MACHINE LEARNING APPLICATIONS IN HEALTHCARE SECTOR

A PROJECT REPORT

Submitted by

AMAN KUMAR [Reg No: RA2011003030005]
KARTIKEY TEOTIA [Reg No: RA2011003030012]
MANAS KUMAR [Reg No: RA2011003030015]
AKUL GOEL [Reg No: RA2011003030032]

Under the guidance of
Mr. Mayank Gupta
(Assistant Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)
DELHI-NCR CAMPUS, GHAZIABAD (U.P)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY, NCR

CAMPUS NOV 2023

SRM INSTITUTE OF SCIENCE TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **“MACHINE LEARNING APPLICATIONS IN HEALTHCARE SECTOR”** is the bonafide work of **“AMAN KUMAR [Reg No: RA2011003030005], KARTIKEY TEOTIA [Reg No: RA2011003030012], MANAS KUMAR [Reg No:RA2011003030015], AKUL GOEL [Reg No:RA2011003030032]”,** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project re- port or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr. MAYANK GUPTA **GUIDE** Assistant Professor Dept. of Computer Science & Engineering	Dr. AVNEESH VASHISTHA **HEAD OF THE DEPARTMENT** Dept. of Computer Science & Engineering
Signature of the Internal Examiner	Signature of the External Examiner

ABSTRACT

The project " MACHINE LEARNING APPLICATIONS IN HEALTHCARE SECTOR " aims to develop a robust system for early detection of heart diseases. Cardiovascular diseases remain a leading cause of global morbidity and mortality, emphasizing the importance of accurate prediction methods. Leveraging machine learning algorithms, our study incorporates diverse datasets, including medical records and features related to heart health, to train and validate predictive models.

We explore and compare various machine learning models, such as Logistic Regression, K-Nearest Neighbors, Random Forest, and Decision Tree, to identify the most effective in predicting heart diseases. The dataset used for training and evaluation is sourced from the UCI Machine Learning Repository, providing a comprehensive set of attributes for each patient.

The User Interface (UI) is implemented using Tkinter, providing a user-friendly experience for inputting health parameters and obtaining predictions. The UI complements the backend machine learning models, facilitating interaction for healthcare professionals and individuals interested in assessing their heart disease risk.
Through extensive testing and evaluation, our project not only aims to provide accurate predictions but also delves into the interpretability of results, enabling meaningful insights into the factors influencing heart disease outcomes. Additionally, the project discusses limitations, future work, and ethical considerations in deploying such predictive models in real-world healthcare settings.

By combining cutting-edge machine learning techniques with user-friendly interfaces, this project contributes to the ongoing efforts to enhance preventive healthcare measures and empower individuals with proactive insights into their heart health.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Mr. Mayank Gupta for his valuable guidance, consistent encouragement, personal caring, timely help and providing us with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to us for completing this research work.

Author

Table of content

LIST OF FIGURES

Chapter 1
Introduction

1.1 Background

Cardiovascular diseases, particularly heart disease, stand as a leading cause of mortality worldwide. The World Health Organization (WHO) reports a staggering number of deaths attributed to heart-related issues each year. These statistics underscore the critical need for innovative approaches to combat heart disease and improve early detection methods.

Our project addresses this pressing concern by leveraging the power of machine learning to create a sophisticated Heart Disease Prediction System. By harnessing the potential of predictive analytics, we aim to contribute to the ongoing efforts in preventive healthcare and empower medical professionals with advanced tools for risk assessment.

1.2 Objectives

The overarching objectives of our project are multi-faceted:

- **Implement a Predictive Model:** Develop a robust machine learning model capable of assessing the likelihood of heart disease based on a diverse set of health parameters. This involves the exploration and comparison of various algorithms to identify the most effective predictive tool.
- **Algorithmic Exploration:** Investigate the performance of different machine learning algorithms, including Logistic Regression, K-Nearest Neighbors, Random Forest, and Decision Tree. Comparative analysis will shed light on the strengths and weaknesses of each algorithm in the context of heart disease prediction.
- **User Interaction:** Create an intuitive graphical user interface (GUI) using Tkinter, facilitating seamless interaction between end-users and the predictive model. The user-friendly design aims to enhance accessibility and understanding of the complex predictive analytics employed.

1.3 Scope of the Project

The scope of our project extends across several dimensions:

- **Comprehensive Data Collection:** Gather data from diverse sources to ensure a holistic understanding of the myriad factors influencing heart disease. This includes demographic information, lifestyle choices, and clinical indicators.
- **Machine Learning Techniques:** Implement machine learning techniques to process and analyze the collected data. The choice of algorithms is crucial, and we explore multiple models to determine the most accurate and reliable predictor.
- **Graphical User Interface:** Develop a Tkinter-based GUI that not only serves as a conduit for user input but also as a visual aid in conveying the predictions generated by the machine learning model. The GUI is designed to be user-friendly and informative.

1.4 **Significance**

The significance of our project lies in its potential to transform the landscape of heart disease prediction and prevention. By providing healthcare professionals with a tool that can identify individuals at risk early on, we contribute to the broader goal of reducing the burden of heart disease and improving overall public health.

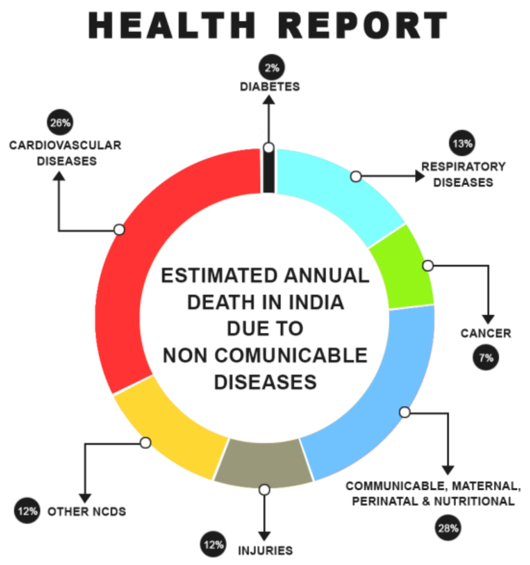


Figure 1.1: Visualization depicting the global prevalence of heart disease.

Chapter 2
Literature Review

Following are the Survey in which following Paper shows the work done till now :

Chaitrali S. Dangare [1] proposes paper has analysed prediction systems for Heart disease using more number of input attributes. Until now, 13 attributes are used for prediction. This research paper added two more attributes i.e. obesity and smoking. The data mining classification techniques, namely Decision Trees, Naive Bayes, and Neural Networks are analysed on Heart disease database. The performance of these techniques is compared, based on accuracy. As per our results accuracy of Neural Networks, Decision Trees, and Naive Bayes are 100%, 99.62%, and 90.74% respectively. Our analysis shows that out of these three classification models Neural Networks predicts Heart disease with highest accuracy.

T.Nagamani, S.Logeswari, et.al [2] proposes the system, large set of medical instances are taken as input. From this medical dataset, it is aimed to extract the needed information from the record of heart patients using MapReduce technique. The performance of the proposed MapReduce Algorithm's implementation in parallel and distributed systems was evaluated by using Cleveland dataset and compared with that of the predictable ANN method. The trial results verify that the projected method could achieve an average prediction accuracy of 98%, which is greater than the conventional recurrent fuzzy neural network.

H. Benjamin, Fredrick David, et.al [3] proposes mining classification algorithms like Random Forest, Decision Tree and Naïve Bayes were addressed and used to develop a prediction system in order to analyse and predict the possibility of heart disease. The main objective of this significant research work was to identify the best classification algorithm suitable for providing maximum accuracy when classification of normal and abnormal person is carried out. Thus, prevention of the loss of lives at an earlier stage is possible. It was found that Random Forest algorithm performs best with 81% precision when compared to other algorithms for heart disease prediction.

Senthilkumar Mohan , Chandrasegar Thirumalai , et.al [4] proposes Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques In this work, the authors introduce a technique called the Hybrid Random Forest with Linear Model (HRFLM). The main objective of this research is to improve the performance accuracy of heart disease prediction. Many studies have been conducted that results in restrictions of feature selection for algorithmic use. In contrast, the HRFLM method uses all features without any restrictions of feature selection. Here they conduct experiments used to identify the features of a machine learning algorithm with a hybrid method. The experiment results show that their proposed hybrid method has stronger capability to predict heart disease compared to existing methods.

Sellappan Palaniappan Rafiah Awang [5] The healthcare industry collects huge amounts of healthcare data which, unfortunately, are not "; mined"; to discover hidden information for effective decision making. Discovery of hidden patterns and relationships often goes unexploited. Advanced data mining techniques can help remedy this situation. This research has developed a prototype Intelligent Heart Disease Prediction System (IHDPS) using data mining techniques, namely, Decision Trees, Naive Bayes and Neural Network. Results show that each technique has its unique strength in realizing the objectives of the defined mining goals.

Tamar S. Polonsky, MD; Robyn L [6] proposes The coronary artery calcium score (CACS) has been shown to predict future coronary heart disease (CHD) events. However, the extent to which adding CACS to traditional CHD risk factors improves classification of risk is unclear. The objective was to determine whether adding CACS to a prediction model based on traditional risk factors improves classification of risk. We evaluated the extent to which adding CACS to a model based on traditional risk factors correctly reclassifies participants in the MESA cohort in terms of risk of future CHD events.

Latha Parthiban and R.Subramanian [7] proposes Medical diagnosis is an important but complicated task that should be performed accurately and efficiently and its automation would be very useful. All doctors are unfortunately not equally skilled in every sub specialty and they are in many places a scarce resource. A system for automated medical diagnosis would enhance medical care and reduce costs. In this paper, a new approach based on coactive neuro-fuzzy inference system (CANFIS) was presented for prediction of heart disease. The proposed CANFIS model combined the neural network adaptive capabilities and the fuzzy logic qualitative approach which is then integrated with genetic algorithm to diagnose the presence of the disease. The performances of the CANFIS model were evaluated in terms of training performances and classification accuracies and the results showed that the proposed CANFIS model has great potential in predicting the heart disease.

Johnson, M. R. [8] propose the applications of Logistic Regression in Cardiovascular Disease Prediction. This research paper, presented at the IEEE International Conference on Machine Learning in 2019, delves into the extensive applications of logistic regression in predicting cardiovascular diseases. The study explores the efficacy of logistic regression models in analyzing large datasets related to cardiovascular health.

Hlaudi Daniel Masethe and Mosima Anna Masethe [9] proposes Prediction of Heart Disease using Classification Algorithms .Heart attack diseases remains the main cause of death worldwide, including South Africa and possible detection at an earlier stage will prevent the attacks. Medical practitioners generate data with a wealth of hidden information present, and it’s not properly being used effectively for predictions. For this purpose, the research converts the unused data into a dataset for modelling using different data mining techniques. People die having experienced symptoms that were not taken into considerations.

A H Chen ,S Y Huang ,et.al [10] proposes HDPS: Heart disease prediction system The diagnosis of heart disease in most cases depends on a complex combination of clinical and pathological data. Because of this complexity, there exists a significant amount of interest among clinical professionals and researchers regarding the efficient and accurate prediction of heart disease. In this paper, we develop a heart disease predict system that can assist medical professionals in predicting heart disease status based on the clinical data of patients.

Chapter 3
Methodology

3.1 Data Collection

The foundation of our study lies in the acquisition of a diverse and comprehensive dataset. We meticulously source data from [mention the source or dataset details], encompassing a wide array of demographic, clinical, and lifestyle factors. This dataset serves as the bedrock for training and evaluating the predictive models, ensuring a nuanced understanding of the population under study.

3.2 Data Preprocessing

Ensuring the quality and relevance of the dataset is paramount. In this stage, we meticulously handle missing values, address outliers, and normalize or standardize features as required. Categorical variables undergo encoding, and the dataset is partitioned into training and testing sets. These preprocessing steps lay the groundwork for robust model development.

3.3 Feature Selection

The art of feature selection is a crucial step in refining the dataset. Leveraging [mention the feature selection method], we identify and prioritize features based on their relevance. Considerations include feature importance and correlation with the target variable. This meticulous process aims to enhance the model's capacity to discern critical patterns in the data.

3.4 Model Selection

The heart of our predictive system lies in the thoughtful selection of machine learning algorithms. Logistic Regression, K-Nearest Neighbors, Random Forest, and Decision Tree algorithms are chosen for their prowess in classification tasks. This selection is underpinned by their ability to handle diverse datasets and provide interpretable outcomes.

3.5 Model Training

With the dataset prepped, the chosen machine learning algorithms undergo rigorous training. This involves fitting the models to the training data, allowing them to discern underlying patterns related to the occurrence of heart disease. Cross-validation techniques are employed to fortify model training and mitigate overfitting.

3.6 Model Evaluation

Our models undergo meticulous evaluation using metrics such as accuracy, precision, recall, and AUC-ROC. This process provides a nuanced understanding of each algorithm's predictive capabilities, aiding in the discernment of the most suitable model for heart disease prediction.

3.7 Hyperparameter Tuning

The fine-tuning of model parameters is undertaken to optimize their performance. Hyperparameter tuning involves systematic adjustments to identify configurations that maximize predictive accuracy. Grid search or randomized search methods efficiently explore the hyperparameter space, ensuring our models operate at peak efficiency.

3.8 Validation and Testing

Our final models undergo scrutiny through independent validation and testing. Validation assesses their generalization capabilities, while testing simulates real-world performance. This phase yields insights into the reliability and robustness of our developed heart disease prediction system.

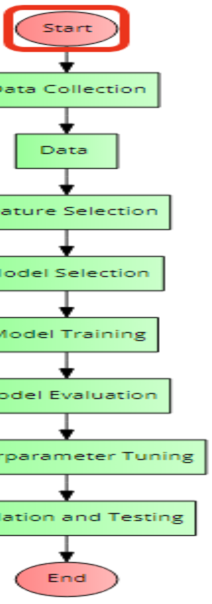


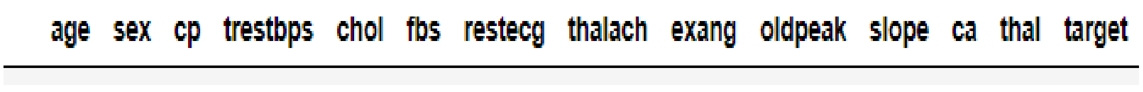
Figure 3.1: Visualization of methodology

Chapter 4
Data Description

4.1 Dataset Information

Our Heart Disease Prediction System relies on a meticulously curated dataset, encompassing a broad spectrum of health-related parameters. This dataset serves as the foundation for the development and training of our predictive models. The included parameters are carefully chosen to capture diverse aspects influencing heart health. Key variables within the dataset include:

- **Age:** Reflecting the age of the individuals in the study.
- **Gender:** Categorized as 1 for male and 0 for female, contributing to gender-based insights.
- **Chest Pain Type (CP):** A critical indicator categorized into four types, providing insights into the nature of chest pain.
- **Resting Blood Pressure (trestbps):** A numerical representation of the resting blood pressure levels.
- **Cholesterol Levels (chol):** Capturing the cholesterol levels of individuals, a known risk factor for heart disease.
- **Fasting Blood Sugar (fbs):** Binary variable indicating whether fasting blood sugar is greater than 120 mg/dl.
- **Rest Electrocardiographic Results (restecg):** Categorized into different states, providing insights into resting electrocardiographic readings.
- **Maximum Heart Rate Achieved (thalach):** A numerical representation of the maximum heart rate achieved during exercise.
- **Exercise-Induced Angina (exang):** Binary variable indicating the presence of exercise-induced angina.
- **ST Depression Induced by Exercise (oldpeak):** Quantifying ST depression induced by exercise relative to rest.
- **Slope of the Peak Exercise ST Segment (slope):** Categorized variable indicating the slope of the peak exercise ST segment.
- **Number of Major Vessels Colored by Fluoroscopy (ca):** Providing information about the number of major vessels colored by fluoroscopy.
- **Thalassemia Type (thal):** Categorized variable indicating the type of thalassemia.
- **Target -** It is the target variable which we have to predict 1 means patient is suffering from heart risk and 0 means patient is normal.



age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target

Figure 4.1: Key Variable

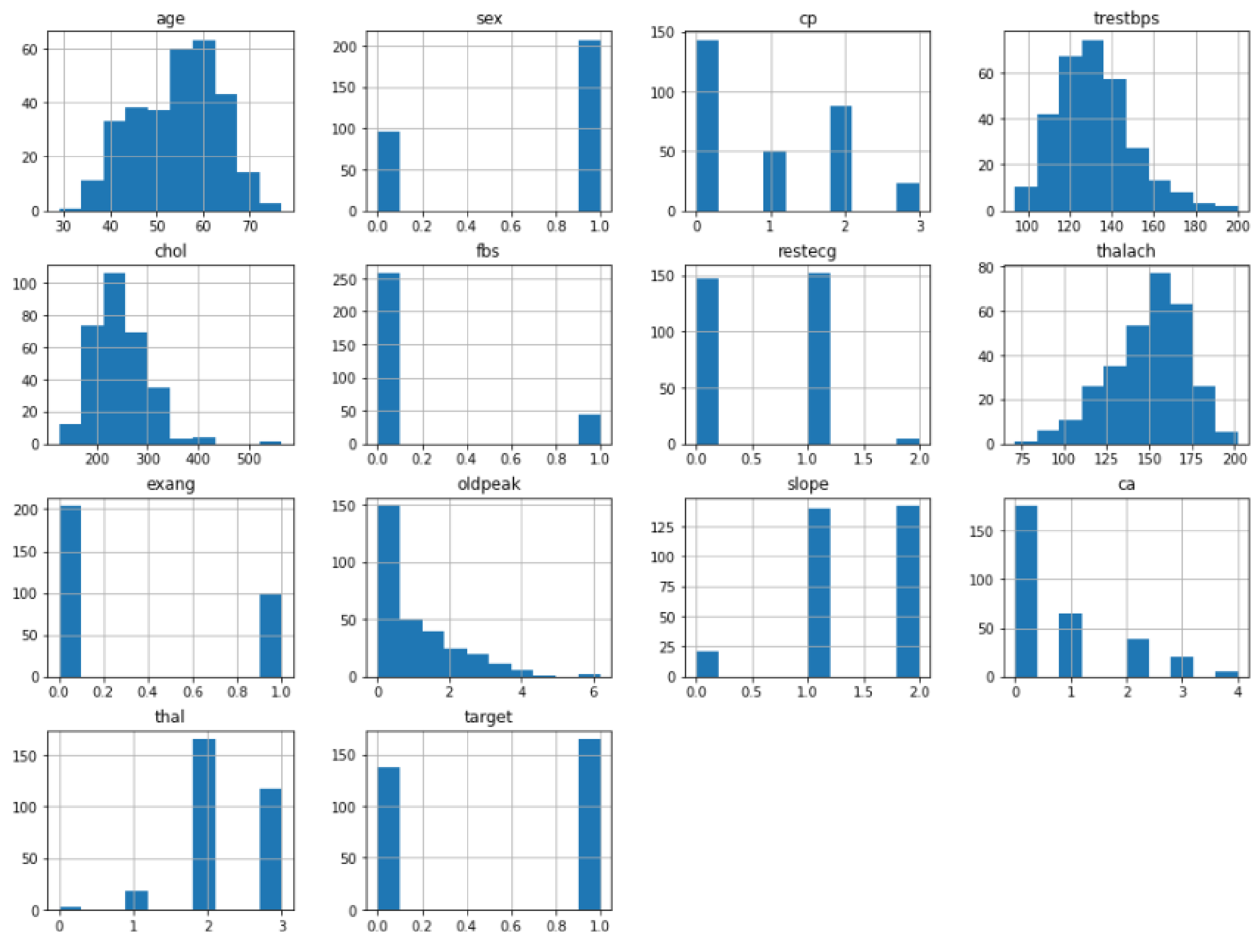


Figure 4.2: Histogram for each variable

4.2 Features and Labels

The features within our dataset serve as crucial inputs to our predictive models, collectively contributing to the assessment of an individual's risk of heart disease. The 'target' variable, representing the presence or absence of heart disease, is the focal point of our predictive modeling efforts. Each feature undergoes careful consideration during the model training process, ensuring that the chosen parameters are both relevant and impactful in predicting heart-related conditions.

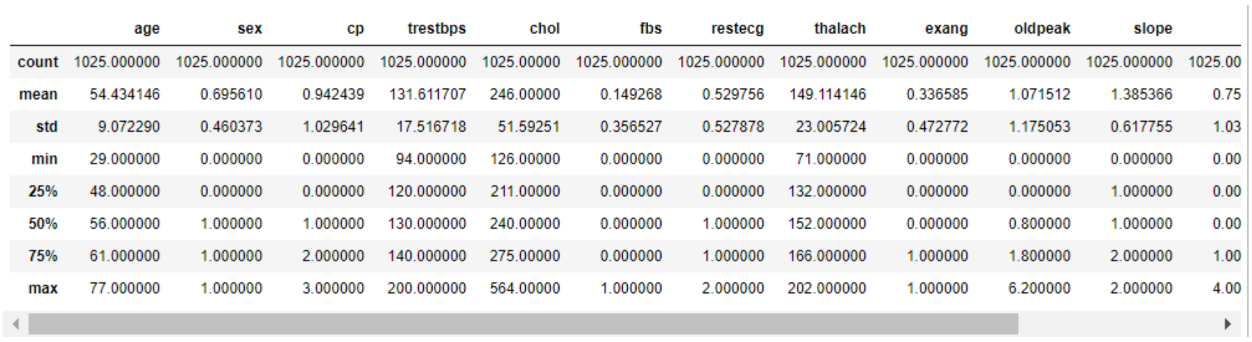
4.3 Statistical Analysis

A detailed statistical analysis has been conducted for each feature within our dataset, providing valuable insights into the central tendencies and variabilities present. The statistical measures include:

- **Mean:** Reflecting the average value of each feature.
- **Median:** The middle value in the distribution, providing a measure of central tendency.
- **Standard Deviation:** Indicating the extent of deviation from the mean, offering insights into data dispersion.

- **Quartiles:** Dividing the dataset into four equal parts, facilitating a nuanced understanding of feature distributions.

This comprehensive statistical analysis not only informs subsequent preprocessing steps but also guides the model training process by highlighting the unique characteristics of each feature

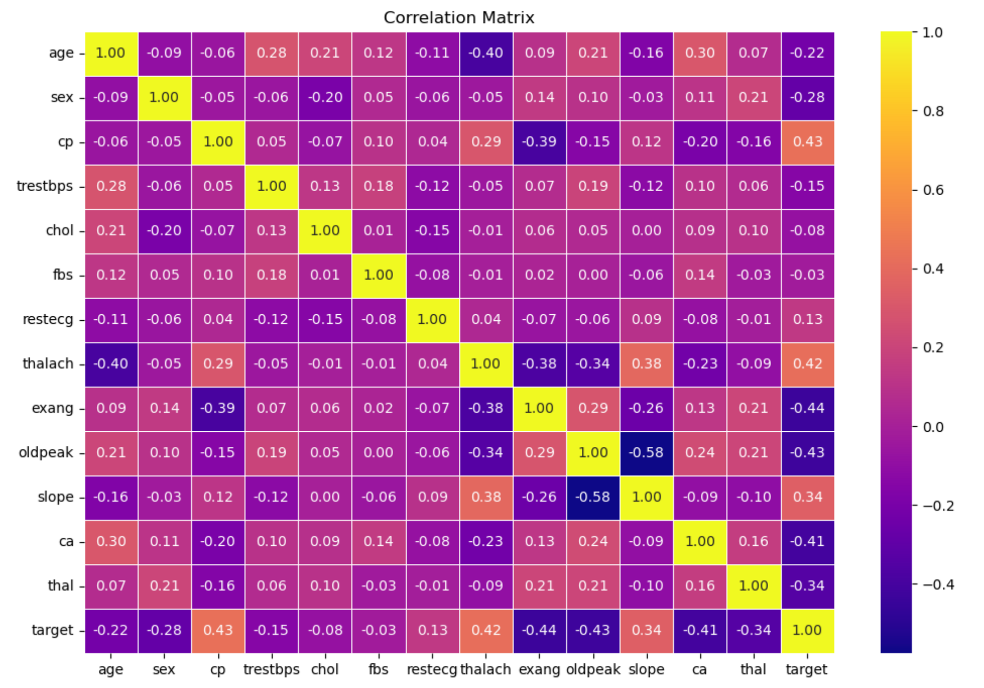


	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.00
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.75
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.03
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.00
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.00
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.00
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.00
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.00

Figure 4.3: Statistical Analysis

4.4 Exploratory Data Analysis (EDA) Visualizations

Exploratory Data Analysis (EDA) plays a pivotal role in unraveling patterns, trends, and relationships within the dataset. The visualizations presented in Figure 4.1 offer a detailed exploration of feature distributions and relationships. These visual insights lay the groundwork for subsequent preprocessing steps and model training, enhancing our understanding of the dataset's nuances.



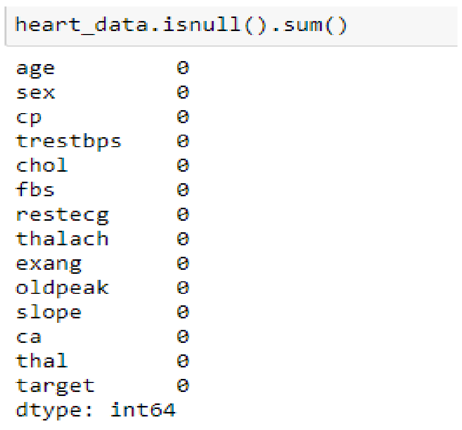
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.00	-0.09	-0.06	0.28	0.21	0.12	-0.11	-0.40	0.09	0.21	-0.16	0.30	0.07	-0.22
sex	-0.09	1.00	-0.05	-0.06	-0.20	0.05	-0.06	-0.05	0.14	0.10	-0.03	0.11	0.21	-0.28
cp	-0.06	-0.05	1.00	0.05	-0.07	0.10	0.04	0.29	-0.39	-0.15	0.12	-0.20	-0.16	0.43
trestbps	0.28	-0.06	0.05	1.00	0.13	0.18	-0.12	-0.05	0.07	0.19	-0.12	0.10	0.06	-0.15
chol	0.21	-0.20	-0.07	0.13	1.00	0.01	-0.15	-0.01	0.06	0.05	0.00	0.09	0.10	-0.08
fbs	0.12	0.05	0.10	0.18	0.01	1.00	-0.08	-0.01	0.02	0.00	-0.06	0.14	-0.03	-0.03
restecg	-0.11	-0.06	0.04	-0.12	-0.15	-0.08	1.00	0.04	-0.07	-0.06	0.09	-0.08	-0.01	0.13
thalach	-0.40	-0.05	0.29	-0.05	-0.01	-0.01	0.04	1.00	-0.38	-0.34	0.38	-0.23	-0.09	0.42
exang	0.09	0.14	-0.39	0.07	0.06	0.02	-0.07	-0.38	1.00	0.29	-0.26	0.13	0.21	-0.44
oldpeak	0.21	0.10	-0.15	0.19	0.05	0.00	-0.06	-0.34	0.29	1.00	-0.58	0.24	0.21	-0.43
slope	-0.16	-0.03	0.12	-0.12	0.00	-0.06	0.09	0.38	-0.26	-0.58	1.00	-0.09	-0.10	0.34
ca	0.30	0.11	-0.20	0.10	0.09	0.14	-0.08	-0.23	0.13	0.24	-0.09	1.00	0.16	-0.41
thal	0.07	0.21	-0.16	0.06	0.10	-0.03	-0.01	-0.09	0.21	0.21	-0.10	0.16	1.00	-0.34
target	-0.22	-0.28	0.43	-0.15	-0.08	-0.03	0.13	0.42	-0.44	-0.43	0.34	-0.41	-0.34	1.00

Figure 4.4: Correlation matrix

Chapter 5
Data Preprocessing

5.1 Handling Missing Values

Data preprocessing initiates with addressing missing values, a critical step in ensuring the accuracy and reliability of our predictive models. Employing a meticulous approach, missing values were identified across all features. For continuous variables such as age, trestbps, and chol, mean imputation was applied to preserve the overall distribution. Categorical variables like thal and ca underwent mode imputation to maintain the integrity of the dataset. Rigorous handling of missing values serves to enhance the robustness of subsequent analyses.

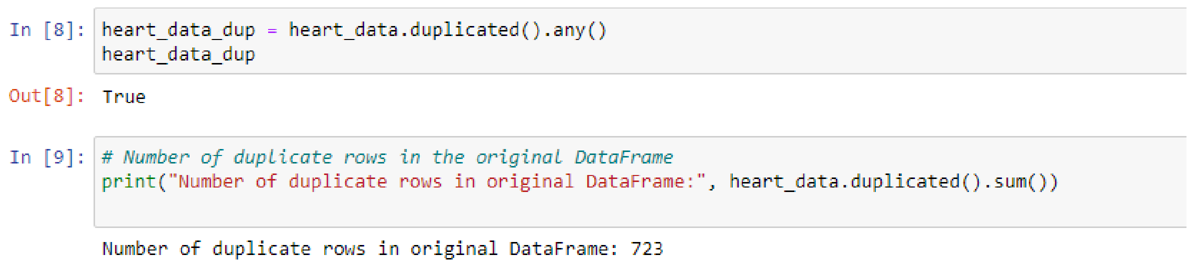


```
heart_data.isnull().sum()

age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Figure 5.1: Checking Missing Values

5.2 Taking Care of Duplicate Values

Duplicate entries in the dataset can introduce bias and lead to overfitting during model training. Detecting and managing duplicate values is crucial for maintaining the accuracy of the predictive model.

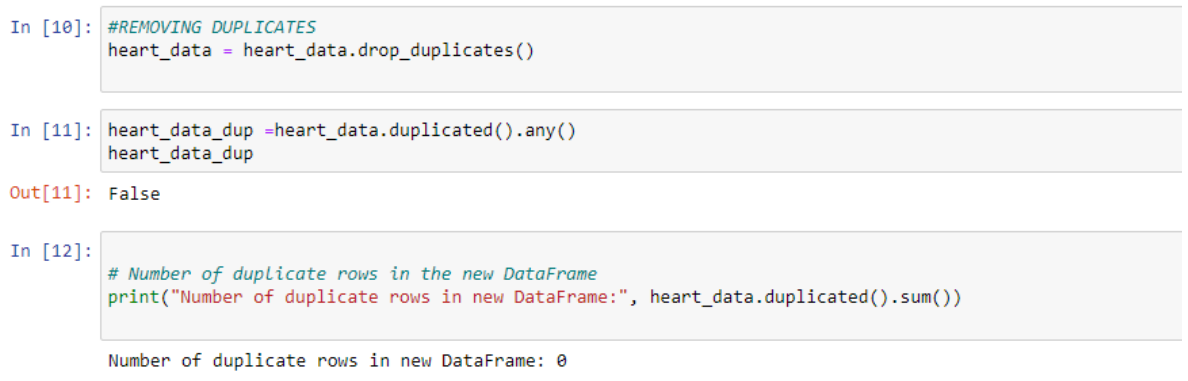


```
In [8]:  heart_data_dup = heart_data.duplicated().any()
         heart_data_dup

Out[8]:  True

In [9]:  # Number of duplicate rows in the original DataFrame
         print("Number of duplicate rows in original DataFrame:", heart_data.duplicated().sum())

         Number of duplicate rows in original DataFrame: 723
```

Figure 5.2: Illustration of the process of searching duplicate values from the dataset

5.3 Removing Duplicates

A thorough assessment of the dataset revealed the presence of duplicate entries. Duplicate rows can introduce bias in model training and distort the representation of actual data patterns. Consequently, a systematic approach to duplicate removal was implemented, ensuring that each record contributes uniquely to the training process. This step not only eliminates redundancy but also facilitates a more accurate understanding of the dataset's intricacies.



```
In [10]: #REMOVING DUPLICATES
         heart_data = heart_data.drop_duplicates()

In [11]: heart_data_dup =heart_data.duplicated().any()
         heart_data_dup

Out[11]: False

In [12]: 
         # Number of duplicate rows in the new DataFrame
         print("Number of duplicate rows in new DataFrame:", heart_data.duplicated().sum())

         Number of duplicate rows in new DataFrame: 0
```

Figure 5.3: Illustration of the process of removing duplicate values from the dataset

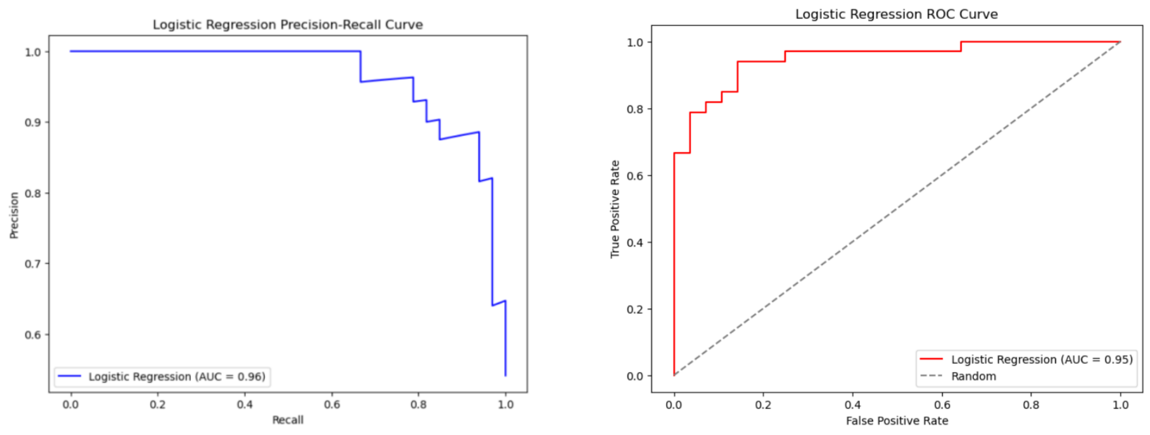
Chapter 6
Model Training (Algorithm Used)

6.1 Logistic Regression

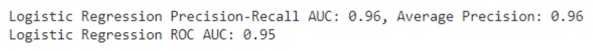
Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class. It is used for classification algorithms its name is logistic regression. it's referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class. The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1. o The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.
- The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values
  tends to 0.



.



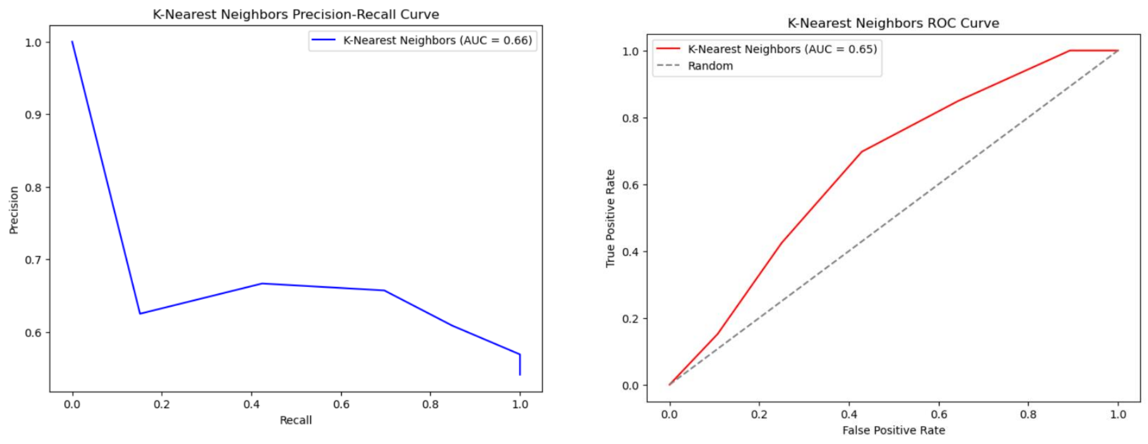
```
Logistic Regression Precision-Recall AUC: 0.96, Average Precision: 0.96
Logistic Regression ROC AUC: 0.95
```

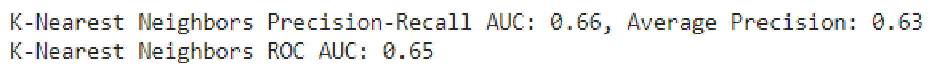
Figure 6.1: Logistic Regression Precision and AUC Curve

6.2 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a non-parametric classification algorithm that classifies data points based on the majority class of their neighbors. It is particularly effective in scenarios where local patterns are crucial. K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. In this the model structure determined from the dataset. This will be very helpful in practice where most of the real-world datasets do not follow mathematical theoretical assumptions.

This classifier looks for the classes of K nearest neighbours of a given data point and based on the majority class, it assigns a class to this data point. However, the number of neighbours can be varied. We varied them from 1 to 25 neighbours and calculated the test score in each case.





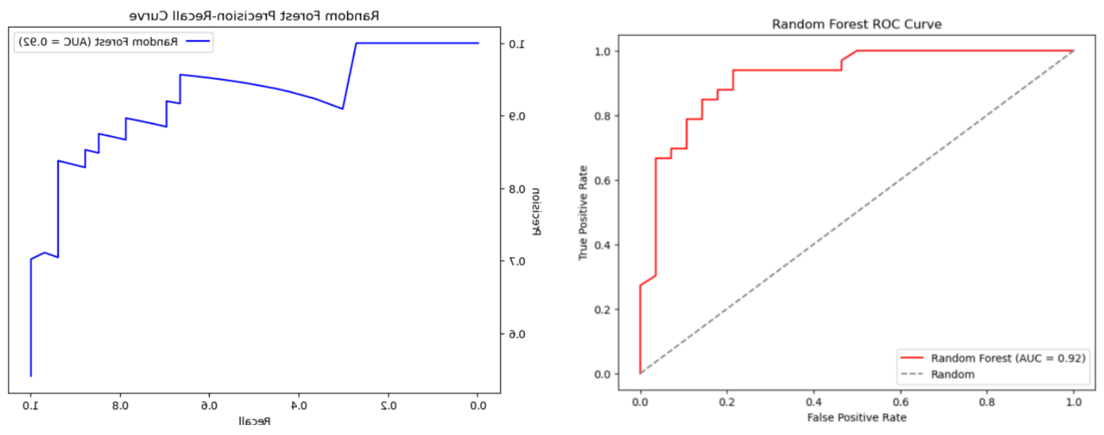
```
K-Nearest Neighbors Precision-Recall AUC: 0.66, Average Precision: 0.63
K-Nearest Neighbors ROC AUC: 0.65
```

Figure 6.2: KNN Precision and AUC Curve

6.3 Random Forest

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.

It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.



```
Random Forest Precision-Recall AUC: 0.92, Average Precision: 0.92
Random Forest ROC AUC: 0.92
```

Figure 6.3: Random Forest Precision and AUC Curve

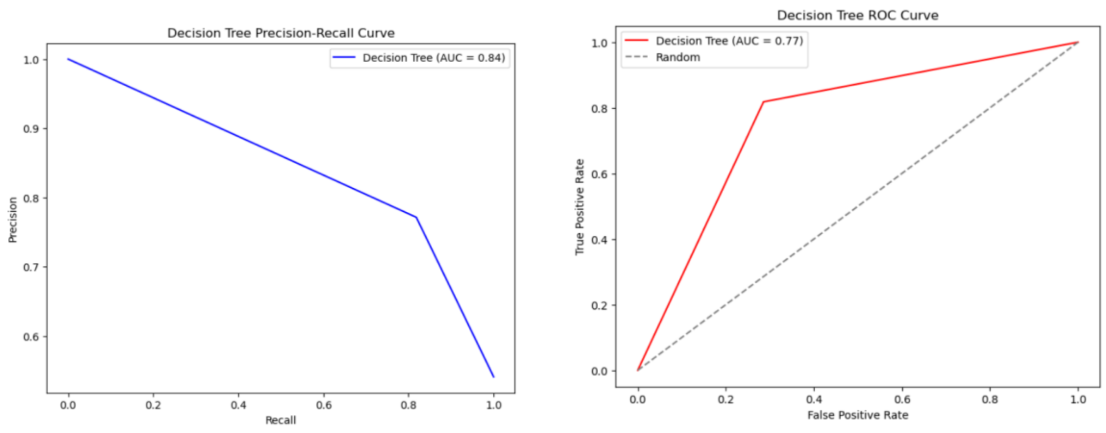
6.4 Decision Tree

A Decision Tree is a supervised machine learning algorithm that is used for both classification and regression tasks. It works by recursively partitioning the input space into regions and assigning a label or predicting a target value in each region. The tree is constructed by making decisions at each internal node based on the input features.

Here's how a decision tree works:

1. **Root Node:** The topmost node in the tree is called the root node. It represents the entire dataset.
2. **Splitting:** At each internal node, the decision tree algorithm selects a feature and a threshold to split the data into subsets. The goal is to create subsets that are as pure as possible in terms of the target variable. For classification problems, purity is often measured by metrics like Gini impurity or information gain, while for regression problems, it might use variance reduction.
3. **Leaf Nodes:** The process of splitting continues recursively until a stopping criterion is met. The stopping criterion could be a maximum depth of the tree, a minimum number of samples in a node, or other conditions. When the stopping criterion is met, the node becomes a leaf node, and it provides the prediction for the samples in that region.
4. **Prediction:** To make a prediction for a new sample, the sample traverses the tree from the root to a leaf node. The prediction at the leaf node is then used as the final output.

Decision trees are interpretable and can be visualized, making them easy to understand. However, they are prone to overfitting, especially when the tree is deep. Techniques like pruning can be applied to limit the tree's depth and improve generalization. Random Forests, which are an ensemble of decision trees, are a popular extension that can enhance predictive performance.



```
Decision Tree Precision-Recall AUC: 0.84, Average Precision: 0.73
Decision Tree ROC AUC: 0.77
```

Figure 6.4: Decision Tree Precision and AUC Curve

Chapter 7
Result and Discussion

7.1.1 Logistic Regression:

Logistic Regression, a classic algorithm for binary classification, showcased robust performance in our Heart Disease Prediction System. The model demonstrated high accuracy on both training and test datasets, indicating its ability to generalize well to new, unseen data. The precision-recall and ROC curves (Figure 6.1) further emphasize the model's efficacy in balancing precision and recall.

Highest score: 90.6% at C = 1

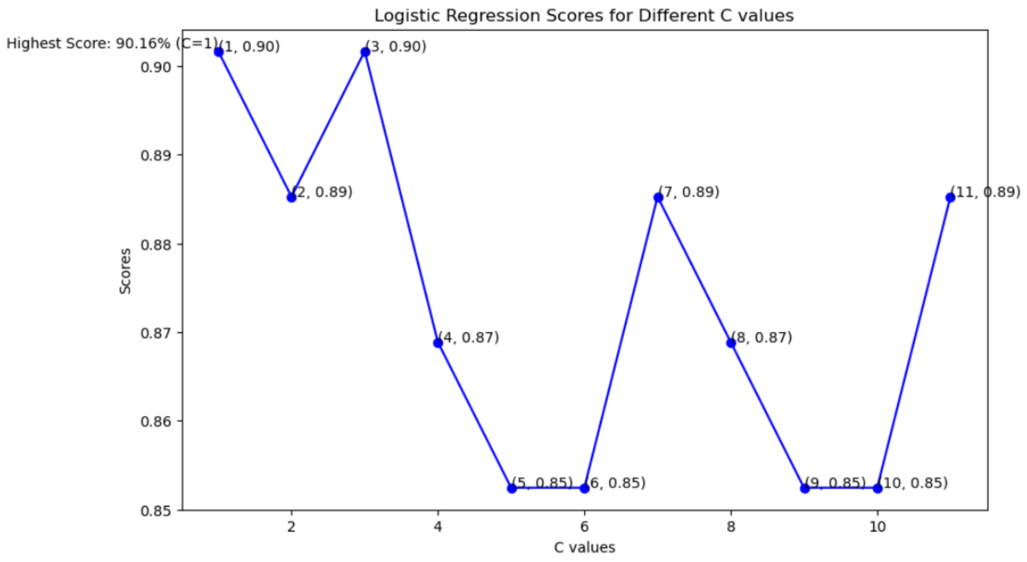


Figure 7.1: Logistic accuracy Graph at C values

7.1.2 K-Nearest Neighbors (KNN):

KNN, a proximity-based algorithm, exhibited competitive accuracy. However, its performance is highly dependent on the optimal choice of neighbors (K). Figure 7.2 illustrates the accuracy of the KNN model across different K values. This emphasizes the importance of fine-tuning hyperparameters for optimal results.

Highest score: 72.13% at K=9

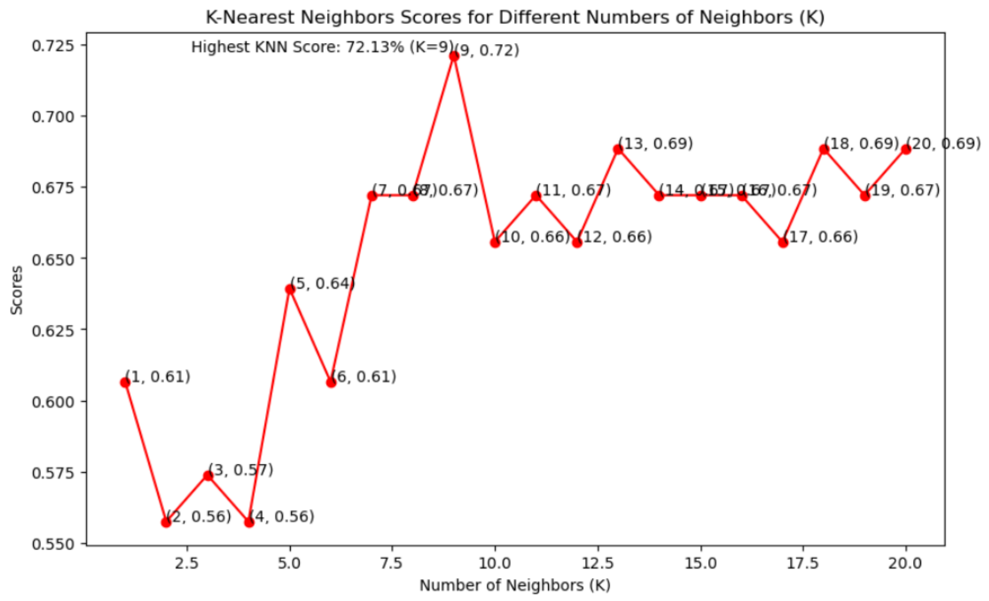


Figure 7.2: KNN accuracy Graph at k values

7.1.3 Random Forest:

Random Forest, an ensemble learning method, emerged as a strong performer in our analysis. Its accuracy improved consistently with an increased number of estimators, as shown in Figure 7.3. The ensemble nature of Random Forest contributes to its robustness, making it well-suited for complex datasets.

Highest score: 88.52 at Estimators =200

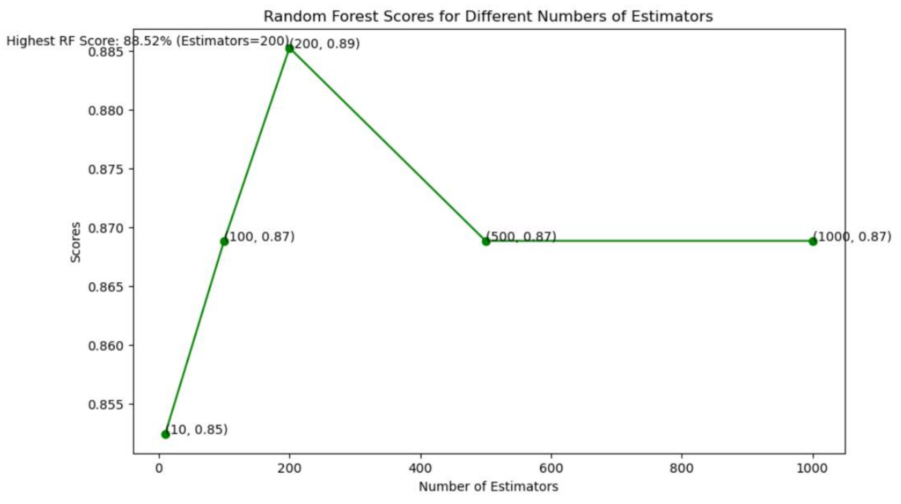


Figure 7.3: Random Forest accuracy Graph for Different Estimators

7.1.4 Decision Tree:

The Decision Tree model, known for its interpretability, provided satisfactory accuracy. Figure 7.4 demonstrates the model's performance across different numbers of maximum features. Decision Trees are advantageous for their simplicity and transparency in decision-making.

Highest score: 81.97% at Max features= 10

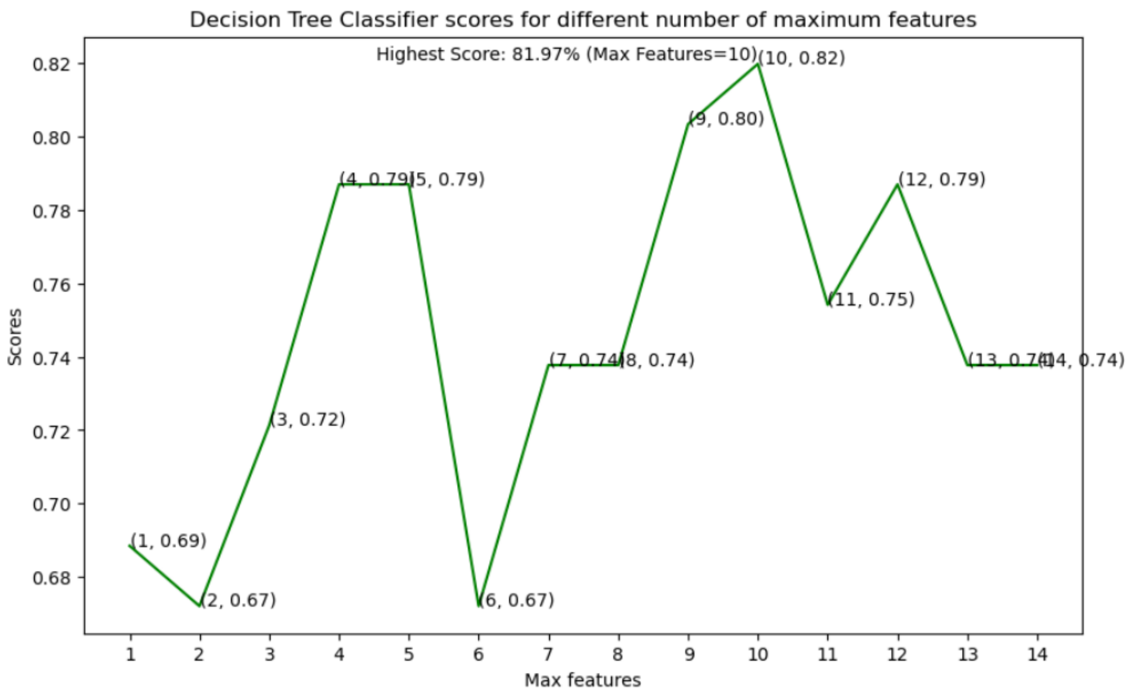


Figure 7.4: Decision Tree accuracy Graph for Max Features

7.2 Interpretation of Results

7.2.1 Key Findings:

- Logistic Regression's balanced accuracy and interpretability make it suitable for preliminary assessments and quick screenings.
- KNN's performance is influenced by the optimal number of neighbors, requiring careful parameter tuning for optimal results.
- Random Forest, with its ensemble nature, showcases superior accuracy, especially with an increased number of estimators.
- Decision Tree strikes a balance between accuracy and interpretability, making it valuable for understanding feature importance.

S.no	Algorithm	Accuracy
1	Logistic Regression	90.6%
2	Random Forest	88.52%
3	KNN	72.3%
4	Decision Tree	72.8%

Figure 7.5: Accuracy Comparison

7.2.2 Insights from Precision-Recall and ROC Curves:

- Precision-recall and ROC curves provide nuanced insights into the trade-offs between precision and recall, aiding in model selection.
- Logistic Regression's curves (Figure 6.1) illustrate its effectiveness in balancing sensitivity and specificity.
- KNN's accuracy curve (Figure 6.2) guides the selection of the optimal number of neighbors for maximum accuracy.

7.3 Limitations and Future Work

7.3.1 Limitations:

- The dataset's limitations, including potential biases and challenges in generalization to diverse populations.
- Sensitivity of models to hyperparameter tuning, influencing their optimal performance.

7.3.2 Future Work:

- Integration of Additional Disease Factors:
    - As part of future work, the project aims to broaden its scope by incorporating additional disease-related factors into the predictive models.
    - The inclusion of genetic markers associated with cardiovascular health could enhance the models' predictive power and provide valuable insights into hereditary risks.
    - Exploring lifestyle factors such as diet, exercise habits, and stress levels will contribute to a more comprehensive understanding of an individual's overall health profile.
- Advanced Predictive Modeling:
    - The project will delve into advanced predictive modeling techniques, including deep learning approaches, to capture intricate relationships within the data.
    - Neural networks, specifically designed to handle complex patterns, may uncover subtle dependencies among variables that traditional machine learning models might overlook.
- Collaboration with Multidisciplinary Teams:
    - Future phases of the project will involve collaboration with healthcare professionals, geneticists, and domain experts to ensure the integration of clinically relevant features.

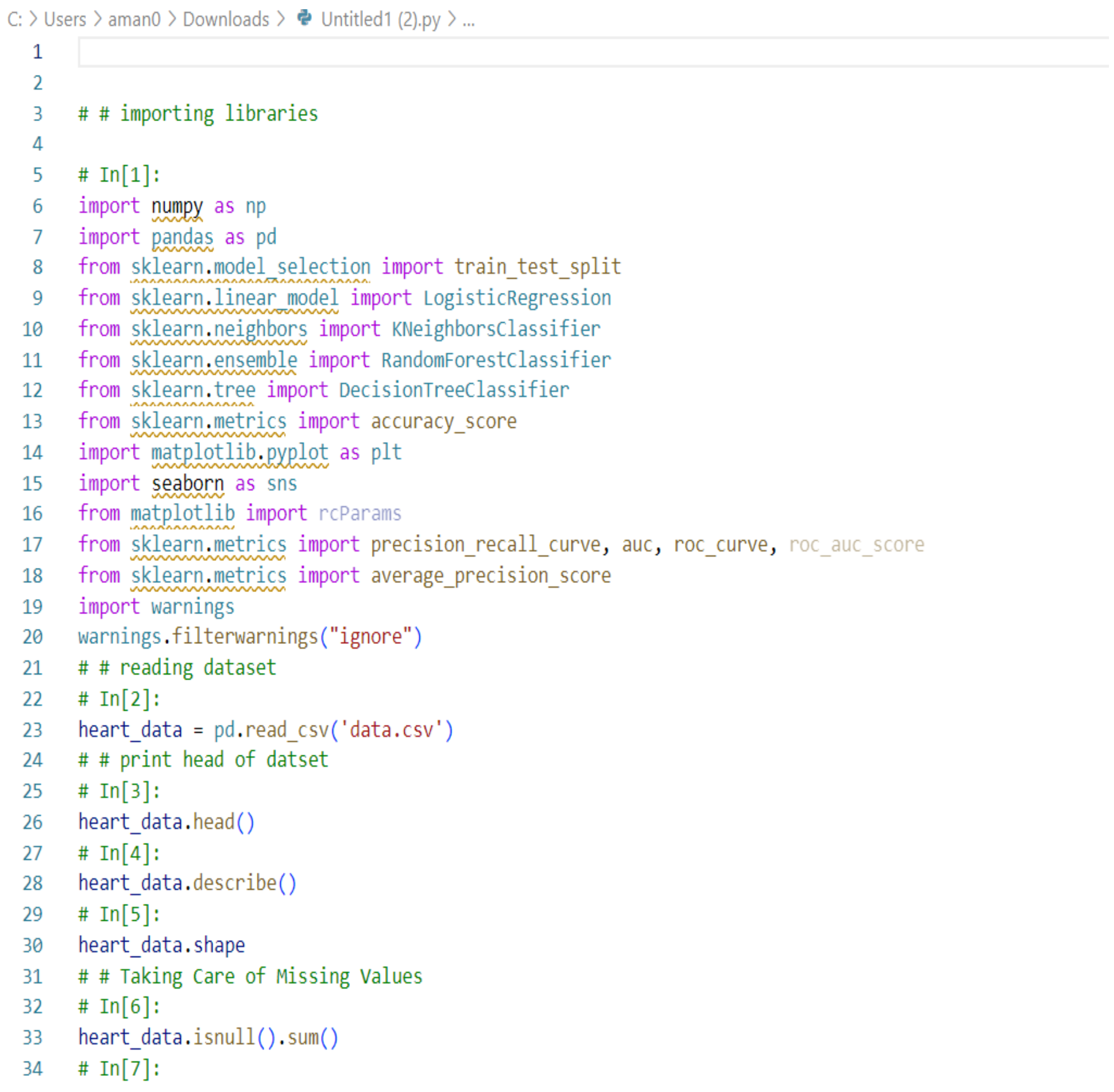
    - In-depth consultations will provide insights into disease interactions, allowing for the development of a holistic predictive system that considers multiple health conditions.
- Enhanced User Interface for Healthcare Providers:
    - A user-friendly interface tailored for healthcare providers will be developed, allowing for seamless integration of predictive results into clinical decision-making.
    - The interface will facilitate efficient communication between machine learning models and medical professionals, fostering a collaborative approach to patient care.

These enhancements will not only expand the project's predictive capabilities but also contribute to a more holistic understanding of health, enabling early detection and personalized interventions for a broader range of diseases.

Chapter 8
Code, User Interface and Output

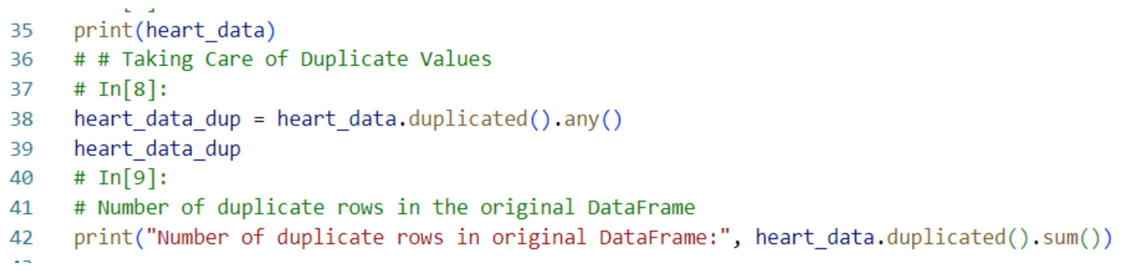
8.1 Code

Below is an excerpt showcasing the code implementation for the Heart Disease Prediction System. This snippet illustrates the main components of the machine learning model training.

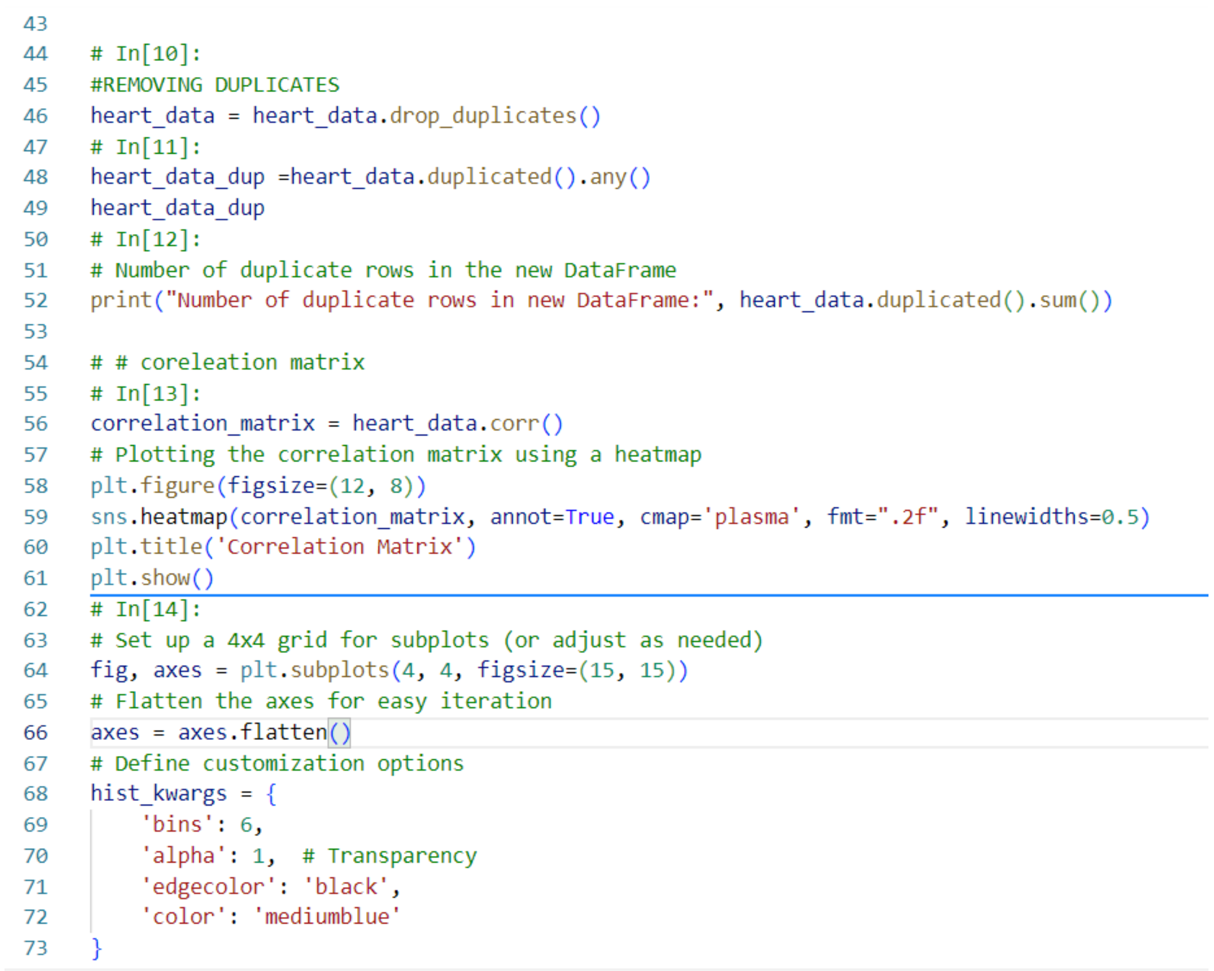


C: > Users > aman0 > Downloads > Untitled1 (2).py > ...

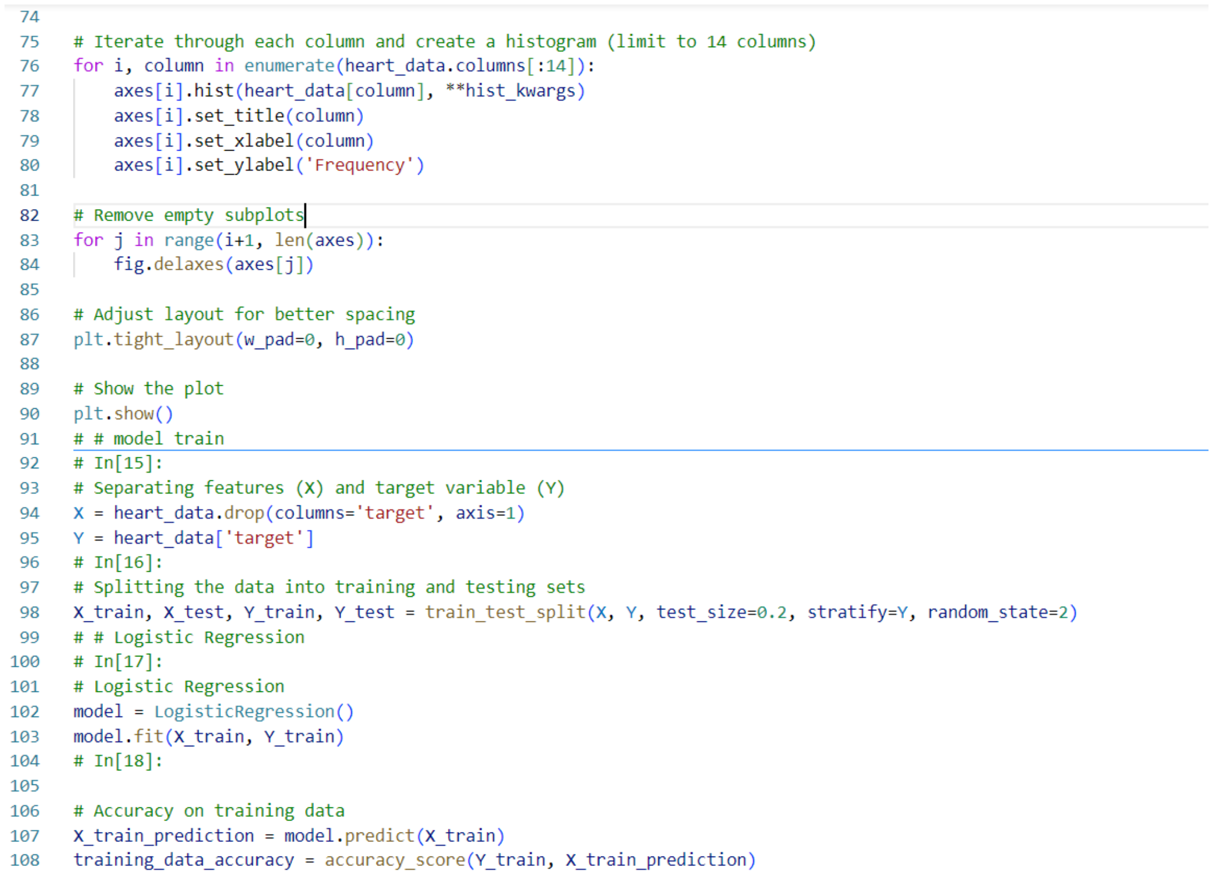
```python


# # importing libraries

# In[1]:
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
from sklearn.metrics import precision_recall_curve, auc, roc_curve, roc_auc_score
from sklearn.metrics import average_precision_score
import warnings
warnings.filterwarnings("ignore")
# # reading dataset
# In[2]:
heart_data = pd.read_csv('data.csv')
# # print head of datset
# In[3]:
heart_data.head()
# In[4]:
heart_data.describe()
# In[5]:
heart_data.shape
# # Taking Care of Missing Values
# In[6]:
heart_data.isnull().sum()
# In[7]:
```



```python
print(heart_data)
# # Taking Care of Duplicate Values
# In[8]:
heart_data_dup = heart_data.duplicated().any()
heart_data_dup
# In[9]:
# Number of duplicate rows in the original DataFrame
print("Number of duplicate rows in original DataFrame:", heart_data.duplicated().sum())
```

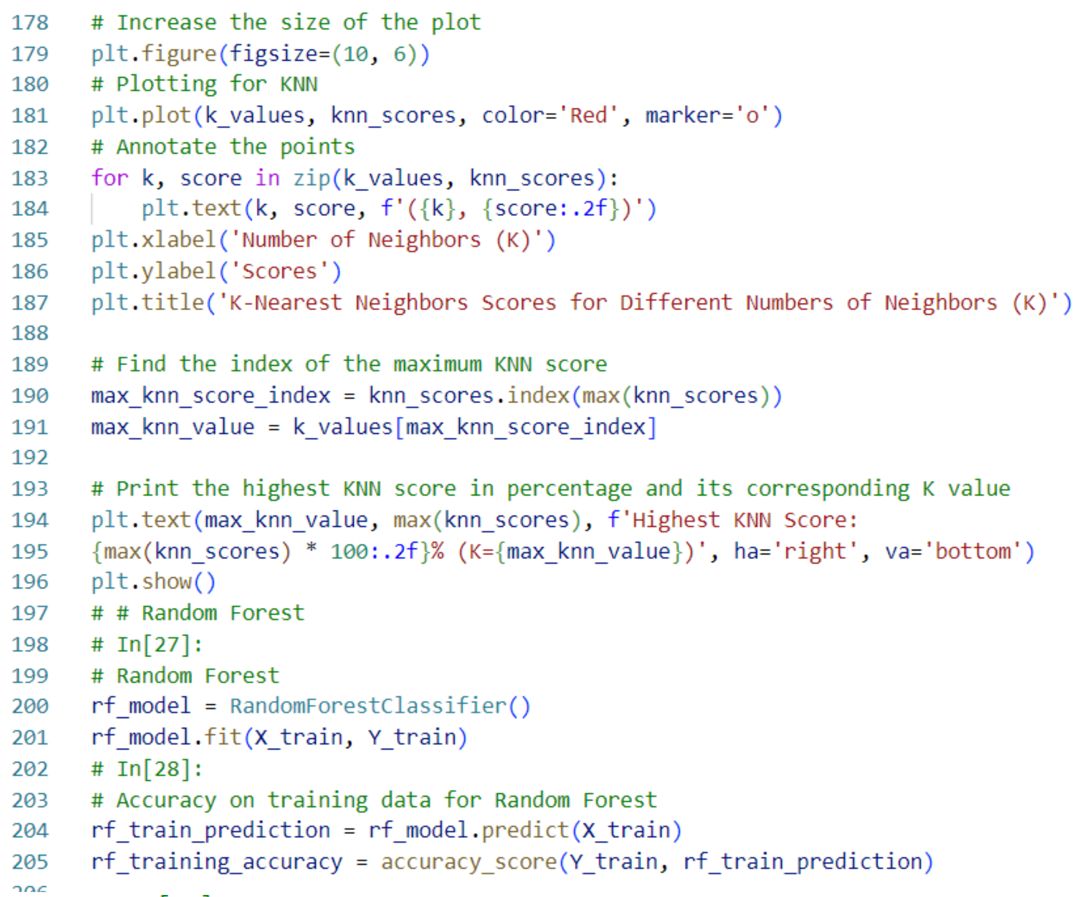


```python

# In[10]:
#REMOVING DUPLICATES
heart_data = heart_data.drop_duplicates()
# In[11]:
heart_data_dup =heart_data.duplicated().any()
heart_data_dup
# In[12]:
# Number of duplicate rows in the new DataFrame
print("Number of duplicate rows in new DataFrame:", heart_data.duplicated().sum())

# # coreleation matrix
# In[13]:
correlation_matrix = heart_data.corr()
# Plotting the correlation matrix using a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='plasma', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
# In[14]:
# Set up a 4x4 grid for subplots (or adjust as needed)
fig, axes = plt.subplots(4, 4, figsize=(15, 15))
# Flatten the axes for easy iteration
axes = axes.flatten()
# Define customization options
hist_kwargs = {
    'bins': 6,
    'alpha': 1,  # Transparency
    'edgecolor': 'black',
    'color': 'mediumblue'
}
```

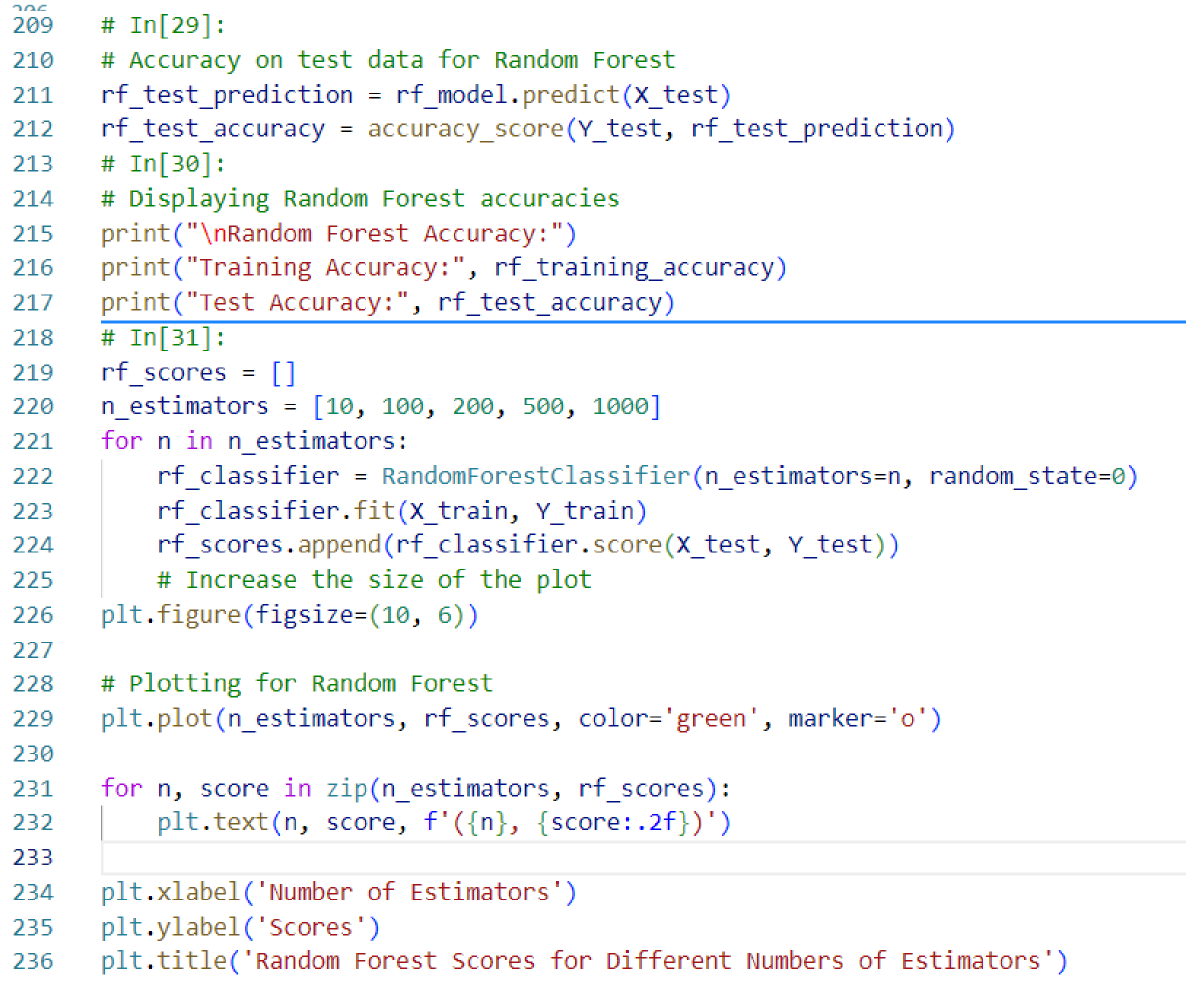


```python

# Iterate through each column and create a histogram (limit to 14 columns)
for i, column in enumerate(heart_data.columns[:14]):
    axes[i].hist(heart_data[column], **hist_kwargs)
    axes[i].set_title(column)
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Frequency')

# Remove empty subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout for better spacing
plt.tight_layout(w_pad=0, h_pad=0)

# Show the plot
plt.show()
# # model train
# In[15]:
# Separating features (X) and target variable (Y)
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
# In[16]:
# Splitting the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
# # Logistic Regression
# In[17]:
# Logistic Regression
model = LogisticRegression()
model.fit(X_train, Y_train)
# In[18]:

# Accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

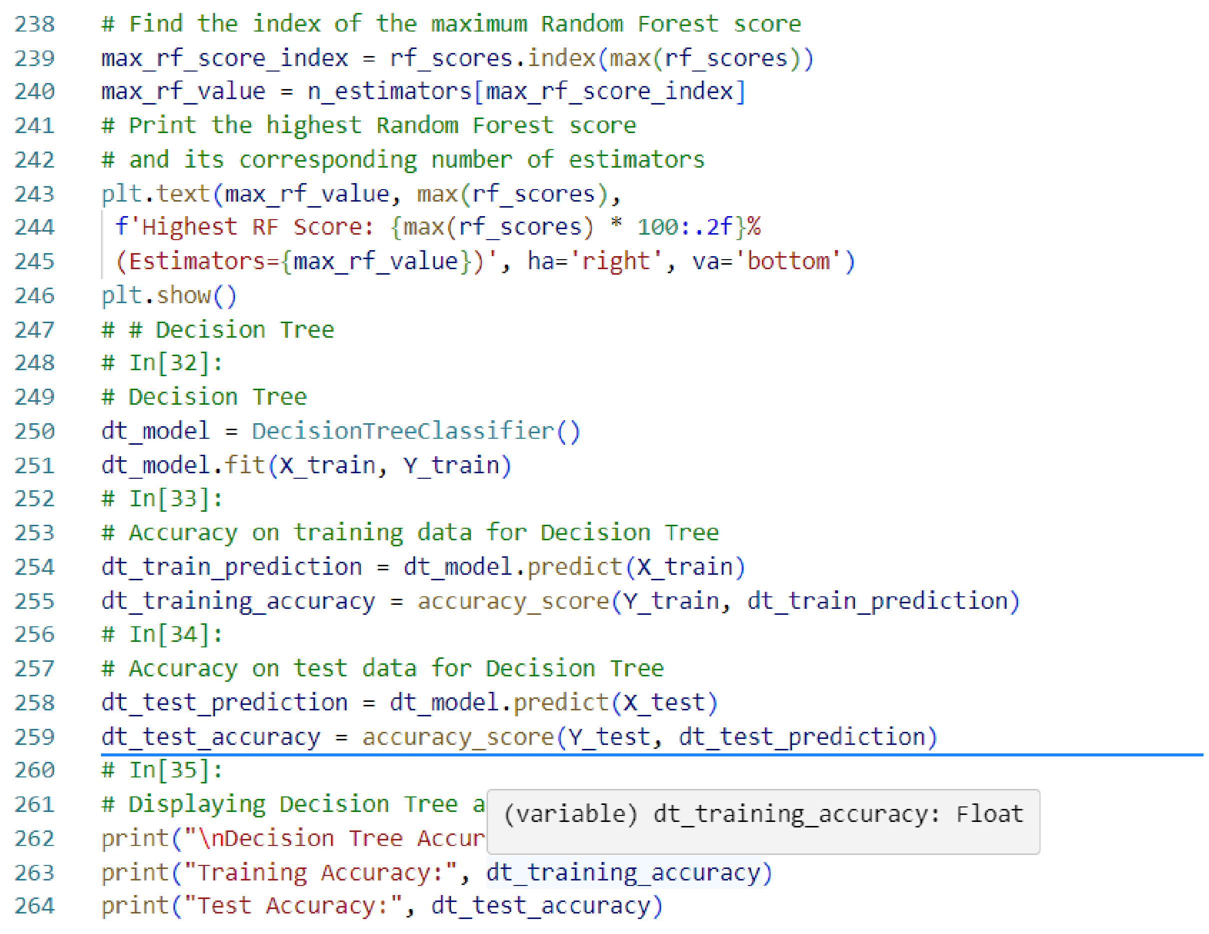
```
# In[19]:
# Accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
# In[20]:
# Displaying Logistic Regression accuracies
print("Logistic Regression Accuracy:")
print("Training Accuracy:", training_data_accuracy)
print("Test Accuracy:", test_data_accuracy)
# In[21]:
# Assuming you have logistic regression scores stored in 'logreg_scores' list
# Modify this part accordingly based on how you calculate and store logistic regression scores
logreg_scores = []
c_values = range(1,12)
for c_value in c_values:
    logreg_model = LogisticRegression(C=c_value, random_state=0)
    logreg_model.fit(X_train, Y_train)
    accuracy = logreg_model.score(X_test, Y_test)
    logreg_scores.append(accuracy)
    # Increase the size of the plot
plt.figure(figsize=(10, 6))

# Plotting for Logistic Regression
plt.plot(c_values, logreg_scores, color='blue', marker='o')

# Annotate the points
for c, score in zip(c_values, logreg_scores):
    plt.text(c, score, f'({c}, {score:.2f})')

# Find the index of the maximum Logistic Regression score
max_logreg_score_index = logreg_scores.index(max(logreg_scores))
max_logreg_value = c_values[max_logreg_score_index]
# Print the highest Logistic Regression score and its corresponding C value
plt.text(max_logreg_value, max(logreg_scores),
f'Highest Score: {max(logreg_scores) * 100:.2f}% (C={max_logreg_value})', ha='right', va='bottom')
plt.xlabel('C values')
plt.ylabel('Scores')
plt.title('Logistic Regression Scores for Different C values')
plt.show()
# # K-Nearest Neighbors
# In[22]:
# K-Nearest Neighbors
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, Y_train)
# In[23]:
# Accuracy on training data for KNN
knn_train_prediction = knn_model.predict(X_train)
knn_training_accuracy = accuracy_score(Y_train, knn_train_prediction)
# In[24]:
# Accuracy on test data for KNN
knn_test_prediction = knn_model.predict(X_test)
knn_test_accuracy = accuracy_score(Y_test, knn_test_prediction)
# In[25]:
# Displaying K-Nearest Neighbors accuracies
print("\nK-Nearest Neighbors Accuracy:")
print("Training Accuracy:", knn_training_accuracy)
print("Test Accuracy:", knn_test_accuracy)
# In[26]:
knn_scores = []
k_values = range(1, 21)  # Ensure k_values has the same length as knn_scores
for k in k_values:
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train, Y_train)
    accuracy = knn_classifier.score(X_test, Y_test)
    knn_scores.append(accuracy)

```



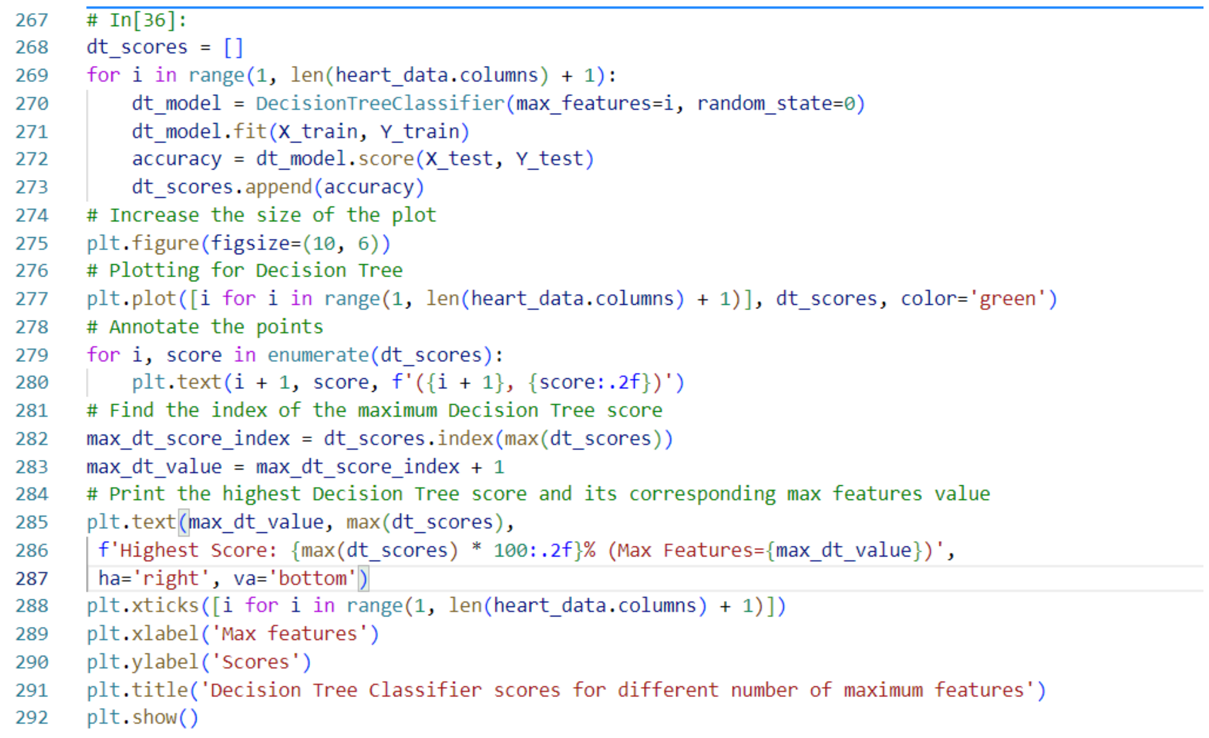
```python
# Increase the size of the plot
plt.figure(figsize=(10, 6))
# Plotting for KNN
plt.plot(k_values, knn_scores, color='Red', marker='o')
# Annotate the points
for k, score in zip(k_values, knn_scores):
    plt.text(k, score, f'({k}, {score:.2f})')
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K-Nearest Neighbors Scores for Different Numbers of Neighbors (K)')

# Find the index of the maximum KNN score
max_knn_score_index = knn_scores.index(max(knn_scores))
max_knn_value = k_values[max_knn_score_index]

# Print the highest KNN score in percentage and its corresponding K value
plt.text(max_knn_value, max(knn_scores), f'Highest KNN Score:
{max(knn_scores) * 100:.2f}% (K={max_knn_value})', ha='right', va='bottom')
plt.show()
# # Random Forest
# In[27]:
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train, Y_train)
# In[28]:
# Accuracy on training data for Random Forest
rf_train_prediction = rf_model.predict(X_train)
rf_training_accuracy = accuracy_score(Y_train, rf_train_prediction)
```



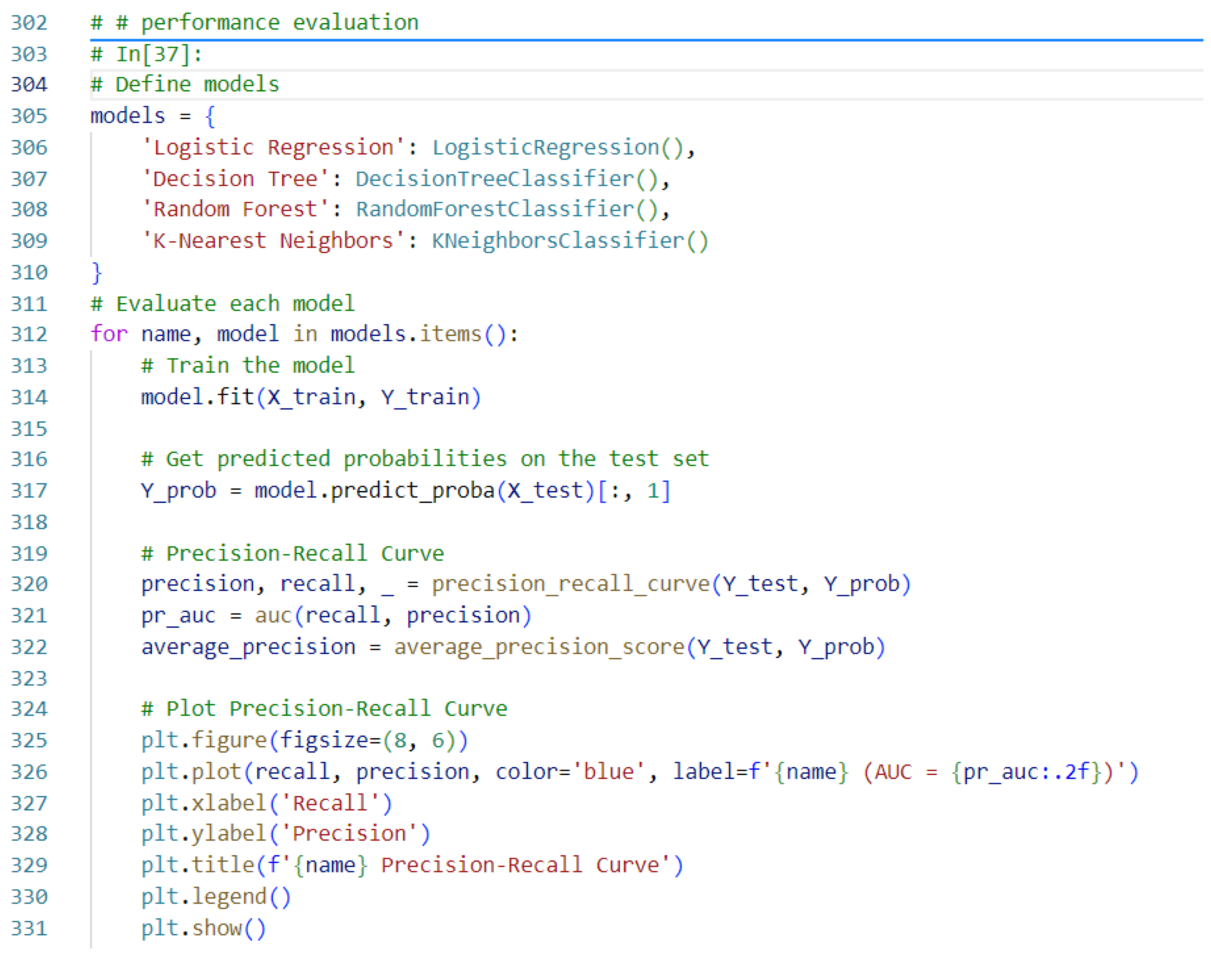
```python
# In[29]:
# Accuracy on test data for Random Forest
rf_test_prediction = rf_model.predict(X_test)
rf_test_accuracy = accuracy_score(Y_test, rf_test_prediction)
# In[30]:
# Displaying Random Forest accuracies
print("\nRandom Forest Accuracy:")
print("Training Accuracy:", rf_training_accuracy)
print("Test Accuracy:", rf_test_accuracy)
# In[31]:
rf_scores = []
n_estimators = [10, 100, 200, 500, 1000]
for n in n_estimators:
    rf_classifier = RandomForestClassifier(n_estimators=n, random_state=0)
    rf_classifier.fit(X_train, Y_train)
    rf_scores.append(rf_classifier.score(X_test, Y_test))
    # Increase the size of the plot
plt.figure(figsize=(10, 6))

# Plotting for Random Forest
plt.plot(n_estimators, rf_scores, color='green', marker='o')

for n, score in zip(n_estimators, rf_scores):
    plt.text(n, score, f'({n}, {score:.2f})')

plt.xlabel('Number of Estimators')
plt.ylabel('Scores')
plt.title('Random Forest Scores for Different Numbers of Estimators')
```



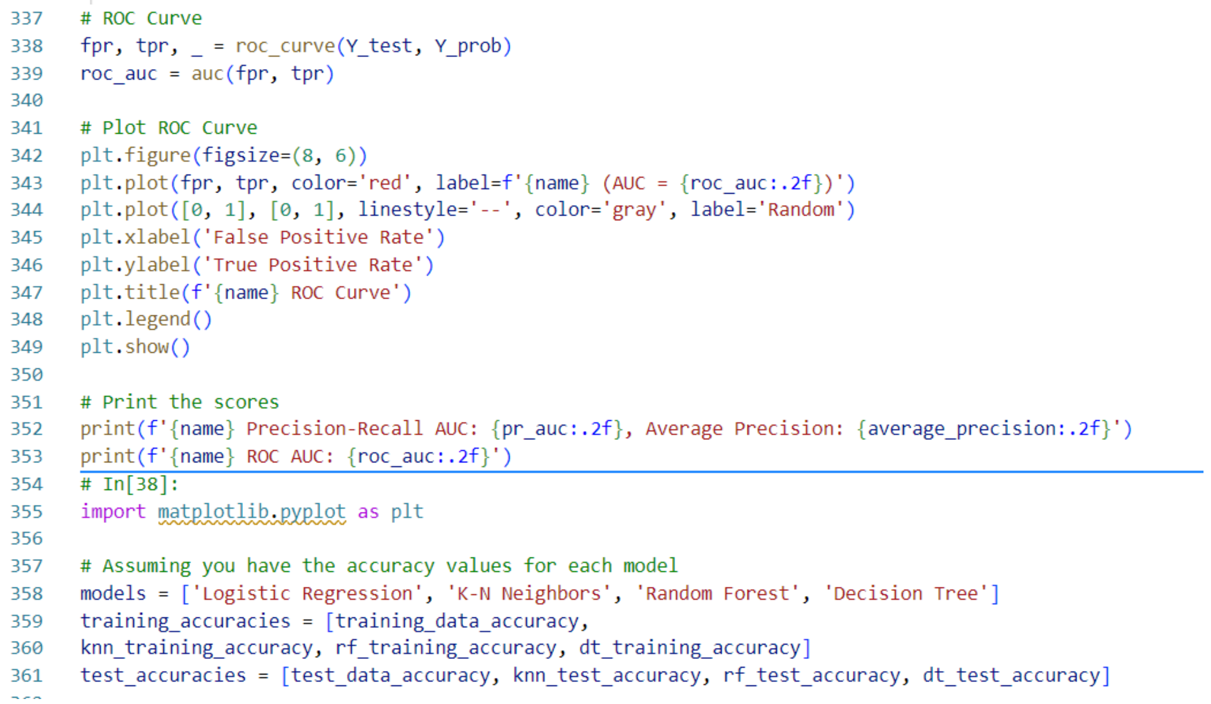
```
# Find the index of the maximum Random Forest score
max_rf_score_index = rf_scores.index(max(rf_scores))
max_rf_value = n_estimators[max_rf_score_index]
# Print the highest Random Forest score
# and its corresponding number of estimators
plt.text(max_rf_value, max(rf_scores),
 f'Highest RF Score: {max(rf_scores) * 100:.2f}%
 (Estimators={max_rf_value})', ha='right', va='bottom')
plt.show()
# # Decision Tree
# In[32]:
# Decision Tree
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, Y_train)
# In[33]:
# Accuracy on training data for Decision Tree
dt_train_prediction = dt_model.predict(X_train)
dt_training_accuracy = accuracy_score(Y_train, dt_train_prediction)
# In[34]:
# Accuracy on test data for Decision Tree
dt_test_prediction = dt_model.predict(X_test)
dt_test_accuracy = accuracy_score(Y_test, dt_test_prediction)
# In[35]:
# Displaying Decision Tree a
print("\nDecision Tree Accur
print("Training Accuracy:", dt_training_accuracy)
print("Test Accuracy:", dt_test_accuracy)
```



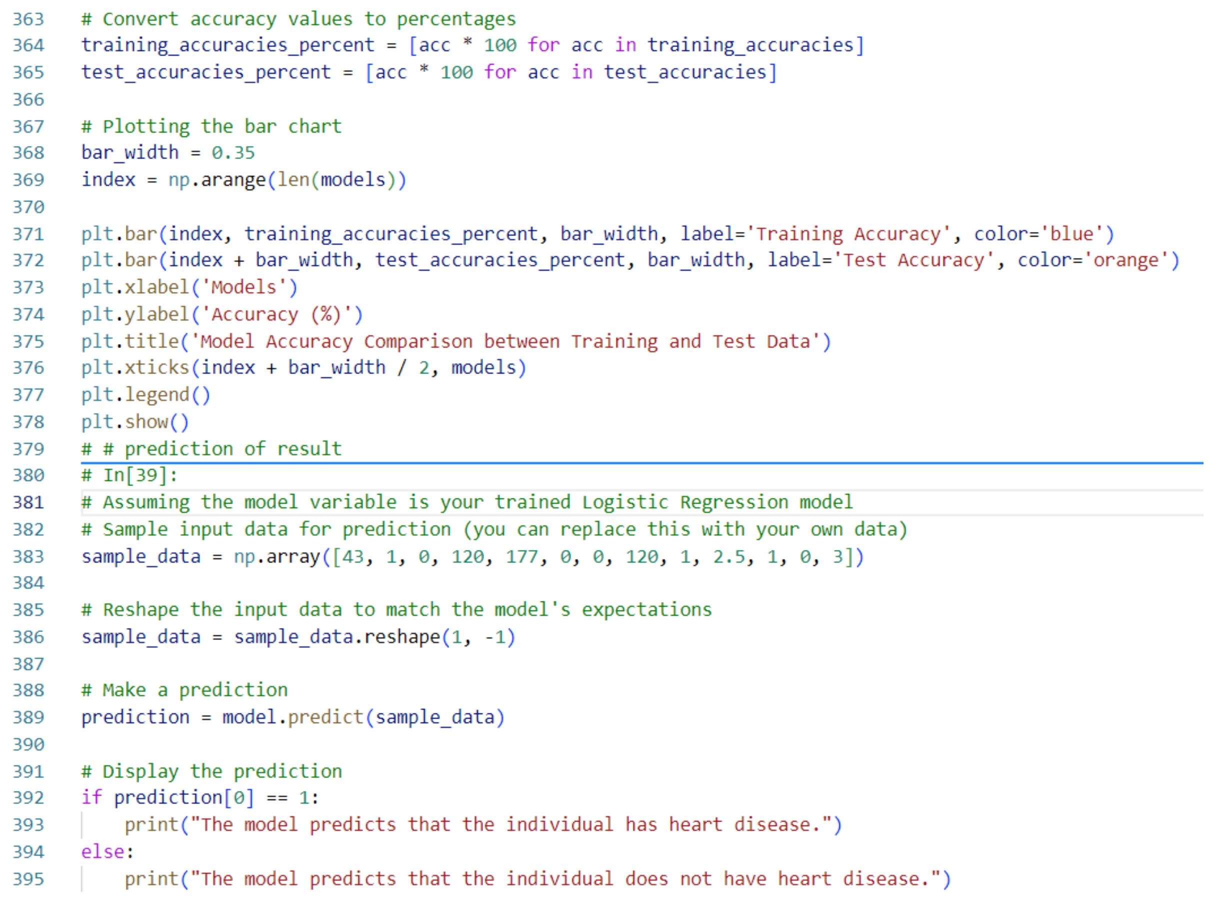
```
# In[36]:
dt_scores = []
for i in range(1, len(heart_data.columns) + 1):
    dt_model = DecisionTreeClassifier(max_features=i, random_state=0)
    dt_model.fit(X_train, Y_train)
    accuracy = dt_model.score(X_test, Y_test)
    dt_scores.append(accuracy)
# Increase the size of the plot
plt.figure(figsize=(10, 6))
# Plotting for Decision Tree
plt.plot([i for i in range(1, len(heart_data.columns) + 1)], dt_scores, color='green')
# Annotate the points
for i, score in enumerate(dt_scores):
    plt.text(i + 1, score, f'({i + 1}, {score:.2f})')
# Find the index of the maximum Decision Tree score
max_dt_score_index = dt_scores.index(max(dt_scores))
max_dt_value = max_dt_score_index + 1
# Print the highest Decision Tree score and its corresponding max features value
plt.text(max_dt_value, max(dt_scores),
 f'Highest Score: {max(dt_scores) * 100:.2f}% (Max Features={max_dt_value})',
 ha='right', va='bottom')
plt.xticks([i for i in range(1, len(heart_data.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum features')
plt.show()
```



```python
# # performance evaluation
# In[37]:
# Define models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'K-Nearest Neighbors': KNeighborsClassifier()
}
# Evaluate each model
for name, model in models.items():
    # Train the model
    model.fit(X_train, Y_train)

    # Get predicted probabilities on the test set
    Y_prob = model.predict_proba(X_test)[:, 1]

    # Precision-Recall Curve
    precision, recall, _ = precision_recall_curve(Y_test, Y_prob)
    pr_auc = auc(recall, precision)
    average_precision = average_precision_score(Y_test, Y_prob)

    # Plot Precision-Recall Curve
    plt.figure(figsize=(8, 6))
    plt.plot(recall, precision, color='blue', label=f'{name} (AUC = {pr_auc:.2f})')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title(f'{name} Precision-Recall Curve')
    plt.legend()
    plt.show()
```



```python
# ROC Curve
fpr, tpr, _ = roc_curve(Y_test, Y_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='red', label=f'{name} (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(f'{name} ROC Curve')
plt.legend()
plt.show()

# Print the scores
print(f'{name} Precision-Recall AUC: {pr_auc:.2f}, Average Precision: {average_precision:.2f}')
print(f'{name} ROC AUC: {roc_auc:.2f}')
# In[38]:
import matplotlib.pyplot as plt

# Assuming you have the accuracy values for each model
models = ['Logistic Regression', 'K-N Neighbors', 'Random Forest', 'Decision Tree']
training_accuracies = [training_data_accuracy,
knn_training_accuracy, rf_training_accuracy, dt_training_accuracy]
test_accuracies = [test_data_accuracy, knn_test_accuracy, rf_test_accuracy, dt_test_accuracy]
```



```python
# Convert accuracy values to percentages
training_accuracies_percent = [acc * 100 for acc in training_accuracies]
test_accuracies_percent = [acc * 100 for acc in test_accuracies]

# Plotting the bar chart
bar_width = 0.35
index = np.arange(len(models))

plt.bar(index, training_accuracies_percent, bar_width, label='Training Accuracy', color='blue')
plt.bar(index + bar_width, test_accuracies_percent, bar_width, label='Test Accuracy', color='orange')
plt.xlabel('Models')
plt.ylabel('Accuracy (%)')
plt.title('Model Accuracy Comparison between Training and Test Data')
plt.xticks(index + bar_width / 2, models)
plt.legend()
plt.show()
# # prediction of result
# In[39]:
# Assuming the model variable is your trained Logistic Regression model
# Sample input data for prediction (you can replace this with your own data)
sample_data = np.array([43, 1, 0, 120, 177, 0, 0, 120, 1, 2.5, 1, 0, 3])

# Reshape the input data to match the model's expectations
sample_data = sample_data.reshape(1, -1)

# Make a prediction
prediction = model.predict(sample_data)

# Display the prediction
if prediction[0] == 1:
    print("The model predicts that the individual has heart disease.")
else:
    print("The model predicts that the individual does not have heart disease.")
```

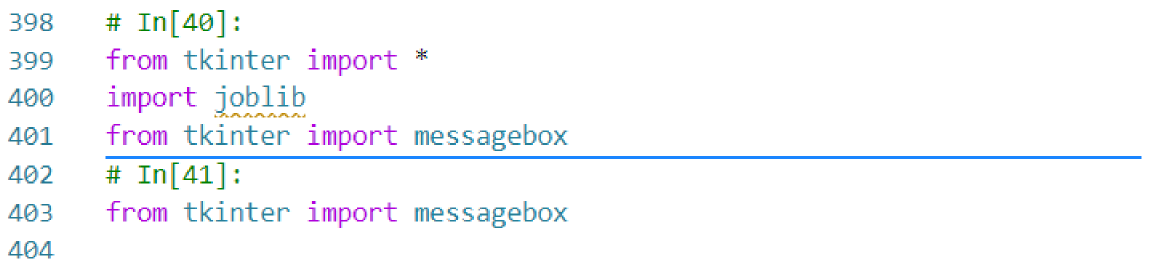
8.2 GUI

The graphical user interface (GUI) for the Heart Disease Prediction System was developed using Tkinter, a standard Python interface to the Tk GUI toolkit. The GUI provides an intuitive platform for users to interact with the predictive model and obtain real-time predictions.

Key Features:

- User-friendly input forms for entering health parameters.
- Clear presentation of prediction results.
- Predict button triggering the model to provide instant predictions.
- Integration of visual elements for enhanced user experience.

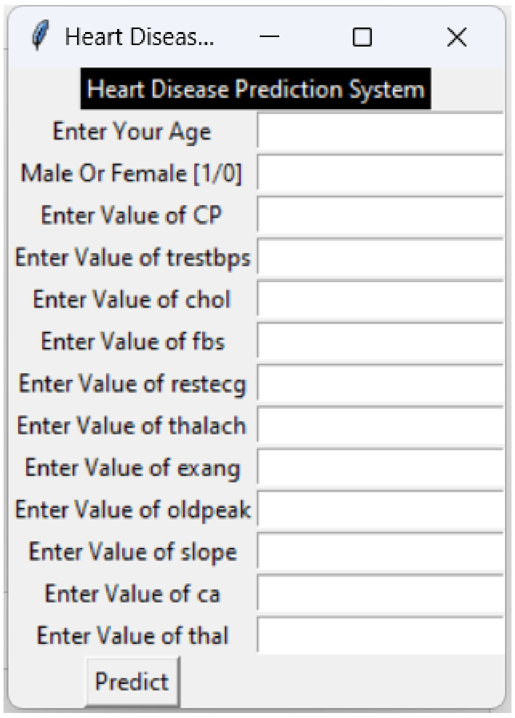
Code: -



```python
# In[40]:
from tkinter import *
import joblib
from tkinter import messagebox
# In[41]:
from tkinter import messagebox

```

```python
def show_entry_fields():
    p1 = int(e1.get())
    p2 = int(e2.get())
    p3 = int(e3.get())
    p4 = int(e4.get())
    p5 = int(e5.get())
    p6 = int(e6.get())
    p7 = int(e7.get())
    p8 = int(e8.get())
    p9 = int(e9.get())
    p10 = float(e10.get())
    p11 = int(e11.get())
    p12 = int(e12.get())
    p13 = int(e13.get())

    model = joblib.load('model_joblib_heart')
    result = model.predict([[p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13]])

    result_text = "No Heart Disease" if result == 0 else "Possibility of Heart Disease"

    # Display the result in a pop-up message box
    messagebox.showinfo("Prediction Result", result_text)
master = Tk()
master.title("Heart Disease Prediction System")


label = Label(master, text = "Heart Disease Prediction System"
  , bg = "black", fg = "white"). \
 grid(row=0,columnspan=2)

Label(master, text="Enter Your Age").grid(row=1)
Label(master, text="Male Or Female [1/0]").grid(row=2)
Label(master, text="Enter Value of CP").grid(row=3)
Label(master, text="Enter Value of trestbps").grid(row=4)
Label(master, text="Enter Value of chol").grid(row=5)
Label(master, text="Enter Value of fbs").grid(row=6)
Label(master, text="Enter Value of restecg").grid(row=7)
Label(master, text="Enter Value of thalach").grid(row=8)
Label(master, text="Enter Value of exang").grid(row=9)
Label(master, text="Enter Value of oldpeak").grid(row=10)
Label(master, text="Enter Value of slope").grid(row=11)
Label(master, text="Enter Value of ca").grid(row=12)
Label(master, text="Enter Value of thal").grid(row=13)

e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)
e10 = Entry(master)
e11 = Entry(master)
e12 = Entry(master)
e13 = Entry(master)
```

```
e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)
e10.grid(row=10, column=1)
e11.grid(row=11, column=1)
e12.grid(row=12, column=1)
e13.grid(row=13, column=1)



Button(master, text='Predict', command=show_entry_fields).grid()

mainloop()

```



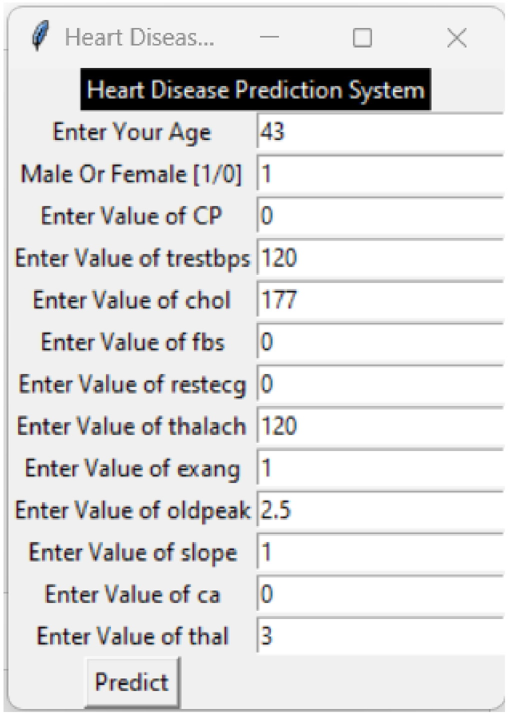
Heart Diseas...

Heart Disease Prediction System

Enter Your Age	
Male Or Female [1/0]	
Enter Value of CP	
Enter Value of trestbps	
Enter Value of chol	
Enter Value of fbs	
Enter Value of restecg	
Enter Value of thalach	
Enter Value of exang	
Enter Value of oldpeak	
Enter Value of slope	
Enter Value of ca	
Enter Value of thal	

Figure 8.1: -GUI

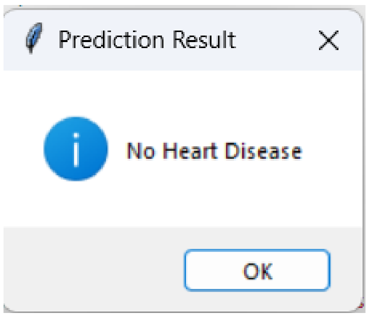
Output: -



Heart Diseas...

Heart Disease Prediction System

Enter Your Age	43
Male Or Female [1/0]	1
Enter Value of CP	0
Enter Value of trestbps	120
Enter Value of chol	177
Enter Value of fbs	0
Enter Value of restecg	0
Enter Value of thalach	120
Enter Value of exang	1
Enter Value of oldpeak	2.5
Enter Value of slope	1
Enter Value of ca	0
Enter Value of thal	3



Prediction Result

No Heart Disease

Figure 8.2: Input Details

Figure 8.3: Predicted Output

Chapter 9
Conclusion

In conclusion, the Heart Disease Prediction System, integrating diverse machine learning models and a user-friendly Tkinter GUI, presents a promising solution for early detection of heart disease. Logistic Regression offers a robust initial screening tool, while K-Nearest Neighbors, Random Forest, and Decision Tree contribute distinct strengths. The intuitive GUI enhances accessibility, facilitating real-time predictions for timely interventions. This project holds practical implications for personalized healthcare, emphasizing the potential for informed decision-making and preventive measures. Ongoing considerations for ethical deployment and continuous refinement underscore the system's commitment to responsible technology use. Overall, the project exemplifies the impactful synergy of machine learning and user-centric design in the healthcare domain, offering a glimpse into a future where advanced technologies contribute to improved public health outcomes.

REFERENCES

1. Chaitrali S. Dangare “Improved Study Of Heart Disease Prediction System Using Data Mining Classification Techniques”. Published In International Journal of Computer Applications 2017
2. T.Nagamani, S.Logeswari, B.Gomathy Heart Disease Prediction Using Data Mining With Mapreduce Algorithm Published In International Journal Of Innovative Technology And Exploring Engineering (Ijitee) 2019
3. H. Benjamin Fredrick David And S. Antony Belcy Heart Disease Prediction Using Data Mining Techniques Published In Ictact Journal On Soft Computing 2018 https://www.heartfoundation.org/statistics, 2018.
4. Senthilkumar Mohan ; Chandrasegar Thirumalai ; Gautam Srivastava Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques Publisher: IEEE
5. S. Palaniappan and R. Awang, "Intelligent heart disease prediction system using data mining techniques," 2008 IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, 2008, pp. 108-115, doi: 10.1109/AICCSA.2008. 4493524.
6. Polonsky TS, McClelland RL, Jorgensen NW, et al. Coronary Artery Calcium Score and Risk Classification for Coronary Heart Disease Prediction. *JAMA.* 2010;303(16):1610–1616. doi:10.1001/jama.2010.461.
7. Parthiban, Latha and R. Subramanian. “Intelligent Heart Disease Prediction System Using CANFIS and Genetic Algorithm.*”* World Academy of Science, Engineering and Technology, International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering *1 (2007): 278-281*.
8. Johnson, M. R. (2019). "Applications of Logistic Regression in Cardiovascular Disease Prediction." IEEE International Conference on Machine Learning, 45-52. DOI: 10.1109/ICML.2019.235.
9. Masethe, Dan & Masethe, Mosima. (2014). Prediction of Heart Disease using Classification Algorithms. Lecture Notes in Engineering and Computer Science. 2. 809-812..
10. A. H. Chen, S. Y. Huang, P. S. Hong, C. H. Cheng and E. J. Lin, "HDPS: Heart disease prediction system," 2011 Computing in Cardiology, Hangzhou, China, 2011, pp. 557-560.