A project report on

HEART DISEASE PREDICTION USING MACHINE LEARNING TECHNIQUES

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

By

AASHISH BANSAL (19BIT0346)

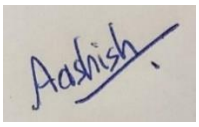
School of Information Technology and Engineering

May, 2023

DECLARATION

I here by declare that the thesis entitled “HEART DISEASE PREDICTION USING MACHINE LEARNING TECHNIQUES” submitted by me, for the award of the degree of Specify the name of the degree VIT is a record of bonafide work carried out by me under the supervision of Placement Authority.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.



Place: Vellore

Date: May 13, 2023 Signature of the Candidate

Internship completion certificate

ABSTRACT

I propose to develop an application which would be able to predict whether someone would be suffering from a coronary heart disease in the coming years. I propose to develop a simple user interface through which the user would just need to answer certain questions which would be provided to the Machine Learning Model as an input and the prediction would be provided to the user on the interface itself. Since this interface is a ready-to-use interface, there is no need for the user to create an account or sign-up. The questions being asked here relate to the information obtained from the tests which the user might have undergone or the user might simply be aware of the information about himself in general. The Machine Learning model is created from the algorithms which are highly effective and could be used to deploy in the real-world scenario. While executing the program, the interface program predicts through the machine learning model which has been trained on similar kind of data. However, at this stage, this project is meant for educational purpose only with a real aim of being deployed into the real-world.

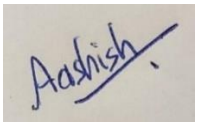
ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Rajanish Trivedi, Vice President, Bank of America Continuum India, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Artificial Intelligence and Machine Learning.

I would like to express my gratitude to Dr. G. Viswanathan, Dr. Rambabu Kodali, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Mr. G. V. Selvam, Dr. Partha Sharathi Mallick, Dr. V. S. Kanchana Bhaaskaran, and Dr. S. Sumathy, School of Information Technology and Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to School of Information Technology and Engineering. Dr. S. Sumathy (Dean), all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. And last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.



Place: Vellore

Aashish Bansal

Date: May 13, 2023

Name of the student

Place: Mumbai

Rajanish Trivedi

Date: May 25, 2023

Name of the Guide

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACO: Ant Colony Optimization
ANN: Artificial Neural Network
ASM: Attribute Selection Measure
AUC: Area Under ROC Curve
BCO: Bee Colony Optimization
C-GA: Clustered Genetic Algorithm
CHD: Coronary Heart Disease
CNN: Convolutional Neural Network
CVD: Cardiovascular Disease
CSV: Comma Separated Values
CCET: Conference on Current Development in Engineering and Technology
CSPA: Colloquium on Signal Processing & Its Applications
DT: Decision Tree
DVC: Data Version Control
FHS: Fetal Health State
GNB: Gaussian Naïve Bayes
GPU: Graphical Processing Unit
GTX: Giga Texel Shader eXtreme
ICCPC: International Conference on Computer, Power and Communications
ICSPC: International Conference on Systems, Process & Control
ICACCS: International Conference on Advanced Computing and Communication Systems
ICACRS: International Conference on Automation, Computing and Renewable Systems
ICIDCA: International Conference on Innovative Data Communication Technologies and Application
ICSCDS: International Conference on Sustainable Computing and Data Communication Systems
IITCEE: International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics
INOCON: International Conference for Innovation in Technology
ICECONF: International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering
ICMNWC: International Conference on Mobile Networks and Wireless Communications
ICPECTS: International Conference on Power, Energy, Control and Transmission Systems
KNN: K-Nearest Neighbors
LR: Logistic Regression
LGBM: Light Gradient Boosting Machine
ML: Machine Learning
MLP: Multi-Layer Perceptron
MCAR: Missing Completely at Random
NB: Naïve Bayes
NaN: Not a Number
NCD: Noncommunicable Disease
NIH: National Heart Institute
NHLBI: National Heart, Lung and Blood Institute
RF: Random Forest
RAM: Random Access Memory
ROC: Receiver Operating Characteristic Curve
SVC: Support Vector Classifier
SVM: Support Vector Machine
TPR: True Positive Rate
UCI: University of California
UPCON: Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering
XGB: eXtreme Gradient Boosting

Chapter 1

Introduction

1.1 INTRODUCTION

According to the World Health Organization, the Peripheral Arterial Diseases and Coronary Heart Diseases are categorized as the cardiovascular diseases. The number of deaths happening every year have started to increase due to an increase in the risk factors and causes of such problems. In 2016, 63% of all NCD-related deaths occurred in India with cardiovascular disease accounting for 27% of all fatalities [2]. Diagnosis and Treatment of heart-related diseases is very challenging. It is not just lack of resources which is the cause of the fatality, but sometimes inadequate medical attention or inappropriate disease treatment also results in fatal ending. Some of the most common cardiovascular disease risk factors are use and consumption of cigarettes, excess drinking, caffeine intake, depression, insufficient physical activity, overweight, hypertension, high cholesterol levels, pre-existing cardiovascular problems, etc. Due to all of this, an efficient, reliable, accurate, agile, and timely medical identification method for such heart diseases is critical and also the general public needs to start taking actions on improving, working and building a healthier lifestyle instead of moving away from it.

Individuals' access to the best medical care is a top priority for healthcare organizations. Accurate patient diagnosis and selection of the most appropriate course of action for treatment are both components of competent care, which also works to prevent inaccurate diagnoses. The vast amount of information the medical industry has to offer is routinely used by researchers to develop new science and technology aimed at reducing the number of heart-related deaths. Machine learning and deep learning are used to manage healthcare data in order to improve categorization and segment data based on criteria. Machine learning is one of the branches of artificial intelligence that is growing most quickly. These algorithms are capable of analyzing vast volumes of data from numerous industries, including the healthcare sector. It is an alternative to the conventional predictions modelling approach using a computer to understand the complex and non-linear interactions between the numerous variables by reducing mistakes in projected and actual results.

The amount and organization of medical information is typically very enormous. Large-scale data management and information mining are capabilities of ML algorithms. Systems using machine learning learn from past data and forecast current data. By encouraging cardiologists to act more rapidly and enabling more patients to obtain treatment in a shorter amount of time, the adoption of this kind of machine learning framework for the prediction of cardiac disease may help save many lives.

Prediction of cardiac disease can have a significant impact on both the medical industry and people's lives. Various writers have created classification models in various methods. Data from daily activities can also be obtained via wearables. A variety of data mining algorithms such as regression, clustering and association rules were used to categorize distinct heart disease features and anticipate heart disease. However, the bulk of them are correct but not to the required level. As a result, this work provides Coronary Heart Disease Prediction and Classification Using Hybrid Machine Learning in order to address these challenges and create an effective cardiovascular disease prediction system. This method will produce superior predictions in terms of accuracy, True Positive Rate (TPR), and specificity.

1.2 PROBLEM STATEMENT

Nowadays, a lot of individuals are found with some type of pain in their chest or some sort of unknown disease around the thoracic region of the body. There are multiple reasons to this kind of pain and disease, from which, the major reason is the modern lifestyle and the eating habits of the individuals. The individuals do not consider having a balanced lifestyle or work-life balance which tends to have an adverse effect on their body. So, the individuals need to work on improving their lifestyle and work towards a better living. The software being developed in this project should be able to provide insights based on certain inputs and it is an attempt to predict accurately for the disease.

Chapter 2

Literature Survey

2.1 SUMMARY OF THE EXISTING WORKS

Table 1: Summary of Literature Survey

S. No.	Title of the Paper	Year	Dataset Name	Algorithms	Methods	Metrics (Accuracy)
1	Heart Disease Prediction using Hybrid ML Algorithms	2023	Framingham Heart Study (FHS) dataset	Naïve Bayes, AdaBoost	Particle Swarm Optimization	NB: 87.6%, Hybrid Algorithms: 97.43%
2	Improving Accuracy of Heart Disease Prediction through Machine Learning Algorithms	2023	UCI Heart Disease dataset; Framingham Heart Study Dataset; Heart Disease Cleveland Uci Dataset; Cardiovascular Disease Dataset	Logistic Regression, Random Forest, Decision Tree, k-Nearest neighbour	Attribute selection is performed through decision tree learning approach and random forest learning approach.	LR: 89.59%, KNN: 80.97%, RF: 83%, DT: 82.63%
3	Coronary Heart Disease Prediction and Classification using Hybrid Machine Learning Algorithms	2023	Framingham Heart Laboratory	Decision Tree and Ada Boosting		Naïve Bayes: 87%. Hybrid Algorithms: 97.43%
4	Machine Learning based Mobile App for Heart Disease Prediction	2023	Not mentioned	Random Forest, Logistic Regression, and ANN Multilayer Perceptron	method bagging	Not provided
5	Heart Disease Prediction using Ensemble ML	2023	UCI Heart Disease dataset	SVM Classifier, KNN, Decision Tree, AdaBoost Classifier, Gaussian NB, Gradient Boosting Algorithm, MLP Classifier, Random Forest Classifier	Normal ML Model	KNN: 91.76%, DT: 95.13%, SVC: 95.88, RF: 95.50%, AdaBoost: 94.38%, MLP: 95.88%, Gaussian NB: 94.38%, DVC: 98.12%, Gaussian Naïve Bias: 95.88%
6	An Effective Heart Disease Prediction Method using Extreme Gradient Boosting Algorithm Compared	2023		KNN, Naive Bayes, SVM, Simple Logistic Regression and ANN	CAD procedures	ANN: ~85%, CNN: ~95%, KNN: ~55%, NB: ~75%, SVM: ~80%,

	with Convolutional Neural Networks					XGBoost: ~90%
7	A Review: Heart Disease Prediction in Machine Learning & Deep Learning	2023				
8	Machine Learning based Cardiac Disease Prediction- A Comparative Analysis	2023	UCI	Naïve Bayes, Logistic Regression, Random Forest	ETL, Data Enrichment	NB: 84.12%, RF: 87%, LR: 80%
9	Heart Disease Prediction: Optimization of Machine Learning Algorithms	2023		KNN, DT, RF, LR, SVM	Smoothing, normalization, and grouping	LR: 87.01%, SVM: 86.04%, KNN: 74.68%, DT: 96.97%, RF: 96.75%
10	A Comparison of Supervised Learning Algorithms to Prediction Heart Disease	2023	UCI	KNN, SVC, MLP		KNN: 91.8%, SVC: 90.16%, MLP: 88.52%
11	A Methodology For Early Prediction and Classification of Heart Diseases in Diabetic Patients With Machine Learning Techniques	2022	UCI	RF Classifier, LR Classifier, SVM		Avg Accuracy: 93.26%
12	Machine Learning Based Classification Algorithms Performance Analysis for Heart Disease Prediction	2022	UCI Cleveland heart disease dataset	LR, NB, KNN, DT, Ensemble Learning, RF, Gradient Boosting, XGBoost	ROC, AUC	LR: 82%, NB: 75%, KNN: 88%, DT: 75%, RF: 82%, AdaBoost: 84%, Gradient Boosting: 81%, XGBoost: 79%
13	Heart Disease Prediction using Clustered Genetic Optimization Algorithm	2023	UCI	K-Means	Clustered Genetic Algorithm, Segmentation Process, Optimization Process	Accuracy: 94.56%
14	Software Development Framework for Cardiac Disease Prediction Using Machine Learning Applications	2022	UCI Cleveland	RF, Ensemble Learning with RF and Bagging	Bootstrap methods	KNN: 90.25%, DT: 79.21%, LR: 85.25%, SVM: 81.97%, RF: 91.1%
15	Machine Learning and Deep Neural Network Techniques for Heart Disease Prediction	2022	UCI repository	KNN, NB, SVM, AdaBoost, RF, DT, LR, XGBoost, LGBM, GB, CatBoost		XGB: 81.1%, Deep Neural Network: 85.9%

2.2 CHALLENGES PRESENT IN EXISTING SYSTEM

Paper 1: Heart Disease Prediction using Hybrid ML Algorithms [1]

This paper focuses on the use of Hybrid Machine Learning. The dataset used by the researchers is the Framingham Heart Disease Dataset. The paper discusses in detail on the features in the dataset which are required for the prediction and procedure followed in Machine Learning. The paper focuses more on the Decision Tree and the AdaBoost Algorithm to create reliable classifiers. They create a training model and then create a second model using the first model in which all the errors are fixed. The paper focuses more on the learning aspect of the algorithms and underlines the importance of optimization required for the learning model as there are several weak learners which create a classification challenge. The predictive algorithm provides the evaluation for the people based on the attribute values. From the results, we can understand the Hybrid Algorithm can perform better as compared to a single standalone algorithm along with optimization. In comparison to Naïve Bayes, the Hybrid ML Algorithm shows higher accuracy, precision and TPR.

Paper 2: Improving Accuracy of Heart Disease Prediction through Machine Learning Algorithms [2]

This paper focuses more on improving the Accuracy of the Model. The datasets used by the researchers is the UCI Heart Disease Dataset, Framingham Heart Disease Dataset, Heart Disease Cleveland UCI Dataset and Cardiovascular Disease Dataset. The researchers use multiple datasets, however, the information on the way the datasets were used during implementation has not been provided. The researchers used Logistic Regression, Random Forest, Decision Tree and K-Nearest Neighbors algorithms and identified several challenges and limitations which need to be resolved. The results obtained on each of the datasets are different and the results obtained are the highest for the UCI Dataset as compared to the other dataset. The researchers concluded that the algorithm Logistic Regression works best on all the dataset, however, if we were to compare it with the previous paper, we can see the results are similar to the algorithm used there. This is a major drawback of this paper as there is no information on why the performance of the other algorithms is lower in comparison to Logistic Regression.

Paper 3: Coronary Heart Disease Prediction and Classification using Hybrid Machine Learning Algorithms [3]

The paper presents the use of the Hybrid ML Algorithms and uses the Framingham Heart Disease Dataset. The paper discusses on the attributes presented in the dataset and the way a feature engineering is being performed to obtain the best possible features to be used for obtaining the results. The paper also discusses the Particle Swarm Optimization technique where each member of the population is considered a particle and the performance of all particles would be evaluated using predetermined cost functions at the end of every iteration. The paper also focuses on Normalizing the data with Gaussian Distribution as it helps in improving efficiency and accuracy of the machine learning models. The paper takes into account an ensemble model which is used to create classifiers. The paper shows that the Ensemble Hybrid model shows better results as compared to a standalone algorithm.

Paper 4: Machine Learning based Mobile App for Heart Disease Prediction [4]

The paper focuses on a well-defined procedure for the implementation for the overall application. The paper focuses more on the Random Forest algorithm as compared to Logistic Regression and focuses more on developing an application which could be used easily by everyone. The application is being developed for mobile devices because they are portable and could be used easily to connect to the application for use which improves the scope of the project and also increases the target audience because almost everyone today has a mobile device and being able to access such application on a mobile would be highly beneficial. The limitation in this paper is that there isn't much information provided on the learning of the model or the way the learning models were trained.

Paper 5: Heart Disease Prediction using Ensemble ML [5]

This paper provides a comparative study on several machine learning algorithms. The dataset being used in this is the UCI Repository dataset. Unlike the paper discussed before, there is only one dataset being used in this paper which is a limitation. The reason behind this being a limitation is that it is a very small dataset and not possible to determine whether the model has learnt adequately or not. Since a large number of algorithms are being used in this paper, so we can consider obtaining useful insights for comparison which happens to be the main paper of the paper as well. The paper follows a structured approach and uses a large number of techniques for validation which could help determine whether the model was learnt properly or not but since the dataset is very small and the model does not provide insights on how it is classifying after learning, it is still difficult to say. The paper discusses on the use of the ensemble learning method to enhance the functionality of the learning model by combining them but the way of combination has not been discussed which creates a challenge for the reader in understanding the paper. Also, the results for the individual classifiers haven been provided but the results for the ensemble model are not provided in the analysis.

Paper 6: An Effective Heart Disease Prediction Method using Extreme Gradient Boosting Algorithm Compared with Convolutional Neural Networks [6]

The paper focuses on comparison of several machine learning algorithms in order to create a smart and user-friendly application. The dataset being used in this paper is the UCI Repository dataset. The machine learning algorithms considered for this paper are MLP, Simple Logistic Regression, Naïve Bayes, Random Forest and SVM and the deep learning algorithms considered for this are ANN and CNN. The results obtained from these algorithms are being compared. The results of machine learning algorithms are being compared and the deep learning algorithms perform better as compared to the machine learning algorithms. A major drawback of this paper is the use of a very small dataset on the deep learning algorithms. Even though the results obtained are better as compared to the machine learning algorithms, this cannot be considered under efficient learning because deep learning is supposed to be used on much larger datasets. It is possible that the features were identified correctly but it is not possible to say whether the model has undergone overfitting or not. Another reason to this is the small amount of testing data because the actual dataset would be split into training and testing data. This is a challenge which must be overcome in order to create a better model which would be considered safer to use.

Paper 7: A Review: Heart Disease Prediction in Machine Learning & Deep Learning [7]

This review paper discusses the machine learning and deep learning techniques which are being by the healthcare industry to create technologies which could help in saving lives and provide facilities which would help people have an opportunity to be warned beforehand so that they can take steps for a healthier future. The paper discusses the studies which were conducted in order to create the dataset which are being used now by the researchers to create the learning models which are being used in the prediction. The paper discusses several key machine learning and deep learning algorithms which are being used to find the features in the datasets. It says that the deep learning models tend to outperform in terms of accuracy because they can identify the features more accurately. The paper also discusses Feature Selection techniques because and the ways to improve them because it is one of the key phases of the learning implementation.

Paper 8: Machine Learning based Cardiac Disease Prediction- A Comparative Analysis [8]

The paper focuses on comparing the prediction through Python Programming and the use of the Orange Tool. It is a powerful tool which is used to perform data analysis and visualization. It also provides data flow visualizations and helps in increasing the productivity of the developer. A limitation of this paper is that it is trying to compare the implementation provided by a user and a software. The reason behind this is that a user would take way more time as compared to the software in providing an analysis because the user would have

to program all the requirements manually which would involve a lot of time being spent on debugging and implementation. Hence this is not a fair comparison to make because in terms of creativity, the humans are better than the software but in terms of processing speed and the capability of handling the data, the software definitely has an advantage. So, even though the results obtained from the analysis of the tool are better in comparison to the results from the implementation of the user, it is because of the way the software has been developed. And also, the reason behind building a software is to help the user in improving his productivity and not comparing his productivity with that of a software.

Paper 9: Heart Disease Prediction: Optimization of Machine Learning Algorithms [9]

The study focuses on predicting whether an individual would suffer from a heart disease or not in the coming years of time. The paper takes several machine learning algorithms and provides a comparative study for the same. The researchers have implemented the algorithms and provided a claim that the learning algorithm could be used for prediction, but they have not mentioned anything on the safety on the use of the model. Also, there isn't much information available which would denote whether the learning algorithms have been optimized or not during the implementation even though the paper discusses on the aspects of the internal attributes of the functions and the learning algorithms and the ways they could be used for the predictions. The paper is very limited in terms of information being provided and needs to be refined further. Another drawback of this paper is that splitting the dataset into 55:45 ratio from training and testing which means that if the dataset happens to be small, then the amount of training which the learning model had was much less than the expected training. The challenge faced in this paper is the lack of organization of information and the structure in which the research was carried out.

Paper 10: A Comparison of Supervised Learning Algorithms to Prediction Heart Disease [10]

The study conducted is revolves around predicting whether someone will suffer from a heart disease in the upcoming future or not. The research paper uses the UCI Repository dataset for the implementation of the learning algorithms. The results obtained in paper are very similar to the ones obtained in the previously discussed papers in terms of implementation and prediction. The drawback of this paper is that it uses very few learning models and also there are neither any optimization techniques discussed, nor any unique internal parameters mentioned which helped in improving the learning curve or the machine learning models. So, a limitation of this paper is that it is very similar to previously discussed papers as the information is redundant.

Paper 11: A Methodology For Early Prediction and Classification of Heart Diseases in Diabetic Patients With Machine Learning Techniques [11]

The paper focuses more on the patients with diabetes as compared to the other papers which take into consideration the general public. So, the scope of this paper is fairly limited to a very small subset of the population. In a way, it could be said that the researchers are trying to understand about the likeliness of a diabetic patient suffering from heart disease in the coming future. This provides a challenge to the researchers because the datasets created till today are already limited in terms of the records which they hold and the information they provide, and the researchers happen to be in need of more specific information for the problem statement which they have. This challenges the research in terms of time as the researchers would need to spend more time collecting the data and then work on the features which would help them predict the required. The paper also focuses on discussing several reasons behind an increase in the number of heart patients. The researchers take a more mathematical approach for the learning models and provide inferences on contributing factors for the predictions which could provide valuable insights during the implementation once a specific dataset is ready for the research in the coming future.

Paper 12: Machine Learning Based Classification Algorithms Performance Analysis for Heart Disease Prediction [12]

This paper takes into consideration several machine learning algorithms and perform a complete analysis of the model evaluations. The researchers consider use of several loss functions and other metrics obtained from the learning algorithms as results and provide a detailed analysis of the results. This provides a lot of valuable information. Also, the paper takes into account several optimization algorithms which are being used in conjunction with the other machine learning algorithms. The only limitation in this paper is the use of the UCI Cleveland Heart Disease Dataset which happens to be an even smaller dataset as compared to the dataset being used by the other research papers, hence, even though the paper provides several impressive insights for the implementation improvement, in terms of learning improvements, it remains a challenge to obtain insights of the same level.

Paper 13: Heart Disease Prediction using Clustered Genetic Optimization Algorithm [13]

This paper focuses more on the Genetic Optimization algorithm, BCO, ACO, Cuckoo and other optimization techniques to improve system's accuracy and consistency as they are able to evaluate the performance of the cluster Genetic Algorithm. The data is split into several clusters are preprocessing and is then used for model construction. The researchers use multiple experiment sets to validate the performance of the suggested system which is verified based on several other metrics for micro and macro averaging. Just like the research papers discussed earlier, the limitation faced by this research is also the use of the small dataset. This dataset is subjected to segmentation processes and several optimization processes which further have several evaluations. The results of the clustering process are further given as input to the Clustered GA Classification Algorithms which would provide the predictions and evaluations. These evaluations provide very useful details. Some of the learning models outperform the models discussed before due to the optimizations performed on them.

Paper 14: Software Development Framework for Cardiac Disease Prediction Using Machine Learning Applications [14]

The paper focuses more on the practical and legal considerations and procedures for the use of the machine learning and deep learning techniques to enhance data storage and providing results and predictions. This tells us about the challenges in the legal procedures because in order to create a software which would help save human lives, it needs to be checked and verified before it is brought completely into practical use. The reason being if there is an increase in the loss of lives after and due to the use of this software, then it could lead to several wide scale issues which would involve several lawsuits against the developers and researchers working on the application. The paper also aims in telling the user, the serious of the situation for the research. The drawback of small amount of data from the UCI Cleveland Dataset is faced by this research paper as well.

Paper 15: Machine Learning and Deep Neural Network Techniques for Heart Disease Prediction [15]

The research paper focusses more on the comparison of the machine learning algorithms with the Boosting Algorithms. The drawback of small amount of data from the UCI Cleveland Dataset is faced by this research paper as well. The paper discusses several unique algorithms which haven't been discussed in several other papers. The challenge of small dataset is faced by the use of the Transfer Learning where the renounced architectures are being used which are already trained on several aspects and hence, they could be trained further for the specific application even with the help of a smaller dataset. the results obtained in this paper are either similar or a little poor in comparison to the previous papers but the approach being used to tackle the challenges is definitely stronger as compared to the approaches used by the other research papers.

2.3 OTHER CHALLENGES AND LIMITATIONS

The difficulties are as follows:

- Because of the illness's intricacy and the numerous elements that can contribute to it, accurately predicting heart disease is challenging.
- It might be challenging to collect reliable and thorough data regarding a patient's medical history and lifestyle habits.
- It might be difficult to detect tiny changes in a patient's health that may suggest an increased risk of heart disease.
- Creating successful predictive models necessitates a huge quantity of data and processing power, both of which may be costly and time-consuming to obtain.

The limitations are:

- The accuracy of the data used to develop predictive models is restricted, therefore any faults or inaccuracies in the data can lead to erroneous forecasts.
- Predictive models are also limited in their ability to forecast future outcomes based on past data; therefore, they may not always be able to predict future occurrences or patterns in a patient's health or lifestyle habits.
- Predictive models are also limited in their ability to detect small changes in a patient's health that may suggest an increased risk of heart disease, because these changes are not always evident or easily detectable with present technology or procedures.

Chapter 3

Requirements

3.1 HARDWARE REQUIREMENTS

Local Machine:

Processor: Intel i7 10^{th} Generation

RAM: 32GB

GPU: NIVIDIA GeForce GTX 1650Ti (4 GB)

GPU Type: Dedicated

Google Collaboratory:

Processor:

RAM:

GPU:

GPU Type:

3.2 SOFTWARE REQUIREMENTS

Some of the software used while implementing the Application are:

- Anaconda Studio
  - Jupyter Notebook
  - Spyder
- Visual Studio Code
- Python
- Google Drive

Chapter 4

Objective and Scope of the Project

4.1 OBJECTIVE

Nowadays, most of the time we would see people using internet to check for almost everything. Most of the times, we can find people checking for symptoms and causes for the any sort of disease which they believe they might be suffering from. In such scenarios, building something which could be used by them easily is more efficient and effective in terms of providing a solution to what people are expecting and need for use. This project aims to help people and provide an application which they can use easily and something which would provide them with a close answer to the question which they have in mind, "Will I be suffering from such a disease?" This would help them to be alert and start working on their habits, if they happen to be from the group of population which tends to take such things seriously. So, the primary goal of the project is to build a machine learning model which can predict whether someone would be suffering from heart disease or not in the coming years. The secondary goal of the project is the build a ready-to-application which can be used by anyone easily to provide input and obtain the result. Furthermore, the project aims to help the researchers provide a better understanding and obtain useful insights for the research which can help improve the upcoming research and develop this project further.

4.2 SCOPE OF THE PROJECT

The project is does not target any specific age group or individuals. The application developed employing the machine learning model is simple and ready-to-use by anyone, provided they know what they are providing as input. The project application could be deployed into hospitals as well to be used in real-time because one of the applications of this project is to provide prediction, using which the doctor should be able to warn the patients of their health and how they need to work on it for improving it. The hospitals have the capability to reach out a large number of people and they have better experience in dealing with the patients, hence it is possible for them to provide useful insights which could help develop this application further. With such an application being deployed into hospitals, they can reach out to patients and warn then along and provide some additional tests based on the prediction, so that the patient's life expectancy could be improved. Such an application could also be used in by the industries in Corporate because the industries have started moving towards a world where they have to help provide facilities to the employees who are working for them. Some companies even tend to have regular medical health checkups for their employees. With this application, the companies would be able to help warn their employees as well for the same. So, overall, this application should be able to help people increase life expectancy.

Figure 1: Representing Population

Chapter 5

Analysis & Design

5.1 PROPSED ARCHITECTURE

The High-level diagram for the Project is:

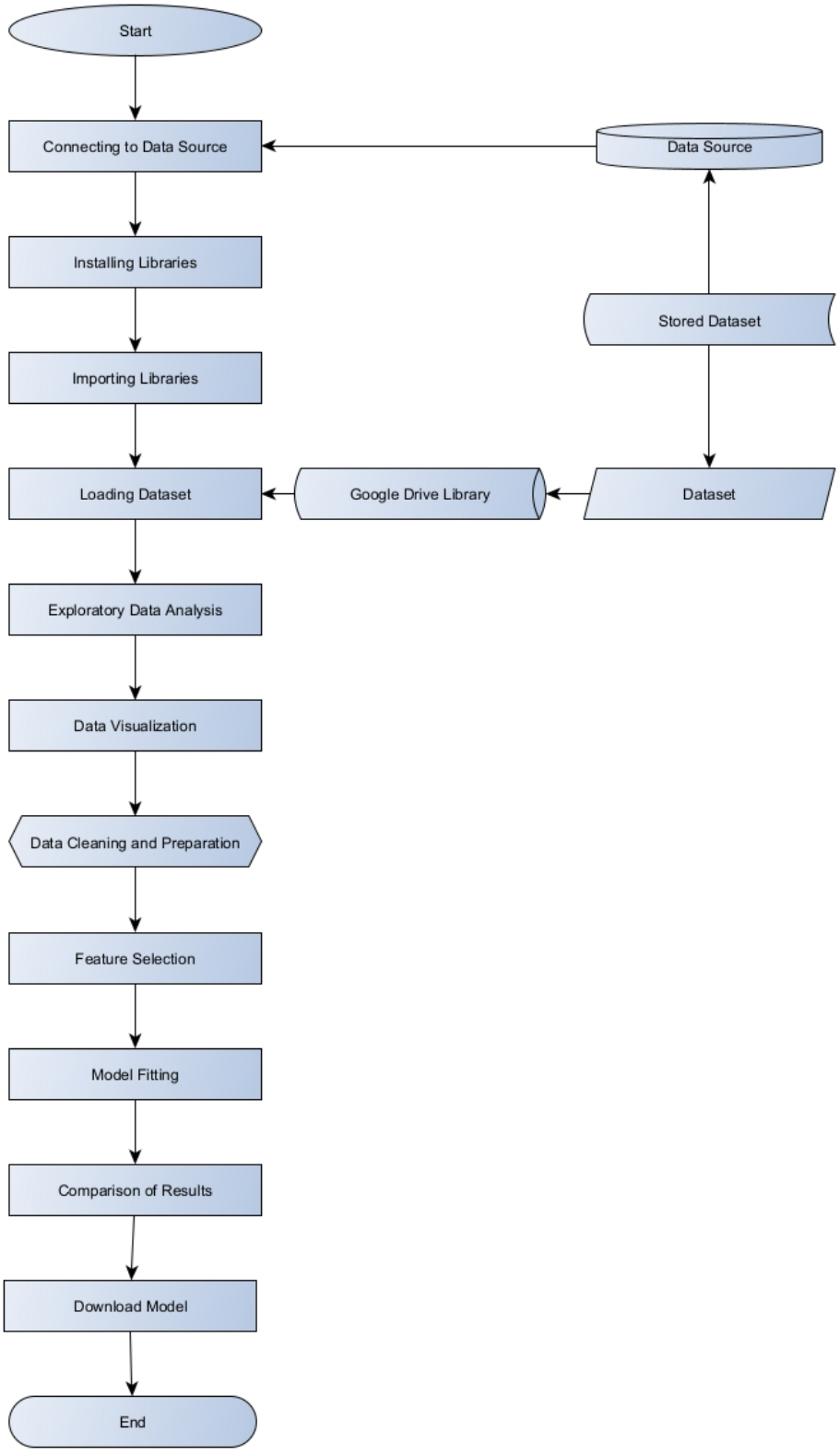


Figure 2: High-level Diagram of the Machine Learning Model

The high-level diagram aims to provide an overview and a simple picture of the way the Machine Learning model works in this project. From the above image, we can see the steps followed by the Machine Learning

model are:

1. Connecting to the Data Source
2. Installing Libraries
3. Importing Libraries
4. Loading Dataset
5. Exploratory Data Analysis
6. Data Visualization
7. Data Cleaning and Preparation
8. Feature Selection
9. Model Fitting and Optimization
10. Comparison of Results
11. Saving/Downloading the Model

The Low-level Diagram for the Learning Model is:

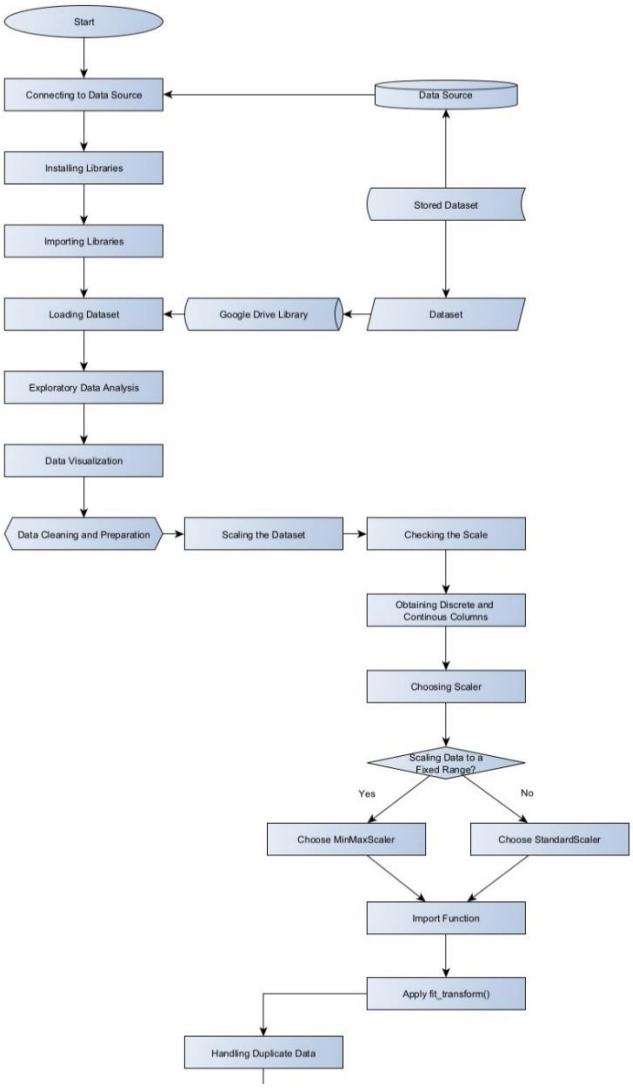


Figure 3: Low-level Diagram of Machine Learning Model (Image 1)

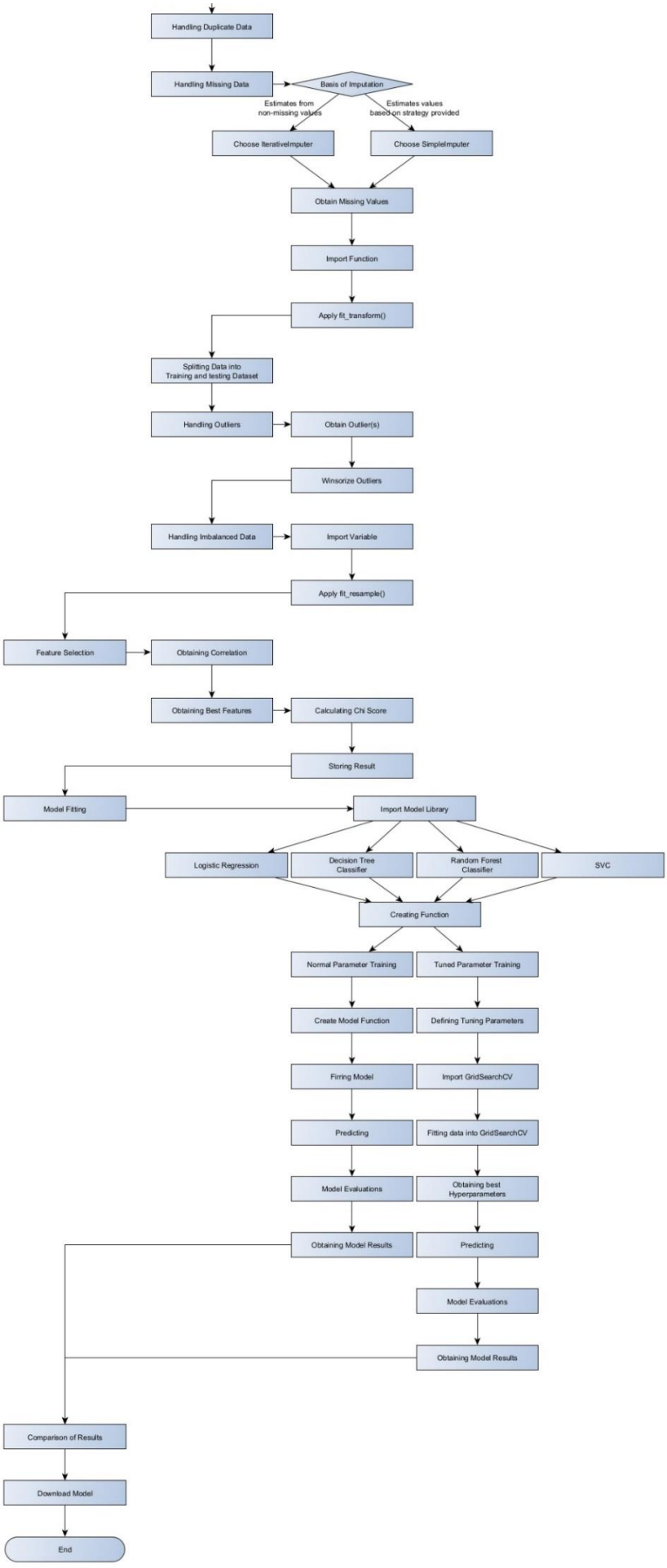


Figure 4: Low-level Diagram of the Machine Learning Model (Image 2)

The Low-level Diagram tends to cover the High-level diagram in much more detail. We can elaborate of the steps from the Low-level Diagram as:

1. Connecting to the Data Source
2. Installing Libraries
3. Importing Libraries
4. Loading Dataset
5. Exploratory Data Analysis
6. Data Visualization
7. Data Cleaning and Preparation:
    a. Scaling the Dataset:
        i. Checking the Scale
        ii. Obtaining Discrete and Continuous Attributes
        iii. Choosing Scaler
        iv. Import Function
        v. Applying Function
    b. Handling Duplicate Values
    c. Handling Missing Data:
        i. Basis of Imputation
        ii. Obtain Missing Values
        iii. Import Function
        iv. Applying Transforms
    d. Splitting Data into Training and Testing Data
    e. Handling Outliers:
        i. Obtaining Outliers
        ii. Winsorize Outliers
    f. Handling Imbalanced Data:
        i. Import Variables
        ii. Apply Transforms
8. Feature Selection
    a. Obtaining Correlation
    b. Obtaining Best Features:
        i. Calculating Chi Score
        ii. Storing Result
9. Model Fitting and Optimization:
    a. Import Model Library
    b. Creating Function Variable

c. If using Normal Parameters:
    i. Create Model Function
    ii. Fit Model on the Data
    iii. Obtain Predictions
    iv. Evaluate Model
    v. Obtain Results
d. If Tuning Hyperparameters:
    i. Defining Tuning Parameters
    ii. Import Search Function
    iii. Find Optimized Parameters
    iv. Obtain Optimized Parameters
    v. Obtain Predictions
    vi. Evaluate Model
    vii. Obtain Results

10. Comparison of Results:
11. Saving/Downloading the Model

5.2 DATA FLOW DIAGRAM

The flow of the data provided by the user(s) could be visualized using the following diagram:

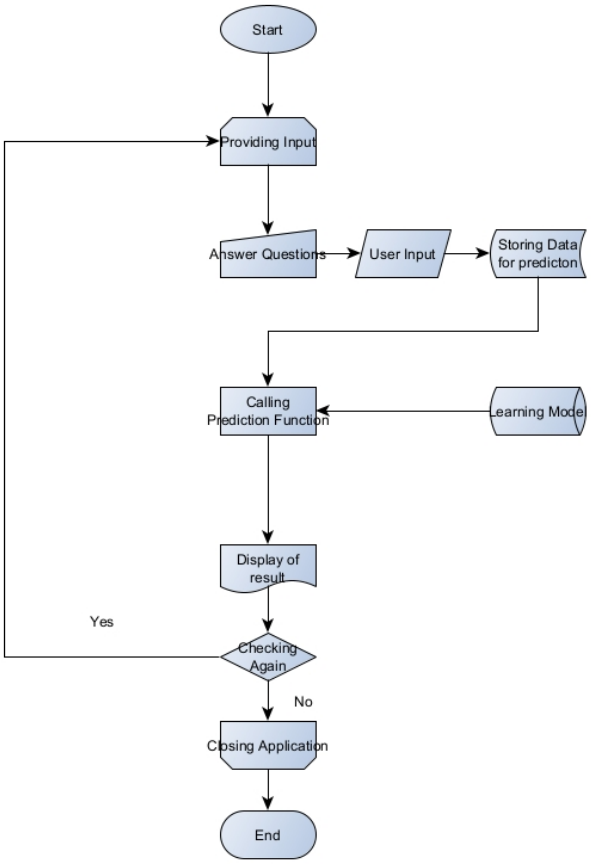


Figure 5: Data Flow Diagram for the Application

The above diagram helps us in understanding what happens with the data in the project. We have the steps as:

1. The user starts the application and the Machine Learning Model present within the application is loaded as it would be used for predictions.
2. The user provides the input to the model by answering the questions asked and hence provides the required data for the prediction(s).
3. The answers to the questions are stored in a standard order using which they would be passed to the model which would in turn use them to process and predict.
4. The Predicting Function would be called from the Learning Model with the data provided as input. The prediction received from the Function would be presented to the user as a result.

5.3 MODULE DESCRIPTION

The modules in this project are:

- Defining Problem Statement: This is the first step of building the Project and it is important for us to consider why we are building the model in the first place.
- Data Selection: This is where we select the data or the attributes which should be present in the data which would used by us to build the machine learning model. The data selected for the learning model in this project is stored in CSV File Format.
- Connecting to the Data Source: This is the first stage of implementing all the Machine Learning Models. We should connect the source or the application where we have stored the data, we collected for us to use it in the Learning Model. The data for this project is stored in Google Drive. Hence, we will authorize our Machine Learning Model to connect with the google drive. It is not necessary to have the data stored in Google Drive only and access, the data could be stored in any location which could be accessed either locally or over the network, and the data could either be stored as a proper CSV or in a raw CSV format.
- Installing Libraries: In Machine Learning, a Library refers to a set of functions and routines written in a specific programming language which are used to perform certain tasks and make them easier for the developers allowing them to build more complex for use. These Libraries are not available before hand and would need to be installed for use. Each Library provides a specific set of functions and routines.
- Importing Libraries: After installing the libraries we need to import them to use the functions which they provide. The libraries could be imported at any point of time and it is not necessary to import them at the beginning, however, we must import the library before using the function desired from that library.
- Loading Dataset: Here, we will load the dataset into the program. We will be performing all our operations on this data and our model would be learning the features obtained from this data.
- Exploratory Data Analysis: This is where we start our basis analysis. Here, we will figure out certain aspects and basic information about our data. In this project, we obtain the following information from Exploratory Data Analysis:
    - Shape of the Dataset: This denotes the number of entries (or rows) and number of features (or columns) in the data. For this data being used, there are 4238 entries with 16 features.
    - Basic Information of the attributes of the data: This gives us the name of the column, the number of non-null columns and the data type of the data present in that particular column.
    - Basic Statistical deductions: The information obtained from this is:
        - Number of entries of the data

        - Mean of the data present in that column
        - Standard Deviation of the data present in that column
        - Minimum and Maximum value of that column
        - The three quartiles of the data.

    The insights obtained from this data are:

        - The spread of age is from 32 to 70.
        - Resting Blood Pressure is distributed from 80 to 200 mm Hg.
        - The heart rate goes from a minimum value of 44 to 143 bpm.
        - A huge number of individuals take medicines for Blood Pressure.

- Data Visualization: This is where we start to visualize our data to gain more insights about it. This is used because it is not possible for us to obtain all the information from numbers or text alone. This helps us in providing a different perspective of looking at the data and gaining insights from it. In the project, we attempt to plot the data in different ways possible, so that we can obtain some more insights about it. First, we plot the data for the entire data, basically, obtain the distribution or spread of all the columns. Also, considering certain attributes, by intuition like the heart disease with respect to age, being a smoker and gender count because we tend to consider these important having some general knowledge on the way these tend to affect us.
- Data Cleaning and Preparation: We know that Data Cleaning is a process of fixing or removing incorrect data, wrong data, incorrectly formatted data, corrupted data, incomplete data, or duplicate data with the data which we have loaded into our Machine Learning Process.
    - Scaling the Dataset: We know that real-world data can contain a lot of features which may be varying in certain units in contrast to our requirements. To deal with this, we use feature scaling.
        - Checking the Scale: We need to check the scale because it is possible that the scale might vary for all the attributes which we have in the data. While performing this, we can also check whether it is feasible for us to scale the data in the same way or not in order to make the scale of the attributes uniform across all. In the project, we have obtained the spread and the scale using the distribution plots because these can help us clearly visualize the way the discrete and continuous values are spread around.
        - Obtaining Discrete and Continuous Attributes: A broad way to categorize the attributes could be on the basis of numeric or non-numeric data. So, the Discrete data is the one which consists of numerical type of data that could include whole numbers or concrete numbers with some specific data values determined by counting whereas Continuous Data includes more than just numbers. It could include Complex numbers, variable data, time series-related data, etc. We get to know about these when we consider the plots when we were scaling the data. This can help us in selecting the features more accurately because we would be able to decide the kind of input we are obtaining from the user and while taking inputs, we should take the input from the user and transform it into the way the model created would be trained upon.
        - Choosing Scaler: There are two types of scalers:
            - StandardScaler: This scaler removes the mean and scales the data to unit variance. However, there is one disadvantage which is that the outliers could influence the computation of the empirical mean and standard deviation. This can change the range of the scale and can affect the selection of features directly. Hence, an important step before this is to handle the outliers. Also, if the outliers tend to have a different spread, then its impact on the feature

selection would also differ.

- MinMaxScaler: This scaler rescales all the features in the data in a way that all the feature values are within the range of [0,1] but a disadvantage of using this is that it will compress all the inner values in a very small range within [0, 1]. For the project, this is the Scaling Technique used because this would be able to provide a clear range for attributes of where most of the values such that we are able to tell the model of the area where it might need to focus more while learning and creating the categorical boundaries.

  - Import Function: Here, we import the function, just the same way we import the other libraries and use it.
  - Applying Function: While applying, the transformed data would be returned, and we will need to store the data. We will store the data in separate variables because this would allow us to compare the obtained result with the original data.

- Handling Duplicate Values: We know that these values are just repeated values in the data which might occur due to several reasons. It is important to handle duplicate values because duplicate values can ruin the split between the training and the testing data because it is possible that all identical entries are not all in the same set. In the project, we just tend to remove the duplicate values because if we were to keep those values, then we might end up making the model more bias towards that particular data leading to overfit for that data and underfit for the remaining data, or basically, the model will not learn properly.
- Handling Missing Data: These values are important and need to be handled because missing values can cause discrepancies for the model in the learning process. This is because if the values are not there, then the model would not be able to make accurate calculations and hence, it will not be able to learn properly.
  - Basis of Imputation: Imputation is used to retain most of the dataset's information by substituting the missing data with a different value determined on certain factors or with the help of certain ways.
  - Obtain Missing Values: These are obtain with the help of a Visualization matrix which helps us in understanding whether the values are missing throughout the column uniformly or not. And how many values are missing with respect to the other columns.
  - Import Function: Here, we import the function, just the same way we import the other libraries and use it.
  - Applying Transforms: Here, we have used the SimpleImputer to handle the missing values which uses the non-missing values to estimate the missing values because if we were to use the IterativeImputer, then it would predict the values based on the estimator. It is not advisable to use IterativeImputer because our dataset is still small because for proper predictions, we would need a bigger dataset.
- Splitting Data into Training and Testing Data: We split the data into training and testing data where 80% of the data is training data and the remaining 20% of the data is the testing data.
- Handling Outliers: It is important to handle the spread of the outliers because these disturb the distribution over which the model would be learning and hence this would create incorrect boundaries for predictions leading to wrong predictions and results.
  - Obtaining Outliers: We obtain some basic insights of the outliers with the help of boxplots because they help us visualize this in a better way.
  - Winsorize Outliers: Winsorization is the process using which we replace the extreme values of the statistical data in order to reduce or at least limit the effect of the outliers

on the calculations and results.

- Handling Imbalanced Data: The imbalanced datasets are the ones which have an uneven distribution in the number of observations for the target class. This hinders accurate prediction due to the difference(s) between the majority and minority classes.
    - Import Variables: We use the library to import and use the required function to obtain the desired result.
    - Apply Transforms: This will resample the complete dataset and balance it.

- Feature Selection: This helps us in identifying the importance of the features which we have in the data and the way they would impact in prediction of the required result. We would discard the features which are not very helpful and also the features which tend to have a negative impact on learning because some features do not correlate to one another.
    - Obtaining Correlation: This helps us in discovering and quantifying the degree to which the variables in the dataset are dependent upon each other.
    - Obtaining Best Features: These are features which highly correlate with one another and could be used for the learning of the model in the further steps.
        - Calculating Chi Score: This is used for continuous random variables with a standard format to complete specifications.
        - Storing Result: We will store the Best Features separate from all the features, so that they could be used for comparison if needed.

- Model Fitting and Optimization: This is where the learning and optimization of the model begins.
    - Import Model Library: We can import the libraries as per the algorithms which we choose.
    - Creating Function Variable: We will store the reference to the function in a variable to access it.
    - If using Normal Parameters:
        - Create Model Function: We will create the model. In this, since we are using the normal parameters, we would leave the parameters as default and continue.
        - Fit Model on the Data: Here, we will allow the model to start learning. The amount of time taken by this process depends on the parameters, the size of the data, the number of features, etc.
        - Obtain Predictions: After our model has completed learning, we will use it predict on the test data which we have.
        - Evaluate Model: Here, we will check the number of predictions which were correct and number of predictions which were incorrect.
        - Obtain Results: Here, we will start working on formulating the reason behind the model performance and determining if the model has learnt properly or if it has undergone underfitting or overfitting.
    - If Tuning Hyperparameters:
        - Defining Tuning Parameters: Here, we will consider the Parameter Space and define the parameters based on our research.
        - Import Search Function: We will import the appropriate search function (GridSearch or RandomSearch).
        - Find Optimized Parameters: For the Project, we have used GridSearch because with this, we can make sure that all the parameters had been checked in the entire Parameter Space and the resulting obtained parameters are the best possible

parameters which we have.

- Obtain Optimized Parameters: These are the parameters which would be taken into consideration during Model Evaluation.
- Obtain Predictions: Since, we have fit the model on the best possible parameters, we will obtain predictions on the test data again.
- Evaluate Model: We will evaluate the model using several metrics which would help us determine if the model has learnt properly or if it has undergone underfitting or overfitting.
- Obtain Results: These results would be obtained from the evaluations which would be used to formulate the reason behind the performance and gain insights.

- Comparison of Results: The results from the normal parameter training and tuned parameter training would be compared and the results would be used to determine which model should be used for the application interface.
- Saving/Downloading the Model: Here, we will download the model by saving the learning in a file and then use in the application.

5.3 MODULE DESCRIPTION

The models implemented in this project are:

- Logistic Regression:
  - It is a supervised machine learning algorithm, and it is sort of a statistical algorithm.
  - It is mainly used for the classification type of problem statements.
  - It is used to predict the probability that an instance of belonging to a given class or not.
  - It analyzes the relationship between the independent variables and the dependent variables.
  - The reason behind this algorithm being given the name of regression because it takes the output of the Linear Regression function as input and then uses the sigmoid function to estimate the probability for the given class. The output of linear regression is a continuous one whereas the output of logistic regression is a prediction involving a probability whether the instance belongs to a target class or not.
- Decision Tree Classifier
  - It is a supervised learning algorithm.
  - It can be used for both regression and classification problem statements.
  - It creates a flowchart-like tree structure, with each internal node representing a test on an attribute, each branch representing a test outcome, and each leaf node (terminal node) holding a class label. It is built by iteratively splitting the training data into subsets depending on attribute values until a stopping requirement, such as the maximum depth of the tree or the minimum number of samples needed to divide a node, is met.
  - The level of impurity or unpredictability in the subsets is measured using metrics like entropy or Gini impurity, and the Decision Tree algorithm chooses the optimum attribute to split the data depending on these metrics during training. Finding the property that maximizes information gain or impurity reduction following the split is the objective.
  - Decision tree algorithms employ the Attribute Selection Measure (ASM) criterion to assess the value of various attributes for segmenting datasets. In order to maximise information acquisition, ASM seeks to determine the attribute that will result in the most homogeneous

groups of data following the split. It is known as recursive partitioning to repeat this operation on each derived subset. When the split no longer improves the predictions or when the subset at a node has the same value for the target variable, the recursion is finished. Due to the lack of domain expertise or parameter setup requirements, decision tree classifier design is suitable for exploratory knowledge discovery. High-dimensional data can be handled via decision trees.

- Random Forest Classifier
    - It is a supervised Learning Algorithm.
    - It can be used for both regression and classification problem statements.
    - The Random Forest Algorithm's ability to handle data sets with continuous variables, as in regression, and categorical variables, as in classification, is one of its most crucial qualities. For classification and regression tasks, it performs better. In this lesson, we will examine the random forest's operation and apply it to a classification job.
    - It is different from Decision Tree because Decision Tree suffers from the problem of overfitting if it is allowed to grow uncontrollably whereas Random Forest does not suffer from overfitting because they are created from subsets of data and the final output is based on rankings which solves the problem as a whole.
- Support Vector Classification
    - It is a supervised Learning algorithm.
    - It can be used for both classification and regression problem statements.
    - It is very effective when we have to find the maximum separating hyperplane between the different classes available in the target feature.
    - Finding a hyperplane in an N-dimensional space that clearly classifies the data points is the goal of the SVM method. The number of features determines the hyperplane's size. The hyperplane is essentially a line if there are just two input features. The hyperplane turns into a 2-D plane if there are three input features. Imagining something with more than three features gets challenging.

Chapter 6

Implementation & Testing

6.1 BASE PAPER

Title of the Paper: Machine Learning based Mobile App for Heart Disease Prediction

Authors: S. Dwijesh Reddy; S. Lohitha; Fathimabi Shaik

URL: https://ieeexplore-ieee-org.egateway.vit.ac.in/document/10099714

The paper starts by stating some statistics on deaths caused by Heart Diseases and the way it impacts people and their health. The paper also talks about the way Machine Learning has made an impact in the field of Healthcare and Medicine. The paper also mentions several risks from the Coronary Heart Diseases which sometimes lead to complications in medical treatment and could become fatal to the life. The mentions the use of several Machine Learning and Deep Learning techniques which are used for Predictions. For the research paper, the researchers create a details literature survey of the papers which they referenced during the course of study. In the Design Methodology, the paper focuses more on the Random Forest algorithm as compared to other algorithms. It also mentions that blending several different models could change the outcome as a whole and focuses more on developing an application which could be used easily by everyone. This application has several features within it which include signup, login, registration, view previous prediction reports, and predict based on input. The application uses a document-type database for the user and is connected and hosted with the help of firebase. The application is being developed for mobile devices because they are portable and could be used easily to connect to the application for use which improves the scope of the project and also increases the target audience because almost everyone today has a mobile device and being able to access such application on a mobile would be highly beneficial. The user is also given the benefit of sharing the prediction and details through several platforms. The limitation in this paper is that there isn't much information provided on the learning of the model or the way the learning models were trained.

6.2 DATASET

Name of Dataset: Framingham Heart Study

Study Website: https://www.framinghamheartstudy.org/

The data now contain event follow-up through 2018, auxiliary data from a few sources, and examination data from the first 32 clinical exams.

The Framingham Study aims to investigate familial patterns of cardiovascular disease and risk factors as well as the incidence and prevalence of cardiovascular disease (CVD) and its risk factors. Estimating disease incidence rates and describing the natural history of cardiovascular disease, including the sequence of clinical signs and systems that come before the clinically recognizable syndrome and the effects and progression of clinically manifest disease, are other important goals.

The Framingham Study began in 1948 as part of the United States Public Health Service and was moved in 1949 to the new National Heart Institute, NIH. Participants included both men and women from Framingham, Massachusetts. This was the first prospective research of cardiovascular disease that established the notion of risk variables and their interactions. The project has continued to analyse participants every two years and is currently funded by a contract from the NHLBI to Boston University as well as other funds for specialized investigations.

The study enrolled 5,209 men and women between the ages of 28 and 62 who lived in Framingham, Massachusetts. As of February 28, 1999, there were 993 survivors.

The Framingham Study is a long-term study of the genetic and environmental variables that influence the development of CVD in men and women. Participants were examined every two years, and the cohort was tracked for morbidity and mortality during that time.

Coronary heart disease (angina pectoris, myocardial infarction, coronary insufficiency, and sudden and non-sudden mortality), stroke, hypertension, peripheral arterial disease, and congestive heart failure are among the cardiovascular disease conditions being studied.

6.3 LANGUAGE PLATFORM

Programming Language: Python

6.4 SOURCE CODE

The Google Drive Link for the Project is:

Overall Drive Link:

Machine Learning Model:

User Interface:

Dataset:

The source code for the Machine Learning Model is:

Google Drive Link: https://drive.google.com/drive/folders/1g3Tsx-nljH1eF0pYW12_SQJSUJyict8X?usp=share_link

```
# -*- coding: utf-8 -*-
"""Heart_Disease_Prediction_using_Machine_Learning_Techniques.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://drive.google.com/drive/folders/1g3Tsx-nljH1eF0pYW12_SQJSUJyict8X?usp=share_link

<center>
    <h1><b>Heart Disease Prediction using Machine Learning Techniques</b></h1>
</center>
<p><b>Name:</b> Aashish Bansal</p>
<p><b>Registration No.:</b> 19BIT0346</p>
<p><b>Department:</b> Information Technology</p>
<p><b>School:</b> School of Information Technology and Engineering</p>

# Connecting to Data Source (Google Drive)
"""

from google.colab import drive
drive.mount('/content/mydrive')

"""# Installing Libraries"""
!pip install pyforest

"""# Importing Libraries"""
# Commented out IPython magic to ensure Python compatibility.
from pyforest import *
lazy_imports()
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# %matplotlib inline

"""# Loading the Dataset

## About the Dataset
"""

#load dataset
df = pd.read_csv("https://raw.githubusercontent.com/aashish22bansal/Heart-Disease-
Prediction/main/data/framingham.csv")

"""# Exploratory Data Analysis

## Shape of the Dataset
"""

# Check number of columns and rows in data frame
#Shape of dataset

print ('Number of Examples :', df.shape[0], '\nNumber of Features : ', df.shape[1])

df.head()

"""## Basic Information from the Dataset"""

df.info()

"""Above results shows us that we've :
* (7) x int64 datatype attributes
* (9) x float64 datatype attributes.

## Basic Statistical Deductions
"""

df.describe()

"""Descriptive Statistics :

* The Spread of Age is from 32 to 74.
* Resting blood pressure is distributed from 80 to 200 (in mm Hg).
* Similarly, thalach (maximum heart rate achieved) ranges from 60 to 182.
* While, oldpeak (ST depression induced by exercise relative to rest) values ranges from -2.60
to 3.70.

## Columns in the Dataset
"""

df.columns
```

```
"""# Data Visualization

## Plotting Data from all Columns
"""

#get familiar with the dataset
df.hist(bins=30,
        figsize=(20,40),
        layout=(15,4));

"""## Age Vs Heart Disease"""

# age vs CHD
plt.figure(figsize=(10,10))
sns.swarmplot(x='TenYearCHD', y='age', data=df)

plt.figure(figsize=(10,10))
sns.violinplot(x='TenYearCHD', y='age', data=df)

"""Observations:
* Violinplot tells that most patients of age around 40-55 have 0 risk
* Most patients of age around 60-65 have risk of disease (CHD)

## Age vs Heart Disease for Smokers and Non-Smokers
"""

# age vs CHD for smokers or non-smoker
plt.figure(figsize=(10,10))
sns.swarmplot(x='TenYearCHD', y='age', data=df, hue='currentSmoker')

plt.figure(figsize=(10,10))
sns.violinplot(x='TenYearCHD', y='age', data=df, hue='currentSmoker', split=True)

"""Observations:
* From this violinplot, we see that most of smokers having no risk of CHD are in age around 40
years
* But most of non-smokers having risk are in age around 65-70 years Also most smokers having
risk are in age around 50 years

## Gender Count
"""

# male and female countplot
sns.countplot(x=df['male'])

"""## Gender vs Heart Disease"""

# male and female having disease or not
sns.countplot(x=df['male'], hue=df['TenYearCHD'])

"""Observations:
* Here from the above countplot, we see that most data are females
```

```
* There are more females having no risk than males having no risk
* There are slightly more males having risk than females having risk
"""

df.iloc[:,:5]

"""## Understanding Correlation"""

# To understand correlation between some features, pairplot is used
plt.figure(figsize=(20,15))
sns.pairplot(df.loc[:,'totChol': 'glucose'])

plt.figure(figsize=(15,15))
sns.heatmap(df.corr(), annot=True, linewidths=0.1)

"""Observations:
* From pairplot and heatmap we see that sysBP and diaBP are highly correlated
* And currentSmoker and cigsPerDay are highly correlated
"""

# dropping features which are highly correlated
# features_to_drop = ['currentSmoker', 'diaBP']

# df.drop(features_to_drop, axis=1, inplace=True)

"""# Data Cleaning and Preparation"""

# education feature is not required as its not predicting the Ten Year CHD
# target is Ten Year CHD (0 or 1)
# df.drop('education', axis=1, inplace=True)

# renaming TenYearCHD to CHD
df.rename(columns={"TenYearCHD": "CHD"}, inplace=True)

# Check 5 rows of data set
df.head()

"""## Scaling data

### Checking the Scale
"""

# Normalization Checking
fig, ax = plt.subplots(figsize=(10,10), nrows=3, ncols=4)
ax = ax.flatten()

i = 0
for k,v in df.items():
    sns.displot(v, ax=ax[i])
    i+=1
    if i==12:
        break
```

```
plt.tight_layout(pad=1.25, h_pad=0.8, w_pad=0.8)

# Normalization Checking
fig, ax = plt.subplots(figsize=(10,10), nrows=3, ncols=4)
ax = ax.flatten()

i = 0
for k,v in df.items():
    sns.histplot(v, ax=ax[i])
    i+=1
    if i==12:
        break
plt.tight_layout(pad=1.25, h_pad=0.8, w_pad=0.8)

"""### Obtaining Discrete and Continous Columns"""

# check for column that is not discrete to scale data

#all-columns list
df_all_colum=list(df.columns)

# discrete/categorical column list
discrete_cols=list(df.columns[df.round(decimals=0).isin([0,1]).all()])
discrete_cols.append('education')
discrete_cols.append('BPMeds')

#continuous column list
continuous_cols=list()
for i in df_all_colum:
    if i not in discrete_cols:
        if i not in['education', 'BPMeds']:
            continuous_cols.append(i)

#check if the two list are correct
print('Discrete cols: ', discrete_cols)

print('Continuous cols:', continuous_cols)

"""### MinMaxScaler"""

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

#scale continuous columns data
array_scaled = scaler.fit_transform(df[continuous_cols])

#turn array scaled to dataframe
df_scaled_cols = pd.DataFrame(array_scaled, columns = [df[continuous_cols]])

#create a complete scaled data
df_scaled = df[df_all_colum]
df_scaled[continuous_cols] = df_scaled_cols[continuous_cols]
```

```
#datafram before scaling
df[continuous_cols].hist(bins=30,
                         figsize=(20,40),
                         layout=(15,4)
                        );

#datafram after scaling
df_scaled[continuous_cols].hist(bins=30,
                                figsize=(20,40),
                                layout=(15,4)
                                );

"""### StandardScaler"""

# # Standardise some features
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# cols_to_standardise = ['age','totChol','sysBP','BMI', 'heartRate', 'glucose', 'cigsPerDay']
# train_data[cols_to_standardise] = scaler.fit_transform(train_data[cols_to_standardise])

# #datafram before scaling
# df[continuous_cols].hist(bins=30,
#                          figsize=(20,40),
#                          layout=(15,4)
#                         );

# #datafram after scaling
# df_scaled[continuous_cols].hist(bins=30,
#                                 figsize=(20,40),
#                                 layout=(15,4)
#                                 );

"""## Handling duplicate data """

df_scaled.drop_duplicates()

"""## Handling missing data
Missing values can be done before EDA or after EDA. But before EDA, it will impute or drop
missing values for all features, whether some features are needed or not
"""

!pip install missingno

df_scaled.isna().sum()

#check if there are any null value in dataset
df_scaled.isnull().sum()

df_scaled.isnull().sum().sum()

missing_values_count = df_scaled.isnull().sum()
```

```
missing_values_count = missing_values_count[missing_values_count > 0]
missing_values_percent = (missing_values_count * 100) / (df_scaled.shape[0])
print("The Percentage of Missing Values for the Columns are:")
print(missing_values_percent)

"""
We'll Fill NAN's of all features with median values of that particular feature because mean / average filling values approach won't make any sense since we've discrete values in some features.

One can also try Backward / Forward fill method.

We can observe that fbs and ca has 60.9% and 95.9% missing values respectively so we can drop these features.

Theoretically, 25 to 30% is the maximum missing values are allowed, beyond which we might want to drop the variable from analysis.
"""

print("Maximum missing percentage is {} and hence Imputation is
required.".format(max(missing_values_percent)))

#see if there is any pattern in missing values
import missingno as msno
msno.matrix(df_scaled)

'''
the visualization(matrix) does not show any obvious pattern or cluster in the missing values, hence, we assume that our data is missing completely at random(MCAR)
'''

#see if there is any correlation between missing values
msno.heatmap(df_scaled, figsize=(20, 5))
'''
There seem to be an insignificant correlation between our missing value in colum(tochol) and column(heartRate). Thus, this reinforce our assumption that our missing value is missing completely
at random (MCAR)
'''

"""### SimpleImputer"""

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='most_frequent')

new_train_data_simple_imputer = pd.DataFrame(imputer.fit_transform(df_scaled))
new_train_data_simple_imputer.columns = df_scaled.columns
new_train_data_simple_imputer.index = df_scaled.index

#visualize to see if there are still missing data
msno.matrix(new_train_data_simple_imputer)
```

```
# Data before Imputation
df_scaled.isnull().sum()

# Data after Imputation
new_train_data_simple_imputer.isnull().sum()

new_train_data_simple_imputer.head()

# df_scaled_simple = new_train_data_simple_imputer.copy()

"""### IterativeImputer"""

#As we confirm that our missing value is MCAR, we will use iterative imputing to fill in NA
value
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

#iterative imputer module, impute on the scaled dataset
imputer = IterativeImputer()
df_impute = imputer.fit_transform(df_scaled)

#impute dataframe
df_impute = pd.DataFrame(data=df_impute, columns=df.columns)

#double check for missing value
df_impute.isnull().values.any()

#visualize to see if there are still missing data
msno.matrix(df_impute)

#after imputing, sometime data isn't rounded correctly
#BPMeds
df_impute['BPMeds'] = df_impute['BPMeds'].round(decimals=0)
df_impute['BPMeds'].value_counts()
#education
# df_impute['education'] = df_impute['education'].round(decimals=0)
# df_impute['education'].value_counts()

"""## Handling ouliers"""

from sklearn.model_selection import train_test_split

#Separate label (y) and predictor (X)
X = df_impute.drop('CHD', axis=1)
y = df_impute['CHD']

#Split the data set for winsorization (onliy winsorize on the train set and NOT on the test
set)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

#check
```

```
print('X_train:',X_train.shape,', X_test:', X_test.shape)

fig, ax = plt.subplots(figsize=(10,10), nrows=3, ncols=4)
ax = ax.flatten()

i = 0
for k,v in df_impute.items():
    sns.boxplot(y=v, ax=ax[i])
    i+=1
    if i==12:
        break
plt.tight_layout(pad=1.25, h_pad=0.8, w_pad=0.8)

#visualize outlier (before winsorization)
plt.rcParams["figure.figsize"] = [17, 8]
plt.rcParams["figure.autolayout"] = True
plt.boxplot(x=df_impute[continuous_cols]);
plt.xticks(ticks=range(1,9), labels=np.array(continuous_cols));

"""Conclusion of Boxplot :

Outliers found in features named ['totChol', 'sysBP', 'BMI','heartRate', 'glucose']
"""

# for i in df.columns:
#     sns.countplot(df[i], hue = df['TenYearCHD'], palette = 'Set1')
#     plt.show()

from scipy.stats.mstats import winsorize
from scipy.stats import mstats

# df_winzore = df_impute
#Winzorize the 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose' , 'cigPerDay' columns
totChol_wins = mstats.winsorize(X_train['totChol'], limits=[0.02, 0.009])
sysBP_wins = mstats.winsorize(X_train['sysBP'], limits=[0.02, 0.03])
diaBP_wins = mstats.winsorize(X_train['diaBP'], limits=[0.02, 0.01])
BMI_wins = mstats.winsorize(X_train['BMI'], limits=[0.02, 0.01])
heartRate_wins = mstats.winsorize(X_train['heartRate'], limits=[0.03, 0.01])
glucose_wins = mstats.winsorize(X_train['glucose'], limits=[0.02, 0.02])
cigsPerDay_wins = mstats.winsorize(X_train['cigsPerDay'], limits=[0.02, 0.01])

# Add the winsorized 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'Glucose' columns back to
the train DataFrame
X_train['totChol'] = totChol_wins
X_train['sysBP'] = sysBP_wins
X_train['diaBP'] = diaBP_wins
X_train['BMI'] = BMI_wins
X_train['heartRate'] = heartRate_wins
X_train['glucose'] =glucose_wins
X_train['cigsPerDay'] =cigsPerDay_wins

# visualize outlier (after winsorization)
```

```
plt.boxplot(x=X_train[continuous_cols]);
plt.xticks(ticks=range(1,9), labels=np.array(continuous_cols));

# oulier distribution before winsorization
df_impute[continuous_cols].hist(bins=30,
                figsize=(20,30),
                layout=(15,4));

# oulier distribution after winsorization
X_train[continuous_cols].hist(bins=30,
                figsize=(20,30),
                layout=(15,4));

"""## Handling imbalanced data"""

!pip install imblearn

#SMOTE-tomek (SMOTE-Tomek is a combination of oversampling (SMOTE) and undersampling (Tomek
links) techniques)
from imblearn.combine import SMOTETomek

#Create the SMOTE-tomek variable
smote_tomek = SMOTETomek(random_state=42)

#Applie SMOTE-tomek to train data
X_train_resampled, y_train_resampled = smote_tomek.fit_resample(X_train, y_train)

#Original train dataset
X_train.count(), y_train.value_counts()

#visualize original imbalanced data
y_train.hist(figsize=(20,10))

#Resample train dataset
X_train_resampled.count(), y_train_resampled.value_counts()

#visualize balanced data after SMOTE-tomek
y_train_resampled.hist(figsize=(20,10))

"""# Feature Selection"""

import seaborn as sns

#visualize features
corrmat= X_train_resampled.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,10))
g=sns.heatmap(X_train_resampled[top_corr_features].corr(),annot=True, cmap='RdYlGn')

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

```
# remove label column from bianry(categorical) list
discrete_cols.remove('CHD')

#calculating chi2 score for CATEGORICAL variables
bestfeature = SelectKBest(score_func=chi2, k=7)
fit = bestfeature.fit(X_train_resampled[discrete_cols], y_train_resampled)

# Pass the result into a dataframe
dfp_values = pd.DataFrame(fit.pvalues_)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X_train_resampled[discrete_cols].columns)
featureScores = pd.concat([dfcolumns,dfscores,dfp_values],axis=1)
featureScores.columns = ['Specs','Score','p-value']
featureScores_largest = featureScores.nlargest(7,'Score')
print(featureScores.nlargest(7,'Score'))

#my feature list
feature_list = list(X_train_resampled.columns)
feature_list.remove('education')
feature_list.remove('currentSmoker')
feature_list.remove('glucose')

feature_list

import seaborn as sns
# visulize top features
plt.figure(figsize=(20,15))
g=sns.heatmap(X_train_resampled[feature_list].corr(),annot=True, cmap='RdYlGn')

"""# Model Fitting

## Logistic Regression
"""

X_train_resampled.shape

"""### Importing Model Library"""

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

"""### Setting Class Weights"""

# Class Weighting (more weight on Minority Class - Class 1)
class_weight_LR = {0: 1, 1: 1.2}

"""### Normal Parameter Training

#### Creating Model Function
"""

# Create LogisticRegression Model
```

```
Model_LogisticRegression_Normal = LogisticRegression(max_iter=1000, solver='liblinear',
class_weight=class_weight_LR)

"""#### Fitting Model"""

Model_LogisticRegression_Normal.fit(X_train, y_train)

"""#### Predicting"""

y_train_proba_LR_normal = Model_LogisticRegression_Normal.predict_proba(X_train)
y_test_proba_LR_normal = Model_LogisticRegression_Normal.predict_proba(X_test)

y_train_pred_LR_normal = Model_LogisticRegression_Normal.predict(X_train)
y_test_pred_LR_normal = Model_LogisticRegression_Normal.predict(X_test)

"""#### Model Evaluations"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test, y_test_pred_LR_normal)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred_LR_normal)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_pred_LR_normal))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred_LR_normal))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_pred_LR_normal))
print('Testing Precision: ', precision_score(y_test, y_test_pred_LR_normal))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_pred_LR_normal))
print('Testing Recall: ', recall_score(y_test, y_test_pred_LR_normal))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_pred_LR_normal))
print('Testing F1-Score: ', f1_score(y_test, y_test_pred_LR_normal))
```

```
"""#### Model Results

##### Classification Report (Precision, Recall & F1 Score)
"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_LR = classification_report(y_test, y_test_pred_LR_normal,
output_dict=True)
pd.DataFrame(classification_report_LR)

"""##### ROC Curve"""

#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_LR, tpr_LR, thresholds_LR = roc_curve(y_test, y_test_pred_LR_normal)
roc_auc_LR = auc(fpr_LR, tpr_LR)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_LR, tpr_LR, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_LR)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Import roc_curve, auc
from sklearn.metrics import roc_curve, auc

# Calculate the probability scores of each point in the training set
y_train_score_LR_Normal = Model_LogisticRegression_Normal.decision_function(X_train)

# Calculate the fpr, tpr, and thresholds for the training set
train_fpr_LR_Normal, train_tpr_LR_Normal, thresholds_LR_Normal = roc_curve(y_train,
y_train_score_LR_Normal)

# Calculate the probability scores of each point in the test set
y_test_score_LR_Normal = Model_LogisticRegression_Normal.decision_function(X_test)

# Calculate the fpr, tpr, and thresholds for the test set
test_fpr_LR_Normal, test_tpr_LR_Normal, test_thresholds_LR_Normal = roc_curve(y_test,
y_test_score_LR_Normal)

# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline

# Seaborn's beautiful styling
```

```
sns.set_style('darkgrid', {'axes.facecolor': '0.9'})

# ROC curve for training set
plt.figure(figsize=(10, 8))
lw = 2
plt.plot(train_fpr_LR_Normal, train_tpr_LR_Normal, color='darkorange', lw=lw, label='ROC
curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.yticks([i/20.0 for i in range(21)])
plt.xticks([i/20.0 for i in range(21)])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) Curve for Training Set')
plt.legend(loc='lower right')
print('Training AUC: {}'.format(auc(train_fpr_LR_Normal, train_tpr_LR_Normal)))
plt.show()

# ROC curve for test set
plt.figure(figsize=(10, 8))
lw = 2
plt.plot(test_fpr_LR_Normal, test_tpr_LR_Normal, color='darkorange', lw=lw, label='ROC curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.yticks([i/20.0 for i in range(21)])
plt.xticks([i/20.0 for i in range(21)])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) Curve for Test Set')
plt.legend(loc='lower right')
print('Test AUC: {}'.format(auc(test_fpr_LR_Normal, test_tpr_LR_Normal)))
print('')
plt.show()

fpr, tpr, _ = roc_curve(y_train, y_train_proba_LR_normal[:,1])

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_test_proba_LR_normal[:,1])

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
```

```
plt.show()

"""##### ROC AUC (On Training Data)"""

from sklearn.metrics import roc_auc_score

roc_auc_score(y_train, y_train_proba_LR_normal[:,1])

"""##### ROC AUC (On Testing Data)"""

roc_auc_score(y_test, y_test_proba_LR_normal[:,1])

"""### Tuned Parameter Training

#### Defining Tunning Parameters
"""

# Defining Tunning Parameters
Hyperparameters_LR = {'C':[0.001, 0.01, 0.1, 1, 10, 100, 1000], 'penalty': ['l1', 'l2']}

"""#### Creating GridSearchCV Model"""

# Creating GridSearchCV Model
Model_GridSearch_LR_Tuned = GridSearchCV(Model_LogisticRegression_Normal, Hyperparameters_LR,
cv = 5, scoring = 'roc_auc')

"""#### Fitting data into GridSearchCV"""

# Fitting data into GridSearchCV
Model_GridSearch_LR_Tuned.fit(X_train_resampled[feature_list] , y_train_resampled)

"""#### Obtaining best Hyperparameters results and Model Score"""

# Obtaining best Hyperparameters results and Model Score
print(Model_GridSearch_LR_Tuned.best_params_, Model_GridSearch_LR_Tuned.best_score_)

# Model_GridSearch_LR_Tuned_Fit = Model_GridSearch_LR_Tuned.fit(X_train_resampled[feature_list]
, y_train_resampled)

"""#### Obtaining Model Coefficients"""

# Model Coefficients
Model_GridSearch_LR_Tuned.best_estimator_.coef_

"""#### Model Intercepts"""

# Model Intercepts
Model_GridSearch_LR_Tuned.best_estimator_.intercept_

"""#### Predicting"""

y_train_proba_LR_Tuned = Model_GridSearch_LR_Tuned.predict_proba(X_train[feature_list])[:,1]
```

```
y_test_proba_LR_Tuned = Model_GridSearch_LR_Tuned.predict_proba(X_test[feature_list])[:,1]

y_train_predict_LR_Tuned = Model_GridSearch_LR_Tuned.predict(X_train[feature_list])
y_test_predict_LR_Tuned = Model_GridSearch_LR_Tuned.predict(X_test[feature_list])

"""#### Checking Threshold Value"""

# Threshold value can be adjusted to prioritize certain types of errors over others.
# ex: customize threshold to increase sensitivity  to patience with the disease (reduce false
negative)
y_pred_test_LR_Tuned = (y_test_proba_LR_Tuned > 0.5).astype(int)

"""#### Model Evaluations"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion matrix """

confusion_matrix(y_test, y_test_predict_LR_Tuned)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_predict_LR_Tuned)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_predict_LR_Tuned))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_predict_LR_Tuned))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_predict_LR_Tuned))
print('Testing Precision: ', precision_score(y_test, y_test_predict_LR_Tuned))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_predict_LR_Tuned))
print('Testing Recall: ', recall_score(y_test, y_test_predict_LR_Tuned))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_predict_LR_Tuned))
print('Testing F1-Score: ', f1_score(y_test, y_test_predict_LR_Tuned))

"""#### Model Results

##### Classification Report (Precision, Recall & F1 Score)
```

```
"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_LR = classification_report(y_test, y_test_predict_LR_Tuned,
output_dict=True)
pd.DataFrame(classification_report_LR)

"""##### ROC Curve"""

#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_LR, tpr_LR, thresholds_LR = roc_curve(y_test, y_test_predict_LR_Tuned)
roc_auc_LR = auc(fpr_LR, tpr_LR)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_LR, tpr_LR, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_LR)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

"""##### ROC AUC (On Training Data)"""

roc_auc_score(y_train, y_train_proba_LR_Tuned)

"""##### ROC AUC (On Testing Data)"""

roc_auc_score(y_test, y_test_proba_LR_Tuned)

"""## Decision Tree Classifier

### Importing Model Library
"""

from sklearn.tree import DecisionTreeClassifier

"""### Normal Parameter Tuning

#### Creating Model Function
"""

Model_DecisionTreeClassifier_Normal = DecisionTreeClassifier(min_samples_split=40,
random_state=0)
# that fraction of samples(if float) or that many number(if int) of samples is atleast present
in the node
# before splitting, then only split that node
```

```
# for min_samples_split as 180 I got a better accuracy and train score and difference was less
# but f1 score was very bad for positive class
# and setting min_samples_split as 40, we got good results for all metrics

"""#### Fitting Model"""

Model_DecisionTreeClassifier_Normal.fit(X_train, y_train)

"""#### Predicting"""

y_train_proba_DT_normal = Model_DecisionTreeClassifier_Normal.predict_proba(X_train)
y_test_proba_DT_normal = Model_DecisionTreeClassifier_Normal.predict_proba(X_test)

y_train_pred_DT_normal = Model_DecisionTreeClassifier_Normal.predict(X_train)
y_test_pred_DT_normal = Model_DecisionTreeClassifier_Normal.predict(X_test)

"""#### Model Evaluations"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test, y_test_pred_DT_normal)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred_DT_normal)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_pred_DT_normal))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred_DT_normal))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_pred_DT_normal))
print('Testing Precision: ', precision_score(y_test, y_test_pred_DT_normal))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_pred_DT_normal))
print('Testing Recall: ', recall_score(y_test, y_test_pred_DT_normal))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_pred_DT_normal))
print('Testing F1-Score: ', f1_score(y_test, y_test_pred_DT_normal))
```

```
"""#### Modal Results

##### Classification Report (Precision, Recall & F1 Score)
"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_LR = classification_report(y_test, y_test_pred_DT_normal,
output_dict=True)
pd.DataFrame(classification_report_LR)

from sklearn.metrics import roc_curve, roc_auc_score

"""##### ROC AUC (On Training Data)"""

roc_auc_score(y_train, y_train_proba_DT_normal[:,1])

"""##### ROC AUC (On Testing Data)"""

roc_auc_score(y_test, y_test_proba_DT_normal[:,1])

"""##### ROC Curve"""

#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_LR, tpr_LR, thresholds_LR = roc_curve(y_test, y_test_pred_DT_normal)
roc_auc_LR = auc(fpr_LR, tpr_LR)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_LR, tpr_LR, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_LR)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

fpr, tpr, _ = roc_curve(y_train, y_train_proba_DT_normal[:,1])

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_test_proba_DT_normal[:,1])

plt.clf()
```

```
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

"""#### Obtaining the Decision Tree"""

# Exporting the tree in text format
from sklearn.tree import export_text
text_format_DT = export_text(Model_DecisionTreeClassifier_Normal, feature_names =
list(df_impute.columns[:15]))
print('Decision tree in text format : \n%s'%text_format_DT)

from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(Model_DecisionTreeClassifier_Normal,
                   feature_names=X_train_resampled.columns,
                   class_names=['No Disease', "Disease"],
                   filled=True)

"""### Tuned Parameter Training

#### Defining Tunning Parameters
"""

hyperparameters_DT = {
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'criterion': ["gini", "entropy"]
}

"""#### Creating GridSearchCV Model"""

# Creating GridSearchCV Model
Model_GridSearch_DT_Tuned = GridSearchCV(Model_DecisionTreeClassifier_Normal,
hyperparameters_DT, cv = 5, scoring = 'roc_auc')

"""#### Fitting data into GridSearchCV"""

# Fitting data into GridSearchCV
Model_GridSearch_DT_Tuned.fit(X_train_resampled[feature_list] , y_train_resampled)

"""#### Obtaining best Hyperparameters results and Model Score"""

# Obtaining best Hyperparameters results and Model Score
print(Model_GridSearch_DT_Tuned.best_params_, Model_GridSearch_DT_Tuned.best_score_)

best_estimator_DT = Model_GridSearch_DT_Tuned.best_estimator_
best_estimator_DT

"""#### Re-fitting Data into the Model"""
```

```
# Model_DecisionTreeClassifier_Refit.fit(X_train, y_train)

"""#### Obtaining Model Coefficients"""

# Model Coefficients
# Model_GridSearch_DT_Tuned.best_estimator_.coef_
Model_GridSearch_DT_Tuned.best_estimator_

"""#### Model Intercepts"""

# Model Intercepts
Model_GridSearch_LR_Tuned.best_estimator_.intercept_

"""#### Predicting"""

y_train_proba_DT_Tuned = Model_GridSearch_DT_Tuned.predict_proba(X_train[feature_list])[:,1]
y_test_proba_DT_Tuned = Model_GridSearch_DT_Tuned.predict_proba(X_test[feature_list])[:,1]

y_train_predict_DT_Tuned = Model_GridSearch_DT_Tuned.predict(X_train[feature_list])
y_test_predict_DT_Tuned = Model_GridSearch_DT_Tuned.predict(X_test[feature_list])

"""#### Model Evaluation"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

"""##### Confusion Matrix"""

confusion_matrix(y_test, y_test_predict_DT_Tuned)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_predict_DT_Tuned)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_predict_DT_Tuned))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_predict_DT_Tuned))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_predict_DT_Tuned))
print('Testing Precision: ', precision_score(y_test, y_test_predict_DT_Tuned))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_predict_DT_Tuned))
```

```
print('Testing Recall: ', recall_score(y_test, y_test_predict_DT_Tuned))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_predict_DT_Tuned))
print('Testing F1-Score: ', f1_score(y_test, y_test_predict_DT_Tuned))

"""##### Accuracy Score"""

accuracy_DT = accuracy_score(y_test_predict_DT_Tuned, y_test)*100
print('Accuracy score for Decision tree is %f'%accuracy_DT)

"""##### Train Score"""

train_score_DT = Model_GridSearch_DT_Tuned.score(X_train_resampled[feature_list],
y_train_resampled)*100
print('Train score for Decision tree is %f'%train_score_DT)

"""##### Difference between Training and Testing Score"""

print('Difference between train and test scores for Decision tree is : %f'%(train_score_DT -
accuracy_DT))

"""#### Model Results"""

score_DT = pd.DataFrame(Model_GridSearch_DT_Tuned.cv_results_)
score_DT

"""##### Classification Report (Precision, Recall & F1 Score)"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_DT = classification_report(y_test, y_test_predict_DT_Tuned,
output_dict=True)
pd.DataFrame(classification_report_DT)

"""##### ROC Curve"""

#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_DT, tpr_DT, thresholds_DT = roc_curve(y_test, y_test_predict_DT_Tuned)
roc_auc_DT = auc(fpr_DT, tpr_DT)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_DT, tpr_DT, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_DT)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
```

```
plt.show()

#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_LR, tpr_LR, thresholds_LR = roc_curve(y_test, y_test_predict_DT_Tuned)
roc_auc_LR = auc(fpr_LR, tpr_LR)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_LR, tpr_LR, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_LR)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

fpr, tpr, _ = roc_curve(y_train, y_train_proba_DT_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_test_proba_DT_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

"""##### Confusion Matrix"""

confusion_matrix(y_test_predict_DT_Tuned, y_test)

"""##### Classification Report"""

print(classification_report(y_test_predict_DT_Tuned, y_test))

"""## Random Forest Classifier

### Importing Model Library
"""

from sklearn.ensemble import RandomForestClassifier
```

```
"""### Normal Parameter Training

#### Creating Model Function
"""

Model_RandomForestClassifier_Normal = RandomForestClassifier(n_estimators = 150,
min_samples_split = 10, random_state = 0)

"""#### Fitting Model"""

Model_RandomForestClassifier_Normal.fit(X_train, y_train)

"""#### Predicting"""

y_train_proba_RF_normal = Model_RandomForestClassifier_Normal.predict_proba(X_train)
y_test_proba_RF_normal = Model_RandomForestClassifier_Normal.predict_proba(X_test)

y_train_pred_RF_normal = Model_RandomForestClassifier_Normal.predict(X_train)
y_test_pred_RF_normal = Model_RandomForestClassifier_Normal.predict(X_test)

"""#### Model Evaluations"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test, y_test_pred_RF_normal)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred_RF_normal)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_pred_RF_normal))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred_RF_normal))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_pred_RF_normal))
print('Testing Recall: ', recall_score(y_test, y_test_pred_RF_normal))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_pred_RF_normal))
print('Testing Precision: ', precision_score(y_test, y_test_pred_RF_normal))
```

```
"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_pred_RF_normal))
print('Testing F1-Score: ', f1_score(y_test, y_test_pred_RF_normal))

"""#### Model Results

##### Classification Report (Precision, Recall & F1 Score)
"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_DT = classification_report(y_test, y_test_predict_DT_Tuned,
output_dict=True)
pd.DataFrame(classification_report_DT)

"""##### ROC Curve"""

roc_auc_score(y_train, y_train_proba_RF_normal[:,1])

roc_auc_score(y_test, y_test_proba_RF_normal[:,1])

fpr, tpr, _ = roc_curve(y_train, y_train_proba_RF_normal[:,1])

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_test_proba_RF_normal[:,1])

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

"""### Tuned Parameter Training

#### Defining Tuning Parameters
"""

n_estimators = [5,20,50,100]                                    # number of trees in the random
forest
max_features = ['sqrt']                                         # number of features in
consideration at every split ('auto' is deprecated)
max_depth    = [int(x) for x in np.linspace(10, 120, num = 12)] # maximum number of levels
allowed in each decision tree
min_samples_split = [2, 6, 10]                                  # minimum sample number to
split a node
```

```
min_samples_leaf = [1, 3, 4]                                    # minimum sample number that
can be stored in a leaf node
bootstrap = [True, False]                                       # method used to sample data
points

hyperparameters_RF = {'n_estimators': n_estimators,
                      'max_features': max_features,
                      'max_depth': max_depth,
                      'min_samples_split': min_samples_split,
                      'min_samples_leaf': min_samples_leaf,
                      'bootstrap': bootstrap
                      }

"""#### Creating GridSearchCV Model"""

# Creating GridSearchCV Model
Model_GridSearchCV_RF_Tuned = GridSearchCV(Model_RandomForestClassifier_Normal,
hyperparameters_RF, cv = 5, scoring = 'roc_auc')

"""#### Creating RandomizedSearchCV Model"""

from sklearn.model_selection import RandomizedSearchCV
Model_RandomizedSearchCV_RF_Tuned = RandomizedSearchCV(estimator =
Model_RandomForestClassifier_Normal,
                                       param_distributions = hyperparameters_RF,
                                       n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs
= -1)

Model_RandomizedSearchCV_RF_Tuned

"""#### Fitting data into RandomizedSearchCV"""

Model_RandomizedSearchCV_RF_Tuned.fit(X_train_resampled[feature_list] , y_train_resampled)

# Fitting data into GridSearchCV
# WAY TOO TIME CONSUMING
# Model_GridSearchCV_RF_Tuned.fit(X_train_resampled[feature_list] , y_train_resampled),

"""#### Obtaining best Hyperparameters results and Model Score"""

print(Model_RandomizedSearchCV_RF_Tuned.best_estimator_)

# Obtaining best Hyperparameters results and Model Score
print(Model_RandomizedSearchCV_RF_Tuned.best_params_,
Model_RandomizedSearchCV_RF_Tuned.best_score_)

print(Model_RandomizedSearchCV_RF_Tuned.cv_results_)

best_estimator_RF = Model_RandomizedSearchCV_RF_Tuned.best_estimator_
best_estimator_RF

"""#### Predicting"""
```

```
y_train_proba_RF_Tuned =
Model_RandomizedSearchCV_RF_Tuned.predict_proba(X_train[feature_list])[:,1]
y_test_proba_RF_Tuned =
Model_RandomizedSearchCV_RF_Tuned.predict_proba(X_test[feature_list])[:,1]

y_train_predict_RF_Tuned = Model_RandomizedSearchCV_RF_Tuned.predict(X_train[feature_list])
y_test_predict_RF_Tuned = Model_RandomizedSearchCV_RF_Tuned.predict(X_test[feature_list])

"""#### Checking Threshold Value"""

# Threshold value can be adjusted to prioritize certain types of errors over others.
# ex: customize threshold to increase sensitivity  to patience with the disease (reduce false
negative)
# y_pred_test_LR_Tuned = (y_test_proba_RF_Tuned > 0.5).astype(int)

"""#### Obtaining Model Coefficients"""

# Model Coefficients
# Model_RandomizedSearchCV_RF_Tuned.best_estimator_.coef_
Model_RandomizedSearchCV_RF_Tuned.best_estimator_

"""#### Model Intercepts"""

# Model Intercepts
# Model_RandomizedSearchCV_RF_Tuned.best_estimator_.intercept_

pd.DataFrame(y_test_proba_RF_Tuned)

"""#### Fitting Data into the Model"""

# Model_RandomizedSearchCV_RF_Tuned.fit(X_train, y_train)

"""#### Model Evaluation

##### Accuracy Score
"""

accuracy_RF = accuracy_score(y_test_predict_RF_Tuned, y_test)*100
print('Accuracy score for Random Forest is %f'%accuracy_RF)

"""##### Training Score"""

train_score_RF = Model_RandomizedSearchCV_RF_Tuned.score(X_train[feature_list], y_train)*100
print('Train score for Random Forest is %f'%train_score_RF)

"""##### Difference between Training and Testing Score"""

print('Difference between train and test scores for Random Forest is : %f'%(train_score_RF -
accuracy_RF))

"""#### Model Results"""
```

```
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test_predict_RF_Tuned, y_test)

# Confusion matrix
conf_matrix = confusion_matrix(y_test_predict_RF_Tuned, y_test)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_predict_RF_Tuned))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_predict_RF_Tuned))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_predict_RF_Tuned))
print('Testing Precision: ', precision_score(y_test, y_test_predict_RF_Tuned))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_predict_RF_Tuned))
print('Testing Recall: ', recall_score(y_test, y_test_predict_RF_Tuned))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_predict_RF_Tuned))
print('Testing F1-Score: ', f1_score(y_test, y_test_predict_RF_Tuned))

"""##### Classification Report"""

print(classification_report(y_test_predict_RF_Tuned, y_test))

"""##### ROC Curve"""

roc_auc_score(y_train, y_train_proba_RF_Tuned)

roc_auc_score(y_test, y_test_proba_RF_Tuned)

fpr, tpr, _ = roc_curve(y_train, y_train_proba_RF_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()
```

```
fpr, tpr, _ = roc_curve(y_test, y_test_proba_RF_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

"""## SVC

### Importing Model Library
"""

from sklearn.svm import SVC

from sklearn.model_selection import cross_val_score

"""### Normal Parameter Training

#### Creating Model Function
"""

Model_SVC_Normal = SVC(random_state=42, probability=True)

"""#### Fitting Model"""

Model_SVC_Normal.fit(X_train, y_train)

"""#### Predicting"""

y_train_pred_SVC_normal = Model_SVC_Normal.predict(X_train)
y_test_pred_SVC_normal = Model_SVC_Normal.predict(X_test)

y_train_proba_SVC_normal = Model_SVC_Normal.predict_proba(X_train)
y_test_proba_SVC_normal = Model_SVC_Normal.predict_proba(X_test)

"""#### Model Evaluations"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test, y_test_pred_SVC_normal)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred_SVC_normal)
plt.figure(figsize=(10,5))
```

```
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_pred_SVC_normal))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred_SVC_normal))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_pred_SVC_normal))
print('Testing Precision: ', precision_score(y_test, y_test_pred_SVC_normal))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_pred_SVC_normal))
print('Testing Recall: ', recall_score(y_test, y_test_pred_SVC_normal))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_pred_SVC_normal))
print('Testing F1-Score: ', f1_score(y_test, y_test_pred_SVC_normal))

"""#### Modal Results

##### Classification Report
"""

# Classification Report (Precision, Recall & F1 Score)
classification_report_SVC = classification_report(y_test, y_test_pred_SVC_normal,
output_dict=True)
pd.DataFrame(classification_report_SVC)

"""##### ROC Curve"""

# Import roc_curve, auc
from sklearn.metrics import roc_curve, auc

# Calculate the probability scores of each point in the training set
y_train_score_SVC_Normal = Model_SVC_Normal.decision_function(X_train)

# Calculate the fpr, tpr, and thresholds for the training set
train_fpr_SVC_Normal, train_tpr_SVC_Normal, thresholds_SVC_Normal = roc_curve(y_train,
y_train_score_SVC_Normal)

# Calculate the probability scores of each point in the test set
y_test_score_SVC_Normal = Model_SVC_Normal.decision_function(X_test)

# Calculate the fpr, tpr, and thresholds for the test set
test_fpr_SVC_Normal, test_tpr_SVC_Normal, test_thresholds_SVC_Normal = roc_curve(y_test,
y_test_score_SVC_Normal)
```

```
# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline

# Seaborn's beautiful styling
sns.set_style('darkgrid', {'axes.facecolor': '0.9'})

# ROC curve for training set
plt.figure(figsize=(10, 8))
lw = 2
plt.plot(train_fpr_SVC_Normal, train_tpr_SVC_Normal, color='darkorange', lw=lw, label='ROC
curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.yticks([i/20.0 for i in range(21)])
plt.xticks([i/20.0 for i in range(21)])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) Curve for Training Set')
plt.legend(loc='lower right')
print('Training AUC: {}'.format(auc(train_fpr_SVC_Normal, train_tpr_SVC_Normal)))
plt.show()

# ROC curve for test set
plt.figure(figsize=(10, 8))
lw = 2
plt.plot(test_fpr_SVC_Normal, test_tpr_SVC_Normal, color='darkorange', lw=lw, label='ROC
curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.yticks([i/20.0 for i in range(21)])
plt.xticks([i/20.0 for i in range(21)])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) Curve for Test Set')
plt.legend(loc='lower right')
print('Test AUC: {}'.format(auc(test_fpr_SVC_Normal, test_tpr_SVC_Normal)))
print('')
plt.show()

roc_auc_score(y_train, y_train_proba_SVC_normal[:,1])

roc_auc_score(y_test, y_test_proba_SVC_normal[:,1])

fpr, tpr, _ = roc_curve(y_train, y_train_proba_RF_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
```

```
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

fpr, tpr, _ = roc_curve(y_test, y_test_proba_RF_Tuned)

plt.clf()
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
plt.show()

"""### Tuned Parameter Training

#### Defining Tuning Parameters
"""

hyperparameters_SVC = {
    'kernel': ['poly', 'rbf', 'sigmoid'],
    'degree': [2, 3, 4, 5, 6],
    'gamma' : ['scale', 'auto']
}

"""#### Creating GridSearchCV Model"""

# Model_GridSearchCV_SVC_Tuned = GridSearchCV(estimator = Model_SVC_Normal, param_grid =
hyperparameters_SVC, cv = 10)

Model_RandomizedSearchCV_SVC_Tuned = RandomizedSearchCV(estimator = Model_SVC_Normal,
                                     param_distributions = hyperparameters_SVC,
                                     n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs
= -1)

"""#### Fitting Data into GridSearchCV Model"""

Model_RandomizedSearchCV_SVC_Tuned.fit(X_train_resampled[feature_list] , y_train_resampled)

"""#### Obtaining Model Best Parameters"""

print(Model_RandomizedSearchCV_SVC_Tuned.best_params_)
print(Model_RandomizedSearchCV_SVC_Tuned.best_score_)

"""#### Predicting"""

y_train_proba_SVC_Tuned =
Model_RandomizedSearchCV_SVC_Tuned.predict_proba(X_train[feature_list])[:,1]
y_test_proba_SVC_Tuned =
Model_RandomizedSearchCV_SVC_Tuned.predict_proba(X_test[feature_list])[:,1]

y_train_predict_SVC_Tuned = Model_RandomizedSearchCV_SVC_Tuned.predict(X_train[feature_list])
y_test_predict_SVC_Tuned = Model_RandomizedSearchCV_SVC_Tuned.predict(X_test[feature_list])
```

```
"""#### Obtaining Model Coefficients"""

# Model Coefficients
Model_RandomizedSearchCV_SVC_Tuned.best_estimator_

if(Model_RandomizedSearchCV_SVC_Tuned.best_estimator_.kernel == 'linear'):
  print(Model_RandomizedSearchCV_SVC_Tuned.best_estimator_.coef_)
else:
  print("Cofficient is available only when kernel is linear.")

"""### Model Evaluation"""

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, classification_report,
auc

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

"""##### Confusion Matrix"""

confusion_matrix(y_test_predict_SVC_Tuned, y_test)

# Confusion matrix
conf_matrix = confusion_matrix(y_test_predict_SVC_Tuned, y_test)
plt.figure(figsize=(10,5))
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.1%', cmap='Oranges',
linewidths=5, annot_kws={"fontsize":16})

"""##### Accuracy"""

print('Training Accuracy: ', accuracy_score(y_train, y_train_predict_SVC_Tuned))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_predict_SVC_Tuned))

"""##### Precision"""

print('Training Precision: ', precision_score(y_train, y_train_predict_SVC_Tuned))
print('Testing Precision: ', precision_score(y_test, y_test_predict_SVC_Tuned))

"""##### Recall"""

print('Training Recall: ', recall_score(y_train, y_train_predict_SVC_Tuned))
print('Testing Recall: ', recall_score(y_test, y_test_predict_SVC_Tuned))

"""##### F1-Score"""

print('Training F1-Score: ', f1_score(y_train, y_train_predict_SVC_Tuned))
print('Testing F1-Score: ', f1_score(y_test, y_test_predict_SVC_Tuned))

"""### Model Results

##### Classification Report (Precision, Recall & F1 Score)
"""
```

```
# Classification Report (Precision, Recall & F1 Score)
classification_report_SVC = classification_report(y_test, y_test_predict_SVC_Tuned,
output_dict=True)
pd.DataFrame(classification_report_SVC)

"""##### ROC Curve"""

# Import roc_curve, auc
from sklearn.metrics import roc_curve, auc

# Calculate the probability scores of each point in the training set
y_train_score_SVC_Tuned =
Model_RandomizedSearchCV_SVC_Tuned.decision_function(X_train[feature_list])

# Calculate the fpr, tpr, and thresholds for the training set
train_fpr_SVC_Tuned, train_tpr_SVC_Tuned, thresholds_SVC_Tuned = roc_curve(y_train,
y_train_score_SVC_Tuned)

# Calculate the probability scores of each point in the test set
y_test_score_SVC_Tuned =
Model_RandomizedSearchCV_SVC_Tuned.decision_function(X_test[feature_list])

# Calculate the fpr, tpr, and thresholds for the test set
test_fpr_SVC_Tuned, test_tpr_SVC_Tuned, test_thresholds_SVC_Tuned = roc_curve(y_test,
y_test_score_SVC_Tuned)

# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline

# Seaborn's beautiful styling
sns.set_style('darkgrid', {'axes.facecolor': '0.9'})

# ROC curve for training set
plt.figure(figsize=(10, 8))
lw = 2
plt.plot(train_fpr_SVC_Tuned, train_tpr_SVC_Tuned, color='darkorange', lw=lw, label='ROC
curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.yticks([i/20.0 for i in range(21)])
plt.xticks([i/20.0 for i in range(21)])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) Curve for Training Set')
plt.legend(loc='lower right')
print('Training AUC: {}'.format(auc(train_fpr_SVC_Tuned, train_tpr_SVC_Tuned)))
plt.show()
```

```
#Roc curve
#pass value in to roc_curve variable: fpr, tpr, thresholds
fpr_SVC, tpr_SVC, thresholds_SVC = roc_curve(y_test, y_test_predict_SVC_Tuned)
roc_auc_SVC = auc(fpr_SVC, tpr_SVC)

#plot it
plt.figure(figsize=(10,5))
plt.plot(fpr_SVC, tpr_SVC, lw=1, label='ROC curve (AUC = %0.2f)' % roc_auc_SVC)
plt.plot([0, 1], [0, 1], '--', color='gray', lw=2)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

"""#### Cross Validation Score"""

Model_GridSearchCV = SVC(degree = 2, gamma = 'scale', kernel= 'poly')
print(cross_val_score(Model_RandomizedSearchCV_SVC_Tuned, X_train, y_train, cv = 10, scoring =
'accuracy'))

"""# Comparison of Results

Out of all the models taken into consideration, the Model with the highest accuracy is the
Random Forest Classifier. Hence, this model would be downloaded and used in the interface.

# Saving the Model for User Interface
"""

#create a pickle file using serialization (for streamlit)
import pickle
pickle_out = open("/content/mydrive/MyDrive/Capstone Project/Heart Disease
Prediction/Outputs/Saved Model/model.pk1", "wb")
pickle.dump(Model_RandomForestClassifier_Normal, pickle_out)
pickle_out.close()
```

The source code for the User Interface is:

```
import streamlit as st
import pickle
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

```
#the model
pickle_in= open("model.pk1","rb")
classifier=pickle.load(pickle_in)

st.set_page_config(layout="wide")

#first section
st.markdown(""" # Coronary Heart Disease (CHD) Prediction """, unsafe_allow_html=True)
#objective
st.divider()
st.subheader('Goal:')
st.write('The goal of this project is to predict whether a candidate will suffer from
heart disease and help people identify high-risk candidates and prevent adverse
selection.')
st.divider()

#predict the label (i.e. 0, 1)
def prediction_label(x):
    prediction_label=classifier.predict(x)
    return prediction_label
#predict the probability (%)
def prediction_proba(x):
    prediction_proba=classifier.predict_proba(x)[:,1]
    return prediction_proba


def main():
    scaler = MinMaxScaler()

    #markdown
    st.markdown(
        """
        # Please answer the following medical questions :male-doctor::
        """
        ,unsafe_allow_html=True)


    #male
    male = 0
    gender_class = ['Male', 'Female']
    gender = st.radio('Gender',gender_class)
    if gender == 'Male':
        male = 1
    else:
        male = 0
```

```
    #age
    age = 0
    age = st.slider('Age',min_value=0, max_value=100, value=0, step=1)

    #cigsPerDay
    cigsPerDay = st.slider('Daily Cigarette Consumption',min_value=0, max_value=40,
value=0, step=None)

    #BPMeds
    BPMeds = 0
    BPMeds_answers=['Yes', 'No']
    BPMeds_response = st.radio('Have you taken Blood Pressure Medicine
before?',BPMeds_answers)
    if BPMeds_response == 'Yes':
        BPMeds = 1
    else:
        BPMeds = 0

    #prevalentStroke
    prevalentStroke = 0
    prevalentStroke_answers=['Yes', 'No']
    prevalentStroke_response = st.radio('Have you experienced stroke in the
past?',prevalentStroke_answers)
    if  prevalentStroke_response == 'Yes':
        prevalentStroke = 1
    else:
        prevalentStroke = 0


    #prevalentHyp
    prevalentHyp = 0
    prevalentHyp_answers=['Yes', 'No']
    prevalentHyp_response = st.radio('Do you have any hypertensive heart disease, such as
high blood pressure?',prevalentHyp_answers)
    if prevalentHyp_response == 'Yes':
        prevalentHyp = 1
    else:
        prevalentHyp = 0



    #diabetes
    diabetes=0
    diabetes_answers=['Yes', 'No']
    diabetes_response = st.radio('Do you have diabetes?',diabetes_answers)
    if diabetes_response == 'Yes':
        diabetes = 1
    else:
        diabetes = 0

    #totChol
    totChol = st.slider('What is your total cholesterol level? \
```

```
                        (according to  Johns Hopkins Medicine, \
                            ranges for total cholesterol in adults: \
                            Normal: Less than 200 mg/dL. \
                            Borderline high: 200 to 239 mg/dL.\
                            High: At or above 240 mg/dL)',min_value=50, max_value=400,
value=0, step=10)


    #sysBP
    sysBP = st.slider('What is your systolic blood pressure level? \
                        (according to American Heart Association (AHA), \
                        Normal: less than 120 mmHg, Elevated: 120-129 mmHg, \
                        Stage 1 hypertension: 130-139 mmHg, \
                        Stage 2 hypertension: 140 mmHg or higher)',min_value=50,
max_value=200, value=0, step=10)



    #diaBP
    diaBP = 0
    diaBP = st.slider('what is the range for diastolic blood pressure? \
                        (according to American Heart Association (AHA), \
                        Normal: less than 80 mmHg, Elevated: 80-89 mmHg,\
                        Stage 1 hypertension: 90-99 mmHg, \
                        Stage 2 hypertension: 100 mmHg or higher)',min_value=50,
max_value=140, value=0, step=10)


    #BMI
    BMI = 0
    BMI = st.slider('What is your Body Mass Index(BMI)? \
                    (according to World Health Organizatio - WHO,\
                        Underweight: less than 18.5, Normal weight: 18.5-24.9, \
                        Overweight: 25-29.9, Obesity class I: 30-34.9, Obesity class II:
35-39.9,\
                        Obesity class III: 40 or higher)',min_value=10, max_value=50,
value=0, step=1)


    #heartRate
    heartRate = 0
    heartRate = st.slider('What is your heart rate per minute (bpm)? \
                            (according to Bristish Heart Foundation, \
                            a normal resting heart rate should be between 60 to 100 beats
per minute)'
                            ,min_value=40, max_value=150, value=0, step=None)



    #input one dimensional array
    X_test_continuous= {'age' : age, 'cigsPerDay': cigsPerDay,'totChol':totChol,'sysBP':
sysBP, 'diaBP': diaBP, 'BMI': BMI,'heartRate': heartRate}
```

```
    df_test = pd.DataFrame(X_test_continuous, index=['test'])

    #scale all
    #fit
    df = pd.read_csv('data/framingham.csv')
    scaler = scaler.fit(df[['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI',
'heartRate']])
    #scale
    scaled_value = scaler.transform(df_test)

    # define variable
    list_= list()
    label=""
    proba=""

    # get scaled values
    age_sc, cigsPerDay_sc, totChol_sc, sysBP_sc, diaBP_sc, BMI_sc, heartRate_sc =
scaled_value.flatten()
    age, cigsPerDay, totChol, sysBP, diaBP, BMI, heartRate =  age_sc, cigsPerDay_sc,
totChol_sc, sysBP_sc, diaBP_sc, BMI_sc, heartRate_sc
    list_= [male, age, cigsPerDay, BPMeds, prevalentStroke, prevalentHyp, diabetes,
totChol, sysBP, diaBP, BMI, heartRate]

    #page layout (the table and the radar chart)
    col1, col2 = st.columns([0.65,1])

    #variable
    list_conti = np.array([age, cigsPerDay, totChol, sysBP, diaBP, BMI, heartRate])

    #if user click "predict"
    st.subheader('Result: ')
    st.write('Please click \'Predict\' button after you finish inputing your information')
    if st.button("Predict"):
        label = prediction_label(np.array(list_).reshape(1,-1))
        proba = np.round(prediction_proba(np.array(list_).reshape(1,-1)) * 100, 2)
        #st.success('The result is {}'.format(label))
        st.warning(f'Your probability of having CHD in the next 10 years is {proba}%')

    #last section
    st.divider()


if __name__ == '__main__':
    main()
```

6.5 OUTPUT

The outputs obtained after executing the source code for the Machine Learning Model are:

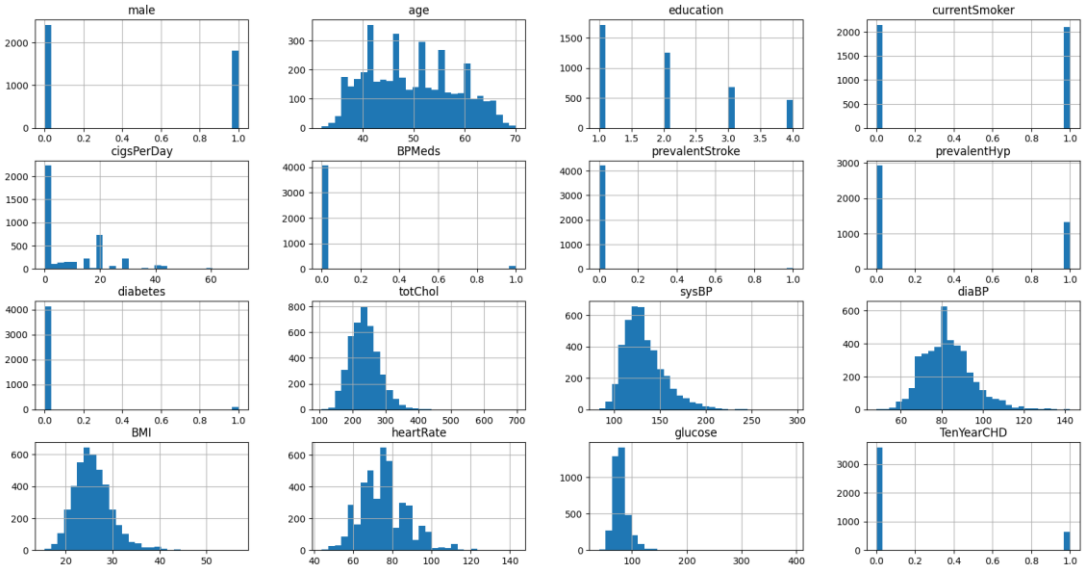


Figure 6: Plotting Data from all Columns

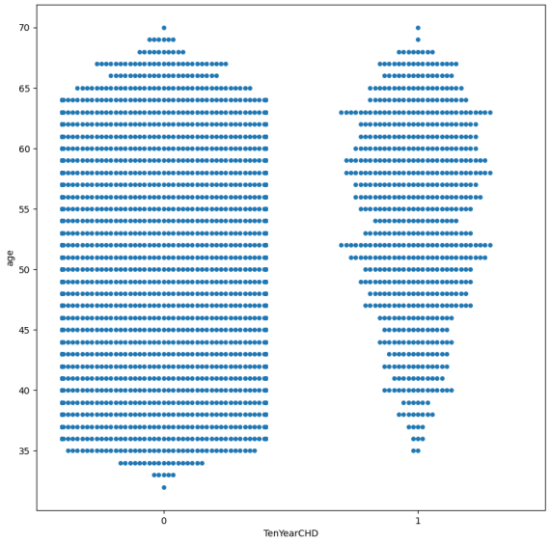


Figure 7: Data Visualization - Age Vs Heart Disease

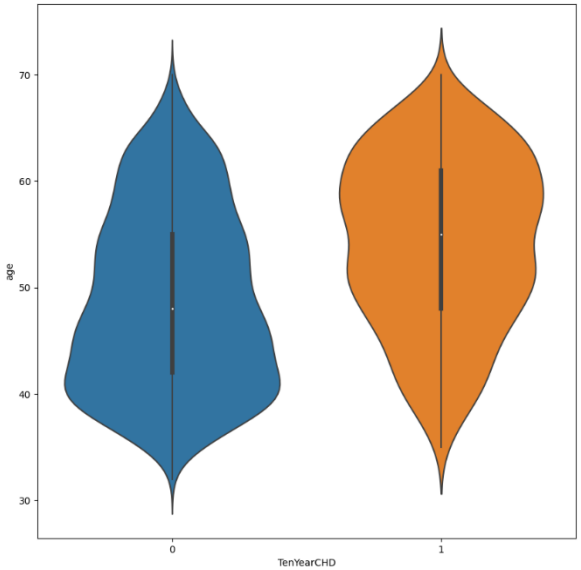


Figure 8: Data Visualization - Age Vs Heart Disease

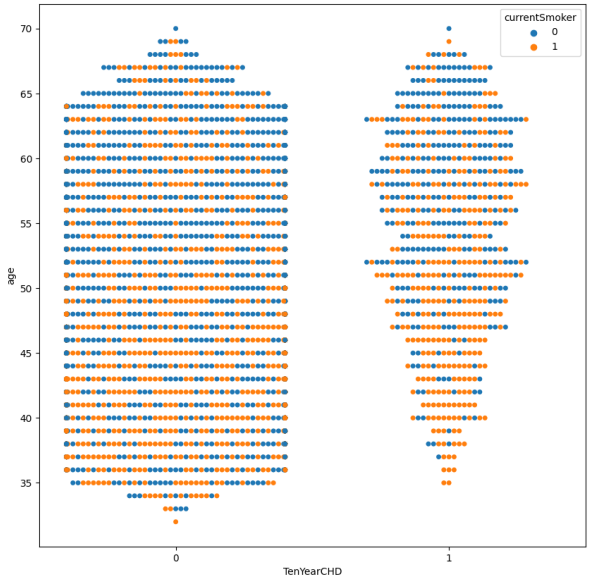


Figure 9: Data Visualization - Age Vs Heart Disease for Smokers and Non-smokers

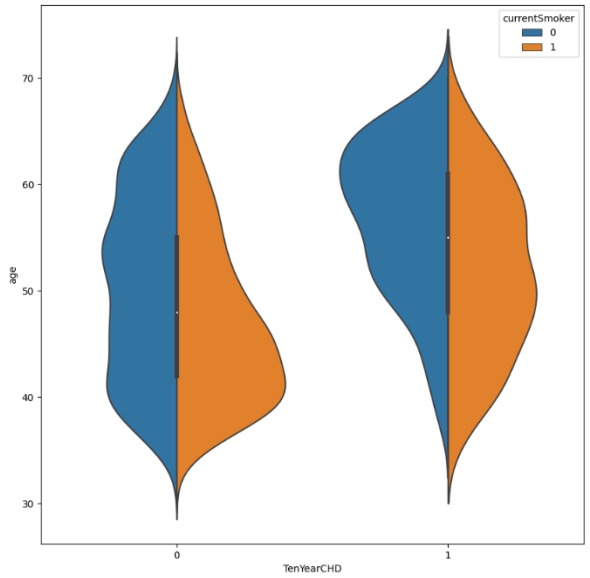


Figure 10: Data Visualization - Age vs Heart Disease for Smokers and Non-smokers

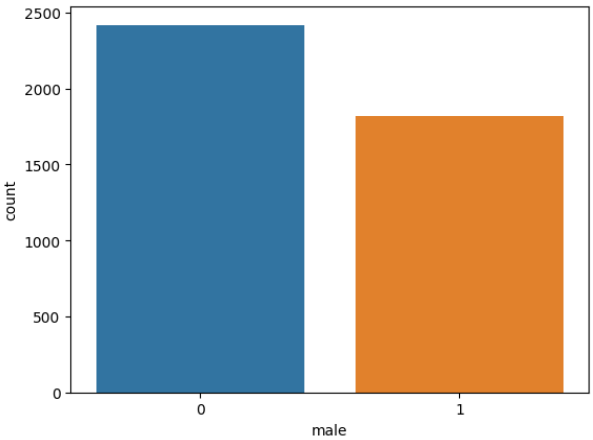


Figure 11: Data Visualization - Gender Count

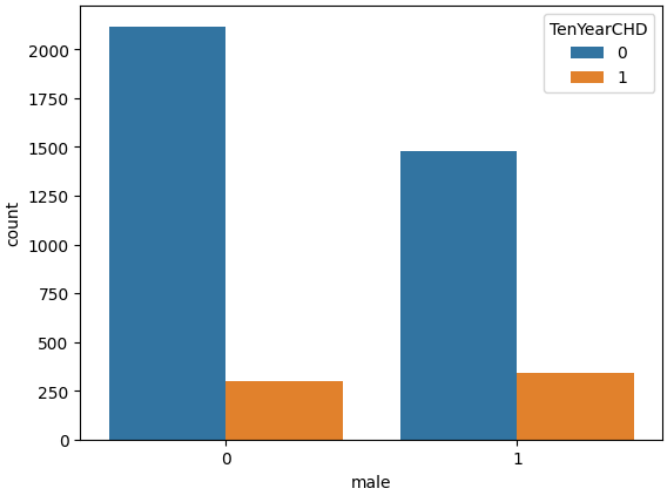


Figure 12: Data Visualization - Gender Vs Heart Disease

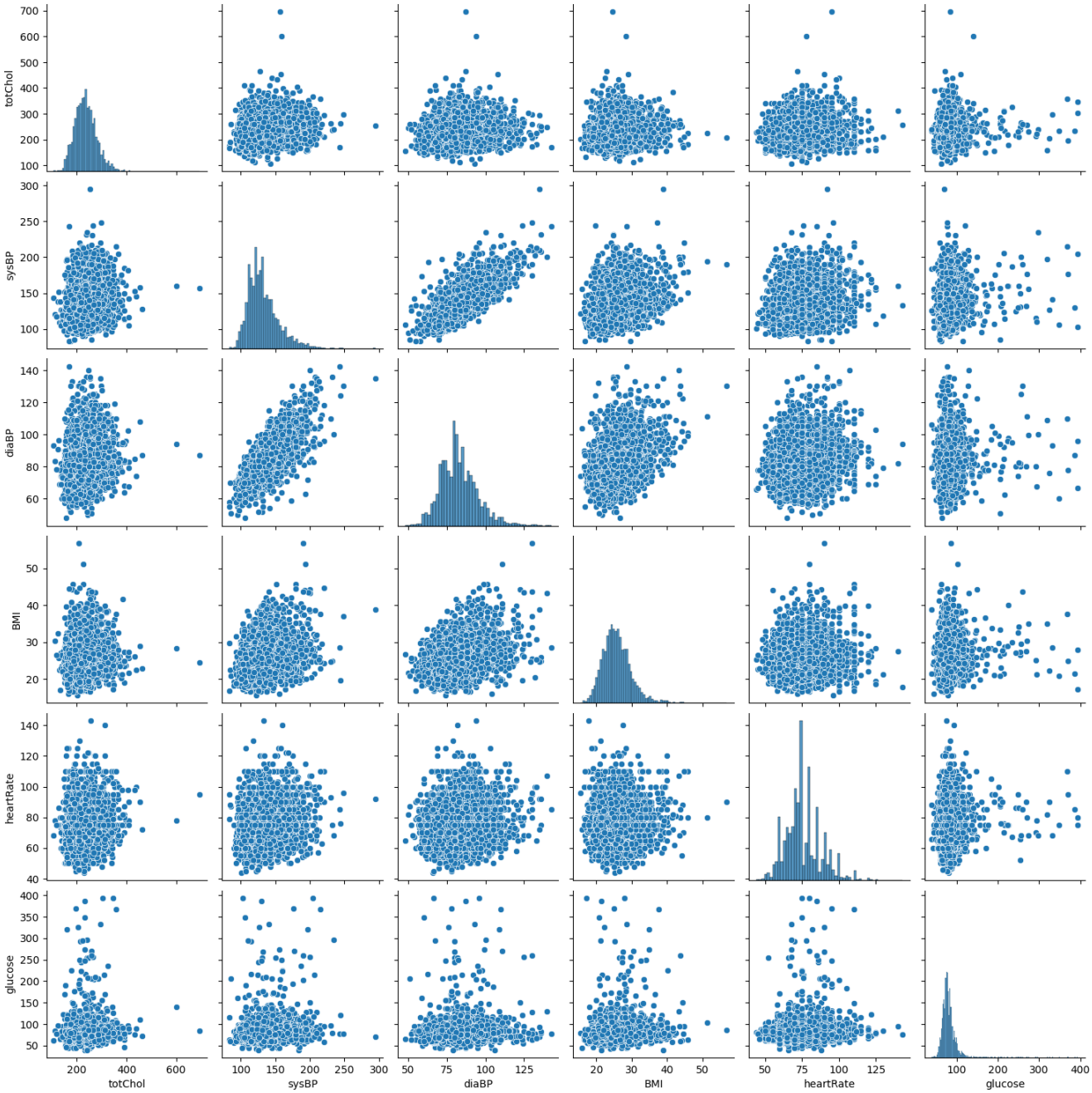


Figure 13: Data Visualization - Understanding Correlation – Pairplot

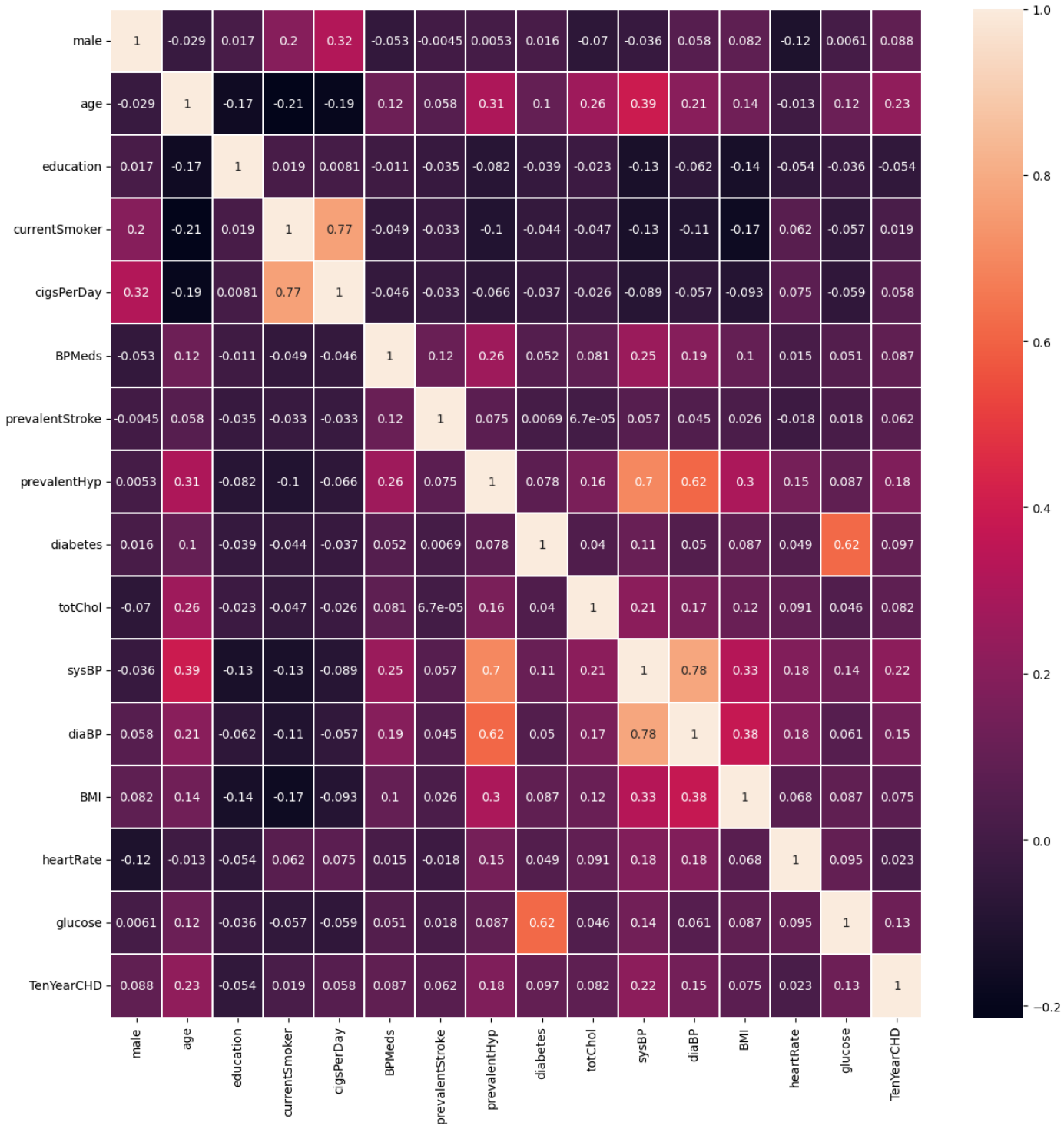


Figure 14: Data Visualization - Understanding Correlation - Heatmap

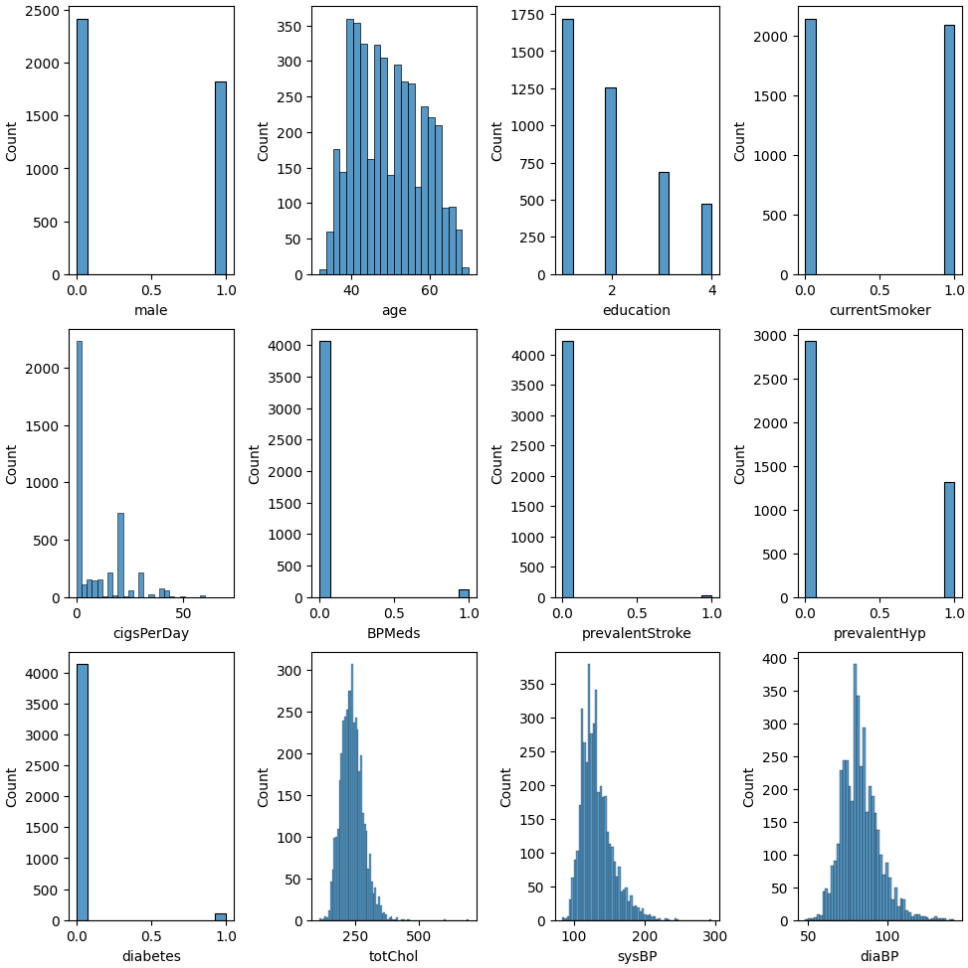


Figure 15: Data Cleaning and Preparation - Normalization Checking

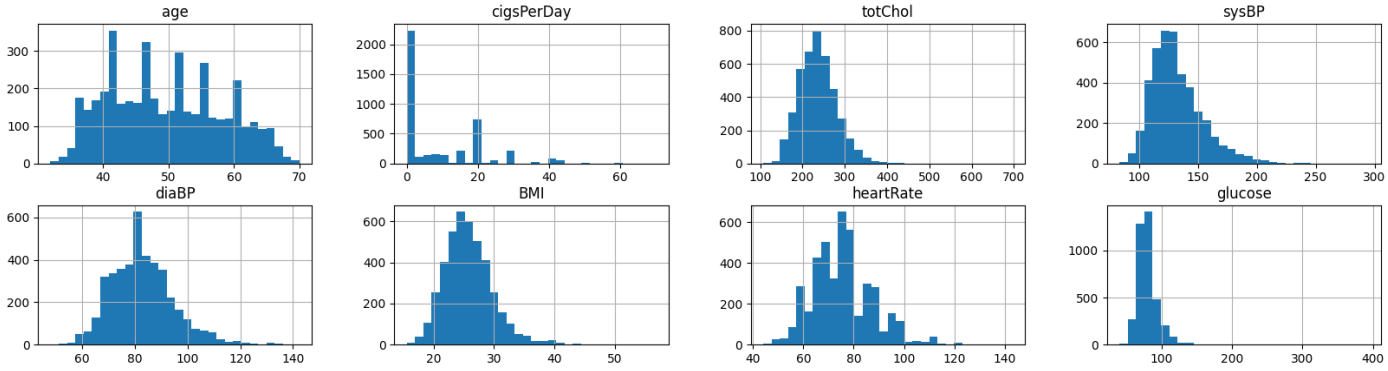


Figure 16: Data Cleaning and Preparation - DataFrame before Scaling

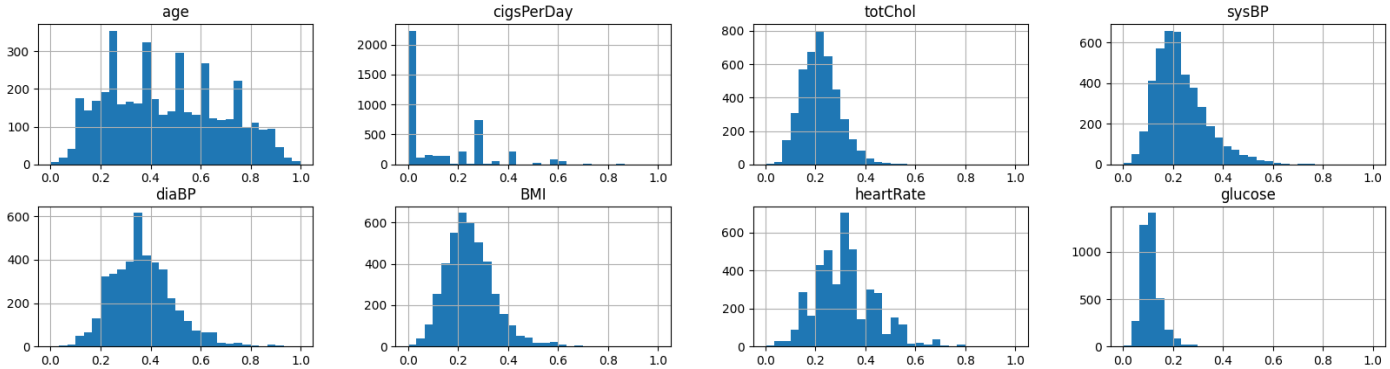


Figure 17: Data Cleaning and Preparation - DataFrame after Scaling

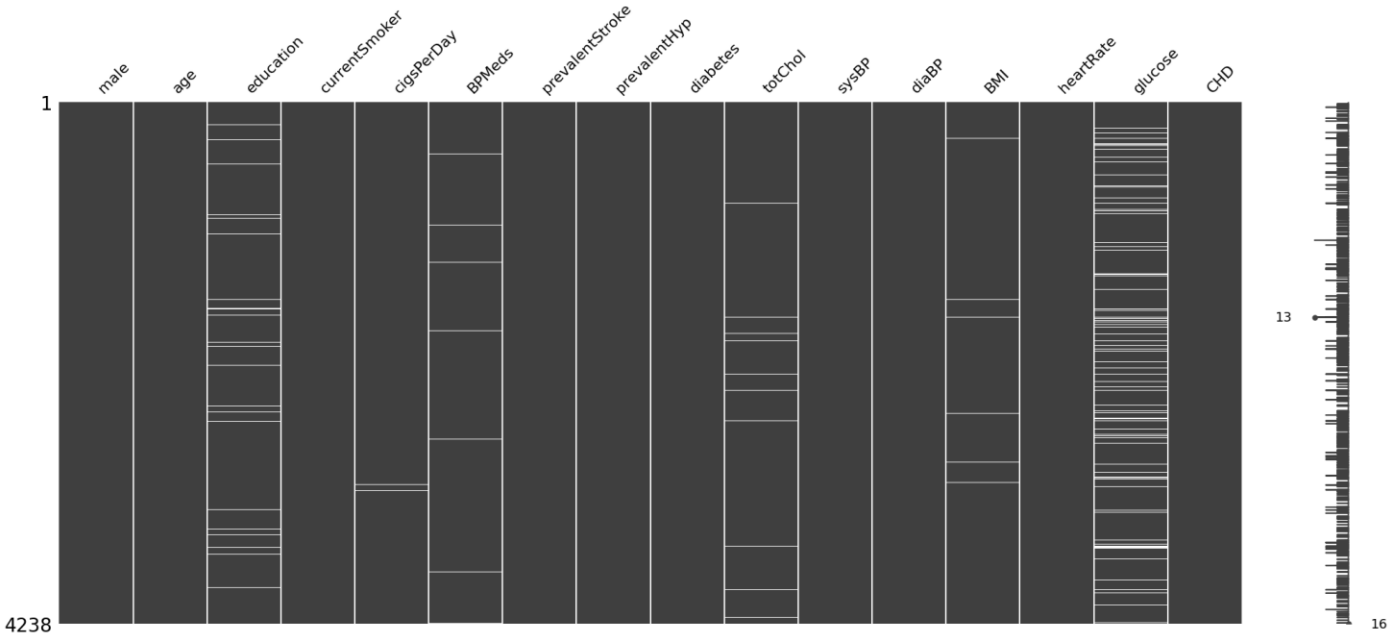


Figure 18: Data Cleaning and Preparation - Handling Missing Data – *Visualizing* Pattern in Missing Values using Matrix

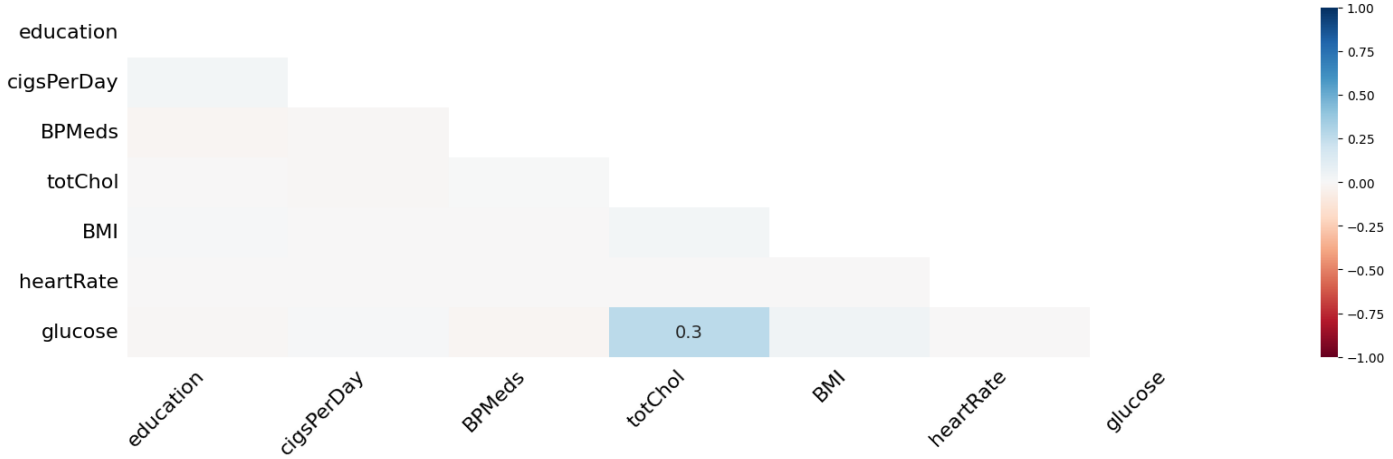


Figure 19: Data Cleaning and Preparation - Handling Missing Data - Checking Correlation between missing values

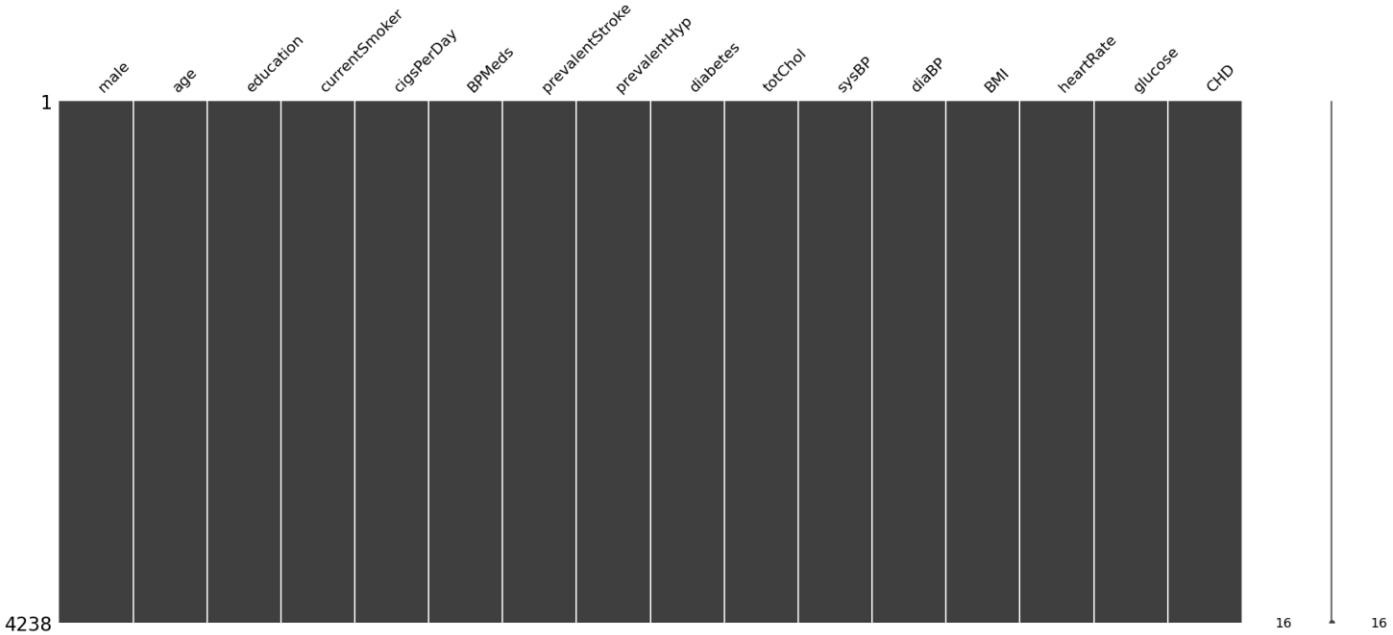


Figure 20: Data Cleaning and Preparation - Handling Missing Data - Visualizing and Checking for missing value patterns after Imputation

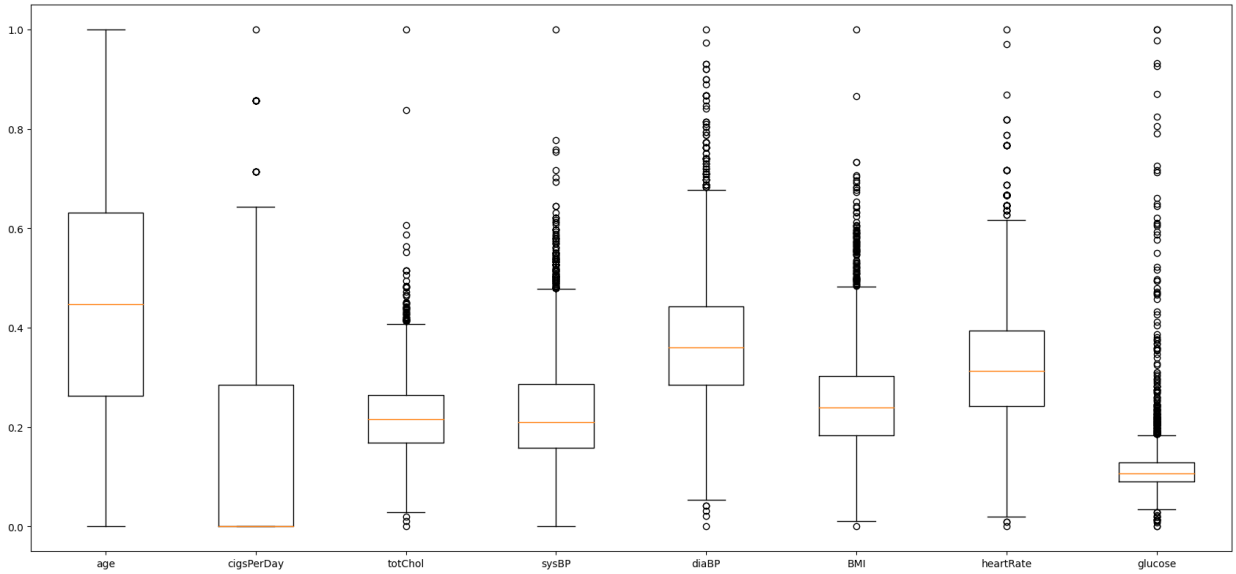


Figure 21: Data Cleaning and Preparation - Handling Outliers - Visualizing Outliers in the Data

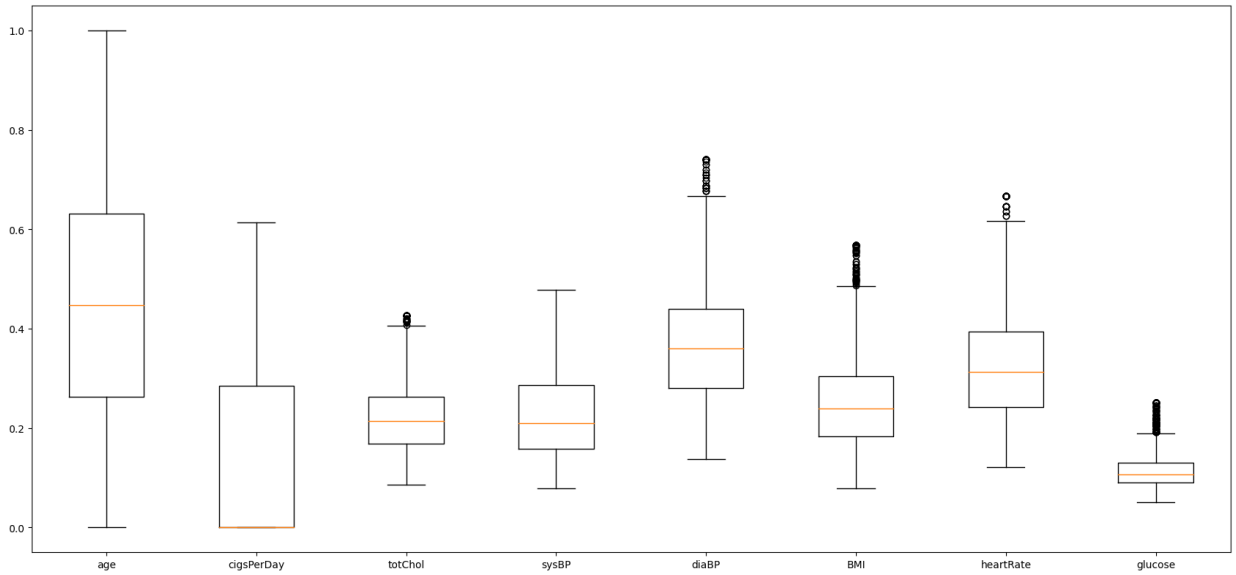


Figure 22: Data Cleaning and Preparation - Handling Outliers - Visualizing Outliers after Winsorization

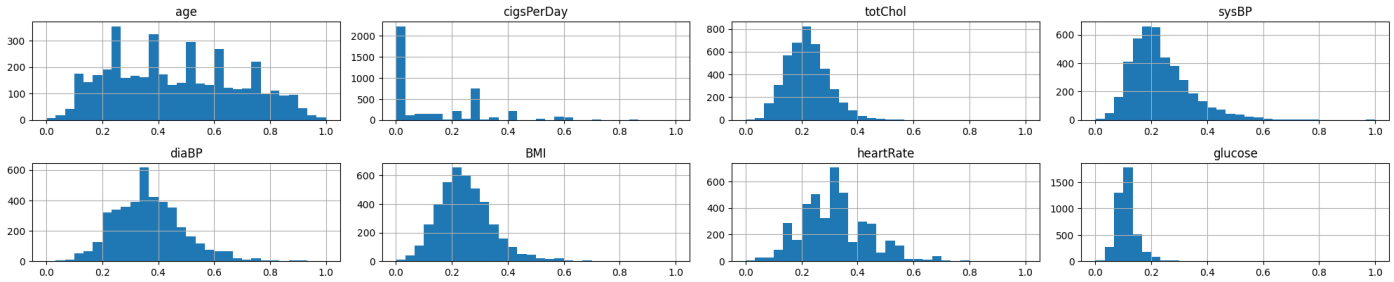


Figure 23: Data Cleaning and Preparation - Handling Outliers - Outlier Distribution before Winsorization

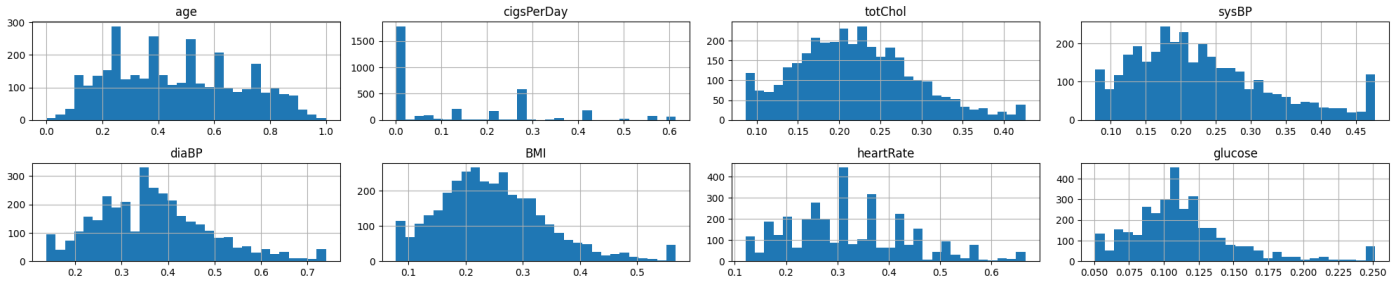


Figure 24: Data Cleaning and Preparation - Handling Outliers - Outlier Distribution after Winsorization

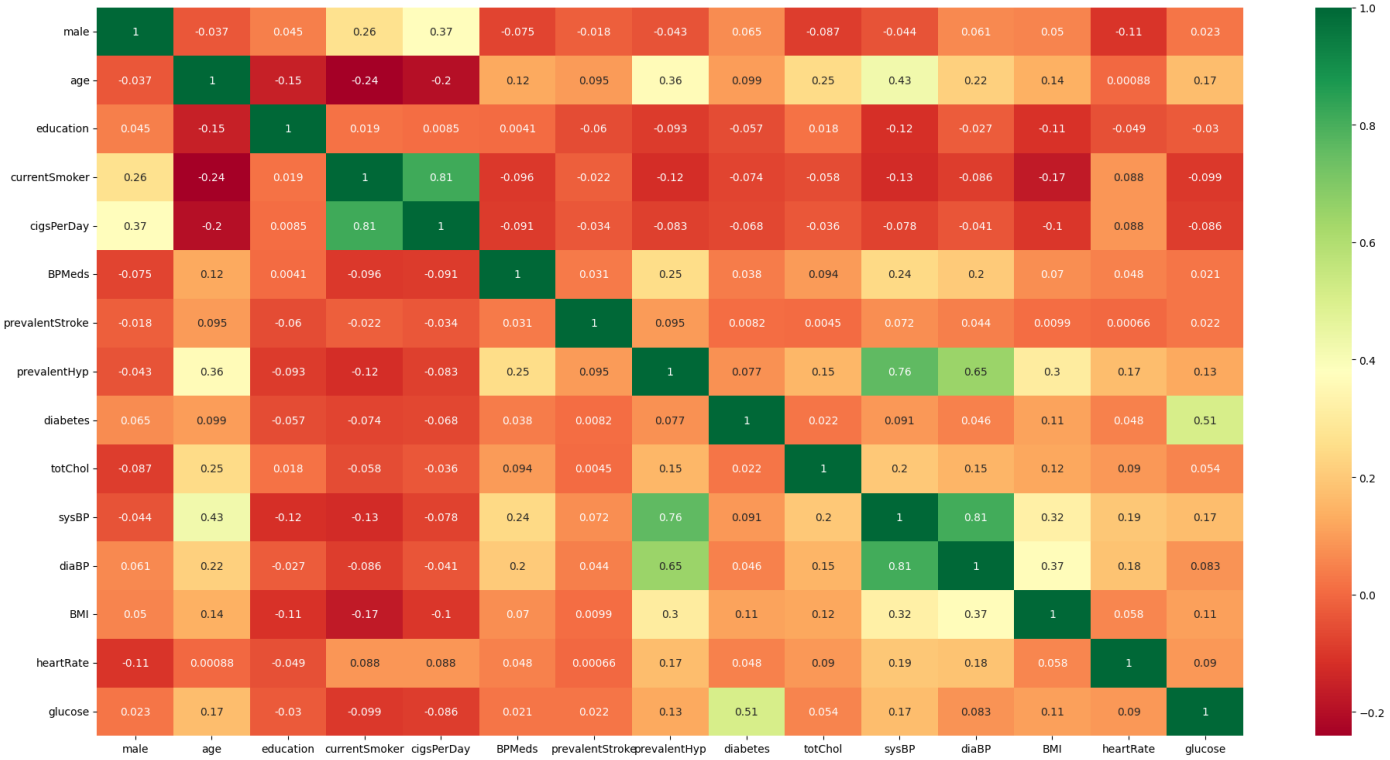


Figure 25: Feature Selection - Visualizing Correlation between all Features

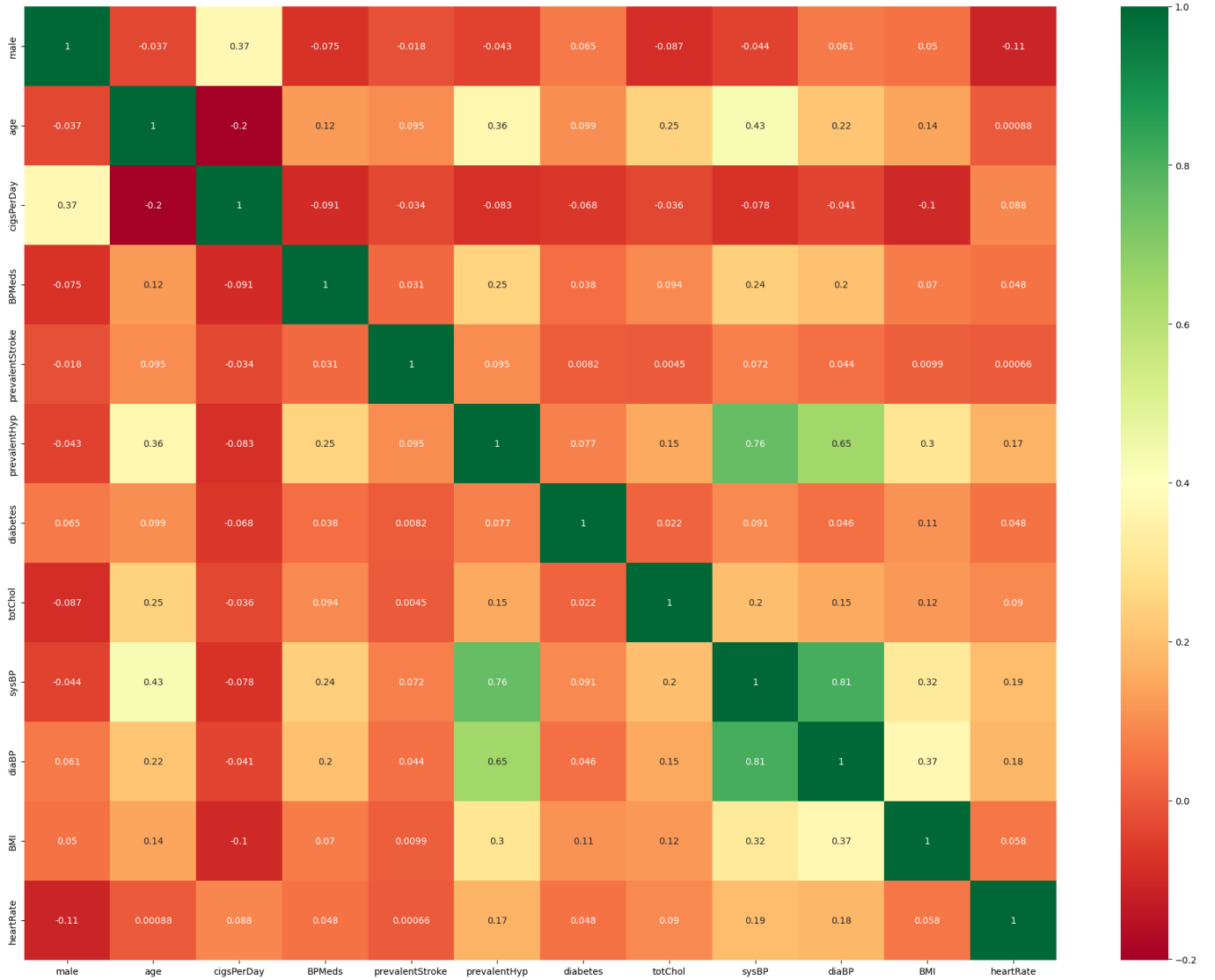


Figure 26: Feature Selection - Visualizing Top Features

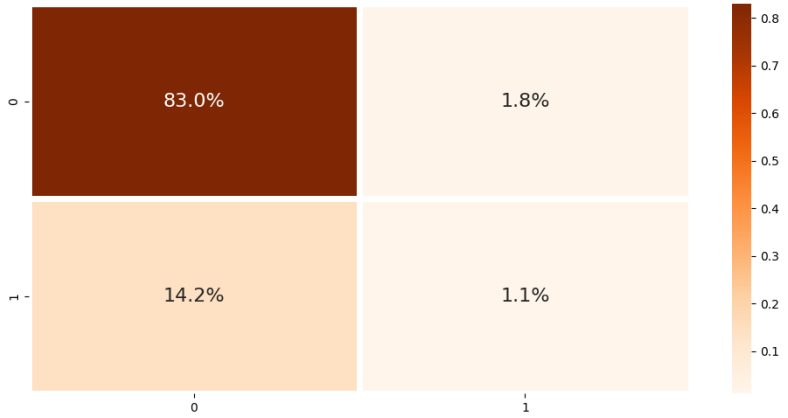


Figure 27: Model Fitting - LR - Normal Parameter Training - Confusion Matrix

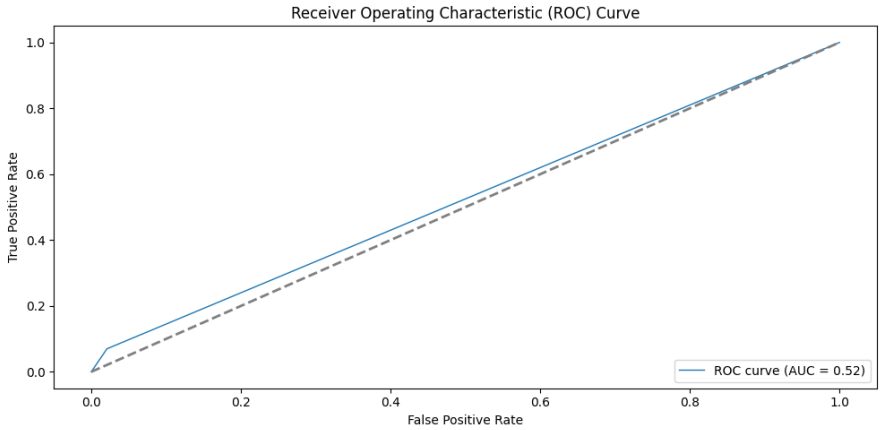


Figure 28: Model Fitting - LR – Normal Parameter Training - ROC Curve

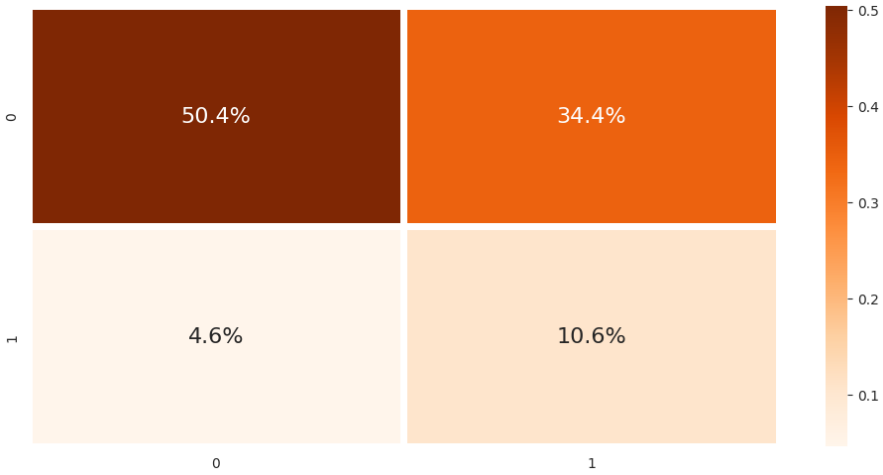


Figure 29: Model Fitting - LR - Tuned Parameter Training - Confusion Matrix

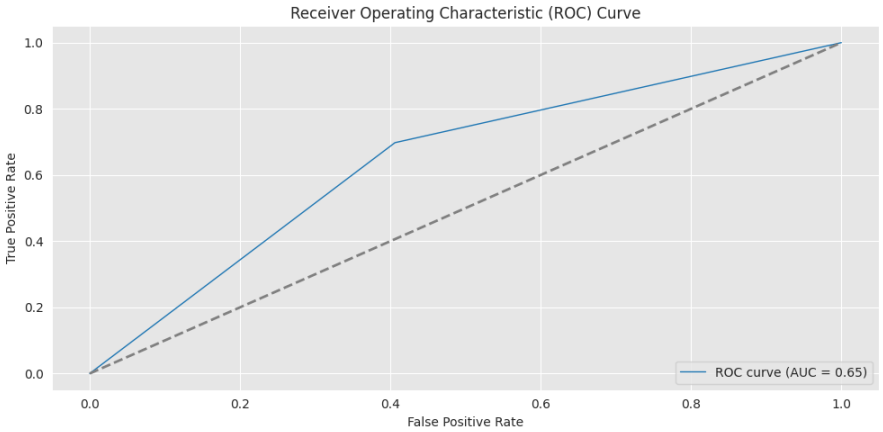


Figure 30: Model Fitting - LR - Tuned Parameter Training - ROC Curve

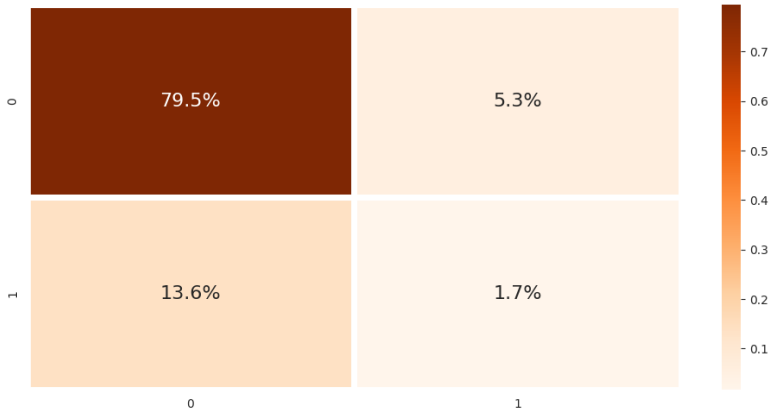


Figure 31: Model Fitting - DT - Normal Parameter Training - Confusion Matrix

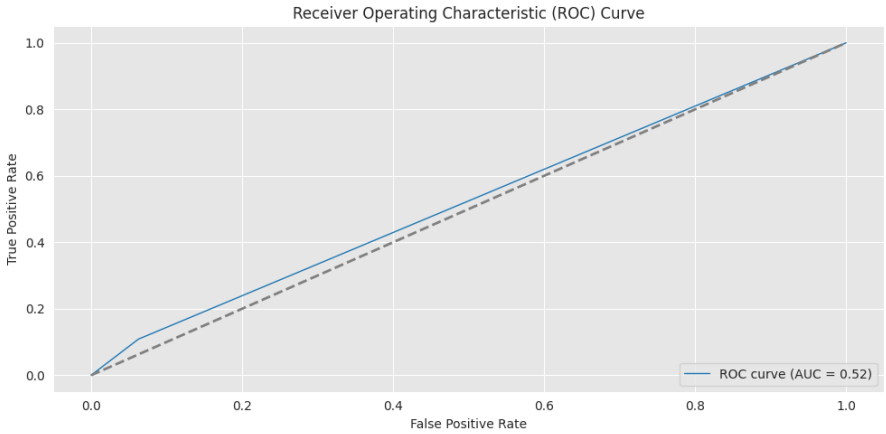


Figure 32: Model Fitting - DT - Normal Parameter Training - ROC Curve

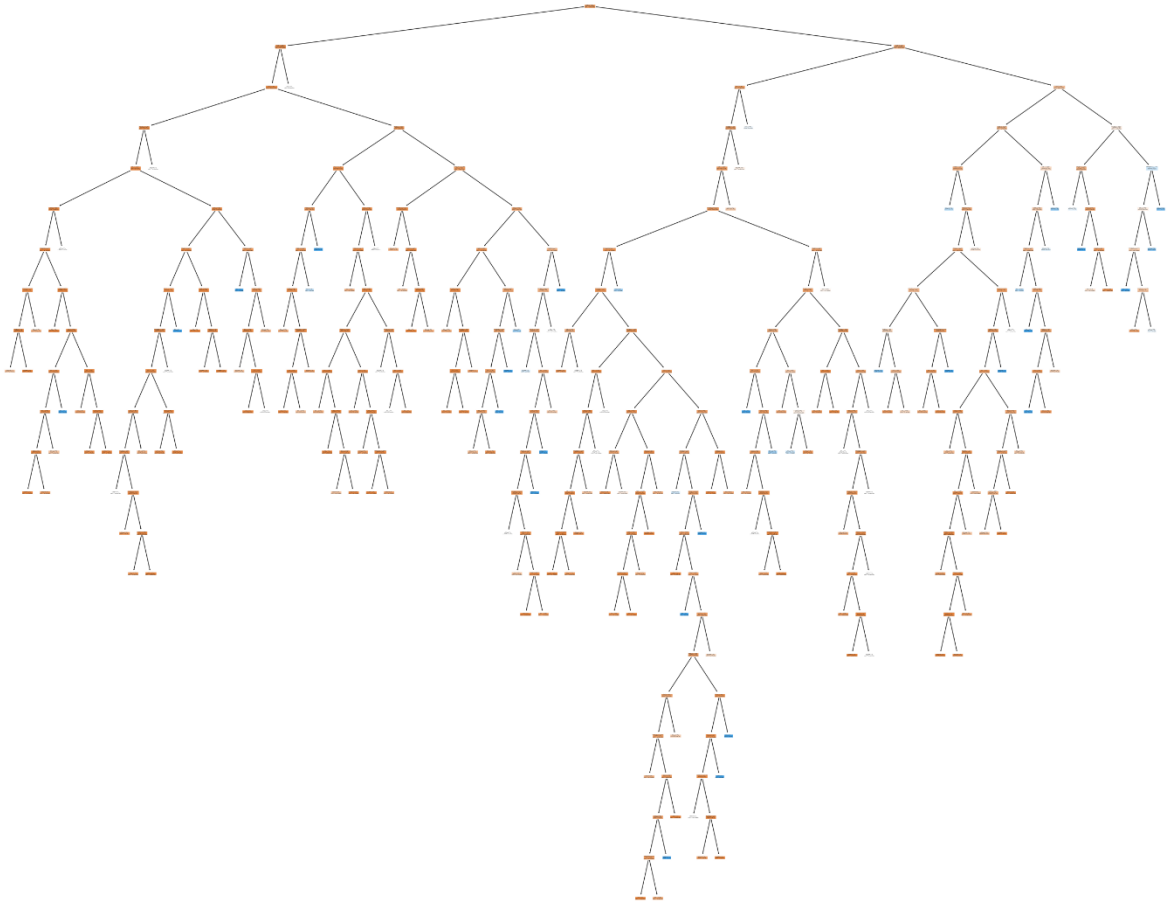


Figure 33: Model Fitting - DT - Obtaining Visual Decision Tree

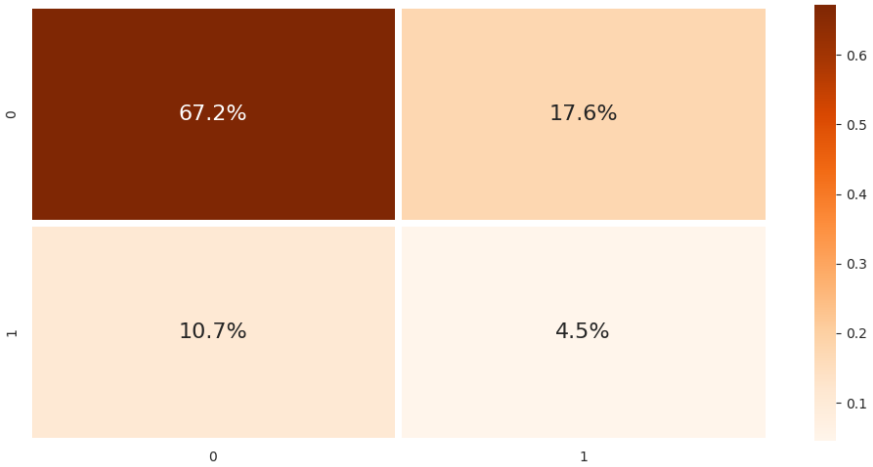


Figure 34: Model Fitting - DT - Tuned Parameter Training - Confusion Matrix

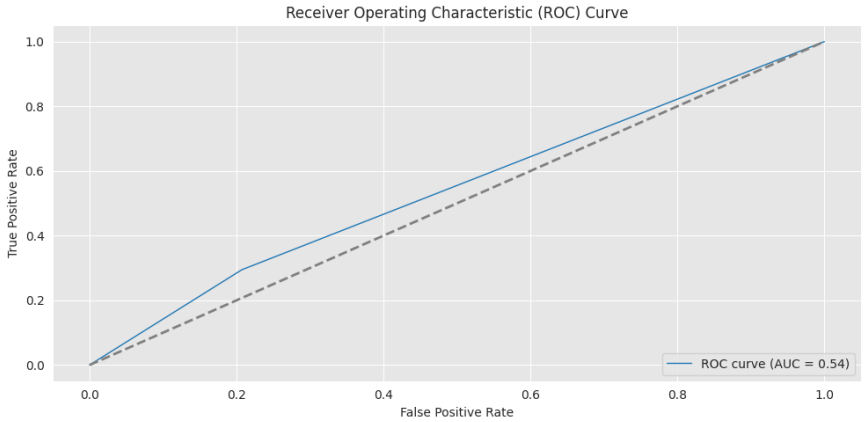


Figure 35: Model Fitting - DT - Tuned Parameter Training - ROC Curve

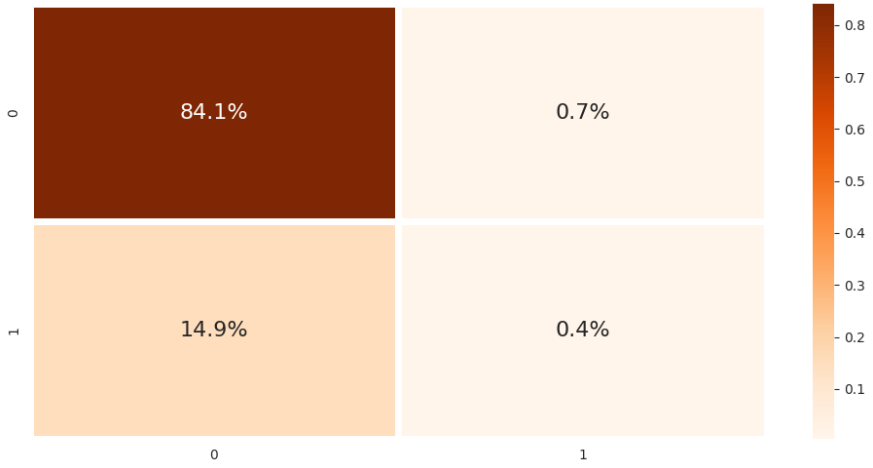


Figure 36: Model Fitting - RF - Normal Parameter Training - Confusion Matrix

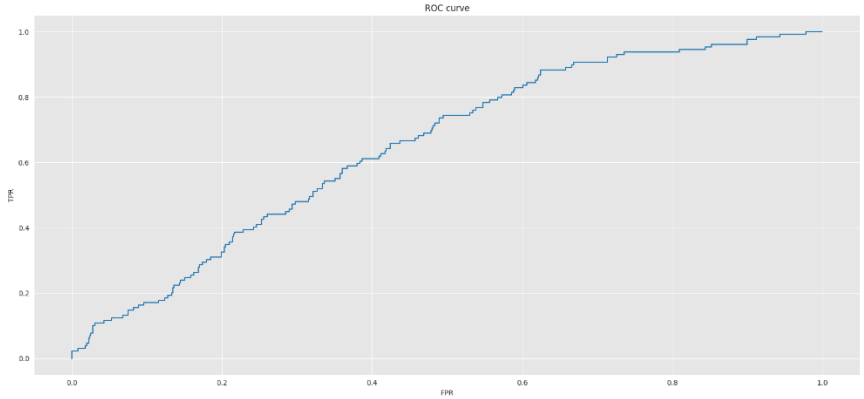


Figure 37: Model Fitting - RF - Normal Parameter Training - ROC Curve

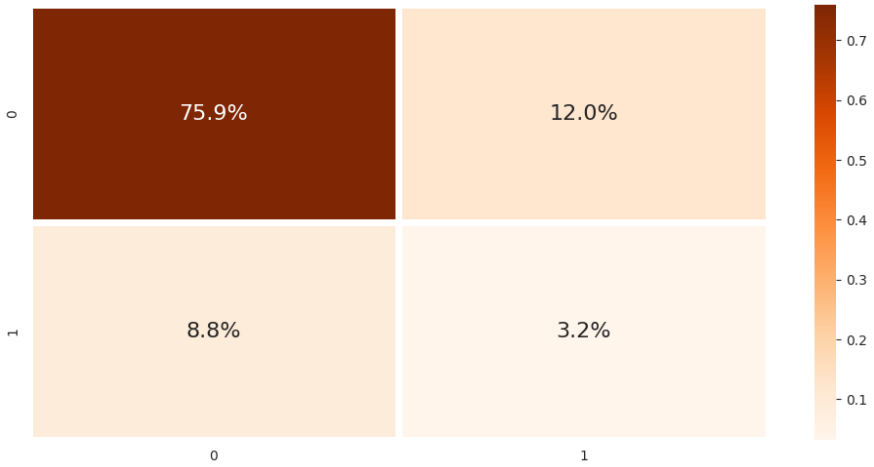


Figure 38: Model Fitting - RF - Tuned Parameter Training - Confusion Matrix

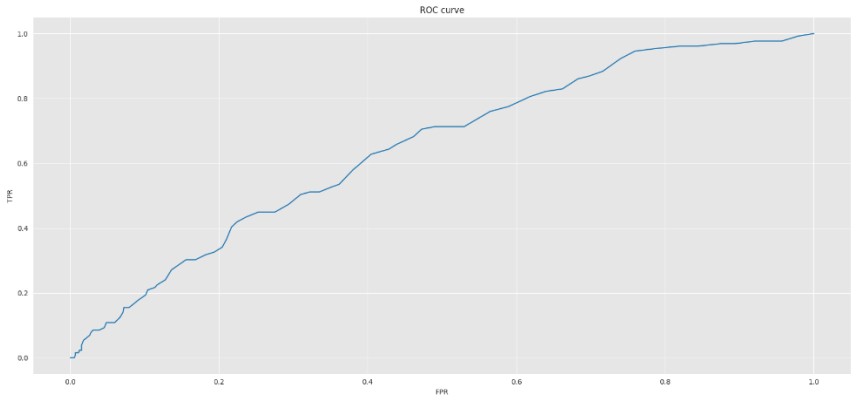


Figure 39: Model Fitting - RF - Tuned Parameter Training - ROC Curve

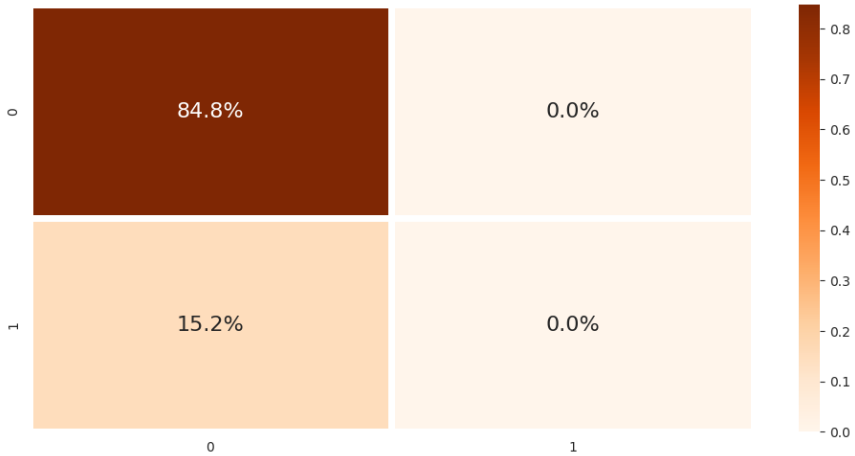


Figure 40: Model Fitting - SVC - Normal Parameter Training - Confusion Matrix

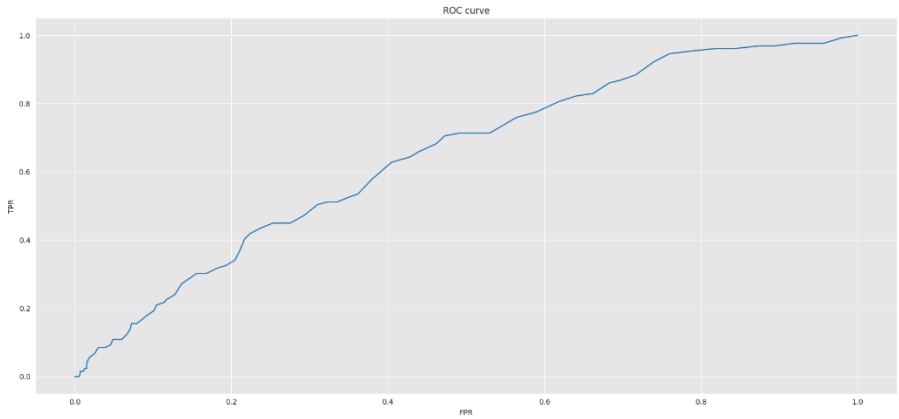


Figure 41: Model Fitting - SVC - Normal Parameter Training - ROC Curve

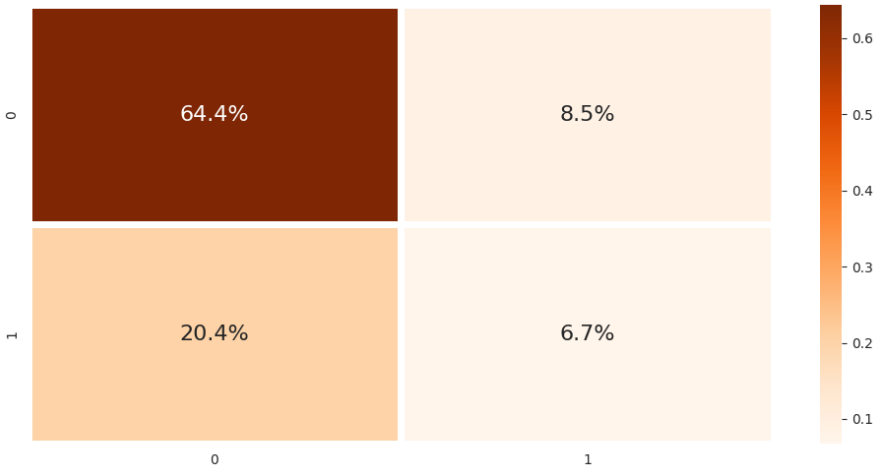


Figure 42: Model Fitting - SVC - Tuned Parameter Training - Confusion Matrix

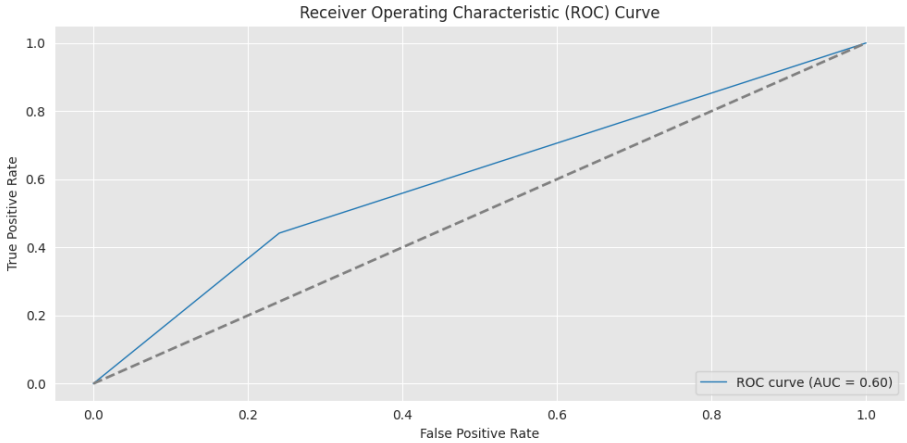


Figure 43: Model Fitting - SVC - Tuned Parameter Training - ROC Curve

The outputs obtained after executing the source code for the User Interface are:

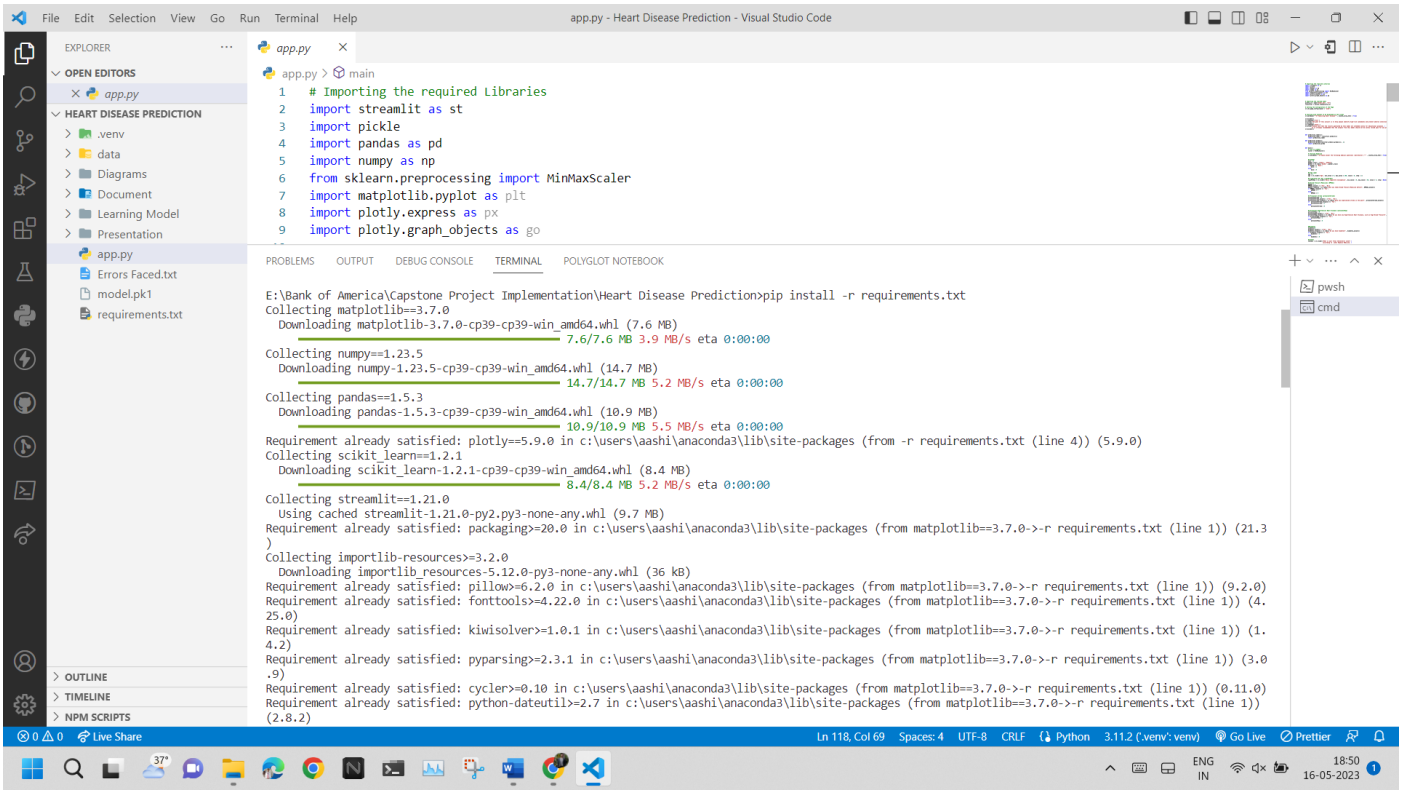


Figure 44: Installing the required Libraries in Local Environment

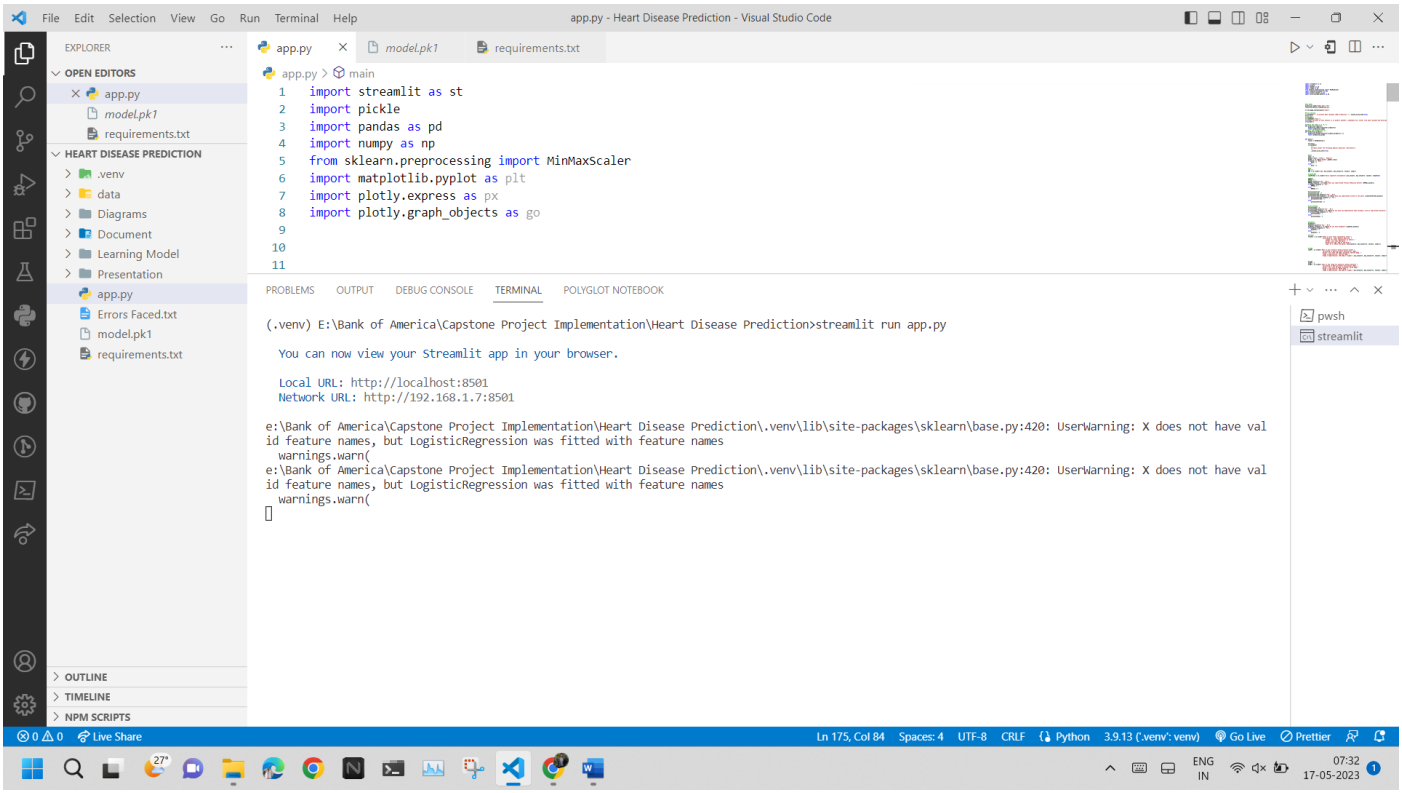


Figure 45: Executing the Interface Code

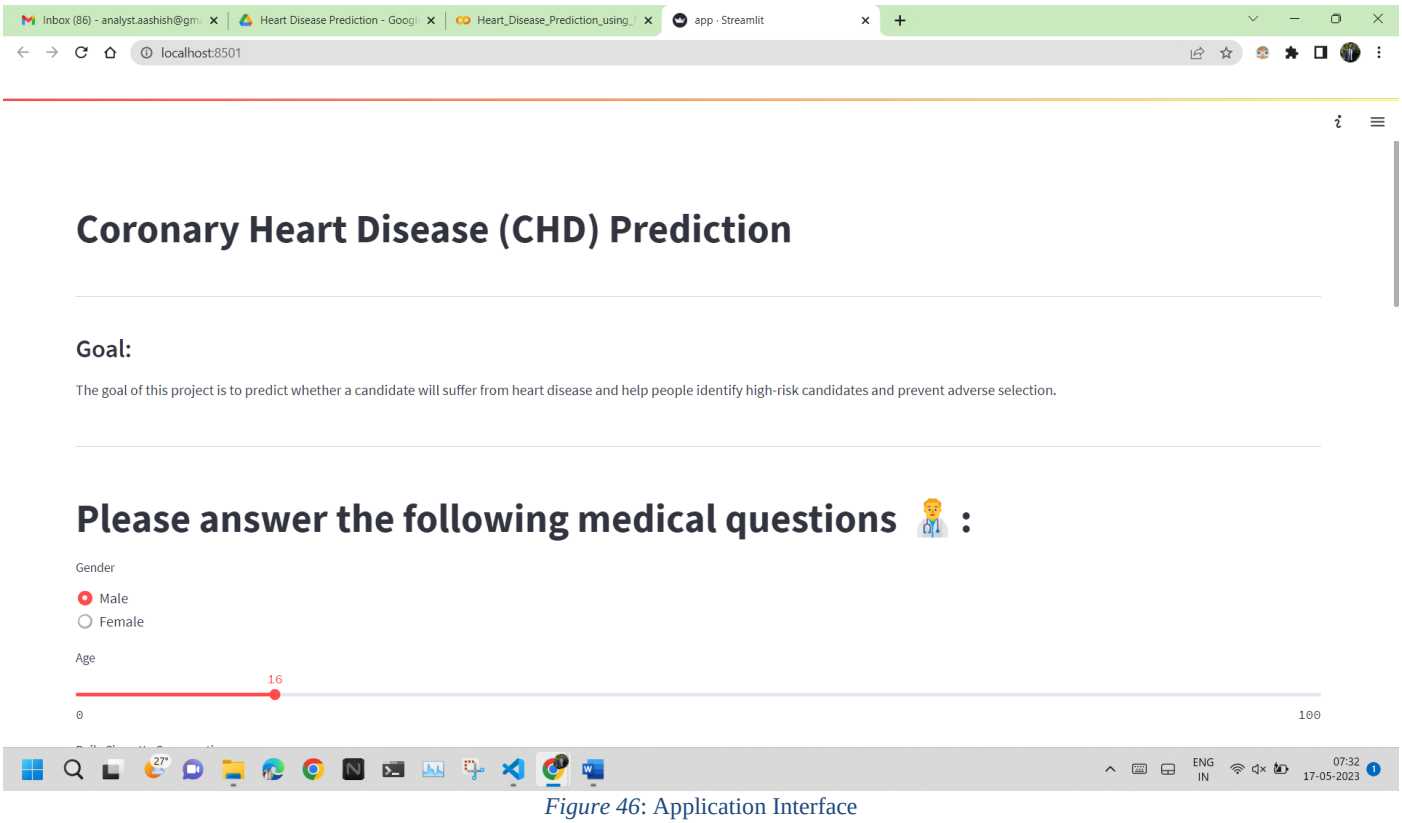


Figure 46: Application Interface

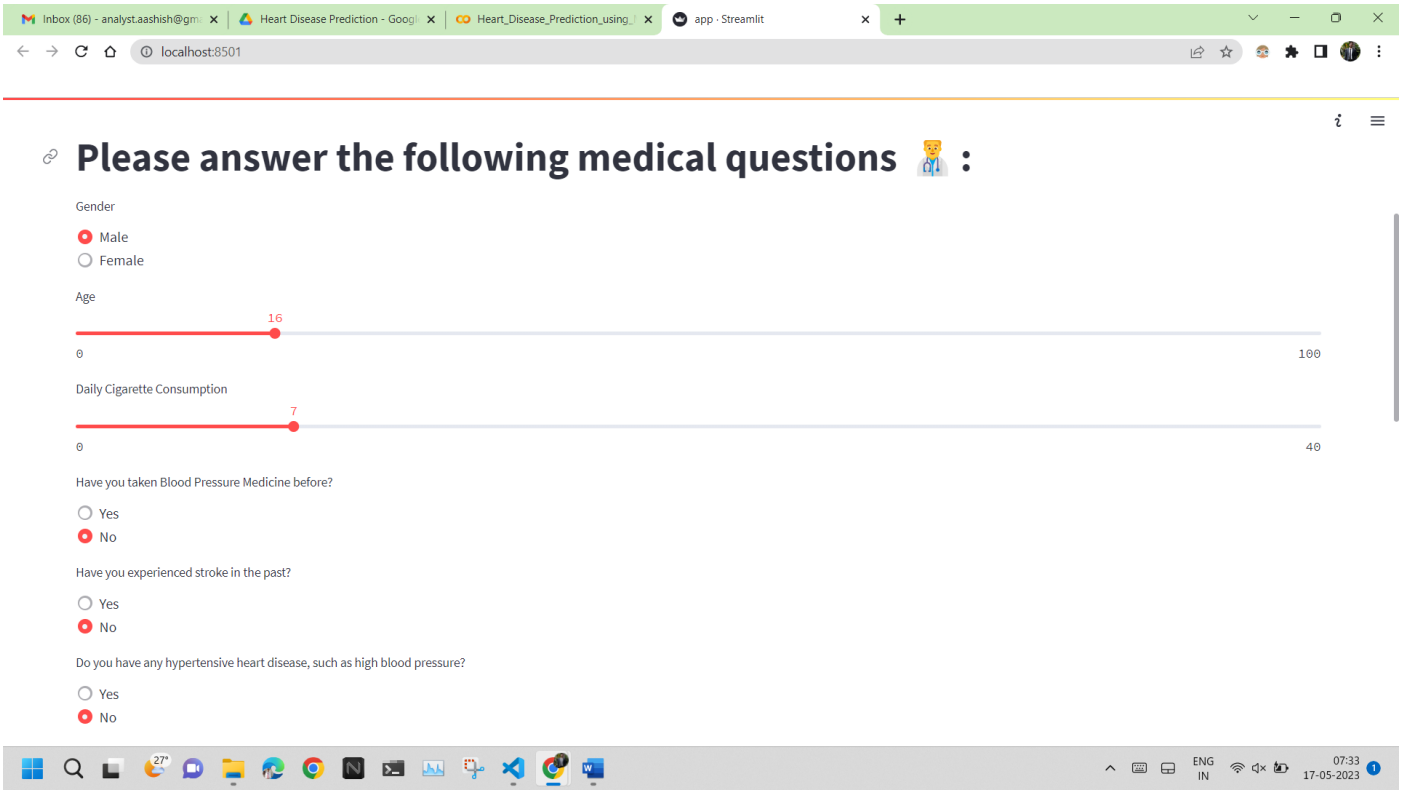


Figure 47: Providing Input (Part 1)

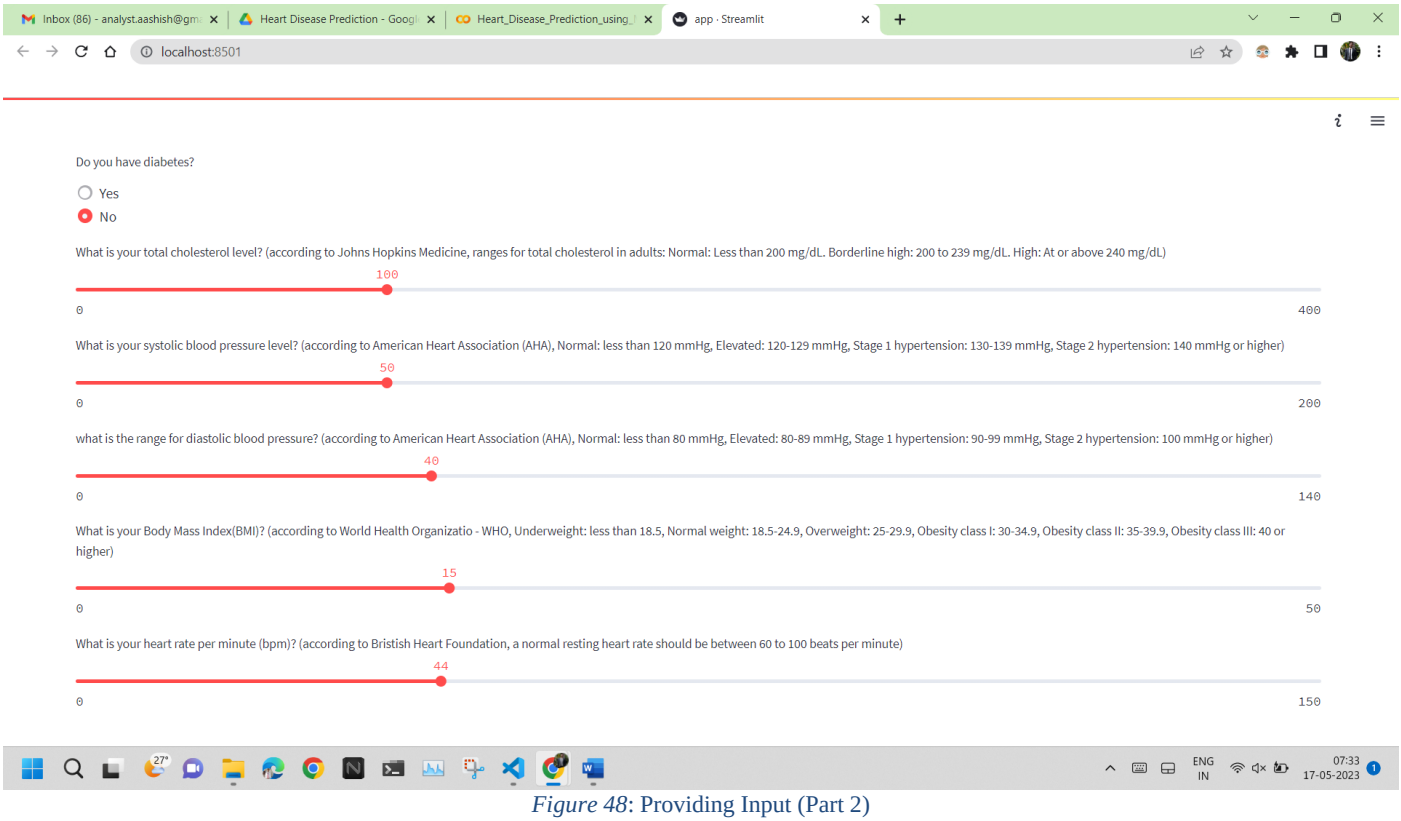


Figure 48: Providing Input (Part 2)

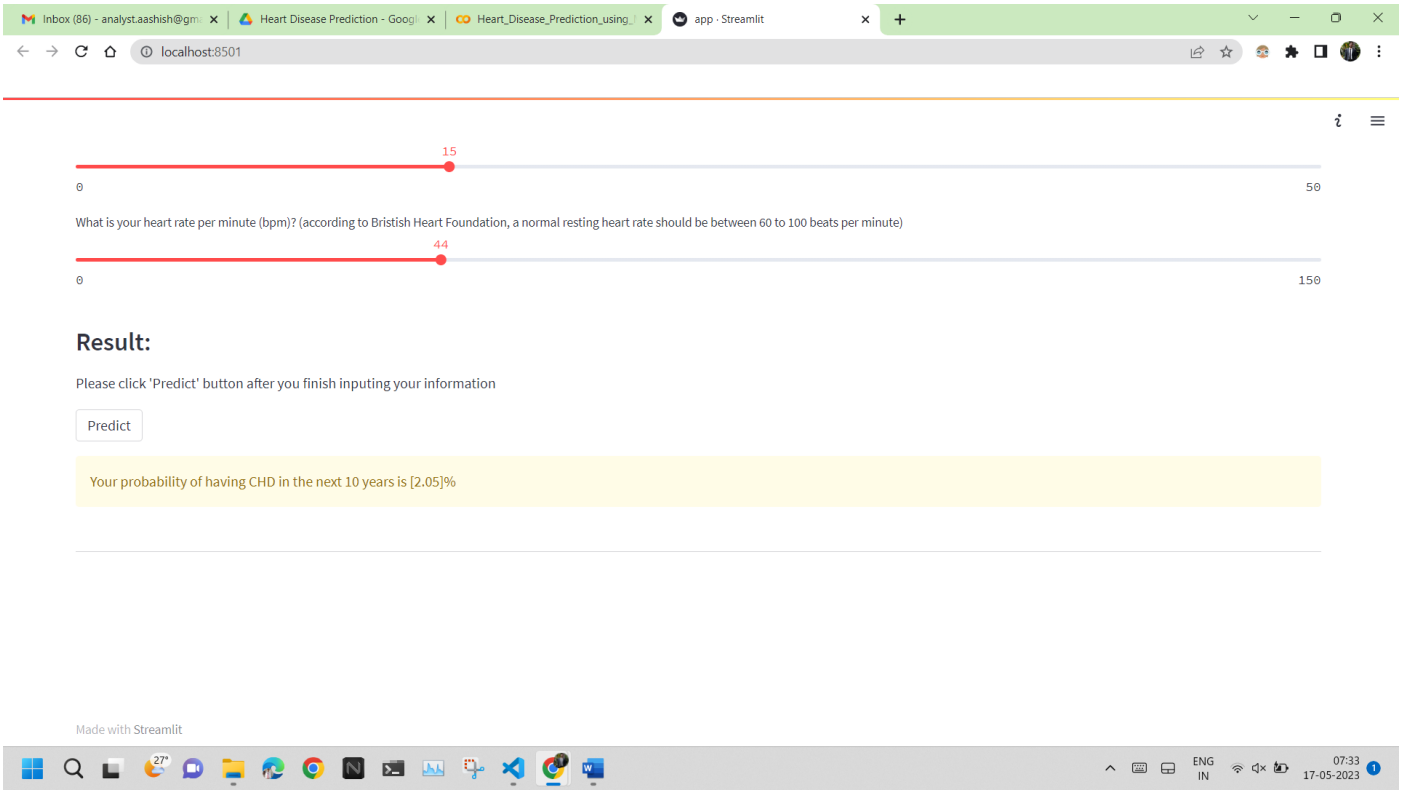


Figure 49: Providing Input (Part 3) and Obtaining Prediction

Chapter 7

Results

7.1 RESULTS AND OUTCOMES

From the Machine Learning Models, we obtain the following results for each of the Machine Learning Algorithm selected:

Table 2: Training and Testing Results

	Untuned							
	Accuracy		Precision		Recall		F1-Score	
Algorithm	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Logistic Regression	0.8542	0.8408	0.6615	0.3750	0.0834	0.0697	0.1482	0.1176
Decision Tree	0.8746	0.8113	0.6642	0.2372	0.3533	0.1085	0.4613	0.1489
Random Forest Classifier	0.9165	0.8443	0.4504	0.0232	1.0000	0.3333	0.6211	0.0434
SVC	0.8480	0.8478	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Tuned							
	Accuracy		Precision		Recall		F1-Score	
Algorithm	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Logistic Regression	0.6182	0.6096	0.2501	0.2356	0.7572	0.6976	0.3760	0.3522
Decision Tree	0.8224	0.7169	0.4389	0.2032	0.6077	0.2945	0.5097	0.2405
Random Forest Classifier	0.9943	0.7912	0.9750	0.2647	0.9883	0.2093	0.9816	0.2337
SVC	0.7699	0.7110	0.3629	0.2478	0.6815	0.4418	0.4736	0.3175

7.2 RESULT ANALYSIS

From the above results, we can see that the Optimization Techniques do not have a positive impact on the implementation. There could be multiple reasons to this situation. It is possible that the metrics identified were wrong or the models underwent overfitting or there were two few hyperparameters. We will need to reanalyze all these and look for a solution if possible.

Conclusions

From the above results, we can see that the Optimization Techniques do not have a positive impact on the implementation. Upon multiple attempts for identifying the metrics, checking for overfitting, the number of hyperparameters, it has been identified that the reason behind a degrade in the performance could be the wrong metric or the number of hyperparameters and the identification of the Parameter Space. For the wrong metric, the underlying assumption is that if the conditions match for a classification, then would work for almost all the cases. Hyperparameter Optimization is used to amplify the evaluation criteria chosen the problem statement but if the considered assumption happens to go wrong, then the results signifying assumptions would also go wrong. The hyperparameters which were chosen were identified from a particular Parameter Space. It is possible that the Hyperparameter Space might be larger than the anticipated space. This gives us a possibility that the values which were picked up were not enough for identification. Since the Optimization Technique had to choose the hyperparameters giving the best performance, it just considered the best option out of all those parameters. It is possible that we can identify that there is a larger space for the identification of the correct set of Hyperparameters and we need to find the best hyperparameters out of these because they would provide us with the most accurate results for the Machine Learning Models but there is a problem in the implementation which is that it requires very high computation power and time which is beyond the capability of my local machine and the cloud platform. This hinders the research further because the machine learning model either ends up crashing or going out of runtime (time limit exceeded). This also teaches us another important lesson that we should trust the default hyperparameters for the machine learning algorithms because those are selected which tend to give a good performance on almost all the cases and could be tuned further because it is not possible that hand-tuning might work every single time.

Future Works

Some of the future works for this project are:

- Identifying and specifying the correct Parameter Space for the Optimization Procedure.
- Implementing the Optimization Processes based on the new Parameter Space.
- Attempting to expand the research to a wider set of Learning Algorithms and implement them.
- Improve the Learning Curve After optimizing the Learning Algorithms.

References

[1]: Katari, S., Likith, T., Sree, M. P. S., & Rachapudi, V. (2023). Heart Disease Prediction using Hybrid ML Algorithms. 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Sustainable Computing and Data Communication Systems (ICSCDS), 2023 International Conference On, 121–125. https://doi-org.egateway.vit.ac.in/10.1109/ICSCDS56580.2023.10104609

[2]: Shaik, M. A., Sreeja, R., Zainab, S., Sowmya, P. S., Akshay, T., & Sindhu, S. (2023). Improving Accuracy of Heart Disease Prediction through Machine Learning Algorithms. 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), Innovative Data Communication Technologies and Application (ICIDCA), 2023 International Conference On, 41–46. https://doi-org.egateway.vit.ac.in/10.1109/ICIDCA56705.2023.10100244

[3]: Sk KB, D R, Priya SS, Dalavi L, Vellela SS, B VR. Coronary Heart Disease Prediction and Classification using Hybrid Machine Learning Algorithms. 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), Innovative Data Communication Technologies and Application (ICIDCA), 2023 International Conference on. March 2023:1-7. doi:10.1109/ICIDCA56705.2023.10099579

[4]: Reddy, S. D., Lohitha, S., & Shaik, F. (2023). Machine Learning based Mobile App for Heart Disease Prediction. 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), Innovative Data Communication Technologies and Application (ICIDCA), 2023 International Conference On, 464–470. https://doi-org.egateway.vit.ac.in/10.1109/ICIDCA56705.2023.10099714

[5]: Bajaj, M., Rawat, P., Bhatt, C., Chauhan, R., & Singh, T. (2023). Heart Disease Prediction using Ensemble ML. 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Sustainable Computing and Data Communication Systems (ICSCDS), 2023 International Conference On, 680–685. https://doi-org.egateway.vit.ac.in/10.1109/ICSCDS56580.2023.10104770

[6]: Anitha, C., Rajkumar, S., & R, D. (2023). An Effective Heart Disease Prediction Method using Extreme Gradient Boosting Algorithm Compared with Convolutional Neural Networks. 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Advanced Computing and Communication Systems (ICACCS), 2023 9th International Conference On, 1, 2224–2228. https://doi-org.egateway.vit.ac.in/10.1109/ICACCS57279.2023.10112952

[7]: Bakar, W. A. W. A., Josdi, N. L. N. B., Man, M. B., & Zuhairi, M. A. B. (2023). A Review: Heart Disease Prediction in Machine Learning & Deep Learning. 2023 19th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), Signal Processing & Its Applications (CSPA), 2023 19th IEEE International Colloquium On, 150–155. https://doi-org.egateway.vit.ac.in/10.1109/CSPA57446.2023.10087837

[8]: Kumar, M. R., A, D. A., Saran, T. M. G., Kumar, R. J. R., Subramanyam, D. V. S. S., & T, M. N. (2023). Machine Learning based Cardiac Disease Prediction- A Comparative Analysis. 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Advanced Computing and Communication Systems (ICACCS), 2023 9th International Conference On, 1, 530–534.

https://doi-org.egateway.vit.ac.in/10.1109/ICACCS57279.2023.10112914

[9]: Mondal, D., & Saini, R. (2023). Heart Disease Prediction: Optimization of Machine Learning Algorithms. 2023 2nd International Conference for Innovation in Technology (INOCON), Innovation in Technology (INOCON), 2023 2nd International Conference For, 1–4. https://doi-org.egateway.vit.ac.in/10.1109/INOCON57975.2023.10101268

[10]: Kumar, K. P., Rohini, V., Yadla, J., & VNRaju, J. (2023). A Comparison of Supervised Learning Algorithms to Prediction Heart Disease. 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), 2023 International Conference On, 1–5. https://doi-org.egateway.vit.ac.in/10.1109/ICECONF57129.2023.10084035

[11]: Swathi, P., & Gunasekaran, M. (2022). A Methodology For Early Prediction and Classification of Heart Diseases in Diabetic Patients With Machine Learning Techniques. 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Mobile Networks and Wireless Communications (ICMNWC), 2022 IEEE 2nd International Conference On, 1–7. https://doi-org.egateway.vit.ac.in/10.1109/ICMNWC56175.2022.10031285

[12]: Darapaneni, N., Rao, S. R., Sagare, D. R., Paduri, A. R., DS, B. N., Desai, S., BG, S., & R, H. (2022). Machine Learning Based Classification Algorithms Performance Analysis for Heart Disease Prediction. 2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Electrical, Electronics and Computer Engineering (UPCON), 2022 IEEE 9th Uttar Pradesh Section International Conference On, 1–8. https://doi-org.egateway.vit.ac.in/10.1109/UPCON56432.2022.9986435

[13]: Vijaya, J. (2023). Heart Disease Prediction using Clustered Genetic Optimization Algorithm. 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), 2023 International Conference, 1072–1077. https://doi-org.egateway.vit.ac.in/10.1109/IITCEE57236.2023.10091050

[14]: Kanna, R. K., Devi, K. Y., Dhivy, A. J. A., Amutha, P. M. D., Gomalavalli, R., & Ambikapathy, A. (2022). Software Development Framework for Cardiac Disease Prediction Using Machine Learning Applications. 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET), Current Development in Engineering and Technology (CCET), 2022 IEEE International Conference On, 1–7. https://doi-org.egateway.vit.ac.in/10.1109/CCET56606.2022.10080016

[15]: Rahman, S., Hasan, M. M., & Sarkar, A. K. (2022). Machine Learning and Deep Neural Network Techniques for Heart Disease Prediction. 2022 25th International Conference on Computer and Information Technology (ICCIT), Computer and Information Technology (ICCIT), 2022 25th International Conference On, 1086–1091. https://doi-org.egateway.vit.ac.in/10.1109/ICCIT57492.2022.10055902

[16]: Tushar, A. M., Wazed, A., Shawon, E., Rahman, M., Hossen, M. I., & Jesmeen, M. Z. H. (2022). A Review of Commonly used Machine Learning Classifiers in Heart Disease Prediction. 2022 IEEE 10th Conference on Systems, Process & Control (ICSPC), Systems, Process & Control (ICSPC), 2022 IEEE 10th

Conference On, 319–323. https://doi-org.egateway.vit.ac.in/10.1109/ICSPC55597.2022.10001742

[17]: Venkatesh, V., Rai, P., Reddy, K. A., Praba, S., & Anushiadevi, R. (2022). An intelligent framework for heart disease prediction deep learning-based ensemble Method. 2022 International Conference on Computer, Power and Communications (ICCPC), Computer, Power and Communications (ICCPC), 2022 International Conference On, 274–280. https://doi-org.egateway.vit.ac.in/10.1109/ICCPC55978.2022.10072285

[18]: Sivaprasad, R., Hema, M., Ganar, B. N., Sunil, D. M., Mehta, V., & Fahlevi, M. (2022). Heart Disease Prediction and Classification using Machine Learning and Transfer Learning Model. 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Automation, Computing and Renewable Systems (ICACRS), 2022 International Conference On, 595–601. https://doi-org.egateway.vit.ac.in/10.1109/ICACRS55517.2022.10029279

[19]: Saju, B., Asha, V., Prasad, A., P, H. K., V, R., & Nirmala, A. P. (2022). Heart Disease Prediction Model using Machine Learning. 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Automation, Computing and Renewable Systems (ICACRS), 2022 International Conference On, 723–729. https://doi-org.egateway.vit.ac.in/10.1109/ICACRS55517.2022.10029273

[20]: A, S. G., Prabha, R., Razmah, M., Veeramakali, T., S, S., & R, Y. (2022). Machine Learning Heart Disease Prediction Using KNN and RTC Algorithm. 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Power, Energy, Control and Transmission Systems (ICPECTS), 2022 International Conference On, 1–5. https://doi-org.egateway.vit.ac.in/10.1109/ICPECTS56089.2022.10047501