

AN EFFICIENT ASSOCIATION RULE MINING FROM DISTRIBUTED MEDICAL DATABASE FOR PREDICTING HEART DISEASE

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in

INFORMATION TECHNOLOGY

By

**ASWIN KUMAR K (Reg No: 38120015)
JOHNWILIFREDDAVID J (Reg No: 38120034)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

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



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC | 12B Status by UGC | Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **ASWIN KUMAR K (Reg no: 38120015)** and **JOHNWILIFREDDAVID J (Reg no: 38120034)** carried out the project entitled **“AN EFFICIENT ASSOCIATION RULE MINING FROM DISTRIBUTED MEDICAL DATADASE FOR PREDICTING HEART DISEASE** “under our supervision from OCTOBER 2021 to JANUARY 2022

Internal Guide

Dr. S. GOWRI Ph.D.,

Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **ASWIN KUMAR K and JOHNWILIFREDDAVID J** hereby declare that the Project Report entitled done by me under the guidance of **Dr. S. GOWRI Ph.D.**, at Sathyabama Institute of Science and Technology (Deemed to be University), Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

1.

2.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing successfully. We are grateful to them.

We convey my thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing** and **Dr. R. SUBHASHINI M.E., Ph.D., Head of the Department, Department of Information Technology** for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. S. GOWRI Ph.D.**, for her valuable guidance, suggestions and constant Encouragement paved way for the successful completion of our project work.

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

CONTENTS

S.NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF SYMBOLS	iii
	LIST OF TABLES	iv
	LIST OF ABBREVIATIONS	v
1	INTRODUCTION	1
1.1	Introduction	1
1.2	Introduction To Project	2
2	LITERATURE SURVEY	4
3	AIM AND SCOPE	10
3.1	Existing System	10
3.2	Proposed system	10
3.2.1	Collection of dataset	11
3.2.2	Selection of attributes	11
3.2.3	Pre-processing of data	12
3.2.4	Balancing of data	13
3.2.5	Prediction of disease	14
4	EXPERIMENTAL OR MATERIALS METHODS ALGORITHMS	15
4.1	SYSTEM ARCHITECTURE	15
4.2	MACHINE LEARNING	16
4.3	ALGORITHM	17
4.3.1	SUPPORT VECTOR MACHINE	17
4.3.2	NAIVE BAYES	19
4.3.3	DECISION TREE	21
4.3.4	RANDOM FOREST	23
4.3.5	LOGISTIC REGRESSION	25
4.3.6	ADABOOST	27
5	RESULT AND DISCUSSION	30
5.1	SYSTEM CONFIGURATION	30
5.2	SAMPLE CODE	30

5.3	DATASET DETAILS	58
5.4	PERFORMANCE ANALYSIS	61
5.5	PERFORMANCE MEASURE	63
5.6	INPUT AND OUTPUT	64
5.6.1	INPUT	64
5.6.2	OUTPUT	66
5.7	RESULT	68
6	CONCLUSION	69
	REFERENCES	70

ABSTRACT

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 possible Heart Diseases in people using Machine Learning algorithms. In this project we perform the comparative analysis of classifiers like decision tree, Naïve Bayes, Logistic Regression, SVM and Random Forest and we propose an ensemble classifier which perform hybrid classification by taking strong and weak classifiers since it can have multiple number of samples for training and validating the data so we perform the analysis of existing classifier and proposed classifier like Ada-boost and XG-boost which can give the better accuracy and predictive analysis..

Keywords: SVM; Naive Bayes; Decision Tree; Random Forest; Logistic Regression; Adaboost; XG-boost; python programming; confusion matrix; correlation matrix.

LIST OF FIGURES

SI.NO	FIGURE DESCRIPTION	PAGE NO
3.2.1	Collection of dataset	11
3.2.2	Correlation of matrix	12
3.2.3	Preprocessing of data	13
3.2.4	Data balancing	14
3.2.5	Prediction of disease	14
4	System Architecture	15
4.3.1	Support vector machine	19
4.3.5	Logistic Regression	27
4.3.7	Xgboost	29
5.3	Heart Disease data set	58
5.4	Confusion matrix	61
5.4	Correlation Matrix	62
5.5	Accuracies of various algorithm	63
5.6.1	Input	65
5.6.2	Output	67

LIST OF SYMBOLS

TP	Number of people with heart diseases
TN	Number of people with heart diseases and no heart disease
FP	Number of people with no heart diseases
FN	Number of people with no heart diseases and with heart disease.
$F(x)$	Output between the 0 and 1 value.
E	Base of the natural logarithm.
X	Input to the function.

LIST OF TABLES

Table No.	Topic Name	Page No.
1	Attribute Table	59
2	Accuracy Table	68

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
NN	Neural Network
SVM	Support Vector Machine
XG	Extreme Gradient

CHAPTER 1

INTRODUCTION

Machine learning is one of the most disruptive technologies of this generation. It is a part of data science wherein the computer systems are made to learn from the different data sets on the basis of patterns generated from the datasets. It basically enables a machine to learn on its own based on some useful algorithms specifically developed for it. Machine learning is being heavily used in all the sectors including manufacturing, healthcare, Research and Development etc. In this work we are predicting the heart disease occurrence in a patient based on some important characteristics which are best suited based on our data set that we have collected.

The heart is one of the main organs of the human body. It pumps blood through the blood vessels of the circulatory system. The circulatory system is extremely important because it transports blood, oxygen and other materials to the different organs of the body. Heart plays the most crucial role in circulatory system. If the heart does not function properly then it will lead to serious health conditions including death.

Types of Cardiovascular Diseases

Heart diseases or cardiovascular diseases (CVD) are a class of diseases that involve the heart and blood vessels. Cardiovascular disease includes coronary artery diseases (CAD) like angina and myocardial infarction (commonly known as a heart attack). There is another heart disease, called coronary heart disease (CHD), in which a waxy substance called plaque develops inside the coronary arteries. These are the arteries which supply oxygen-rich blood to heart muscle. When plaque begins to build up in these arteries, the condition is called atherosclerosis. The development of plaque occurs over many years. With the passage of time, this plaque can harden or rupture (break open). Hardened plaque eventually narrows the coronary arteries which in turn reduces the flow of oxygen-rich blood to the heart. If this plaque ruptures, a blood clot can form on its surface. A large blood clot can most of the time completely block blood flow through a coronary artery. Over time, the ruptured plaque also hardens and narrows the coronary arteries. If the stopped blood flow isn't restored quickly, the section of heart muscle begins to die. Without quick treatment, a heart attack can lead to serious health problems and even death. Heart attack is a common cause of death worldwide. Some of the common symptoms of heart attack are as follows.

Chest pain

It is the most common symptom of heart attack. If someone has a blocked artery or is having a heart attack, he may feel pain, tightness or pressure in the chest.

Nausea, Indigestion, Heartburn and Stomach Pain

These are some of the often overlooked symptoms of heart attack. Women tend to show these symptoms more than men.

Pain in the Arms

The pain often starts in the chest and then moves towards the arms, especially in the left side.

Feeling Dizzy and Light Headed

Things that lead to the loss of balance.

Fatigue

Simple chores which begin to set a feeling of tiredness should not be ignored.

Prevalence of Cardiovascular Diseases

An estimated 17.5 million deaths occur due to cardiovascular diseases worldwide. More than 75% deaths due to cardiovascular diseases occur in the middle-income and low-income countries. Also, 80% of the deaths that occur due to CVDs are because of stroke and heart attack. India too has a growing number of CVD patients added every year. Currently, the number of heart disease patients in India is more than 30 million. Over two lakh open heart surgeries are performed in India each year. A matter of growing concern is that the number of patients requiring coronary interventions has been rising at 20% to 30% for the past few years.

INTRODUCTION TO PROJECT

The Random Forest algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label. It creates a forest of trees where each tree is formed by a random selection of features from the total features. Here, we can vary the number of trees that will be used to predict the class. We are also implementing the support vector machine to classify the datasets based on the class label. The support vector machine works on kernel and uses hyper plane to group datasets. The logistic regression is also known as sigmoid function which helps in the easy representation in graphs. It also provides high accuracy. In this algorithm first the data should be

imported and then trained. By using equation the logistic regression algorithm the training data estimate the best and approximate coefficient and representation. By implementing these algorithms is used and the health care data which predicts the patient whether they are having heart disease

CHAPTER 2

LITERATURE SURVEY

1. Prediction of Heart Disease Using Machine Learning

Authors: Aditi Gavhane ; Gouthami Kokkula ; Isha Pandya ; Prof. Kailas Devadkar,2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)

With the rampant increase in the heart stroke rates at juvenile ages, we need to put a system in place to be able to detect the symptoms of a heart stroke at an early stage and thus prevent it. It is impractical for a common man to frequently undergo costly tests like the ECG and thus there needs to be a system in place which is handy and at the same time reliable, in predicting the chances of a heart disease. Thus we propose to develop an application which can predict the vulnerability of a heart disease given basic symptoms like age, sex, pulse rate etc. The machine learning algorithm neural networks has proven to be the most accurate and reliable algorithm and hence used in the proposed system.

2. Heart Disease Prediction using Evolutionary Rule Learning

Authors: Akash Chauhan ; Aditya Jain ; Purushottam Sharma ; Vikas Deep, 2018 4th International Conference on Computational Intelligence & Communication Technology (CICT)

In modern society, Heart disease is the noteworthy reason for short life. Large population of people depends on the healthcare system so that they can get accurate result in less time. Large amount of data is produced and collected by the healthcare organization on the daily basis. To get intriguing knowledge, data innovation permits to extract the data through atomization of processes. Weighted Association Rule is a type of data mining technique used to eliminate the manual task which also helps in extracting the data directly from the electronic records. This will help in decreasing the cost of services and also helps in saving lives. In this paper, we will find the rule to predict patient's risk of having coronary disease. Test results have shown that vast majority of the rules helps in the best prediction of coronary illness.

3. Blood Viscosity based Heart Disease Risk Prediction Model in Edge/Fog Computing

Authors: R Latha ; P Vetrivelan, 2019 11th International Conference on Communication Systems & Networks (COMSNETS)

In this paper, heart disease prediction modeled using partially observable Markova decision process (POMDP) is proposed. In emergency, the patient is alerted through the doctor by fog computing. Ambulance sent to the location of patient at critical situations. The doctor gets the data through fog computing iFogSim. Fog computing in healthcare is a new area, which gains more attraction in research community. Many researches focus on cardiovascular disease i.e. heart disease. The important risk factor for cardiovascular disease is increase in blood viscosity. The highly viscous nature of blood does not allow the blood to flow creating a resistance in the blood flow. Heart disease risk factors are high blood pressure, obesity, diabetes, increased blood viscosity, etc. With the help of POMDP's states, observations, beliefs, probability transitions the patient health is noted. The POMDP model for heart disease prediction computes the policy approximation using states and timeslots. Rewards are tabulated using policy approximations over different iterations.

4.An Investigation of Heart Disease Prediction Techniques Authors; G. Shanmugasundaram ; V. Malar Selvam ; R. Saravanan ; S. Balaji, 2018 iee international conference on system, computation, automation and networking (icscan)

Heart disease indicates the type of condition which leads to heart malfunction. Not all people with coronary artery disease have chest pain as a symptom. There are other factors such as R-Blood Pressure, S-Cholesterol, F-Blood Sugar, R-ECG, Thalach, Ex-Ang, Number of major Vessels blocked, Thallium Scan which also results in heart disease. At an initial stage the prediction of heart disease can save human lives. Data mining techniques such as Naive Bayes, Decision Tree, K-nearest neighbor, etc., are used in the Heart Disease prediction based on the parameters / factors. The objective of this research is to investigate about the various factors and its significance in identifying the heart disease. Further it also explores about the existing techniques and models used for prediction. This work concludes with the key challenges which are not yet explored.

5. Newfangled Approach for Early Detection and Prevention of Ischemic Heart Disease using Data Mining

Authors: Dhara B. Mehta ; Nirali C. Varnagar, 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)

The significant advances in biotechnology and health science have led to produce large amount of data. As the heart disease causing the major problem because it is very difficult to identify disease based on symptoms. For that we need a lot of experience and knowledge. Finding the exact cause of disease can help to patient cure at early stage of risk level. Ischemic heart Disease is one of the most common causes of death nowadays. In this research work, we have applied data mining classification techniques on the stroke dataset and proposed a recommendation system in which a person can come to know their risk level of IHD. By taking proper cure and treatment can help to survive more years if they have high risk of IHD. We have used five data mining algorithms Logistic Regression, Decision Tree, K nearest Neighbor, Naïve Bayes and SVM on Ischemic Stroke Dataset and got the highest accuracy 97.91% using Support Vector Machine algorithm.

6. Real-time machine learning for early detection of heart disease using big data approach

Authors: Abderrahmane Ed-Daoudy; Khalil Maalmi, 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)

Over the last few decades, heart disease is the most common cause of global death. So early detection of heart disease and continuous monitoring can reduce the mortality rate. The exponential growth of data from different sources such as wearable sensor devices used in Internet of Things health monitoring, streaming system and others have been generating an enormous amount of data on a continuous basis. The combination of streaming big data analytics and machine learning is a breakthrough technology that can have a significant impact in healthcare field especially early detection of heart disease. This technology can be more powerful and less expensive. To overcome this issue, this paper propose a real-time heart disease prediction system based on apache Spark which stand as a strong large scale distributed computing platform that can be used successfully for streaming data event against machine learning through in-memory computations. The system consists of two main sub parts, namely streaming processing and data storage and visualization. The first uses Spark

MLlib with Spark streaming and applies classification model on data events to predict heart disease. The seconds uses Apache Cassandra for storing the large volume of generated data.

7. Predictive Analysis of Rapid Spread of Heart Disease with Data Mining

Authors: Radhanath Patra; Bonomali Khuntia, 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)

Cardiovascular disease is the leading cause of mortality for both sexes in worldwide. Heart disease is increasing at a rapid rate in both older and younger generation of males and females now days. So in need demand of right strategies, development and implementation of effective health monitoring policies should be emphasized to combat the epidemic of heart related diseases. So early detection and treatment with the use of both conventional and innovative technique must be preferred in this paper we have used the UCI machine learning repository Cleveland heart disease database having 303 instance and 76 attributes. For the proposed method we have used the Information gain concept for selection of best attribute and processes the selected features using weka and python. This paper identifies the gap of research on prediction of heart disease based on python Anaconda navigator, spyder and weka platform on which we have much emphasized. The various techniques, processes which have used to train the model of heart datasets such as feature selection, numpy, pandas library, decision tree classifier, KNN classifier, entropy, gini-index, confusion matrix. The result shows that decision tree classifier is most effective and appropriate for prediction of UCI repository Cleveland heart dataset.

8. Heart Disease Prediction System using CART-C

Authors: Priyal Chotwani; Asmita Tiwari; Vikas Deep; Purushottam Sharma, 2018 International Conference on Computer Communication and Informatics (ICCCI)

Cardiovascular illness is the most widely recognized infection and many individuals are experiencing this malady from an extremely youthful age. It is the enormous purpose behind mortality. Accordingly it is imperative to analyze the issue and treat it on time before it turns out to be deadly. It is elusive the able specialists nowadays and also even a pro doesn't know about all the sub claim to fame so there is the need of

computerized framework which can analyze the issue. A computerized framework will help in indicating how lethal the issue is and how soon it should be dealt with. This will likewise decrease the endeavors of a specialist to assemble all the moment subtle elements of the individual to look for the issue inside and out. Subsequently by diminishing time and exertion of a specialist and giving exact outcomes, this sickness can be dealt with on time and a man's life can be spared. A robotized framework in healthful analysis would upgrade restorative care and it will likewise reduce prices. During this examination, we've got composed a framework that may proficiently notice the tenets to foresee the hazard level of patients seeable of the given parameter regarding their wellbeing. The rules are often organized in lightweight of the client's necessity. The execution of the framework is assessed as way as grouping accuracy and therefore the outcomes demonstrates that the framework has awing potential in foreseeing the coronary hazard level all the additional exactly.

9. Automated Prediction of Heart Disease Patients using Sparse Discriminant Analysis

Authors: K. M. Zubair Hasan ; Shourob Datta ; Md Zahid Hasan ; Nusrat Zahan, 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)

Linear Discriminant Analysis (LDA) is an easy and efficient method for pattern classification, while it is also broadly utilized for the initial discovery of diseases using Electronic Health Records (EHR) data. Nonetheless, the performance of LDA for EHR data classification is recurrently influenced by two major factors: poor evaluation of LDA parameters (e.g., covariance matrix), and “linear inseparability” of the EHR data for classification. In this paper, we propose a novel classifier SDA -Sparse Discriminant Analysis method for heart disease detection. The time complexity will be reduced in this algorithm by optimal scoring analysis of LDA and will be comprehensive to execute sparse discrimination through the combination of Gaussians if limits between classes are nonlinear or if subgroups are available inside every class. On the whole, compared to previous techniques, our proposed technique is more appropriate for the diagnosis of heart disease patients with higher accuracy.

10. Non-Laboratory-Based Risk Factors for Automated Heart Disease Detection

Authors: H. Mai; T. T. Pham; D. N. Nguyen; E. Dutkiewicz, 2018 12th International Symposium on Medical Information and Communication Technology (ISMICT)

Developing a heart disease detection model using simple non-laboratory risk factors plays an important role in preventive care, especially for high risk subjects. The model allows physicians/epidemiologists to effectively diagnose a person as having heart disease. In this work, we aim to develop a non-invasive risk prediction model for automated heart disease detection that involves age, gender, rest blood pressure, maximum heart rate, and rest electrocardiography. We examine four public datasets from 1071 participants who were referred for a special X-ray of the heart's arteries (i.e., to see if they are narrowed or blocked). The subjects also undertook a physical examination and three non-invasive tests. To estimate the heart disease status, we apply a generalized linear model with regularization paths via coordinate descent. Even without laboratory-based data (e.g., serum cholesterol, fasting blood sugar), we observed a prediction accuracy as high as 72%, compared with 76% of other comprehensive models. This observation suggests that few non-invasive factors utilizing recent advances in data analytics can replace the current practices of heart disease risk assessment

CHAPTER 3

AIM AND SCOPE

3.1 EXISTING SYSTEM

Heart disease is even being highlighted as a silent killer which leads to the death of a person without obvious symptoms. The nature of the disease is the cause of growing anxiety about the disease & its consequences. Hence continued efforts are being done to predict the possibility of this deadly disease in prior. So that various tools & techniques are regularly being experimented with to suit the present-day health needs. 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, classifying them under suitable headings & finally analyzing to extract the desired data we can conclude. This technique can be very well adapted to the do the prediction of heart disease. As the well-known quote says “Prevention is better than cure”, early prediction & its control can be helpful to prevent & decrease the death rates due to heart disease.

3.2 PROPOSED SYSTEM

The working of the system starts with the collection of data and selecting the important attributes. 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. The accuracy of the system is obtained by testing the system using the testing data. This system is implemented using the following modules.

- 1.) Collection of Dataset
- 2.) Selection of attributes
- 3.) Data Pre-Processing
- 4.) Balancing of Data
- 5.) Disease Prediction

3.2.1 Collection of dataset

Initially, we collect a dataset for our heart disease prediction system. After the collection of the dataset, we split the dataset 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, 70% of training data is used and 30% of data is used for testing. The dataset used for this project is Heart Disease UCI. The dataset consists of 76 attributes; out of which, 14 attributes are used for the system.

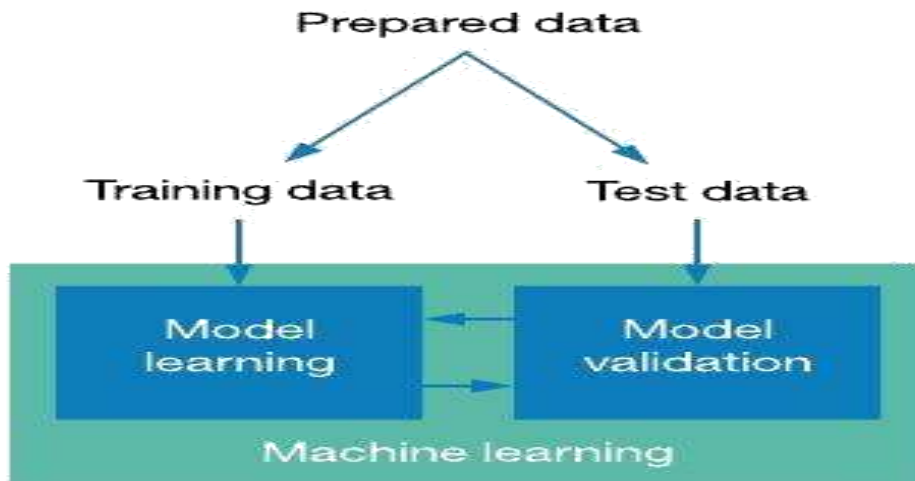


Figure [3.2.1]: Collection of Data

3.2.2 Selection of attributes

Attribute or Feature selection includes the selection of appropriate attributes for the prediction system. This is used to increase the efficiency of the system. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exam, etc. are selected for the prediction. The Correlation matrix is used for attribute selection for this model.



Figure [3.2.2]: Correlation of matrix

3.2.3 Pre-processing of Data

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. Preprocessing of data is required for improving the accuracy of the model.

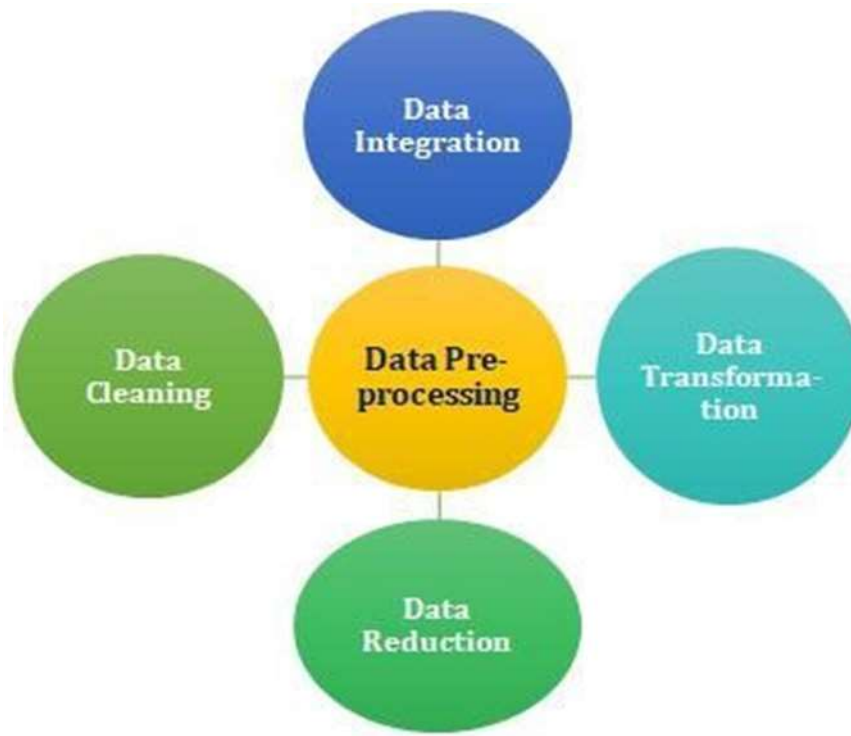


Figure [3.2.3]:Pre-processing of data

3.2.4 Balancing of Data

Imbalanced datasets can be balanced in two ways. They are Under Sampling and over Sampling

(a) Under Sampling:

In Under Sampling, dataset balance is done by the reduction of the size of the ample class. This process is considered when the amount of data is adequate.

(b) Over Sampling:

In Over Sampling, dataset balance is done by increasing the size of the scarce samples. This process is considered when the amount of data is inadequate.

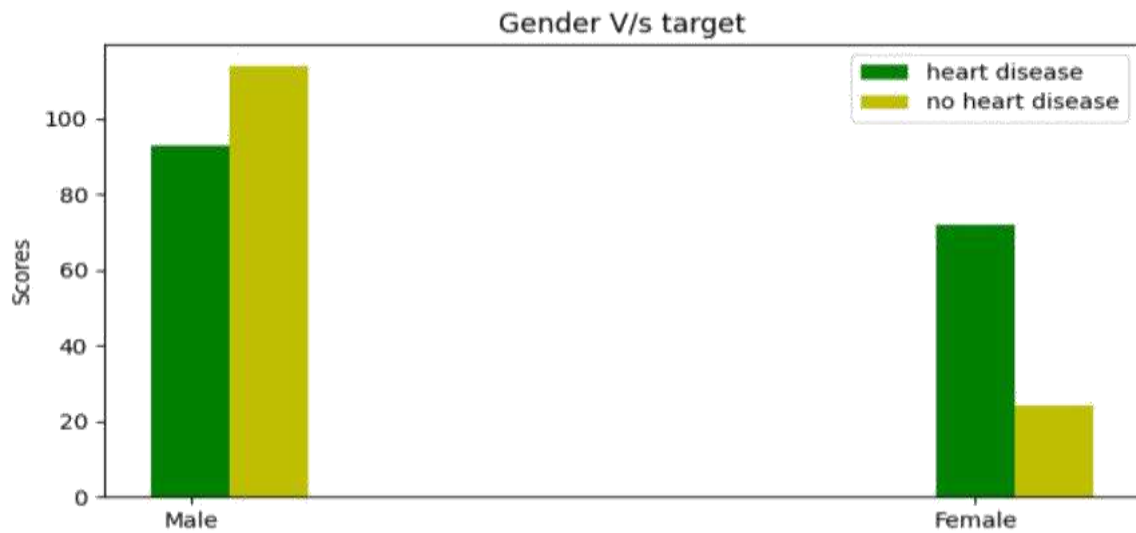


Figure [3.2.4]: Data Balancing

3.2.5 Prediction of Disease

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, Ada-boost, and Xg-boost are used for classification. Comparative analysis is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

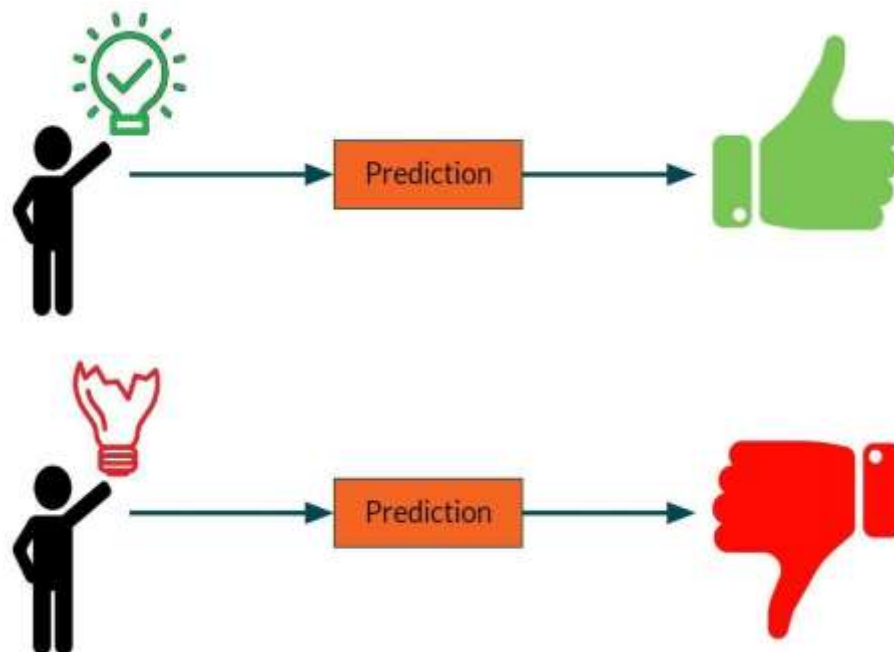


Figure [3.2.5]: prediction of disease

CHAPTER 4

EXPREMENTAL OR MATERIAL METHODS ALGORITHM

4.1 SYSTEM ARCHITECTURE

The system architecture gives an overview of the working of the system.

The working of this system is described as follows:

Dataset collection is collecting data which contains patient details. Attributes selection process selects the useful attributes for the prediction of heart disease. After identifying the available data resources, they are further selected, cleaned, made into the desired form. Different classification techniques as stated will be applied on preprocessed data to predict the accuracy of heart disease. Accuracy

Measure compares the accuracy of different classifiers.

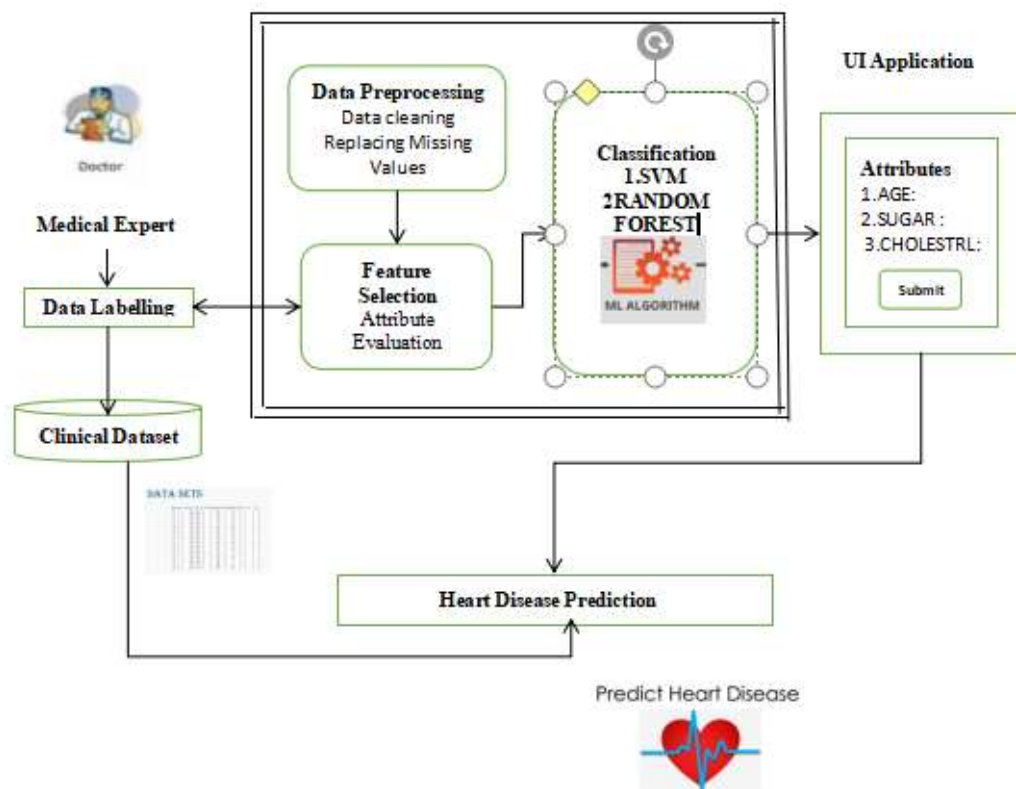


Figure [4]: SYSTEM ARCHITECTURE

4.2 MACHINE LEARNING

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

- *Supervised Learning*

Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on the basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

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 to map the input variable(x) with the output variable(y).

- *Unsupervised learning*

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar to how a human learns to think by their own
- Experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make Unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

- *Reinforcement learning*

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific

situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

4.3 ALGORITHMS 4.3.1 Support Vector Machine (Svm):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In the 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

The followings are important concepts in SVM -

Support Vectors - Data Points that are closest to the hyperplane are called support vectors. Separating line will be defined with the help of these data points.

Hyperplane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

Types of SVM:

SVM can be of two types:

- *Linear SVM*: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM Classifier.
- *Non-linear SVM*: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.
- The objective of the support vector machine algorithm is to find a hyperplane in a Dimensional space (N - the number of features) that distinctly classifies the data points.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where the number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different kernel functions can be specified for the decision function.
- Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid Over-fitting in choosing Kernel functions and regularization term is crucial.

SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

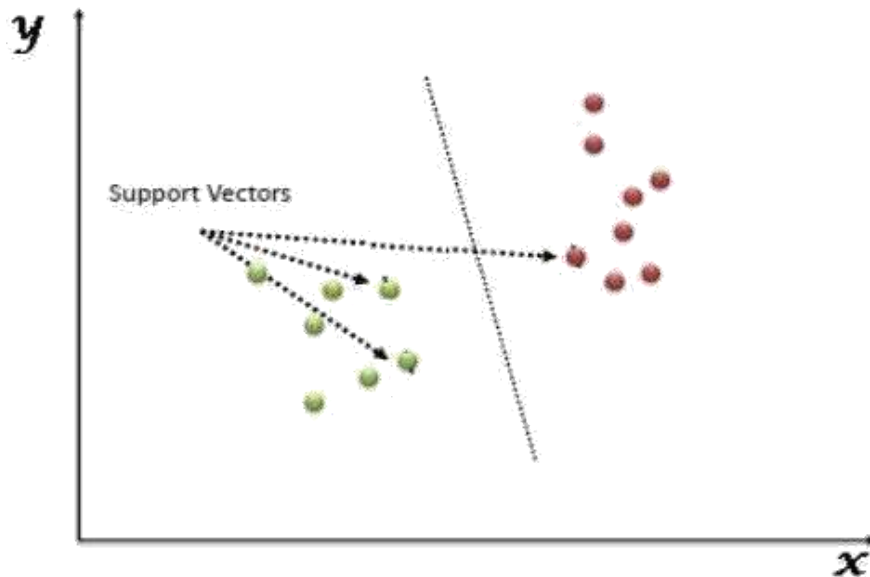


Figure 4.3.1]: Support Vector Machine

4.3.2 Naive Bayes Algorithm:

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's 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.

Types of Naive Bayes model:

There are three types of Naive Bayes Model, which are given below:

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

Multinomial: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.

Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular

word is present or not in a document. This model is also famous for document classification tasks.

4.3.3 DECISION TREE ALGORITHM

Decision Tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision Tree, there are two nodes, which are the Decision Node and Leaf Node.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a Decision Tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A Decision Tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

The Decision Tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for a regression problem.

The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision Tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

In Decision Tree the major challenge is to identify the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain:

When we use a node in a Decision Tree to partition the training instances into smaller subsets, the entropy changes. Information gain is a measure of this change in entropy.

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples.

The higher the entropy the more the information content.

2. Gini Index:

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred. Sklearn supports "Gini" criteria for Gini Index and by default, it takes "Gini" value.

The most notable types of Decision Tree algorithms are:-

ID3:

This algorithm uses Information Gain to decide which attribute is to be used to classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

1. *C4.5:* This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
2. *Classification and Regression Tree (CART):* It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Working:

In a Decision Tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the Decision Tree node, which contains the best Attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

4.3.4 RANDOM FOREST ALGORITHM

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. 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. The time complexity of the worst case of learning with Random Forests is $O(M(d \log n))$, where M is the number of growing trees, n is the number of instances, and d is the data dimension.

It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. 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. It also provides a pretty good indicator of the feature importance.

Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. 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 overfitting.

Assumptions:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Algorithm Steps:

It works in four steps:

- Select random samples from a given dataset.
- Construct a Decision Tree for each sample and get a prediction Result from each Decision Tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final prediction.

Advantages:

• Random Forest is capable of performing both Classification and Regression tasks.

- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages:

Although Random Forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

4.3.5 LOGISTIC REGRESSION ALGORITHM

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Advantages:

Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.

The predicted parameters (trained weights) give inference about the importance of each feature. The direction of association i.e. positive or negative is also given. So we can use Logistic Regression to find out the relationship between the features.

This algorithm allows models to be updated easily to reflect new data, unlike Decision Tree or Support Vector Machine. The update can be done using stochastic gradient descent.

Logistic Regression outputs well-calibrated probabilities along with classification results. This is an advantage over models that only give the final classification as results. If a training example has a 95% probability for a class, and another has a 55% probability for the same class, we get an inference about which training examples are more accurate for the formulated problem.

Disadvantages:

Logistic Regression is a statistical analysis model that attempts to predict precise probabilistic outcomes based on independent features. On high dimensional datasets, this may lead to the model being over-fit on the training set, which means overstating the accuracy of predictions on the training set and thus the model may not be able to predict accurate results on the test set. This usually happens in the case when the model is trained on little training data with lots of features. So on high dimensional datasets, Regularization techniques should be considered to avoid overfitting (but this makes the model complex). Very high regularization factors may even lead to the model being under-fit on the training data.

Nonlinear problems can't be solved with logistic regression since it has a linear decision surface. Linearly separable data is rarely found in real world scenarios. So the transformation of nonlinear features is required which can be done by increasing the number of features such that the data becomes linearly separable in higher dimensions.

Non-Linearly Separable Data:

It is difficult to capture complex relationships using logistic regression. More powerful and complex algorithms such as Neural Networks can easily outperform this algorithm

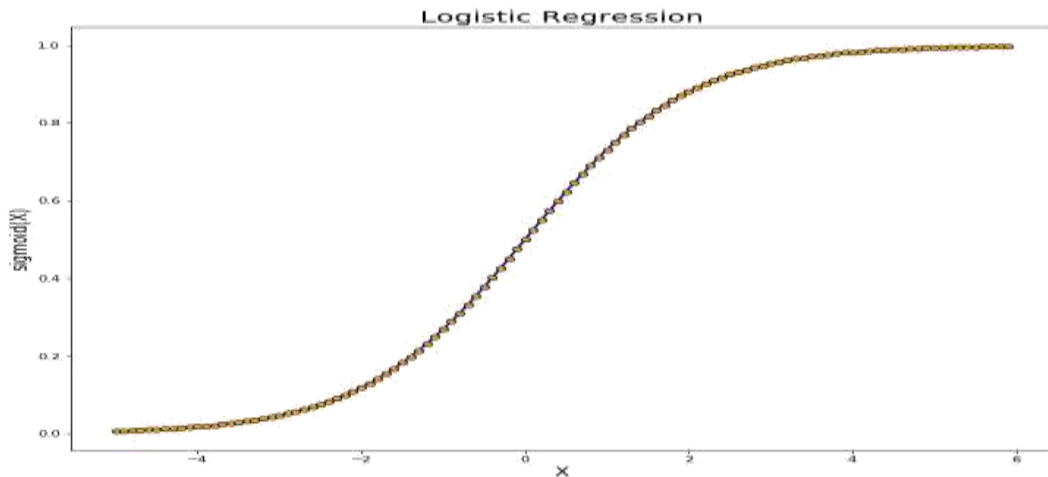


Figure [4.3.5]: Logistic Regression

4.3.6 ADABOOST ALGORITHM

Adaboost was the first really successful boosting algorithm developed for the purpose of binary classification. Adaboost is short for Adaptive Boosting and is a very popular boosting technique which combines multiple “weak classifiers” into a single “Strong classifier”

Algorithm:

1. Initially, Adaboost selects a training subset randomly.
2. It iteratively trains the Adaboost machine learning model by selecting the training set based on the accurate prediction of the last training.
3. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
4. Also, it assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
5. This process iterates until the complete training data fits without any error or until reached to the specified maximum number of estimators.
6. To classify, perform a "vote" across all of the learning algorithms you built

Advantages:

Adaboost has many advantages due to its ease of use and less parameter tweaking when compared with the SVM algorithms. Plus Adaboost can be used with SVM though theoretically, overfitting is not a feature of Adaboost applications, perhaps because the parameters are not optimized jointly and the learning process is slowed due to estimation stage-wise. This link is useful to understand mathematics. The

flexible Adaboost can also be used for accuracy improvement of weak classifiers and cases in image/text classification.

Disadvantages:

Adaboost uses a progressively learning boosting technique. Hence high-quality data is needed in examples of Adaboost vs Random Forest. It is also very sensitive to outliers and noise in data requiring the elimination of these factors before using the data. It is also much slower than the XG-boost algorithm.

4.3.7 XGBOOST ALGORITHM

XG-boost is an implementation of Gradient Boosted decision trees. It is a type of Software library that was designed basically to improve speed and model performance. In this algorithm, decision trees are created in sequential form. Weights play an important role in XG-boost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then assemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined predict.

1. Regularization: XG-boost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XG-boost is also called regularized form of GBM (Gradient Boosting Machine). While using Sickie Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. Alpha is used for L1 regularization and lambda is used for L2 regularization.
2. Parallel Processing: XG-boost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Sickie Learn library, nthread hyper-parameter is used for parallel processing. Nthread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for thread and the algorithm will detect automatically.
3. Handling Missing Values: XG-boost has an in-built capability to handle missing values. When XG-boost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

4. Cross Validation: XG-boost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.
5. Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XG-boost on the other hand make splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

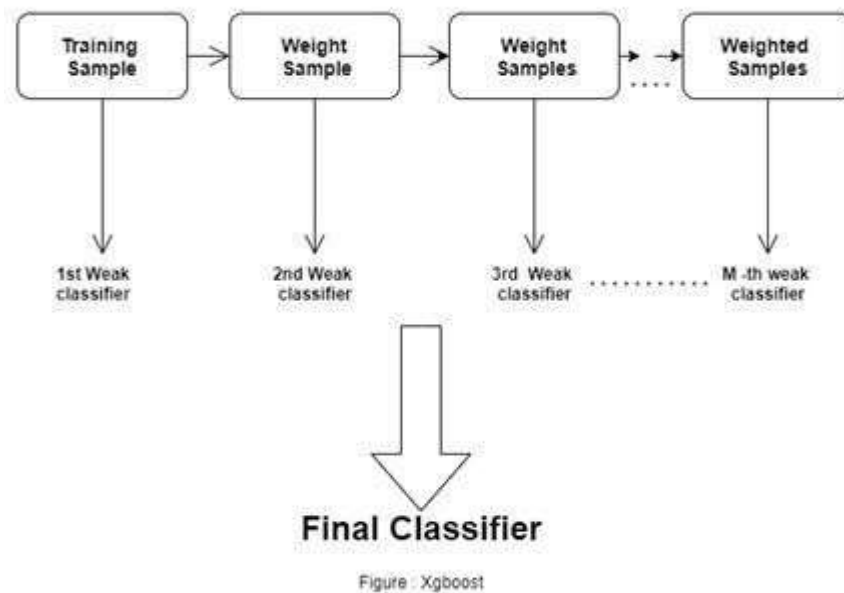


Figure 4.3.7]: xgboost

CHAPTER 5 Result And Discussion

5.1 SYSTEM CONFIGURATION

5.1.1 *Hardware requirements:*

Processor : Any Update Processor

Ram : Min 4GB

Hard Disk : Min 100GB

5.1.2 *Software requirements:*

Operating System : Windows family

Technology : Python3.7

IDE : Jupiter notebook

5.2 SAMPLE CODE

```
#import the libraries
```

```
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, StratifiedKFold, cross_val_score  
from sklearn.pipeline import make_pipeline, Pipeline from sklearn.model_selection  
import GridSearchCV
```

```
from sklearn.svm import SVC
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.tree import DecisionTreeClassifier
```

```

from sklearn.ensemble import RandomForestClassifier

from sklearn.externals import joblib

from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


from sklearn.metrics import log_loss


import warnings

warnings.simplefilter(action = 'ignore', category= FutureWarning)


from sklearn.ensemble import BaggingClassifier from sklearn.ensemble import
AdaBoostClassifier import numpy as np

from flask import Flask,request,jsonify, render_template

import pickle

app=Flask(__name__,template_folder='template')

app._static_folder = 'static'

model1=pickle.load(open('model1.pkl','rb'))

model2=pickle.load(open('model2.pkl','rb'))

@app.route('/home')

def homepage():
    return render_template('index.html')

@app.route('/precautions')

def precautions():
    return render_template('precautions.html')

```

```
@app.route('/advancedpage')
```

```
def advancedpage():
```

```
    return render_template('index.html')
```

```
@app.route('/quick',methods=['POST'])
```

```
def quick():
```

```
    def bmi(height,weight):
```

```
        bmi=int(weight)/((int(height)/100)**2) return bmi
```

```
    int_features1 = [float(x) for x in request.form.values()]
```

```
    age=int_features1[1]
```

```
    cigs=int_features1[3]
```

```
    height=int_features1[8]
```

```
    weight=int_features1[9]
```

```
    hrv=int_features1[10]
```

```
    int_features1.pop(8)
```

```
    int_features1.pop(9)
```

```
    bmi=round(bmi(height,weight),2)
```

```
    int_features1.insert(8,bmi)
```

```
    if int(int_features1[0])==1.0:
```

```
        sex="Male"
```

```
    else:
```

```
        sex="Female"
```

```

if int(int_features1[2])==1.0:
    smoking="Yes"
else:
    smoking="No"

if int(int_features1[4])==1.0:
    stroke="Yes"
else:
    stroke="No"

if int(int_features1[5])==1.0:
    hyp="Yes"
else:
    hyp="No"

if int(int_features1[7])==1.0:
    dia="Yes"

else:
    dia="No"
if int(int_features1[6])==1.0: bpmeds="Yes"
else:
    bpmeds="No"

final_feature1=[np.array(int_features1)]
prediction1= model1.predict(final_feature1)
result=prediction1[0]

if result==0:
    result="No need to worry"
else:

result="You are detected with heart problems. You need to consult
a doctor immediately"

return render_template('quick_report.html',prediction_text1=
result,gender=sex,age=age,smoking=smoking,cigs=cigs,stroke=stroke,hyp=hyp,dia=
di a,bpmeds=bpmeds,bmi=bmi,hrv=hrv)

@app.route('/quickpage')

def quickpage():
    return render_template('index1.html')

@app.route('/customersupport')

```

```

def customersupport():
    return render_template('customercare.html')

@app.route('/Doctorconsult')

def Doctorconsult():
    return render_template('Doctorconsult.html')

@app.route('/')

def home():
    returnrender_template('Home.html')

@app.route ('/advanced',methods=['POST']) def advanced():
int_features2      =      [int(x)      for      x      in      request.form.values()]
final2_feature=[np.array(int_features2)] prediction2= model2.predict(final2_feature)
result=prediction2[0]

age =int_features2[0]
trestbps= int_features2[3]
chol=int_features2[4]
oldspeak =int_features2[7]
thalach= int_features2[7]
ca=int_features2[10]

if int(int_features2[1])==1:
sex="Male"
else:
sex="Female"

if int(int_features2[2])==1:
cp="Typical angina"
elif int(int_features2[2])==2:
cp="Atypical angina"
elif int(int_features2[2])==3:
cp="Non-angina pain"
else:
cp="Asymtomatic"

if int(int_features2[5])==1:
fbs="Yes"
else:
fbs="No"

```

```

if int(int_features2[6])==1:
restecg="ST-T wave abnormality" elif int(int_features2[6])==2:
    restecg="showing probable or definite left ventricular hypertrophy by
Estes"
else:
restecg="Normal"

if int(int_features2[8])==1:
exang="Yes"
else:
exang="No"

if int(int_features2[9])==1:
slope="upsloping"
elif int(int_features2[9])==2:
slope="flat"
else:
slope="downsloping"

if int(int_features2[11])==3:
thal="Normal"
elif int(int_features2[11])==6:
thal="Fixed defect"
else:
thal=" reversable defect"
if result==0:
result="No need to worry"
else:

result="You are detected with heart problems. You need to consult
a doctor immediately"

```

```

return render_template('advance_report.html',prediction_text2=
result,age=age,sex=sex,cp=cp,trestbps=trestbps,chol=chol,fbs=fbs,restecg=restecg,
old_peak=oldspeak,exang=exang,slope=slope,ca=ca,thal=thal)

if __name__=="__main__":
app.run(debug=True)

#read the csv dataset

data = pd.read_csv("heart.csv", encoding='ANSI')

data.columns

data.head()
#Total number of rows and columns
data.shape
# Plot a line graph for Age V/s heart
disease plt.subplots(figsize =(8,5))
classifiers = ['<=40', '41-50', '51-60','61 and Above'] heart_disease = [13, 53, 64, 35]
no_heart_disease = [6, 23, 65, 44]
l1 = plt.plot(classifiers, heart_disease , color='g', marker='o', linestyle ='dashed',
markerfacecolor='y', markersize=10)

l2 = plt.plot(classifiers, no_heart_disease, color='r',marker='o', linestyle ='dashed',
markerfacecolor='y', markersize=10 )
plt.xlabel('Age')
plt.ylabel('Number of patients')
plt.title('Age V/s Heart disease')
plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
plt.show()

# Plot a bar graph for Gender V/s target
N = 2
ind = np.arange(N)
width = 0.1
fig, ax = plt.subplots(figsize =(8,4))
heart_disease = [93, 72]
rects1 = ax.bar(ind, heart_disease, width, color='g')
no_heart_disease = [114, 24]
rects2 = ax.bar(ind+width, no_heart_disease, width, color='y')

```

```

ax.set_ylabel('Scores')
ax.set_title('Gender V/s target')
ax.set_xticks(ind)
ax.set_xticklabels(('Male','Female'))

ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))

plt.show()

#Pie charts for thal:Thalassemia
# Having heart disease
labels= 'Normal', 'Fixed defect', 'Reversible defect'
sizes=[6, 130, 28]
colors=['red', 'orange', 'green']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', shadow=True,
startangle=140)

plt.axis('equal')

plt.title('Thalassemia blood disorder status of patients having heart disease')

plt.show()

# Not having heart disease
labels= 'Normal', 'Fixed defect', 'Reversible defect'
sizes=[12, 36, 89]
colors=['red', 'orange', 'green']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', shadow=True,
startangle=140)

plt.axis('equal')

plt.title('Thalassemia blood disorder status of patients who do not have heart
disease')

plt.show()

## Feature selection
#get correlation of each feature in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(13,13))

#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

```

```

data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)

target=data['target']
data = data.drop(['target'],axis=1)
data.head()

# We split the data into training and testing set:
x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3,
random_state=10)

## Base Learners clfs = []

kfolds = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
np.random.seed(1)

#Support Vector Machine(SVM)

pipeline_svm = make_pipeline(SVC(probability=True, kernel="linear",
class_weight="balanced"))

grid_svm = GridSearchCV(pipeline_svm,
param_grid = {'svc__C': [0.01, 0.1, 1]},
cv = kfolds,
verbose=1,
n_jobs=-1)
grid_svm.fit(x_train, y_train)
grid_svm.score(x_test, y_test)
print("\nBest Model: %f using %s" % (grid_svm.best_score_,
grid_svm.best_params_))

print("\n")

print('SVM LogLoss {score}'.format(score=log_loss(y_test,
grid_svm.predict_proba(x_test))))

clfs.append(grid_svm)

# save best model to current working directory joblib.dump(grid_svm,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_grid_svm = joblib.load("heart_disease.pkl" )

# get predictions from best model above
y_preds = model_grid_svm.predict(x_test)

print('SVM accuracy score: ',accuracy_score(y_test, y_preds))

```

```

print('\n')
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111) cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier') fig.colorbar(cax) ax.set_xticklabels([''] +
labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))
# Multinomial Naive Bayes(NB) classifierNB=MultinomialNB()
classifierNB.fit(x_train,y_train) classifierNB.score(x_test, y_test)
print('MultinomialNBLogLoss {score}'.format(score=log_loss(y_test,
classifierNB.predict_proba(x_test))))
clfs.append(classifierNB)
# save best model to current working directory joblib.dump(classifierNB,
"heart_disease.pkl")
# load from file and predict using the best configs found in the CV step
model_classifierNB = joblib.load("heart_disease.pkl" )
# get predictions from best model above
y_preds = model_classifierNB.predict(x_test)
print('MultinomialNB accuracy score: ',accuracy_score(y_test, y_preds))
print('\n')
import pylab as plt labels=[0,1] cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx) fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

```

```

fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print("\n")
print(classification_report(y_test, y_preds))

# Logistic Regression(LR) classifierLR=LogisticRegression()

classifierLR.fit(x_train,y_train)

classifierLR.score(x_test, y_test)
print('LogisticRegressionLogLoss {score}'.format(score=log_loss(y_test,
classifierLR.predict_proba(x_test))))

clfs.append(classifierLR)

# save best model to current working directory joblib.dump(classifierLR,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_classifierLR = joblib.load("heart_disease.pkl" )

# get predictions from best model above y_preds =
model_classifierLR.predict(x_test)

print('Logistic Regression accuracy score: ',accuracy_score(y_test, y_preds))
print("\n")

import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
Print (cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

Cax = ax.matshow (cmx)

```

```

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print('\n')

print(classification_report(y_test, y_preds))

# Decision Tree (DT)
classifierDT=DecisionTreeClassifier(criterion="gini", random_state=50,
max_depth=3, min_samples_leaf=5)

classifierDT.fit(x_train,y_train)

classifierDT.score(x_test, y_test)

print('Decision Tree LogLoss {score}'.format(score=log_loss(y_test,
classifierDT.predict_proba(x_test))))

clfs.append(classifierDT)
# save best model to current working directory joblib.dump(classifierDT,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_classifierDT = joblib.load("heart_disease.pkl" )

# get predictions from best model above

y_preds = model_classifierDT.predict(x_test)

print('Decision Tree accuracy score: ',accuracy_score(y_test, y_preds))

print('\n')

import pylab as plt

labels=[0,1]

cmx=confusion_matrix(y_test,y_preds, labels)

```

```

print(cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([""] + labels)

ax.set_yticklabels([""] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print("\n")

print(classification_report(y_test, y_preds))

# Random Forest(RF) classifierRF=RandomForestClassifier()
classifierRF.fit(x_train,y_train) classifierRF.score(x_test, y_test)
print('RandomForestLogLoss {score}'.format(score=log_loss(y_test,
classifierRF.predict_proba(x_test))))

clfs.append(classifierRF)
# save best model to current working directory joblib.dump(classifierRF,
"heart_disease.pkl")
# load from file and predict using the best configs found in the CV step
model_classifierRF = joblib.load("heart_disease.pkl" )

# get predictions from best model above

y_preds = model_classifierRF.predict(x_test)

print('Random Forest accuracy score: ',accuracy_score(y_test, y_preds))

print("\n")
import pylab as plt

labels=[0,1]

```

```

cmx=confusion_matrix(y_test,y_preds, labels)

print(cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()


print("\n")

print(classification_report(y_test, y_preds))
print("\n")
print('Accuracy of svm: {}'.format(grid_svm.score(x_test, y_test)))

print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))

Print ('Accuracy of logistic regression: {}'.format(classifierLR.score (x_test, y_test)))

print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))

print('Accuracy of random forest: {}'.format(classifierRF.score(x_test, y_test)))

###

#import the libraries

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, StratifiedKFold, cross_val_score
from sklearn.pipeline import make_pipeline, Pipeline from sklearn.model_selection
import GridSearchCV
from sklearn.svm import SVC

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.externals import joblib

from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import log_loss import warnings warnings.simplefilter(action =
'ignore', category= FutureWarning) from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
#read the csv dataset

data = pd.read_csv("heart.csv", encoding='ANSI')

data.columns
data.head()
#Total number of rows and columns
data.shape
# Plot a line graph for Age V/s heart
disease plt.subplots(figsize =(8,5))

classifiers = ['<=40', '41-50', '51-60','61 and Above'] heart_disease = [13, 53, 64, 35]
no_heart_disease = [6, 23, 65, 44]

l1 = plt.plot(classifiers, heart_disease , color='g', marker='o', linestyle ='dashed',
markerfacecolor='y', markersize=10)

l2 = plt.plot(classifiers, no_heart_disease, color='r',marker='o', linestyle ='dashed',
markerfacecolor='y', markersize=10 )

plt.xlabel('Age')

plt.ylabel('Number of patients')

plt.title('Age V/s Heart disease')

```

```
plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
```

```
plt.show()
```

```
# Plot a bar graph for Gender V/s target
```

```
N = 2
```

```
ind = np.arange(N)
```

```
width = 0.1
```

```
fig, ax = plt.subplots(figsize=(8,4)) heart_disease = [93, 72]
```

```
rects1 = ax.bar(ind, heart_disease, width, color='g')
```

```
no_heart_disease = [114, 24]
```

```
rects2 = ax.bar(ind+width, no_heart_disease, width, color='y')
```

```
ax.set_ylabel('Scores')
```

```
ax.set_title('Gender V/s target')
```

```
ax.set_xticks(ind)
```

```
ax.set_xticklabels(('Male','Female'))
```

```
ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))
```

```
plt.show()
```

```
#Pie charts for thal:Thalassemia
```

```
# Having heart disease
```

```
labels= 'Normal', 'Fixed defect', 'Reversible defect'
```

```
sizes=[6, 130, 28]
```

```
colors=['red', 'orange', 'green']
```

```
plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', shadow=True,  
startangle=140)
```

```
plt.axis('equal')
```

```
plt.title('Thalassemia blood disorder status of patients having heart disease')
```

```

plt.show()

# Not having heart disease

labels= 'Normal', 'Fixed defect', 'Reversable defect'

sizes=[12, 36, 89]

colors=['red', 'orange', 'green']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', shadow=True,
startangle=140)

plt.axis('equal')

plt.title('Thalassemia blood disorder status of patients who do not have heart
disease')

plt.show()

## Feature selection

#get correlation of each feature in dataset

corrmat = data.corr()

top_corr_features = corrmat.index

plt.figure(figsize=(13,13))

#plot heat map

g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)

target=data['target']

data = data.drop(['target'],axis=1)

data.head()

```

```

# We split the data into training and testing set:

x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3,
random_state=10)

## Base Learners clfs = []

kfolds = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
np.random.seed(1)

#Support Vector Machine(SVM)
pipeline_svm = make_pipeline(SVC(probability=True, kernel="linear",
class_weight="balanced"))

grid_svm = GridSearchCV(pipeline_svm,

param_grid = {'svc__C': [0.01, 0.1, 1]},
cv = kfolds,
verbose=1,

n_jobs=-1)

grid_svm.fit(x_train, y_train)

grid_svm.score(x_test, y_test)

print("\nBest Model: %f using %s" % (grid_svm.best_score_,
grid_svm.best_params_))

print('\n')

print('SVM LogLoss {score}'.format(score=log_loss(y_test,
grid_svm.predict_proba(x_test))))

clfs.append(grid_svm)

# save best model to current working directory joblib.dump(grid_svm,
"heart_disease.pkl")

```

```
# load from file and predict using the best configs found in the CV step
model_grid_svm = joblib.load("heart_disease.pkl" )
```

```
# get predictions from best model above
y_preds = model_grid_svm.predict(x_test)
print('SVM accuracy score: ',accuracy_score(y_test, y_preds))
print('\n')
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111) cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print('\n')
```

```

print(classification_report(y_test, y_preds))

# Multinomial Naive Bayes(NB) classifierNB=MultinomialNB()
classifierNB.fit(x_train,y_train) classifierNB.score(x_test, y_test)

print('MultinomialNBLogLoss {score}'.format(score=log_loss(y_test,
classifierNB.predict_proba(x_test))))

clfs.append(classifierNB)

# save best model to current working directory joblib.dump(classifierNB,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_classifierNB = joblib.load("heart_disease.pkl" )

# get predictions from best model above

y_preds = model_classifierNB.predict(x_test)

print('MultinomialNB accuracy score: ',accuracy_score(y_test, y_preds))

print('\n')

import pylab as plt labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)

print(cmx)

```

```

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print("\n")

print(classification_report(y_test, y_preds))

# Logistic Regression(LR) classifierLR=LogisticRegression()

classifierLR.fit(x_train,y_train)

classifierLR.score(x_test, y_test)

print('LogisticRegressionLogLoss {score}'.format(score=log_loss(y_test,
classifierLR.predict_proba(x_test))))

clfs.append(classifierLR)

# save best model to current working directory joblib.dump(classifierLR,
"heart_disease.pkl")

```

```

# load from file and predict using the best configs found in the CV step
model_classifierLR = joblib.load("heart_disease.pkl" )
# get predictions from best model above
y_preds = model_classifierLR.predict(x_test)

print('Logistic Regression accuracy score: ',accuracy_score(y_test, y_preds))

print('\n')

import pylab as plt

labels=[0,1]

cmx=confusion_matrix(y_test,y_preds, labels)

Print (cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()
print('\n')

print(classification_report(y_test, y_preds))
# Decision Tree (DT)

classifierDT=DecisionTreeClassifier(criterion="gini", random_state=50,
max_depth=3, min_samples_leaf=5)

classifierDT.fit(x_train,y_train)

classifierDT.score(x_test, y_test)

```

```

print('Decision Tree LogLoss {score}'.format(score=log_loss(y_test,
classifierDT.predict_proba(x_test))))

clfs.append(classifierDT)

# save best model to current working directory joblib.dump(classifierDT,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_classifierDT = joblib.load("heart_disease.pkl" )

# get predictions from best model above

y_preds = model_classifierDT.predict(x_test)

print('Decision Tree accuracy score: ',accuracy_score(y_test, y_preds))

print('\n')

import pylab as plt

labels=[0,1]

cmx=confusion_matrix(y_test,y_preds, labels)

print(cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

print('\n')

print(classification_report(y_test, y_preds))

```

```

# Random Forest(RF) classifierRF=RandomForestClassifier()
classifierRF.fit(x_train,y_train) classifierRF.score(x_test, y_test)
print('RandomForestLogLoss {score}'.format(score=log_loss(y_test,
classifierRF.predict_proba(x_test))))

clfs.append(classifierRF)

# save best model to current working directory joblib.dump(classifierRF,
"heart_disease.pkl")

# load from file and predict using the best configs found in the CV step
model_classifierRF = joblib.load("heart_disease.pkl" )

# get predictions from best model above

y_preds = model_classifierRF.predict(x_test)

print('Random Forest accuracy score: ',accuracy_score(y_test, y_preds))

print('\n')

import pylab as plt

labels=[0,1]

cmx=confusion_matrix(y_test,y_preds, labels)

print(cmx)

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(cmx)

plt.title('Confusion matrix of the classifier')

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

```

```

plt.show()

print('\n')

print(classification_report(y_test, y_preds))

print('\n')

print('Accuracy of svm: {}'.format(grid_svm.score(x_test, y_test))) print('Accuracy of
naive bayes: {}'.format(classifierNB.score(x_test, y_test))) print('Accuracy of logistic
regression: {}'.format(classifierLR.score(x_test, y_test))) print('Accuracy of decision
tree: {}'.format(classifierDT.score(x_test, y_test))) print('Accuracy of random forest:
{}'.format(classifierRF.score(x_test, y_test)))
///  

import numpy as np

from flask import Flask,request,jsonify, render_template

import pickle

app=Flask(__name__,template_folder='template')

app._static_folder = 'static'

model1=pickle.load(open('model1.pkl','rb'))

model2=pickle.load(open('model2.pkl','rb'))

@app.route('/home')

def homepage():
    return render_template('index.html')

@app.route('/precautions')

def precautions():
    return render_template('precautions.html')

@app.route('/advancedpage')

def advancedpage():
    return render_template('index.html')

@app.route('/quick',methods=['POST'])

def quick():
    def bmi(height,weight):
        bmi=int(weight)/((int(height)/100)**2)

```

```

return bmi
int_features1 = [float(x) for x in request.form.values()]

age=int_features1[1]

cigs=int_features1[3] height=int_features1[8] weight=int_features1[9]
hrv=int_features1[10] int_features1.pop(8) int_features1.pop(9)
bmi=round(bmi(height,weight),2)
int_features1.insert(8,bmi)

if int(int_features1[0])==1.0:
    sex="Male"
else:
    sex="Female"
if int(int_features1[2])==1.0:
    smoking="Yes"
else:
    smoking="No"
if int(int_features1[4])==1.0:
    stroke="Yes"
else:
    stroke="No"

if int(int_features1[5])==1.0:
    hyp="Yes"
else:
    hyp="No"
if int(int_features1[7])==1.0:
    dia="Yes"
else:
    dia="No"
if int(int_features1[6])==1.0:
    bpmeds="Yes"
else:
    bpmeds="No"

final_feature1=[np.array(int_features1)] prediction1= model1.predict(final_feature1)
result=prediction1[0]
if result==0:
    result="No need to worry"
else:

result="You are detected with heart problems. You need to consult
a doctor immediately"

return render_template('quick_report.html',prediction_text1=
result,gender=sex,age=age,smoking=smoking,cigs=cigs,stroke=stroke,hyp=hyp,dia=
di a,bpmeds=bpmeds,bmi=bmi,hrv=hrv)

@app.route('/quickpage')

```

```

def quickpage():
    return render_template('index1.html')

@app.route('/customersupport')

def customersupport():
    return render_template('customercare.html')

@app.route('/Doctorconsult')

def Doctorconsult():
    return render_template('Doctorconsult.html')

@app.route('/')

def home():
    return render_template('Home.html')

@app.route('/advanced',methods=['POST'])

def advanced():

    int_features2 = [int(x) for x in request.form.values()]
    final2_feature=[np.array(int_features2)] prediction2= model2.predict(final2_feature)
    result=prediction2[0]
    age=int_features2[0] trestbps=int_features2[3] chol=int_features2[4]
    oldspeak=int_features2[7] thalach=int_features2[7] ca=int_features2[10]

    if int(int_features2[1])==1:
        sex="Male"
    else:
        sex="Female"

    if int(int_features2[2])==1:
        cp="Typical angina"
    elif int(int_features2[2])==2:
        cp="Atypical angina"
    elif int(int_features2[2])==3:
        cp="Non-angina pain"
    else:
        cp="Asymtomatic"
    if int(int_features2[5])==1:
        fbs="Yes"
    else:
        fbs="No"

    if int(int_features2[6])==1:
        restecg="ST-T wave abnormality"
    elif int(int_features2[6])==2:

```

```

        restecg="showing probable or definite left ventricular hypertrophy by
Estes"
    else:
        restecg="Normal"

    if int(int_features2[8])==1:

        exang="Yes"
    else:
        exang="No"

    if int(int_features2[9])==1:
        slope="upsloping"
    elif int(int_features2[9])==2:
        slope="flat"
    else:
        slope="downsloping"

    if int(int_features2[11])==3:
        thal="Normal"
    elif int(int_features2[11])==6:
        thal="Fixed defect"
    else:
        thal=" reversable defect"
    if result==0:
        result="No need to worry"
    else:

        result="You are detected with heart problems. You need to consult
a doctor immediately"
    return render_template('advance_report.html',prediction_text2=
result,age=age,sex=sex,cp=cp,trestbps=trestbps,chol=chol,fbs=fbs,restecg=restecg,
old peak=oldspk,exang=exang,slope=slope,ca=ca,thal=thal)

if __name__=="__main__":

    app.run(debug=True)

```

5.3 DATASET DETAILS

- Of the 76 attributes available in the dataset, 14 attributes are considered for the prediction of the output.
- Heart Disease UCI : <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target	
2	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	
3	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	
4	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	
5	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	
6	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	
7	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1	
8	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1	
9	44	1	1	120	263	0	1	173	0	0	2	0	3	1	
10	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1	
11	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1	
12	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1	
13	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1	
14	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1	
15	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1	
16	58	0	3	150	283	1	0	162	0	1	2	0	2	1	
17	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1	
18	58	0	2	120	340	0	1	172	0	0	2	0	2	1	
19	66	0	3	150	226	0	1	114	0	2.6	0	0	2	1	
20	43	1	0	150	247	0	1	171	0	1.5	2	0	2	1	
21	69	0	3	140	239	0	1	151	0	1.8	2	2	2	1	
22	59	1	0	135	234	0	1	161	0	0.5	1	0	3	1	
23	44	1	2	130	233	0	1	179	1	0.4	2	0	2	1	
24	42	1	0	140	226	0	1	178	0	0	2	0	2	1	
25	61	1	2	150	243	1	1	137	1	1	1	0	2	1	
26	40	1	3	140	199	0	1	178	1	1.4	2	0	3	1	
27	71	0	1	160	302	0	1	162	0	0.4	2	2	2	1	
28	59	1	2	150	212	1	1	157	0	1.6	2	0	2	1	

Figure [5.3]: Dataset Attributes

Input dataset attributes

- Gender (value 1: Male; value 0 : Female)
- Chest Pain Type (value 1: typical type 1 angina, value 2: typical type angina, value 3: non-angina pain; value 4: asymptomatic)
- Fasting Blood Sugar (value 1: > 120 mg/dl; value 0:< 120 mg/dl)
- Exang – exercise induced angina (value 1: yes; value 0: no)
- CA – number of major vessels colored by fluoroscopy (value 0 – 3)
- Thal (value 3: normal; value 6: fixed defect; value 7:reversible defect)
- Trest Blood Pressure (mm Hg on admission to the hospital)
- Serum Cholesterol (mg/dl)
- Thalach – maximum heart rate achieved

- Age in Year
- Height in cms
- Weight in Kgs.
- Cholestrol
- Restecg

Table [1]: Attributes Of the Dataset

S. No.	Attribute	Description	Type
1	Age	Patient's age (29 to 77)	Numerical
2	Sex	Gender of patient(male-0 female-1)	Nominal
3	Cp	Chest pain type	Nominal
4	Trestbps	Resting blood pressure(in mm Hg on admission to hospital ,values from 94 to 200)	Numerical
5	Chol	Serum cholesterol in mg/dl, values from 126 to 564)	Numerical
6	Fbs	Fasting blood sugar>120 mg/dl, true-1 false-0)	Nominal

7	Resting	Resting electrocardiographics result (0 to 1)	Nominal
8	Thali	Maximum heart rate achieved(71 to 202)	Numerical
9	Exang	Exercise included agina(1-yes 0- no)	Nominal
10	Oldpeak	ST depression introduced by exercise relative to rest (0 to .2)	Numerical
11	Slope	The slop of the peak exercise ST segment (0 to 1)	Nominal
12	Ca	Number of major vessels (0-3)	Numerical
13	Thal	3-normal	Nominal
14	Targets	1 or 0	Nominal

5.4 PERFORMANCE ANALYSIS

In this project, various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Forest, Logistic Regression, Adaboost, XG-boost are used to predict heart disease. Heart Disease UCI dataset, has a total of 76 attributes, out of those only 14 attributes are considered for the prediction of heart disease. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exang, etc. are considered for this project. The accuracy for individual algorithms has to be measured and whichever algorithm is giving the best accuracy that is considered for the heart disease prediction. For evaluating the experiment, various evaluation metrics like accuracy, confusion matrix, precision, recall, and f1-score are considered.

Accuracy- Accuracy is the ratio of the number of correct predictions to the total number of inputs in the dataset. It is expressed as:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

Confusion Matrix- It gives us a matrix as output and gives the total performance of the system.

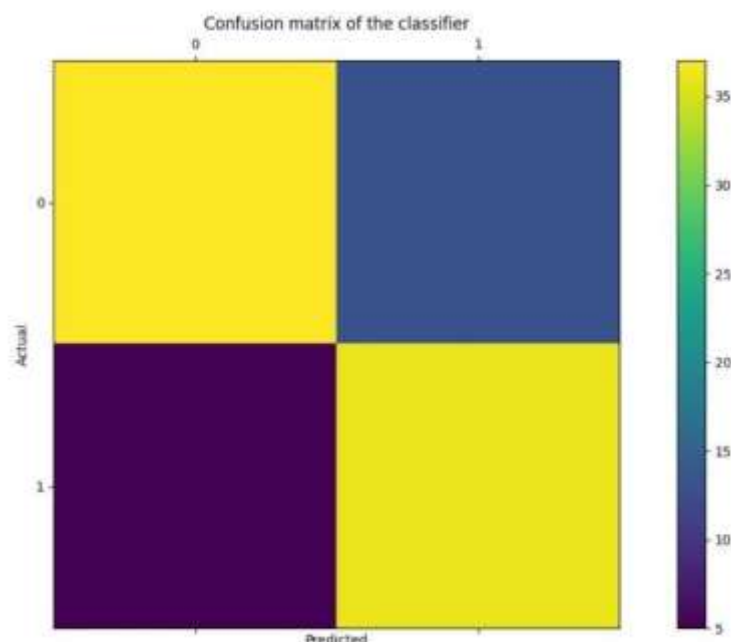


Figure [5.4]: Confusion Matrix

Where

TP: True positive

FP: False Positive

FN: False Negative

TN: True Negative

Correlation Matrix: The correlation matrix in machine learning is used for feature selection. It represents dependency between various attributes.



Figure [5.5]: Correlation matrix

Precision- It is the ratio of correct positive results to the total number of positive results predicted by the system.

It is expressed as:

Recall-It is the ratio of correct positive results to the total number of positive results Predicted by the system.

It is expressed as:

F1 Score-It is the harmonic mean of Precision and Recall. It measures the test accuracy. The range of this metric is 0 to 1.

5.5 PERFORMANCE MEASURES

- The highest accuracy is given by XG-boost.

```
Majority Voting accuracy score: 0.7912087912087912  
Weighted Average accuracy score: 0.8131868131868132  
Bagging_accuracy score: 0.8021978021978022  
Ada_boost_accuracy score: 0.7362637362637363  
Gradient_boosting_accuracy score: 0.8131868131868132
```

5.6 INPUT AND OUTPUT

5.6.1 Input

Home Know Your Status

QUICK DIAGNOSIS

Age: 22

Sex: Male

Weight: 10

Height: 172

Heart rate: 70

BP: 72

Print

Home Know Your Status

ADVANCED DIAGNOSIS

Age: 22

Sex: Male

Weight: 10

Height: 172

Heart rate: 70

BP: 72

Print

ADVANCED DIAGNOSIS

63

♂ Male

asymtomatic

123

233

Yes

showing probable or definite left ventricu

76

Yes

3

Flat

0

Normal

Predict

Figure [5.6.1]: Input

5.6.2 Output

[Home](#) [New User Status](#)

[Quick](#)
[Advanced](#)

QUICK DIAGNOSIS REPORT

Features	User Data
Sex	Male
Age	63.0
Current Smoker	Yes
Cigarette per day	8.0
Prevalent Stroke	Yes
Prevalent Hypertension	Yes
Prevalent Diabetes	Yes
Blood Pressure medication	Yes
Body Mass Index	29.74
Maximum Heart Rate Achieved	78.0
Result	No need to worry

*This result is does not have standard medical approval. So for final result please approach a doctor.
For more accurate result use Advanced diagnosis.

[Generate PDF](#)

[Home](#) [New User Status](#)

[Quick](#)
[Advanced](#)

ADVANCED DIAGNOSIS REPORT

Features	User Data
Age	63
Sex	Female
Chest pain	Atypical angina
Resting blood pressure	125
Serum cholesterol	232
Fasting blood sugar greater than 125mg/dL	Yes
Resting electrocardiogram results	showing probable or definite left ventricular hypertrophy by ECG
Exercise induced angina	No
ST depression induced by exercise relative to rest	36
The slope of the peak exercise ST segment	downsloping
Number of major vessels (0-3) colored by fluoroscopy	1
Thal	reversible defect
Result	No need to worry

*This result is does not have standard medical approval. So for final result please approach a doctor.
For more accurate result use Advanced diagnosis.

[Generate PDF](#)

ADVANCED DIAGNOSIS REPORT

Features

Age:

Sex:

Chest pain:

Resting blood pressure:

Serum cholestoral:

Fasting blood sugar greater than 120mg/dl:

User Data

63

Male

Asymtomatic

123

233

Yes

Resting electrocardiographic results:

Exercise induced anigna:

ST depression induced by exercise relative to rest:

The slope of the peak exercise ST segment:

Number of major vessels (0-3) colored by flourosopy:

Thal:

Result

showing probable or definite left ventricular hypertrophy by Estes

Yes

76

downsloping

2

reversable defect

No need to worry

*This result is does not have standard medical approval. So for final result please approach a doctor.
For more accurate result use Advanced diagnosis

Generate PDF

Figure [5.6.2]: output

5.7 RESULT

After performing the machine learning approach for training and testing we find that accuracy of the XG-boost is better compared to other algorithms. Accuracy is calculated with the support of the confusion matrix of each algorithm, here the number count of TP, TN, FP, and FN is given and using the equation of accuracy, value has been calculated and it is concluded that extreme gradient boosting is best with 81% accuracy and the comparison is shown below.

TABLE [2] Accuracy comparison of algorithms

Algorithm	Accuracy
XG-boost	81.3%
SVM	80.2%
Logistic Regression	79.1%
Random Forest	79.1%
Naive Bayes	76.9%
Decision Tree	75.8%
Adaboost	73.6%

CHAPTER 6

SUMMARY AND CONCLUSION

Heart diseases are a major killer in India and throughout the world, application of promising technology like machine learning to the initial prediction of heart diseases will have a profound impact on society. The early prognosis of heart disease can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine. The number of people facing heart diseases is on a raise each year. This prompts for its early diagnosis and treatment. The utilization of suitable technology support in this regard can prove to be highly beneficial to the medical fraternity and patients. In this paper, the seven different machine learning algorithms used to measure the performance are SVM, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression, Adaptive Boosting, and Extreme Gradient Boosting applied on the dataset.

The expected attributes leading to heart disease in patients are available in the dataset which contains 76 features and 14 important features that are useful to evaluate the system are selected among them. If all the features taken into the consideration then the efficiency of the system the author gets is less. To increase efficiency, attribute selection is done. In this n features have to be selected for evaluating the model which gives more accuracy. The correlation of some features in the dataset is almost equal and so they are removed. If all the attributes present in the dataset are taken into account then the efficiency decreases considerably.

All the seven machine learning methods accuracies are compared based on which one prediction model is generated. Hence, the aim is to use various evaluation metrics like confusion matrix, accuracy, precision, recall, and f1-score which predicts the disease efficiently. Comparing all seven the extreme gradient boosting classifier gives the highest accuracy of 81%.

REFERENCE

- [1] .Machine Learning towards Prediction Of cardio vascular disease, 2018 Second International Conference for Electronic, Communication, and Aerospace Technology, Aditi Gavhane, Gouthami Kokkula, Isha Pandya, and Prof. Kailas Devadkar (ICECA).
- [2] . Prediction about Cardiovascular Disease Have used Evolutionary Rule Learning, 2018 1st International Conference upon Computation Intelligence and Communication Technology, Aakash Chauhan; Aditya Jain; Purushottam Sharma; Vikas Deep (CICT).
- [3] . R Latha and P Vetrivelan, 2019 11th International Conference on Telecommunication Systems and Networks, "Blood Viscosity based Cardiovascular Risk Prediction Model in Edge/Fog Computing" (COMSNETS).
- [4] . An Evaluation of Strategies for Predicting Heart Disease, 2018 ieee international conference on network, computing, robotics, and networking, G. Shanmugasundaram, V. Malar Selvam, R. Saravanan, and S. Balaji (icscan).
- [5] . Using Data Mining, a Novel Methodology for Earlier Detection of Myocardial infarction mi Heart Disease, 2019 3rd International Conference on Trends in Electrical and Informatics, Dhara B. Mehta and Nirali C. Varnagar (ICOEI).
- [6] . Using a big data strategy, real-time machine learning can be used to detect cardiac problems early. 2019 International Conference in Wireless Technologies, Ubiquitous and Machine Intelligence, Abderrahmane Ed-Daoudy and Khalil Maalmi (WITS).
- [7] . Radhanath Patra and Bonomali Khuntia, (2019) Institute of electrical and electronics International Conference on Electronics, Computer, and Modern Communications, Predictive Analysis of Rapid Spread of Cardiovascular Disease using Data Mining (ICECCT).
- [8] . CART-C is a system for heart disease diagnosis. 2018 International Conference on Computer Networks and Bioinformatics, Priyal Chotwani, Asmita Tiwari, Vikas Deep, and Purushottam Sharma (ICCCI).

- [9] . K. M. Zubair Hasan, Shourob Datta, Md Zahid Hasan, and Nusrat Zahan, 2019 International Conference on Electronics, Computer, and Communication Engineering, Automation Prediction of Cardiac Diseases Using Sparse Discriminant Analysis (ECCE).
- [10] AutomatedCardiovascular Disease Detection Using Non-Laboratory Health Risks,2018 12th International Conference on Medical Communication and Information Technology, H. Mai, T. T. Pham, D. N. Nguyen, and E. Dutkiewicz (ISMICT).

