



**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY**

**COLLEGE OF PURE AND APPLIED SCIENCES**

**SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY**

**DEPARTMENT OF COMPUTING**

**BACHELOR OF SCIENCE IN COMPUTER TECHNOLOGY**

**BCT 2315: COMPUTER SYSTEMS PROJECT**

**HEART DISEASE PREDICTION SYSTEM DOCUMENTATION**

**AUTHOR: ONSARE MOSES ORUCHO**

**REGISTRATION NUMBER: SCT212-0070/2018**

**SUPERVISOR: DR. ISAIAH MULANG'**

<b>Abstract</b>	<b>4</b>
<b>1.0 Introduction</b>	<b>5</b>
1.1 Introduction	5
1.3 Motivation	7
1.4 Objectives	8
1.5 Software Development Life Cycle(SDLC) Model	9
1.6 Scope and Limitations	11
1.6.1 Scope	11
1.6.2 Limitations	11
1.7 Summary	12
<b>2.0 Literature Review/Related Work</b>	<b>13</b>
2.1 Previous Works	13
2.2 Machine Learning Algorithms Employed	15
Supervised and Unsupervised Learning Algorithms	15
2.2.1 Logistic Regression	15
2.2.2 Decision Tree	17
2.2.3 Random Forest Classifier	18
2.2.4 Support Vector Machine Algorithm	20
2.2.5 K Nearest Neighbors Classifier	21
2.2.6 XGBoost Classifier	22
2.3 Feature Selection Methods	23
2.3.1 Filter Methods	23
2.3.1.1 Feature Selection by Information Gain	23
2.3.1.2 Feature Selection by Chi-square Test	25
2.3.1.3 Feature Selection by Fisher's Score	25
2.3.1.4 Feature Selection by Correlation Matrix	25
2.3.1.5 Feature Selection by Variance Threshold	26
2.3.1.6 Feature Selection by Mean Absolute Difference (MAD)	26
2.3.2 Wrapper Methods	26
2.3.2.1 Feature Selection By Forward Feature Selection	26
2.3.2.2 Feature Selection By Backward Feature Elimination	26
2.3.2.3 Feature Selection By Exhaustive Feature Selection	26
2.3.3 Embedded Methods	27
2.3.3.1 Feature Selection By LASSO Regularization	27
2.3.3.2 Feature Selection By Random Forest Importance	27
2.4 Performance Measures of Machine Learning Algorithms	27
2.4.1 Confusion Matrix	27

2.5 Open source tools for data mining	29
2.5.1 WEKA	29
2.5.2 TANAGRA	29
2.5.3 RapidMiner	29
2.5.4 Orange	30
2.6 Examples of Prediction Systems	30
2.6.1 Predictive Age Progression Software	30
2.6.2 Automated Disease Prediction System (ADPS)	31
<b>3.0 System Analysis</b>	<b>31</b>
3.1 Feasibility Study	31
3.1.1 Operational Feasibility	32
3.1.2 Technical Feasibility	33
3.1.3 Economic Feasibility	34
3.1.3.1 Development Costs	34
3.1.3.2 Operational Costs	34
3.1.4 Schedule Feasibility	35
3.2 Requirements Specification	35
3.2.1 Functional Requirements	35
3.2.2 Non-Functional Requirements	36
3.2.3 System(Hardware) Requirements	37
3.2.4 Software Requirements	37
<b>4.0 System Design</b>	<b>38</b>
Overview	38
4.1 Architecture diagram	38
4.2 Flowchart Diagram	39
4.3 Use Case Diagram	40
4.4 Sequence Diagram	41
4.5 State Diagram	42
<b>5.0 Implementation</b>	<b>43</b>
5.1 Dataset	43
5.2 Exploratory Data Analysis of provided Dataset	43
5.1.1 Dataset overview	43
5.1.2 Target values analysis	45
5.1.3 Heart disease by sex	45
5.1.4 General Summary of Feature Analysis	46
5.1.5 Feature Selection by Correlation Matrix	48
5.2 Data Processing	49
5.3 Building the Model	50
5.4 Algorithms Used	50
5.4.1 Logistic Regression	50

5.4.2 K-Nearest Neighbors Classifier.	51
5.4.3 Support Vector Machine	52
5.4.4 Decision Tree Classifier	53
5.4.5 Random Forest Classifier	54
5.4.6 Hyperparameter Tuning	54
5.5 Graphical User Interface	57
5.6 Code Documentation Github Link	59
<b>6.0 Conclusion</b>	<b>60</b>
<b>6.1 Results</b>	<b>60</b>
6.2 Remarks	60
6.3 Future Work	61
<b>7.0 References</b>	<b>62</b>

## **Abstract**

The health care industries collect huge amounts of data that contain some hidden information, which is useful for making effective decisions. For providing appropriate results and making effective decisions on data, some advanced data mining techniques are used. In this study, a Heart Disease Prediction System (HDPS) is developed using the KNN algorithm for predicting the risk level of heart disease. The system uses 15 medical parameters such as age, sex, blood pressure, cholesterol, and obesity for prediction. The HDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge. E.g. Relationships between medical factors related to heart disease and patterns, to be established. The obtained results have illustrated that the designed diagnostic system can effectively predict the risk level of heart diseases.

\*Keywords: Data Mining, Decision Tree, Disease Diagnosis, KNN algorithm.

# 1.0 Introduction

## 1.1 Introduction

Coronary heart disease is a narrowing of the small blood vessels that supply blood and oxygen to the heart. This is also called coronary artery disease. Coronary heart disease is usually caused by a condition called atherosclerosis, which occurs when fatty material and a substance called plaque builds up on the walls of arteries. This causes them to get narrow. As the coronary arteries narrow, blood flow to the heart can slow down or stop, causing chest pain, shortness of breath, heart attack, and other symptoms. Men in their 40's have higher risk of Coronary heart disease than women, but as women get older, their risk increases so that it is almost equal to a man's risk. Major risk factors for Coronary heart disease are

1. Diabetes
2. High blood pressure
3. High LDL (bad) cholesterol
4. Low HDL (good) cholesterol
5. Not getting enough physical activity
6. Obesity
7. Smoking

There are certain signs which the American Heart Association lists like the persons having sleep issues, a certain increase and decrease in heart rate (irregular heartbeat), swollen legs, and in some cases weight gain occurring quite fast; it can be 1-2 kg daily. All these symptoms resemble different diseases also like it occurs in aging persons, so it becomes a difficult task to get a correct diagnosis, which results in a fatality in the near future.

Machine learning, a wide discipline with mathematics and computer science as its core, provides a number of new algorithms to build predictive data-driven models. With ML frameworks and large repositories of data, ML started concentrating on the healthcare sectors. Currently it has been shown in clinical cardiology; ML is more skilled in predicting cardiac and all-cause death than manual approaches used individually for clinical purposes. Most frequently used machine Learning algorithms are Logistic Regression, Artificial Neural Networks, Support Vector Machines and Tree Based methods. Typically, the data sets used in ML projects are divided into training, validation and test subsets; training sets covering the bulk of all available data are used to primarily develop the model, validation sets are used to estimate the overall performance of the model or to fine-tune its hyper parameters.

Having discussed the significance of the problem and machine learning algorithms, this study gives an insight on related works in the rest of this document.

## 1.2 Problem Statement

Among various life-threatening diseases, heart disease has garnered a great deal of attention in medical research. The diagnosis of heart disease is a challenging task, which can offer automated predictions about the heart condition of a patient so that further treatment can be made effective. The diagnosis of heart disease is usually based on signs, symptoms and physical examination of the patient. There are several factors that increase the risk of heart

disease, such as smoking habit, body cholesterol level, family history of heart disease, obesity, high blood pressure, and lack of physical exercise.

A major challenge faced by healthcare organizations is the provision of quality services at affordable costs. The quality service implies diagnosing patients properly and administering effective treatments. The available heart disease database consists of both numerical and categorical data. Before further processing, cleaning and filtering are applied on these records in order to filter the irrelevant data from the database. The proposed system can determine the onset of heart disease early on from a historical heart disease database. It can also answer the complex queries for diagnosing heart disease; therefore, it can be helpful to health care practitioners to make intelligent clinical decisions.

## 1.3 Motivation

Having grown up being aptly interested in medicine, I have developed the curiosity to find out how machine learning in the medical world is applied. This project is a way for me to throw myself into machine learning in medicine, and its various applications. I chose this system because of how widely heart disease has affected the world today, being a leader in causes of death worldwide. No one is safe from heart disease, it seems.

By creating this system, a person can download it at home and input data if they cannot afford to go to a hospital for diagnosis, hence avoiding those extra costs and at the same time gaining insight into their status. This provides a very affordable way of diagnosis. It can also be used in hospitals by doctors or nurses after running tests on patients and feeding the data into the system to determine whether the patient has heart disease or not.

I am also highly motivated to implement algorithms on machine learning, which have been a point of interest in my university education. Machine learning is a wide and interesting field which actively creates solutions for most problems, and has also helped further modern systems such as recommender systems which make use of algorithms such as linear regressions, k-means algorithms and so on. I am looking to implement these algorithms in my project, a heart disease prediction system which takes data from a dataset, splits this data into training and testing sets, and determines whether a person has heart disease or not when input that was previously not in the dataset is introduced.

This project has offered me a challenge to conduct research into heart disease and how it affects the society, and finding the solution through using machine learning has proved an interesting learning curve for me. This has improved my mastery of the machine learning algorithms, and to pursue more knowledge in this field.



## 1.4 Objectives

The main objective of this project is to predict whether the patient has heart disease or not by using this heart disease prediction system. First we have to remove the outlier from the chosen dataset. These outliers will differ the prediction a lot so we have to remove these outliers for accurate prediction by the preferred algorithms which we have chosen. After that with the highest accuracy percentage we will choose one algorithm from that and use that algorithm in the testing process to test if the algorithm is good or not. The final objective will be to determine if the patient has heart disease or not.

The proposed system should be able to achieve the following **objectives**:

1. Accept various parameters as input. These parameters include the age of the patient, their blood pressure, etc.
2. Give output to indicate whether the patient could have heart disease or not.
3. Split the data from the provided dataset into two: training data and test data.
4. Provide the accuracy of the model trained.
5. Present results on an interface.

## 1.5 Software Development Life Cycle(SDLC) Model

In this section, I talk about the waterfall model, which is my preferred project management model. The SDLC(Software Development Life Cycle) is popular and for good reason: it is exhaustive and ensures a phase is complete before moving to the next one.

The Software Development Life Cycle (SDLC) is a method that specifies the many steps required in software development in order to create a high-quality result. SDLC stages encompass the whole life cycle of a software product, from conception through retirement. Following the SDLC process ensures that software is developed in a methodical and disciplined manner. Under SDLC, I implemented the Waterfall Model.

Below are the phases and their descriptions:

### **Requirement Analysis**

First, requirements are gathered and analyzed. Only once the requirement has been frozen can the System Design begin. The SRS document developed in this case is the result of the Requirement phase and serves as an input to the System Design phase.

### **System Analysis**

Documents that serve as input for the following step, i.e. implementation and coding, are prepared in System Design Software Architecture and Design.

### **Implementation**

Coding is completed in the Implementation phase, and the software created serves as the basis for the following phase, testing.

### **Testing**

The created code is rigorously checked throughout the testing process to uncover software faults. Defects are recorded in a defect tracking tool and retested once they've been rectified. Bug logging, retesting, and regression testing continue until the program is ready to go live.

### **Deployment**

After the client has granted his or her approval, the produced code is placed into production in the Deployment phase.

### **Maintenance**

Any difficulties that arise in the production environment are dealt with by the developers who are in charge of maintenance.

### **Advantages**

1. The waterfall model is a basic, easy-to-understand approach in which all phases are completed one after the other.

2. Each phase's deliverables are carefully defined, which reduces project complexity and makes it easier to manage.

**Disadvantages**

1. The waterfall paradigm is time-consuming and cannot be employed in short-term projects since a new phase cannot begin until the previous one is completed.
2. The waterfall model cannot be used for projects with ambiguous requirements or where the requirements change frequently because it expects the requirements to be clear during the requirement gathering and analysis phase, and any changes made later would result in higher costs because changes would be required in all phases.

## 1.6 Scope and Limitations

### 1.6.1 Scope

Heart Disease Prediction System is a computer based disease prediction system which takes information about a patient and detects whether the patient has heart disease or not. The system is mainly focused for:

- Medical Center
- Hospitals
- Patients who have sufficient information that is needed to the system.

The user is provided with an input field for the various data input.

### 1.6.2 Limitations

There will be practice for making the system more reliable and flexible in various perspectives. In spite of that there will be the following limitations.

1. Machine Learning approach cannot assure 100% accuracy
2. Uses of various language as input are not feasible at the moment (only English)
3. System is not fully automated.
4. System need a lot of data to predict the result

## 1.7 Summary

This documentation is divided into five Chapters:

### **Chapter 1: Introduction**

In this chapter the problem at hand is discussed in detail and a proposed solution presented.

### **Chapter 2: Literature Review**

In this section, previous works and literature done on the topic are described. We also look at what people have done on the task and area under study.

### **Chapter 3: System Analysis**

Here I collect and interpret facts, identify the problems, and decompose the system into its components. It is in this chapter that requirements are collected and feasibility studies done.

### **Chapter 4: System Design**

In this chapter, the focus is to describe how the system works as clearly as possible through the use of diagrams.

### **Chapter 5: Implementation**

In this chapter, various sub chapters describe analysis of the dataset, and the implementation of the system itself.

### **Chapter 6: Conclusion**

Here we write the results found, what I have learnt about the methods and system I designed and used. I also discuss what I expect as a future extension of what I did.

## 2.0 Literature Review/Related Work

This section studies previous literature work that has been done on Heart Disease Prediction using Machine Learning.

### 2.1 Previous Works

K. Polaraju et al proposed Prediction of Heart Disease using Multiple Regression Model and it proves that Multiple Linear Regression is appropriate for predicting heart disease chance. The work is performed using a training data set consisting of 300 instances with 13 different attributes. The data set is divided into two parts that is 70% of the data are used for training and 30% used for testing. Based on the results, it is clear that the classification accuracy of the Regression algorithm is better compared to other algorithms.

Marjia et al developed heart disease prediction using KStar, j48, SMO, and Bayes Net and Multilayer perceptron using WEKA software. Based on performance from different factors SMO and Bayes Net achieve optimum performance than KStar, Multilayer perceptron and J48 techniques using k-fold cross validation. The accuracy performances achieved by those algorithms are still not satisfactory. Therefore, the accuracy's performance is improved more to give better decisions to diagnose disease.

S. Seema et al focuses on techniques that can predict chronic disease by mining the data contained in historical health records using Naïve Bayes, Decision tree, Support Vector Machine(SVM) and Artificial Neural Network(ANN). A comparative study is performed on classifiers to measure the better performance on an accurate rate. From this experiment, SVM gives the highest accuracy rate, whereas for diabetes Naïve Bayes gives the highest accuracy.

Chala Beyene et al recommended Prediction and Analysis of the occurrence of Heart Disease Using Data Mining Techniques. The main objective is to predict the occurrence of heart disease for early automatic diagnosis of the disease within a short time. The proposed methodology is also critical in healthcare organizations with experts that have no more knowledge and skill. It uses different medical attributes such as blood sugar and heart rate, age, sex are some of the attributes are included to identify if the person has heart disease or not. Analyses of the dataset are computed using WEKA software.

R. Sharmila et al proposed to use non- non-linear classification algorithms for heart disease prediction. It is proposed to use big data tools such as Hadoop Distributed File System (HDFS), Mapreduce along with SVM for prediction of heart disease with optimized attribute set. This work made an investigation on the use of different data mining techniques for predicting heart diseases. It suggests using HDFS for storing large data in different nodes and executing the prediction algorithm using SVM in more than one node simultaneously using SVM. SVM is used in parallel fashion which yields better computation time than sequential SVM.

S.Prabhavathi et al proposed a Decision tree based Neural Fuzzy System (DNFS) technique to analyze and predict various heart diseases. This paper reviews the research on heart disease diagnosis. DNFS stands for Decision tree based Neural Fuzzy System. This research is to create an intelligent and cost effective system, and also to improve the performance of the existing system. Specifically in this paper, data mining techniques are used to enhance heart disease prediction. The result of this research shows that the SVM and neural networks result in a highly positive manner to predict heart disease. Still the data mining techniques are not encouraging for heart disease prediction.

Sairabi H.Mujawar et al, used k-means and naïve bayes to predict heart disease. This paper is to build the system using a historical heart database that gives diagnosis. 13 attributes have been considered for building the system. To extract knowledge from databases, data mining techniques such as clustering, classification methods can be used. 13 attributes with a total of 300 records were used from the Cleveland Heart Database. This model is to predict whether the patient has heart disease or not based on the values of 13 attributes.

Boshra Brahmi et al, developed different data mining techniques to evaluate the prediction and diagnosis of heart disease. The main objective is to evaluate the different classification techniques such as J48, Decision Tree, KNN, SMO and Naïve Bayes. After this, evaluating some performance in measures of accuracy, precision, sensitivity, specificity are evaluated and compared. J48 and decision trees give the best technique for heart disease prediction.

Jaymin Patel et al, suggested data mining techniques and machine learning to predict heart disease. There are two objectives to predict the heart system.

1. This system does not assume any knowledge prior about the patient's records.
2. The system which is chosen must be scalar to run against the large number of records. This system can be implemented using WEKA software. For testing, the classification tools and explorer mode of WEKA are used.

In Hlaudi Daniel Masethe's paper Prediction of Heart Disease Using Classification Algorithms (Hlaudi Daniel Masethe, 2014), an experiment was carried out to predict heart attacks and a comparison was made to discover the best technique of prediction. This can be a useful tool for doctors to forecast dangerous patients in their practice and provide appropriate recommendations. The prediction accuracy determined by the J48, REPTREE, and SIMPLE CART algorithms shows that the parameters utilized are solid markers for detecting the existence of heart disorders, although this research has the drawback of not being able to forecast heart disease in real time.

Two classifiers, KNN and ID3, are employed in this study by Thomas, J. and Princy, R.T. (Thomas, 2016) Human Heart Disease Prediction System using Data Mining Techniques, where the KNN approach surpassed the ID3 approach in terms of accuracy with an accuracy of 80.6 percent. However, this approach is limited by the selection of fake qualities and lacks real-time prediction.

The work of K. Sudhakar et al. (Sudhakar, 2014) Study of Heart Disease Prediction using Data Mining presents the different techniques that are deployed in the recent years for calculating the prediction rate in heart disease. These techniques include-ANN, BN, Decision Trees and Classification Algorithms but this work has the limitation of not predicting heart disease in real-time.

In this work of Ebenezer Obaloluwa Olaniyi and Oyebade Kayode Oyedotun (Olaniyi, Oyedotun and Adnan, 2015) study of Heart disease diagnosis using neural networks arbitration, two techniques as SVM and ANN are used wherein support vector machine is the best algorithm for diagnosis of heart disease which gave accuracy rate of 87.5%, with high value of sensitivity and specificity but prediction of heart disease is static and no real-time prediction.

The work of Joshi, S. and Nair, M.K (Joshi, 2015) Prediction of Heart Disease Using Classification Based Data Mining Techniques, three techniques as Decision Trees, Naïve Bayes and K Nearest Neighbor Wherein Decision Tree is the most accurate with 92.3% accuracy to build the model. In this work no real time prediction has been done.

## **2.2 Machine Learning Algorithms Employed**

Having come across multiple algorithms already implemented in previous works described above, I will describe them here in this section.

### **Supervised and Unsupervised Learning Algorithms**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. The supervised machine learning algorithms are those algorithms which need external assistance. The input dataset is divided into train and test dataset. The train dataset has an output variable which needs to be predicted or classified. All algorithms learn some kind of patterns from the training dataset and apply them to the test dataset for prediction or classification. The workflow of supervised machine learning algorithms is given in fig below. The most used supervised machine learning algorithms are discussed below.

#### **2.2.1 Logistic Regression**

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign, Has Heart disease or does not have heart disease. Logistic regression transforms its output using a complex cost function defined as the 'Sigmoid function' to return a probability value.

#### **Cost Function**

The cost function represents optimization objectives i.e. we create a cost function and minimize it so that we can develop an accurate model with minimum error.



## What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoids to map predictions to probabilities. Below is the formula for a sigmoid curve function.

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

## Advantages of Logistic Regression

1. Logistic regression is one of the most basic machine learning algorithms. It is simple to build and, in some situations, delivers excellent training efficiency. Because of these factors, training a model with this technique does not necessitate a lot of computing resources.
2. The inferences regarding the relevance of each characteristic are based on the expected parameters (trained weights). The association's orientation, positive or negative, is also specified. As a result, logistic regression may be used to determine the connection between the characteristics.
3. Unlike decision trees or support vector machines, this approach allows models to be quickly modified to incorporate new data. Stochastic gradient descent can be used to update the data.
4. Along with classification findings, Logistic Regression produces well-calibrated probability. This is a benefit over models that just provide results for the final classification. We can deduce which training examples are more accurate for the specified issue if one has a 95 percent probability for a class and another has a 55 percent probability for the same class.
5. Logistic regression is less prone to over-fitting in a low-dimensional dataset with a sufficient number of training instances.
6. Because it is relatively quick and straightforward to develop, logistic regression is frequently used as a benchmark model to test performance rather than starting with a sophisticated model right away.
7. When the dataset comprises linearly separable characteristics, Logistic Regression appears to be quite efficient.

## Disadvantages of Logistic Regression

1. Complex correlations are difficult to represent with logistic regression. This approach is readily outperformed by more powerful and complicated algorithms such as Neural Networks.
2. Independent variables are the characteristics of the training. Moderate or no multicollinearity between independent variables is required for logistic regression. This indicates that if the correlation between two independent variables is high, just one of them should be employed. Repetition of information may result in incorrect parameter (weight) training while minimizing the cost function. Dimensionality reduction methods can be used to eliminate multicollinearity.
3. The independent and dependent variables in Linear Regression should be connected linearly. However, in order to use Logistic Regression, independent variables must be linearly connected to the log chances ( $\log(p/(1-p))$ ).
4. Only crucial and relevant features should be employed to construct a model; otherwise, the model's probabilistic predictions may be wrong, and its predictive value may suffer.

5. Because this technique is sensitive to outliers, the existence of data values that differ from the anticipated range in the dataset may result in inaccurate findings.
6. Logistic regression needs a big dataset as well as enough training samples for all of the categories to be identified.

### **2.2.2 Decision Tree**

Recall that supervised learning can either be based on a classification or regression algorithm, and decision tree algorithm can be used as both although it is mainly used for classification. The algorithm emulates a tree, and it sorts attributes through groupings based on data values. Just like a conventional tree, the algorithm has branches and nodes with nodes representing variable groups for classification and branches, assuming the values that the attribute can take as part of the class. The pseudocode illustrating the decision tree algorithm is as shown below. In the algorithm,  $D$  is the dataset, while  $x$  and  $y$  are the input and target variables, respectively.

**Protocol DT Inducer ( $D, x, y$ )**

1. **T=Tree Growing ( $D,x,y$ )**
2. Return Tree Pruning ( $D, T$ )

**Method Tree Growing ( $D, x, y$ )**

1. Create a tree  $T$
2. **if** at least one of the Stopping Criteria is satisfied **then**;
3. label the root node as a leaf with the most frequent value of  $y$  in  $D$  as the correct class.
4. **else**;
5. Establish a discrete function  $f(x)$  of the input variable so that splitting  $D$  according to the function's outcomes produces the best splitting metric
6. **if** the best metric is greater or equal to the threshold **then**;
7. Mark the root node in  $T$  as  $f(x)$
8. **for** each outcome of  $f(x)$  at the node **do**;
9. Subtree =TreeGrowing  $\delta f(x)=t1,D,x,y$
10. Connect the root of  $T$  to Subtree and label the edge  $t1$
11. **end for**
12. **else**
13. Label the root node  $T$  for a leaf with the frequent value of  $y$  in  $D$  as the assigned class
14. **end i**
15. **end if**
16. Return  $T$

**Protocol Tree Pruning ( $D, T, y$ )**

1. **repeat**
2. Select a node  $t$  in  $T$  to maximally improve pruning evaluation procedure
3. **if**  $t \neq 0$  **then**;
4. **T=pruned ( $T,t$ )**
5. **end if**
6. **until**  $t=0$

## 7. Return T

As illustrated in the pseudocode, Decision Tree achieves classification in three distinct steps. Firstly, the algorithm induces both tree growing and tree pruning functionalities. Secondly, it grows the tree by assigning each data value to a class based on the value of the target variable that is the most common one at the instance of iteration. The final step deals with pruning the grown tree to optimize the performance of the resultant model. Most of the reviewed studies involved application of decision trees for different applications, although most involved classification cancer and lung cancer studies, clinical medicine especially diagnosis of conditions based on historical data.

### **Advantages of Decision Tree Algorithm**

1. Decision trees need less work for data preparation during pre-processing than other methods.
2. A decision tree does not need data normalization.
3. A decision tree does not need data scalability.
4. In addition, missing values in the data have no significant impact on the decision tree-building process.
5. A decision tree model is simple to understand and communicate to technical teams and stakeholders.

### **Disadvantages of Decision Tree Algorithm**

1. A slight change in the data can result in a substantial change in the decision tree's structure, resulting in instability.
2. When compared to other algorithms, a decision tree's calculation might get rather complicated at times.
3. The training period for a decision tree is often longer.
4. Because of the intricacy and time required, decision tree training is relatively costly.
5. When it comes to using regression and predicting continuous values, the Decision Tree method falls short.

### **2.2.3 Random Forest Classifier**

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.

The fundamental concept behind random forest is the wisdom of crowds. In data science, the reason that the random forest model works so well is:

1. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.
2. The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions.

The reason for this effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. The prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

### **Ensuring that the Models Diversify Each Other**

So how does random forest ensure that the behavior of each individual tree is not too correlated with the behavior of any of the other trees in the model? It uses the following two methods:

#### **Bagging**

Decision trees are very sensitive to the data they are trained on. Small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

Notice that with bagging we are not subsetting the training data into smaller chunks and training each tree on a different chunk. Rather, if we have a sample of size  $N$ , we are still feeding each tree a training set of size  $N$  (unless specified otherwise). But instead of the original training data, we take a random sample of size  $N$  with replacement. For example, if our training data was  $[1, 2, 3, 4, 5, 6]$  then we might give one of our trees the following list  $[1, 2, 2, 3, 6, 6]$ . Notice that both lists are of length six and that "2" and "6" are both repeated in the randomly selected training data we give to our tree (because we sample with replacement).

#### **Feature Randomness**

In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs. those in the right node. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification.

### **Advantages of Random Forest Classifier**

1. Random Forest is a method that employs Ensemble Learning and is based on the bagging algorithm. It grows as many trees as possible on a subset of the data and then merges the results of all of the trees. As a result, the overfitting problem in decision trees is reduced, as is the variance, which increases accuracy.
2. Random Forest may be used to address issues in both classification and regression.
3. Both categorical and continuous variables function well with Random Forest.
4. Missing values may be handled automatically using Random Forest.

5. There's no need to scale the features: Random Forest does not require feature scaling (standardization and normalization) since it employs a rule-based method rather than distance computation.
6. Effectively handles nonlinear parameters: Unlike curve-based algorithms, non linear parameters have no effect on the performance of a Random Forest. As a result, if the independent variables are very nonlinear, Random Forest may outperform conventional curve-based methods.
7. Missing values may be handled automatically using Random Forest.
8. Outliers are frequently tolerated well by Random Forest, which can manage them automatically.
9. The Random Forest method has a high level of consistency. Even if a new data point is added to the dataset, the overall method remains unaffected since the new data may have an influence on one tree, but it is extremely unlikely to have an impact on all trees.
10. Random Forest is less affected by noise than other methods.

### **Disadvantages of Random Forest Classifier**

1. **Complexity:** Random Forest generates a large number of trees (as opposed to a single tree in a decision tree) and then mixes their results. In the Python sklearn package, it builds 100 trees by default. This approach necessitates a significant increase in processing power and resources. On the other hand, a decision tree is straightforward and does not need a large amount of computer power.
2. **Longer Training Period:** Random Forests take significantly longer to train than decision trees since they create several trees (rather than just one) and make decisions based on the majority of votes.

### **2.2.4 Support Vector Machine Algorithm**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. Its principle is to create the margins such that the distance between each class and the nearest margin is maximized and in effect leading to the minimum possible classification error. The margins are defined as the distance between two supporting vectors separated by a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called **support vectors**, and hence the algorithm is termed as Support Vector Machine.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a

two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

### **2.2.5 K Nearest Neighbors Classifier**

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. In most cases most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of distance, proximity, or closeness with some simple mathematics— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the Euclidean distance is a popular and familiar choice.

#### **The KNN Algorithm**

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
  - Calculate the distance between the query example and the current example from the data.
  - Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

#### **Choosing the right value for K**

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions

(up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.

3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

### **2.2.6 XGBoost Classifier**

XGBoost is an open source library providing a high-performance implementation of gradient boosted decision trees. An underlying C++ codebase combined with a Python interface sitting on top makes for an extremely powerful yet easy to implement package.

The performance of XGBoost is really high — it's become the go-to library for many machine learning enthusiasts. Its gradient boosting implementation is really good and there's only more to come as the library continues to garner praise.

#### **Boosting Trees**

With a regular machine learning model, like a decision tree, we'd simply train a single model on our dataset and use that for prediction. We might play around with the parameters for a bit or augment the data, but in the end we are still using a single model. Even if we build an ensemble, all of the models are trained and applied to our data separately. Boosting, on the other hand, takes a more iterative approach. It's still technically an ensemble technique in that many models are combined together to perform the final one, but takes a more clever approach. Rather than training all of the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the previous ones. Models are added sequentially until no further improvements can be made. The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes, which is what we are trying to prevent.

#### **Advantages of XGBoost Classifier**

1. There isn't as much feature engineering necessary (No need for scaling, normalizing data, can also handle missing values well)
2. It is possible to determine the significance of a feature (it output importance of each feature, can be used for feature selection)
3. Quick to comprehend
4. Outliers have a negligible effect.
5. Large datasets are easily handled.
6. Good speed of execution
7. Model performance is excellent (wins most of the Kaggle competitions)
8. Overfitting is less likely.

#### **Disadvantages XGBoost Classifier**

1. Interpretation is difficult, and visualizing is difficult.

2. If parameters aren't calibrated appropriately, overfitting is a possibility.
3. Because there are so many hyperparameters, tuning is more difficult.

## **2.3 Feature Selection Methods**

When creating a predictive model, feature selection is the process of minimizing the number of input variables. The number of input variables should be reduced to lower the computational cost of modeling and, in some situations, to increase the model's performance.

The relationship between each input variable and the goal variable is evaluated using statistics, and the input variables having the strongest link with the target variable are selected. Although the choice of statistical measures depends on the data type of both the input and output variables, these approaches can be quick and successful. As a result, selecting an acceptable statistical measure for a dataset while doing filter-based feature selection might be difficult.

In order to forecast the target variable, feature selection approaches are used to minimize the number of input variables to those that are thought to be most relevant to a model. Some predictive modeling issues contain a huge number of variables, which can slow down model construction and training and necessitate a lot of system memory. Additionally, certain models' performance might suffer when input variables that are unrelated to the target variable are included.

Feature selection approaches may be divided into two categories: supervised and unsupervised. Unsupervised feature selection strategies disregard the target variable, such as correlation-based methods for removing redundant variables, whereas supervised feature selection techniques employ the target variable, such as methods for removing irrelevant variables.

### **2.3.1 Filter Methods**

Filter approaches, rather than cross-validation performance, focus on the inherent qualities of the features as assessed by univariate statistics. These techniques are both quicker and less expensive to compute than wrapper methods. Filter techniques are computationally economical when dealing with high-dimensional data.

Various filter methods are discussed below:

#### **2.3.1.1 Feature Selection by Information Gain**

The reduction in entropy caused by a dataset modification is calculated as information gain. It may be used to choose features by assessing each variable's information gain in relation to the target variable.



### 2.3.1.2 Feature Selection by Chi-square Test

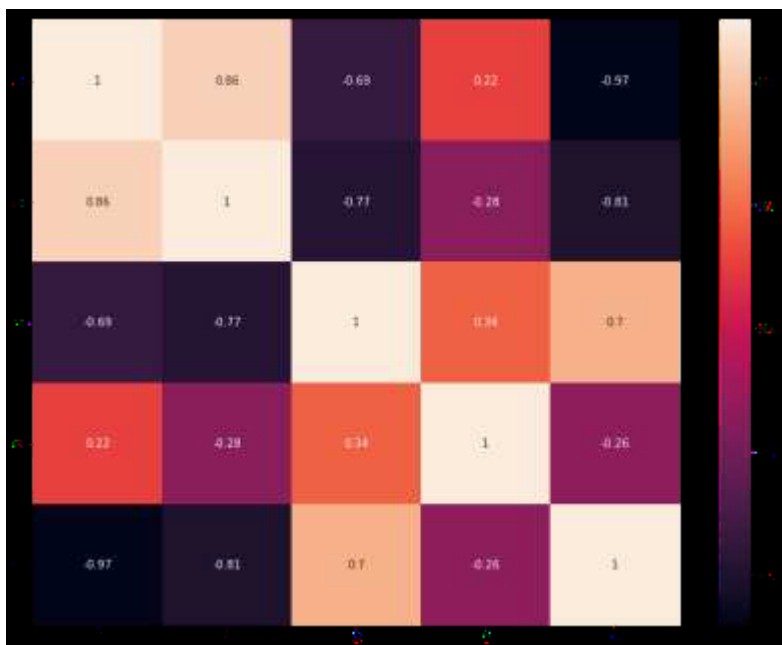
For categorical characteristics in a dataset, the Chi-square test is utilized. We calculate the Chi-square between each feature and the target and choose the features with the highest Chi-square scores. The following characteristics must be followed in order to appropriately use the chi-squared to assess the relationship between various features in the dataset and the target variable: the variables must be categorical, sampled independently, and values must have an anticipated frequency greater than 5.

### 2.3.1.3 Feature Selection by Fisher's Score

One of the most extensively used supervised feature selection approaches is Fisher score. The method we'll employ returns the variables' rankings in decreasing order depending on the fisher's score. The variables can then be chosen according to the circumstances.

### 2.3.1.4 Feature Selection by Correlation Matrix

The correlation coefficient ranges from -1 to 1 and describes how the qualities are connected to one another. A number around 0 indicates a lesser connection (an exact 0 indicates no association). A number around 0 indicates a lower association (exact 0 implying no correlation). A negative correlation is greater if the value is closer to -1. The figure below illustrates a basic correlation heat map example.



### **2.3.1.5 Feature Selection by Variance Threshold**

A basic baseline technique to feature selection is the variance threshold. It eliminates any characteristics whose variance falls below a certain level. It removes all zero-variance features by default, that is, features that have the same value across all samples. We think that features with a greater variance contain more important information, but keep in mind that we aren't accounting for the link between feature variables or between feature and goal variables, which is one of the downsides of filter approaches.

### **2.3.1.6 Feature Selection by Mean Absolute Difference (MAD)**

'The mean absolute difference (MAD) is a formula that calculates the absolute difference between two values. The lack of the square in MAD measurements is the major distinction between variance and MAD measures. The MAD is a scale variation, much like the variance.' [1] As a result, the greater the MAD, the greater the discriminating power.

## **2.3.2 Wrapper Methods**

Wrappers need a way to explore the space for all potential feature subsets and measure their quality by learning and evaluating a classifier with that subset of features. The feature selection procedure is based on a machine learning technique that we are attempting to apply to a particular dataset. It uses a greedy search method, assessing all potential feature combinations against the evaluation criterion. Wrapper approaches are frequently more accurate predictors than filter methods. We discuss some of the techniques under wrapper methods below:

### **2.3.2.1 Feature Selection By Forward Feature Selection**

This is an iterative strategy in which we begin with the variable that performs the best against the target. After that, we choose another variable that, when combined with the first, produces the greatest results. This method is repeated until the predetermined criterion is met.

### **2.3.2.2 Feature Selection By Backward Feature Elimination**

The Forward Feature Selection approach is the polar opposite of this one. We'll start with all of the features and develop a model from there. Then we choose the model variable that produces the best evaluation measure value. This cycle is repeated until the predetermined criterion is met.

### **2.3.2.3 Feature Selection By Exhaustive Feature Selection**

This is the most reliable approach of feature selection we've seen so far. Each feature subset is evaluated by a brute-force method. This implies it tries every conceivable combination of the variables and delivers the subset with the highest performance.

### **2.3.2.4 Feature Selection By Recursive Feature Elimination**

The purpose of recursive feature elimination (RFE) is to pick features by iteratively examining smaller and smaller sets of features, given an external estimator that gives weights to features (e.g., the coefficients of a linear model). The estimator is first trained on the

original set of features, and the significance of each feature is determined using either the `coef_` or `feature importances_` attributes. Then, given the existing collection of features, the least significant features are trimmed. On the trimmed set, this approach is continued recursively until the required number of features to pick is attained.

### **2.3.3 Embedded Methods**

These approaches combine the advantages of both the wrapper and filter methods by including feature interactions while keeping a low computational cost. Embedded approaches are iterative in the sense that they look after each iteration of the model training process and carefully extract the characteristics that make the biggest difference in the training for that iteration. The techniques under embedded methods are discussed below:

#### **2.3.3.1 Feature Selection By LASSO Regularization**

Regularization entails applying a penalty to the machine learning model's various parameters in order to decrease the model's freedom, i.e. to avoid overfitting. The penalty is imposed on each of the predictors' coefficients in linear model regularization. Lasso or L1 has the feature of being able to reduce part of the coefficients to zero, which distinguishes it from other methods of regularization. As a result, the model's functionality can be disabled.

#### **2.3.3.2 Feature Selection By Random Forest Importance**

Random Forests are a type of Bagging Algorithm that aggregates a set of decision trees. Random forests' tree-based tactics are naturally ranked by how effectively they increase node purity, or, in other words, how well they reduce impurity (Gini impurity) over all trees. The nodes with the largest drop in impurity are found at the beginning of the trees, while the nodes with the least decrease in impurity are found at the conclusion. We may produce a subset of the most essential characteristics by trimming trees below a certain node.

## **2.4 Performance Measures of Machine Learning Algorithms**

### **2.4.1 Confusion Matrix**

A confusion matrix is a table that is frequently used to describe a classification model's performance on a set of test data for which the true values are known. It enables the visualization of an algorithm's performance. The figure below illustrates the concept of a confusion matrix to further understand it.

		Predicted class	
		Absence of HD	Presence of HD
Actual class	Absence of HD	TP	FN
	Presence of HD	FP	TN

### True Positive(TP)

These are successfully predicted positive values, indicating that the value of the real class is yes, as well as the value of the anticipated class. For example, when the actual class value indicates that this passenger survived and the anticipated class also indicates that this passenger survived.

### True Negative(TN)

These are accurately predicted negative values, indicating that the value of the real class is no and the value of the projected class is 'no' as well. For example, if the real class reports that this passenger did not survive and the forecast class confirms this.

### False positive(FP)

When the expected class is yes and the actual class is no. For example, if the real class says this passenger did not survive but the forecast class predicts this passenger will.

### False Negative(FN)

When the actual class is positive while the anticipated class is negative. For example, if the passenger's actual class value indicates that he or she survived yet the anticipated class suggests that the passenger would die.

Different performance metrics are computed based on the TP, TN, FP, and FN as given below:

### Accuracy

The simplest basic performance metric is accuracy, which is just the ratio of accurately predicted observations to total observations.

### Precision

The ratio of accurately predicted positive observations to the total expected positive observations is known as precision.

**Recall (Sensitivity)**

The ratio of accurately predicted positive observations to all observations in the actual class is known as recall.

**F1 score**

F1 Score is the weighted average of Precision and Recall

**2.5 Open source tools for data mining**

This section describes the open source tools which are used for data mining, includes WEKA, TANAGRA, Rapidminer, Orange.

**2.5.1 WEKA**

WEKA is a data mining system created by the University of Waikato in New Zealand that uses the JAVA programming language to execute data mining methods. WEKA is a cutting-edge research and development laboratory for machine learning techniques and their application to real-world data mining challenges. It's a collection of data mining-related machine learning algorithms. The methods are immediately applied to a dataset. WEKA offers visualization tools as well as algorithms for data pre-processing, classification, regression, clustering, and association rules. This package may also be used to create new machine learning methods. WEKA is a piece of open source software. WEKA's standard data file format is ARFF, which uses specific tags to describe distinct aspects of the data file (Kaur and Singh, 2014).

**2.5.2 TANAGRA**

Tanagra is a free data mining program designed for academic and research use. It presents a variety of data mining techniques in the areas of exploratory data analysis, statistical learning, machine learning, and database management. Tanagra is an open source project, which means that any researcher can access the source code and modify it as long as he agrees to the software distribution agreement. The major goal of the Tanagra project is to provide researchers and students with simple data mining software that adheres to current software development standards and can be used to analyze actual or fake data (Kaur and Singh, 2014).

**2.5.3 RapidMiner**

RapidMiner is without a doubt the most popular open source data mining solution in the world. It is offered as a standalone data analysis program as well as a data mining engine for integration into custom solutions. RapidMiner's tens of thousands of applications in more than 40 countries provide their users a competitive advantage (Kaur and Singh, 2014).

### 2.5.4 Orange

Orange is a free data visualization and analysis tool that is suitable for both beginners and specialists. Components for machine learning are utilized in data mining via visual programming in Python scripting. Bioinformatics and text mining add-ons are available. This is jam-packed with data analytics features (Kaur and Singh, 2014).

## 2.6 Examples of Prediction Systems

Here we talk about existing systems that I illustrate what I am trying to achieve. The following is a list of the systems I came across;

1. Predictive Age Progression Software
2. Automated Disease Prediction System (ADPS)

### 2.6.1 Predictive Age Progression Software

University of Washington researchers have developed software that automatically generates images of a young child's face as it ages through a lifetime. The technique is the first fully automated approach for aging babies to adults that works with variable lighting, expressions and poses.

The shape and appearance of a baby's face – and variety of expressions – often change drastically by adulthood, making it hard to model and predict that change. This technique leverages the average of thousands of faces of the same age and gender, then calculates the visual changes between groups as they age to apply those changes to a new person's face.

More specifically, the software determines the average pixel arrangement from thousands of random Internet photos of faces in different age and gender brackets. An algorithm then finds correspondences between the averages from each bracket and calculates the average change in facial shape and appearance between ages. These changes are then applied to a new child's photo to predict how she or he will appear for any subsequent age up to 80.

The researchers tested their rendered images against those of 82 actual people photographed over a span of years. In an experiment asking random users to identify the correct aged photo for each example, they found that users picked the automatically rendered photos about as often as the real-life ones.

Real-life photos of children are difficult to age-progress, partly due to variable lighting, shadows, funny expressions and even milk mustaches. To compensate for these effects, the algorithm first automatically corrects for tilted faces, turned heads and inconsistent lighting, then applies the computed shape and appearance changes to the new child's face.

Perhaps the most common application of age progression work is for rendering older versions of missing children. These renderings usually are created manually by an artist who uses

photos of the child as well as family members, and editing software to account for common changes to a child's face as it ages, including vertical stretching, wrinkles and a longer nose.

But this process takes time, and it's significantly harder to produce an accurate image for children younger than age 5, when facial features more closely resemble that of a baby. The automatic age-progression software can run on a standard computer and takes about 30 seconds to generate results for one face. While this method considered gender and age, the research team hopes to incorporate other identifiers such as ethnicity, and cosmetic factors such as hair whitening and wrinkles to build a robust enough method for representing every human face.

### **2.6.2 Automated Disease Prediction System (ADPS)**

Rapid proliferation of Internet technology and handheld devices has opened up new avenues for the online healthcare system. There are instances where online medical help or healthcare advice is easier or faster to grasp than real world help. People often feel reluctant to go to the hospital or physician for minor symptoms. However, in many cases, these minor symptoms may trigger major health hazards. As online health advice is easily reachable, it can be a great head start for users. Moreover, existing online health care systems suffer from lack of reliability and accuracy. Herein, we propose an automated disease prediction system (ADPS) that relies on guided (to be described later) user input. The system takes input from the user and provides a list (topmost diseases have greater likelihood of occurrence) of probable diseases. The accuracy of ADPS has been evaluated extensively. It ensured an average of 14.35% higher accuracy in comparison with the existing solution.

## 3.0 System Analysis

### Overview

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do. System analysis includes conducting a feasibility study and requirement specification.

### 3.1 Feasibility Study

The objective of a feasibility study is to find out if a system project can be done or not and whether it is justified or not. It may also suggest possible alternative solutions. This feature is used by managers to analyze in detail a project, from the beginning to the end, to try and see whether the proposed system is worth the resources and time. Both positive and negative results are keenly analyzed here before the system is given a green light. A feasibility study should provide people who are in charge of a project with enough information to decide:

1. Whether the project can be done
2. Whether the final product will benefit its intended users
3. What are the alternatives among which a solution will be chosen?
4. Is there a preferred alternative?

After a feasibility study has been conducted, the management makes a decision to proceed or stop and drop the project, depending on the outcome of the feasibility study.

Feasibility studies can be split up into the following types:

1. Operational
2. Technical
3. Economic
4. Schedule

#### 3.1.1 Operational Feasibility

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the team handling development. The project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

In this system's case, the developer(I)is quite competent with the technical abilities to deliver the system as intended. I am well versed in python language which is the main language used throughout this project's development. Operational feasibility also makes use of the PIECES framework to help in identifying problems to be solved, and their urgency. This framework is explained further below:



**Performance**

This aspect looks at the throughput and response time of the system. Throughput is defined as the total amount of items processed/produced by the system over the defined period of time. The system possesses very high throughput as all the inputs are processed at the same time. Response time refers to the time between when the user initiates an action, and when the computer starts to display the result. The system has a really low response time as results are displayed after 3 seconds or less.

**Information**

This addresses questions about whether the system end users with timely, pertinent, accurate and usefully formatted information. The information in this system is only used during run time. This information is not stored anywhere at any time. The information is perfectly fitted in a user-friendly graphical user interface before being presented to the user. The results are very timely, utilizing the low response time of the system.

**Economy**

This addresses the questions about whether the mode of operation provides cost-effective information services to the developer, and whether reduction in costs and/or an increase in benefits is possible. This system is low budget as it is minimally made, and does not need extra platforms to run, hence production costs are very low. It does not provide any monetary benefits as it is completely free to use.

**Control**

Here, we answer questions about whether the current mode of operation offers effective controls to protect against fraud and to guarantee accuracy and security of data and information. This system, as mentioned before, does not store any data, and does not take any personal information. The users of the system remain anonymous. This provides adequate security of data as the data is discarded after every run of the system.

**Efficiency**

Under this we discuss whether the current mode of operation makes maximum use of available resources which include the processing power of a computer and time. The system is highly efficient as it has a low response time. If implemented in a high power computer, the response time is even lower. It uses up little space hence minimizing resource usage in a computer.

**Services**

We answer questions such as ‘Does current mode of operation provide reliable service? Is it flexible and expandable?’ The service is very reliable. The system is available and provides relevant results consistently. It is possible to expand the system in future works.

### 3.1.2 Technical Feasibility

This is a study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

An important issue for the development of a project is the selection of suitable languages to use. When I decided to develop the project, I went through an extensive study to determine the most suitable platform that helps in development of the project. The aspects of my study included the following factors.

1. It must have a graphical user interface.
2. Platform independent.
3. Easy to debug and maintain
4. Efficient data handling.
5. Efficient data retrieval and maintenance
6. Operating System compatible
7. Easy to install

After reviewing all this, I settled on these languages to build the HDPS. These languages are as follows: Python and Jupyter Notebooks. I am sure that by using these languages I can build an efficient & satisfied application, which will fulfill all requirements mentioned already in the proposal.

### 3.1.3 Economic Feasibility

It is evaluating the effectiveness of the proposed system by using a cost/benefit analysis method. It demonstrates the net benefit from the candidate system in terms of benefits and costs to the organization. The main aim of Economic Feasibility is to estimate the economic requirements of the proposed system before the investors put in the money for the proposed system. After conducting it, the preferable alternative which will maximize the net worth of the organization by earliest and highest return of funds along with lowest level of risk involved in developing the candidate system is chosen.

#### 3.1.3.1 Development Costs

This refers to the total cost necessary for the developer to create the system. It may include the computer to be used or purchase of an operating system to be used. It encompasses all the necessary hardware and software tools needed, and their cost.

Description of item	Quantity	Cost incurred per item	Total cost
HP Laptop	1	Ksh. 30,000	Ksh. 30,000
32 gb flash disk	1	Ksh. 1,500	Ksh. 1,500

Internet accessibility router	1	Ksh. 4,000	Ksh. 4,000
<b>Total Costs</b>			<b>Ksh. 35,500</b>

### 3.1.3.2 Operational Costs

Refers to the cost of setting up the environments necessary for development.

Description of framework	Cost incurred per item	Total cost incurred
Internet connectivity	Ksh. 1,000 per month for two months	Ksh. 2,000
Electricity connectivity to development site	Ksh. 1,000 per month for 2 months.	Ksh. 2000
Printer costs	Ksh. 1000	Ksh. 1000
<b>Grand Total</b>		<b>Ksh. 5, 000</b>

### 3.1.4 Schedule Feasibility

Schedule Feasibility is defined as the probability of a project to be completed within its scheduled time limits, by a planned due date. My main target is to achieve high level components of the application and I am sure that I will achieve my goals in the given time. I pride myself on being able to adhere to timelines strictly. Below is an exhaustive schedule set up for the project.

Event Description	Event Dates	Event Duration
Project proposal writeup	24th November-30th November 2021	1 week
System design document	1st-7th December 2021	1 week
System implementation	8th Dec 2021-8th Jan 2022	6 Weeks
System Documentation	19th-26th January 2022	1 Week

## 3.2 Requirements Specification

Requirements analysis focuses on the tasks that determine the needs or conditions to meet the new project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software and system requirements. A stakeholder is a person with an interest or concern in something, in our case,

the project at hand. The stakeholders could be doctors, patients or nurses. This section covers the following under requirements specification: functional and non-functional requirements, system requirements(hardware and software), etc.

### 3.2.1 Functional Requirements

A Functional Requirement is a description of the service that the software must offer. It describes a software system or its components. This system's functional requirements are described below in detail:

1. The user should be able to access a user interface.
2. The user should be able to key in data into the form on the user interface.
3. The user should see output after clicking the submit button.
4. The user should be able to exit the program if they so wish.

### 3.2.2 Non-Functional Requirements

**Security:** This refers to how secure the system is and tries to answer these questions: Does the system store or transmit sensitive information? What security best practices are used in the industry?

- In the case of this system made for private use, security is highly guaranteed as the data fed into the system by the user is not stored at all. The system does not run on the internet, boosting its security.

**Capacity:** This refers to the system's storage requirements.

- The system implemented should require very little space to run and can work in minimal capacity storages.

**Compatibility:** This addresses questions like the minimum hardware requirements, operating systems and their versions supported.

- My system should run on any operating system provided the Operating system has python and an IDE installed.

**Reliability and Availability:** A system is reliable when it returns correct results consistently. It is considered to have good availability when it is accessible at any given point in time and has low critical failure time under normal usage. It also refers to how the system can be used even when it has faults e.g. bugs or even hardware bugs.

**Maintainability and Manageability:** Refers to how much time does it take to fix components, and how easily an administrator can manage the system.

- The system should be easily maintained and any error should be traced in the IDE and debugged immediately.
- It should also offer easy manageability for the user, being user friendly.

**Scalability** :This answers the question ‘What are the highest workloads under which the system will still perform as expected?’ The system implemented can only be used minimally and cannot be scaled up a lot.

**Usability** :This refers to how easy it is to use the product.

- The system should offer a very good experience when being used.

**Performance**: This refers to the speed, capacity and reliability of the system. Includes:

- The processing of each request should be done within 10 seconds.
- The program should exit with 0 seconds delay if the exit button is pressed.
- The program should load up with a delay of <10 seconds when it is launched.

**Operational**: - This refers to the physical and technical environments in which the system will operate. They include:

- The system is a desktop application, it therefore should be able to run on any operating system

### 3.2.3 System(Hardware) Requirements

<b>Operating system</b>	Windows 10 64-bit, Windows 8.1 64-bit, Windows 8 64-bit, Windows 7 Service Pack 1 64-bit, Windows Vista Service Pack 2 64-bit
<b>CPU</b>	Core i5 at 2.4 GHz or above
<b>Memory</b>	2 GB RAM
<b>Free space</b>	10MB of free space

The program can basically run on minimum hardware requirements

### 3.2.4 Software Requirements

<b>Operating System</b>	Windows 7 and above, Mac OS, Linux-based distributions.
<b>Programming Languages</b>	Python 3.8
<b>IDEs</b>	VSCoDe, PyCharm

## 4.0 System Design

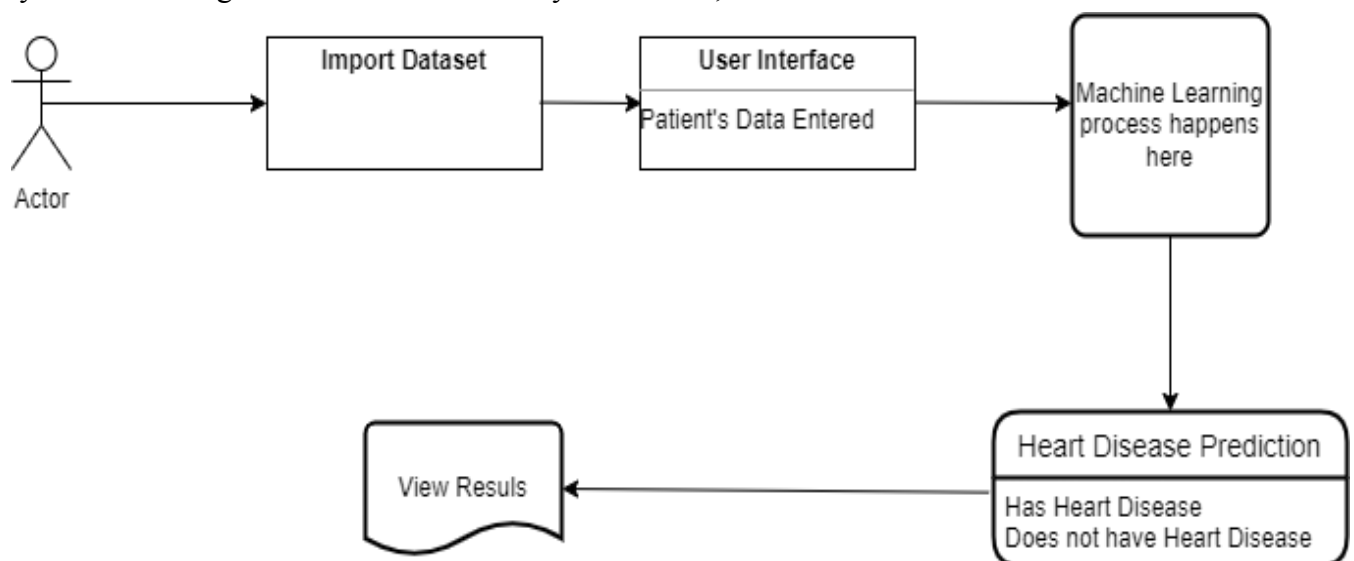
### Overview

System Design comes immediately after System Analysis where we have described requirements and conducted feasibility tests. Building on this, the requirements are now described further and made tangible through designing, which happens here. The system is visualized through various ways.

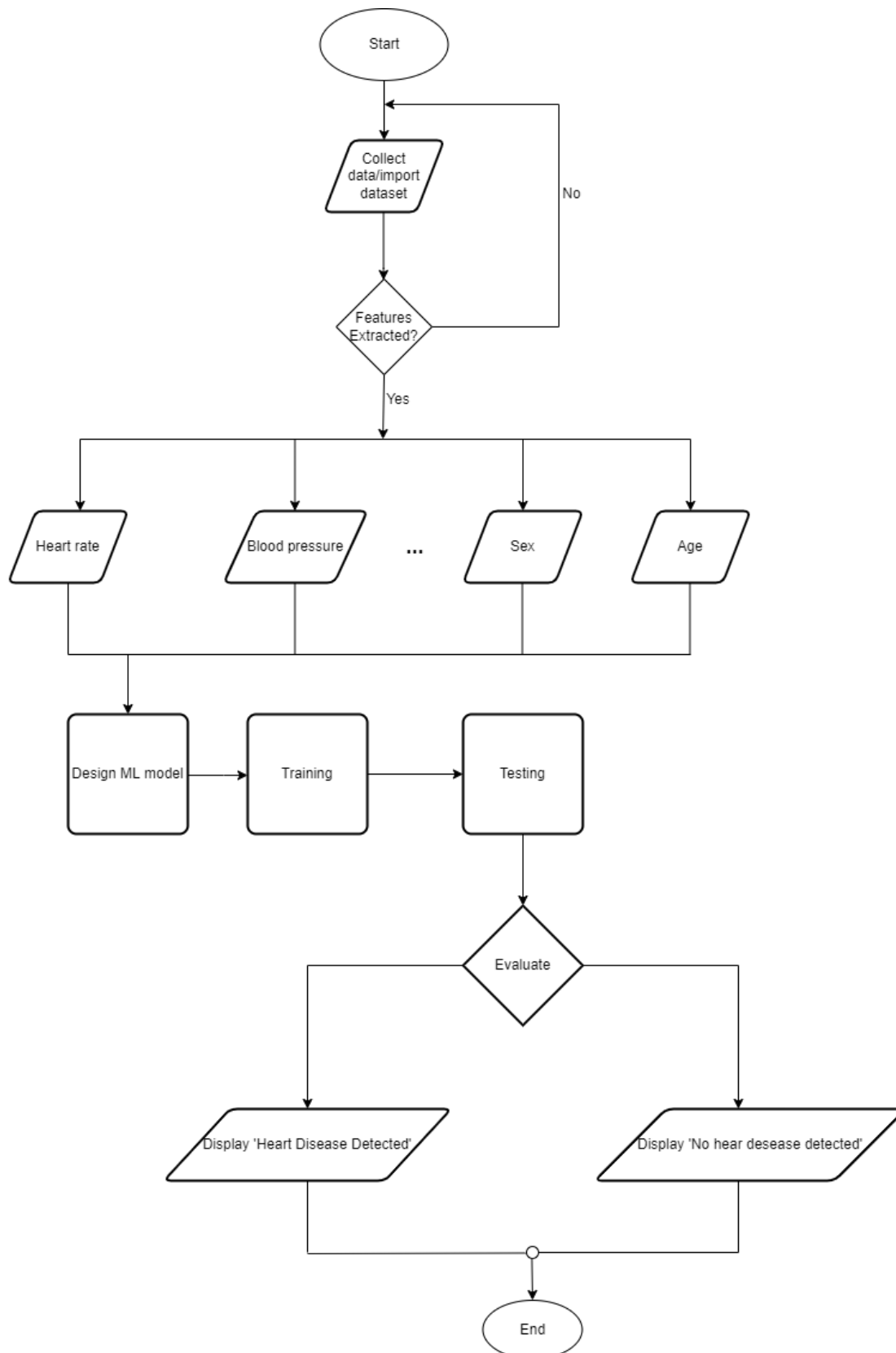
In this chapter, the focus is to describe how the system works as clearly as possible through the use of diagrams. It will cover the system architecture, a flowchart diagram, state diagram, use case diagram and a sequence diagram. These diagrams help to understand how the system works, before it is actually developed.

### 4.1 Architecture diagram

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. The image below shows how the system works, and the basic architecture.



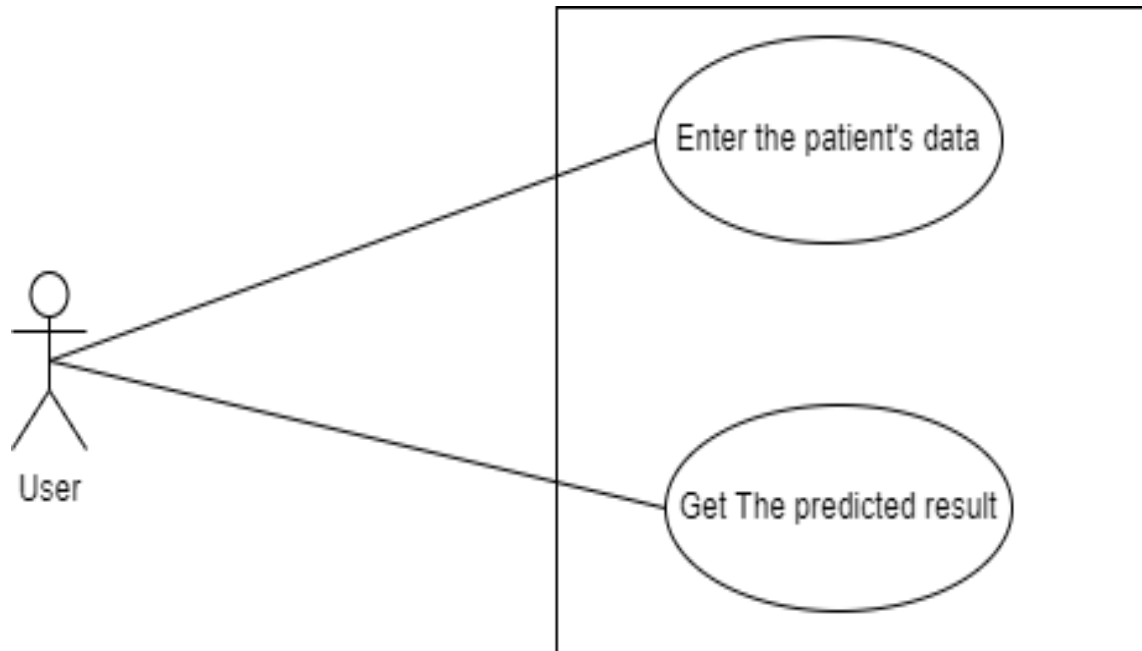
## 4.2 Flowchart Diagram



The flowchart above represents how the system works, and the processes involved throughout the process.

### **4.3 Use Case Diagram**

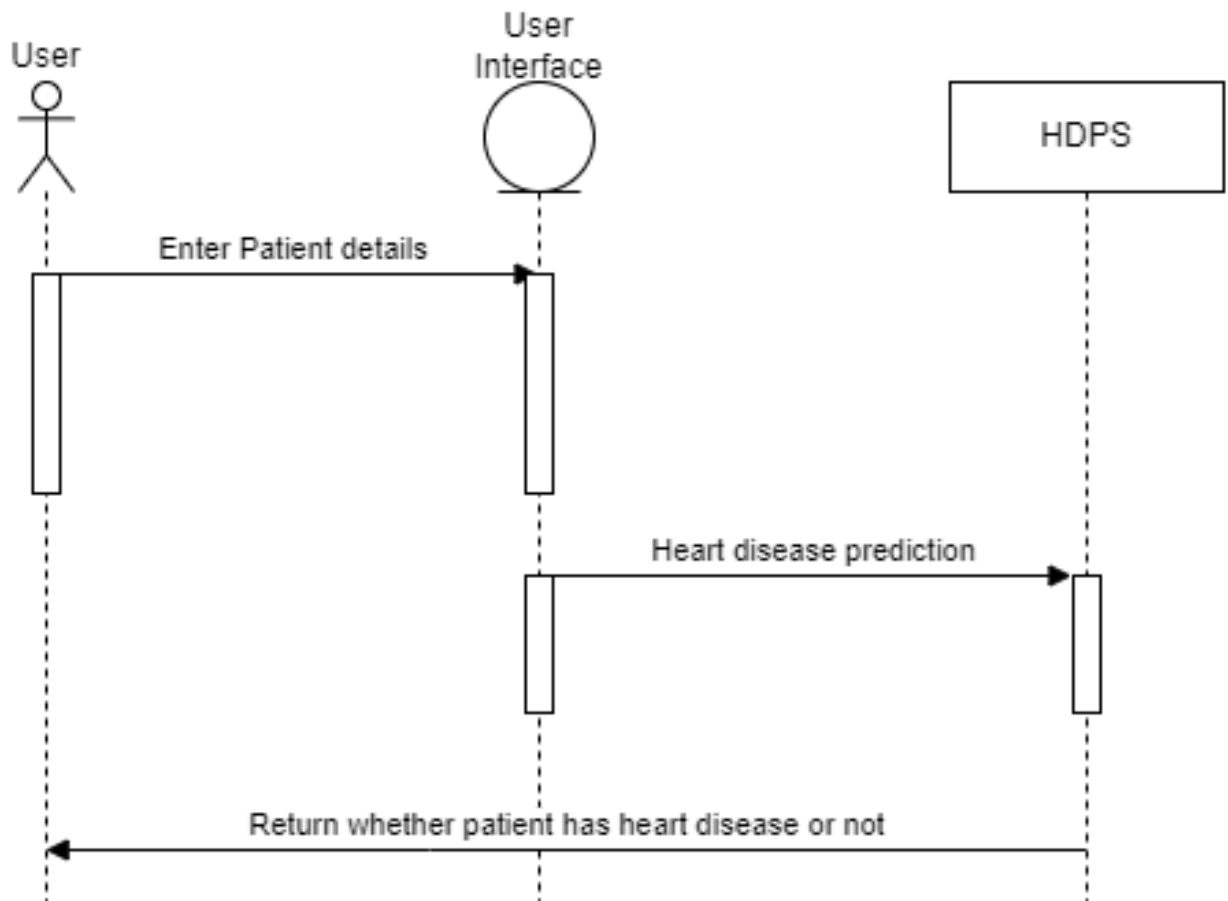
A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. In this system, there exists only one actor who inputs the data into the system. After data is fed in, after clicking a button, the system performs a learning operation after which the results are displayed on the screen.





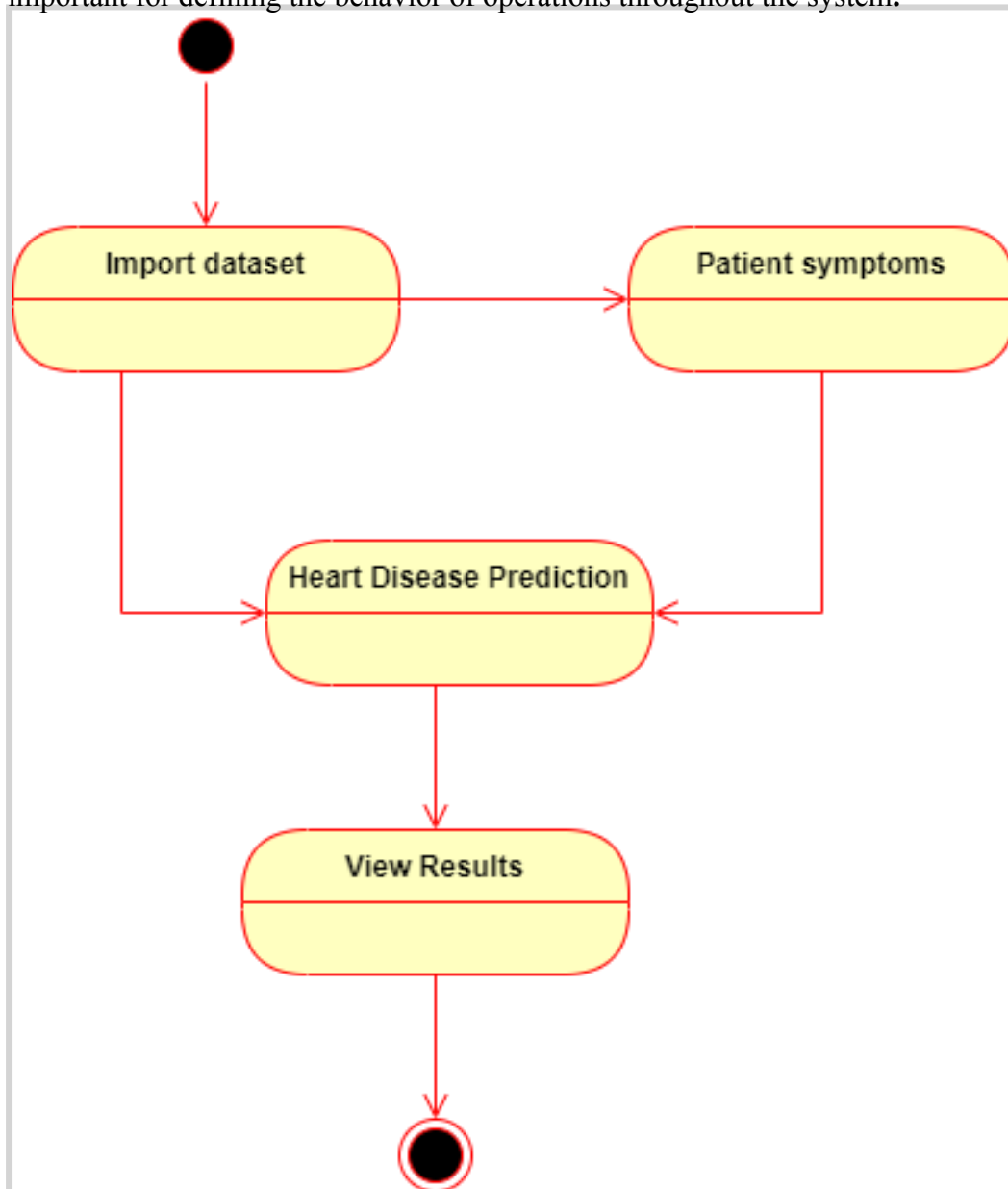
#### 4.4 Sequence Diagram

A sequence diagram, in the context of UML, represents object collaboration and is used to define event sequences between objects for a certain outcome. The sequence diagram below shows the lifeline of the system and how various components work together, and at what point in time in the lifeline of the system.



#### **4.5 State Diagram**

For some operations, the behavior of the operation depends upon the state the receiver object is in. A state diagram is a tool for describing the states the object can assume and the events that cause the object to move from one state to another. State diagrams are particularly important for defining the behavior of operations throughout the system.



## 5.0 Implementation

### Overview

In this chapter, various sub chapters describe analysis of the dataset, and the implementation of the system itself. First, the dataset being used is imported, and goes through a process called exploratory data analysis where the dataset is analyzed closely. Here we find useful information such as hat features are highly correlated, and generally familiarize ourselves with the data. After this process is done, we move on to illustrate how the gui is coded onto the machine learning model. In a nutshell, every time the system runs, the whole process of machine learning is done. This includes splitting the dataset, training, testing and then the resulting output is used to predict the outcome of the data that has been input. The implementation features a GUI created from the python language, and will be illustrated later in this chapter.

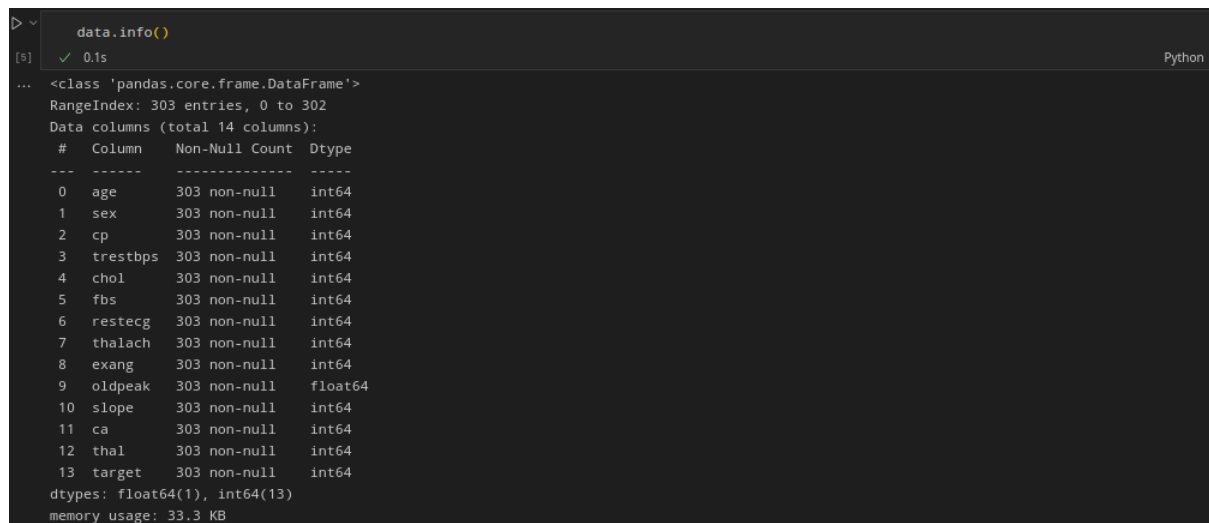
### 5.1 Dataset

This project was carried out on a publicly available database for heart disease sourced from Kaggle. The dataset contains a total of 303 records that were divided into two sets, the training set (67%) and the rest considered as the testing set(33%).

### 5.2 Exploratory Data Analysis of provided Dataset

The goal here is to find out more about the data and become a subject matter expert on the dataset being worked with.

#### 5.1.1Dataset overview



```
data.info()
[s] ✓ 0.1s Python

... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   age         303 non-null    int64
1   sex         303 non-null    int64
2   cp          303 non-null    int64
3   trestbps    303 non-null    int64
4   chol        303 non-null    int64
5   fbs         303 non-null    int64
6   restecg     303 non-null    int64
7   thalach     303 non-null    int64
8   exang       303 non-null    int64
9   oldpeak     303 non-null    float64
10  slope       303 non-null    int64
11  ca          303 non-null    int64
12  thal        303 non-null    int64
13  target      303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

This visual indicates that there are 303 entries (rows) and a total of 14 columns.

## **Data Dictionary**

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
  - \* 0: Typical angina: chest pain related decrease blood supply to the heart
  - \* 1: Atypical angina: chest pain not related to heart
  - \* 2: Non-anginal pain: typically esophageal spasms (non heart related)
  - \* 3: Asymptomatic: chest pain not showing signs of disease
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholesterol in mg/dl
  - \* serum = LDL + HDL + .2 \* triglycerides
  - \* above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
  - \* '>126' mg/dL signals diabetes
7. restecg - resting electrocardiographic results
  - \* 0: Nothing to note
  - \* 1: ST-T Wave abnormality
    - \* can range from mild symptoms to severe problems
    - \* signals non-normal heart beat
  - \* 2: Possible or definite left ventricular hypertrophy
    - \* Enlarged heart's main pumping chamber
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during exercise unhealthy heart will stress more
11. slope - the slope of the peak exercise ST segment
  - \* 0: Upsloping: better heart rate with exercise (uncommon)
  - \* 1: Flatsloping: minimal change (typical healthy heart)
  - \* 2: Downsloping: signs of unhealthy heart
12. ca - number of major vessels (0-3) colored by flourosopy
  - \* colored vessel means the doctor can see the blood passing through
  - \* the more blood movement the better (no clots)
13. thal - thallium stress result
  - \* 1,3: normal
  - \* 6: fixed defect: used to be defect but ok now
  - \* 7: reversible defect: no proper blood movement when exercising
14. target - have disease or not (1=yes, 0=no) (= the predicted attribute)

### 5.1.2 Target values analysis

I further analyze the dataset to find out how many records are found to have heart disease, and those discovered to not have heart disease.



We see that there are 165 persons with heart disease, and 138 without heart disease, our dataset is more or less balanced.

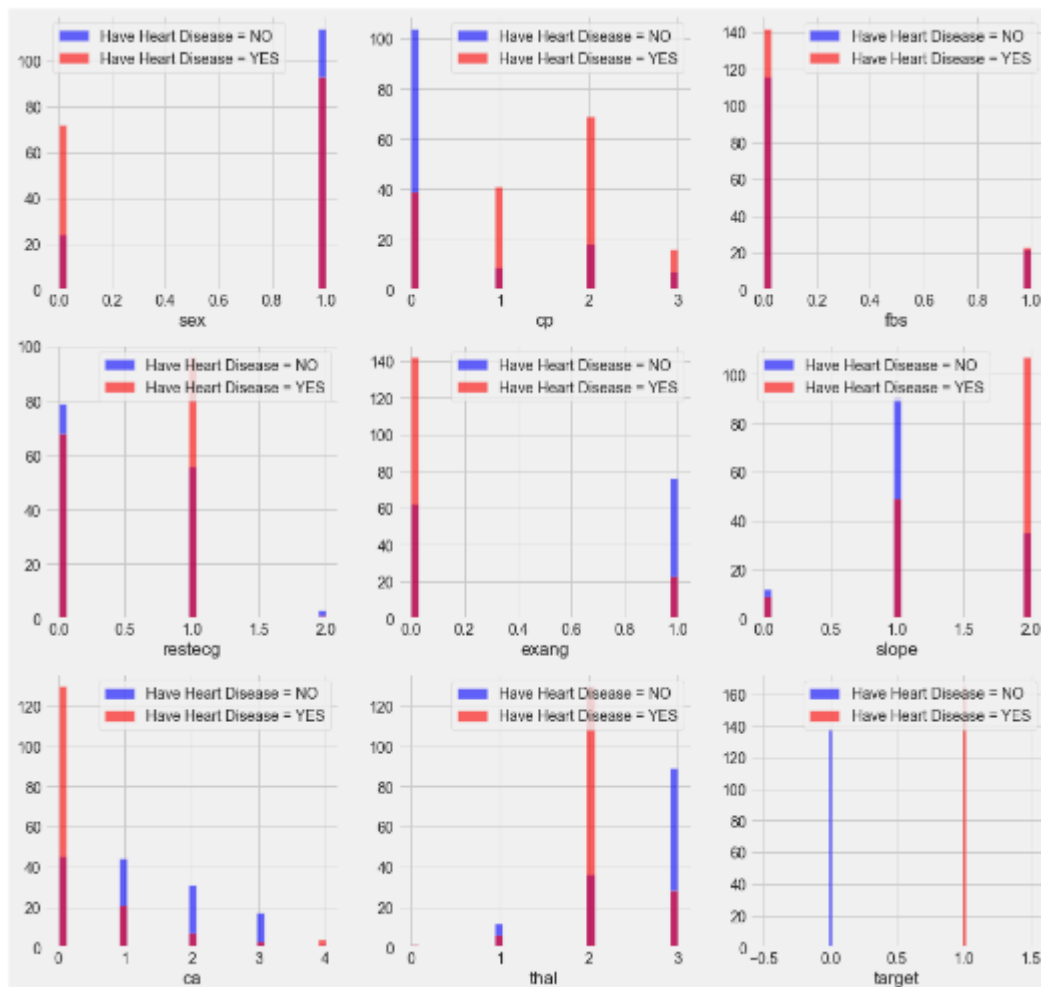
### 4.1.3 Heart disease by sex



From visualization we note that the male gender is more likely to contract heart disease as compared to the female gender. (1=male, 0=female).

### 5.1.4 General Summary of Feature Analysis

Below is a summarized graph of the features of the database and their interpretation as to how they can help understand the dataset better



*cp {Chest Pain}* : People with cp equal to 1, 2, 3 are more likely to have heart disease than people with cp equal to 0.

*restecg {resting electrocardiographic results}* : People with value 1 (signals non-normal heart beat, can range from mild symptoms to severe problems) are more likely to have heart disease.

*exang {exercise induced angina}* : People with value 0 (No ==> exercise induced angina) have heart disease more than people with value 1 (Yes ==> exercise induced angina)

*slope {the slope of the peak exercise ST segment}* : People with slope value equal to 2 (Downsloping: signs of unhealthy heart) are more likely to have heart disease than people with slope value equal to 0 (Upsloping: better heart rate with exercise) or 1 (Flatsloping: minimal change (typical healthy heart)).

*ca {number of major vessels (0-3) colored by fluoroscopy}* : the more blood movement the better so people with ca equal to 0 are more likely to have heart disease.

*thal {thallium stress result}* : People with thal value equal to 2 (fixed defect: used to be a defect but ok now) are more likely to have heart disease.

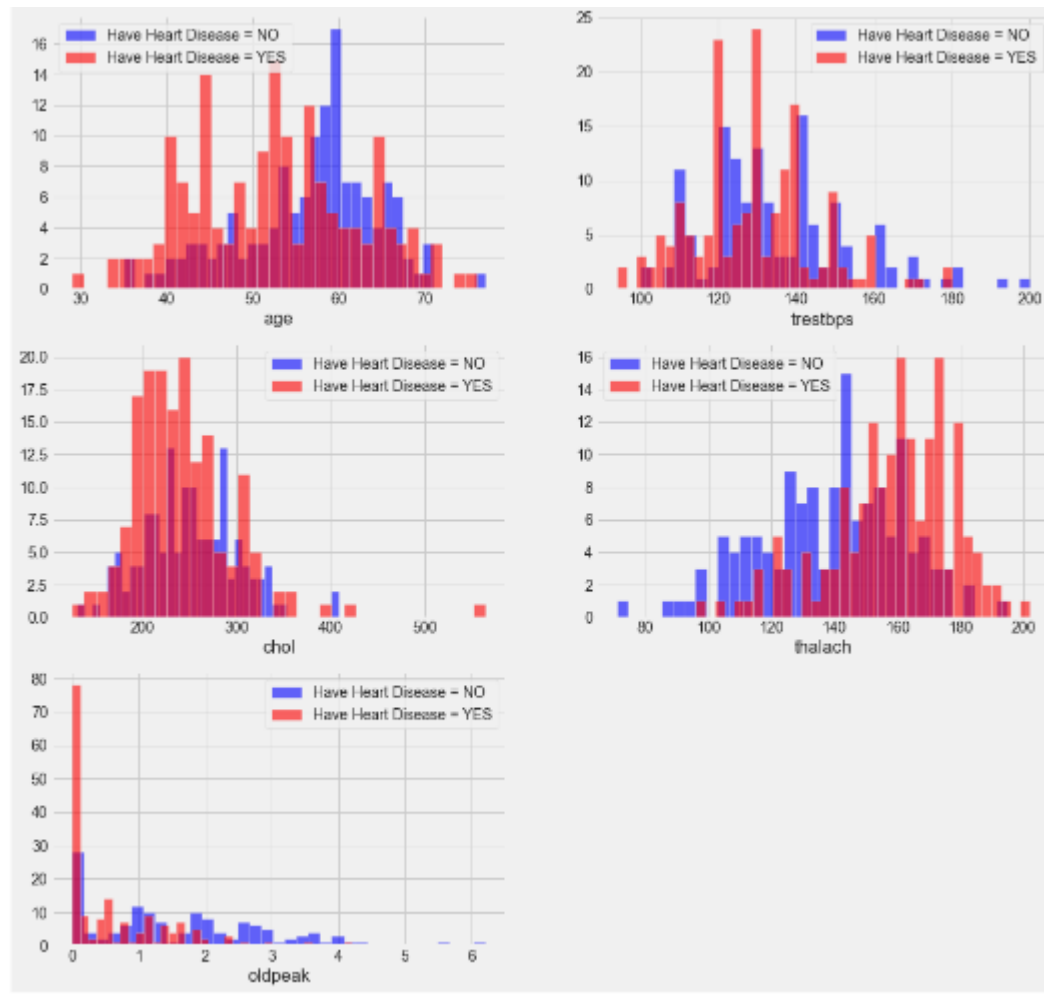
A deeper analysis of these features can be done to understand the dataset further.

*trestbps* : resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern

*chol* {serum cholesterol in mg/dl} : above 200 is cause for concern.

*thalach* {maximum heart rate achieved} : People who achieve a maximum more than 140 are more likely to have heart disease.

*oldpeak* ST depression induced by exercise relative to rest looks at stress of the heart during exercise. Unhealthy heart will stress more. This is shown in the graph below.



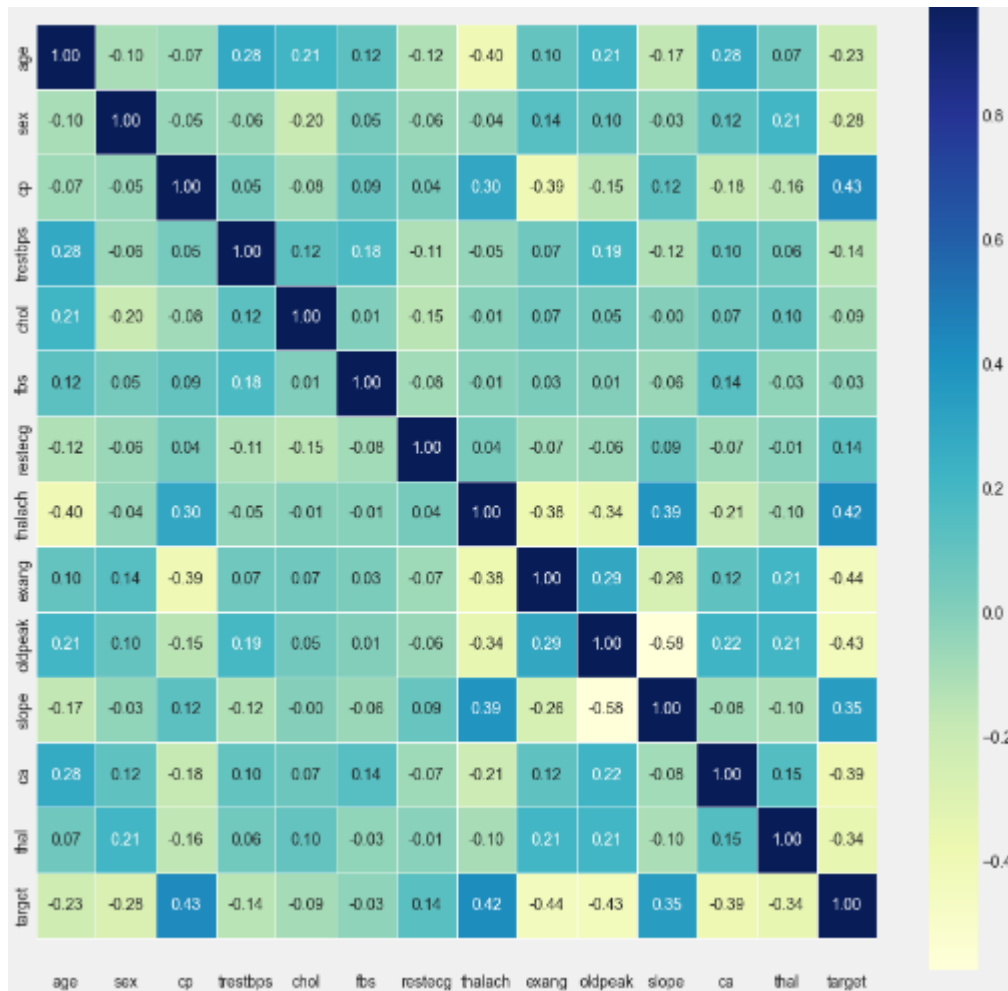
### 5.1.5 Feature Selection by Correlation Matrix

The correlation coefficient ranges from -1 to 1 and describes how the qualities are connected to one another. A number around 0 indicates a lesser connection (an exact 0 indicates no association). A number around 0 indicates a lower association (exact 0 implying no correlation). A negative correlation is greater if the value is closer to -1.

```
# Let's make our correlation matrix a little prettier
corr_matrix = data.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                  annot=True,
                  linewidths=0.5,
                  fmt=".2f",
                  cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

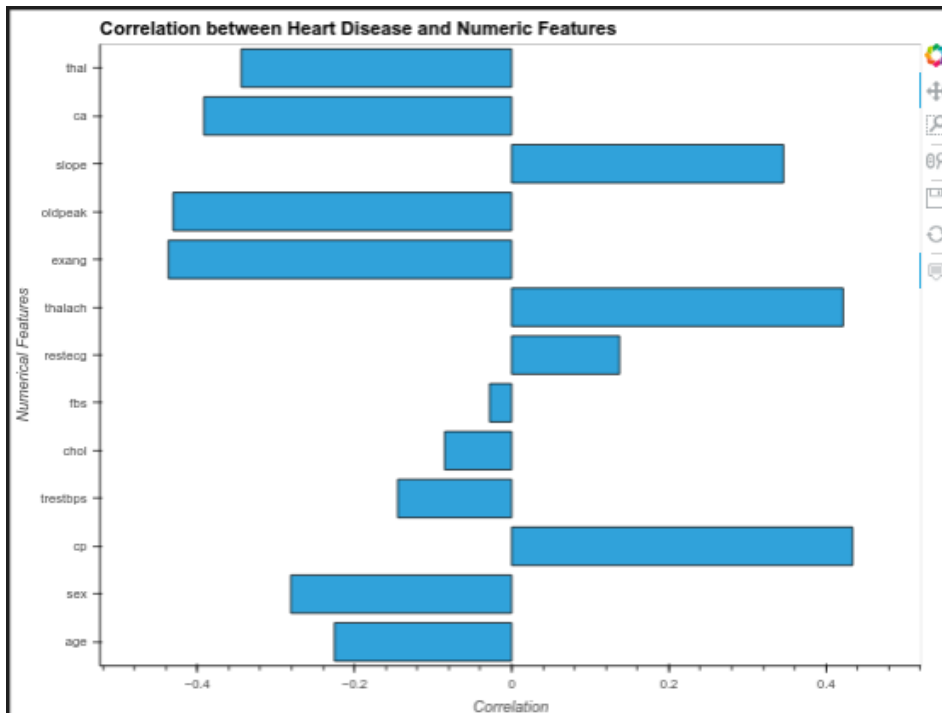
✓ 1.5s

After running the above code, we get this correlation matrix.



We see from the correlation matrix that not a lot of features are correlated to the 'target' feature. Some are even negative. The graph below better presents how these features are correlated to the target feature.





fbs and chol are the lowest correlated with the target variable. All other variables have a significant correlation with the target variable. I decided to use the following features in the machine learning process: **age, trestbps, chol, thalach and oldpeak.**

## 5.2 Data Processing

After exploring the dataset, I observed that I need to convert some categorical variables into dummy variables and scale all the values before training the Machine Learning model.

First, I'll use the 'get\_dummies' method to create dummy columns for categorical variables.

```

categorical_val.remove('target')
dataset = pd.get_dummies(data, columns = categorical_val)
✓ 0.7s
+ Code + Markdown Python
dataset.head()
✓ 0.3s Python

```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	63	145	233	150	2.30	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	37	130	250	187	3.50	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	41	130	204	172	1.40	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	56	120	236	178	0.80	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	57	120	354	163	0.60	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0

5 rows × 31 columns

```

from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])
✓ 0.4s
Python
dataset.head()
✓ 0.6s Python

```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0.95	0.76	-0.26	0.02	1.09	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	-1.92	-0.09	0.07	1.63	2.12	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	-1.47	-0.09	-0.82	0.98	0.31	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	0.18	-0.66	-0.20	1.24	-0.21	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	0.29	-0.66	2.08	0.58	-0.38	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0

5 rows × 31 columns

## 5.3 Building the Model

Before the model is built, we need to split the data into a training set(67%) and a testing set(33%).

```
y = heart['target']
X = heart.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

print(len(X_train))
len(X_test)
```

---

```
203
100
```

We see that the records have now been split: 203 records will be used as the training set while 100 records will be used as the testing set. I tried 6 different machine learning models:

1. Logistic Regression
2. K-Nearest Neighbors Classifier
3. Support Vector machine
4. Decision Tree Classifier
5. Random Forest Classifier
6. XGBoost Classifier

Out of these, the K-Nearest Neighbors Classifier came out on top with a testing accuracy of 87.91%.

## 5.4 Algorithms Used

### 5.4.1 Logistic Regression

Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable and the outcome is binary or dichotomous in nature. I will illustrate how I applied this algorithm in my code.

```
from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

The block of code above imports the model from the module sklearn and proceeds to use the training data set to train the model and test it. Below are the results achieved.

#### **Train Result:**

=====

**Accuracy Score: 86.79%**

---

Confusion Matrix:

```
[[ 80  17]
 [ 11 104]]
```

#### **Test Result:**

=====

**Accuracy Score: 86.81%**

---

Confusion Matrix:

```
[[34  7]
 [ 5 45]]
```

### **5.4.2 K-Nearest Neighbors Classifier.**

This algorithm takes data from a dataset, and forms a k number of centroids, determined by the user. It then assigns each record to the nearest centroid. In my system's case, the features are grouped to the closest centroid. This can be done on any feature, for example age. Records whose ages are close to the centroids selected are grouped together.

A goal number k, the number of centroids required in the dataset, must be defined. A centroid is a hypothetical or real point that represents the cluster's center. By lowering the in-cluster sum of squares, each data point is assigned to each cluster. To put it another way, the K-means algorithm finds k centroids and then allocates each data point to the closest cluster while keeping the centroids as small as feasible.

The algorithm has the following advantages and disadvantages:

#### **Advantages**

1. Easy to use and understand — The algorithm is simple to comprehend and put into practice.
2. Memory-based method — Allows it to quickly adapt to fresh training data.
3. Variety of distance measures – Users have the option of selecting the distance meter that is most suited to their needs (Euclidean, Minkowski, Manhattan distance etc.)

#### **Disadvantages**

1. Computational complexity – As the size of your training data grows, so does the speed at which computations are performed.
2. Poor performance while dealing with unbalanced data — When the bulk of the data used to train the model reflects one label, that label has a high chance of being predicted.
3. The model will be underfitted or overfitted to the data if the K value is set improperly.

## Implementation

This is the code for training and testing the dataset using the K-Nearest Neighbors Classifier.

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
✓ 0.6s
```

The following results are achieved after running this code:

### **Train Result:**

=====

**Accuracy Score: 86.79%**

Confusion Matrix:

```
[[ 82  15]
 [ 13 102]]
```

### **Test Result:**

=====

**Accuracy Score: 86.81%**

Confusion Matrix:

```
[[35  6]
 [ 6 44]]
```

## **5.4.3 Support Vector Machine**

Having described these algorithms, I will proceed to highlight their implementation and results.

The block of code below was used to import the svm model from sklearn and train it and carry out testing on the test dataset.

```
from sklearn.svm import SVC
svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)
svm_clf.fit(X_train, y_train)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

The results from this were;

#### **Train Result:**

=====

**Accuracy Score: 93.40%**

---

Confusion Matrix:

```
[[ 89   8]
 [  6 109]]
```

#### **Test Result:**

=====

**Accuracy Score: 87.91%**

---

Confusion Matrix:

```
[[36   5]
 [  6 44]]
```

### **5.4.4 Decision Tree Classifier**

The block of code below was used to import the Decision Tree Classifier model from sklearn and train it and carry out testing on the test dataset.

```
from sklearn.tree import DecisionTreeClassifier
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

Results are as shown below:

#### **Train Result:**

=====

**Accuracy Score: 100.00%**

---

Confusion Matrix:

```
[[ 97   0]
 [  0 115]]
```

#### **Test Result:**

=====

**Accuracy Score: 78.02%**

---

Confusion Matrix:

```
[[34   7]
 [13 37]]
```

### 5.4.5 Random Forest Classifier

The block of code below was used to import the Random Forest Classifier model from sklearn and train it and carry out testing on the test dataset.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
rf_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
rf_clf.fit(X_train, y_train)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

The Results are:

**Train Result:**

=====

**Accuracy Score: 100.00%**

---

Confusion Matrix:

```
[[ 97   0]
 [   0 115]]
```

**Test Result:**

=====

**Accuracy Score: 82.42%**

---

Confusion Matrix:

```
[[33   8]
 [  8 42]]
```

### Summary of Algorithms used and their scores

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	86.79	86.81
1	K-nearest neighbors	86.79	86.81
2	Support Vector Machine	93.40	87.91
3	Decision Tree Classifier	100.00	78.02
4	Random Forest Classifier	100.00	82.42

### 5.4.6 Hyperparameter Tuning

The process of selecting a set of ideal hyperparameters for a learning algorithm is known as hyperparameter tuning. A hyperparameter is a model argument whose value is determined prior to the start of the learning process. Hyperparameter tweaking is the cornerstone to machine learning algorithms.

Data is used to learn model parameters, and hyper-parameters are tweaked to get the best fit. Because finding the ideal hyper-parameter can be time-consuming, search techniques such as grid search and random search are employed.

## Grid Search

Grid search is the most fundamental way of hyperparameter tweaking. We simply generate a model for each conceivable combination of all of the hyperparameter values supplied, evaluate each model, and choose the architecture that delivers the best results using this method.

## Random Search

We don't supply a discrete set of values to explore for each hyperparameter in random search; instead, we provide a statistical distribution for each hyperparameter from which values can be randomly picked.

## Implementation

These are the results before I used hyperparameter tuning.

```
Train Result:
=====
Accuracy Score: 86.79%

CLASSIFICATION REPORT:

```

		0	1	accuracy	macro avg	weighted avg
precision	0.86	0.87	0.87	0.87	0.87	0.87
recall	0.85	0.89	0.87	0.87	0.87	0.87
f1-score	0.85	0.88	0.87	0.87	0.87	0.87
support	97.00	115.00	0.87	212.00	212.00	212.00

```

Confusion Matrix:
[[ 82  15]
 [ 13 102]]

Test Result:
=====
Accuracy Score: 86.81%

CLASSIFICATION REPORT:

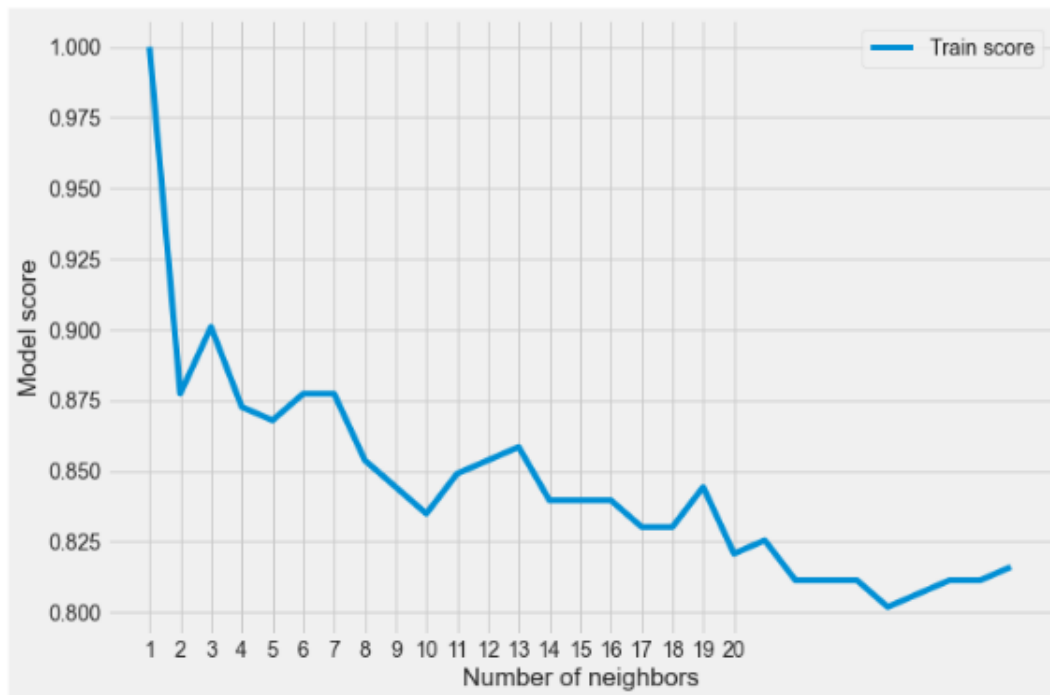
```

		0	1	accuracy	macro avg	weighted avg
precision	0.85	0.88	0.87	0.87	0.87	0.87
recall	0.85	0.88	0.87	0.87	0.87	0.87
f1-score	0.85	0.88	0.87	0.87	0.87	0.87
support	41.00	50.00	0.87	91.00	91.00	91.00

```

Confusion Matrix:
[[35  6]
 [ 6 44]]
```

When hyperparameter tuning is done we notice that the test accuracy is at its peak, and it was still the highest among the models I tried out.



In hypertuning we introduce an additional element, which is the specific number of neighbors we want to use.

```
knn_clf = KNeighborsClassifier(n_neighbors=27)
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
✓ 0.1s
```

These are the results from hypertuning where we used 27 neighbors.

```
Train Result:
=====
Accuracy Score: 81.13%

CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.84	0.80	0.81	0.82	0.81
recall	0.73	0.88	0.81	0.81	0.81
f1-score	0.78	0.83	0.81	0.81	0.81
support	97.00	115.00	0.81	212.00	212.00

```

Confusion Matrix:
[[ 71  26]
 [ 14 101]]

Test Result:
=====
Accuracy Score: 87.91%

CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.89	0.87	0.88	0.88	0.88
recall	0.83	0.92	0.88	0.87	0.88
f1-score	0.86	0.89	0.88	0.88	0.88
support	41.00	50.00	0.88	91.00	91.00

```

Confusion Matrix:
[[34  7]
 [ 4 46]]
```



## 5.5 Graphical User Interface

The GUI is a form designed using the python language(code attached). The form accepts all the parameters that are needed as shown below. The user is required to have every field filled. After this is done, the user can then click the Analyze/Predict button which then determines if the patient whose details have been provided has heart disease or not.

### UI design

HEART DISEASE PREDICTION

# HEART DISEASE PREDICTION MODEL

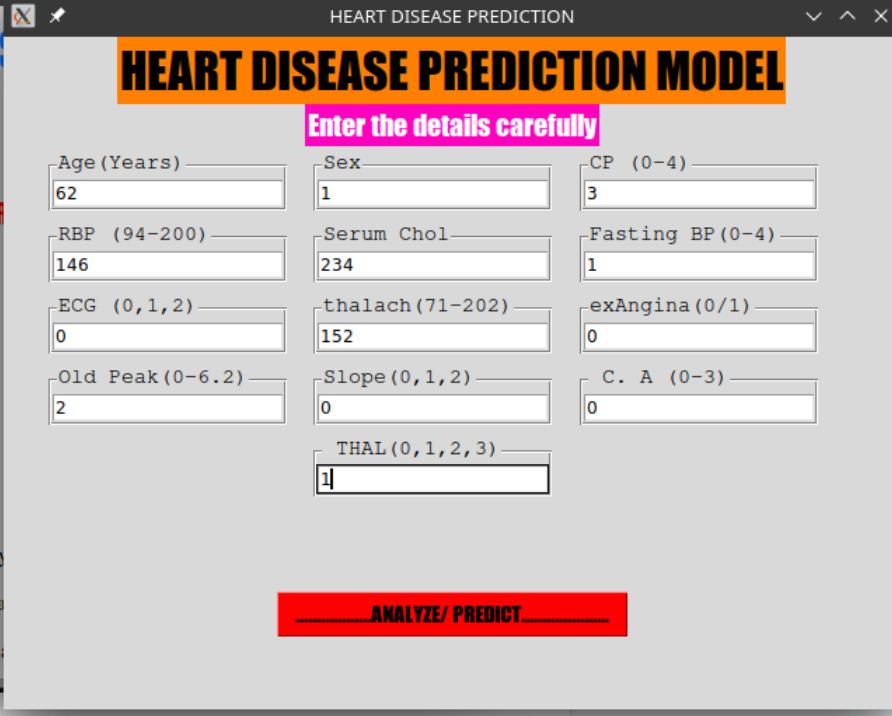
Enter the details carefully

Age (yrs)	Sex	CP (0-4)
<input type="text"/>	<input type="text"/>	<input type="text"/>
RBP (94-200)	Serum Chol	Fasting BP (0-4)
<input type="text"/>	<input type="text"/>	<input type="text"/>
ECG (0,1,2)	thalach (71-202)	exAngina (0/1)
<input type="text"/>	<input type="text"/>	<input type="text"/>
Old Peak (0-6.2)	Slope (0,1,2)	C. A (0-3)
<input type="text"/>	<input type="text"/>	<input type="text"/>
	THAL (0,1,2,3)	
	<input type="text"/>	

**ANALYZE/ PREDICT**

### Heart disease positive case

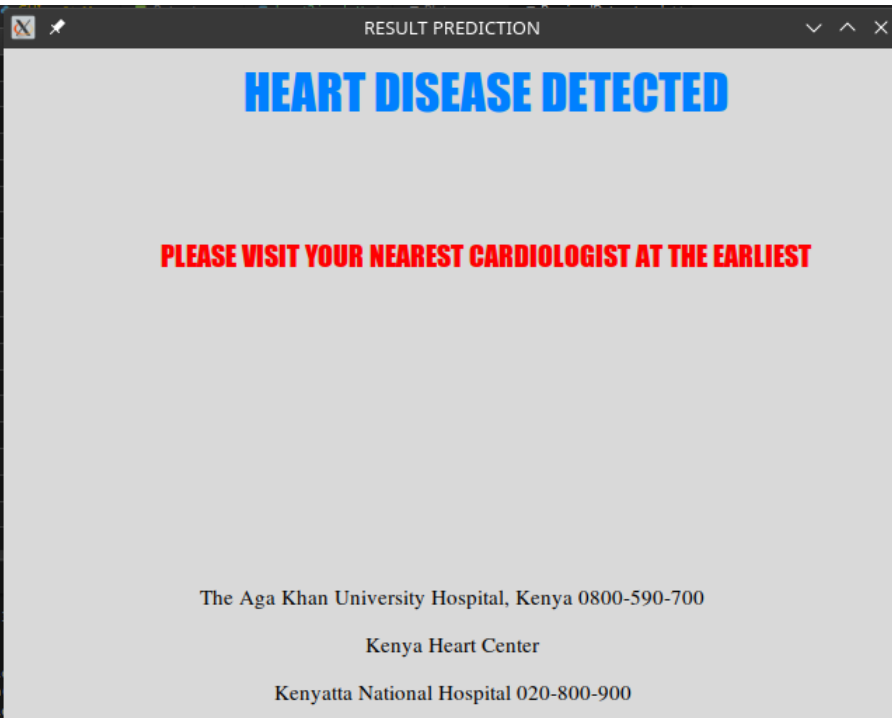
As shown below, this is a case where the system predicts the presence of Heart Disease. Each text box is labeled well to reduce errors made during input of data.



The screenshot shows a web application window titled "HEART DISEASE PREDICTION". At the top, there is a large orange banner with the text "HEART DISEASE PREDICTION MODEL". Below this, a pink banner reads "Enter the details carefully". The form contains several input fields arranged in a grid:

Age (Years) — 62	Sex — 1	CP (0-4) — 3
RBP (94-200) — 146	Serum Chol — 234	Fasting BP (0-4) — 1
ECG (0, 1, 2) — 0	thalach (71-202) — 152	exAngina (0/1) — 0
Old Peak (0-6.2) — 2	Slope (0, 1, 2) — 0	C. A (0-3) — 0
THAL (0, 1, 2, 3) — 1		

At the bottom of the form, there is a red button labeled "ANALYZE/ PREDICT".



The screenshot shows a web application window titled "RESULT PREDICTION". The main content area has a light gray background. At the top, there is a large blue banner with the text "HEART DISEASE DETECTED". Below this, a red banner reads "PLEASE VISIT YOUR NEAREST CARDIOLOGIST AT THE EARLIEST". At the bottom of the screen, there is contact information for The Aga Khan University Hospital, Kenya, including the phone number 0800-590-700, the Kenya Heart Center, and the Kenyatta National Hospital phone number 020-800-900.

### No heart disease case

In this case, the system outputs a negative case where the patient whose data is entered is predicted to not have any heart disease present.

HEART DISEASE PREDICTION

**HEART DISEASE PREDICTION MODEL**

Enter the details carefully

Age (Years) 51 Sex 0 CP (0-4) 0

RBP (94-200) 131 Serum Chol 305 Fasting BP (0-4) 0

ECG (0, 1, 2) 1 thalach (71-202) 143 exAngina (0/1) 1

Old Peak (0-6.2) 1 Slope (0, 1, 2) 1 C. A (0-3) 0

THAL (0, 1, 2, 3) 3

**ANALYZE/ PREDICT**

RESULT PREDICTION

**NO DETECTION OF HEART DISEASES**

Do Not Forget to Exercise Daily And Eat Healthy Food.

### 5.6 Code Documentation Github Link

[https://github.com/Onsareorucho/HeartDiseasePredictionSystem\\_schoolproject](https://github.com/Onsareorucho/HeartDiseasePredictionSystem_schoolproject)

## **6.0 Conclusion**

### **6.1 Results**

The main objective that I had at the beginning of this system's development was a system that would be able to accept input from a user and display whether or not the patient whose details have been entered has heart disease or not.

The system created has achieved the following results:

#### **Accept Input from the user**

During development a user interface, clearly described in the previous section is deployed. The user interface is written purely in python and is the interface through which the user of the system inputs data. After data is input, the user then clicks a button to run the system's prediction model.

#### **Splitting Data into Training and Testing Sets**

This is also accomplished in the system's codework where the system is able to split the data in the dataset provided into two: the training set and the testing set. These sets are used to come up with a trained model which is then tested using test data. After this, the model is now ready to accept user input.

#### **Provide the accuracy of the model trained**

In the Jupyter Notebook attached, every algorithm used also has its accuracy measure indicated next to it. This helped me to decide on which algorithm was best suited for my case.

#### **Give output**

The system is able to give output after the user has input data. The output is either 'Heart Disease Detected' or 'No Heart Disease Detected'.

The above mentioned were all objectives of the system, which have all been implemented.

## **6.2 Remarks**

Creating this system has presented a challenge in itself to conduct thorough research into both the fields of medicine and machine learning. It has proved a steep learning curve but I believe the skills gained will help me in future projects. Heart Disease has been a nuisance to the world and I believe this project, though small, can bring change in areas where expensive systems are not available, or even from the comfort of a person's computer.

## 6.3 Future Work

This system had a few shortcomings which I look forward to addressing in future work.

I intend to:

1. Improve the user interface. Many aspects were not taken into consideration when developing the UI. It felt lackluster and in future works I will work to make the user interface more lively.
2. Add functionalities: The current system has very limited functionalities i.e it does not have a login functionality, a database functionality. These are functionalities I intend to add to the system in my future work.
3. The algorithms implemented may not be the best. I will continue to research on the best possible algorithms to use and integrate them into my system in future releases.

## **7.0 References**

1. K. Polaraju, D. Durga Prasad, "Prediction of Heart Disease using Multiple Linear Regression Model", International Journal of Engineering Development and Research Development, ISSN:2321-9939, 2017.
2. Marjia Sultana, Afrin Haider, "Heart Disease Prediction using WEKA tool and 10-Fold cross-validation", The Institute of Electrical and Electronics Engineers, March 2017.
3. Dr.S.Seema Shedole, Kumari Deepika, "Predictive analytics to prevent and control chronic disease", <https://www.researchgate.net/publication/316530782>, January 2016.
4. Mr. Chala Beyene, Prof. Pooja Kamat, "Survey on Prediction and Analysis of the Occurrence of Heart Disease Using Data Mining Techniques", International Journal of Pure and Applied Mathematics, 2018.
5. R. Sharmila, S. Chellammal, "A conceptual method to enhance the prediction of heart diseases using the data techniques", International Journal of Computer Science and Engineering, May 2018.
6. Jaymin Patel, Prof. Tejal Upadhyay, Dr.Samir Patel, "Heart Disease Prediction using Machine Learning and Data Mining Technique", International Journal of Computer Science and Communication, September 2015-March 2016, pp.129-137.
7. Boshra Brahmi, Mirsaeid Hosseini Shirvani, "Prediction and Diagnosis of Heart Disease by Data Mining Techniques", Journals of Multidisciplinary Engineering Science and Technology, vol.2, 2 February 2015, pp.164-168.
8. S.Prabhavathi, D.M.Chitra, "Analysis and Prediction of Various Heart Diseases using DNFS Techniques", International Journal of Innovations in Scientific and Engineering Research, vol.2, 1, January 2016, pp.1-7.
9. Sairabi H.Mujawar, P.R.Devale, "Prediction of Heart Disease using Modified K-means and by using Naïve Bayes", International Journal of Innovative research in Computer and Communication Engineering, vol.3, October 2015, pp.10265-10273.
10. Masethe, H.D. and Masethe, M.A., 2014, October. Prediction of heart disease using classification algorithms. In Proceedings of the world Congress on Engineering and Computer Science (Vol. 2, pp. 22-24).
11. Thomas, J. and Princy, R.T., 2016, March. Human heart disease prediction system using data mining techniques. In 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT) (pp. 1-5). IEEE.
12. Yildirim, S. (2021, January 6). *15 must-know machine learning algorithms*. Medium. Retrieved January 20, 2022, from <https://towardsdatascience.com/15-must-know-machine-learning-algorithms-44faf6bc758e>