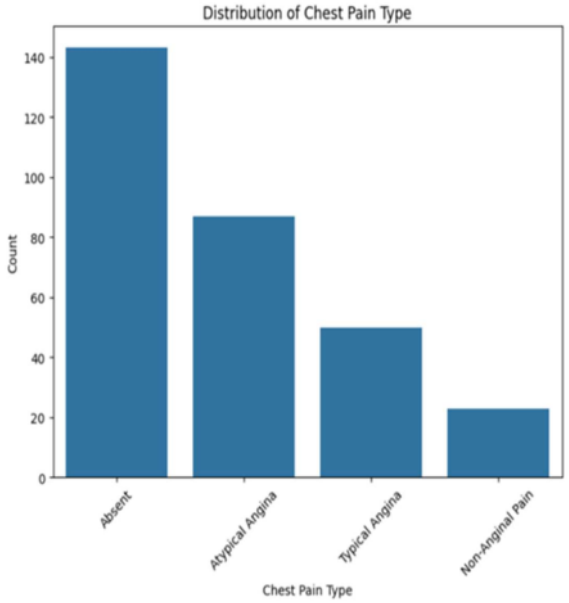
[11]:
```
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))
```

```
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
```

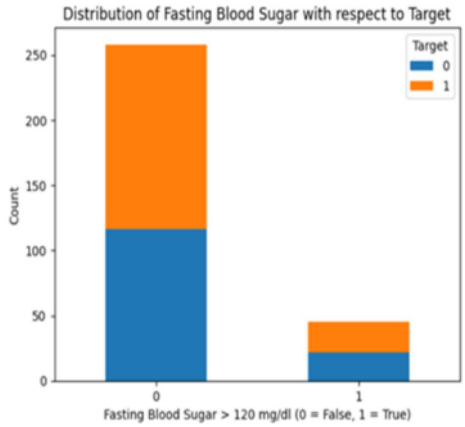



[14]:
```
# Display summary statistics for the "fbs" column in the dataset
dataset["fbs"].describe()
```

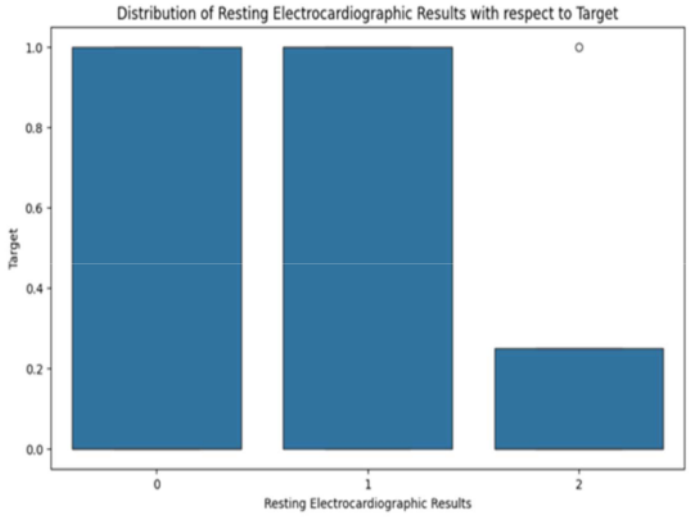
[14]:
```
count    303.000000
mean       0.148515
std        0.356198
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
Name: fbs, dtype: float64
```

[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()
```

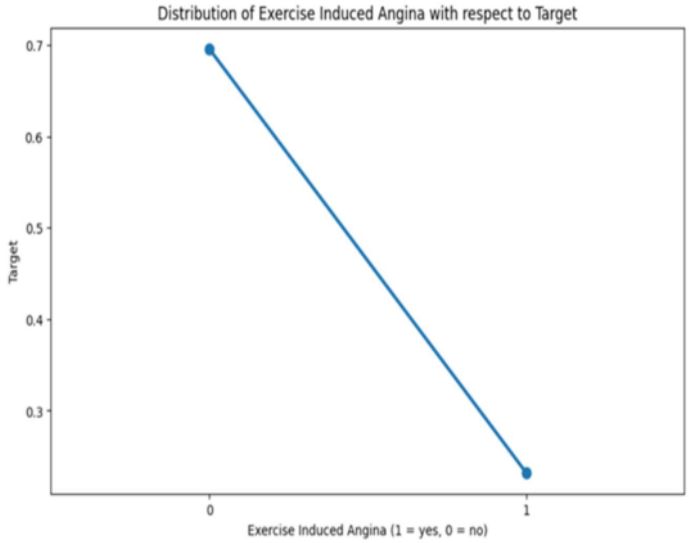
```
<Figure size 800x600 with 0 Axes>
```



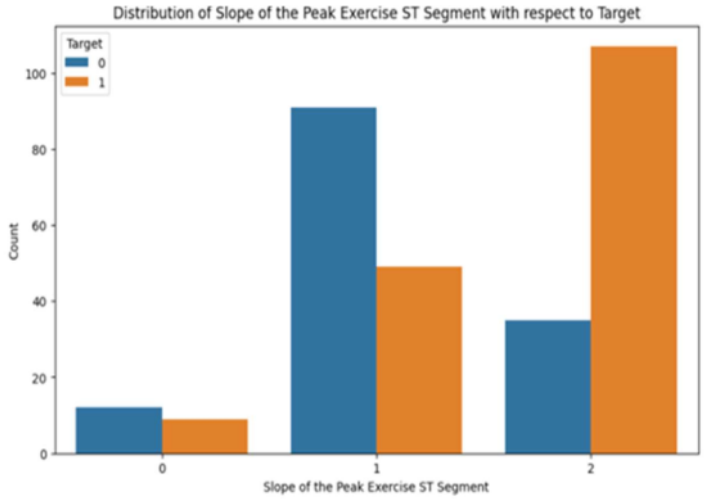

[16]:
```
# Set the figure size
plt.figure(figsize=(10, 6))  # Set the size of the figure for better visualization

# 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
```



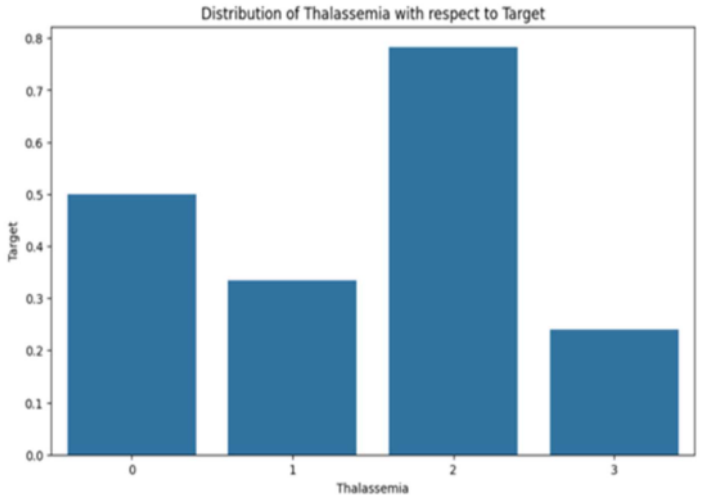
[17]:
```
# Set the figure size for better visualization
plt.figure(figsize=(10, 6))

# 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
```

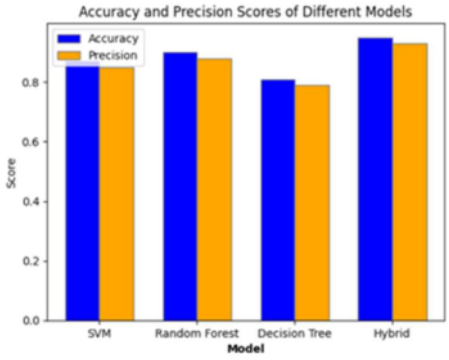
[26]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# 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)
```

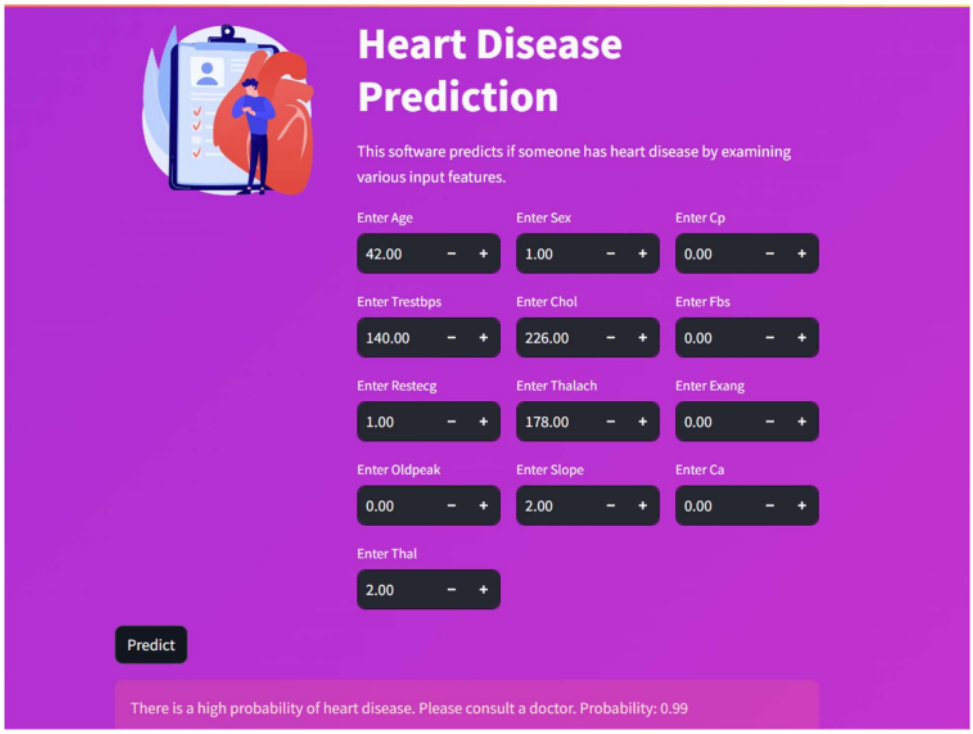
```
Max Accuracy for Hybrid Model: [95.48]
```

[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



## 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<br>Internet Source | 1 % |
| 2 | www.slideshare.net<br>Internet Source | 1 % |
| 3 | de.slideshare.net<br>Internet Source | 1 % |
| 4 | everdark.github.io<br>Internet Source | 1 % |
| 5 | git.io-warnemuende.de<br>Internet Source | 1 % |
| 6 | rstudio-pubs-static.s3.amazonaws.com<br>Internet Source | 1 % |
| 7 | www.ijcspub.org<br>Internet Source | 1 % |
| 8 | journals.sdu.edu.kz<br>Internet Source | 1 % |
| 9 | huggingface.co<br>Internet Source | 1 % |

| | | |
|---|---|---|
| 10 | fastercapital.com<br>Internet Source | 1 % |
| 11 | David B. Olawade, Nicholas Aderinto, Gbolahan Olatunji, Emmanuel Kokori, Aanuoluwapo C. David-Olawade, Manizha Hadi. "Advancements and Applications of Artificial Intelligence in Cardiology: Current Trends and Future Prospects", Journal of Medicine, Surgery, and Public Health, 2024<br>Publication | 1 % |
| 12 | Submitted to Liverpool John Moores University<br>Student Paper | <1 % |
| 13 | Submitted to University of Hertfordshire<br>Student Paper | <1 % |
| 14 | docplayer.net<br>Internet Source | <1 % |
| 15 | Marwah Abdulrazzaq Naser, Aso Ahmed Majeed, Muntadher Alsabah, Taha Raad Al-Shaikhli, Kawa M. Kaky. "A Review of Machine Learning's Role in Cardiovascular Disease Prediction: Recent Advances and Future Challenges", Algorithms, 2024<br>Publication | <1 % |
| 16 | Submitted to University of Salford<br>Student Paper | <1 % |

| | | |
|---|---|---|
| 17 | dspace.lpu.in:8080<br>Internet Source | <1 % |
| 18 | www.ijraset.com<br>Internet Source | <1 % |
| 19 | Submitted to University of Bedfordshire<br>Student Paper | <1 % |
| 20 | soriic.com<br>Internet Source | <1 % |
| 21 | github.com<br>Internet Source | <1 % |
| 22 | open-innovation-projects.org<br>Internet Source | <1 % |
| 23 | Submitted to Clark University<br>Student Paper | <1 % |
| 24 | iabac.org<br>Internet Source | <1 % |
| 25 | Submitted to University College London<br>Student Paper | <1 % |
| 26 | reviewcontent.website2.me<br>Internet Source | <1 % |
| 27 | Submitted to University of Chichester<br>Student Paper | <1 % |
| 28 | online.wpunj.edu<br>Internet Source | <1 % |

| | | |
|---|---|---|
| 29 | Ahmad Ayid Ahmad, Huseyin Polat. "Prediction of Heart Disease Based on Machine Learning Using Jellyfish Optimization Algorithm", Diagnostics, 2023<br>Publication | <1 % |
| 30 | www.ncbi.nlm.nih.gov<br>Internet Source | <1 % |
| 31 | www.researchgate.net<br>Internet Source | <1 % |
| 32 | Submitted to University of Bradford<br>Student Paper | <1 % |
| 33 | easychair.org<br>Internet Source | <1 % |
| 34 | www.ahu.edu<br>Internet Source | <1 % |
| 35 | Submitted to NCC Education<br>Student Paper | <1 % |
| 36 | discuss.streamlit.io<br>Internet Source | <1 % |
| 37 | www.lri.fr<br>Internet Source | <1 % |
| 38 | ledgerliveapp.gitbook.io<br>Internet Source | <1 % |
| 39 | www.researchandmarkets.com<br>Internet Source | <1 % |

| | | |
|---|---|---|
| 40 | 1library.net<br>Internet Source | <1 % |
| 41 | Isabel Voigt, Hernan Inojosa, Anja Dillenseger, Rocco Haase, Katja Akgün, Tjalf Ziemssen. "Digital Twins for Multiple Sclerosis", Frontiers in Immunology, 2021<br>Publication | <1 % |
| 42 | git.eleves.ens.fr<br>Internet Source | <1 % |
| 43 | ijarsct.co.in<br>Internet Source | <1 % |
| 44 | utilitiesone.com<br>Internet Source | <1 % |
| 45 | patents.google.com<br>Internet Source | <1 % |
| 46 | www.coursehero.com<br>Internet Source | <1 % |
| 47 | www.irjmets.com<br>Internet Source | <1 % |
| 48 | Daniyal Asif, Mairaj Bibi, Muhammad Shoaib Arif, Aiman Mukheimer. "Enhancing Heart Disease Prediction through Ensemble Learning Techniques with Hyperparameter Optimization", Algorithms, 2023<br>Publication | <1 % |

| | | |
|---|---|---|
| 49 | codeclimate.com<br>Internet Source | <1% |
| 50 | git.spip.net<br>Internet Source | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | Off |
| Exclude bibliography | On | | |