A Project Report on

MYOCARDIAL INFARCTION PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING

By

B. DILEEP VENKATA PRASAD	**17A91A04C4**
P. APUROOP SRI DURGESH	**17A91A04D9**
S. SRIDEVI NAGA KALANJALI	**17A91A04G8**
V. SRI SAI KAMAL	**18A95A0438**
B. MANIKANTA	**18A95A0427**

Under the Esteemed guidance of

Mr. P.BALA SRINIVAS, M.Tech.,(Ph.D.)

Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ADITYA ENGINEERING COLLEGE (A)

(Approved by AICTE, Permanently Affiliated to JNTUK & Accredited by NBA, NAAC with 'A' Grade. Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956)

Aditya Nagar, ADB Road, Surampalem – 533 437

(2017-2021)

ADITYA ENGINEERING COLLEGE (A)

(Approved by AICTE, Permanently Affiliated to JNTUK & Accredited by NBA, NAAC with 'A' Grade. Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956)

Aditya Nagar, ADB Road, Surampalem – 533 437

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Certificate

This is to certify that the project report entitled **"MYOCARDIAL INFARCTION PREDICTION USING MACHINE LEARNING"** being submitted by

B. DILEEP VENKATA PRASAD	**17A91A04C4**
P. APUROOP SRI DURGESH	**17A91A04D9**
S. SRIDEVI NAGA KALANJALI	**17A91A04G8**
V. SRI SAI KAMAL	**18A95A0438**
B. MANIKANTA	**18A95A0427**

for the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in Department Of Electronics & Communication Engineering of Aditya Engineering College to Jawaharlal Nehru Technological University, Kakinada is a record of bonafide work carried out by them under the guidance and supervision during Academic Year of 2020-21.

Project Guide **Head of the Department**

Mr. P.Bala Srinivas **Mr. V.Satyanarayana**

External Examiner

ACKNOWLEDGEMENT

We take this opportunity as a privilege to thank all individuals without whose support and guidance we could not have completed our project in this stipulated period oftime.

We express our deep sense of gratitude to our guide **Mr. P. Bala Srinivas** for his valued suggestions and inputs during the course of the project work, readiness for consultation atall times, his educative comments and inputs, his concern and assistance even with practical things have been extremely helpful.

We highly indebted to our Head of the Department **Mr. V. Satyanarayana** for his motivational guidance and the vision in providing the necessary resources and timely inputs.

We are also thankful to **Dr. M. Sreenivasa Reddy,** Principal, Aditya EngineeringCollege for providing appropriate environment required for this project and thankful to Faculty of Electronics and Communication Engineering Department for the encouragement and cooperation for this successful completion of the project.

B. DILEEP VENKATA PRASAD	**17A91A04C4**
P. APUROOP SRI DURGESH	**17A91A04D9**
S. SRIDEVI NAGA KALANJALI	**17A91A04G8**
V. SRI SAI KAMAL	**18A95A0438**
B. MANIKANTA	**18A95A0427**

CONTENTS

		Page No.
List of Figures		i
List of Tables		iii
Abstract		iv
1.	**INTRODUCTION**	**1-3**
	1.1 Introduction	1
	1.2 Software Requirements	1
	1.2.1 Python	1
	1.2.2 Machine Learning	1
	1.2.3 React Native	2
	1.2.4 Flask	2
	1.2.5 Heroku	2
	1.3 Motivation	3
	1.4 Problem statement	3
	1.5 Objective and Scope	3
	1.6 Organization of Project	3
2.	**LITERATURE SURVEY**	**4-6**
	2.1 Research Papers	4
3.	**HISTORICAL BACKGROUND**	**7**
4.	**IMPLEMENTATION**	**8-32**
5.	**BACKEND DESIGN**	**33-37**
	5.1 Backend Development	33
	5.2 Restful API Design	34
	5.3 Database	35
	5.4 Server	35
	5.5 Flask Framework	36
	5.6 Heroku	36
	5.7 Android App Development	36
	5.8 React Native	37
6.	**RESULTS**	**38-44**
7.	**CONCLUSIONS & FUTURE SCOPE**	**45-47**
	References	**48**
	Appendix	**49**

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
4.0	Workflow of the project	09
4.2	Output for checking what type of data in dataset	14
4.3	Output of the Correlation for all the parameters	16
4.4	Heat Map	17
4.5	Output for counting the patients with and without heart problem	19
4.6	Predicted output of the Logistic Regression algorithm	22
4.7	Predicted output of the Naïve Bayes algorithm	24
4.8	Predicted output of the Support Vector Machine algorithm	25
4.9	Working of Random Forest algorithm	26
4.10	Predicted output of the Random Forest algorithm	28
4.11	Predicted output of the K-Nearest Neighbor algorithm	29
5.1	Backend design architecture	33
6.0	Confusion Matrix for Logistic Regression algorithm	38
6.1	Confusion Matrix for Naïve Bayes algorithm	38
6.2	Confusion Matrix for Support Vector Machine algorithm	39
6.3	Confusion Matrix for Random Forest algorithm	39
6.4	Confusion Matrix for K-Nearest Neighbor algorithm	40
6.6	Distribution plot for Logistic Regression algorithm	41
6.7	Distribution plot for Naïve Bayes algorithm	41
6.8	Distribution plot for Support Vector Machine algorithm	42

6.9	Distribution plot for Random Forest algorithm	42
6.10	Distribution plot for K-Nearest Neighbor algorithm	42
6.11	Bar plot for Accuracy	43
6.12	Bar plot for Inaccuracy	43
6.13	Bar plot for Sensitivity	44
6.14	Bar plot for Specificity	44
6.15	Bar plot for Precision	44

LIST OF TABLES

Table No.	Name of the table	Page No.
4.1	Parameters of heart disease dataset	11
4.12	Confusion matrix	30
6.5	Evaluation measures table for ML algorithms	40

ABSTRACT

Myocardial Infarction (MI) is the scientific name for Heart attack. Heart plays a crucial role in living organisms. Diagnosis and prediction of heart related diseases is very essential in order to avoid any death or serious issues for a living organism. Myocardial Infarction occurs when there is a diminished blood supply to the heart, which leads to myocardial cell damage and ischemia. The incidence of Myocardial Infarction increases with age, and more common in females. This disease requires more exactness, precision and accuracy because a little mistake can cause fatigue problem or death of the person. Day by day, the death cases are increasing exponentially related to heart. 90% of the patients with MI do not have any symptoms (silent MI) which are the reason for increasing mortality rate and the patient will have post MI complications even after recovery. To deal with this problem, there is essential need of prediction system for awareness about diseases.

Machine Learning (ML) is a part of Artificial Intelligence (AI). It is the study of computer algorithms that improves automatically through experience and by the use of data. In this project, we have used the dataset from the Kaggle website, which consists of several parameters with a size of 303 rows and 14 columns related to heart disease. We used machine learning algorithms such as Logistic Regression, Random Forest, Support Vector Machine (SVM), Naïve Bayes and K-Nearest Neighbour for the development of the model. For implementation of Python Programming, we use Jupyter Notebook platform. After analyzing all these algorithms, we got Random Forest as the best accuracy algorithm and can be used for predicting the future values. This model can be helpful to the medical practitioners at their clinic as a decision support system.

CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

Heart plays a crucial role in living organisms, diagnosis and prediction of heart diseases is very essential in order to avoid any death or serious issues for a living organism. The term heart disease is sometimes used interchangeably with the term cardiovascular disease. A cardiovascular failure happens when the progression of blood to the heart is impeded. The blockage is regularly a development of fat, cholesterol and different substances, which structure plaque in the supply routes that feed the heart (coronary veins). The plaque at last splits away and structures coagulation. The interfered with blood stream can harm or annihilate a piece of the heart muscle. Other heart conditions, such as those that affect your heart's muscle, valves etc., also are considered forms of heart disease. 17.9 million People die each year from cardiovascular diseases, an estimated 31% of all deaths worldwide and 85% of all cardiovascular deaths are due to heart attacks and strokes. Nowadays healthcare sector produces large amount of information about patients, disease diagnosis etc. however this data is not used efficiently by the researchers and practitioners. Today a major challenge faced by Healthcare industry is quality of service (QoS). QoS implies diagnosing disease correctly & provides effective treatments to patients. Poor diagnosis can lead to disastrous consequences which are unacceptable.

1.2 SOFTWARE REQUIREMENTS

1.2.1 PYTHON

Python is the language used. Python files are saved with ".py" extension. Same is also applied on Jupyter notebook whose extension is .ipynb. You need to install python as well as Jupyter for using this project. If you want to envision the result using only python, you should install Pycharm where Jupyter notebook is used to visualize the result on the web browser.

1.2.2 MACHINE LEARNING

Machine Learning is used for numerous purpose such as color based division, forecasting diseases, image processing applications such as object recognition, image classification and transfer learning. When deep learning came the computation power has upgraded to such level that now it is possible for the machines to effort like humans. Companies like Interest, Google, Facebook and Amazon is consuming this technology to so large extent that their incomes have increased dramatically. In this project, we use with different machine learning algorithms such as Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest, and K-Nearest Neighbor to perform the classification for heart disease.

1.2.3 REACT NATIVE

It was introduced by facebook in 2015, React Native is an open-source framework for building cross-platform native apps. Using React and Java script as programming languages, we can build mobile apps, indistinguishable from native apps built using Objective-C, Swift, orJava.

React Native is an extension of React, which is a Javascript library for building UI blocks for web applications. Declarative programming style, virtual DOM, reusable components to build UI are some of the premium advantages that gives this JS framework an edge over hundreds of options out there for building mobile apps.

1.2.4 FLASK

Flask is a web framework that provides libraries to build lightweight web applications in python. Flask is an API of Python that allows us to build up web-applications. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application.

There are many other python web frameworks that we can use such as Django but they increase complexity since we don't need of any database connectivity and complex routing.

So flask is the great and simple framework that we used to process our data in server. All the machine learning algorithms are moved to the flask code which can be useful for processing of data.

1.**2.5 HEROKU**

Heroku is a container-based cloud Platform as a Service (PaaS). It is used to deploy, manage, and scale modern apps. It is elegant, flexible, and easy to use. There are many other cloud based services such as AWS, Google Cloud, Digital ocean but in this project, we used Heroku to deploy application written in Python, Node.js.

1.3 MOTIVATION

Heart disease affects millions of people, and it remains the main cause of death in the world. Data mining is a software technology that helps computers to build and classify various attributes. This project uses classification techniques to predict heart disease.

1.4 PROBLEM STATEMENT

A major challenge faced by health care organizations, such as hospitals and medical centres, is the provision of quality services at affordable costs. The quality service implies diagnosing patients properly and administering effective treatments.

1.5 OBJECTIVE AND SCOPE

We would like to make a Machine Learning Model where we can train our AI to learn & improve from experience. Thus, we would want to predict the percentage of occurrence of heart attack for a person. The proposed work makes an attempt to detect these heart diseases at early stage to avoid disastrous consequences.

The available heart disease database consists of both numerical and categorical data. Before further processing, cleaning and filtering are applied on these records in order to filter the irrelevant data from the database. The proposed system can determine an exact hidden knowledge, i.e., patterns and relationships associated with heart disease from a historical heart disease database. It can also answer the complex queries for diagnosing heart disease therefore; it can be helpful to health care practitioners to make intelligent clinical decisions. Results showed that the proposed system has its unique potency in realizing the objectives of the defined mining goals.

1.6 ORGANIZATION OF PROJECT

In our project, chapter-1 provides an introduction to the project as a whole, chapter-2 deals with the literature survey, chapter-3 deals with the historical background, chapter-4 deals with the implementation, chapter-5 deals with the backend design, chapter-6 deals with results and finally, Chapter-7 deals with conclusion and future scope.

CHAPTER-2

LITERATURE SURVEY

2.1 RESEARCH PAPERS

In the part of literature survey, number of works have done related to heart disease prediction systems using different machine learning algorithms is as follows

Research Paper-1 [2]

Anjan Nikhil Repaka, Sai Deepak Ravikanti, Ramya G Franklin et al. proposed “Design And Implementing Heart Disease Prediction Using Naives Bayesian”. In this work, it mainly focuses on heart disease diagnosis by considering previous data and information. To achieve this SHDP (Smart Heart Disease Prediction) is built via Naive Bayesian in order to predict risk factors concerning heart disease. For predicting the chances of heart disease in a patient, the following attributes are being fetched from the medical profiles, these include: age, BP, cholesterol, sex, blood sugar etc... The collected attributes acts as input for the Naives Bayesian classification for predicting heart disease. The dataset utilized is split into two sections, 80% dataset is utilized for training and rest 20% is utilized for testing. The proposed approach includes following stages: dataset collection, user registration and login (Application based), classification via Navies Bayesian, prediction and secure data transfer by employing AES (Advanced Encryption Standard). Thereafter result is produced. The output reveals that the established diagnostic system effectively assists in predicting risk factors concerning heart diseases.

Research Paper-2 [3]

Mamatha Alex P and Shaicy P Shaji et al. proposed “Prediction and Diagnosis of Heart Disease Patients using Data Mining Technique” proposed by. In this paper, a total 20 attributes of nearly 2200 and above patients were collected. This collected data were then sorted and arranged systematically in Excel format. Using this data, it can be subjected to different data mining algorithms. From the medical profiles twenty attributes are extracted such as age, sex, blood pressure and blood sugar etc. to predict the likelihood of patient getting heart diseases. These attributes are fed in to SVM, Random forest, KNN, and ANN classification Algorithms in which ANN gave the best result with the highest accuracy. Valid performance is achieved using ANN algorithm in diagnosing heart diseases and can be further improved by increasing the number of attributes.

Research Paper-3 [4]

Senthil Kumar Mohan, Chandrasegar Thirumalai et al. proposed "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques" in which strategy that objective is to finding critical includes by applying Machine Learning bringing about improving the exactness in the expectation of cardiovascular disease. The expectation model is created with various blends of highlights and a few known arrangement strategies. In this paper, it produce an improved exhibition level with a precision level of 88.7% through the prediction model for heart disease with hybrid random forest with a linear model (HRFLM) they likewise educated about Diverse data mining approaches and expectation techniques, Such as, KNN, LR, SVM, NN, and Vote have been fairly famous of late to distinguish and predict heart disease.

Research Paper-4 [6]

Chaithra N and Madhu B et al. proposed "Classification Models on Cardiovascular Disease Prediction using DataMining Techniques"performed a comparative analysis using some data mining techniques to design a cardiovascular disease prediction model after analyzing some existing models. Data used was obtained from Transthoracic Echocardiography database, which contains 336 instances and 24 attributes. They used three of the popular machine learning models: DT-J48, Naïve Bayes (NB), and Neural Network (NN) for the analysis and classification processes. The performance measure was done based on False Negative, False Positive, True Negative, True Positive, Precision, Recall, and Accuracy. Three different experiments were conducted. Their experimental results showed that NN model performed much better in heart disease prediction with 97.91% accuracy.

Research Paper-5 [7]

Ashwini Shetty A and Chandra Naik et al. proposed "different data mining approaches for predicting heart disease". Their work analyses the neural network and genetic algorithm to predict heart diseases. The initial weight of the neural network is found using genetic algorithm which is the main advantage of this method. Here, the neural network uses 13 input layers, 10 hidden layers and 2 output layers. The inputs are the attribute layers (here 13 attributes are used namely age, resting heart rate, blood pressure, blood sugar and others). Levenberg-Marquardt back propagation algorithm is used for training and testing. Optimization Toolbox is used to implement this system. 'configure' function is used with neural network where each weight lies between -2 to 2.Fitness function that is being used in the genetic algorithm is the Mean Square Error (MSE). Genetic algorithm is used for adjustment of weights. Based on MSE, fitness function will be calculated for each chromosome. Once selection is done, crossover and mutation in genetic algorithm replaces the chromosome having lower adaption with the better values. Fitter strings are obtained by optimizing the solution which corresponds to interconnecting weights and threshold of neural network. The resulting lower values those are close to zero, represent the generalized format of the network which is ready for classification problem. The system calculates accuracy using MATLAB. Preprocessing is done using WEKA. The results show that the hybrid system of genetic algorithm and neural network works much better than the performance of neural network alone.

Research Paper-6 [8]

Haleh Ayatollahi, Leila Gholamhosseini and Masoud Salehi et al. proposed "Predicting coronary artery disease: a comparison between two data mining algorithms". It is a comparative study between ANN and SVM classification algorithms based on Positive Predictive value (PPV) of cardiovascular diseases. Their data was obtained from three selected hospitals affiliated to AJA University of Medical Sciences, Iran. The sample is composed of 1324 instances and 25 features. The sample is a medical records of patients with coronary artery diseases who were hospitalized in the three mentioned hospitals between March 2016 and March 2017. The data was collected based on the variables used in the guideline of the Cleveland heart disease data policy in UCI machine learning repository. The collected data were controlled using different methods, such data preparation, integration, cleaning, normalization and reduction. The data was fed SPSS (v23.0) and Microsoft Excel 2013, then R 3.3.2 was used for statistical computing. The sample was divided into 70% and 30% for algorithm training and testing respectively. Results of their experiments showed that SVM algorithm presented higher accuracy and better performance than the ANN model, and was characterized by higher power and sensitivity.

CHAPTER-3

HISTORICAL BACKGROUND

Among all fatal disease, heart attacks diseases are considered as the most prevalent. Medical practitioners conduct different surveys on heart diseases and gather information of heart patients, their symptoms and disease progression.

Increasingly are reported about patients with common diseases, who have typical symptoms. In this fast-moving world people want to live a very luxurious life so they work like a machine in order to earn lot of money and live a comfortable life therefore in this race they forget to take care of themselves, because of this their food habits change their entire lifestyle, in this type of lifestyle they are more tensed they have blood pressure, sugar at a very young age and they don't give enough rest for themselves and eat what they get and they even don't bother about the quality of the food if sick the go for their own medication as a result of all these small negligence it leads to a major threat that is the heart disease. The term 'heart disease' includes the diverse diseases that affect heart.

The number of people suffering from heart disease is on the rise (health topics, 2010). The report from world health organization shows us a large number of people that die every year due to the heart disease all over the world. Heart disease is also stated as one of the greatest killers in Africa.

Heart disease can be managed effectively with a combination of lifestyle changes, medicine and, in some cases, surgery. With the right treatment, the symptoms of heart disease can be reduced and the functioning of the heart improved.

The predicted results can be used to prevent and thus reduce cost for surgical treatment and other expensive. The overall objective of my work will be to predict accurately with few tests and attributes the presence of heart disease.

Attributes considered form the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with few attributes and faster efficiency the risk of having heart disease.

CHAPTER-4

IMPLEMENTATION

In order to implement this project, there are some essential steps i.e. data pre-processing, training and testing the data, ML models, and finding the best accuracies of the model.

Data pre-processing is an essential step used to clean the data and make it useful for any experiment associated with machine learning. In this, multiple pre-processing steps applied to the selected dataset. Firstly importing the essential libraries required for the project and then load the external dataset. Check the size of the dataset and make sure that the data in the dataset is error-free then do some of the analytical methods such as correlation, heat map and bar plots to find the relation between variables.

Training and testing the data i.e. the external dataset is divided into two parts, one is used for training the data and the other is used for testing the data. Once training and testing is done, the data is applied to the ML models such as Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest, K-Nearest Neighbor and predict the output for the patients, who will get heart attack or not.

Once we got the predicted values for the above ML models, we can calculate the accuracies for each model and choose the best accuracy model for analyzing and predicting the output for future values.

Now we can see the python implementation, and in the next section, we will see the backend design for mobile application. Below shows the workflow of this project:

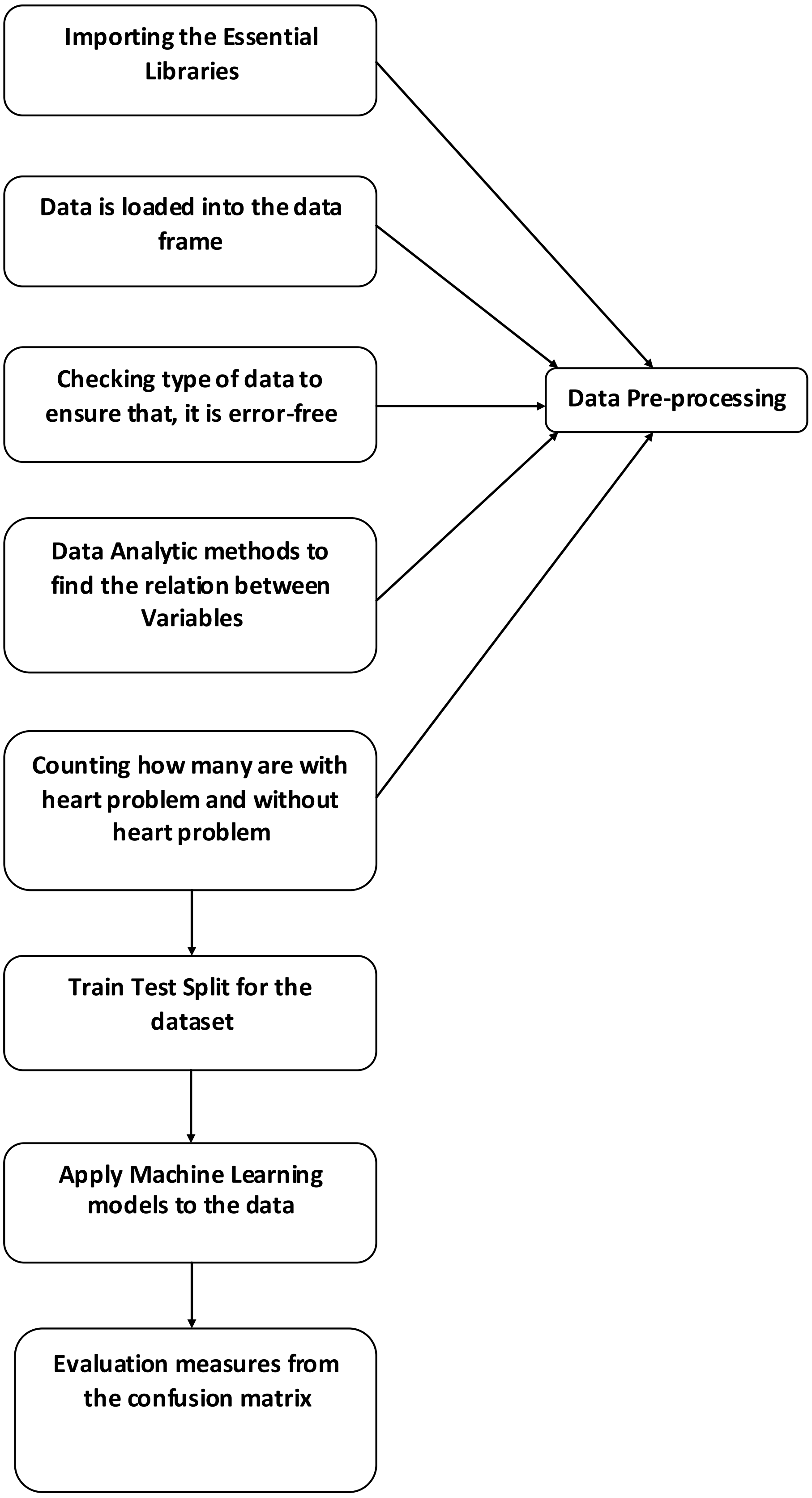

Importing the Essential Libraries
Data is loaded into the data frame
Checking type of data to ensure that, it is error-free
Data Analytic methods to find the relation between Variables
Counting how many are with heart problem and without heart problem
Data Pre-processing
Train Test Split for the dataset
Apply Machine Learning models to the data
Evaluation measures from the confusion matrix

Fig 4.0: Workflow of the Project

From the fig 4.0,

STEP-1: Importing the Essential Libraries

The following are the essential libraries needed for the project.

- Pandas
- NumPy
- Seaborn
- Matplotlib

PANDAS

- ➢ It is an open source library which is used for analyzing the data.
- ➢ It is built on the top of the NumPy library.
- ➢ It has function for analyzing, cleaning, exploring and manipulating data.
- ➢ Pandas is fast, high performance and easy to use.

NUMPY

- ➢ It is a python library which is used for working with arrays.
- ➢ It has functions for working in the domain of linear algebra, Fourier transform and matrices.
- ➢ It is faster than using traditional python lists.

SEABORN

- ➢ It is data visualization library which is used for making statistical graphics.
- ➢ Visualization is the central part of seaborn which helps in exploration and understanding of data.
- ➢ Using seaborn we can plot wide variety of plots like: distribution plots, pie chart and bar chart, scatter plots and heat maps etc.

MATPLOTLIB

- ➢ It is data visualization library which is used for plotting 2D graphs.
- ➢ It uses pyplot for providing the MATLAB like interface free and open source.
- ➢ It is capable of dealing with various operating systems and their graphical back ends.

STEP-2: Data is loaded into Data Frame

In order to load the data into the data frame, the dataset need to be taken. The dataset should be in the format of CSV (Comma Separated Values). CSV files are a common file format for transferring and storing data. The ability to read, manipulate, and write data to and from CSV files using Python is a key skill to master for any data scientist or business analysis.

- Pandas are the most popular data manipulation package in Python and Data Frames are the Pandas data type for storing tabular 2D data.
- The basic process of loading data from a CSV file into a Pandas Data Frame (with all going well) is achieved using the "read_csv" function in Pandas.

Loading the data from a CSV file to pandas Data Frame

```
import pandas as pd
dataset = pd.read_csv("heart.csv")
```

- In the above block, "**heart.csv**" file is the Heart disease dataset, which is downloaded from the Kaggle website, consisting of so many parameters with a size of 303 rows and 14 columns.

Now, let's see some information about parameters which we considered in a dataset [4].

Table 4.1: Parameters of the Heart disease dataset

S.No	Feature Name	code	Description
1	Age	age	age in years
2	Sex	sex	1 = Male 0 = Female
3	Chest Pain Type	cp	**0: Typical angina** : It means chest pain related, decrease blood supply to the heart. These symptoms usually include chest, arm, or jaw pain described as dull, heavy, tight, or crushing. **1: Atypical angina**: It means chest pain not

			related to heart. These symptoms include epigastria, or back pain or pain that is described as burning, stabbing, or characteristics of indigestion. **2: Non anginal pain:** It means chest pain not related to heart. It is typically a problem with the esophagus, such as gastro esophageal reflux disease. Other causes include muscle or bone problems, lung conditions or diseases, stomach problems, stress, anxiety and depression. For this, the patient feels a pressure or squeezing pain behind the breast bone (sternum). Some people also report the pain spreads to the neck, left arm, or back. The pain can last for a few minutes or for hours. **3: Asymptomatic : **It means chest pain not showing signs of disease.
4	Resting Blood Pressure	trtbps	Resting Blood Pressure (in mm Hg on admission to the hospital) anything above 130 - 140 is typically cause for concern.
5	Serum Cholesterol	chol	In mg/dl Serum = LDL + HDL + .2 * triglycerides The serum value above 200 is cause for concern
6	Fasting Blood Sugar > 120mg/dl	fbs	1 = true 0 = false fbs > 126 mg/dl signal diabetes.
7	Resting Electrocardiographic Results	restecg	**0:** Nothing to note **1:**ST-T wave abnormality • can range from mild symptoms to severe problems • signals non-normal heart beat. **2:**Possible or definite left ventricular

			hypertrophy • Enlarged heart's main pumping chamber.
8	Maximum heart rate achieved	thalachh	____
9	Exercise Induced Angina	exang	1 = Yes 0 = No
10	ST depression induced by exercise relative to rest	old peak	_____
11	The slope of peak exercise ST segment	slp	**0: Upsloping** : better heart rate with exercise (uncommon). **1:Flatsloping**: minimal change (typical healthy heart). **2: Downsloping**: signs of unhealthy heart.
12	Number of major vessels (0-3) colored by fluoroscopy.	caa	colored vessels means the doctor can see the blood passing through. The more blood movement the better (no clots).
13	Thallium Stress result	thall	A blood disorder called thalassemia **1: Normal** **2: fixed defect:** used to be defect but ok now. **3:Reversible defect:** No proper blood movement when exercising.
14	Output	output	Healthy = 0 Heart disease patient = 1

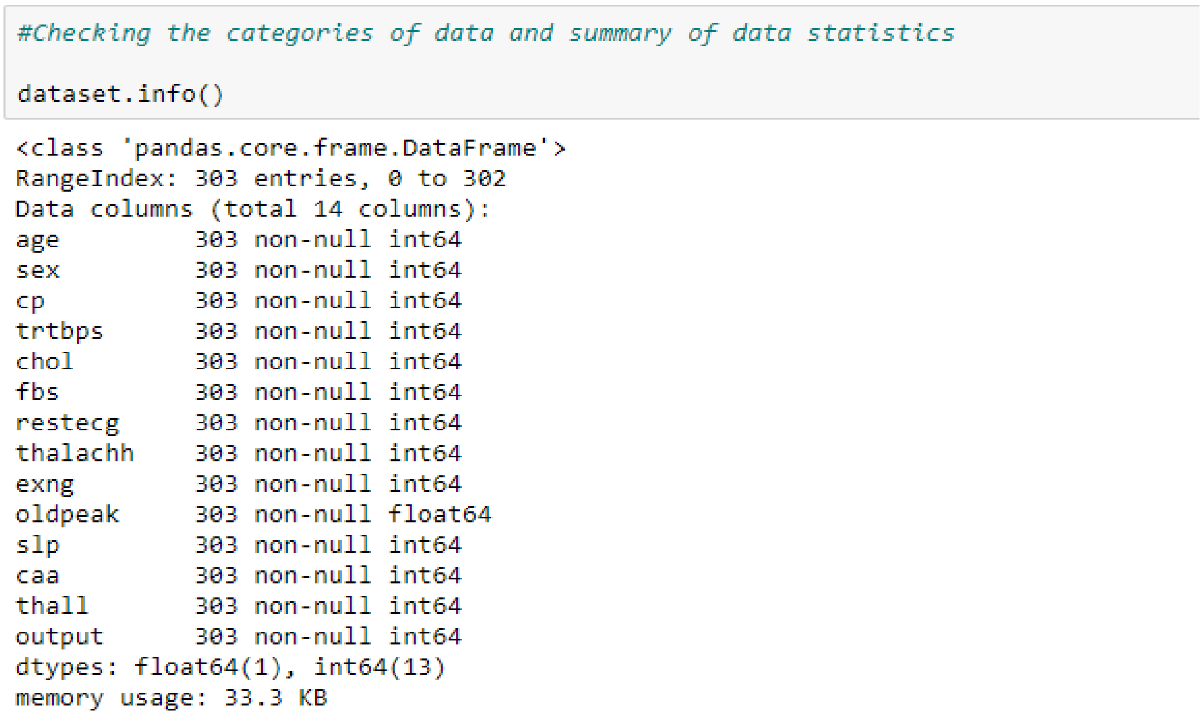
STEP-3: Checking type of data

We do this step because, sometimes for 'int' type data, a 'string' value may be noted or vice versa, we must check in order to safely process data so that, graphs are obtained for analysis.

Checking the type of data in dataset

```
dataset.info()
```

- The above block gives the information of all the parameters in the dataset, to ensure that it is error-free.



```
#Checking the categories of data and summary of data statistics

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trtbps      303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalachh    303 non-null int64
exng        303 non-null int64
oldpeak     303 non-null float64
slp         303 non-null int64
caa         303 non-null int64
thall       303 non-null int64
output      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Fig 4.2: Output for checking what type of data in the dataset

STEP-4: Data Analytic methods to find the relation between Variables

In order to find the relation between the variables, some methods are used for analyzing the data. By this step, we can easily understand, how the parameters are linking with each other and also we can analyze the data, by observing the bar plots.

Some of the data analytic methods are:

- **CORRELATION**

  Correlation analysis is used to quantify the degree to which two variables are related. Through the correlation analysis, you evaluate correlation coefficient that tells you how much one variable changes when the other one does. Correlation analysis provides you with a linear relationship between two variables.

  Correlation can be lies in between -1 to +1. It can be calculated by using "Pearson's r method". Let's see some of the steps for manually calculating the correlation by using Pearson's r method.

  STEP-1: Find out the two parameters to find the correlation and name it as x and y.

  STEP-2: Calculate mean of x and y.

  STEP-3: Calculate the deviation score for each variable. Deviation score means

  Subtracts the mean from each value.

  STEP-4: Perform square of deviation scores to get rid of negative numbers.

  STEP-5: Perform sum of the squared deviation scores (or SS) for x and y.

  STEP-6: Perform the cross product of STEP-2.

  STEP-7: Sum of the cross Product and name it as SP.

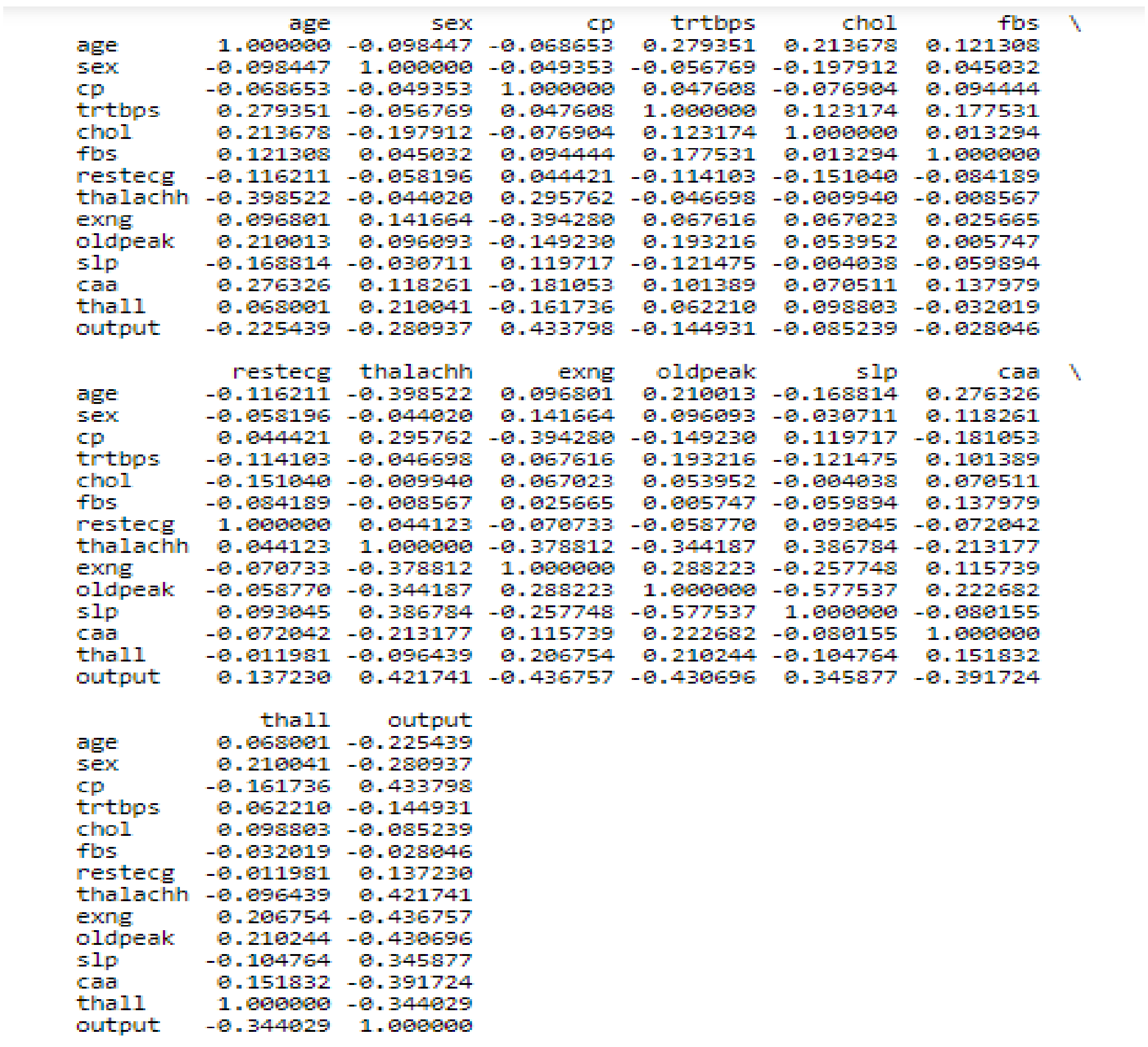
  STEP-8: By using Pearson's r formula:

$$r = \frac{SP}{[\,(\sqrt{\text{SSx}})\;(\sqrt{\text{SSy}})\,]}$$

- If r = 1, it means Positive Correlation and r = -1, it means Negative correlation.
- If r = 0, it means no correlation.

```
dataset.corr()
```

➢ The above block of code gives the relationship between all the parameters.



```
               age       sex        cp    trtbps      chol       fbs  \
age       1.000000 -0.098447 -0.068653  0.279351  0.213678  0.121308
sex      -0.098447  1.000000 -0.049353 -0.056769 -0.197912  0.045032
cp       -0.068653 -0.049353  1.000000  0.047608 -0.076904  0.094444
trtbps    0.279351 -0.056769  0.047608  1.000000  0.123174  0.177531
chol      0.213678 -0.197912 -0.076904  0.123174  1.000000  0.013294
fbs       0.121308  0.045032  0.094444  0.177531  0.013294  1.000000
restecg  -0.116211 -0.058196  0.044421 -0.114103 -0.151040 -0.084189
thalachh -0.398522 -0.044020  0.295762 -0.046698 -0.009940 -0.008567
exng      0.096801  0.141664 -0.394280  0.067616  0.067023  0.025665
oldpeak   0.210013  0.096093 -0.149230  0.193216  0.053952  0.005747
slp      -0.168814 -0.030711  0.119717 -0.121475 -0.004038 -0.059894
caa       0.276326  0.118261 -0.181053  0.101389  0.070511  0.137979
thall     0.068001  0.210041 -0.161736  0.062210  0.098803 -0.032019
output   -0.225439 -0.280937  0.433798 -0.144931 -0.085239 -0.028046

           restecg  thalachh      exng   oldpeak       slp       caa  \
age      -0.116211 -0.398522  0.096801  0.210013 -0.168814  0.276326
sex      -0.058196 -0.044020  0.141664  0.096093 -0.030711  0.118261
cp        0.044421  0.295762 -0.394280 -0.149230  0.119717 -0.181053
trtbps   -0.114103 -0.046698  0.067616  0.193216 -0.121475  0.101389
chol     -0.151040 -0.009940  0.067023  0.053952 -0.004038  0.070511
fbs      -0.084189 -0.008567  0.025665  0.005747 -0.059894  0.137979
restecg   1.000000  0.044123 -0.070733 -0.058770  0.093045 -0.072042
thalachh  0.044123  1.000000 -0.378812 -0.344187  0.386784 -0.213177
exng     -0.070733 -0.378812  1.000000  0.288223 -0.257748  0.115739
oldpeak  -0.058770 -0.344187  0.288223  1.000000 -0.577537  0.222682
slp       0.093045  0.386784 -0.257748 -0.577537  1.000000 -0.080155
caa      -0.072042 -0.213177  0.115739  0.222682 -0.080155  1.000000
thall    -0.011981 -0.096439  0.206754  0.210244 -0.104764  0.151832
output    0.137230  0.421741 -0.436757 -0.430696  0.345877 -0.391724

             thall    output
age       0.068001 -0.225439
sex       0.210041 -0.280937
cp       -0.161736  0.433798
trtbps    0.062210 -0.144931
chol      0.098803 -0.085239
fbs      -0.032019 -0.028046
restecg  -0.011981  0.137230
thalachh -0.096439  0.421741
exng      0.206754 -0.436757
oldpeak   0.210244 -0.430696
slp      -0.104764  0.345877
caa       0.151832 -0.391724
thall     1.000000 -0.344029
output   -0.344029  1.000000
```

Fig 4.3: Output of the Correlation for all the parameters

➢ **HEAT MAP**

A Heat map is a graphical representation of data that uses a system of colour-coding to represent different values.

```
corr_matrix = dataset.corr()
sns.heatmap(corr_matrix, annot=True, cmap="BrBG")
```

- The graphical diagram can be drawn by using seaborn. In order to draw the heat map, we must have the following parameters.
  1. Correlation values
  2. Annot

     If True, write the data value in each cell. If an array-like with the same shape as data, then use this to annotate the heat map instead of the data. Note that Data Frames will match on position, not index.
  3. Cmap

     The mapping from data values to color space. If not provided, the default will depend on whether center is set.

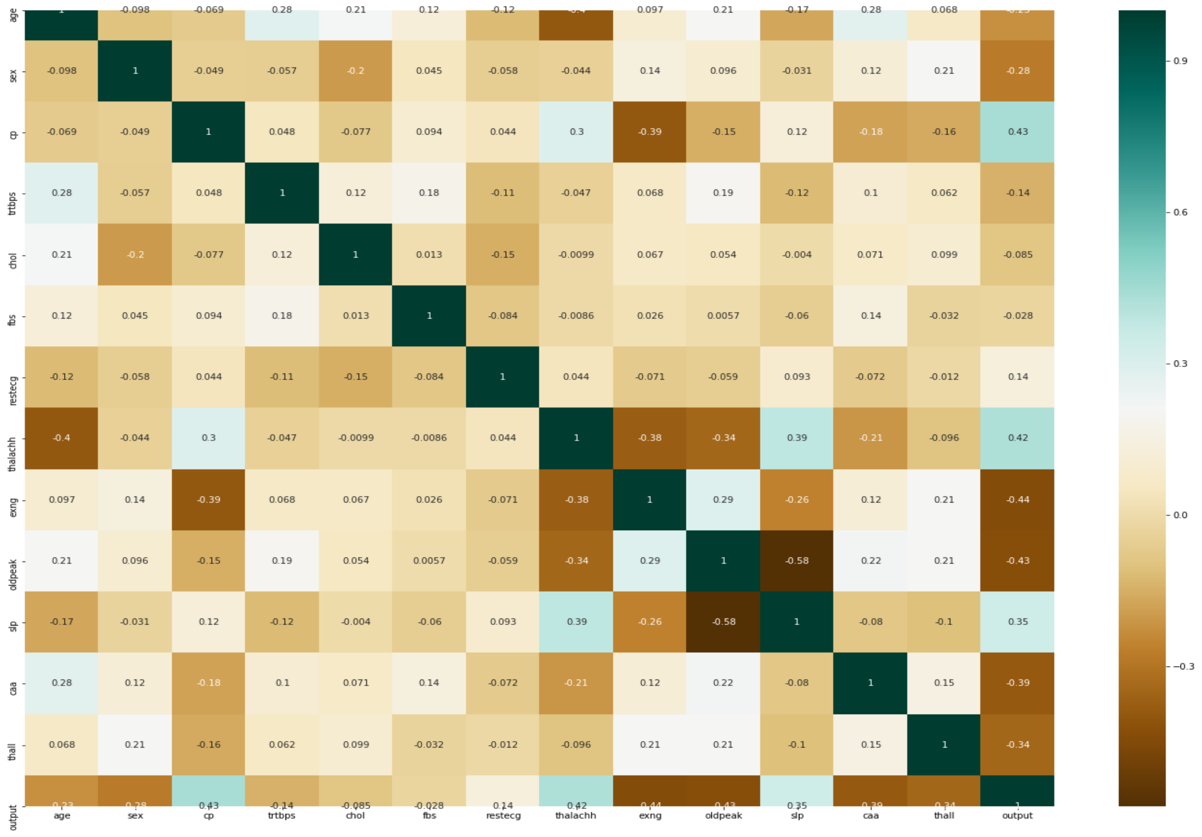


Fig 4.4: Heat map

- **BARPLOTS**

  A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

STEP-5: Counting how many are with heart problem and without heart problem

In this step, we are counting the number of patients with heart problem and without heart problem data in the dataset and the percentage is calculated.

- 1 means with heart problem
- 0 means without heart problem

```
y = dataset["output"]
sns.countplot(y)
target_temp = dataset.output.value_counts()
print(target_temp)
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))
```

- The above block gives the information about the percentage of number of patients, with and without heart problems and drawn the count plot using seaborn.
- In the above dataset, 165 patients are suffering with heart problem and 138 patients are not suffering with heart problem.
- The output for the last two lines in the above block is:

  Percentage of patients without heart problems: 45.54

  Percentage of patients with heart problems: 54.46

```
1    165
0    138
Name: output, dtype: int64
```

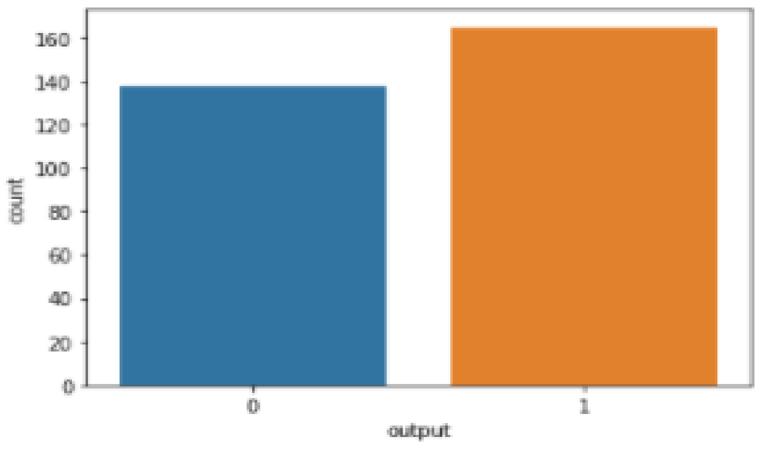


Fig 4.5: Output for counting the patients with and without heart problem

STEP-6: Train Test Split for the dataset

Before discussing train_test_split, you should know about Sklearn (or Scikit-learn). It is a Python library that offers various features for data processing that can be used for classification, clustering, and model selection.

Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.

To do that, you need to train your model by using a specific dataset. Then, you test the model against another dataset. If you have one dataset, you'll need to split it by using the Sklearn train_test_split function first.

Train_Test_Split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

```
from sklearn.model_selection import train_test_split

predictors = dataset.drop("output", axis=1)

target = dataset["output"]

X_train,X_test,Y_train,Y_test =

train_test_split(predictors, target, train_size=0.8, test_size=0.2, random_state=0)
```

- In the above block, Predictors gives all the columns except output and target gives only output column. This gives as an input to the train_test_split.
- **Train_size:** This parameter sets the size of the training dataset. In this project, we take 80% as training data. The default size of training data is 25%.

  Normally, there are three options: None, which is the default, Int, which requires the exact number of samples, and float, which ranges from 0.1 to 1.0.

- **Test_size:** This parameter specifies the size of the testing dataset. In this project, we take 20% as testing data. The default size of testing data is 25%.
- **Random_state:** Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls.
- After splitting is done,

  X_train consists of 242 rows and 13 columns.

  Y_train consists of 242 rows and 1 column.

  X_test consists of 61 rows and 13 columns.

  Y_test consists of 61 rows and 1 columns.

STEP-7: Apply Machine Learning models to the data

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

As we taken the classification heart disease dataset, there are several machine learning algorithms in order to analyze and predict the data. In this project, we took machine learning algorithms such as Logistic Regression, Naïve Bayes, Support Vector Machine, Random Forest and K-Nearest Neighbor.

Let us discuss the above algorithms one by one:

1. LOGISTIC REGRESSION ALGORITHM

Logistic regression(LR) is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is Dichotomous (which means there would be only two possible classes).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as heart attack prediction, Diabetes prediction, cancer detection etc [5].

TYPES OF LOGISTIC REGRESSION

Based on the category, logistic regression can be classified into three types. They are:

- **Binomial or Binary :** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, Yes or No etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep".
- **Ordinal :** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

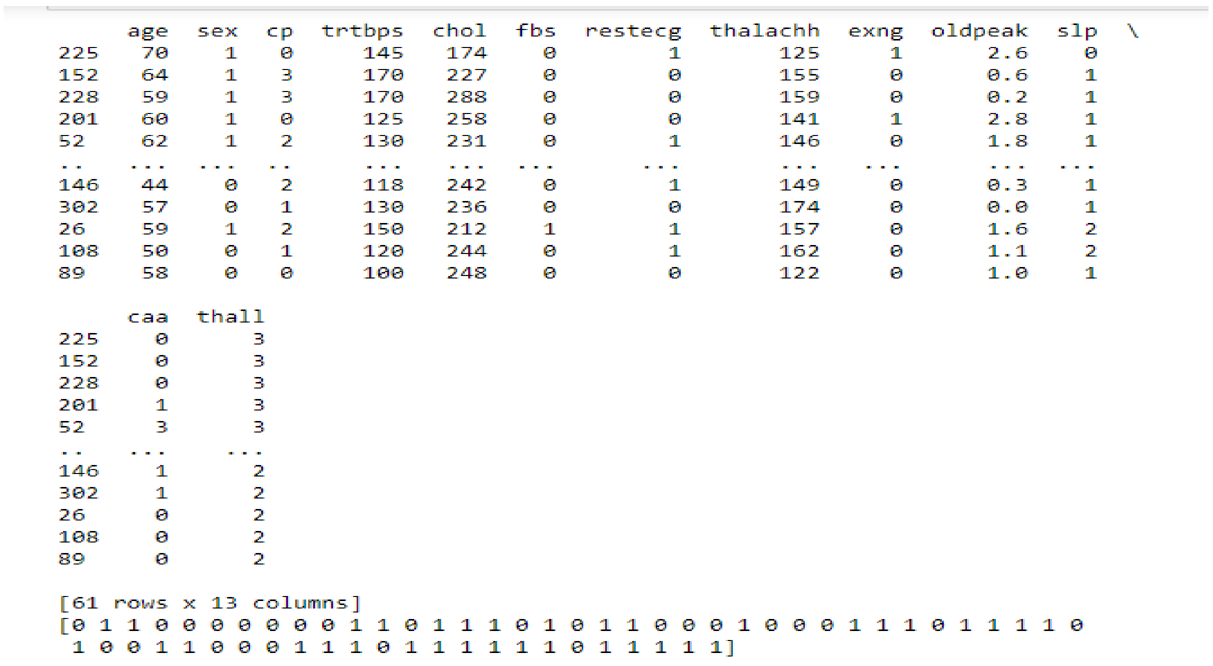
In this project, we use Binomial or Binary type of Logistic Regression, which gives the output as 0 or 1. Below is the code for Logistic Regression algorithm

```
from sklearn.linear_model import LogisticRegression

logreg=LogisticRegression()

logreg.fit(X_train,Y_train)
```

- We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the LogisticRegression class of the sklearn library. After importing the class, we will create a logreg object and use it to fit the model to the logistic regression.

```
Y_predictor_logreg=logreg.predict(X_test)
```

- Once after fitting the model, we will now predict the result by using test set data.



```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
225   70    1   0     145   174    0        1       125     1      2.6    0
152   64    1   3     170   227    0        0       155     0      0.6    1
228   59    1   3     170   288    0        0       159     0      0.2    1
201   60    1   0     125   258    0        0       141     1      2.8    1
52    62    1   2     130   231    0        1       146     0      1.8    1
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
146   44    0   2     118   242    0        1       149     0      0.3    1
302   57    0   1     130   236    0        0       174     0      0.0    1
26    59    1   2     150   212    1        1       157     0      1.6    2
108   50    0   1     120   244    0        1       162     0      1.1    2
89    58    0   0     100   248    0        0       122     0      1.0    1

     caa  thall
225    0      3
152    0      3
228    0      3
201    1      3
52     3      3
..   ...    ...
146    1      2
302    1      2
26     0      2
108    0      2
89     0      2

[61 rows x 13 columns]
[0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Fig 4.6: Predicted output of the Logistic Regression algorithm

2. NAIVE BAYES ALGORITHM

Naïve Bayes(NB) classifier is a supervised learning algorithm, which is used for solving classification problems. It is based on Bayes theorem. Naive Bayes classifier is one most effective algorithm which helps in building the fast machine learning models that can make quick predictions. It predicts the output on the basis of the probability of an object[2].

BAYES THEOREM

Bayes theorem is stated as probability of the event B given A is equal to the probability of the event A given B multiplied by the probability of A upon probability of B.

The formula for Bayes theorem is given as:

$$P(A|B) = [(P(B|A)P(A)]/P(B)$$

Where, **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

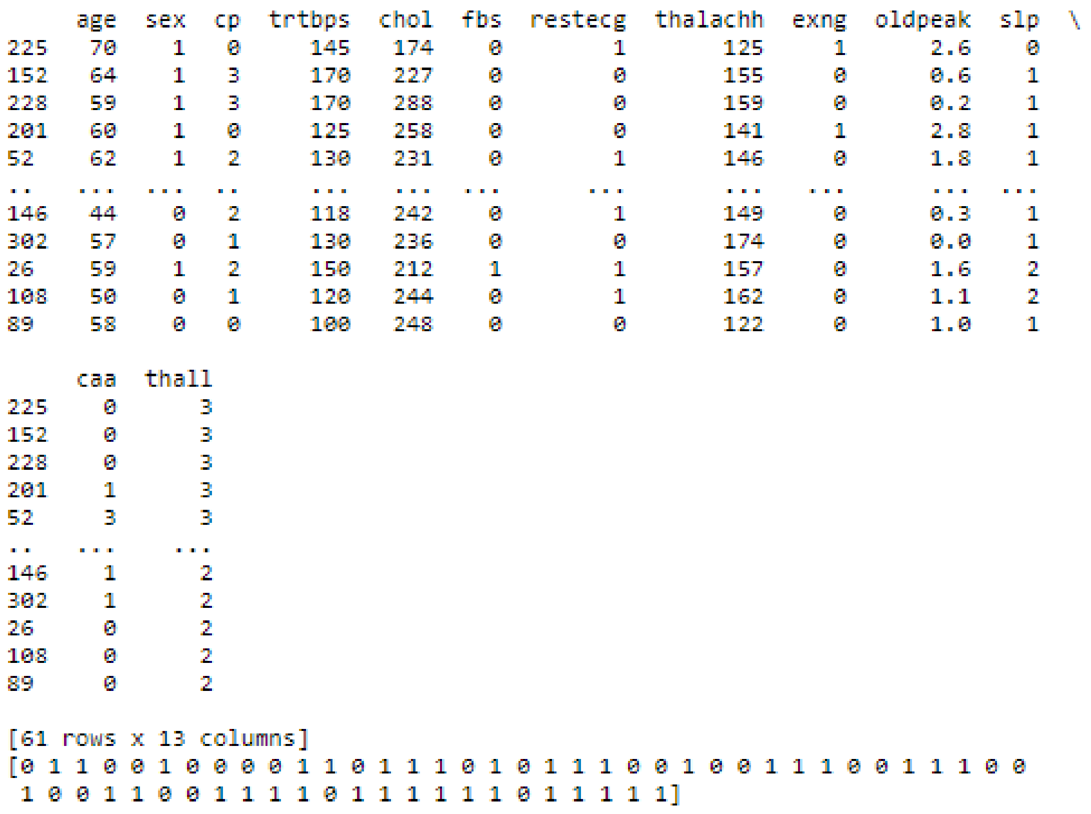
Below is the code for Naïve Bayes algorithm

```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(X_train,Y_train)
```

- We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the GaussianNB class of the sklearn library. After importing the class, we will create a NB object and use it to fit the model to the Naïve Bayes.

```
Y_predictor_nb=NB.predict(X_test)
```

- Once after fitting the model, we will now predict the result by using test set data.



```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
225   70    1   0     145   174    0        1       125     1      2.6    0
152   64    1   3     170   227    0        0       155     0      0.6    1
228   59    1   3     170   288    0        0       159     0      0.2    1
201   60    1   0     125   258    0        0       141     1      2.8    1
52    62    1   2     130   231    0        1       146     0      1.8    1
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
146   44    0   2     118   242    0        1       149     0      0.3    1
302   57    0   1     130   236    0        0       174     0      0.0    1
26    59    1   2     150   212    1        1       157     0      1.6    2
108   50    0   1     120   244    0        1       162     0      1.1    2
89    58    0   0     100   248    0        0       122     0      1.0    1

     caa  thall
225    0      3
152    0      3
228    0      3
201    1      3
52     3      3
..   ...    ...
146    1      2
302    1      2
26     0      2
108    0      2
89     0      2

[61 rows x 13 columns]
[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0
 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Fig 4.7: Predicted output of the Naïve Bayes algorithm

3. SUPPORT VECTOR MACHINE ALGORITHM

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used for solving classification and regression problems. However, it is primarily used for classification problems in ML. It is good because it gives reliable results even if there is less data [3].

An SVM model is basically a representation of different classes in a hyper plane in multidimensional space. The hyper plane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane (MMH) and it can be done in the following two steps:

1. SVM will generate hyper planes iteratively that segregates the classes in best way.

2. Then, It will choose the hyper plane that separates the classes correctly.

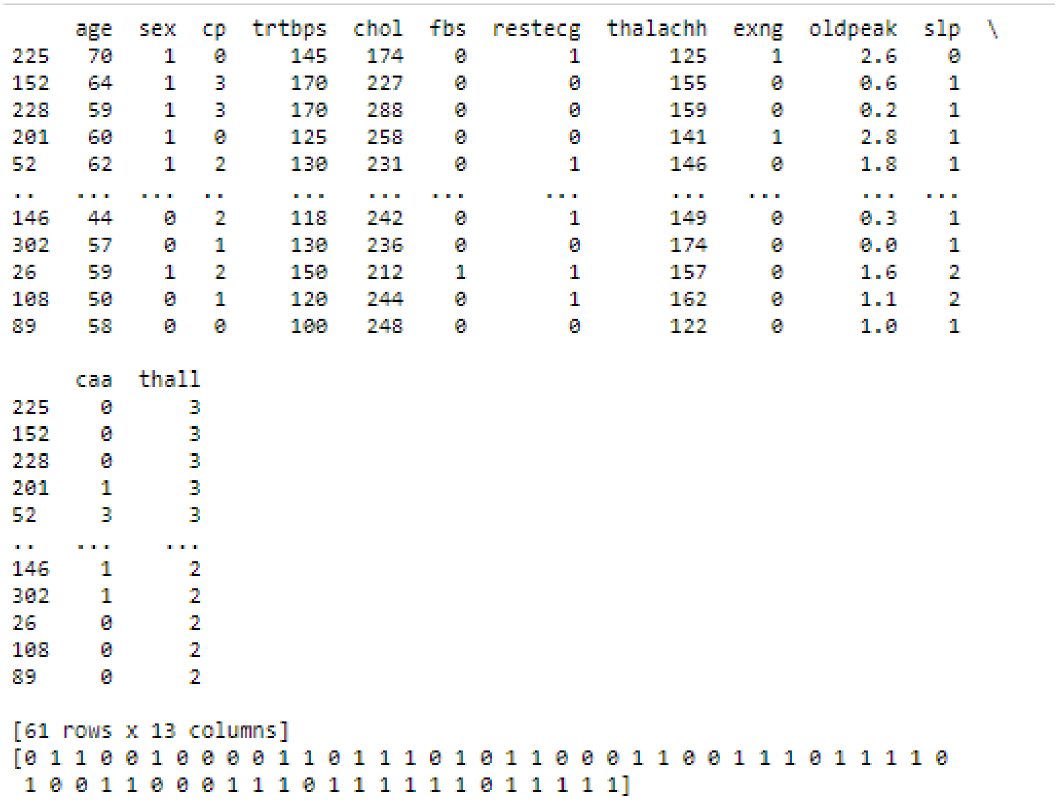
Below is the code for Support Vector Machine algorithm

```
from sklearn import svm
sv = svm.SVC(kernel='linear')
sv.fit(X_train, Y_train)
```

- We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the svm class of the sklearn library. After importing the class, we will create a sv object and use it to fit the model to the Support Vector Machine.

```
Y_predictor_svm=sv.predict(X_test)
```

- Once after fitting the model, we will now predict the result by using test set data.



```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
225   70    1   0     145   174    0        1       125     1      2.6    0
152   64    1   3     170   227    0        0       155     0      0.6    1
228   59    1   3     170   288    0        0       159     0      0.2    1
201   60    1   0     125   258    0        0       141     1      2.8    1
52    62    1   2     130   231    0        1       146     0      1.8    1
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
146   44    0   2     118   242    0        1       149     0      0.3    1
302   57    0   1     130   236    0        0       174     0      0.0    1
26    59    1   2     150   212    1        1       157     0      1.6    2
108   50    0   1     120   244    0        1       162     0      1.1    2
89    58    0   0     100   248    0        0       122     0      1.0    1

     caa  thall
225    0      3
152    0      3
228    0      3
201    1      3
52     3      3
..   ...    ...
146    1      2
302    1      2
26     0      2
108    0      2
89     0      2

[61 rows x 13 columns]
[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Fig 4.8: Predicted output of the Support Vector Machine algorithm

4. RANDOM FOREST ALGORITHM

Random Forest(RF) is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in Machine Learning. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting. The below figure explains the working of Random Forest Algorithm[3].

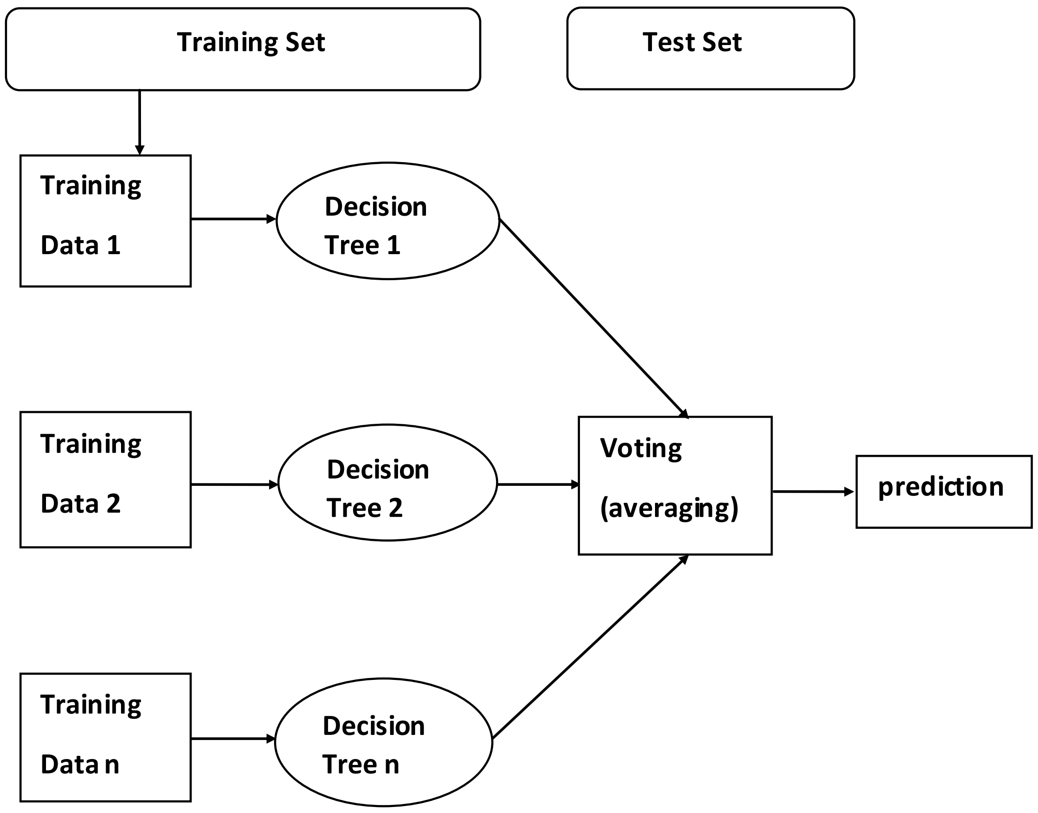


Fig 4.9: Working of Random Forest Algorithm

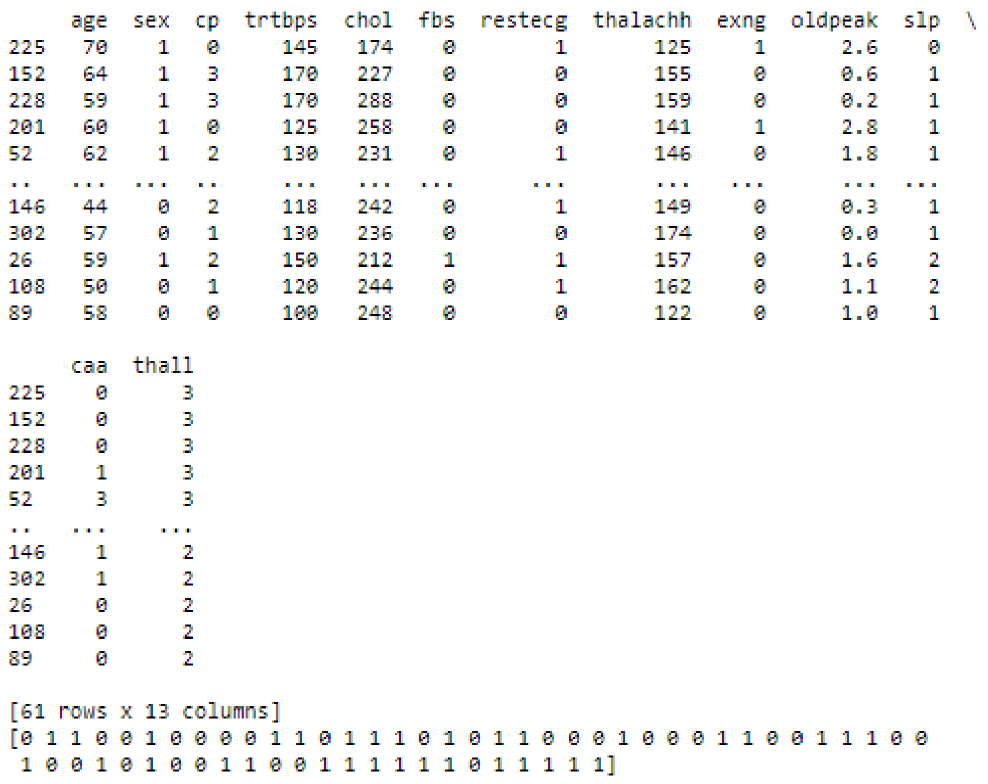
Below is the code for Random Forest algorithm

```
from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(10):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x
rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
```

- We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the RandomForestClassifier of the sklearn.ensemble library. After importing the class, we will create an rf object and use it to fit the model to the Random Forest.
- Here we process this, iteratively. Every time, once after fitting the model, we will check the accuracy. After successful completion of 10 iterations, we will get the best accuracy and also we will check, which iteration gives the best accuracy and store it on the best_x variable.
- Now, once again we will fit the model to the Random Forest.

```
Y_pred_rf = rf.predict(X_test)
```

- Once after fitting the model, we will now predict the result by using test set data.



```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
225   70    1   0     145   174    0        1       125     1      2.6    0
152   64    1   3     170   227    0        0       155     0      0.6    1
228   59    1   3     170   288    0        0       159     0      0.2    1
201   60    1   0     125   258    0        0       141     1      2.8    1
52    62    1   2     130   231    0        1       146     0      1.8    1
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
146   44    0   2     118   242    0        1       149     0      0.3    1
302   57    0   1     130   236    0        0       174     0      0.0    1
26    59    1   2     150   212    1        1       157     0      1.6    2
108   50    0   1     120   244    0        1       162     0      1.1    2
89    58    0   0     100   248    0        0       122     0      1.0    1

     caa  thall
225    0      3
152    0      3
228    0      3
201    1      3
52     3      3
..   ...    ...
146    1      2
302    1      2
26     0      2
108    0      2
89     0      2

[61 rows x 13 columns]
[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0
 1 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Fig 4.10: Predicted output of the Random Forest algorithm

5. K-NEAREST NEIGHBOR ALGORITHM

K-Nearest Neighbor(K-NN) is one of the simplest Machine Learning algorithms based on the Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories [3].

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for Classification problems.

The following two properties would define K-NN:

1. Lazy Learning algorithm: K-NN is a Lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

2. Non-parametric algorithm: K-NN is also non-parametric learning algorithm because it doesn't assume anything about the underlying data.

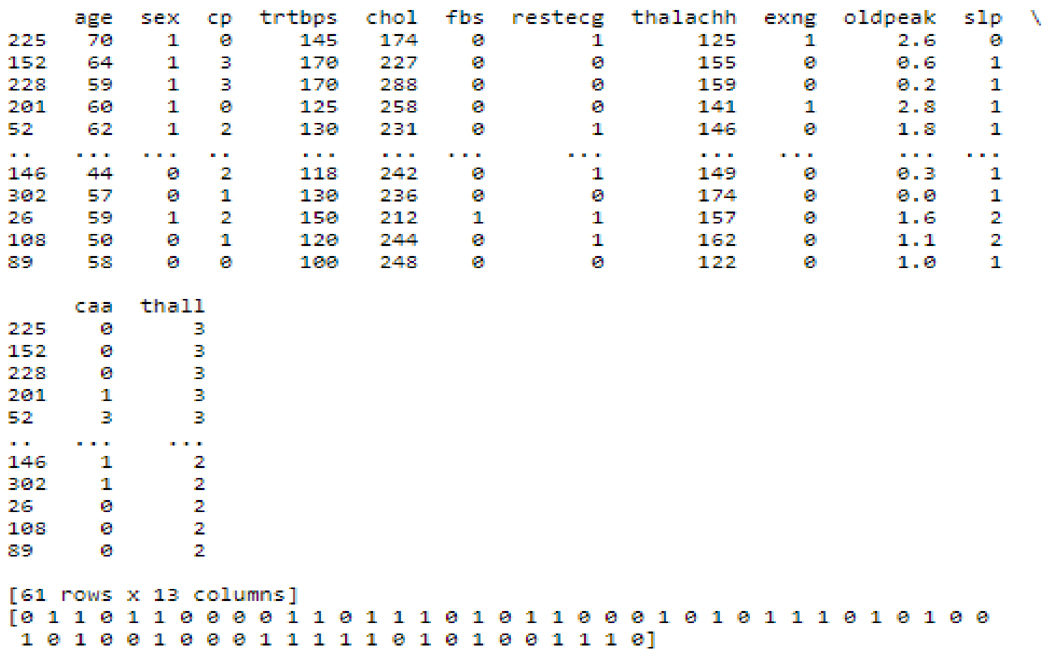
Below is the code for K-Nearest Neighbor algorithm

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=21)
knn.fit(X_train,Y_train)
```

- ➢ We have well prepared our dataset, and now we will train the dataset using the training set. Now we will fit the K-NN classifier into the training data. To do this we will import the KNeighborsClassifier of Sklearn.neighbors library. After importing the class, we will create the knn object of the class. The Parameter of this class will be
    - **n_neighbors:** To define the required neighbors of the algorithm.
- ➢ Now we will fit the model to the K-Nearest Neighbor.

```
Y_pred_knn = knn.predict(X_test)
```

- ➢ Once after fitting the model, we will now predict the result by using test set data.



```
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
225   70    1   0     145   174    0        1       125     1      2.6    0
152   64    1   3     170   227    0        0       155     0      0.6    1
228   59    1   3     170   288    0        0       159     0      0.2    1
201   60    1   0     125   258    0        0       141     1      2.8    1
52    62    1   2     130   231    0        1       146     0      1.8    1
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
146   44    0   2     118   242    0        1       149     0      0.3    1
302   57    0   1     130   236    0        0       174     0      0.0    1
26    59    1   2     150   212    1        1       157     0      1.6    2
108   50    0   1     120   244    0        1       162     0      1.1    2
89    58    0   0     100   248    0        0       122     0      1.0    1

     caa  thall
225    0      3
152    0      3
228    0      3
201    1      3
52     3      3
..   ...    ...
146    1      2
302    1      2
26     0      2
108    0      2
89     0      2

[61 rows x 13 columns]
[0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0
 1 0 1 0 0 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0]
```

Fig 4.11: Predicted output of K-Nearest Neighbor algorithm

STEP-8: Evaluation measures from the confusion matrix

CONFUSION MATRIX:

The confusion matrix is a two by two table that contains four outcomes produced by a binary classifier. Various measures, such as accuracy, inaccuracy(error-rate), specificity, sensitivity, and precision, are derived from the confusion matrix.

A binary classifier predicts all data instances of a test dataset as either positive or negative. This classification (or prediction) produces four outcomes – true positive, true negative, false positive and false negative.

- True positive (TP): correct positive prediction
- False positive (FP): incorrect positive prediction
- True negative (TN): correct negative prediction
- False negative (FN): incorrect negative prediction

Table 4.12: Confusion Matrix

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative(TN)	False Positive(FP)
	Positive	False Negative(FN)	True Positive (TP)

- To implement the confusion matrix in python, we need to import the confusion_matrix from sklearn.metrics library.

Inaccuracy (error-rate) and accuracy are the most common and intuitive measures derived from the confusion matrix.

INACCURACY (ERROR-RATE):

Inaccuracy is calculated as the number of all incorrect predictions divided by the total number of the dataset. It is also known as error-rate. The best error rate is 0.0, whereas the worst is 1.0.

$$\text{Inaccuracy} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

ACCURACY:

Accuracy is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by 1 – Inaccuracy.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Other basic measures from the confusion matrix:

SENSITIVITY:

Sensitivity is calculated as the number of correct positive predictions divided by the total number of positives. It is also called recall (REC) or true positive rate (TPR). The best sensitivity is 1.0, whereas the worst is 0.0.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

SPECIFICITY:

Specificity is calculated as the number of correct negative predictions divided by the total number of negatives. It is also called true negative rate (TNR). The best specificity is 1.0, whereas the worst is 0.0.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

PRECISION:

Precision is calculated as the number of correct positive predictions divided by the total number of positive predictions. It is also called positive predictive value (PPV). The best precision is 1.0, whereas the worst is 0.0.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Calculation of accuracies, precision, specificity, sensitivity for all algorithms can be seen in the Result section.

CHAPTER-5

BACKEND DESIGN

5.1 BACKEND DEVELOPMENT

Back-end Development refers to the server-side development. It focuses on databases, scripting, and website architecture. It contains behind-the-scene activities that occur when performing any action on a website. It can be an account login or making a purchase from an online store. Code written by back-end developers helps browsers to communicate with database information.

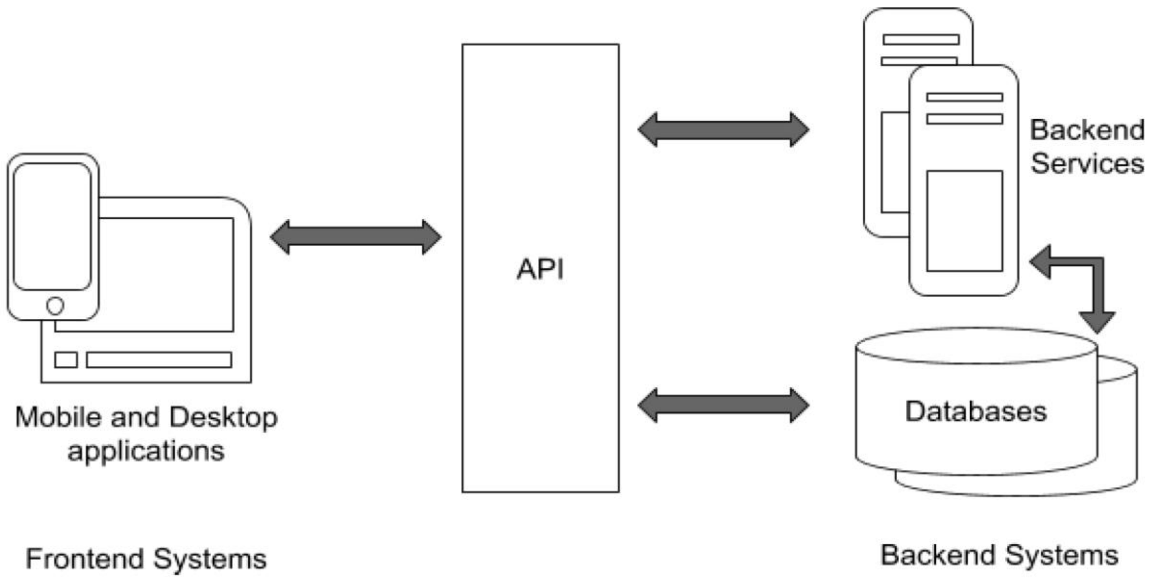


Fig 5.1: Backend design architecture

Most common example of Backend programming is when you are reading an article on the blog. The fonts, colors, designs, etc. constitute the frontend of this page. While the content of the article is rendered from a server and fetched from a database. This is the backend part of the application.

Backend can be done with most of the popular languages such as Java, JavaScript, Ruby, Python but we choose python for simplicity.

The entire machine learning modals and algorithms are done with Python but the code only runs through the terminal or using Jupiter Notebook and only on the single computer.

So, we migrated all our Python Machine Learning Algorithms into a backend API which can run on a server and get access to all the users remotely with a URL.

By sending data using the API, the given data is processed by the algorithms in the server which then sends a JSON Response or the data. The response contains useful information such as predicted output, accuracy and machine learning algorithm used.

5.2 RESTFUL API DESIGN

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

An API is a set of definitions and protocols for building and integrating application software. All the data that we sent to the server with the Rest API. The REST API takes the input from the user using a POST Method which contains a form data. The form data having useful information that can be processed on the server.

- **GET:** This is used to send the data in a without encryption of form from the server.
- **HEAD:** Provides response body to the form
- **POST:** Sends the data to the server. Data received the post method is not cached by the server.
- **PUT:** Replaces current representation of target resource with URL.
- **DELETE:** Deletes the target from the database

The code parameter can take the following values to handle the error accordingly:

- **400** – For Bad Request
- **401** – For Unauthenticated
- **403** – For Forbidden request
- **404** – For Not Found
- **406** – For Not acceptable
- **425** – For Unsupported Media
- **429** – Too many Requests

5.3 DATABASE

Generally, in the backend applications, we use database to store user's information such as emails, usernames. The database is connected using backend languages. But in our case, we don't use backend since we are not storing any user's information. The data that users pass is collected through the API, process the data in the server and return some JSON Response.

5.4 SERVER

A server is simply a computer that listens for incoming requests. Though there are machines made and optimized for this particular purpose, any computer that is connected to a network can act as a server. In fact, you will often use your very own computer as server when developing apps.

A server is 24/7 online so if anyone sends a request to the server it responds immediately with the response that user needs.

The server runs our Machine Learning Python App that contains logic and implementation of the heart attack prediction. The server takes gets the requests and know about how to respond to various requests based on the http and the URI. The pair of an HTTP verb and a URI is called a route and matching them based on a request is called "routing"**.**

Some of these handler functions will be middleware. In this context, middleware is any code that executes between the server receiving a request and sending a response. These middleware functions might modify the request object, query the database, or otherwise process the incoming request. Middleware functions typically end by passing control to the next middleware function, rather than by sending a response.

Eventually, a middleware function will be called that ends the request-response cycle by sending an HTTP response back to the client. Often, servers can run frameworks like Express, Flask or Ruby on Rails to simplify the logic of routing.

5.5 FLASK FRAMEWORK

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO).

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask’s framework is more explicit than Django’s framework and is also easier to learn because it has less base code to implement a simple web-Application.

A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine.

There are many other python web frameworks that we can use such as Django but they increase complexity since we don’t need of any database connectivity and complex routing.

So flask is the great and simple framework that we used to process our data in server. All the machine learning algorithms are moved to the flask code which can be useful for processing of data.

5.6 HEROKU

Heroku is a container-based cloud Platform as a Service (PaaS). It is used to deploy, manage, and scale modern apps. It is elegant, flexible, and easy to use. There are many other cloud based services such as AWS, Google Cloud, Digital ocean but in this project, we used Heroku to deploy application written in Python, Node.js.

5.7 ANDROID APP DEVELOPMENT

So, the backend is designed and ready for use. To send data to backend or server we normally use some complex software’s such as Postman to send data. All people don’t use this type of software’s to get his data he needs.

The backend or the server returns a JSON response, which is representation of output in object notation. This object notation is good for computers but not for users.

To fix these problems we came up with a mobile application which takes all the users' inputs through form fields. The fields such as Age, Gender, Chest pain type etc. The user needs to fill the data from the app and click on submit. The app sends all the information that used entered in his device. The data will be collected and sent to the server.

The server processes the data that user submitted and sends a response. This response is not user friendly. So we made an another screen which collects the data received from the server and represents in user friendly format.

We used technologies such as React Native for android app development and Axios for connecting android app to the backend API.

5.8 REACT NATIVE

React Native is a JavaScript framework used for developing a real, native mobile application for iOS and Android. It uses only JavaScript to build a mobile application. It is like React, which uses native component rather than using web components as building blocks.

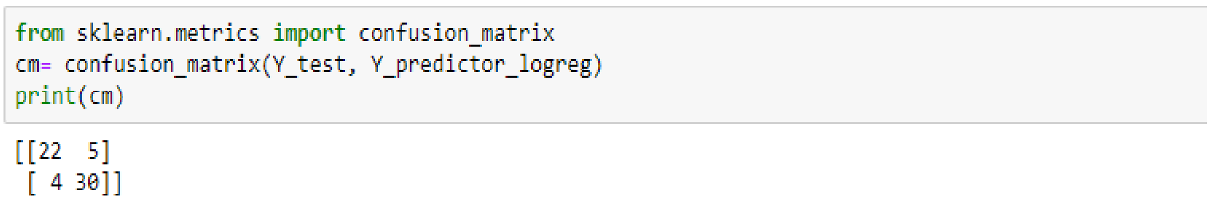
React Native is based on React, JavaScript library of Facebook, and XML-esque markup (JSX) for developing the user interface. It targets the mobile platform rather than the browser.

React Native apps are not web application. They are running on a mobile device, and it does not load over the browser. It is also not a Hybrid app that builds over Ionic, Phone Gap, etc. that runs over Web View component. React Native apps are the real native app, the JavaScript code stays as JavaScript, and they run in some extra thread by the compiled app. The user interface and everything are compiled to native code.

CHAPTER-6

RESULTS

For Logistic Regression algorithm,



```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test, Y_predictor_logreg)
print(cm)
```

```
[[22  5]
 [ 4 30]]
```

Fig 6.0: Confusion Matrix for Logistic Regression algorithm

In the above fig, 22+30 = 52 correct predictions and 5+4 = 9 incorrect predictions.

Accuracy = (52/61)*100 = 85.25%

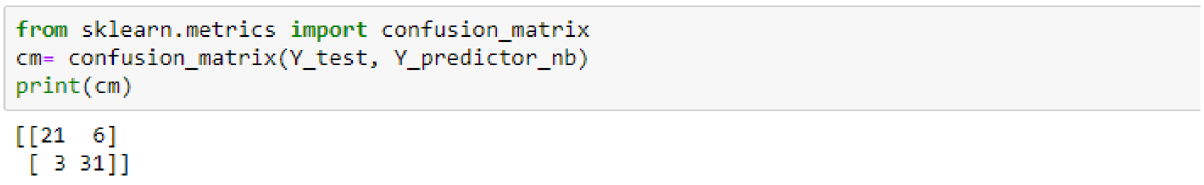
Inaccuracy = (9/61)*100 = 14.75%

Sensitivity = (30/(30+4)) = (30/34) = 0.88

Specificity = (22/(22+5)) = (22/27) = 0.81

Precision = (30/(30+5)) = (30/35) = 0.86

For Naïve Bayes algorithm,



```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test, Y_predictor_nb)
print(cm)
```

```
[[21  6]
 [ 3 31]]
```

Fig 6.1: Confusion Matrix for Naïve Bayes algorithm

In the above fig, 21+31 = 52 correct predictions and 6+3 = 9 incorrect predictions.

Accuracy = (52/61)*100 = 85.25%

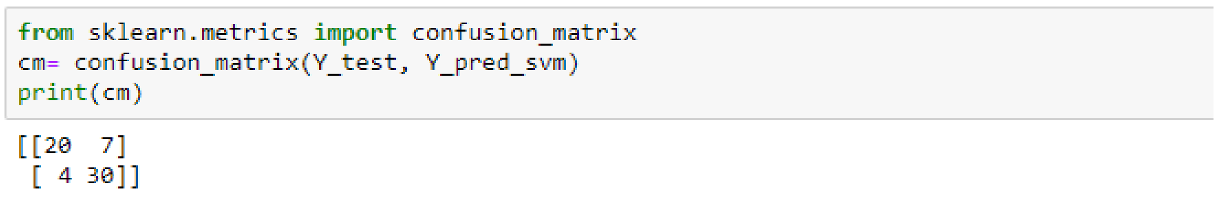
Inaccuracy = (9/61)*100 = 14.75%

Sensitivity = (31/(31+3)) = (31/34) = 0.91

Specificity = (21/(21+6)) = (21/27) = 0.78

Precision = (31/(31+6)) = (31/37) = 0.84

For Support Vector Machine algorithm,



```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test, Y_pred_svm)
print(cm)
```

```
[[20  7]
 [ 4 30]]
```

Fig 6.2: Confusion Matrix Support Vector Machine algorithm

In the above fig, 20+30 = 50 correct predictions and 7+4 = 11 incorrect predictions.

Accuracy = (50/61)*100 = 81.97%

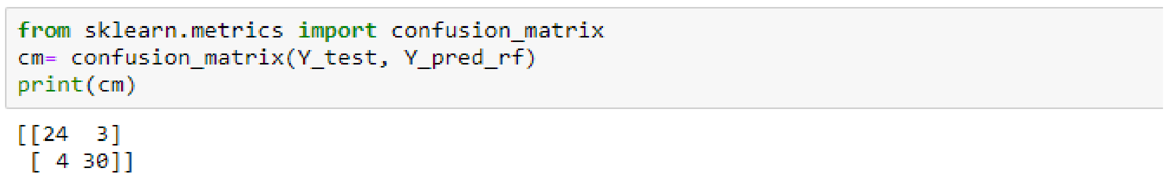
Inaccuracy = (11/61)*100 = 18.03%

Sensitivity = (30/(30+4)) = (30/34) = 0.88

Specificity = (20/(20+7)) = (20/27) = 0.74

Precision = (30/(30+7)) = (30/37) = 0.81

For Random Forest algorithm,



```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test, Y_pred_rf)
print(cm)
```

```
[[24  3]
 [ 4 30]]
```

Fig 6.3: Confusion Matrix for Random Forest algorithm

In the above fig, 24+30 = 54 correct predictions and 3+4 = 7 incorrect predictions.

Accuracy = (54/61)*100 = 88.52%

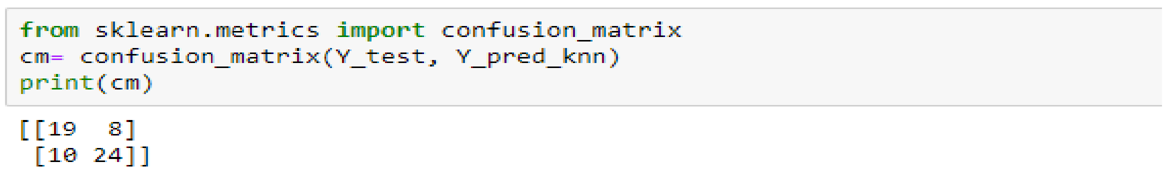
Inaccuracy = (7/61)*100 = 11.47%

Sensitivity = (30/(30+4)) = (30/34) = 0.88

Specificity = (24/(24+3)) = (24/27) = 0.89

Precision = (30/(30+3)) = (30/33) = 0.91

For K-Nearest Neighbor algorithm,



```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test, Y_pred_knn)
print(cm)
```

```
[[19  8]
 [10 24]]
```

Fig 6.4: Confusion Matrix for K-Nearest Neighbor algorithm

In the above fig, 19+24 = 43 correct predictions and 8+10 = 18 incorrect predictions.

Accuracy = (43/61)*100 = 70.49%

Inaccuracy = (18/61)*100 = 29.51%.

Sensitivity = (24/(24+10)) = (24/34) = 0.71

Specificity = (19/(19+8)) = (19/27) = 0.70

Precision = (24/(24+8)) = (24/32) = 0.75

Table 6.5: Evaluation measures table for ML algorithms

Classifier	**Accuracy(%)**	**Inaccuracy(%)**	**Sensitivity**	**Specificity**	**Precision**
LR	85.25	14.75	0.88	0.81	0.86
NB	85.25	14.75	0.91	0.78	0.84
SVM	81.97	18.03	0.88	0.74	0.81
RF	88.52	11.48	0.88	0.89	0.91
KNN	70.49	29.51	0.71	0.70	0.75

DISTPLOTS:

- A Distplot or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables.
- The seaborn.distplot() function accepts the data variable as an argument and returns the plot with the density distribution.
- Below shows the distribution plots for all algorithms

```
sns.distplot(Y_test-Y_predictor_logreg)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f97b9e7208>
```

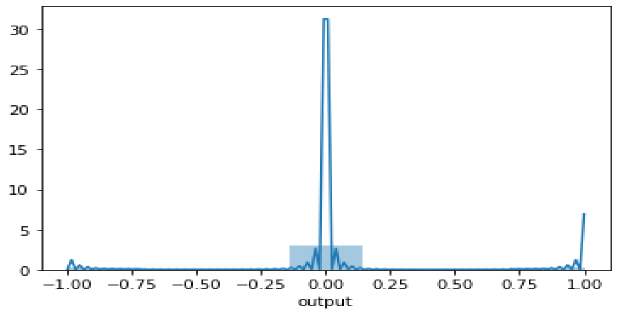


Fig 6.6: Distribution plot for Logistic Regression algorithm

```
sns.distplot(Y_test-Y_predictor_nb)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f97bb23bc8>
```

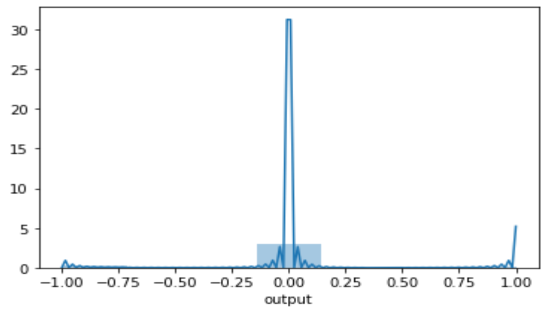


Fig 6.7: Distribution plot for Naïve Bayes algorithm

```
sns.distplot(Y_test-Y_pred_svm)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f97bbb54c8>
```

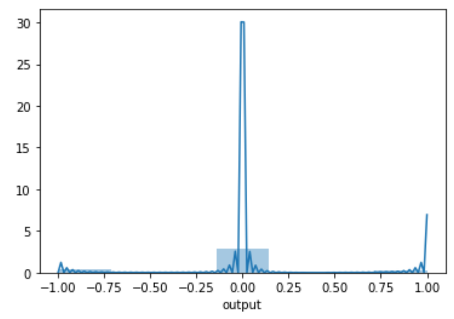


Fig 6.8: Distribution plot for Support Vector Machine algorithm

```
sns.distplot((Y_test-Y_pred_rf))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f97be38e88>
```

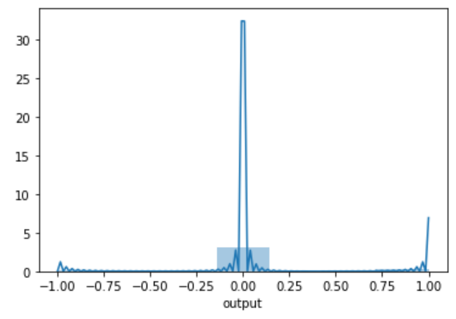


Fig 6.9: Distribution plot for Random Forest algorithm

```
sns.distplot((Y_test-Y_pred_knn))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f97be63388>
```

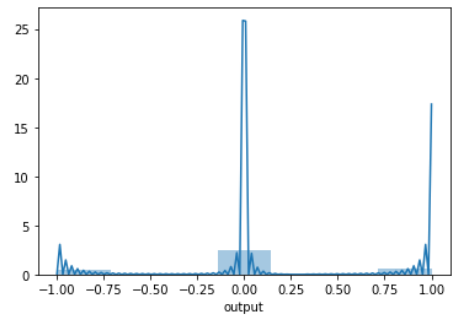


Fig 6.10: Distribution plot for K-Nearest Neighbor algorithm

BARPLOTS:

- A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.
- Below shows the barplots for accuracy, inaccuracy, sensitivity, specificity and precision.

```
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm_Names", x="Accuracy", data=df)
```

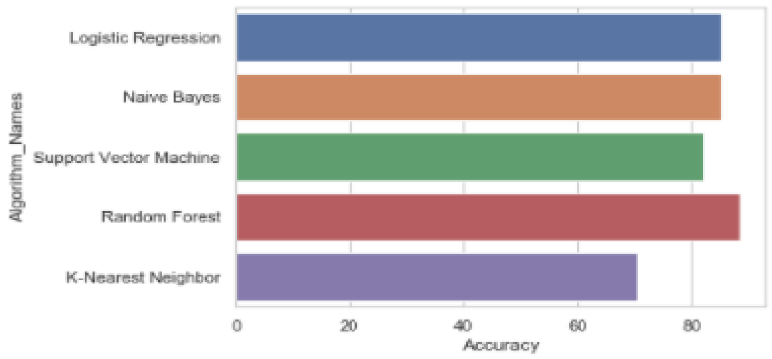


Fig 6.11: Bar plot for accuracy

```
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm_Names", x="Inaccuracy", data=df)
```

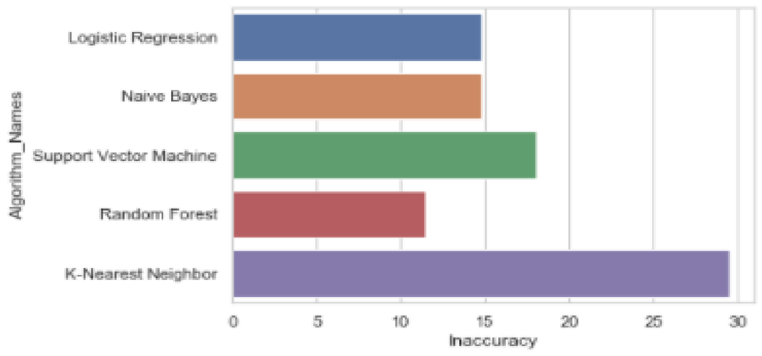


Fig 6.12: Bar plot for Inaccuracy

```
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm_Names", x="Sensitivity", data=df)
```

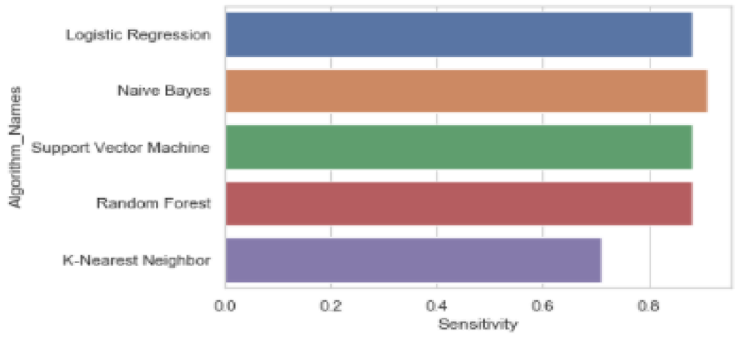


Fig 6.13: Bar plot for Sensitivity

```
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm_Names", x="Specificity", data=df)
```

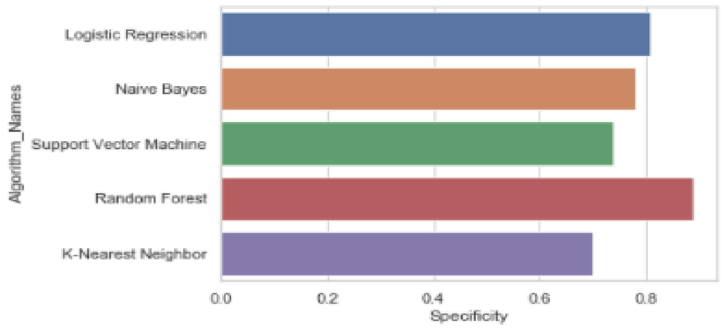


Fig 6.14: Bar plot for Specificity

```
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm_Names", x="Precision", data=df)
```

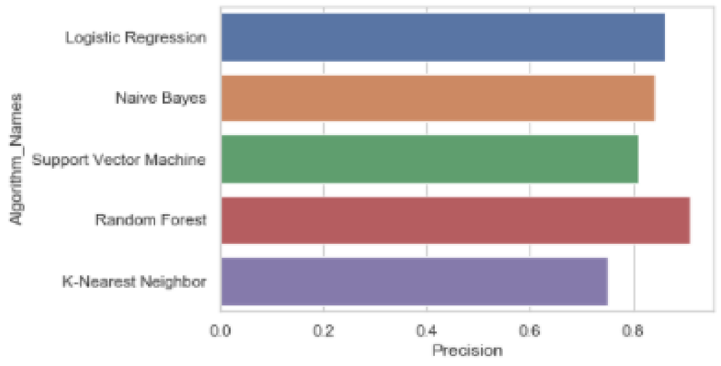


Fig 6.15: Bar plot for Precision

CHAPTER-7

CONCLUSION & FUTURE SCOPE

CONCLUSION

Algorithms are studied and analysed in order to find the most suitable algorithm to predict diseases related to heart. From table 6.5, we observed that the Random Forest algorithm gives the highest accuracy, precision etc., among all other algorithms. So, it can be taken as the best algorithm and can be used for predicting future values.

Now, we are giving all the 13 parameter values to the Random Forest algorithm in both jupyter notebook and mobile application and check that, if the predicted value is 1 or 0.

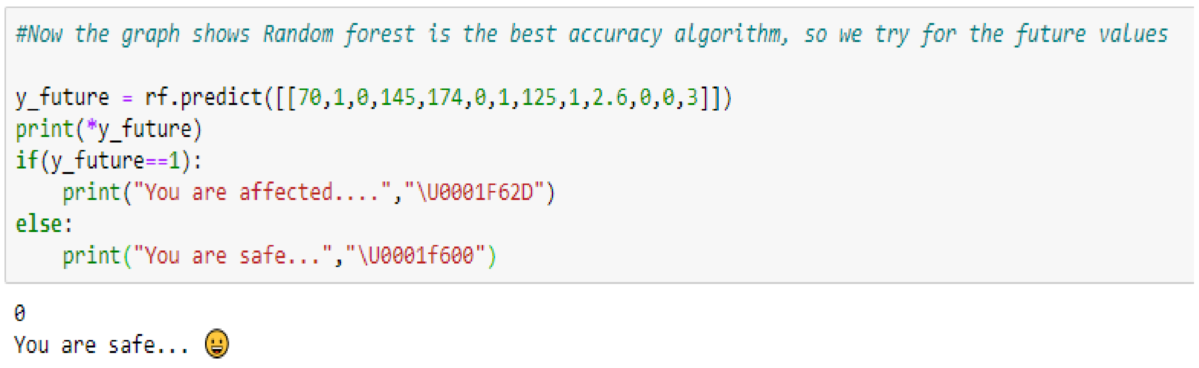
If the predicted value is 1, it shows the output as "You are affected".

If the predicted value is 0, it shows the output as "You are safe".

Example-1

Using Jupiter Notebook.

[70, 1, 0, 145, 174, 0, 1, 125, 1, 2.6, 0, 0, 3]

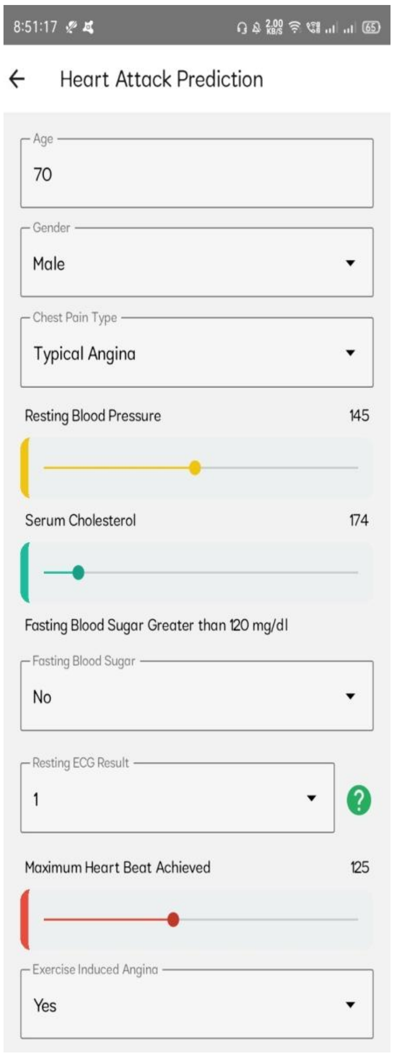


```
#Now the graph shows Random forest is the best accuracy algorithm, so we try for the future values

y_future = rf.predict([[70,1,0,145,174,0,1,125,1,2.6,0,0,3]])
print(*y_future)
if(y_future==1):
    print("You are affected....","\U0001F62D")
else:
    print("You are safe...","\U0001f600")
```

```
0
You are safe... 😀
```

From the fig, it is observed that the predicted value is 0 i.e., "You are safe…"

Using Mobile application:


8:51:17

Heart Attack Prediction

Age: 70

Gender: Male

Chest Pain Type: Typical Angina

Resting Blood Pressure 145

Serum Cholesterol 174

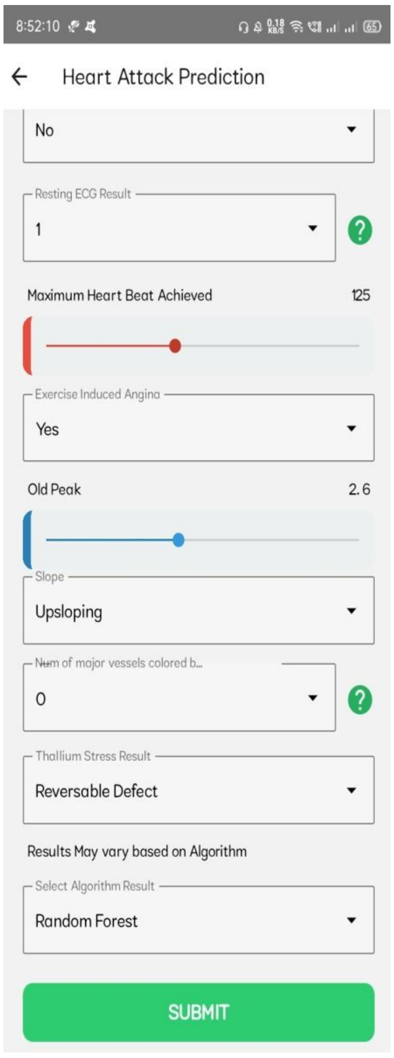
Fasting Blood Sugar Greater than 120 mg/dl

Fasting Blood Sugar: No

Resting ECG Result: 1

Maximum Heart Beat Achieved 125

Exercise Induced Angina: Yes


8:52:10

Heart Attack Prediction

No

Resting ECG Result: 1

Maximum Heart Beat Achieved 125

Exercise Induced Angina: Yes

Old Peak 2.6

Slope: Upsloping

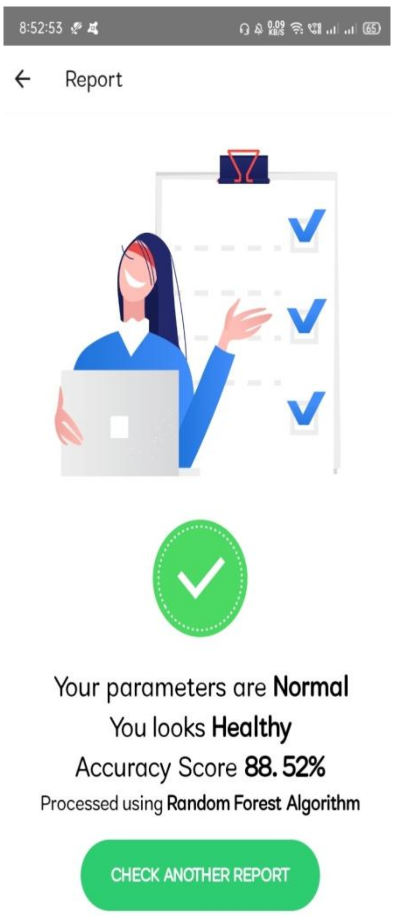
Num of major vessels colored b...: 0

Thallium Stress Result: Reversable Defect

Results May vary based on Algorithm

Select Algorithm Result: Random Forest

SUBMIT


8:52:53

Report

Your parameters are **Normal**

You looks **Healthy**

Accuracy Score **88.52%**

Processed using **Random Forest Algorithm**

CHECK ANOTHER REPORT

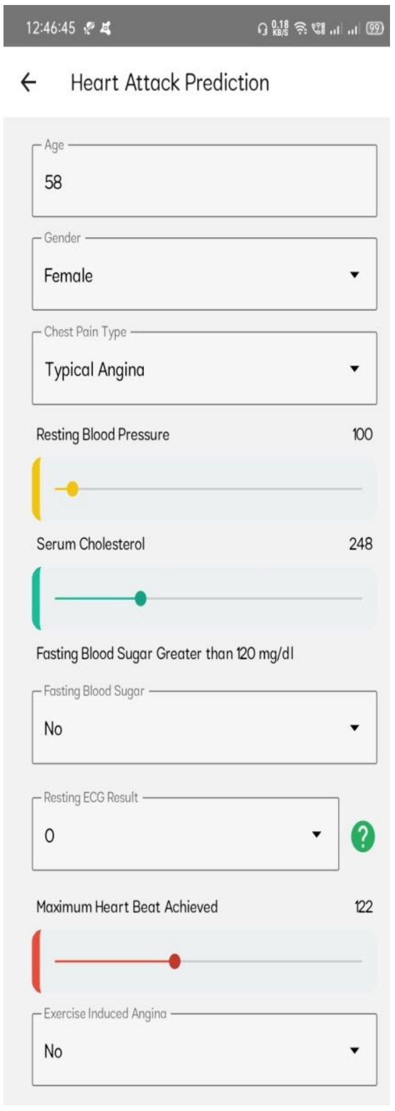
Example-2:

[58,0,0,100,248,0,0,122,0,1.0,1,0,2]

```
y_future = rf.predict([[58,0,0,100,248,0,0,122,0,1.0,1,0,2]])
print(*y_future)
if(y_future==1):
    print("You are affected....","\U0001F62D")
else:
    print("You are safe...","\U0001f600")
```

```
1
You are affected.... 😭
```

From the above fig, it is observed that the predicted value is 1 i.e., “You are affected…”.


Heart Attack Prediction

Age: 58

Gender: Female

Chest Pain Type: Typical Angina

Resting Blood Pressure 100

Serum Cholesterol 248

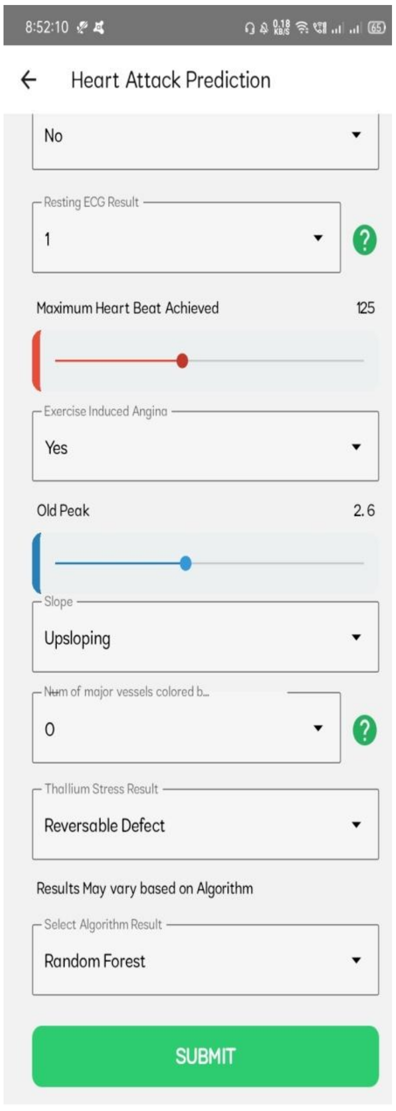
Fasting Blood Sugar Greater than 120 mg/dl

Fasting Blood Sugar: No

Resting ECG Result: 0

Maximum Heart Beat Achieved 122

Exercise Induced Angina: No


Heart Attack Prediction

No

Resting ECG Result: 1

Maximum Heart Beat Achieved 125

Exercise Induced Angina: Yes

Old Peak 2.6

Slope: Upsloping

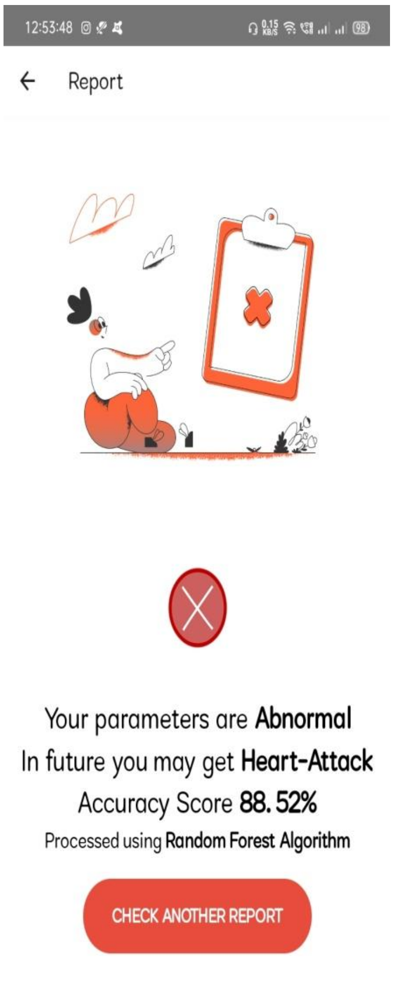
Num of major vessels colored b...: 0

Thallium Stress Result: Reversable Defect

Results May vary based on Algorithm

Select Algorithm Result: Random Forest

SUBMIT


Report

Your parameters are **Abnormal**

In future you may get **Heart-Attack**

Accuracy Score **88.52%**

Processed using **Random Forest Algorithm**

CHECK ANOTHER REPORT

FUTURE SCOPE

There are many possible requirements that can be explored to improve the accuracy of this prediction system. As we have developed a generalized system, in future we can use this system for the analysis of different datasets. The performance of health's diagnosis can be improved significantly by handling numerous class labels in the prediction process. Generally, the dimensionality of heart database is high, so identification and selection of significant attributes for better diagnosis of heart disease are very challenging tasks for future projects

REFERENCES

[1] Gavhane, G. Kokkula, I. Pandya, and K. Devadkar, ``Prediction of heart disease using machine learning'' in Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Mar. 2018, pp. 1275_1278.

[2] Anjan Nikhil Repaka, Sai Deepak Ravikanti, Ramya G Franklin, "Design And Implementing Heart Disease Prediction Using Naives Bayesian." In Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019).

[3] Mamatha Alex P and Shaicy P Shaji, "Prediction and Diagnosis of Heart Disease Patients using Data Mining Technique." International Conference on Communication and Signal Processing, April 4-6, 2019, India

[4] Senthil kumar mohan, Chandrasegar thirumalai, and Gautam srivastava, "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques.", Jun 2019.

[5] N. Komal Kumar, G.Sarika Sindhu, D.Krishna Prashanthi, A.Shaeen Sulthana, "Analysis and Prediction of Cardio Vascular Disease using Machine Learning Classifiers.", 6th International Conference on Advanced Computing & Communication Systems (ICACCS), 2020.

[6] Chaithra, N., & Madhu, B. (2018). Classification models on cardiovascular disease prediction using data mining techniques. Journal of Cardiovascular Diseases and Diagnosis. doi: 10.4172/2329-9517.1000348.

[7] Ashwini Shetty A, and Chandra Naik, May 2016,"Different Data Mining Approaches for Predicting Heart Disease", International Journal of Innovative Research in Science,Engineering and Technology(An ISO 3297: 2007 Certified Organization), Vol. 5, Special Issue 9, pp. 277-281.

[8] Ayatollahi, H., Gholamhosseini, L., & Salehi, M. (2019). Predicting coronary artery disease: a comparison between two data mining algorithms. BMC Public Health. doi: 10.1186/S12889-019-6721-5.

APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv("heart.csv")
predictors = dataset.drop("output",axis=1)
target = dataset["output"]
X_train, X_test, Y_train, Y_test = train_test_split (predictors, target, test_size=0.2,
random_state=0)
X_train.shape
X_test.shape
Y_train.shape
Y_test.shape
```

#LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(X_train,Y_train)
Y_predictor_logreg = logreg.predict(X_test)
print(X_test)
print(Y_predictor_logreg)
cm = confusion_matrix(Y_test, Y_predictor_logreg)
print(cm)
TN=cm[0][0]
FP=cm[0][1]
```

```
FN=cm[1][0]
TP=cm[1][1]
print("For logistic regression")
Acc=round(((TP+TN)/(TP+TN+FP+FN))*100,2)
print("Correct Prediction Accuracy: "+str(Acc)+" %")
Inacc=round(((FP+FN)/(TP+TN+FP+FN))*100,2)
print("Incorrect Prediction Accuracy: "+str(Inacc)+" %")
sen=TP/(TP+FN)
print("Sensitivity:",round(sen,2))
spe = TN/(TN+FP)
print("Specificity:",round(spe,2))
prec = TP/(TP+FP)
print("Precision:",round(prec,2))
sns.distplot((Y_test - Y_predictor_logreg),bins=50)
```

#NAIVE BAYES

```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(X_train,Y_train)
Y_predictor_nb = NB.predict(X_test)
print(X_test)
print(Y_predictor_nb)
cm1 = confusion_matrix(Y_test, Y_predictor_nb)
print(cm1)
TN = cm1[0][0]
FP = cm1[0][1]
FN = cm1[1][0]
TP = cm1[1][1]
print("For Naive Bayes")
Acc = round(((TP+TN)/(TP+TN+FP+FN))*100,2)
print("Correct Prediction Accuracy: "+str(Acc)+" %")
Inacc = round(((FP+FN)/(TP+TN+FP+FN))*100,2)
```

```
print("Incorrect Prediction Accuracy: "+str(Inacc)+" %")
sen = TP/(TP+FN)
print("Sensitivity:",round(sen,2))
spe = TN/(TN+FP)
print("Specificity:",round(spe,2))
prec = TP/(TP+FP)
print("Precision:",round(prec,2))
sns.distplot((Y_test - Y_predictor_nb),bins=50)

#SUPPORT VECTOR MACHINE
from sklearn import svm
sv = svm.SVC(kernel = 'linear')
sv.fit(X_train, Y_train)
Y_pred_svm = sv.predict(X_test)
print(X_test)
print(Y_pred_svm)
cm2 = confusion_matrix(Y_test, Y_pred_svm)
print(cm2)
TN=cm2[0][0]
FP=cm2[0][1]
FN=cm2[1][0]
TP=cm2[1][1]
print("For Support Vector Machine")
Acc=round(((TP+TN)/(TP+TN+FP+FN))*100,2)
print("Correct Prediction Accuracy: "+str(Acc)+" %")
Inacc=round(((FP+FN)/(TP+TN+FP+FN))*100,2)
print("Incorrect Prediction Accuracy: "+str(Inacc)+" %")
sen=TP/(TP+FN)
print("Sensitivity:",round(sen,2))
spe=TN/(TN+FP)
print("Specificity:",round(spe,2))
prec=TP/(TP+FP)
```

```
print("Precision:",round(prec,2))
sns.distplot((Y_test-Y_pred_svm),bins=50)
```

#RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(10):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x
rf = RandomForestClassifier(random_state = best_x)
rf.fit(X_train,Y_train)
Y_pred_rf = rf.predict(X_test)
print(X_test)
print(Y_pred_rf)
cm3= confusion_matrix(Y_test, Y_pred_rf)
print(cm3)
TN=cm3[0][0]
FP=cm3[0][1]
FN=cm3[1][0]
TP=cm3[1][1]
print("For Random Forest algorithm")
Acc=round(((TP+TN)/(TP+TN+FP+FN))*100,2)
print("Correct Prediction Accuracy: "+str(Acc)+" %")
Inacc=round(((FP+FN)/(TP+TN+FP+FN))*100,2)
print("Incorrect Prediction Accuracy: "+str(Inacc)+" %")
sen=TP/(TP+FN)
print("Sensitivity:",round(sen,2))
```

```
spe=TN/(TN+FP)
print("Specificity:",round(spe,2))
prec=TP/(TP+FP)
print("Precision:",round(prec,2))
sns.distplot((Y_test-Y_pred_rf))
```

#K-NEAREST NEIGHBOR

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=21)
knn.fit(X_train,Y_train)
Y_pred_knn = knn.predict(X_test)
print(X_test)
print(Y_pred_knn)
cm4= confusion_matrix(Y_test, Y_pred_knn)
print(cm4)
TN=cm4[0][0]
FP=cm4[0][1]
FN=cm4[1][0]
TP=cm4[1][1]
print("For K-Nearest Neighbor")
Acc=round(((TP+TN)/(TP+TN+FP+FN))*100,2)
print("Correct Prediction Accuracy: "+str(Acc)+" %")
Inacc=round(((FP+FN)/(TP+TN+FP+FN))*100,2)
print("Incorrect Prediction Accuracy: "+str(Inacc)+" %")
sen=TP/(TP+FN)
print("Sensitivity:",round(sen,2))
spe=TN/(TN+FP)
print("Specificity:",round(spe,2))
prec=TP/(TP+FP)
print("Precision:",round(prec,2))
sns.distplot((Y_test-Y_pred_knn))
```

#FUTURE VALUES

```
y_future = rf.predict([[70,1,0,145,174,0,1,125,1,2.6,0,0,3]])
print(*y_future)
if(y_future==1):
    print("You are affected....","\U0001F62D")
else:
    print("You are safe...","\U0001f600")
```

STUDENT DETAILS

1.	Roll number	: 17A91A04C4
	Name of the Student	: Bocha. Dileep Venkata Prasad
	Phone Number	: 8142804595
	Mail id	: 17A91A04C4@aec.edu.in
2.	Roll number	: 17A91A04D9
	Name of the Student	: Pendurthi. Apuroop Sri Durgesh
	Phone Number	: 7382000738
	Mail id	: 17A91A04D9@aec.edu.in
3.	Roll number	: 17A91A04G8
	Name of the Student	: Sirigineedi. Sridevi Naga Kalanjali
	Phone Number	: 9705191398
	Mail id	: 17A91A04G8@aec.edu.in
4.	Roll number	: 18A95A0438
	Name of the Student	: Vasamsetti. Sri Sai Kamal
	Phone Number	: 9063611633
	Mail id	: 18A95A0438@aec.edu.in
5.	Roll number	: 18A95A0427
	Name of the Student	: Beeraka. Manikanta
	Phone Number	: 9177615501
	Mail id	: 18A95A0427@aec.edu.in