A PROJECT REPORT

On

HEART DISEASE PREDICTION USING MACHINE LEARNING

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

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

BY

19WH1A0564	**P. NAGA MANOOJA**
19WH1A0592	**SHAIK AYESHA AMREEN**
19WH1A0575	**M.ANUSHA**

Under the esteemed guidance of

Dr. Reya Sharma

Assistant Professor

Department of Computer Science & Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited EEE, ECE, CSE, IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090.

June, 2023

DECLARATION

We hereby declare that the work presented in this project entitled **"Heart Disease Prediction using Machine Learning"** submitted towards completion of Project work in IV Year of B.Tech of CSE at 'BVRIT HYDERABAD College of Engineering for Women'**,** Hyderabad is an authentic record of our original work carried out under the guidance of Dr. Reya Sharma, Assistant Professor, Department of CSE.

P. Naga Manooja
(19WH1A0564)

Shaik Ayesha Amreen
(19WH1A0592)

M. Anusha
(19WH1A0575)

BVRIT HYDERABAD
College of Engineering for Women

(NBA Accredited EEE, ECE, CSE, IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090.

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the major project entitled "**Heart Disease Prediction using Machine Learning**" is a bonafide work carried out by Ms. P. Naga Manooja(19WH1A0564), Ms. Shaik Ayesha Amreen(19WH1A0592), Ms. M. Anusha(19WH1A0575), in partial fulfillment for the award of B.Tech degree in **Computer Science & Engineering** , **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. E. Venkateswara Reddy
Professor and HoD,
Department of CSE

Guide
Dr. Reya Sharma
Assistant Professor, CSE

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K V N Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women,** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. E. Venkateswara Reddy, Professor** & **Head,** Department of CSE, **BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Dr. Reya Sharma, Assistant Professor,** Department of CSE, **BVRIT HYDERABAD College of Engineering for Women,** for her constant guidance and encouragement throughout the project.

Finally, we would like to thank our Major Project Coordinator, all Faculty and Staff of **CSE** department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

P. Naga Manooja
(19WH1A0564)

Shaik Ayesha Amreen
(19WH1A0592)

M. Anusha
(19WH1A0575)

LIST OF CONTENTS

S.No	Topic	Page No.
	Abstract	i
	List of Figures	ii
1	INTRODUCTION	1
	1.1 Problem Statement	1
	1.2 Motivation	2
	1.3 Objectives	2
	1.4 Methodology	3
	1.4.1 Dataset	3
	1.5 Proposed System	4
2	LITERATURE REVIEW	5
3	REQUIREMENTS	8
	3.1 Software Requirements	8
	3.2 Hardware Requirements	8
	3.3 Technologies Description	8
4	DESIGN	11
	4.1 Introduction	11
	4.2 Architecture	13
5	SYSTEM ARCHITECTURE	14
	5.1 Supervised Learning	14
	5.2 Types of Supervised ML Techniques	15
	5.2.1 Regression	15
	5.2.2 Classification	15

	5.3 Supervised Algorithms used in this model	16
	5.3.1 Naïve Bayes	16
	5.3.2 Random Forest	18
	5.3.3 XGBoost	20
6	IMPLEMENTATION	22
	6.1 Coding	22
	6.2 Dataset Screenshot	22
	6.3 Evaluation Metrics	22
	6.4 ROC Curves	24
	6.4.1 Random Forest ROC Curve	24
	6.4.2 Naïve Bayes ROC Curve	25
	6.4.3 XGBoost ROC Curve	25
	6.5 Accuracy Comparison	26
	6.6 Evaluation Metrics Comparison	26
7	RESULTS	27
8	CONCLUSION & FUTURE SCOPE	28
9	REFERENCES	29
10	APPENDIX	30

ABSTRACT

In recent times, Heart Disease prediction is one of the most complicated tasks in medical field. In the modern era, approximately one person dies per minute due to heart disease. Machine Learning is used across many ranges around the world. The healthcare industry is no exclusion. Machine Learning can play an essential role in predicting presence/absence of locomotors disorders, Heart diseases and more. Such information, if predicted well in advance, can provide important intuitions to doctors who can then adapt their diagnosis and dealing per patient basis. We work on predicting Heart Disease in people using Machine Learning algorithms.

The proposed work predicts the chances of Heart Disease by implementing different machine learning techniques such as XGBoost, Random Forest and Naïve Bayes. Thus, our project presents a comparative study by analysing the performance of different machine learning algorithms and chooses the algorithm that gives better accuracy for prediction.

LIST OF FIGURES

S.No	Description	Page. No
1	Dataset Description	4
2	Architecture	13
3	Supervised Learning	14
4	Classification Vs Regression	15
5	Random Forest	18
6	Sample Dataset	22
7	Confusion Matrix	23
8	Random Forest ROC Curve	24
9	Naïve Bayes ROC Curve	25
10	XGBoost ROC Curve	25
11	Comparison of Algorithms	26
12	Table of Comparison	26
13	Heart Disease Prediction	27

1.INTRODUCTION

According to the World Health Organization, every year 12 million deaths occur worldwide the due to Heart Disease. Heart disease is one of the biggest causes of morbidity and mortality among population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications.

Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the health care industry. This project aims to predict future Heart Disease by analyzing data of patients which classifies whether they have heart disease or not using machine-learning algorithm. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources & we can do the prediction of heart disease.

1.1 Problem statement:

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either it are expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. Since we have a good amount of data in today's world, we can use various machine learning algorithms to predict heart disease at early stage.

1.2 Motivation:

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards identifying the best algorithm for identifying the possibility of heart disease in a patient. Various attributes are taken into consideration and using ml algorithms prediction is done. This will provide researchers and medical practitioners to establish a better health environment.

1.3 Objective:

This project predicts people with cardiovascular disease by extracting the patient history that leads to a fatal heart disease from a dataset that includes patient's medical history such as chest pain, sugar level, blood pressure, etc.

1.4 Methodology:

To predict heart disease large collection of patients medical history is required. Various methodologies used are Random Forest, Naïve Bayes and XGBoost. In System Architecture, the description of these algorithms is explained in detailed manner.

1.4.1 Dataset:

The collected dataset is available on Kaggle. It consists of several medical analyst variables and one target variable. The dataset consists of several independent variables and one dependent variable, i.e., the outcome. This dataset contains 13 medical attributes of 1025 patients that helps us detecting if the patient is at risk of getting a heart disease or not and it helps us classify patients that are at risk of having a heart disease and that who are not at risk. This dataset gives us the much-needed information i.e., the medical attributes such as age, resting blood pressure, fasting sugar level etc. of the patient that helps us in detecting the patient that is diagnosed heart disease or not.

S.No	Observation	Description	Values
1.	Age	Age in years	Continuous
2.	Sex	Sex of subject	Male/Female
3.	CP	Chest pain	Four types
4.	Trestbps	Resting blood pressure	Continuous
5.	Chol	Cholesterol	Continuous
6.	FBS	Fasting blood sugar	<,or> 120mg/dl
7.	Restecg	Resting Electrocardiograph	Three values(0,1,2)
8.	Thalach	Maximum heart rate achieved	Continuous
9.	Exang	Exercise Induced Angina	Yes/No
10.	Oldpeak	ST Depression	Continuous
11.	Slope	Slope of peak exercise ST segment	Up/Flat/Down
12.	Ca	No. of major vessels	0-3
13.	Thal	Blood disorder	Reversible/Fixed/Normal
14.	Target	Heart disease	Present-1/Not present-0

Fig 1.4.1 Dataset Description

1.5 Proposed system:

We have a system that predicts heart disease. The working of the system starts with the collection of data. Then the required data is preprocessed into the required format. The data is then divided into two parts training and testing data. The algorithms are applied and the model is trained using the training data. Using various machine learning algorithms(Random Forest, Naïve Bayes, XGBoost) the accuracy of the system is obtained by testing the system using the testing data. To improve the performance of the model, various evaluation metrics like confusion matrix, precision, recall and F1 score are used. The proposed framework will consider the high accuracy algorithm for predicting heart disease.

2. LITERATURE REVIEW

2.1 Title : Machine learning based heart disease prediction system.

Authors : M. Snehith Raja, M. Anurag, Ch. Prachetan Reddy, Nageshwara Rao Sirisala.

Summary : The datasets are been processed in python using ML Algorithm i.e., Random Forest Algorithm. This technique uses the past old patient records for getting prediction of new one at early stages preventing the loss of lives. In this work, reliable heart disease prediction system is implemented using strong Machine Learning algorithm which is the Random Forest algorithm which read patient record data set in the form of CSV file. After accessing dataset the operation is performed and effective heart attack level is produced.

2.2 Title : Heart disease prediction using machine learning algorithms.

Authors : Archana Singh, Rakesh Kumar

Summary : So in this paper they used the biological parameter as testing data such as cholesterol, Blood pressure, sex, age, etc. and on the basis of these, comparison is done in the terms of accuracy of algorithms such as in this project we have used four algorithms which are decision tree, linear regression, k-neighbour, SVM. In this paper, we calculate the accuracy of four different machine learning approaches and on the basis of calculation we conclude that which one is best among them.

2.3 Title : Effective heart disease prediction using machine learning techniques

Authors : Senthil Kumar Mohan, Chandrasegar Thirumalai and Gautam Srivastava.

Summary : The data of various patients is collected from the UCI laboratory to predict heart disease using decision tree algorithm. Several standard performance metrics such as accuracy, precision and error in classification have been considered for the computation of performance of this model. To identify the significant features of heart disease, three performance metrics are used which will help in better understanding the behavior of the various combinations of the feature-selection. ML technique focuses on the best performing model compared to the existing models. They produced high accuracy and less classification error in the prediction of heart disease.

2.4 Title : Prediction of heart disease using machine learning

Authors : Aditi Gavhane, Isha pandya, Gouthami kokkula, Kailas Devadkar.

Summary : In this paper they used the neural network algorithm multi-layer perceptron (MLP) to train test the dataset. Here there will be multiple layers like one for input, second for output and one or more layers are hidden layers between these two input and output layers. Each node in input layer is connected to output nodes through hidden layers. This connection is assigned with some weights. There is another identity input called bias which is with weight b, which added to node to balance the perceptron. The connection between the nodes can be feedforwarded or feedback.

2.5 Title : Machine learning for real-time heart disease prediction.

Authors : Dimitris Bertsimas, Luca Mingardi and Bartolomeo Stellato.

Summary : In this paper, XGBoost algorithm, a leading machine learning method is used. They presented a novel procedure to accurately detect heart diseases in real-time from the analysis of short single-lead ECGs (9-61 seconds). They observed the characteristics of ventricular response and analyze the predictability of the inter-beat timing of the QRS complexes in the ECG to detect irregular patterns in the data. . The model is meant to be used as a fast detector for heart diseases, able to recognize various types of outcomes: Normal, AF, Tachycardia, Bradycardia, Arrhythmia, Other (label to indicate other types of heart anomaly) and Noisy (can't be classified, and needs to be recorded again). If an anomaly is detected, the patient should be visited by a specialist to assess the stage of the disease and possible treatments.

3.REQUIREMENTS

3.1 Software Requirements:

- Windows 10
- Python 3.8
- Google colab

3.2 Hardware Requirements:

- Intel core i5 processor
- RAM 8GB

3.3 Technologies Description:

- **Google Colaboratory:**

  Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. It is a cloud-based tool. Colab is free of charge with limited resources. However, you should not expect that you can store your artificial intelligence or machine learning models indefinitely on Colab's free infrastructure. Google Colaboratory is a cloud-based tool. You can start coding fantastic ML and data science models using a Chrome browser. Colab is free of charge with limited resources. You can access Google Colaboratory from its official website.

- **Libraries:**

1.Numpy:

NumPy is a powerful Python library widely used for scientific computing and data analysis. It provides a multidimensional array object that allows efficient storage and manipulation of large numerical datasets. NumPy's key feature is its ability to perform fast mathematical and logical operations on entire arrays, making it an essential tool for tasks such as linear algebra, statistical analysis, and signal processing. It also offers a range of functions for array manipulation, array indexing, array indexing, and mathematical operations, enabling users to perform complex computations with ease. NumPy's intuitive and efficient syntax, along with its extensive collection of mathematical functions, makes it a fundamental component of the scientific Python ecosystem and an invaluable resource for data scientists, researchers, and developers.

2.Pandas:

Pandas is a widely used Python library that simplifies data manipulation and analysis. It provides easy-to-use data structures, such as Data Frames, which are two-dimensional tabular structures capable of holding diverse data types. With Pandas, users can efficiently load, clean, transform, and analyses structured data from various sources. Its rich functionalities include filtering, sorting, grouping, merging, and reshaping data, along with powerful indexing and slicing capabilities. Overall, Pandas is a versatile tool that streamlines data handling, making it indispensable for data scientists, analysts, and researchers.

3. Matplotlib:

Matplotlib is a widely used Python library for creating high-quality visualizations and plots. It provides a comprehensive set of functions and tools for generating a wide range of plots, charts, histograms, and other visual representations of data. With Matplotlib, users can create static, animated, or interactive visualizations to effectively communicate insights and patterns in data. The library offers a flexible and customizable interface, allowing users to control various aspects of their plots, including axes, labels, colors, and styles. Matplotlib supports multiple plotting styles and can output visualizations in various formats, such as PNG, PDF, SVG, and more. It integrates well with other libraries in the scientific Python ecosystem, making it a valuable tool for data analysis, scientific research, and data visualization tasks. Overall, Matplotlib empowers users to create visually appealing and informative plots, enabling effective data exploration and presentation.

4. Sklearn:

Sklearn, also known as Scikit-learn, is a popular Python library for machine learning tasks. It offers a comprehensive set of tools and algorithms for classification, regression, clustering, and dimensionality reduction. Built on top of NumPy and SciPy, scikit-learn provides a user-friendly API, making it accessible to both beginners and experienced practitioners. With its emphasis on code readability, model interpretability, and reproducibility, scikit-learn simplifies the development of machine learning models. It integrates well with other Python libraries, enabling seamless data manipulation, visualization, and analysis. Overall, scikit-learn is a powerful and versatile library that serves as a go-to resource for various machine learning applications.

4. DESIGN

4.1 Introduction:

Our project can help predict the people who are likely to diagnose with a heart disease by help of their medical history . It recognizes who all are having any symptoms of heart disease such as chest pain or high blood pressure and can help in diagnosing disease with less medical tests and effective treatments, so that they can be cured accordingly.

The system consists of following components:

- **Data Collection:** An Organized Dataset of individuals had been selected Keeping in mind their history of heart problems and in accordance with other medical conditions.
- **Pre-processing:** Data pre-processing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model which can cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values of the dataset. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Pre-processing of data is required for improving the accuracy of the model.
- **Algorithm Selection:** The model predicts whether the person has heart disease or not using a variety of machine learning methods, including Naive Bayes, XGBoost, Random Forest, etc.
- **Model Training:** Training a machine learning model is a process in which a machine learning algorithm is fed with training data from which it can learn. The dataset is divided into training data and testing data. The training dataset is used for prediction model learning and testing

data is used for evaluating the prediction model. For this project, 80% of training data is used and 20% of data is used for testing.

- **Model Evaluation:** On the testing set, all the chosen methods will be assessed for accuracy, precision, recall, and F1 score.
- **Comparative Analysis:** Based on their evaluation measures, the system will compare how well various machine learning algorithms perform in predicting heart disease. It is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

4.2 Architecture:

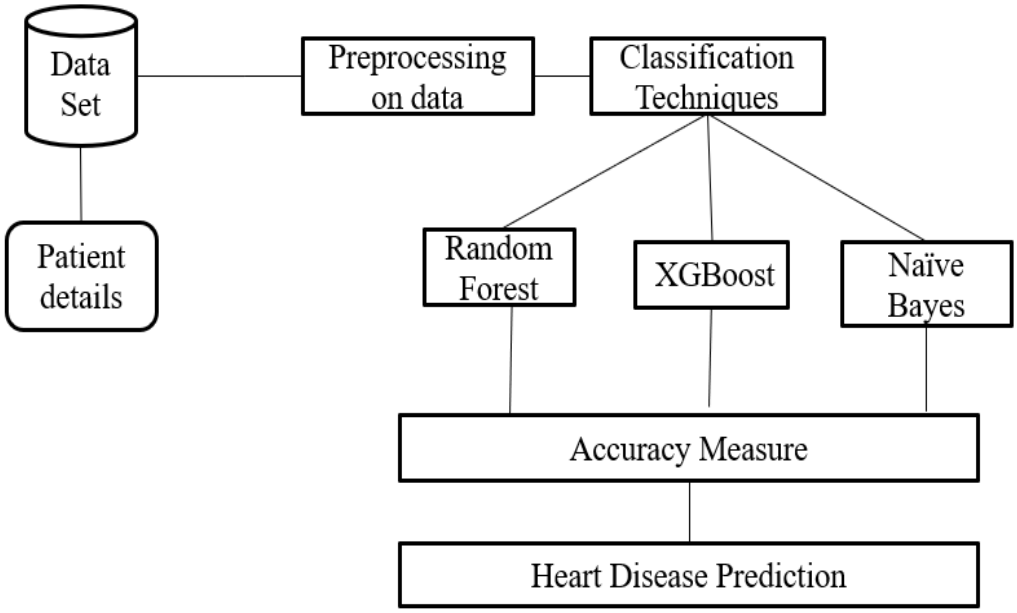

Fig 4.2: Architecture

Dataset collection is collecting data which contains patient details then Preprocessing is done and different machine learning techniques as stated will be applied to predict the accuracy. The highest accuracy algorithm is then used to predict heart disease.

5. SYSTEM ARCHITECTURE

5.1 Supervised Learning:

• The presence of a supervisor who also acts as an instructor is required for supervised learning, as the name suggests. In a nutshell, supervised learning refers to the process of teaching or training a computer using labelled data. This indicates that the right answer has already been assigned to some material.

• In order to analyze the training data (set of training examples) and get an accurate output from labelled data using the supervised learning approach, the machine is then given a fresh set of examples (data).

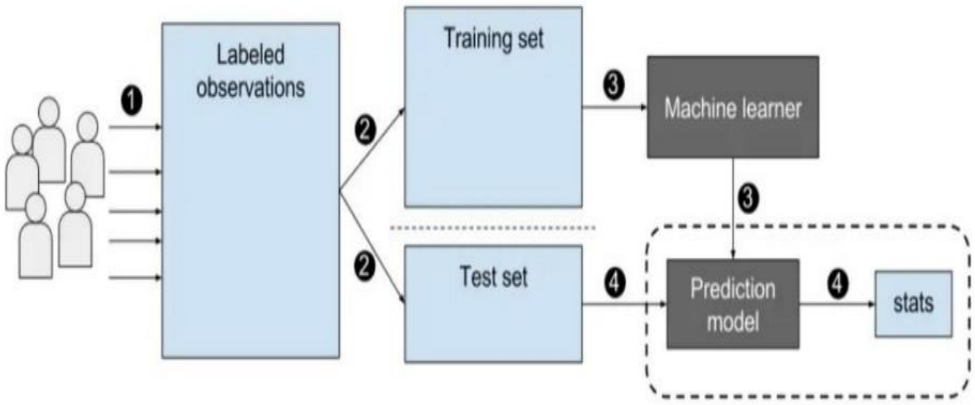

Fig 5.1: Supervised Learning

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function.

5.2 Types of Supervised Machine Learning Techniques

5.2.1 Regression:

• The regression method predicts a single output value using training data.

• For example, you can use regression to predict the price of a house based on training data. Locality, house size, and so on will be input variables.

5.2.2 Classification:

• The process of classifying data into various categories is referred to as "classification." When an algorithm tries to divide data into two groups, it uses binary classification.

• For Example: Determining whether or not someone will fail on a debt.

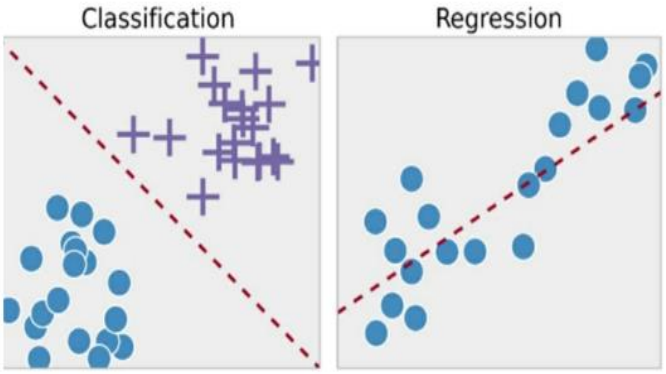

Fig 5.2.2: Classification Vs Regression

5.3 Supervised algorithms used in this model

5.3.1 Naive Bayes:

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles. It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. The Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods. The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

● Naive: It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the basis of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

● Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes theorem: Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$$

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.

5.3.2 Random Forest:

Random Forests are an ensemble(combination) of decision trees. It is a Supervised Learning algorithm that is used for regression and classification. Several decision trees are used to process the input data. When it runs, it builds a variety of decision trees during the training phase and outputs the class that represents the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees.

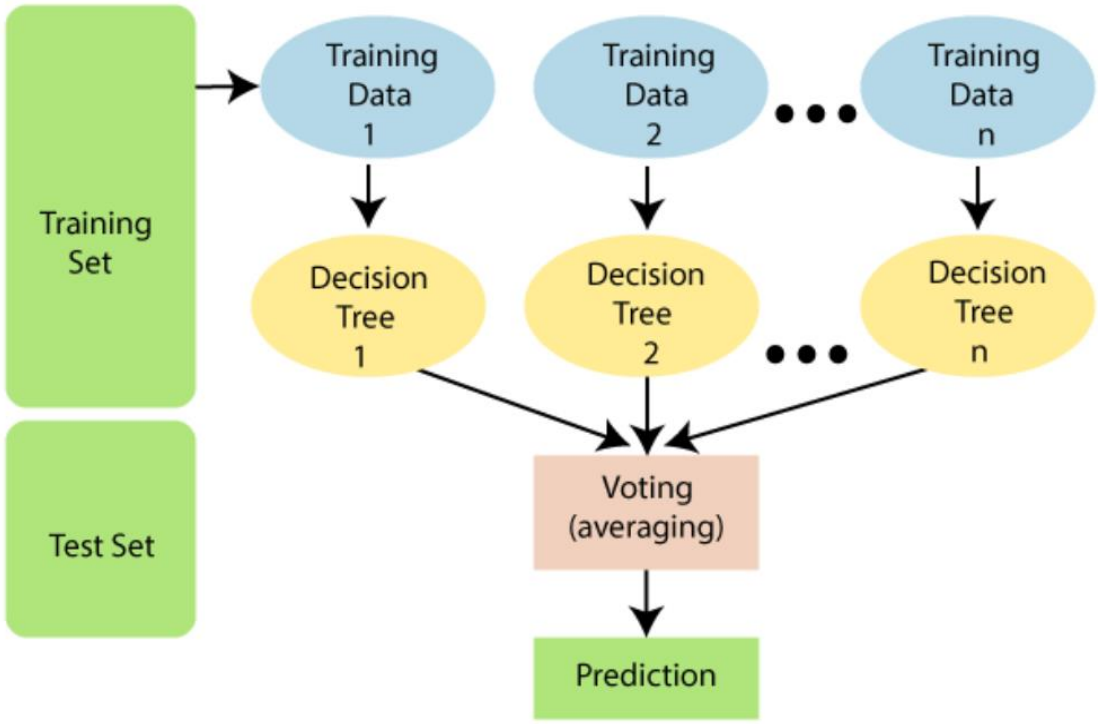

Figure 5.3.2: Random forest

Random Forest is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. The distribution of all trees are the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting.

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. In this tutorial, we will understand the working of random forest and implement random forest on a classification task. Random forests are created from subsets of data, and the final output is based on average or majority ranking; hence the problem of overfitting is taken care of. Random forest randomly selects observations, builds a decision tree, and takes the average result. It doesn't use any set of formulas.

5.3.3 XGBoost:

XGBoost stands for “Extreme Gradient Boosting”. The XGBoost has a tremendous predictive capacity, making it the ideal choice for event accuracy because it incorporates both a linear model and a tree learning method, making the approach about 10 times faster than existing gradient booster techniques.

Various objective functions, including as regression, classification, and ranking, are included in the support. The fact that the XGBoost is also known as a regularized boosting approach is one of the most intriguing aspects of it. This aids in the reduction of overfit modelling. One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. It offers regularization, which allows you to control overfitting by introducing L1/L2 penalties on the weights and biases of each tree. This feature is not available in many other implementations of gradient boosting.

Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. The XGBoost algorithm performs well in machine learning competitions because of its robust handling of a variety of data types, relationships, distributions, and the variety of hyperparameters that you can fine-tune. You can use XGBoost for regression, classification (binary and multiclass), and ranking problems.

6. IMPLEMENTATION

6.1 Coding:

Git link: https://github.com/manuja-reddy/Major-Project

6.2 Dataset Screenshot:

The dataset is taken from Kaggle which consists of 13 attributes and 2 classes Yes - 1, No - 0 with Train size - 820 and Test size - 205.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Fig 6.2 Sample Dataset

6.3 Evaluation Metrics:

- **Confusion Matrix:**

  The confusion matrix is a matrix used to determine the performance of the classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

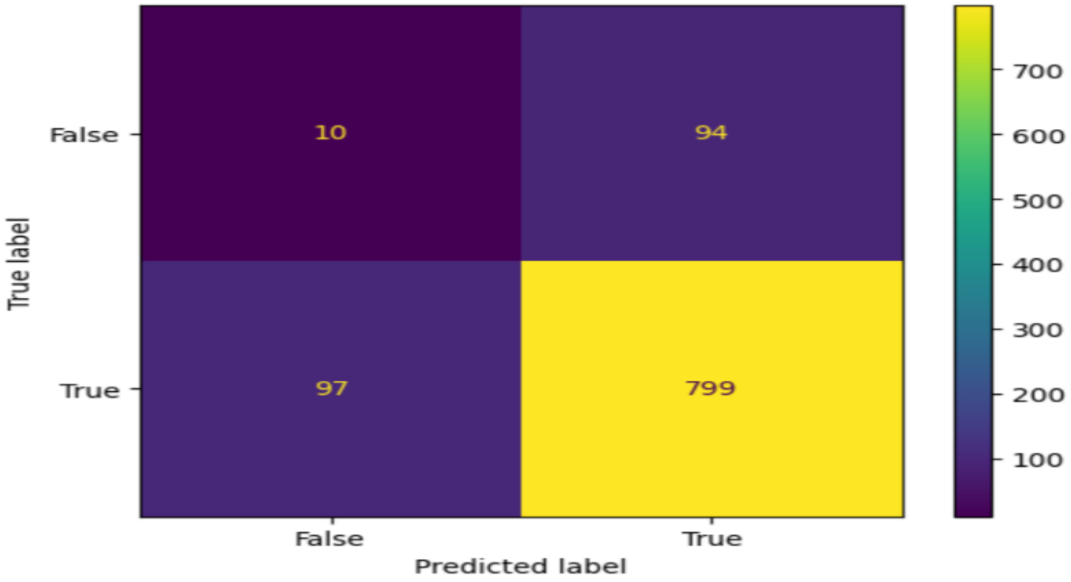

Fig 6.3 Confusion Matrix

- **Accuracy:**

  It is one of the classification metric and it can be used to determine the number of correct predictions to the total number of predictions.

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

- **Recall:**

  It corresponds to the ratio of the number of correctly classified positive items to the number of actual positive items.

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

- **Precision:**

  It refers to the ratio of true positive and the total positives predicted.

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

- **F1 Score:**

  It relates precision and recall metrics to obtain a quality measure that balances the relative importance of these two metrics.

$$\text{F1 SCORE} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}$$

- **6.4 ROC Curves:**

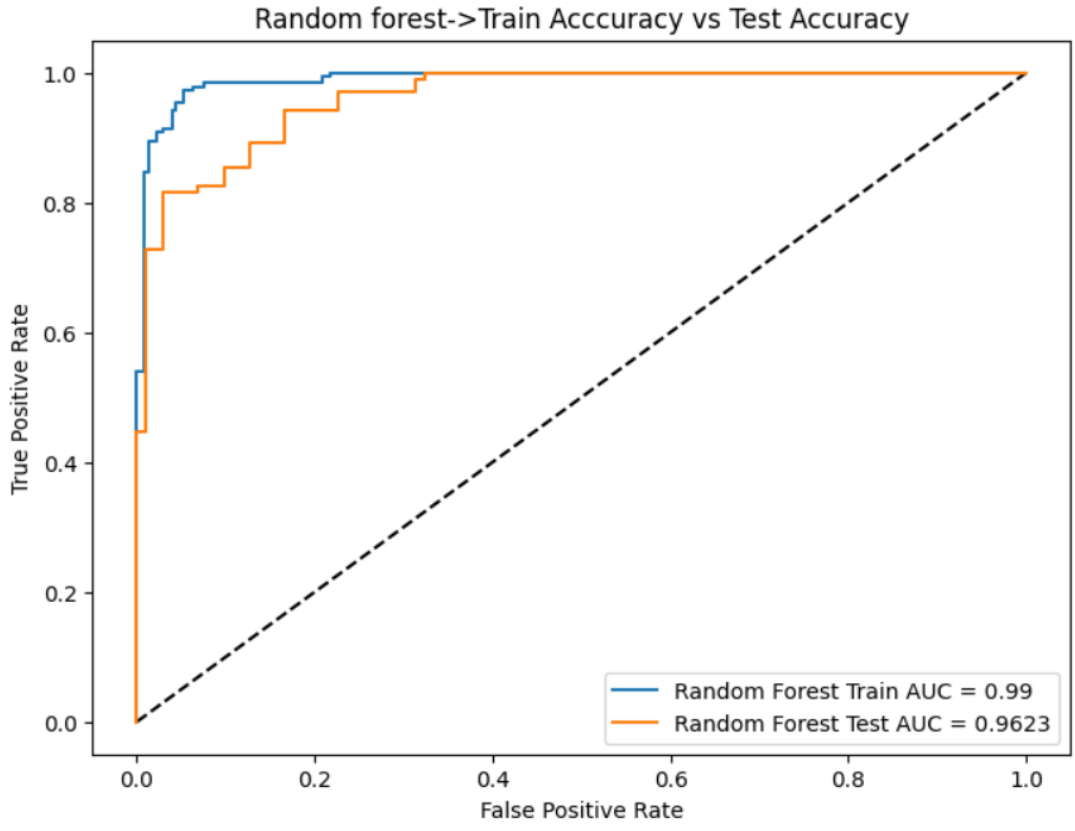

Fig 6.4.1 Random Forest ROC Curve

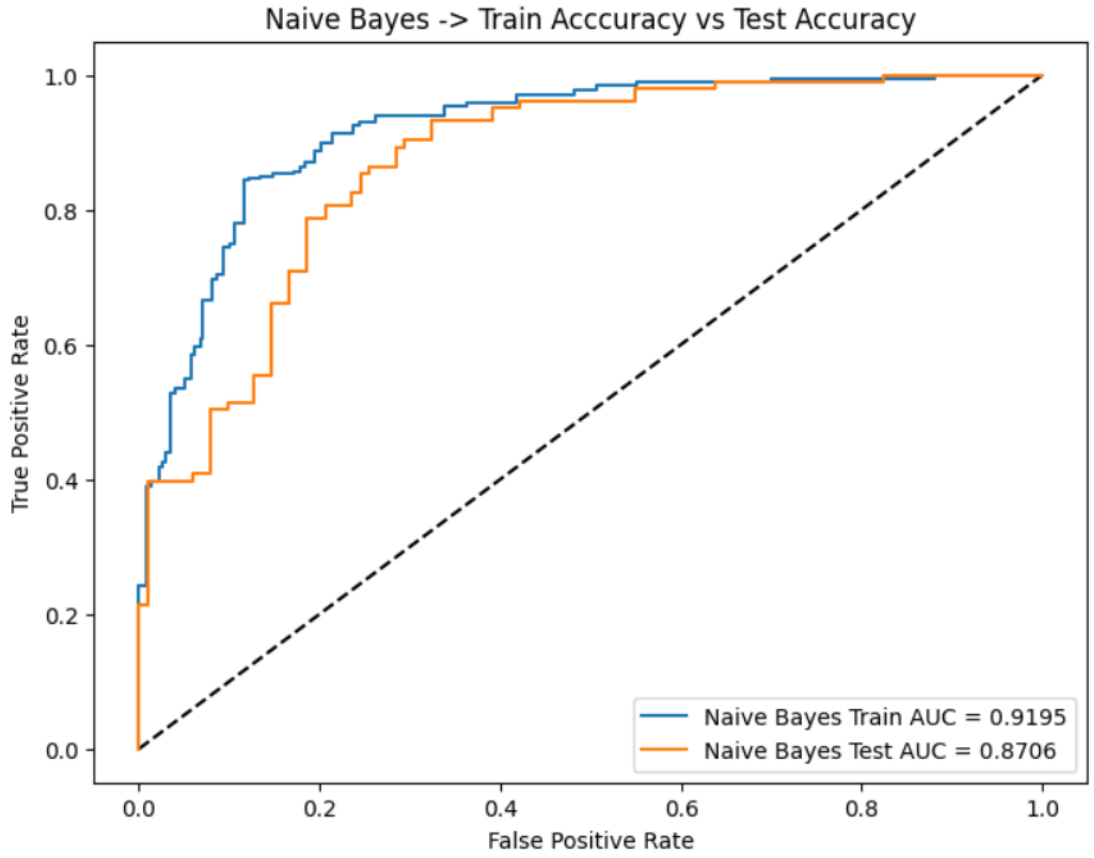

Fig 6.4.2 Naïve Bayes ROC Curve

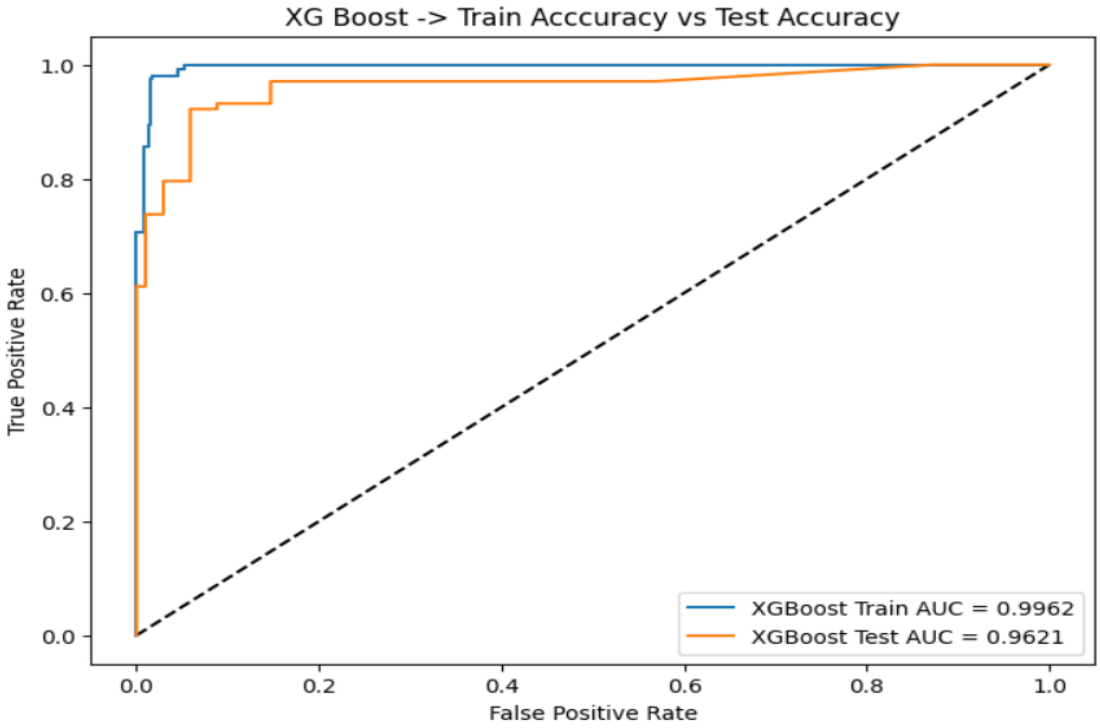

Fig 6.4.3 XGBoost ROC Curve

- **6.5 Accuracy Comparison:**

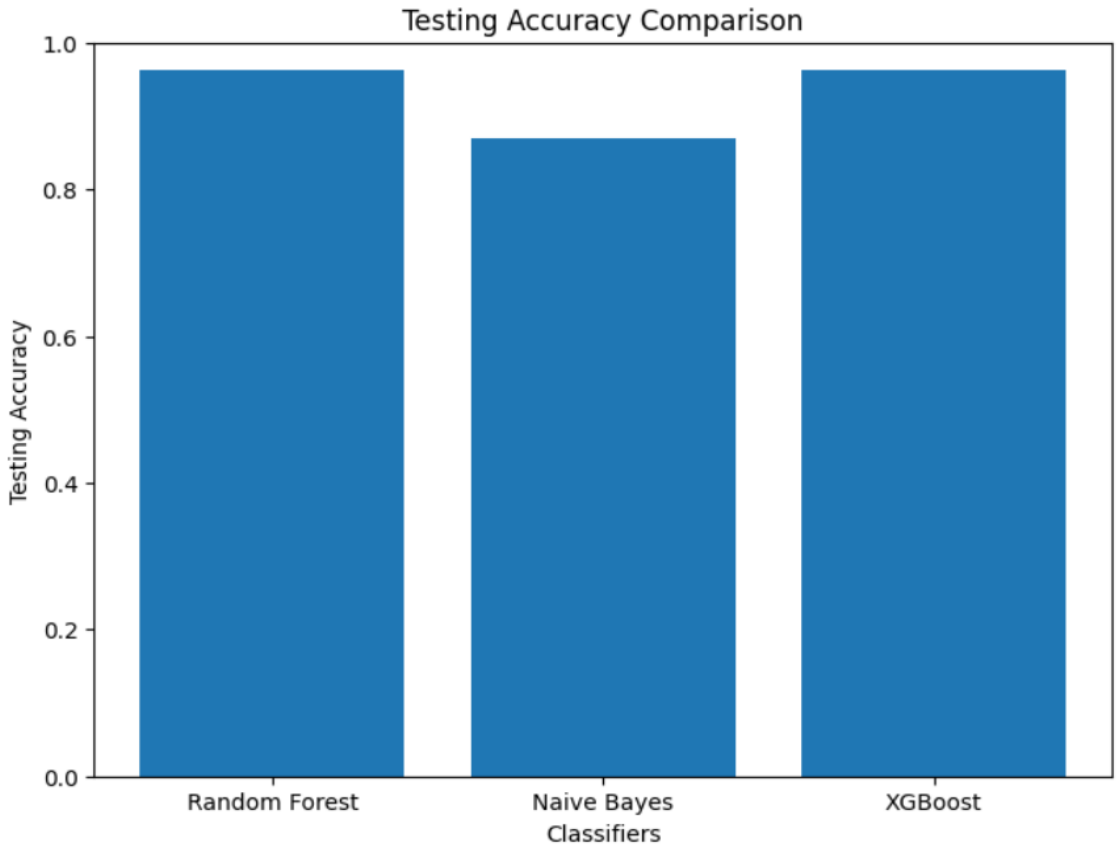

Fig 6.5 Comparison of Algorithms

6.6 Evaluation Metrics Comparison:

S.No	Algorithm	Accuracy(%)	Precision	Recall	F1 Score
1	Random Forest	96	0.88	0.87	0.87
2	XGBoost	96	0.90	0.90	0.90
3	Naïve Bayes	87	0.81	0.80	0.80

Fig 6.6 Table of Comparison

7. RESULTS

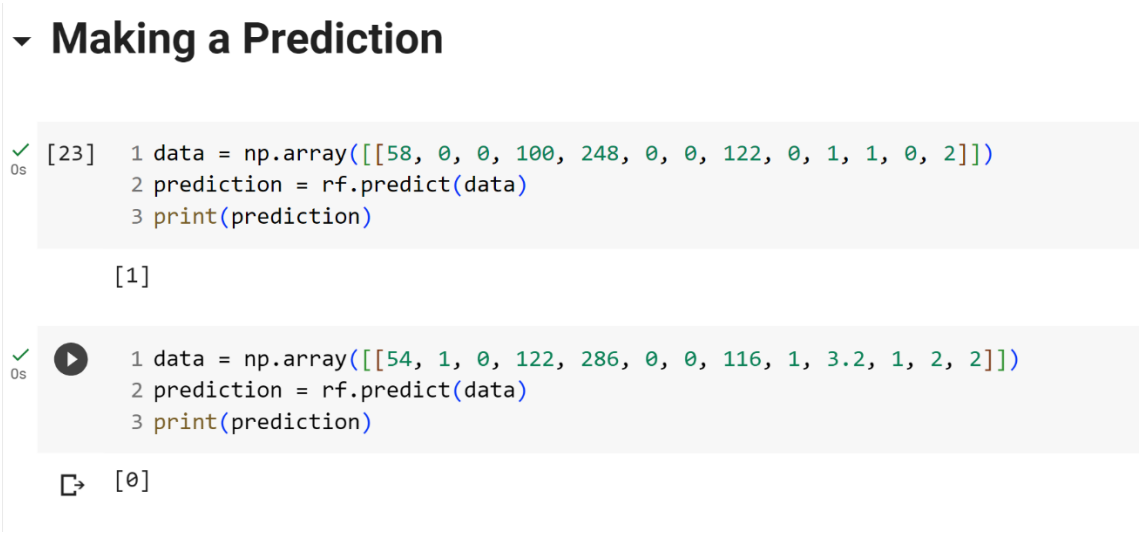

Fig 7.1: Heart Disease Prediction

The above figure shows the prediction of heart disease in which it takes the array of input details which contains patient's medical history. If it predicts output as 1 then the person is having heart disease and if it predicts output as 0 then the person is not having heart disease.

8. CONCLUSION AND FUTURE SCOPE

- Heart disease prediction is a major challenge in the present modern life. This project aims to predict the disease on the basis of the symptoms.

- With this application if the patient/user is away from reach of doctor, they can make use of the application in prediction of disease just by entering the values.

- For the future scope, Deep Learning approaches can be used for better analysis of heart disease and for earlier prediction of diseases so that the rate of death cases can be minimized.

9. REFERENCES

[1] M. Snehith Raja, M. Anurag, Ch.Prachetan Reddy, NageswaraRao Sirisala, “Machine Learning Based Heart Disease Prediction System”, 2021.

[2] Sivaranjan S,Ananya S, Aravinth J, Karthika R, “Diabetes Prediction using Machine Learning Algorithms with Feature Selection and Dimensionality Reduction”, 2021.

[3] Arwatki Chen Lyngdoh, Soumen Moulik, “Diabetes Disease Prediction using Machine Learning Algorithms”,2021.

[4] Archana Singh, Rakesh Kumar, “Heart Disease Prediction using Machine Learning Algorithms”, 2020.

[5] Senthil Kumar Mohan, Chandrasegar and Gautam Srivastava, “Effective Heart Disease Prediction using Machine Learning Techniques”, 2019.

[6] Dimitris Bertsimas, Luca Mingardi, Bartolomeo Stellato, “Machine

Learning for Real-Time Heart Disease Prediction”, 2019.

[7] Aditi Gavhane, Gouthami Kokkula, Isha Pandya, “Prediction of Heart using Machine Learning”, 2018.

[8] Santhana Krishnan J and Geetha S, “Prediction of Heart Disease using Machine Learning Algorithms", 2019.

[9] Mai Shouman, Tim Turner, Rob Stocker, “Using Decision Tree for Diagnosing Heart Disease Patients”,2018.

[10] V. Ramalingam, V Dandapath, Ayantan, Karthik Raja M, “Heart disease prediction using machine learning techniques”, 2018.

[11] M.A. Jabbar, B.L.Deekshatulu, and Priti Chandra, “Intelligent heart disease prediction system using random forest”,2016.

10. APPENDIX

#MOUNT THE GOOGLE DRIVE TO GOOGLE COLAB

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

#IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb
from sklearn.metrics import roc_curve, auc
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score
import warnings
warnings.filterwarnings('ignore')
```

#LOADING THE DATASET

```
df = pd.read_csv('/content/drive/MyDrive/Heart disease/heart.csv')
```

#DATA PREPROCESSING

```
#Shape shows the number of rows and columns present in the dataset
df.shape
```

```
#Head returns the first 5 rows of the dataset

df.head()

#Info method returns some information about dataset

df.info()


# The describe function is used to generate descriptive statistics that
summarize the dataset.
df.describe()


#isnull().sum() returns the number of missing values in the dataset
df.isnull().sum()


#Checks the distribution of target variable
df['target'].value_counts()


#Histogram function is a graphical representation of the frequency distribution
of data.
p = df.hist(figsize = (20,20))
```

#TRAINING AND TESTING

```
#Splitting the data into training and testing data
#80% is training data and 20% is testing data
X_train, X_test, y_train, y_test = train_test_split(df.drop('target', axis=1),
df['target'], test_size=0.2, random_state=42)


#Returns number of rows present in training data
X_train.shape
```

```
#Returns number of rows present in testing data
X_test.shape
```

#CONFUSION MATRIX

```
actual = np.random.binomial(1,.9,size = 1000)
predicted = np.random.binomial(1,.9,size = 1000)
confusion_matrix = metrics.confusion_matrix(actual, predicted)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```

#RANDOM FOREST

```
rf = RandomForestClassifier(random_state=42, max_depth=5)
rf.fit(X_train, y_train)
rf_y_pred = rf.predict(X_test)
rf_train_preds = rf.predict_proba(X_train)[:, 1]
rf_test_preds = rf.predict_proba(X_test)[:, 1]
rf_train_fpr, rf_train_tpr, _ = roc_curve(y_train, rf_train_preds)
rf_test_fpr, rf_test_tpr, _ = roc_curve(y_test, rf_test_preds)
rf_train_auc = round(auc(rf_train_fpr, rf_train_tpr), 4)
rf_test_auc = round(auc(rf_test_fpr, rf_test_tpr), 4)
```

#Plotting ROC Curve

```
plt.figure(figsize=(8, 6))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(rf_train_fpr, rf_train_tpr, label=f'Random Forest Train AUC = {rf_train_auc}')
plt.plot(rf_test_fpr, rf_test_tpr, label=f'Random Forest Test AUC = {rf_test_auc}')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random forest->Train Acccuracy vs Test Accuracy')
plt.legend()
plt.show()
```

#Calculation of Precision, Recall and F1 Score

```
rf_precision = precision_score(y_test, rf_y_pred, average='macro')
rf_recall = recall_score(y_test, rf_y_pred, average='macro')
rf_f1score = f1_score(y_test, rf_y_pred, average='macro')
print("Random Forest Precision: {:.2f}, Recall: {:.2f}, F1-score: {:.2f}".format(rf_precision, rf_recall, rf_f1score))
```

#NAIVE BAYES

```
nb = GaussianNB()
nb.fit(X_train, y_train)
nb_y_pred = nb.predict(X_test)
nb_train_preds = nb.predict_proba(X_train)[:, 1]
nb_test_preds = nb.predict_proba(X_test)[:, 1]
nb_train_fpr, nb_train_tpr, _ = roc_curve(y_train, nb_train_preds)
nb_test_fpr, nb_test_tpr, _ = roc_curve(y_test, nb_test_preds)
nb_train_auc = round(auc(nb_train_fpr, nb_train_tpr), 4)
nb_test_auc = round(auc(nb_test_fpr, nb_test_tpr), 4)
```

#Plotting ROC Curve

```
plt.figure(figsize=(8, 6))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(nb_train_fpr, nb_train_tpr, label=f'Naive Bayes Train AUC = {nb_train_auc}')
plt.plot(nb_test_fpr, nb_test_tpr, label=f'Naive Bayes Test AUC = {nb_test_auc}')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Naive Bayes -> Train Acccuracy vs Test Accuracy')
plt.legend()
plt.show()
```

#Calculation of Precision, Recall and F1 Score

```
nb_precision = precision_score(y_test, nb_y_pred, average='macro')
nb_recall = recall_score(y_test, nb_y_pred, average='macro')
nb_f1score = f1_score(y_test, nb_y_pred, average='macro')
print("Naive Bayes Precision: {:.2f}, Recall: {:.2f}, F1-score: {:.2f}".format(nb_precision, nb_recall, nb_f1score))
```

#XGBoost

```
xgb_model = xgb.XGBClassifier(random_state=42, max_depth=5, learning_rate=0.001, n_estimators=1000, reg_lambda=0.01, reg_alpha=0.01, early_stopping_rounds=10)
xgb_model.fit(X_train, y_train, eval_set=[(X_test, y_test)], verbose=False)
xgb_y_pred = xgb_model.predict(X_test)
xgb_train_preds = xgb_model.predict_proba(X_train)[:, 1]
xgb_test_preds = xgb_model.predict_proba(X_test)[:, 1]
xgb_train_fpr, xgb_train_tpr, _ = roc_curve(y_train, xgb_train_preds)
xgb_test_fpr, xgb_test_tpr, _ = roc_curve(y_test, xgb_test_preds)
xgb_train_auc = round(auc(xgb_train_fpr, xgb_train_tpr), 4)
xgb_test_auc = round(auc(xgb_test_fpr, xgb_test_tpr), 4)
```

#Plotting ROC Curve

```
plt.figure(figsize=(8, 6))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(xgb_train_fpr, xgb_train_tpr, label=f'XGBoost Train AUC = {xgb_train_auc}')
plt.plot(xgb_test_fpr, xgb_test_tpr, label=f'XGBoost Test AUC = {xgb_test_auc}')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('XG Boost -> Train Acccuracy vs Test Accuracy')
plt.legend()
plt.show()
```

#Calculation of Precision, Recall and F1 Score

```
xgb_precision = precision_score(y_test, xgb_y_pred, average='macro')
xgb_recall = recall_score(y_test, xgb_y_pred, average='macro')
xgb_f1score = f1_score(y_test, xgb_y_pred, average='macro')
print("XGBoost Precision: {:.2f}, Recall: {:.2f}, F1-score: {:.2f}".format(xgb_precision, xgb_recall, xgb_f1score))
```

#ACCURACY COMPARISON

```
classifiers = ['Random Forest', 'Naive Bayes', 'XGBoost']
test_accuracy = [rf_test_auc, nb_test_auc, xgb_test_auc]
plt.figure(figsize=(8, 6))
plt.bar(classifiers, test_accuracy)
plt.xlabel('Classifiers')
plt.ylabel('Testing Accuracy')
plt.title('Testing Accuracy Comparison')
plt.ylim(0, 1)
plt.show()
```

#MAKING A PREDICTION

```
data = np.array([[58, 0, 0, 100, 248, 0, 0, 122, 0, 1, 1, 0, 2]])
prediction = rf.predict(data)
print(prediction)
```

```
data = np.array([[54, 1, 0, 122, 286, 0, 0, 116, 1, 3.2, 1, 2, 2]])
prediction = rf.predict(data)
print(prediction)
```