

HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITHM

Submitted in partial fulfillment of the requirements for the award of
Master of Engineering degree in Computer Science and Engineering

by

ABHAY KUMAR [Reg No: 42310001]



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

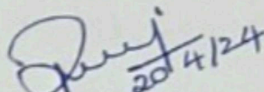
Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

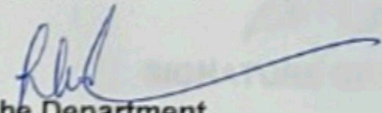
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **ABHAY KUMAR (Reg.No-42310001)** who carried out the Project entitled "**HEART DISEASE PREDICATION USING MACHINE LEARNING**" under my supervision from January 2024 to April 2024.

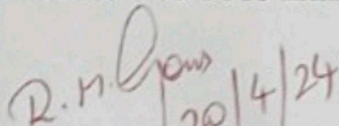

20/4/24
Internal Guide

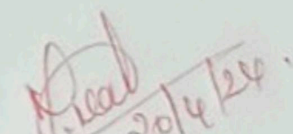
Dr. M.D. ANTO PRAVEENA M.E., Ph.D.,


Head of the Department

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DR. L. LAKSHMANAN, M.E., Ph.D.,
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva Voce Examination held on 20/4/2024


20/4/24
Internal Examiner

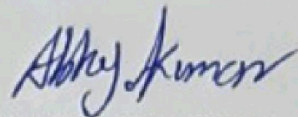

20/4/24
External Examiner

DECLARATION

I, Abhay Kumar (Reg. No - 42310001), hereby declare that the Project Report entitled "HEART DISEASE PREDICATION USING MACHINE LEARNING AND ALGORITHMS" done by me under the guidance of Dr. M.D. Anto Praveena M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Master of Engineering degree in Computer Science and Engineering.

DATE: 20/4/2024

PLACE: Chennai



SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing. **Dr.L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide of **Dr. M.D. Anto Praveena M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The application of Machine Learning is highly successful in fields like e- business, retail and marketing has led to a vast spread in its application in various other industries. Among these sectors is the healthcare industry. This industry is known to be “information rich”, and has a great scope for effective decision making & discovering hidden patterns. Advanced machine learning techniques can help us with this situation. Talking about the healthcare industry, one of the most talked about is the heart diseases. Heart is a vital organ. It pumps blood and supplies it to all organs to function. Prediction of the occurrences of heart diseases is significant work in this field.

In this work, eight machine learning algorithms which include Random Forest, SVM, Decision Tree, Naïve Bayes, ANN, KNN, Logistic Regression and Deep Neural Network. Xg-boost, Ada boost These algorithms are used to unfold a prediction system which will analyze and predict whether the particular patient is pertaining to any heart disease or not. The main objective is to identify the best algorithm suitable for this, and which provides maximum accuracy results. The algorithms are compared based on features. We are working on the algorithm with the best accuracy results. This will help the doctors to assist the heart patient's problem easily. Advanced machine learning techniques can help us with this situation.

Talking About the healthcare industry, one of the most talked about the heart disease. The heart is important to the human body. Millions of number deaths in the world are due to heart disease. Some symptoms like chest pain, faster blood pressure, discomfort in breathing are recorded in medical history this all data will be analysis on the regular basis of heart diseases. In this research review and overview of heart diseases and its current model procedure it's firstly introduce. And in-depth analysis of the most relevant of machine learning technics algorithms available in literature of heart diseases in prediction system.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF TABLES	VIII
	LIST OF FIGURES	IX
	LIST OF ABBREVIATIONS	XI
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Motivation	3
	1.3 Proposed Research	4
	1.4 Research Objectives	5
2	LITERATURE REVIEW	6
3	SYSTEM ANALYSIS	8
	3.1 Overview of the System	8
	3.2 Advantages of the System	10
	3.3 Machine learning Algorithms	11
	3.3 SVM	11
	3.3 Naïve Bayes	13
	3.3 KNN	14
	3.3 Decision Tree	15
	3.3 Random Forest	16
	3.3 Logistic Regression	16
	3.3 ANN	17
	3.3 XG-Booster	19

	3.3 Ada-Boost	19
4	SYSTEM DESIGN	20
	4.1 System Architecture	20
	4.2 Data Flow Diagram	21
5	COADING & TESTING	23
	5.1 Software Requirements	23
	5.2 Objectives and types of testing	24
	5.3 Testing in Machine learning	26
	5.4 Test Cases	28
6	RESULTS AND ANALYSIS	30
	6.1 Data Set	30
	6.2 Metrics For Performance Analysis	33
	6.3 Performance of Machine learning Algorithms	34
	6.4 Results from the test data set	47
7	CONCLUSION AND FUTURE WORK	53
	7.1 Concluding Remarks	53
	7.2 Future Work	53
	REFERENCES	54
	APPENDIX	56
	A SOURCE CODE	56
	B SCREENSHOTS	63
	C RESEARCH PAPER	77

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
1.1	Different types of heart disease	2
5.1	Test Cases	28
6.1	Confusion Matrix	33
6.2	SVM Confusion Matrix	34
6.3	Bayes Confusion Matrix	35
6.4	KNN Confusion Matrix	36
6.5	Decision Tree Confusion Matrix	37
6.6	Random Forest Confusion Matrix	38
6.7	Logistic Regression Confusion Matrix	40
6.8	ANN Confusion Matrix	41
6.9	XG-boost Confusion Matrix	43
6.10	Ada-boost Confusion matrix	45
6.11	Accuracy Comparison of Algorithms	46

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	Possible hyperplanes	11
3.2	Hyperplane with the maximum margin	12
3.3	Naïve Bayes	13
3.4	KNN Clustering	14
3.5	Decision tree	15
3.6	Random forest with two trees	16
3.7	Sigmoid function graph	17
3.8	ANN	18
4.1	System Architecture	20
4.2	Data Flow Diagram	21
6.1	Dataset used	31
6.2	SVM Result	35
6.3	Naïve Bayes Result	36
6.4	KNN Results	37
6.5	Decision tree Result	38
6.6	Random forest Result	39
6.7	Logistic regression Result	40
6.8	ANN Result	42
6.9	Xg-boost Result	44
6.10	Ada-boost Result	45
6.11	Percentage of men and women in dataset	47
6.12	Risk of men and women having a heart attack	47
6.13	Analysis of positive heart attack in men out of a total man	48
6.14	Analysis of positive Heart attack in women out of a total woman	48

6.15	Distribution of people vs age	49
6.16	Age	49
6.17	Distribution of serum cholesterol in mg/dl	50
6.18	Cholesterol	50
6.19	Distribution of maximum heartrate achieved	51
6.20	Thalach	51
6.21	Distribution of rest blood pressure	52
6.22	Tresbps	52
7.B1	Screenshot for input GUI.	75
7.B2	Screenshot for output GUI.	76

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ANN	Artificial Neural Network
BP	Blood Pressure
CVD	Cardiovascular Disease
DNN	Deep Neural Network
ECG	Electrocardiogram
EKG	Electrocardiography
FP	False Positive
FN	False Negative
KNN	K-Nearest Neighbor's
MCC	Maxwells Correlations Coefficients
SVM	Support vector machines
TP	True Positive
TN	True Negative
WEKA	Waikato Environment for knowledge analysis

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In our day to day life, there are many factors that affect the human heart. The report by World Health Statistics puts emphasis on this fact: one out of three adults all over the world today, is suffering from hypertension, i.e., elevated levels of blood pressure – a condition which results in deaths from a stroke or leads to heart attacks and other heart related diseases. Heart diseases, also called by the name cardiovascular diseases, include a no. of factors that can affect the heart

– which not only include the heart attacks. Cardiovascular diseases can be considered a primary reason of casualty in numerous countries. Cardiovascular diseases result in the death of one person in an average of 34 seconds as stated by a study held in USA.

Coronary Cardiovascular disease, Heart disease and Cardiomyopathy, all fall under the categories of heart/ cardiovascular diseases. The term Cardio Vascular Disease consists of various conditions that affects heart, blood vessels, and the way in which the pumping and circulation of blood takes place all through the body. Diagnosis is quite a complex as well as a necessary task that needs to be carried out efficiently as well as accurately. Mostly, the diagnosis/detection is made and based over the doctor's knowledge, skill and professional experience. This provides many undesirable outcomes and over the budget medical cost for the treatments given to the patients.

Hence, it is extremely beneficial to have a system for automating the medical diagnosis for predicting the possibility of heart disease before it happens based on certain physiological parameters. So before going into the details of our work, we will look into usage of Data analytics in heart disease prediction.

Table 1.1 Different types of heart disease

Arrhythmia	The irregular beating of heart. It is either too slow or too fast.
Congestive heart failure	The heart doesn't pump blood as it normally should which leads to be a chronic disease.
Cardiac arrest	An unexpected loss of consciousness, heart function, and breathing that occurs suddenly.
Congenital heart disease	The abnormality in the heart that develops before birth.
High Blood Pressure	The condition in which the force of blood against the walls of the artery is too high.
Coronary artery disease	The damage in the heart's major blood vessels or any disease that occurs in the blood vessels.
Peripheral artery disease	The blood Vessels are narrowed which indeed reduces flow of blood in the limbs.
Stroke	Interruption of blood supply to the brain.

1.2 DATA ANALYTICS IN HEART DISEASE PREDICTION

Data analytics and data mining algorithms have been used across various datasets to extract patterns and predict the occurrence of heart diseases by numerous papers.

The most commonly used algorithms have been SVM, Logistic Regression, Naive Bayes, KNN and decision tree, out of which SVM and Logistic regression have been known to give more accurate results. Some papers have also used Hadoop based MapReduce and HDFS algorithms to store data across different nodes and run the system parallelly.

The most commonly used attributes against the datasets have been Blood pressure and heart rate and the age group used in the datasets is generally over the age of 30.

Although all known papers have used the above-mentioned algorithms, none have employed deeplearning in their studies, which tends to give a higher accuracy rate. Our system, along with all these algorithms uses a deep neural network and more attributes like cholesterol, angina, age, sex etc. for more accurate prediction.

1.3 MOTIVATION

Disease predication systems are best practices for eliminating human errors in disease.

And aiding in disease prevention through early identifications Diagnosis of cardiac arrest

Based on patients' health record characteristics is a multidimensional decision-life styling

The motivation for heart disease predication is to reduce the number of people who death.

And to reduce the number of people who die from heart disease. Which is the leading the

Cause of death worldwide health organization, nearly 20 million people will die from heart

Disease by 2030. Disease predication System can help prevent disease by identifying it

Early, which can help to reduce human error diagnosis. For example, machine learnings

It is well known fact that patients when affected with heart disease are subjected to lot of tests like ECG, EKG, and so on. But these tests are carried out only when the person gets chest pain or kind of symptom leading to heart disease. In today's world there are lot of body wearable devices which could give the pulse rate, Blood pressure and so. There is nothing like person would get heart disease only after the age of 40. The current generation is under lot of stress and pressure due to work and other factors. So, there is an urgent need to perform data analysis of the physiological parameter towards possibility of heart disease before heart attack happens. So, towards this some amount of research carried out by employing Machine learning algorithm for prediction of heart disease like Regression, KNN, SVM, Bayes, Decision Tree and so on.

So, with the upcoming Deep learning, we here have proposed to develop a heart disease prediction system with highest accuracy as compared to other machine learning algorithm based on thirteen physiological parameters as compared to most important ones which are heart rate, age and sex alone.

1.3 PROPOSED RESEARCH

Based on above mentioned motivation, we here have proposed to develop a Deep learning-based prediction system of heart diseases which would be based on historical physiological data set. It is clear that good amount of work done in the past in predicting heart diseases with the help of machine learning algorithms like SVM, Regression, Bayes, KNN and even Artificial Neural Network. Most of the system have taken two parameters like heart rate, age and so. But we in our work are taking thirteen parameters including cholesterol, Blood pressure, heart rate which is unique in prediction of heart disease.

So our system encompasses the Deep Learning framework called Deep Artificial Neural Network for prediction of heart disease based on physiological data set of people based on age group and sex which includes thirteen parameters like cholesterol, resting BP, resting ECG, maximum heart rate, age, sex, chest pain, fasting blood sugar, exercise included angina, number of major vessel, slope of peak exercise, thalassemia, depression. This is compared with other machine learning algorithms like SVM, Bayes, KNN, Decision Tree, Random Forest, Logistic Regression and ANN for prediction of heart

1.4 RESEARCH OBJECTIVES

The major objective of our project work is focused on the following

- Collection of heart disease related data set and preprocessed
- Validation of SVM, Regression, Bayes, KNN algorithm for heart disease prediction on the dataset.
- Validation of random forest and decision tree on the given set of data for heart disease prediction
- Validation of ANN and Deep Neural Network on data set for prediction of heart disease.
- Accuracy as well as error computation on above mentioned algorithm
- Comparative analysis of algorithm in terms of accuracy

1.5 REPORT ORGANISATION

This report deals with various the aspect of the proposed system. We begin with chapter 1 that gives the basic overview of the project. It explains how data analytics is used in heart disease prediction and gives our research objectives. Chapter 2 starts with explaining all the different research papers that gave us ideas to carry forward their work. These papers gave us a strong background. The next chapter analyses the present system as well as the proposed system. It gives a basic idea as to how the project will work and what objectives it will work with. It also specifies all the algorithms used in the project. In chapter 4 we have described the data set used, its attributes, workflow of the system and system architecture. Chapter 5 gives the hardware and software requirements along with testing. Chapter 6 gives the results and accuracy of our project. Conclusion gives us a review about this project and describes about how this system is useful to the society. Next chapter deals with the enhancements of this project and how it can be modified for better results and the last chapter lists the references.

CHAPTER 2

LITERATURE REVIEW

Purshottam [1] did the work of Predicting Heart Disease using the model of Multiple Regression, which implied that Multiple Linear Regression was more suitable for heart disease prediction. The dataset used, consisted of 3000 entries with 13 attributes and was partitioned into training and testing, where training data was 70% and testing data was 30%. Upon considering the results, it was concluded that the regression algorithm was more accurate than others.

Krishanan, [2] did the prediction using SMO, J48, KStar, Multilayer perception and Bayes Net using WEKA. From the results of the prediction, it was found that the performance of Bayes Net and SMO was optimum, by using the k-fold cross validation technique. The other algorithms did not give satisfactory performance. Hence, accuracy performance was improved for better diagnostic decisions.

Soumen Nikhel, [3] focused on predicting general chronic diseases rather than just heart disease prediction. She utilized the hospital health records by data mining using Decision tree, Support Vector Machine (SVM), Artificial Neural Network and Naive Bayes. From this prediction, on applying the mentioned algorithms, it was concluded that highest accuracy was given by SVM, while for diabetes, the highest accuracy rate was achieved by Naive Bayes.

Aditi Sharma, [4] used various algorithms consisting of SVM, Classification Tree, Logistic Regression, Naive Bayes, SVM, Naive Bayes, KNN and ANN. The highest accuracy rate among these was attained by Logistic regression.

Avinash Singh [5] used data mining for heart disease prediction. WEKA was used for the diagnosis of heart disease and to give service qualities in the healthcare centers. Various algorithms like SVM, KNN, Naïve Bayes, ANN, Association rule and Decision Tree. The results suggested that SVM gave a higher accuracy rate than other algorithms and data mining techniques.

L.Lakshman Rai, [6] Predicted and Analyzed the occurrence rate of Heart Disease Using techniques of Data Mining. The primary aim of the paper was the prediction of the occurrence of heart diseases for prior automatic diagnosis within the result in less time. It utilized various factors like blood sugar, age, heart rate, sex to predict whether a person is likely to have a heart disease or not. The analysis of data was done using WEKA.

Abhay Sinha, [7] used non-linear algorithms for classification. Some bigdata tools like Mapreduce and HDFS along with SVM were used with an optimum set of attributes for the prediction of heart diseases. HDFS was used to store large datasets in separate nodes and this distributed data was executed simultaneously in multiple nodes using SVM. Here, SVM was used in a parallel fashion, which gave optimum computational time compared to sequentially used SVM.

Susil Kumar, [8] proposed the use of machine learning and data mining algorithms for heart disease prediction. The objective of this paper was to retrieve hidden patterns by the application of techniques of data mining. The results showed that based on UCI data, J48 had the highest rate of accuracy in comparison to LMT.

Anjani, [9] used data mining techniques for the prediction of heart diseases. This study aided the medical practitioner to take better decisions based on a particular parameter. By training and testing a particular parameter, it retrieved an accuracy rate of 86.3% in testing, and an accuracy rate of 87.3% during training.

Akash Chandel, [10] also proposed a prediction of multiple disease using the techniques of data mining. These days, data mining is being extensively used for multi disease prediction, as the number of tests is subsequently reduced. The study majorly focused on predicting breast cancer, heart diseases and diabetes etc.

CHAPTER 3

SYSTEM ANALYSIS

3.1 OVERVIEW OF THE SYSTEM

Heart Disease prediction is done at hospital based on patient visiting hospital based on current heart condition like high pulse rate, chest pain or so. Accordingly, patient is tested for heart blockage using ECG, Angiogram and so. Based on the test, patient is treated by performing pacemaker, Angioplasty and so.

Now with ever increasing pressure of work and stress, there is no such thing that heart problem comes only after age of 50 or so. Now even adults at the age of 20, 30s, 40s get heart disease due to work pressure and stress. Some even get in their childhood too. So, there is an urgent need to come with an automated medical diagnosis system for heart disease based on physiological data based on Blood pressure, heart rate, blood sugar, cholesterol and so for predicting the heart disease. Some amount of work done by applying machine learning algorithm based on data collected for predicting the heart disease using algorithms like Regression, SVM, Bayes, KNN, Decision Tree and ANN for heart disease prediction.

But in all these systems, only few parameters taken for prediction which are heart rate, age and so. Also none have tried using Deep neural network in this field of heart disease prediction. So, we in this work have applied Deep Neural Network for the heart disease prediction based on 13 physiological parameters and compared with other algorithms like Regression, SVM, Bayes, KNN, Decision Tree, Random Forest and ANN in terms of error and accuracy.

Now for our work, we have taken dataset that was publicly available on Kaggle for predicting the heart diseases. The parameters used as input for data analysis using Machine learning algorithm are as follows:

1. age: which is taken in years
2. sex: 1 for male, 0 for female
3. cp: short for chest pain.
4. trestbps: blood
 pressure taken when
 the body is resting
5. chol: level of cholesterol
6. Fasting blood sugar: (1 for true; 0 for false)
7. resting electrocardiographic

 resultsValue 0: normal

 Value 1: having the ST-T wave which is not normal

 Value 2: shows the probability of having left ventricular hypertrophy
8. thalach: the max level of heart beats achieved by the heart
9. exercise induced angina: 1 for yes; 0 for no
10. oldpeak = the depression by exercise compared to the one at rest
11. slope: peak of the ST segment during

 exerciseValue 1: no slope

 Value 2: straight line

 Value 3: down sloping

12.ca: major vessels (0-3) coloured by fluoroscopy

13.thal: 3 = perfect; 6 = permanently defected; 7 = defect that can be altered

14.num: heart disease diagnosis

3.2 ADVANTAGES OF THE SYSTEM

- Our study focuses not only on Blood pressure and heart rate of the patient, but also on other factors like age, sex, angina, chest pain location, cholesterol etc., that are not usually taken in consideration.
- With a greater number of attributes against which the system is being tested and trained, the data can be divided into various clusters according to different factors
- Our system tends to predict when the chances of occurrence of heart disease are more, prior to the actual occurrence, unlike other systems, which generate an alert only when the extreme values have been attained.
- Our system divides the data according to factors like age and sex.
- Deep Neural Network has been used in our system, which hasn't been employed in any other paper dealing with heart disease prediction, yet. DNN has the highest accuracy among all.
- Our system uses all the viable machine learning and data mining algorithms.

3.3 MACHINE LEARNING ALGORITHMS

In this work, eight machine learning algorithms which include Random Forest, SVM, Decision Tree, Naïve Bayes, ANN, KNN, Logistic Regression and Deep Neural Network was employed. These algorithms are used to unfold a prediction system which will analyze and predict whether the particular patient is pertaining to any heart disease or not with best accuracy.

3.3.1 Support Vector Machine

1. The main objective of the support vector machine algorithm is finding hyperplane that distinctly plots and classifies data that is plotted in an N-dimensional space.

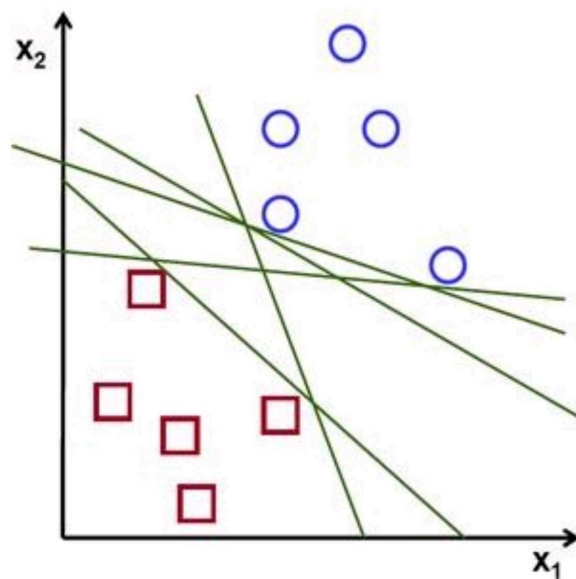


Fig.3.1: Possible hyperplanes

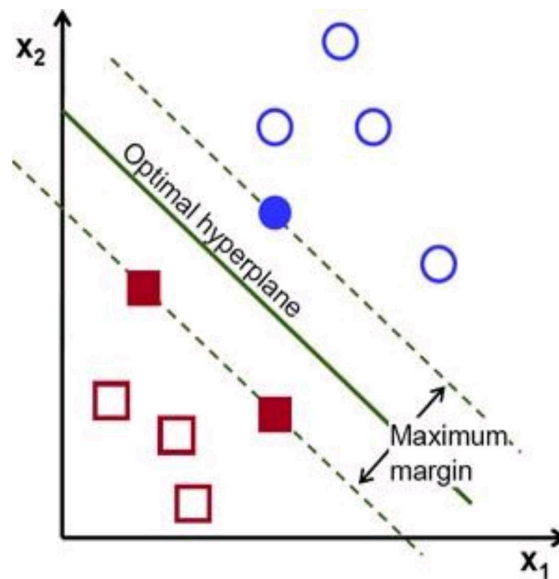


Fig.3.2: Hyperplane with maximum margin

The two classes are separated by choosing hyperplanes from the possible one. Our major objective here, would be to find the plane containing the maximum margin of all the planes, i.e. finding the data points with max distance between the two classes. When we maximize this margin, the distance provides us with more accurate classification when future data points are plotted.

SVM are basically of two types:

1. Linear SVM: It gives us a linear hyperplane that distinguishes the classes. The objective for us is to maximize the distance of the hyperplane to the point of the classes should be maximized.
2. Non-Linear SVM: It has a non-linear hyperplane, and depicts a graph which is closer to the real-world scenario.

3.3.2 Naïve Bayes

It is a machine learning technique that works on the strategy of the Bayes' Theorem. It basically assumes that there would be no attributes dependent on each other. It is a group of algorithms that have a common principle that every feature is independent of the other. Bayes' Theorem tells us the probability of an event that will occur when another event has already occurred. The mathematical equation is:

$$\text{Probability}(a|z) = (\text{Probability}(z|a) * \text{Probability}(a)) /$$

Probability(z)Where

- **Probability(a|z):** Gives us the probability of a (the hypothesis) gives the data is a.
- **Probability (z|a):** Gives the probability of the data when the hypothesis is true.
- **Probability (a):** Regardless of the data, the hypothesis is said to be true.
- **Probability (d):** Regardless if the data, the probability of the hypothesis is given.

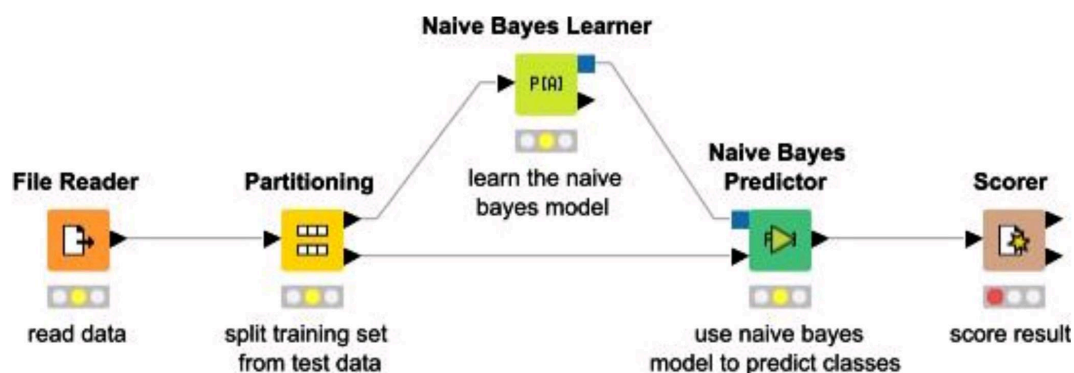


Fig.3.3: Naïve bayes

3.3.3 KNN

K nearest neighbors abbreviated as KNN is an algorithm that clusters data into classes and then classifies it as per their similarity measures. Classification is based on the majority of the votes to its neighbors. Data is assigned to the classes that have the nearest neighbors. As we increase the number of nearest neighbors, i.e. the value of k , the accuracy might increase. KNN is broadly used for pattern recognition and statistical prediction.

It divides the data into clusters, based upon the distance from the nearest neighbor.

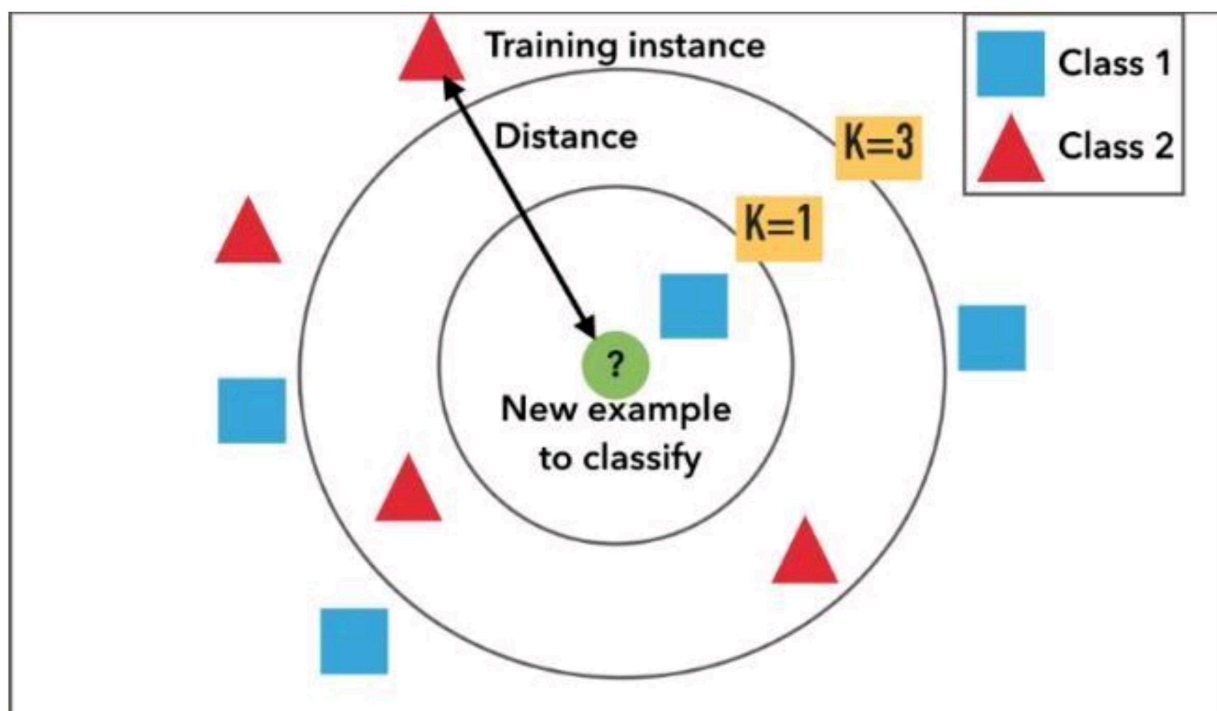


Fig.3.4: KNN clustering

3.3.4 Decision Tree

Decision Tree is one of the supervised learning algorithms. In this the data is continuously split based on a certain parameter after which we end up getting the decision nodes and the leaves. What makes it different from the other supervised algorithms is that it can also solve the regression and classification problems easily. The main aim is to create a system that can predict the results that we desire just by learning the decision rules from the prior data, i.e., the training set.

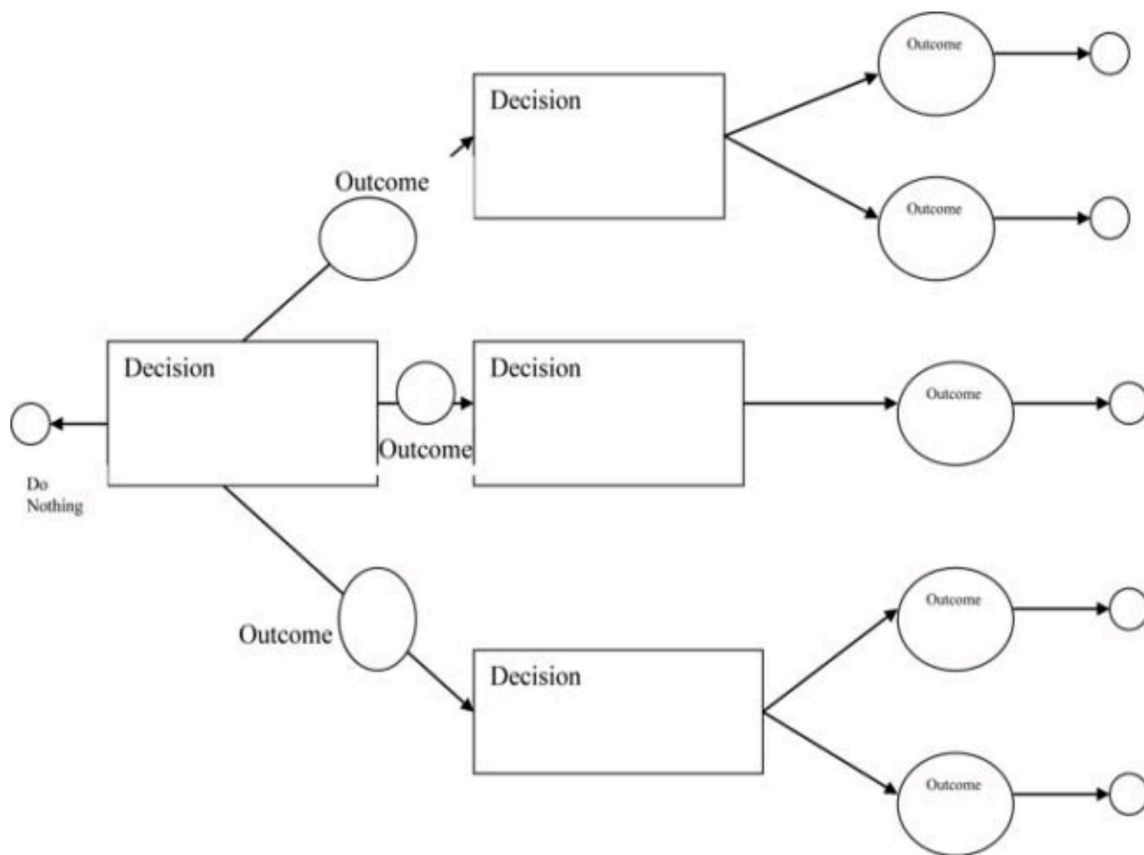


Fig.3.5: Decision tree

3.3.5 Random forest

Random Forest, just as the name suggest creates a number of random forest due to which it is also called the random decision forest. It is one of the supervised learning algorithms. It builds random forests that are basically just a group of decision trees. It is mostly trained with the bagging method as it is the most efficient. What the bagging method does is, it combines all the learning models which in turn helps us with the overall result. Just like decision trees, it can be used for both, regression and classification problems.

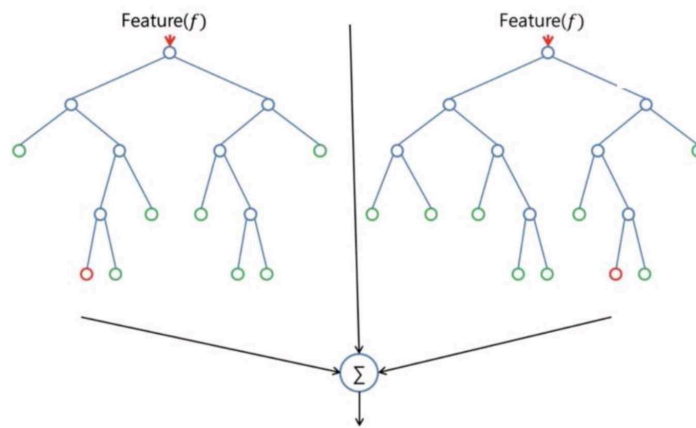


Fig.3.6: Random forest with two trees

3.3.6 Logistic Regression

Logistic Regression is rather a predictive analysis. It is used to analyze the relation between the data among which one is a dependent variable and the other one is independent. What logistic regression does is, it rounds off all the values of the results to the closest binary value for the ease of prediction. This can sometimes cause a few errors, and not predict very precisely. In order to remove this error, multiple regression is used which helps in getting the closest to the

true result. Logistic Regression uses complex cost functions such as the 'Sigmoid function'. In machine learning sigmoid is used to predict the probabilities.

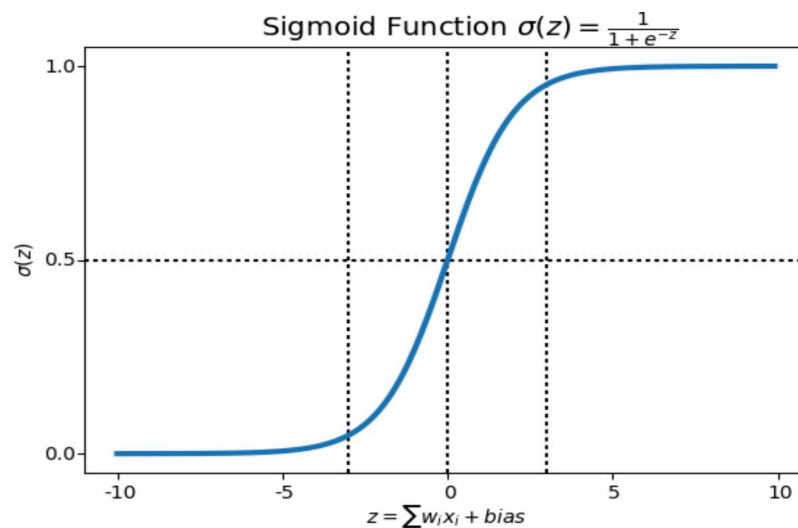


Fig.3.7: Sigmoid Function Graph

3.3.7 Artificial Neural Network (ANN)

An artificial neural network abbreviated as ANN is a model that works as the human brain or neural network i.e. the neurons. It is called a computational model that processes all the complex data. It isn't given any task specific goal, but it learns from the examples or data that is given to it just like the brain. It is based on a collection of nodes called the artificial neuron. More the no. of neurons, better the system. The neurons transmit signals from one to another making a proper connection which resembles the human neural network. A neural network has the following 3 layers:

- Input layer – It consists of the complex raw info that we feed to the neurons.

- Hidden layer – These are the computational layers which take the input and the weight of a node from the previous layers, processes it with the activation function, and sends the output to the next layer.
- Output layer – This depends of the output of the hidden layers, and the functions taking place in there.

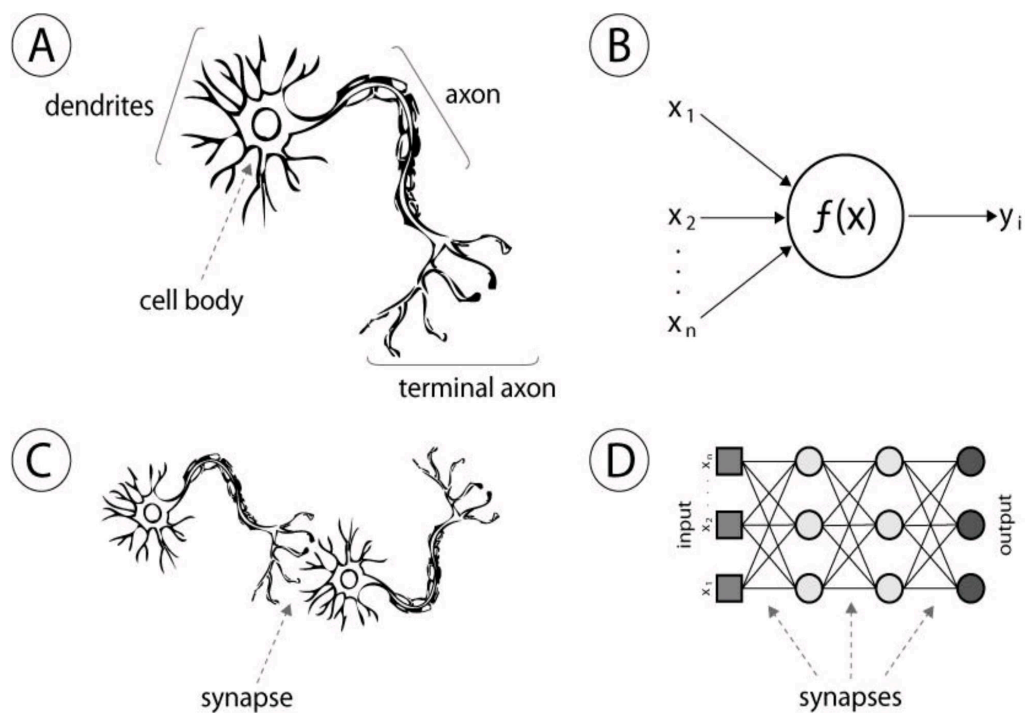


Fig.3.8: ANN

The basic computational unit is the neuron. It receives inputs from the sources provided, and each input carries a weight, which is given according to the relative importance of all the other inputs. Then the function is applied to it as shown in figure 3.8.

This function is called the **Activation function** which basically introduces non-linearity to the function and sends it in the form of the output. The function used in our module is the **sigmoid**

which takes inputs as real values and further fits them in the range of 0 to 1. The equation is: $\sigma(x) = 1 / (1 + \exp(-x))$

3.4 Xg-booster

Xg-boosters is shorting's Extremes Gradient Boosting. Xg-Boost are ensembles Methods depending on Decision-trees that developed Out a Stronglylearnings from Several Weeks learnings. Xg-Boost Algorithms are usings boosters the performing of the Models that is usings to providing good accuracies. In Decisions Trees are Developing Sequential. Weights are Randomly assigning to Each-all Features that are Independents Each Other's and feedings too Decisions-trees that Predicting the Results. The Weights of Wrongs Predictions Features Decisions-trees are increasing thus Features are sending to the Nest Decision trees. The correctness predictions Weights Features inside Decisions-trees are reducing. This processing are continuously Sequentially until corrections Results are predictions. Systems These independent Classifiers then combines Producers a strongly and More precisions Models. Booster is usings for Solving both regress and classifiers Problems.

3.5 Ada-boost:

Ada-Booster are shootings for Adaptiveness Boastings are a wider accepting booster Techniques that's are combinations Multiplex Weaks Classifier's to building a strongly classifieds. that has assignees by developing's a Models are using a Series of weeks Models. Ada-Booster developed a binaries Classifiers' selectees Trainings Subbranches Randomly and builds a developing. Models. That's iteratively Trains the Ada-Booster Machines Learnings Model by Selected the Trained Sets depends in accuracies Predicts of the latest Traineds. That has been assigning the highest Weights to Erroneousness classifiers Observer so that is forward Iterations thus Observer so that's having a highest Prospect classified. That's completely to correcting the Errors Presents in starting Models. assigning Weights too bee Trainings classifieds is Every-all Iterations According to the accuracies of the classifiers. The Must Acerating's Classifying's will be getting.

CHAPTER 4

SYSTEM DESIGN

This chapter gives the system architecture of our project towards heart disease analysis using Machine Learning. Followed by the architecture, we give the Data Flow Diagram and Data set used.

4.1 SYSTEM ARCHITECTURE



Fig.4.1: System Architecture

The above architecture shows only 3 machine learning algorithms. But our work here uses a total of 8 algorithms. For the training set, we included 80% of the data, and for the testing we used 20% of the data. We test the model with all the 8 algorithms and check the accuracy. The one that best fits, and gives us the most accuracy is the one which can be used to for further work in this field.

4.2 DATA FLOW DIAGRAM

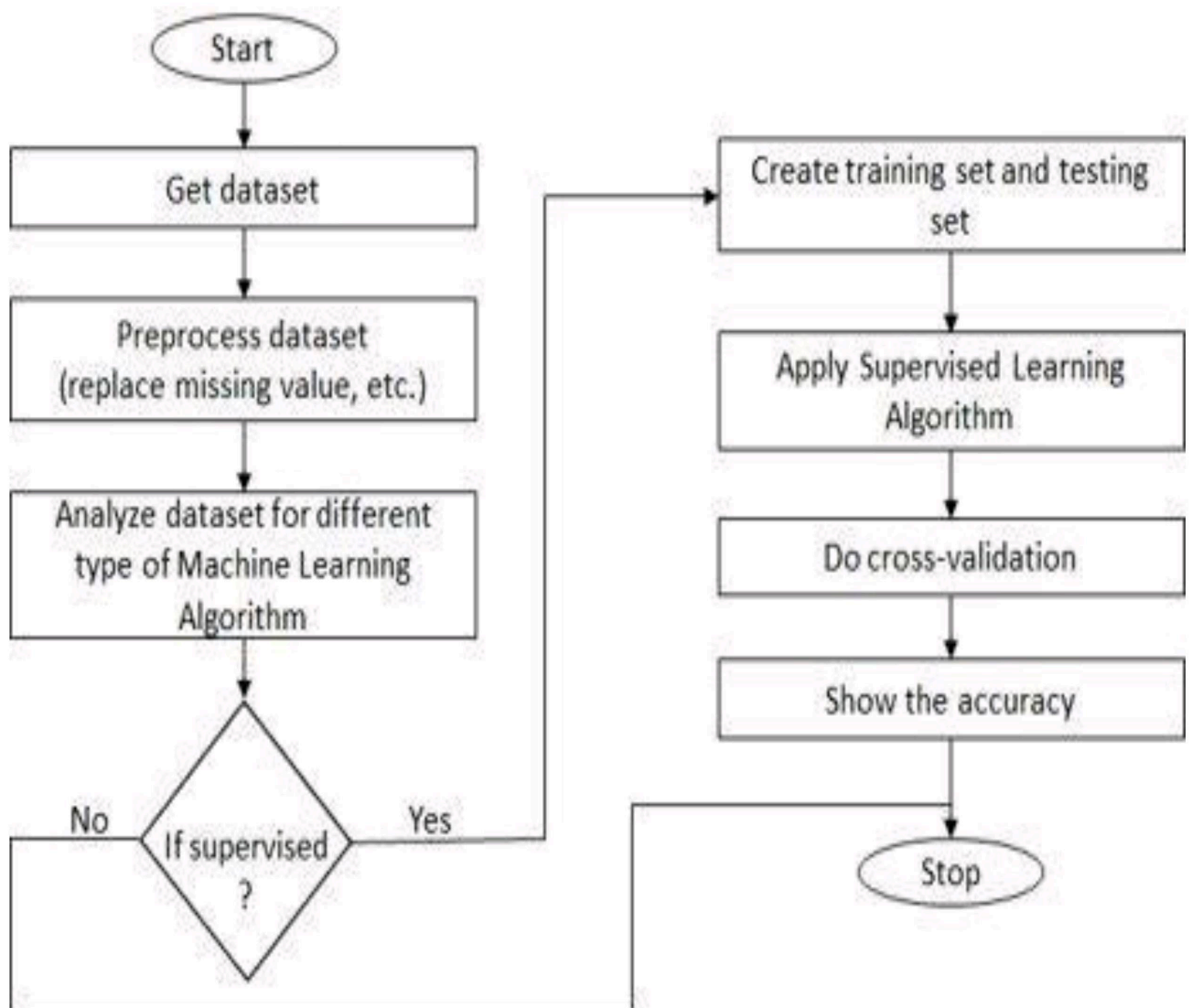


Fig.4.2: Data flow diagram of

the workflow of our system: Explanation

- 1) Our system is first divided into training and testing data..

- 2) The training data is 70% and is given supervised inputs and outputs.
- 3) The testing data is 30% and shows us how well the system is trained.
- 4) The dataset we have chosen consists of 13 attributes according to which various algorithms perform their calculations and approximations.
- 5) The system starts with first pre processing of the dataset we have fed to it.
- 6) It studies and analysis it, and then applies the required machine learning algorithm.
- 7) If it finds that the dataset is supervised, it will separate it into training data and testing data.
- 8) Otherwise it will stop.
- 9) The algorithms we are using, are all supervised.
- 10) After the application of algorithm, internal validation is done .
- 11) Accuracy is printed in the code itself.
- 12) Different accuracy is given for different algorithms.
- 13) We compare the accuracies of all the algorithms and the algorithm that gives the highest accuracy is the one, which is chosen for prediction, eventually.
- 14) In our system, the algorithm that gets the highest accuracy rate is DNN.

CHAPTER 5

CODING & TESTING

5.1 SOFTWARE REQUIREMENTS

1. **Anaconda**- It is an open source software available to us which enables us to easily code in using python or R on different operating systems such as the windows, Linux, and Mac OS. It has millions of users worldwide, and is well known as the industry which helps us in developing systems, testing them, and training the machines. This further enables us to:

- Manage all the imported libraries, their dependencies, and the environments of developing with Anaconda.
- In developing techniques to train our machine with TensorFlow, scikit-learn, etc.
- Analyze the datasets and manipulate them with Dask, NumPy, pandas, and Numba
- Visualize or plot the results with Matplotlib, Holoviews, Bokeh, and Datashader.

It also provides us with jupyter notebooks which has all the in-built libraries embedded in the already. This eases our coding stress, and also helps us code with more efficiency.

2. **Python**: The most abundantly used general level programming language. It is used for both a small scale and big scale systems. It can easily be interpreted. It is said to support multiple programming paradigms. It includes features of procedural, object-oriented, and functional programming together. It is already garbage-collected which makes it more efficient.

3. **Numpy**- It is a python programming library. This basically helps us deal with large datasets, matrices, and multi-dimensional arrays. It also provides us with a number of mathematical functions which help us and ease the calculations. It is an open-source software available to all.

4. **Pandas**- It is a library that is written in python language. It helps us with the analyzing of data. It also provides us with tools and functions to manipulate a large amount of data.

5. **Sklearn**- It is a library used in machine learning in python programming language. It mainly helps us with the classification of data, regression of models, and in clustering algorithms. These algorithms include SVM, random forest etc.

6. **Tensorflow**- It is a highly know open-source software, which is free and available to all. It is used for differentiable programming with a large number of tasks. It is used against largedata sets to help us with the dataflow, and its manipulation. It is a basically a math library withvarious features.

5.2 OBJECTIVES AND TYPES OF TESTING

Testing is performed so that we can locate problems. It is the method of identifying all possible faults of any system. Also, it acts as method to test the functionality of all the individual components and the system as a whole. It is the process of verifying that the hardware system performs its functionalities and meets user needs and expectations. Testing ensures that even if theproduct fails, it fails in a controlled condition that can be managed later. There are various types of test. Each of the test is used to address a particular testing needs.

5.2.1 UNIT TESTING

This testing includes the creation of test scenarios that verify that program is working in the way it supposed to. In this type of testing, the objective is to validate that each module or unit is performing the tasks that it is supposed to do. A unit is the smallest part that has to be tested.

5.2.2 INTEGRATION TESTING

These tests are made just to check the different parts after they have been integrated to determinewhether they are functioning properly as a unit or not. This type of testing is driven by events andis more curious about the outcomes of the component as a whole. Integration tests show that although the components are working properly as individual but also the combination of components is also working properly.

5.2.1 FUNCTIONAL TESTING

These kinds of tests provide systematic demonstration of the functions tested and the proof of their functioning being successful. Functional testing is targeted at a valid input, invalid input, functions, output, procedures. The preparation on functional tests is focused around the requirements, key functionalities, different and extreme test cases

5.2.2 SYSTEM TESTING

This type of tests tells us whether the whole system after all its parts have been joined together meets the need of shareholders. It checks and confirms the known and expected results. These kinds of testing are done to check whether the system delivers what it was supposed to. It checks whether all the parts work well with each other or not.

5.2.3 WHITE BOX TESTING

It is done in a way such that the tester has information about all the components, workings, structure and architecture of the hardware/software. It is used for deep level testing to ensure that those places that are not accessible by a black box test are covered in white box testing. It basically means that there shall be no blind spots in the system.

5.2.4 BLACK BOX TESTING

Here the testing that is done on the software/hardware by someone who has no information on how to operate it. Before a product is developed some documentation is done, they include requirement analysis document that has all the details about what a product must be. Now these tests are based on those sources only. Imagine a black colored box, you cannot see into it. You know nothing about what it contains. This is exactly how the testing is done.

5.3 TESTING IN MACHINE LEARNING

Very easily put, machine learning is a type of system/application which is based upon models that use prediction and analysis. Building these systems is a hard task, but testing and validating the entire system is even more difficult. Traditional methods of testing are always based on stationary inputs. The output will also be fixed as the input remains fixed, but in the case of machine learning, the system is built along with the input, which will change or modify as we come to know more and more about it, therefore unlike the traditional system, the output here will never remain fixed. Hence, the strategies for testing a machine learning system have to be a bit different.

Below are some of these strategies mentioned that can be used in testing the machine learning systems:

1. Development of training datasets:

Here, a subset of dataset is used for training purpose, which means it is used to train the system to obtain the given prediction. It is supervised, i.e., the output is given along with the input.

2. Development of testing data sets:

The testing data set is a training data subset that is built in an intelligent way to check how robustly our system has been trained and to check all the combinations that are possible. The resultant model will be finely tuned according to the outputs of the testing dataset.

3. Development of validation and correction test suite:

This is based on algorithms and test datasets. For eg, in our system scenarios consist of clustering results based upon different factors/attributes and creating profiles of risk depending upon behaviors and demography.

4. Understanding the algorithm:

All of this is dependent upon calculations and how different algorithms use different calculations. Some algorithms like regression based algorithms give some numeric results regressively, or continuously. On the other hand, there are some algorithms that calculate the results by dividing the outcomes into different parts or behaviors, and then some algorithms create multiple layers between the input and output layer.

Continuous numeric variables such as return on investment.

5. Using Statistics for conveying the results

Outcomes are generally predicted in the form of working model, or a numeric value or a working interface or something like that. But in machine learning, the use of bars, histograms and other statistical diagrams is used to depict the prediction or analysis in a more comparative or mathematical way.

The machine learning models will usually have approximate, and not actual results upon validation. In conclusion, software testing is as crucial and important a task in machine learning, as is in any other traditional system, but unlike those systems, our testing is based on more dynamic factors and generally will produce a relative or approximate result which can be best shown statistically.

5.4 TEST CASES

Table 5.1 Test Cases

SR NO.	TEST CASE NAME	TEST CASE DESCRIPTION	EXAMPLE TEST CASE	EXPECTED OUTPUT						
1.	Equivalence Partitioning	This divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.	Our dataset can be divided into two classes on the basis of the number mapped with state. The valid partitions are : <table><tr><th colspan="2">Mapped State</th></tr><tr><th>Valid</th><th>Invalid</th></tr><tr><td>0,1</td><td>Any no. other than 0,1</td></tr></table>	Mapped State		Valid	Invalid	0,1	Any no. other than 0,1	If valid: Successful If invalid: Unsuccessful
Mapped State										
Valid	Invalid									
0,1	Any no. other than 0,1									
2.	Boundary Value Analysis	In this we focus on the boundary values of the various valid and invalid inputs.	We want to test the value of the age to be a positive integer.	If valid: Successful If invalid: Unsuccessful						
3.	Data input	To check what happens when data is inputted in the wrong format.	Instead of giving the input in 0's and 1's. The data is fed into the system as alphabets. For example Sex of the person.	The code must throw an error pertaining to change of format of the data inputted.						

			For thalach If it is <120 (0) = lowthalach Else if it is >140 (1) =high thalach Else (2) = normal thalach	Depending on thevalue of thalach the testing of data set takes place. Ifnegative value is entered then throw error.
			For trestbp If <80 and >120 =Normal Else if < 120 = High Blood pressure	If negative valueis entered then throw error.

CHAPTER 6

RESULTS AND ANALYSIS

This chapter discusses the comparative analysis of different machine learning towards the prediction of heart disease. In addition we also have give the graphical representation of data set on basis of age, Serum cholesterol, BP and so on .

6.1 DATA SET

For eg: Our system here, uses a dataset to predict the occurrence of heart diseases by dividing the dataset into test data and training data. 70% data is used for training and the rest 30% is used for testing. Now, the input used in our case is the attributes we have used that correspond to factors that result in heart diseases. The output would be a binary digit indicating whether a person is susceptible to heart diseases or Not. The algorithms used in the system for prediction, are learning algorithms which will always change over periods of time based on various input factors . Therefore, the results might change when we learn even more about the data fed as input.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2	1
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2	1
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2	1
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2	1

Fig.6.1: Data set used

For our system, we have taken dataset that was publicly available on Kaggle for predicting the heart diseases. The parameters used as input for data analysis using Machine learning algorithm are as follows:

1. age: which is taken in years
2. sex: 1 for male, 0 for female
3. cp: short for chest pain
4. trestbps: blood pressure taken when the body is resting
5. chol: level of cholesterol
6. fasting blood sugar: (1 for true; 0 for false)
- resting electrocardiographic
7. resultsValue 0: normal

Value 1: having the ST-T wave which is not normal

Value 2: shows the probability of having left ventricular hypertrophy

8. thalach: the max level of heartbeats achieved by heart
9. exercise induced angina: 1 for yes, 0 for no
10. oldpeak: the depression by exercise compared to the one at rest
11. slope: peak of the ST segment during

exerciseValue 1: no slope

Value 2: straight line Value 3: down

6.2 METRICS FOR PERFORMANCE ANALYSIS

The implementation of the machine learning algorithms are compared to identify the best algorithm for the prediction of the occurrence of flood. Before the results, a brief explanation about the confusion matrix, accuracy and other terms is given below.

Confusion Matrix: It is a binary classifier. A confusion matrix can be of any size depending upon the different number of parameters inputted (labels in our case). The confusion matrix in our case is a 2x2 matrix.

Table 6.1 Confusion Matrix

TP	FN
FP	TN

TP-True Positive, FN-False Negative, FP-False Positive, TN-True Negative

TP and TN denote the number of instances which have been correctly classified as no flood occurrence and flood occurrence respectively. FP and FN signify the number of instances which have been wrongly classified as no flood occurrence and flood occurrence respectively.

Accuracy: The accuracy can be calculated with the help of the formula given below.

$$Accuracy = (TP + TN) / (TP + FP + FN + TN)$$

Precision: Termed as the positive predictive value, it is calculated as given below.

$$Precision = TP / (TP + FP)$$

Recall: Also called as sensitivity, it is calculated as given below.

$$Recall = TP / (TP + FN)$$

$$F1 \text{ score} = 2 * (Precision * Recall) / (Precision + Recall)$$

MCC: Matthews Correlation Coefficient considers all four divisions of the confusion matrix when calculated. MCC lies within the range -1 to +1 where a model with a positive score is considered to be perfect whereas the negative score is poor. This makes this metric really useful as it is easy to interpret

$$MCC = (TP * TN - FP * FN) / \sqrt{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))}$$

6.3 PERFORMANCE OF MACHINE LEARNING ALGORITHMS

6.3.1 SVM RESULTS

ACCURACY: 0.8852

RECALL: 0.8666

PRECISION: 0.89

MCC: 0.84699

Table 6.2 SVM Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:26	FN:28
Predicted Value	False	FP:3	TN:4


```
# Support Vector Machine Linear

model = train_model(x_train, y_train, x_test, y_test, SVC, C=0.05, kernel='linear')

Confusion matrix
[[26  4]
 [ 3 28]]
Train accuracy: 83.06%
Test accuracy: 88.52%
```

Fig.6.2: SVM Results

6.3.2 NAÏVE BAYES RESULTS

ACCURACY: 0.8852

RECALL: 0.9

PRECISION: 0.8666

MCC: 0.2104

Table 6.3 Naïve Bayes Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:26	FN:27
Predicted Value	False	FP:4	TN:4

```
#Gaussian Naive Bayes
```

```
model = train_model(x_train, y_train, x_test, y_test, GaussianNB)
```

```
Confusion matrix
```

```
[[26  4]
```

```
 [ 4 27]]
```

```
Train accuracy: 82.64%
```

```
Test accuracy: 86.89%
```

Fig.6.3: Naïve Bayes Results

6.3.3 KNN RESULTS

ACCURACY: 0.639

RECALL:0.4666

PRECISION:0.76

MCC:0.7465

Table 6.4 KNN Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicated Value	True	TP: 14	TN: 16
Predicated Value	False	FP: 6	FN: 25

```
# KNN
```

```
model = train_model(x_train, y_train, x_test, y_test, KNeighborsClass  
ifier)
```

```
Confusion matrix
```

```
[[14 16]
```

```
 [ 6 25]]
```

```
Train accuracy: 76.86%
```

```
Test accuracy: 63.93%
```

Fig.6.4: KNN Results

6.3.4 DECISION TREE RESULTS

ACCURACY: 0.8033

RECALL: 0.7666

PRECISION:

0.821MCC:0.271

Table 6.5 Decision tree Confusion matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:23	TN:7
Predicted Value	False	FP:5	FN:26

```
# Decision Tree
model = train_model(x_train, y_train, x_test, y_test, DecisionTreeClassifier, random_state=2606)
```

```
Confusion matrix
[[23  7]
 [ 5 26]]
Train accuracy: 100.00%
Test accuracy: 80.33%
```

Fig.6.5: Decision Tree Results

6.3.5 RANDOM FOREST RESULTS

ACCURACY:

0.7734RECALL:0.8

PRECISION:0.8125

MCC:0.6607

Table 6.6 Random Forest Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:24	FN:23
Predicted Value	False	FP:8	TN:6

```
#####      using random forest model

# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=7)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

for i in range(len(y_pred)):
    if y_pred[i]>=0.5:
        y_pred[i]=1
    else:
        y_pred[i]=0
```

Fig.6.6: Random Forest Results

6.3.6 LOGISTIC REGRESSION RESULTS

ACCURACY:0.906

RECALL:0.8666

PRECISION:0.9285

MCC:0.6394

Table 6.7 Logistic Regression Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicated Values	True	TP: 26	TN: 29
Predicated Values	False	FP: 2	FN: 4

```
# LOGISTIC REGRESSION
```

```
model = train_model(x_train, y_train, x_test, y_test, LogisticRegression)
```

```
Confusion matrix
```

```
[[26  4]
```

```
 [ 2 29]]
```

```
Train accuracy: 83.47%
```

```
Test accuracy: 90.16%
```

Fig.6.7: Logistic Regression Results

6.3.7 ANN RESULTS

ACCURACY: 0.8852

RECALL: 0.8666

PRECISION:0.89

65MCC:0.84699

Table 6.8 ANN Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:26	FN:28
Predicted Value	False	FP:3	TN:4

```
# USING ANN Multilayer Perceptron Classifier

from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier

ann_clf = MLPClassifier()

parameters = {'solver': ['lbfgs'], 'alpha':[1e-4], 'hidden_layer_size':(9,14,14,2), 'random_state':[1]}

acc_scorer = make_scorer(accuracy_score)
grid_obj = GridSearchCV(ann_clf, parameters, scoring=acc_scorer)
grid_obj = grid_obj.fit(x_train, y_train)
ann_clf= grid_obj.best_estimator_
```

```
y_pred_ann = ann_clf.predict(x_test)
from sklearn.metrics import confusion_matrix
cm_ann = confusion_matrix(y_test, y_pred_ann)
cm_ann
```

```
array([[26,  4],
       [ 3, 28]])
```

```
ann_result = accuracy_score(y_test, y_pred_ann)
ann_result
```

```
0.8852459016393442
```

Fig.6.8: ANN Results

6.8 XG-boost

ACCURACY: 0.918

RECALL: 0.960

PRECISION:0.857

MCC:0.905

Table 6.9 Xg-boost Confusion Matrix

		Actual Value	Actual Value
		Positive	Negative
Predicted Value	True	TP:32	FN:1
Predicted Value	False	FP:4	TN:24

```
# Tuning
```

```
tuned_model = xgboost.XGBClassifier(gamma=3, learning_rate=0.05, use_label_encoder=False)
tuned_model.fit(X_train, y_train, eval_metric='error')
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=3, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.05, max_delta_step=0,
              max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints=()), n_estimators=100, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', use_label_encoder=False,
              validate_parameters=1, verbosity=None)
```

```
print("Training accuracy: ", accuracy_score(tuned_model.predict(X_train), y_train))
y_pred_tuned = tuned_model.predict(X_test)
print("Test accuracy: ", accuracy_score(y_pred_tuned, y_test))
```

Training accuracy: 0.910377358490566

Test accuracy: 0.8351648351648352

Fig.6.9: Xg-boost Results

6.10 Ada-booster:

ACCURACY: 0.829

PRECISION: 0.860

RECALL: 0.841

MCC: 0.841

Table 6.10 Ada-boost Confusion Matrix

		Actual Value	Actual Value
		Positive:	Negative
Predicated Value	True	TP:26	TN:6
Predicated Value	False	FP:7	FN:37

```

for j in range(5,15):
    for k in range(5,13):
        x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)
        clf = RandomForestClassifier(n_estimators=i,max_depth=j,max_features=k)
        clf.fit(x_train,y_train)
        prediction = clf.predict(x_test)
        accuracy = accuracy_score(prediction,y_test)
        cm = confusion_matrix(prediction,y_test)
        prfs = precision_recall_fscore_support(prediction,y_test)
        if accuracy > maxin:
            maxin = accuracy
            n_estimators = i
            max_depth = j
            max_features = k
            max_cm = cm
            max_prfs=prfs

print(str(i)+" "+str(j)+" "+str(k)+" "+str(maxin))
print("\n")
print('Confusion Matrix: ',cm)
print("\n")
print('Precision: ', prfs[0])
print('Recall: ', prfs[1])
print('Fscore: ', prfs[2])
print('Support: ', prfs[3])

14 14 12 0.8947368421052632

Confusion Matrix: [[24  6]
 [ 9 37]]

Precision: [0.72727273 0.808434783]
Recall: [0.8 0.808434783]
Fscore: [0.76190476 0.83146067]
Support: [30 46]

```

Fig.6.10: Ada-boost Results

Table 6.11 Accuracy Comparisons of Algorithms

ML Algorithms	Accuracy
Xg-boost	81.3%
Ada-boost	72.66%
SVM	80.2%
KNN	67.21%
Decision Tree	75.8%
ANN	72.66%
Random Forest	79.1%
Logistic Regression	79.1%

6.4 RESULTS FROM THE TEST DATA SET:

We have done extrapolation of data set in regards to heart disease on basis of various factors which are Sex, Age, Blood pressure, Serum cholesterol and so.

% of men and women in our dataset

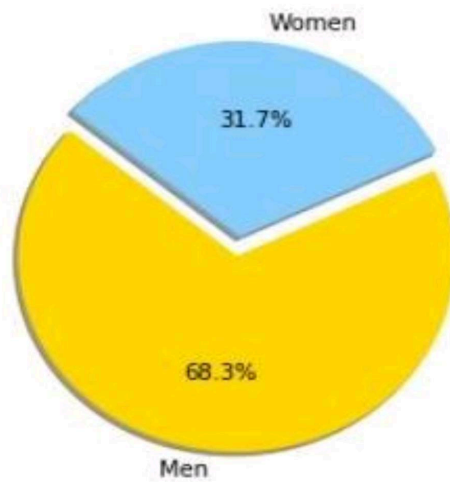


Fig. 6.11: Percentage of men and women in our dataset

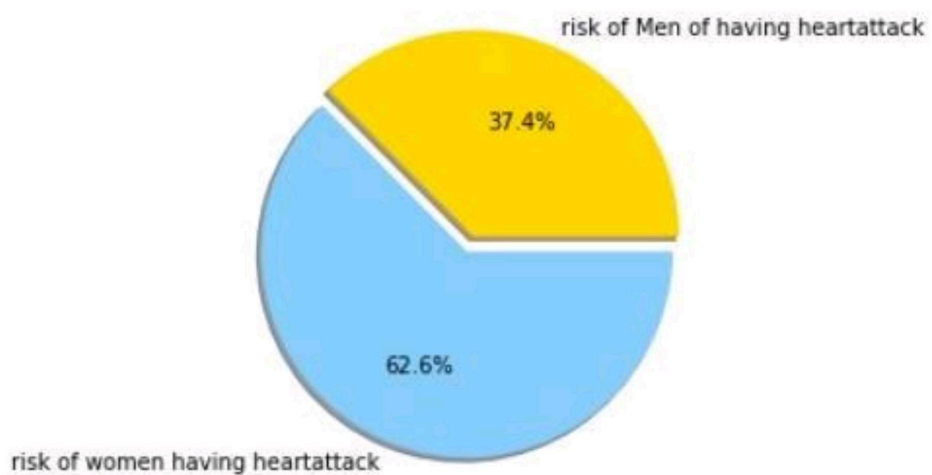


Fig. 6.12: Risk of men and women having a heart attack

ANALYSIS OF POSITIVE HEART ATTACK IN MEN OUT OF TOTAL MEN

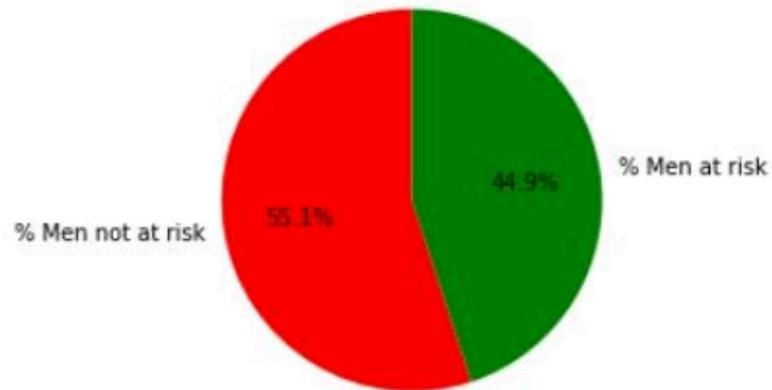


Fig. 6.13: Analysis of positive heart attack in men out of total men

ANALYSIS OF POSITIVE HEART ATTACK IN WOMEN OUT OF TOTAL WOMEN

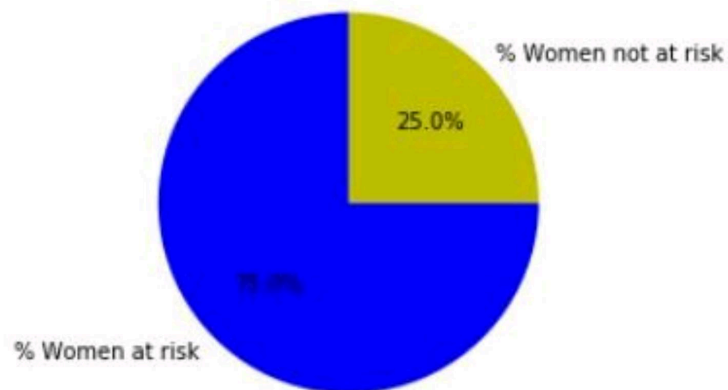


Fig. 6.14: Analysis of positive heart attack in women out of total women

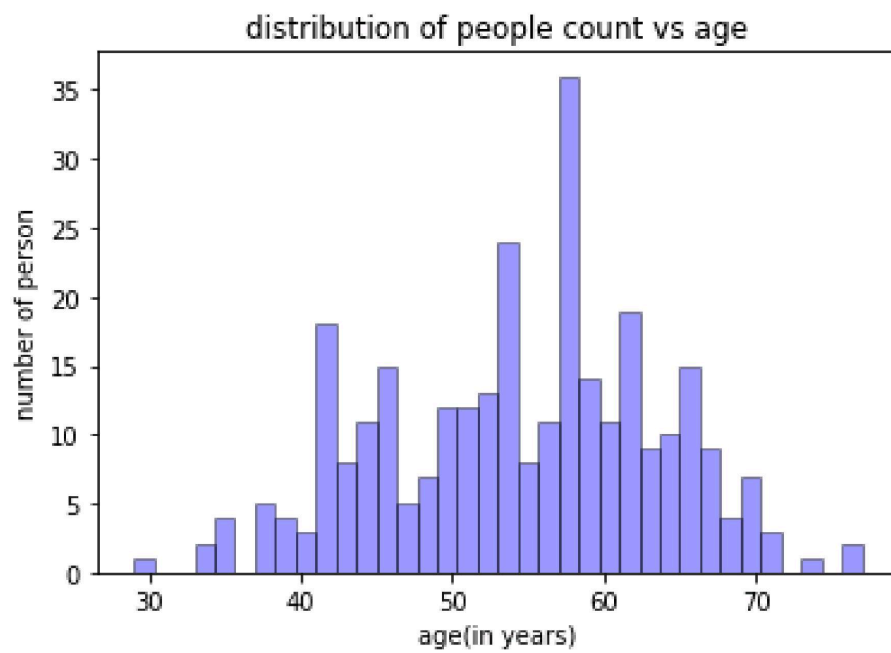


Fig.6.15: Distribution of people vs age

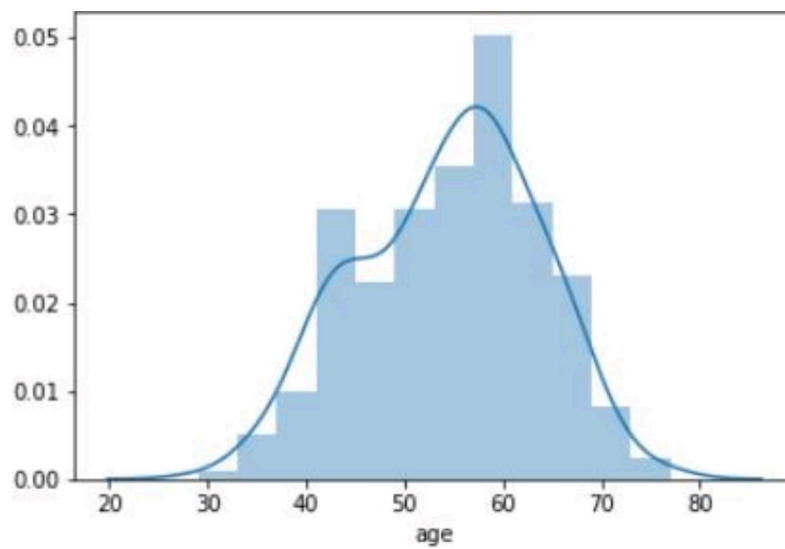


Fig.6.16: Age

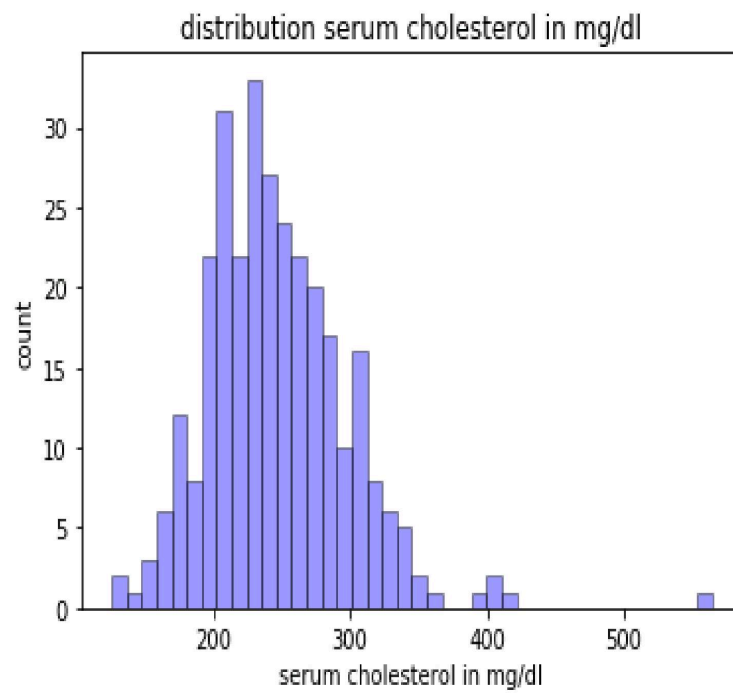


Fig.6.17: Distribution serum cholesterol in mg/dl

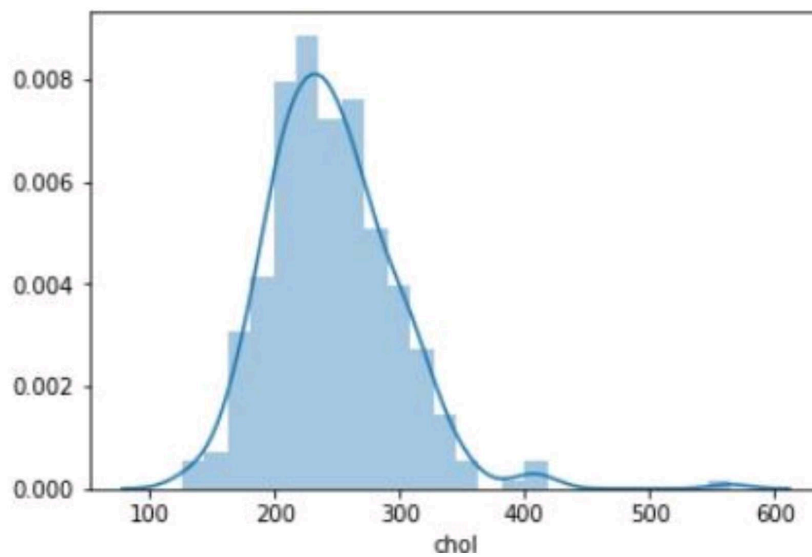


Fig.6.18: Cholestrol

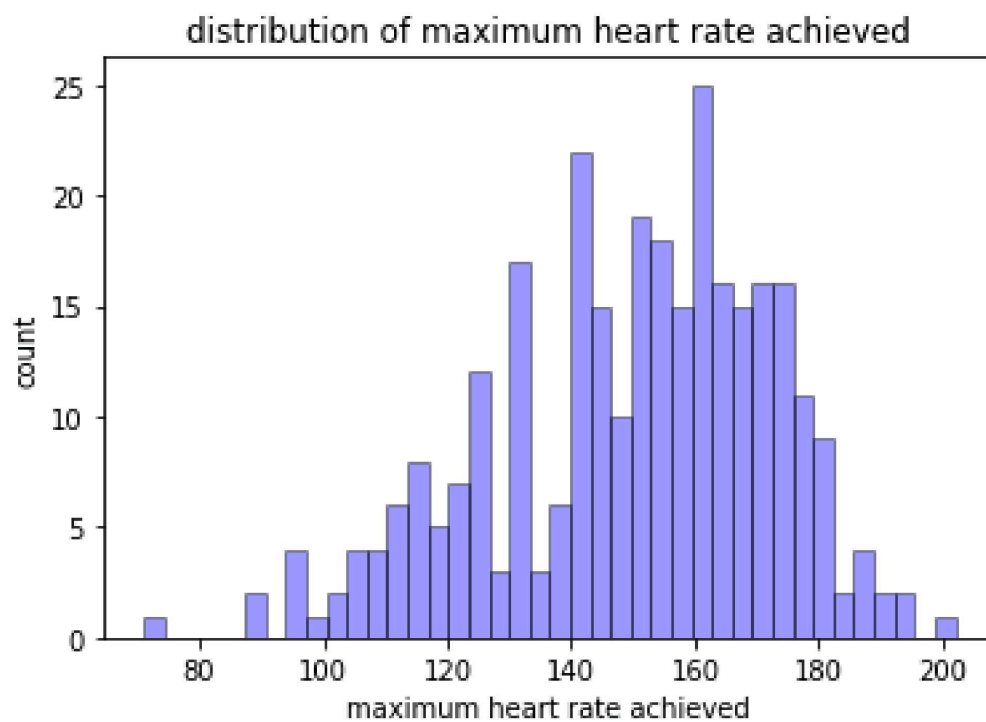


Fig.6.19: Distribution of maximum heart rate achieved

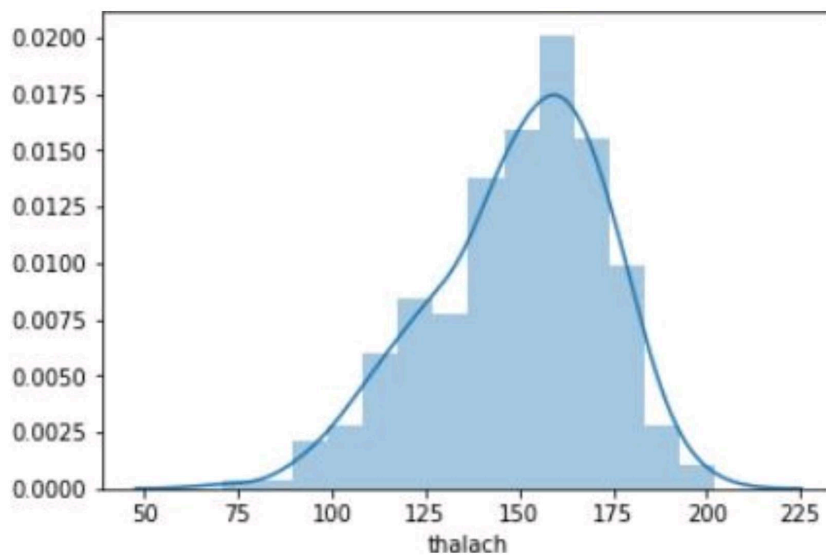


Fig.6.20: Thalach

distribution of resting blood pressure (in mm Hg on admission to the hospital)

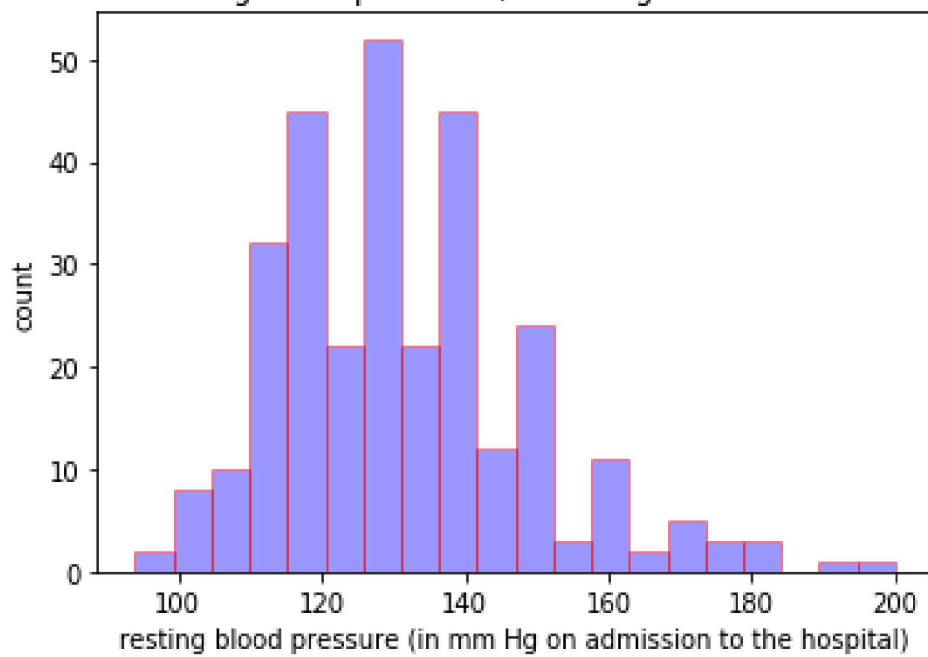


Fig.6.21: Distribution of resting blood pressure

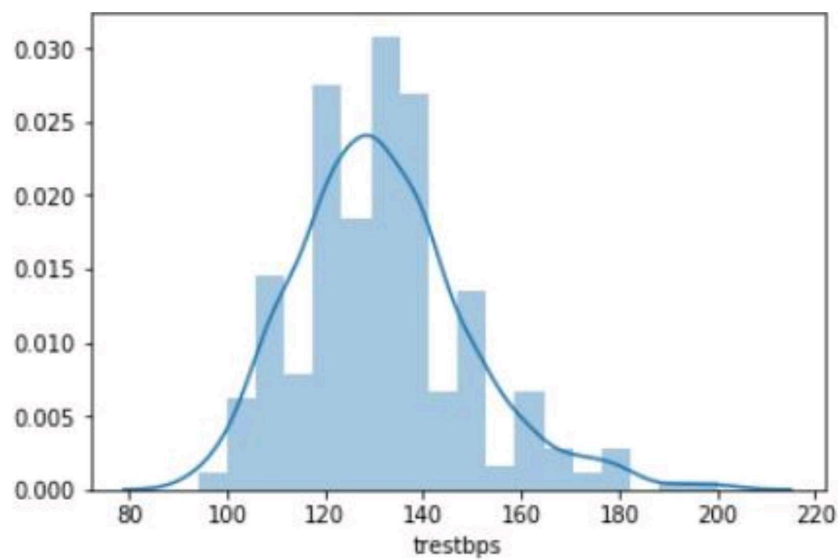


Fig.6.22: Trestbps

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUDING REMARKS

In Conclusion, XG-boost shows great potential in predictive analysis. As per the results of the so far analysis, XG-boost is the best algorithm that can predict the output as it has the highest accuracy. XG-boost shows great potential in predictive analysis when fed with a very large data set, and keeps improving the accuracy as and when you keep training it. Given more data sets to the system, it can train itself much better and would be of great use in predicting the heart diseases much ahead of it being critical. This when fed with a very large data set, and keeps improving the accuracy as and when you keep training it. Given more data sets to the system, it can train itself much better and would be of great use in predicting the heart diseases much ahead of it being critical.

7.2 FUTURE WORK

The work done here trains the system with a limited number of datasets. The machine learning algorithms become more accurate once they are fed with a huge number of data sets. So, this system can be trained with a huge number of data sets that would increase the accuracy in predicting the heart diseases. The analysis part of the system is done, and in order to be more useful can be integrated with electronic systems which give the system real-time inputs and would help the patient get the results then and there. There are multiple combinations of algorithms that can be tested against these data sets in order to yield better results.

REFERENCES

- [1] Aadesh kumar & Pankaj kumar Sharma D & Soni S (2012). validator Data-Minings for medical prognostic: inspections of heart disease predictions. Worldwides research paper of System Applications, 17(8), 43-8
- [2] Ankit Dewgan and meghu sharma S S (2015). modifieds data storage in instruments. worldwidesl researchpapers of system Applications, 46(11), 48-7.
- [3] Aldallal (2006). The instutions rule searching with the trainings and testings approachings of heart disease prediction. IEEE Transferrings in Information Technology in Biomedicine, 10(3), 344-43.
- [4] Brown N. Young T, Gray D. Skilful for heart disease prediction systems are using k-Nearest clustering in Bayes algorithms. Worldwides papers of Computer Science and Information Technologies, 5(9), 738-5.
- [5] Bashir S, Qamar U & Javed M Y (2014, November). thinkings founded Supportings structuredraft in preuditings HeartProblems In Internationally Conventions in knowledges Society (i-Society 2016) (p-p. 259- 64). IEEE. ICCRDA 2020 IOP Conf. Series: articles intellect technolgy 1023 (2022) 012022 IOP Popularize s.n:10.1087/1777- 9599X/1024/1/01207223 9
- [6] Chandel A M H & Linda (2013). CAD heart disease predictings models:K-Nearest for Heart read. BMJ open, 4(5), e005025.
- [7] Dangareese & Apteress, Magnusson PN K, Pedersen M, Ärnlov J & Ingelsson E (2012). Multilocus genetic risk scores for heart disease prediction. Arteriosulyes, theroums, and vasuallys biology, 35(8), 3257-82.
- [8] Folosam AR, Prineas Chandrans P (2020, Aprils).For Heart problems predictings using associatings classifieds. From 2020 in worldwides Mutliplexrepresentings on Automatives, in System, Communication, Control and Compressed Sensing (iMac4s) (pp. 40- 6). IEEE.
- [9] Gaana N, Young T, Gray D, Skene AKM & Hampton J R (1996). Inpatientdeaths from acute myocardial infarction, 1982-92: analysis of data in the Nottingham for heart attack register. BMJ, 315(7101), 159-64.
- [10] Gabbar A R, Prineas RJ, & Soler J T (1997). Body pulp distributings and self-reportings prevalence of blood-presures, of heart attack, and other heart problems for aged ladye

- [11] Jee S H AMH, & Linda (2019, Octubers). HDPS: for Heart disease prediction system. In 2012 Computing in Cardiology (pp. 557-60). IEEE.
- [12] Ordenez, Latha and R Subramanian. representings for best heart Disease Predicts Using and geneticings algorithm.s" worldwides papers of organic, and Medical Sciences 4.3 (2005).
- [13] Pilterus J, Johansson E & Manefjordeers (2018). Wireless body area network in heart attack detection [Education Corner]. IEEE anthomous and proprogramming magazine, 60(7), 88-99.
- [14] Patel S & Chauhan Y (2014). Heart attack detection and medical attention using motion sensing device in International Journal of Scientific and Research Publications, 4(1), 1-4.
- [15] Parthiban WilliamsonJ D (2005). Validitings of heart failure tappers in the Anti high-blood-presures and Lipids Treatments Predictings Heart Attack Trial (ALLHAT) participates assignments to Doxazosins and Chalestrols. Current controlled trials in cardiovascular medicine
- [16] Raihan M, Mondal S, Sikder G, Majumder M A & Ghosh K (2017, December). Smart works depend on heart disease (heart attack) predictings Systems using clinical data to approachings, a prototype design. In 2017 19th International Conference in Computer and Information Technology (ICCIT) (pp. 299-303). IEEE.
- [17] Soni j,Ansari U "Using Machine Learnings Techniques Tools for Predict Diabetes and Heart Diseases", 2018 5th International Conference on Frontiers of Signal Processing (ICFSP), pp. 150-154, 2018, September.
- [18] Takci H (2019). Improvement of heart attack prediction by the include the features selection methods. publishings in tarkies papers of mechanical Engineering & CSE, 26(2), 5510.
- [19] Ungalathavi Sharma and Meghu Sharma, "Predictings of heart problems using a hybrid tools for Data classifieds", 2017 3nd International Conference on Computing for Sustainable Globally Development
- [20] Wongastores J. Johanson "before heart disease predictings using Data tools", Computer Science & Information Technology Journal, pp. 555-69, 2017

APPENDIX

A. Source code.

```
# 1 importing essential libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import re

train = pd.read_csv("heart_disease_data.csv")

# loading data from heart.csv

train.dropna()

# deletes rows with empty cells

# Percentage of men and women in our dataset

label = 'Men','Women' sizes = [0.6831, 0.3168] colors = ['gold','lightskyblue']
explode = (0.1, 0)

# explode 1st slice

# Plot

plt.pie(sizes, explode=explode, labels=label, colors=colors, autopct='%1.1f%%',
shadow=True, startangle=140)

plt.title('% of men and women in our dataset\n\n\n')

plt.axis('equal') plt.show()

# Plot of men at risk vs women at risk

label = 'risk of Men of having heartattack','risk of women having heartattack' sizes
= [(0.449), (0. 75)] colors = ['gold','lightskyblue'] explode = (0.08, 0)

# explode 1st slice
```

```

# Plot

plt.pie(sizes, explode=explode, labels=label, colors=colors, autopct='%1.1f%%',
shadow=True, startangle=0) plt.axis('equal')

plt.show()

# percentage of men at risk

slices_hours = [207-93, 93] activities = ['% Men not at risk', '% Men at risk'] colors
= ['r', 'g','b',' y'] plt.pie(slices_hours, labels=activities, colors=colors, startangle=90,
autopct='%1.1f%%') plt.ti

tle('ANALYSIS OF POSITIVE HEART ATTACK IN MEN OUT OF TOTAL MEN')

plt.show( )

# percentage of women at risk

slices_hours = [96-72, 72] activities = ['% Women not at risk', '% Women at risk']
colors = ['y','b' ] plt.pie(slices_hours, labels=activities, colors=colors, startangle=0,
autopct='%1.1f%%') plt.title ('ANALYSIS OF POSITIVE HEART ATTACK IN
WOMEN OUT OF TOTAL WOMEN')

plt.s how()

# age distribution of people affected

import seaborn as sns

# seaborn histogram

sns.distplot(train['age'], hist=True, kde=False, bins=int(180/5), color = 'blue',
hist_kws={'edgecolor':'black'})

# Add labels

plt.title('distribution of people count vs age') plt.xlabel('age(in years)')
plt.ylabel('number of pers on')

plt.show() sns.distplot(train['age'])

# distribution of cholesterol level in affected people

sns.distplot(train['chol'], hist=True, kde=False, bins=int(200/5), color = 'blue',
hist_kws={'edgecolor':'black'}) plt.title('distribution serum cholesterol in mg/dl')
plt.xlabel('serum cholesterol in m g/dl') plt.ylabel('count')

```

```

plt.show() sns.distplot(train['chol'])

# max heart rate distribution

sns.distplot(train['thalach'], hist=True, kde=False, bins=int(200/5), color = 'blue',
hist_kws={'ed_gcolor':'black'}) plt.title('distribution of maximum heart rate
achieved') plt.xlabel('maximum heart rate achieved') plt.ylabel('count')

plt.show() sns.distplot(train['thalach'])

# resting heart rate distribution

sns.distplot(train['trestbps'], hist=True, kde=False, bins=int(100/5), color = 'blue',
hist_kws={'ed_gcolor':'red'}) plt.title('distribution of resting blood pressure (in mm
Hg on admission to the hospital)') plt.xlabel('resting blood pressure (in mm Hg on
admission to the hospital)') plt.ylabel('count')

plt.show() sns.distplot(train['trestbps'])

# IMPORTING REQUIRED LIBRARIES

from sklearn.linear_model import LogisticRegression from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier from sklearn.ensemble import
RandomForestClassifier from sklearn.neighbors import KNeighborsClassifier from
sklearn.naive_bayes import GaussianNB

# SPLITTING DATA

from sklearn.model_selection import train_test_split train1 = train.copy() feature_df
=
train1[["age","sex","cp","trestbps","thalach","chol","restecg","exang","oldpeak","slo
pe","ca","thal"]] x = n p.asarray(feature_df) y =
np.asarray(train["target"].astype('int')) x_train,x_test,y_train,y_test = tr
ain_test_split(x,y,test_size = 0.2,random_state = 5)

def train_model(x_train, y_train, x_test, y_test, classifier, **kwargs):

# instantiate model

model = classifier(**kwargs) # train model model.fit(x_train,y_train) y_pred =
model.predict(x_ test) from sklearn.metrics import confusion_matrix
print("Confusion matrix") print(confusion_m atrix(y_test,y_pred))

# check accuracy and print out the results

fit_accuracy = model.score(x_train, y_train) test_accuracy = model.score(x_test,
y_test) print(f" Train accuracy: {fit_accuracy:0.2%}")

print(f"Test accuracy: {test_accuracy:0.2%}") return model

```



```
# KNN
```

```
model = train_model(x_train, y_train, x_test, y_test, KNeighborsClassifier)
```

```
# Seek optimal 'n_neighbours' parameter
```

```
for i in range(1,10): print("n_neighbors = "+str(i)) train_model(x_train, y_train, x_test, y_test, KNeighborsClassifier, n_neighbors=i)
```

```
# Decision Tree
```

```
model = train_model(x_train, y_train, x_test, y_test, DecisionTreeClassifier, random_state=2606)
```

```
optimal 'max_depth' parameter
```

```
for i in range(1,8): print("max_depth = "+str(i)) train_model(x_train, y_train, x_test, y_test, DecisionTreeClassifier, max_depth=i, random_state=2606)
```

```
# LOGISTIC REGRESSION
```

```
model = train_model(x_train, y_train, x_test, y_test, LogisticRegression)
```

```
#Gaussian Naive Bayes
```

```
model = train_model(x_train, y_train, x_test, y_test, GaussianNB)
```

```
# Support Vector Machines
```

```
model = train_model(x_train, y_train, x_test, y_test, SVC)
```

```
# Support Vector Machine Linear
```

```
model = train_model(x_train, y_train, x_test, y_test, SVC, C=0.05, kernel='linear')
```

```
# Random Forests
```

```
model = train_model(x_train, y_train, x_test, y_test, RandomForestClassifier, random_state=2606)
```

```
# USING ANN Multilayer Perceptron Classifier
```

```
from sklearn.metrics import make_scorer, accuracy_score from  
sklearn.model_selection import GridSearchCV from sklearn.neural_network import  
MLPClassifier ann_clf = MLPClassifier() parameters = {'solver': ['lbfgs'], 'alpha': [1e-  
4], 'hidden_layer_sizes': (9,14,14,2), 'random_state': [1]} a cc_scorer =
```

```

make_scorer(accuracy_score) grid_obj = GridSearchCV(ann_clf, parameters,
scoring=accuracy_score) grid_obj = grid_obj.fit(x_train, y_train) ann_clf=
grid_obj.best_estimator_

```

```

ann_clf.fit(x_train,y_train)

```

```

y_pred_ann = ann_clf.predict(x_test) from sklearn.metrics import confusion_matrix
cm_ann = confusion_matrix(y_test, y_pred_ann) cm_ann

```

```

ann_result = accuracy_score(y_test, y_pred_ann) ann_result

```

FOR TEST CASES :

```

import pandas

```

```

import os

```

```

import numpy as np

```

```

from sklearn.linear_model import LinearRegression

```

```

from sklearn.cross_validation import KFold

```

```

from sklearn import cross_validation

```

```

from sklearn.linear_model import LogisticRegression

```

```

heart = pandas.read_csv("pc.csv")

```

```

heart.loc[heart["heartpred"]==2,"heartpred"]=1

```

```

heart.loc[heart["heartpred"]==3,"heartpred"]=1

```

```

heart.loc[heart["heartpred"]==4,"heartpred"]=1

```

```

heart["slope"] = heart["slope"].fillna(heart["slope"].median())

```

```

heart["thal"] = heart["thal"].fillna(heart["thal"].median())

```

```

heart["ca"] = heart["ca"].fillna(heart["ca"].median())

```

```

print(heart.describe())

```

```

predictors=["age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","old
peak","slope","ca","thal"]

```

```

alg=LogisticRegression(random_state=1)

```

```

predictions = []

```

```

train_predictors = (heart[predictors].iloc[:,:])
train_target = heart["heartpred"].iloc[:]
alg.fit(train_predictors, train_target)
l=len(heart.index)
while True:
x1=raw_input("Input age (67.0):")
x2=raw_input("Input sex (1.0):")
x3=raw_input("Input cp (4.0):")
x4=raw_input("Input trestbps (160.0):")
x5=raw_input("Input chol (286.0):")
x6=raw_input("Input fbs (0.0):")
x7=raw_input("Input restecg (2.0):")
x8=raw_input("Input thalach (108.0):")
x9=raw_input("Input exang (1.0):")
x10=raw_input("Input oldpeak (1.5):")
x11=raw_input("Input slope (2.0):")
x12=raw_input("Input ca (3.0):")
x13=raw_input("Input thal (3.0):")

f = open('pc.csv','a+')

f.write(x1+', '+x2+', '+x3+', '+x4+', '+x5+', '+x6+', '+x7+', '+x8+', '+x9+', '+x10+', '+x11+', '+x1
2+', '+x13+', ')

f.close()

heart = pandas.read_csv("pc.csv")

#print(l+1)

print(heart[predictors].iloc[l,:])

test_predictions = alg.predict(heart[predictors].iloc[l,:])

```

```
test_predictions[test_predictions > .5] = 1
test_predictions[test_predictions <=.5] = 0
print(test_predictions.item(0))
x14=test_predictions.item(0)
print(type(x14))
f = open('pc.csv','a+')
f.write("%s"%(x14))
f.write(os.linesep)
f.close()
l+=1
```

B. Screenshot

```

In [1]: # import the libraries

In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re

In [3]: # loading data from heart_disease_data.csv

In [4]: train = pd.read_csv("heart_disease_data.csv")

In [5]: # deletes rows with empty cells

In [6]: train.dropna()
Out[6]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

```
[11]: # Percentage of men and women in our dataset
```

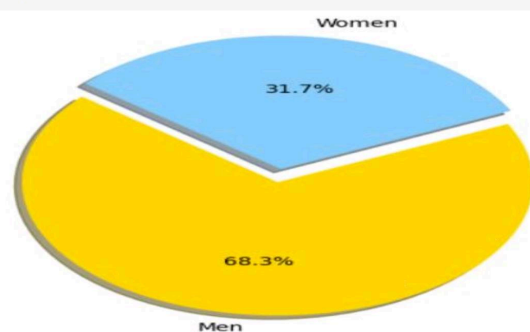
```

[12]: import matplotlib.pyplot as plt

labels = ['Men', 'Women']
sizes = [0.6831, 0.3168]
colors = ['gold', 'lightskyblue']
explode = (0.1, 0)

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
plt.axis('equal')
plt.show()

```



```

In [13]: # explode 1st slice

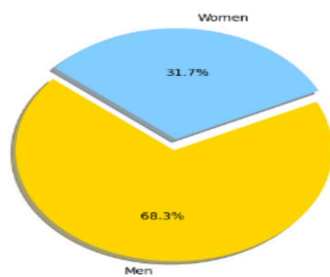
In [14]: # Plot

In [16]: import matplotlib.pyplot as plt
        sizes = [0.6831, 0.3168]
        explode = (0.1, 0)
        labels = ['Men', 'Women']
        colors = ['gold', 'lightskyblue']

        plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
        plt.title('% of men and women in our dataset\n\n\n')
        plt.axis('equal')
        plt.show()

```

% of men and women in our dataset



```

In [17]: # Plot of men at risk vs women at risk

```

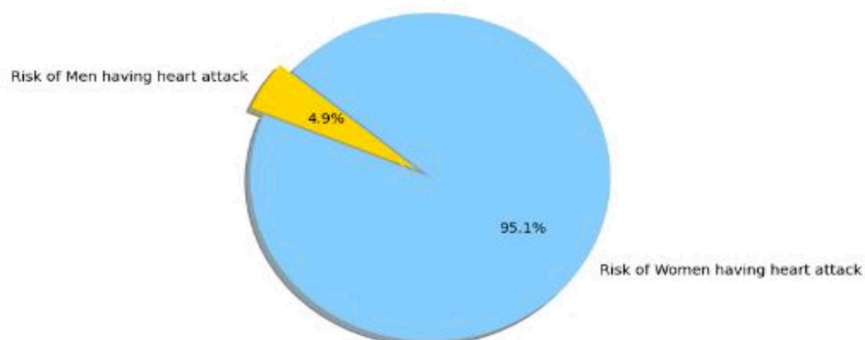
```

In [19]: import matplotlib.pyplot as plt
        sizes = [0.449, 8.75]
        explode = (0.08, 0) # explode the first slice
        labels = ['Risk of Men having heart attack', 'Risk of Women having heart attack']
        colors = ['gold', 'lightskyblue']

        plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
        plt.title('Risk of Heart Attack by Gender\n\n\n')
        plt.axis('equal')
        plt.show()

```

Risk of Heart Attack by Gender



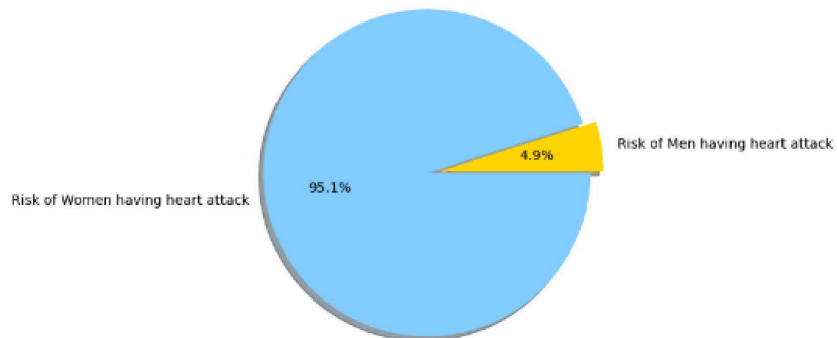
```
[20]: # Plot
```

```
[22]: import matplotlib.pyplot as plt
```

```
sizes = [0.449, 8.75]  
explode = (0.08, 0) # explode the first slice
```

```
labels = ['Risk of Men having heart attack', 'Risk of Women having heart attack']  
colors = ['gold', 'lightskyblue']
```

```
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=0)  
plt.axis('equal')  
plt.show()
```

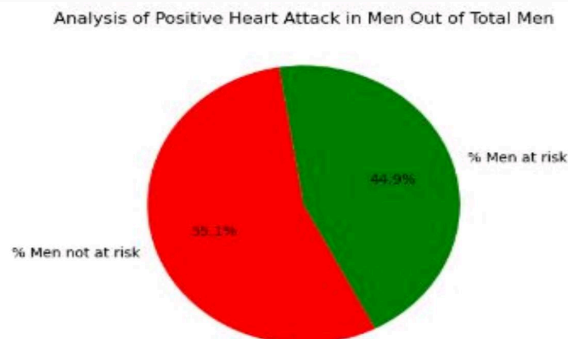


```
[23]: # percentage of men at risk
```

```
[25]: import matplotlib.pyplot as plt
```

```
slices_hours = [207-93, 93]  
activities = ['% Men not at risk', '% Men at risk']  
colors = ['r', 'g']
```

```
plt.pie(slices_hours, labels=activities, colors=colors, startangle=99, autopct='%1.1f%%')  
plt.title('Analysis of Positive Heart Attack in Men Out of Total Men')  
plt.show()
```

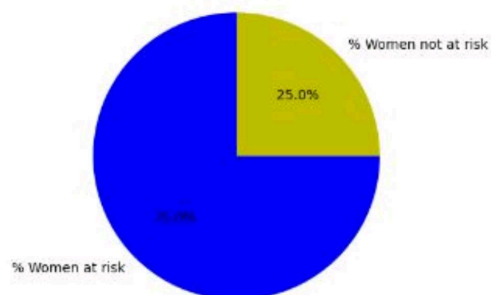


```
In [26]: # percentage of women at risk
```

```
In [28]: import matplotlib.pyplot as plt
```

```
slices_hours = [96-72, 72]  
activities = ['% Women not at risk', '% Women at risk']  
colors = ['y', 'b']  
  
plt.pie(slices_hours, labels=activities, colors=colors, startangle=0, autopct='%1.1f%%')  
plt.title('Analysis of Positive Heart Attack in Women Out of Total Women')  
plt.show()
```

Analysis of Positive Heart Attack in Women Out of Total Women



```
In [29]: # age distribution of people affected
```

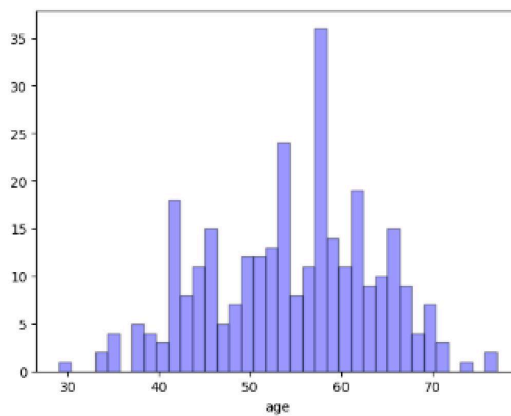
```
In [30]: import seaborn as sns
```

```
In [31]: # seaborn histogram
```

```
In [33]: import seaborn as sns
```

```
sns.distplot(train['age'], hist=True, kde=False, bins=int(180/5), color='blue', hist_kws={'edgecolor': 'black'})
```

```
Out[33]: <Axes: xlabel='age'>
```

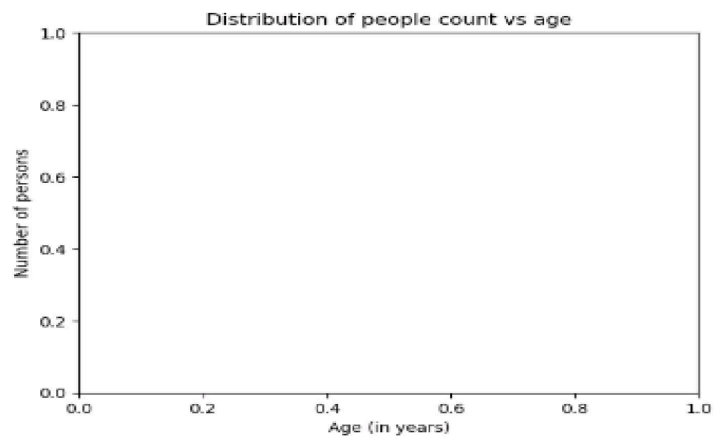



```
In [34]: # Add Labels
```

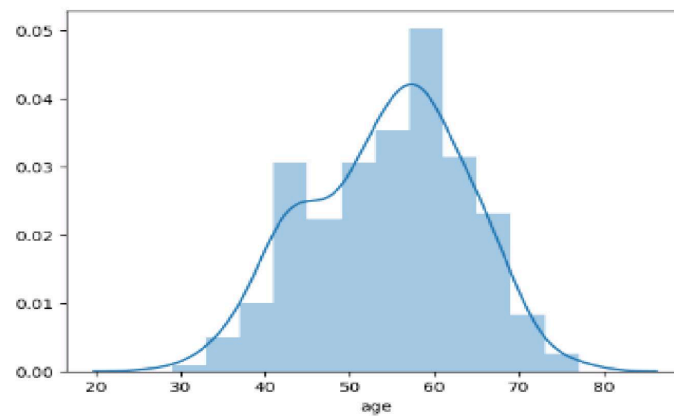
```
In [36]: import matplotlib.pyplot as plt
import seaborn as sns

plt.title('Distribution of people count vs age')
plt.xlabel('Age (in years)')
plt.ylabel('Number of persons')
plt.show()

sns.distplot(train['age'])
```



```
Out[36]: <Axes: xlabel='age'>
```

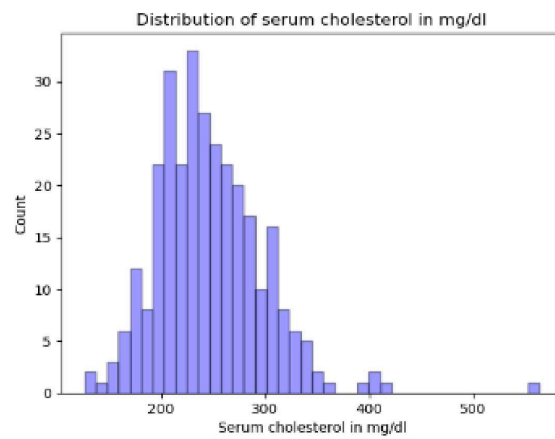


```
In [37]: # distribution of cholesterol Level in affected people
```

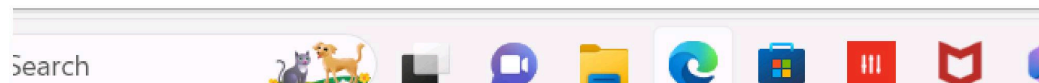
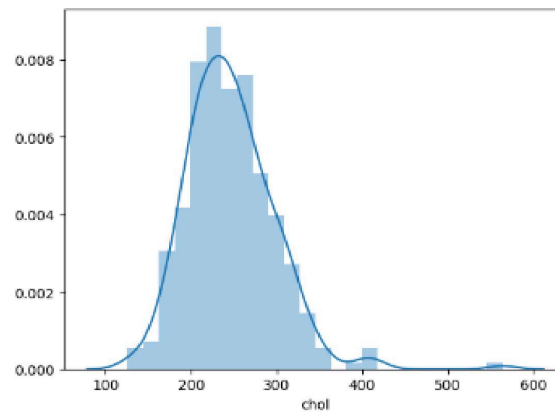
```
In [39]: import seaborn as sns
import matplotlib.pyplot as plt

sns.distplot(train['chol'], hist=True, kde=False, bins=int(280/5), color='blue', hist_kws={'edgecolor': 'black'})
plt.title('Distribution of serum cholesterol in mg/dl')
plt.xlabel('Serum cholesterol in mg/dl')
plt.ylabel('Count')
plt.show()

sns.distplot(train['chol'])
```



```
Out[39]: <Axes: xlabel='chol'>
```

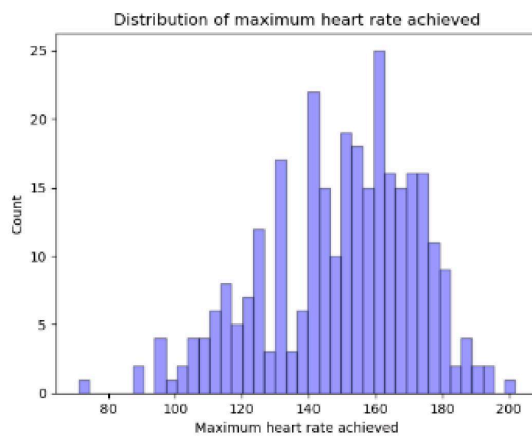


```
In [40]: # max heart rate distribution
```

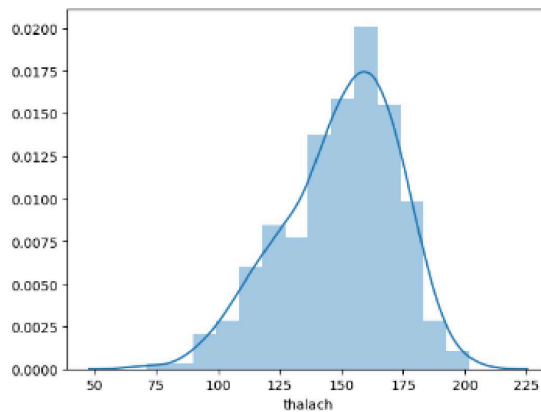
```
In [42]: import seaborn as sns
import matplotlib.pyplot as plt

sns.distplot(train['thalach'], hist=True, kde=False, bins=int(200/5), color='blue', hist_kws={'edgecolor': 'black'})
plt.title('Distribution of maximum heart rate achieved')
plt.xlabel('Maximum heart rate achieved')
plt.ylabel('Count')
plt.show()

sns.distplot(train['thalach'])
```



```
Out[42]: <Axes: xlabel='thalach'>
```



search



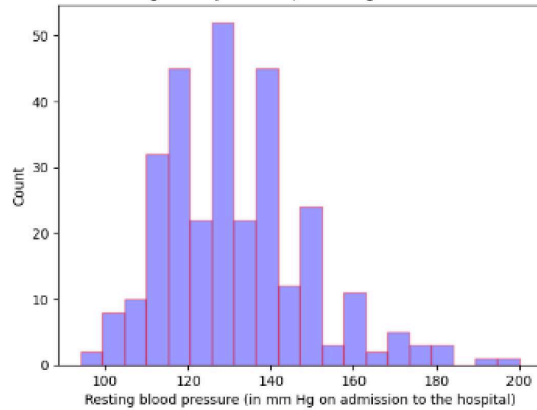
```
In [43]: # resting heart rate distribution
```

```
In [45]: import seaborn as sns
import matplotlib.pyplot as plt

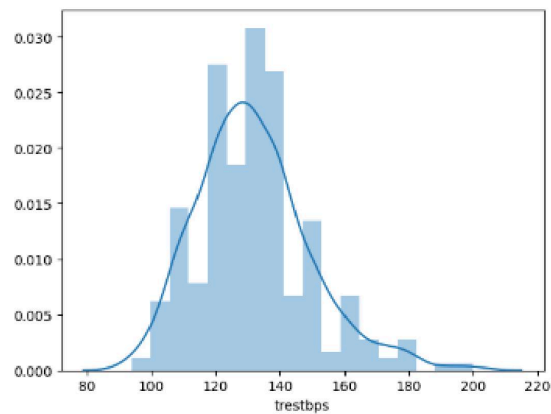
sns.distplot(train['trestbps'], hist=True, kde=False, bins=int(180/5), color='blue', hist_kws={'edgecolor': 'red'})
plt.title('Distribution of resting blood pressure (in mm Hg on admission to the hospital)')
plt.xlabel('Resting blood pressure (in mm Hg on admission to the hospital)')
plt.ylabel('Count')
plt.show()

sns.distplot(train['trestbps'])
```

Distribution of resting blood pressure (in mm Hg on admission to the hospital)



```
Out[45]: <Axes: xlabel='trestbps'>
```



```
In [46]: # IMPORTING REQUIRED LIBRARIES
```

```
In [48]: from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

```
In [49]: # SPLITTING DATA
```

```
In [51]: from sklearn.model_selection import train_test_split
train1 = train.copy()
feature_df = train1[["age", "sex", "cp", "trestbps", "thalach", "chol", "restecg", "exang", "oldpeak", "slope", "ca", "thal"]]
x = np.asarray(feature_df)
y = np.asarray(train1["target"].astype('int'))
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=5)
```

```
In [53]: def train_model(x_train, y_train, x_test, y_test, classifier, **kwargs):
# Function body goes here
pass # Placeholder for the function body if you haven't implemented it yet
```

```
In [54]: # instantiate model
```

```
In [68]: from sklearn.linear_model import LogisticRegression
# Import other classifiers as needed

# Define the classifier you want to use
classifier = LogisticRegression() # Example: Logistic Regression

# Now you can proceed with the rest of your code
model = classifier.fit(x_train, y_train)
y_pred = model.predict(x_test)

# Import confusion_matrix separately
from sklearn.metrics import confusion_matrix

print("Confusion matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion matrix:
[[27  3]
 [ 2 29]]
```

```
C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
```

search



```
In [57]: # check accuracy and print out the results
```

```
In [61]: fit_accuracy = model.score(x_train, y_train)
test_accuracy = model.score(x_test, y_test)
print(f"Train accuracy: {fit_accuracy:0.2%}")
print(f"Test accuracy: {test_accuracy:0.2%}")
return model
```

Train accuracy: 83.88%
Test accuracy: 91.88%

Cell In[61], line 5
return model

```
In [62]: # KNN
```

```
In [63]: model = train_model(x_train, y_train, x_test, y_test, KNeighborsClassifier)
```

```
In [64]: # Seek optimal 'n_neighbours' parameter
```

```
In [66]: for i in range(1, 10):
          print(f"n_neighbors = " + str(i))
          train_model(x_train, y_train, x_test, y_test, KNeighborsClassifier, n_neighbors=i)
```

```
n_neighbors = 1
n_neighbors = 2
n_neighbors = 3
n_neighbors = 4
n_neighbors = 5
n_neighbors = 6
n_neighbors = 7
n_neighbors = 8
n_neighbors = 9
```

```
In [67]: # Decision Tree
```

```
In [68]: model = train_model(x_train, y_train, x_test, y_test, DecisionTreeClassifier, random_state=2606
)
```

```
In [69]: # Check optimal 'max_depth' parameter
```

```
In [71]: for i in range(1, 8):
          print(f"max_depth = " + str(i))
          train_model(x_train, y_train, x_test, y_test, DecisionTreeClassifier, max_depth=i, random_state=2606)
```

```
max_depth = 1
max_depth = 2
max_depth = 3
max_depth = 4
max_depth = 5
max_depth = 6
max_depth = 7
```

```
In [72]: # LOGISTIC REGRESSION
```

```
In [73]: model = train_model(x_train, y_train, x_test, y_test, LogisticRegression)
```

```
In [74]: #Gaussian Naive Bayes
```

```
In [75]: model = train_model(x_train, y_train, x_test, y_test, GaussianNB)
```

```

6]: # Support Vector Machines

7]: model = train_model(x_train, y_train, x_test, y_test, SVC)

8]: # Support Vector Machine Linear

9]: model = train_model(x_train, y_train, x_test, y_test, SVC, C=0.05, kernel='linear')

10]: # Random Forests

2]: model = train_model(x_train, y_train, x_test, y_test, RandomForestClassifier, random_state=2006)

5]: # USING ANN Multilayer Perceptron Classifier

6]: from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier

ann_clf = MLPClassifier()
parameters = {'solver': ['lbfgs'], 'alpha': [1e-4], 'hidden_layer_sizes': (9, 14, 14, 2), 'random_state': [1]}
acc_scorer = make_scorer(accuracy_score)
grid_obj = GridSearchCV(ann_clf, parameters, scoring=acc_scorer)
grid_obj.fit(x_train, y_train)
ann_clf = grid_obj.best_estimator_

C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

```

```

In [87]: ann_clf.fit(x_train,y_train)

C:\Users\abhay\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)

Out[87]: MLPClassifier(hidden_layer_sizes=(14, 14, 2), random_state=1, solver='lbfgs')
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```

```

In [89]: y_pred_ann = ann_clf.predict(x_test)
from sklearn.metrics import confusion_matrix
cm_ann = confusion_matrix(y_test, y_pred_ann)
cm_ann

Out[89]: array([[26,  4],
               [ 3, 28]], dtype=int64)

In [91]: ann_result = accuracy_score(y_test, y_pred_ann)
ann_result

Out[91]: 0.8852459016393442

```

In [93]: #FOR TEST CASES :

```
In [97]: import pandas
import os
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.cross_validation import KFold
from sklearn import cross_validation
from sklearn.linear_model import LogisticRegression
heart = pandas.read_csv("heart_disease_data.csv")
heart.loc[heart["heartpred"]==2,"heartpred"]=1
heart.loc[heart["heartpred"]==3,"heartpred"]=1
heart.loc[heart["heartpred"]==4,"heartpred"]=1
heart["slope"] = heart["slope"].fillna(heart["slope"].median())
heart["thal"] = heart["thal"].fillna(heart["thal"].median())
heart["ca"] = heart["ca"].fillna(heart["ca"].median())
print(heart.describe())
predictors=["age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak","slope","ca","thal"]
alg=LogisticRegression(random_state=1)
predictions = []
train_predictors = (heart[predictors].iloc[:,:])
train_target = heart["heartpred"].iloc[:]
alg.fit(train_predictors, train_target)
l=len(heart.index)
while True:
x1=raw_input("Input age (67.0):")
x2=raw_input("Input sex (1.0):")
x3=raw_input("Input cp (4.0):")
x4=raw_input("Input trestbps (160.0):")
x5=raw_input("Input chol (286.0):")
x6=raw_input("Input fbs (0.0):")
x7=raw_input("Input restecg (2.0):")
x8=raw_input("Input thalach (108.0):")
x9=raw_input("Input exang (1.0):")
x10=raw_input("Input oldpeak (1.5):")
x11=raw_input("Input slope (2.0):")
x12=raw_input("Input ca (3.0):")
x13=raw_input("Input thal (3.0):")
f = open('pc.csv','a+')
f.write(x1+', '+x2+', '+x3+', '+x4+', '+x5+', '+x6+', '+x7+', '+x8+', '+x9+', '+x10+', '+x11+', '+x12+', '+x13+', '\n')
f.close()
heart = pandas.read_csv("heart_disease_data.csv")
#print(l+1)
print(heart[predictors].iloc[l,:])
test_predictions = alg.predict(heart[predictors].iloc[l,:])
test_predictions[test_predictions > .5] = 1
test_predictions[test_predictions <=.5] = 0
print(test_predictions.item(0))
x14=test_predictions.item(0)
print(type(x14))
f = open('heart_disease_data.csv','a+')
f.write("%s"%(x14))
f.write(os.linesep)
f.close()
l+=1
```

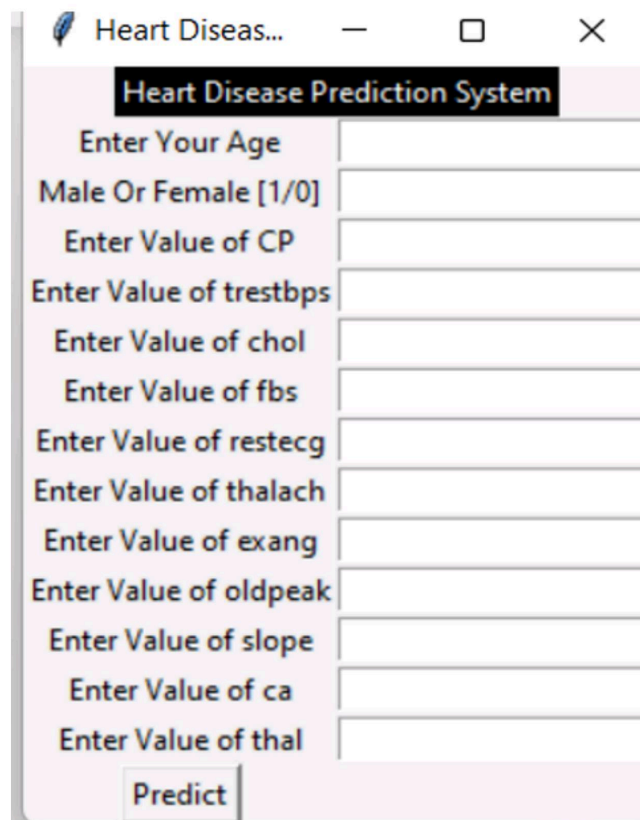
Cell In[97], line 16

```
predictors=["age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak","slope",
```


B. Screenshot of the GUI with input and output:

Here we are going to create and predict the patients details through GUI for our project.
So, anyone can perform predictions using this GUI.

INPUT:



The screenshot shows a window titled "Heart Diseases..." with a subtitle "Heart Disease Prediction System". The window contains a list of input fields for patient data, each with a label and a corresponding text box. The labels are: "Enter Your Age", "Male Or Female [1/0]", "Enter Value of CP", "Enter Value of trestbps", "Enter Value of chol", "Enter Value of fbs", "Enter Value of restecg", "Enter Value of thalach", "Enter Value of exang", "Enter Value of oldpeak", "Enter Value of slope", "Enter Value of ca", and "Enter Value of thal". At the bottom of the window is a button labeled "Predict".

Fig.7.B1: Input

OUTPUT

Heart Disease Pre... — □ ×

Heart Disease Prediction System

Enter Your Age	63
Male Or Female [1/0]	1
Enter Value of CP	3
Enter Value of trestbps	145
Enter Value of chol	233
Enter Value of fbs	1
Enter Value of restecg	0
Enter Value of thalach	150
Enter Value of exang	0
Enter Value of oldpeak	2.3
Enter Value of slope	0
Enter Value of ca	0
Enter Value of thal	1

Possibility of Heart Disease

Heart Disease Pre... — □ ×

Heart Disease Prediction System

Enter Your Age	57
Male Or Female [1/0]	0
Enter Value of CP	0
Enter Value of trestbps	140
Enter Value of chol	241
Enter Value of fbs	0
Enter Value of restecg	1
Enter Value of thalach	123
Enter Value of exang	1
Enter Value of oldpeak	0.2
Enter Value of slope	1
Enter Value of ca	0
Enter Value of thal	3

Post: No Heart Disease ease

Fig.7.B2: Output

