ABSTRACT

Prediction of a heart attack is very important since it is one of the leading causes of sudden death. The healthcare industry produces large amounts of health care data daily that can be used to extract information for predicting heart attack that can happen to a patient in future while using the treatment history and health status. This hidden information in the healthcare data will be later used for affective decision making for patient's health. Researchers are focusing on developing software with the help of machine learning algorithms which can help doctors to take decision about patient's health regarding both prediction and diagnosing of heart disease. The main objective of this project is predicting the heart disease of a patient using machine learning algorithms. Comparative study of the various performances of machine learning algorithms will be done through graphical representation of the results. The objective and scope of this project is to predict the chances of getting heart attack before it occurs, then early treatment can be given to the patients which can reduce risk to the life and save life of patients and reduce cost of the treatment. In this project we perform the comparative analysis of classifiers like decision tree, Naïve Bayes, Logistic Regression, SVM and XG-boost 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 Random Forest which can give the better accuracy and predictive analysis.

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

INDEX

LIST OF FIGURES

LIST OF SYMBOLS

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

LIST OF TABLES

LIST OF ABBREVATIONS

ML	Machine Learning
SVM	Support Vector Machine
XG	Extreme Gradient
LR	Logistic Regression
NB	Naïve Bayes
RF	Random Forest

CHAPTER 1
INTRODUCTION

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 and important role in circulatory system [1]. If the heart doesn't function properly in that case it will lead to serious health conditions including death. It is very important to know about the difference between normal heart and diseased heart and to know whether your heart is a normal one or diseased one. In the existing scenario, the person's blood sample has to be collected and should perform ECG scan in the laboratory for the result. So, this process will take a long time to get into the diagnosis stage. The objective and scope of this project is to predict the chances of getting heart attack before it actually occurs, then early treatment can be given to the patients which can reduce risk to the life. Our approaches include two steps:

✓ Firstly, we select 13 features, i.e., age, gender, chest pain value, rest bps, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise induced angina, old peak, slope, number of vessels colored, and thal. The dataset is collected from the standard dataset repository www.kaggle.com. The preprocessing was done by identifying all the corresponding fields and s

✓ Secondly, we create a Machine Learning Techniques for predicting Heart Attack based on some of these features. We take several approaches for this problem to check accuracy i.e., Decision Trees [1], Support Vector Machines [2], Logistic Regression [3] and Naive Bayes [4]. Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the health care industry. This project aims to predict future heart disease by analyzing data of patients whichclassifies whether they have heart disease or not using machine-learning algorithm. Machine Learning techniques can be a boon in this regard.

Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease. Machine learning is an application of Artificial Intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine Learning focuses on the development of computer programs that can access and use it learn for themselves.

1.1 FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation.

Technical Feasibility

Operation Feasibility

Economic Feasibility

1.2 EXISTING SYSTEM

In the existing system for a person to know or predict his Heart Attack condition he has to consult a doctor and take some tests which is time taking and tiresome.

1.3 PROPOSED SYSTEM

In the existing system for a person to know or predict his Heart Attack condition he has to consult a doctor and take some tests which is time taking and tiresome. This system is implemented using the following modules.

1. Collection of Dataset
2. Selection of attributes
3. Data Preprocessing
4. Balancing of Data
5. Disease Prediction

1.3.1 Collection of data set

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 [1]. 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 [2]. The following fig 3.1.1 shows the data collection.

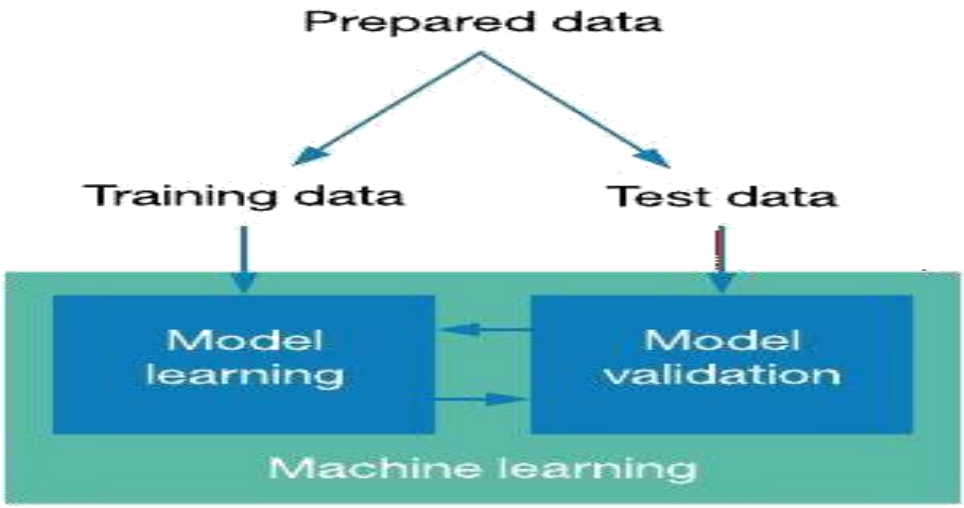


Fig1.3.1.: Depicts the collection of data

1.3.2Selection 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, exang, etc are selected for the prediction [3]. The Correlation matrix is used for attribute selection for this model. The following fig shows the correlation matrix.

1.3.3 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 [4]. 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 [5]. The following diagram shows the data pre-processing. The following fig shows the data pre-processing.

Fig 1.3.3: Data Pre-processing

1.3.4 Balancing of Data

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

(a) Under Sampling:

In Under Sampling, data set balance is done by the reduction of the size of the ampleclass. 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. The following chart shows the information of data balancing. The below fig 1.3.4 shows the data balancing.

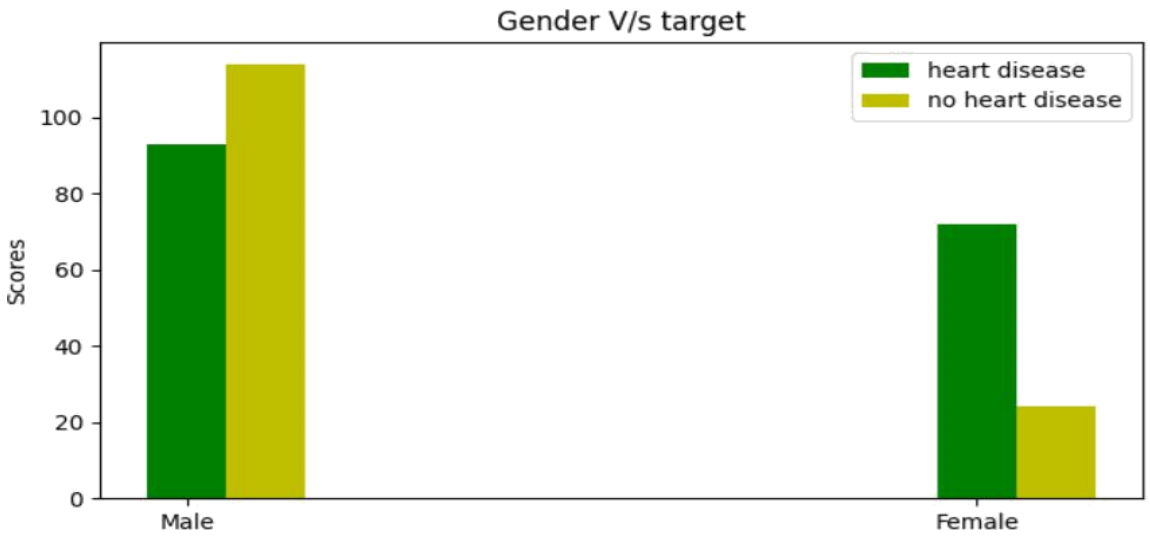


Fig 1.3.4: Data Balancing

1.3.5 Prediction of Disease

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, Ada-boost, 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 [5]. The following fig shows the how to predict heart disease.

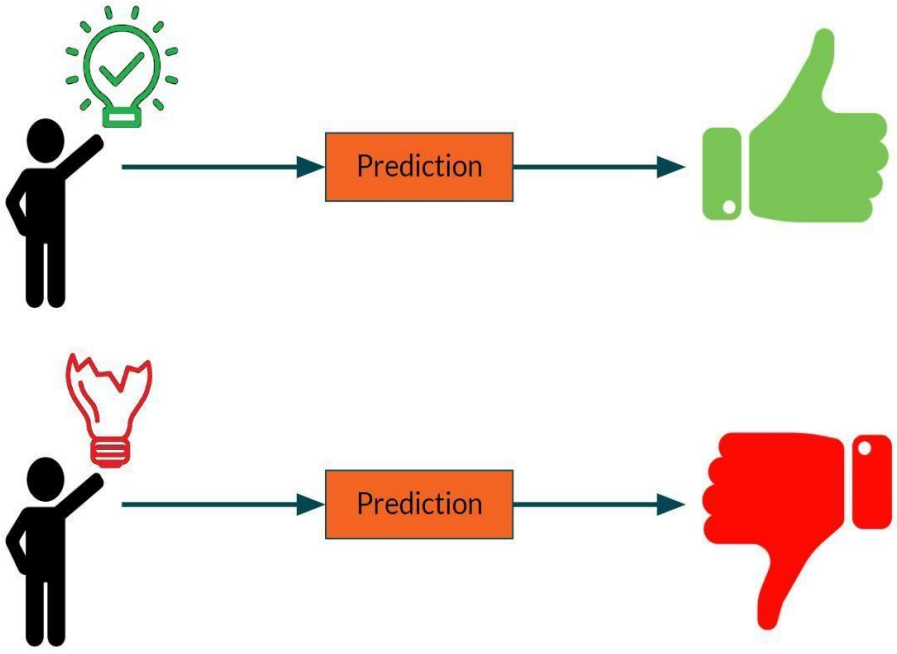


Fig 1.3.5: Predicts the heart diseases

CHAPTER 2

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

Software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system [6]. To achieve this we need to continuous communication with customers to gather all requirements. A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real-life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers create test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results. It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules [5]. Used appropriately, software requirements specifications can help prevent software project failure. The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

2.1.1 Purpose

Improve cardio muscular health and reduce deaths from heart disease and stroke. Heart disease is the leading cause of death in the United States, and stroke is the fifth leading cause. Healthy People 2030 focuses on preventing and treating heart disease and stroke and improving overall cardiovascular health.

2.1.2 Scope

Machine learning approach will be used for best analysis of the heart diseases and for earlier prediction of diseases so that the rate of the death cases can be minimized by the awareness about the diseases. Internet of Things (IOT), Healthcare and Machine Learning Increasing use of Internet of Things has promising benefits in healthcare. Dynamically collecting patient data using remote sensors can help in early detection of health problems and aid in preventive care.

2.1.3 Overview

The aim of our heart disease prediction project is to determine if a patient should be diagnosed with heart disease or not, which is a binary outcome.

Positive result = 1, the patient will be diagnosed with heart disease.

Negative result = 0, the patient will not be diagnosed with heart disease.

We have to find which classification model has the greatest accuracy and identify correlations in our data. Finally, we also have to determine which features are the most influential in our heart disease diagnosis.

2.1.4 Main Objective

The main objective of this research is to develop a heart prediction system. The system can discover and extract hidden knowledge associated with diseases from a historical heart data set

Heart disease prediction system aims to exploit data mining techniques on medical data set to assist in the prediction of the heart diseases.

2.1.5 Specific Objectives

• Provides new approach to concealed patterns in the data.

• Helps avoid human biasness.

• To implement Naïve Bayes Classifier that classifies the disease as per the input of the user.

• Reduce the cost of medical tests.

2.1.6 Justification

Clinical decisions are often made based on doctor's insight and experience rather than on the knowledge rich data hidden in the dataset. This practice leads to unwanted biases, errors and excessive medical costs which affect the quality of service provided to patients. The proposed system will integrate clinical decision support with computer-based patient records (Data Sets).

This will reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modelling and analysis tools,

e.g., data mining, have the potential to generate a knowledge rich environment which can

help to significantly improve the quality of clinical decisions [7].There are voluminous records in medical data domain and because of this, it has become necessary to use data mining techniques to help in decision support and prediction in the field of healthcare.

2.2 EXTERNAL INTERFACE REQURIMENTS

2.2.1 Hardware requirements:

Processer : i3 or AMD Ryzen 3 or above

RAM : Min 4GB

Hard Disk : Min 100GB

2.2.2 Software requirements:

Operating System : Windows 7 and Above

Technology : Python3.7

IDE : Jupyter notebook

2.3 SYSTEM FEATURES

FUNCTIONAL REQUIREMENTS

2.3.1 Data Collection

Data collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques [2]. A researcher can evaluate their hypothesis based on collected data. In most cases, data collection is the primary and most important step for research, irrespective of the field of research. The approach of data collection is different for different fields of study, depending on the required information [4]. The most critical objective of data collection is ensuring that information-rich and reliable data is collected for statistical analysis so that data-driven decisions can be made for research.

2.3.2 Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data [3].

2.3.3 Data Labelling

Supervised machine learning, which we'll talk about below, entails training a predictive model on historical data with predefined target answers. An algorithm must be shown which target answers or attributes to look for [8]. Mapping these target attributes in a dataset is called labelling. Data labelling takes much time and effort as datasets sufficient for machine learning may require thousands of records to be labelled. For instance, if your image recognition algorithm must classify types of bicycles.

2.3.4 Data Selection

Data selection is defined as the process of determining the appropriate data type and source, as well as suitable instruments to collect data. Data selection precedes the actual practice of data collection [2]. This definition distinguishes data selection from selective data reporting (selectively excluding data that is not supportive of a research hypothesis) and interactive/active data selection (using collected data for monitoring activities/events, or conducting secondary data analyses) [9]. The process of selecting suitable data for a research project can impact data integrity. After having collected all information, a data analyst chooses a subgroup of data to solve the defined problem. For instance, if you save your customers' geographical location, you don't need to add their cell phones and bank card numbers to a dataset. But purchase history would be necessary. The selected data includes attributes that need to be considered when building a predictive model.

2.3.5 Data Pre-processing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors [10]. Data pre-processing is a proven method of resolving such issues. The purpose of pre-processing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model.

The technique includes data formatting, cleaning, and sampling.

2.3.5.1 Data formatting

The importance of data formatting grows when data is acquired from various sources by different people. The first task for a data scientist is to standardize record formats. A specialist checks whether variables representing each attribute are recorded in the same way [9]. Titles of products and services, prices, date formats, and addresses are examples of variables. The principle of data consistency also applies to attributes represented by numeric ranges.

2.3.5.2 Data cleaning

This set of procedures allows for removing noise and fixing inconsistencies in data. A data scientist can fill in missing data using imputation techniques, e.g., substituting missing values with mean attributes. A specialist also detects outlier's observations that deviate significantly from the rest of distribution [8]. If an outlier indicates erroneous data, a data scientist deletes or corrects them if possible. This stage also includes removing incomplete and useless data objects.

2.3.5.3 Data anonymization

Sometimes a data scientist must anonymize or exclude attributes representing sensitive

information (i.e., when working with healthcare and banking data).

2.3.5.4 Data sampling

Big datasets require more time and computational power for analysis. If a dataset is too large, applying data sampling is the way to go. A data scientist uses this technique to select a smaller but representative data sample to build and run models much faster, and at the same time to produce accurate outcomes [7].

2.3.6 Data Transformation

Data transformation is the process of converting data from one format or structure into another format or structure. Data transformation is critical to activities such as data integration and data management. Data transformation can include a range of activities: you might convert data types, cleanse data by removing nulls or duplicate data, enrich the data, or perform aggregations, depending on the needs of your project. Scaling [6]. Data may have numeric attributes (features) that span different ranges, for example, millimeters, meters, and kilometers. Scaling is about converting these attributes so that they will have the same scale, such as between 0 and 1, or 1 and 10 for the smallest and biggest value for an attribute [1]. Decomposition Sometimes finding patterns in data with features representing complex concepts is more difficult. Decomposition technique can be applied in this case. During decomposition, specialist converts higher level features into lower-level ones. In other words, new features based on the existing ones are being added [5]. Decomposition is mostly used in time series analysis. For example, to estimate a demand for air conditioners per month, a market research analyst converts data representing demand per quarters.

Aggregation: - Unlike decomposition, aggregation aims at combining several features into a feature that represents them all. For example, you have collected basic information about your customers and particularly their age. To develop a demographic segmentation strategy, you need to distribute them into age categories, such as 16-20, 21-30, 31-40, etc. You use aggregation to create large-scale features based on small-scale ones [4]. This technique allows you to reduce the size of a dataset without the loss of information.

2.3.7 Data Splitting

A dataset used for machine learning should be partitioned into three subsets - training, test, and validation sets.

Training set: - A data scientist uses a training set to train a model and define its optimal parameters -parameters it must learn from data.

Test set: - A test set is needed for an evaluation of the trained model and its capability for generalization. The latter means a model's ability to identify patterns in new unseen data after having been trained over a training data. It is crucial to use different subsets for training and

testing to avoid model overfitting.

Validation set: - The purpose of a validation set is to tweak a model's hyperparameters -higher-level structural settings that cannot be directly learned from data[9]. These settings can express, for instance, how complex a model is and how fast it finds patterns in data.

The more training data a data scientist uses, the better the potential model will perform. Consequently, more results of model testing data lead to better model performance and generalization capability.

2.3.8 Modelling

After pre-processing the collected data and split it into three subsets, we can proceed with a model training [3]. This process entails "feeding" the algorithm with training data. An algorithm will process data and output a model that is able to find a target value (attribute) in new data — an answer you want to get with predictive analysis. The purpose of model training is to develop a model.Two model training styles are most common — supervised and unsupervised learning. The choice of each style depends on whether you must forecast specific attributes or group data objects by similarities.

Model evaluation and testing: The goal of this step is to develop the simplest model able to formulate a target value fast and well enough. A data scientist can achieve this goal through model tuning. That is the optimization of model parameters to achieve an algorithm's best performance. One of the more efficient methods for model evaluation and tuning is cross-validation.

Cross-validation: Cross-validation is the most used tuning method. It entails splitting a training dataset into ten equal parts (folds). A given model is trained on only nine folds and then tested on the tenth one (the one previously left out). Training continues until every fold is left aside and used for testing. As a result of model performance measure, a specialist calculates a cross-validated score for each set of hyperparameters [1]. A data scientist trains models with different sets of hyperparameters to define which model has the highest prediction accuracy. The cross-validated score indicates average model performance across ten hold-out folds.

2.3.9 Model Deployment

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the machine learning life cycle and can be one of the most cumber some [10]. Often, an organization's IT systems are incompatible with traditional model-building languages, forcing data scientists and programmers to spend valuable time and brainpower rewriting them.

2.4 Non-Functional Requirements

Nonfunctional necessities describe however a system should behave and establish constraints of its practicality. This type of requirements is also known as the system's quality attributes. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Some Non-Functional Requirements are as follows:

2.4.1 Reliability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

2.4.2 Maintainability

The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

2.4.3 Performance

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. As the structure should not capitulate when various customers would use everything the while [8]. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a updating of patient details including 15 parameters [7]. Report generation is done by considering these details. It generates the report of the patient by considering the parameters and defines whether the patient has risk or not.

2.4.4 Usability

The system should be easy to use. The system also should be user friendly for users because anyone can use it instead of programmers**.**

2.4.5 Supportability

The system should require Python knowledge to maintenance. If any problem acquire in user side and deep learning methods, it requires code knowledge and deep learning background to solve.

CHAPTER 3

SYSTEM ANALYSIS

LITERATURE SURVEY

With growing development in the field of medical science alongside machine learning various experiments and researches has been carried out in these recent years releasing the relevant significant papers.

[5] Purushottam, et, al proposed a paper "Efficient Heart Disease Prediction System "using hill climbing and decision tree algorithms. They used Cleveland dataset andpreprocessing of data is performed before using classification algorithms. TheKnowledge Extraction is done based on Evolutionary Learning (KEEL), an open-source data mining tool that fills the missing values in the data set. A decision treefollows top-down order. For each actual node selected by hill-climbing algorithm anode is selected by a test at each level. The parameters and their values used areconfidence. Its minimum confidence value is 0.25. The accuracy of the system is about 86.5%.

[6] Santhanas Krishnan. J, et, al proposed a paper "Prediction of Heart Disease Using Machine Learning Algorithms" using decision tree and Naive Bayes algorithm forprediction of heart disease. In decision tree algorithm the tree is built using certainconditions which gives True or False decisions. The algorithms like SVM, KNN areresults based on vertical or horizontal split conditions depends on dependent variables.But decision tree for a tree like structure having root node, leaves and branches baseon the decision made in each of tree Decision tree also help in the understating theimportance of the attributes in the dataset. They have also used Cleveland data set.Dataset splits in 70% training and 30% testing by using some methods. This algorithmgives 91% accuracy. The second algorithm is Naive Bayes, which is used forclassification. It can handle complicated, nonlinear, dependent data so it is foundsuitable for heart disease dataset as this dataset is also complicated, dependent andnonlinear in nature. This algorithm gives an 87% accuracy.

[7] Sonam Nikhar et al proposed paper "Prediction of Heart Disease Using MachineLearning Algorithms" their research gives point to point explanation of Naïve Bayes and decision tree classifier that are used especially in the prediction of Heart Disease.

[8] Some analaysis has been let to think about the execution of data on mining strategy on the same dataset, and the result decided that Decision Tree has highest accuracy that Bayesian classifier.

3.1 UML DIAGRAMS

In the field of software engineering, the Unified Modelling Language (UML) is a standardized visual specification language for object modelling. UML is a general-purpose modelling language that includes a graphical notation that includes a graphical notation used to create an abstract model of a team, referred to as a UML model. The model also contains a "Semantic backplane"-documentation such as written use cases that drive the model elements and diagrams.

The Importance of uml diagrams

A modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. A modelling language such as UML is thus a standard language for software blueprints. The UML is not a visual programming language, but its models can be directly connected to various programming languages. This means that it is possible to map from a model in the UML to a programming language Java, C++ or Visual Basic, or even to tables in a relational database or the persistent store of an object-oriented database. This mapping permits forward engineering: the generation of code from a UML model into a programming language. The reverse is also possible you can reconstruct a model from an implementation back into UML. This is a programming language that is used for object-oriented software development. To organize program code more efficiently, programmers often create "objects"that are sets of structured data within programs. UML, which has been standardized by the Object Management Group (OMG), was designed for this purpose. The language has gained enough support that it has become a standard language for visualizing and constructing software programs.

A Conceptual model of uml

The three major elements of UML are

1.The UML's basic building blocks.

2.The rules that dictate how those building blocks may be put together.

3.Some common mechanism that applies throughout the UML.

3.1.1 Use case diagram

The below use case diagram we can clearly observe the detail flow for a specific use case diagram**.** In this use case diagram sends a Build request to the model and also to train the model according to the given input. Now the end user sends a test input to the model so that it analysis the given data. Now the model sends the final result so that the end user can view the result. The following fig 3.1.1 shows the use case diagram.

Fig 3.1.1: Depicts use case diagram

3.1.2 Sequence diagram

The below sequence diagram we can clearly observe the detail flow for a specific sequence diagram diagram**.** In this sequence diagram sends a Build request to the model and also to train the model according to the given input. Now the end user sends a test input to the model so that it analysis the given data. Now the model sends the final result so that the end user can view the result. The following fig 3.1.2 shows the sequence diagram.

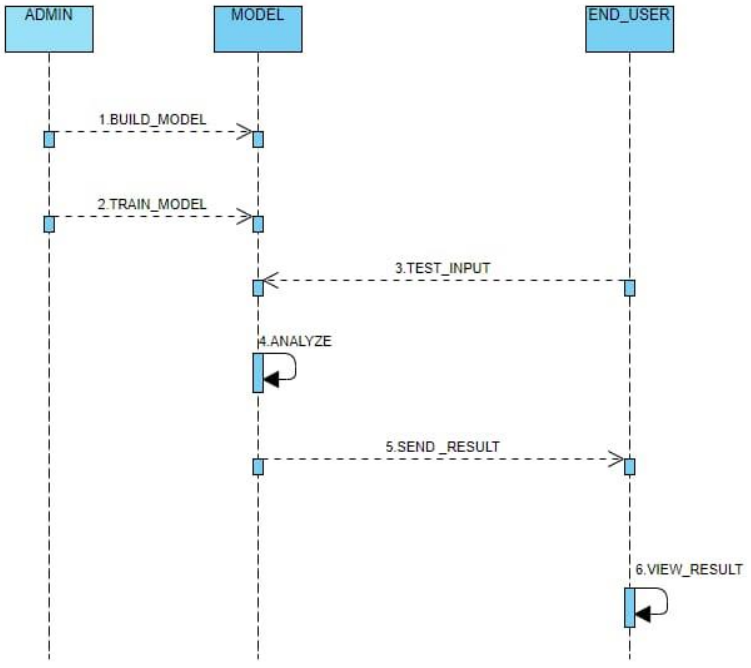


Fig 3.1.2: Depicts Sequence diagram

3.1.3 Collaboration diagram

The collaboration diagram we can see that the objects interact to perform the behavior of a particular use case or a part of a use case. The following fig 3.1.3 shows the collaboration diagram**.** diagram sends a Build request to the model and also to train the model according to the given input. Now the end user sends a test input to the model so that it analysis the given data. Now the model sends the final result so that the end user can view the result.

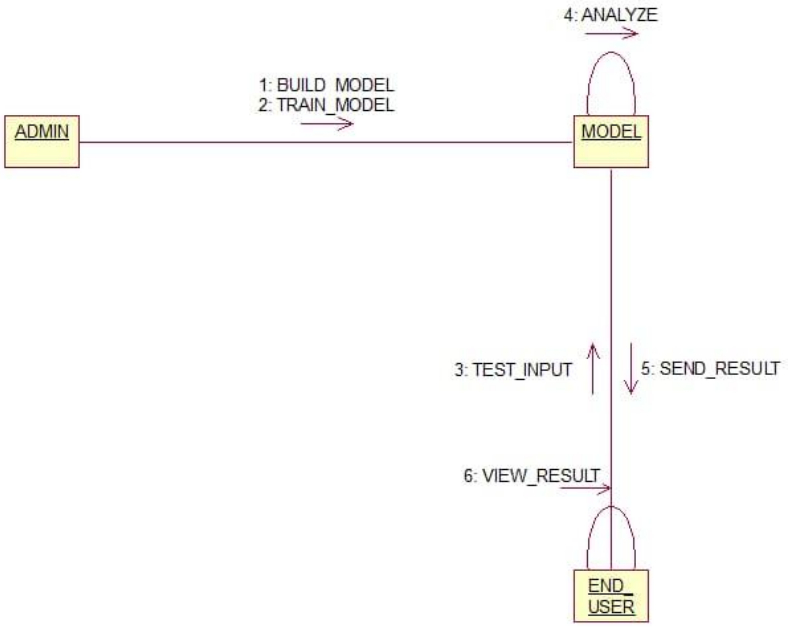


Fig 3.1.3: Depicts Collaboration Diagram

CHAPTER 4

SYSTEM DESIGN

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. The following fig 4.1 shows the architecture of the system.

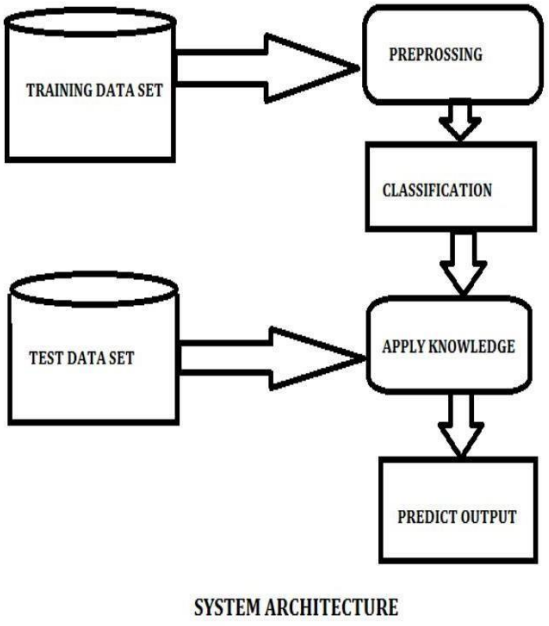


Fig 4.1: System Architecture

4.1.1 Training data

Training Data is nothing but enriched or labeled data you need to train your models [6]. You might just need to collect more of it to sharpen your model accuracy. But, the chances ofusing your data pretty low because, as you build a great model you need great training data at scale.

4.1.2 Test data set

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are includedin the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

4.1.3 Preprocessing

Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn.

4.1.4 Preprocessing Steps

The following are the Preprocessing steps:

1.Handling Null Values

2.Standardization

3.Handling Categorical Variables

4. One-Hot Encoding

5. Multicollinearity

4.1.5 Classification

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

4.1.6 Classification Models

In Classification we are using the following four models.

1. Decision Tree
2. SVM
3. Naive Bayes
4. Logistic Regression

4.2 DATA FLOW DIAGRAM

The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

4.2.1 RULES FOR CREATING A DFD

The name of the entity should be easy and understandable without any extra assistance (like comments).

The processes should be numbered or put in ordered list to be referred easily.

The DFD should maintain consistency across all the DFD levels.

A single DFD can have maximum processes up to 9 and minimum 3 processes.

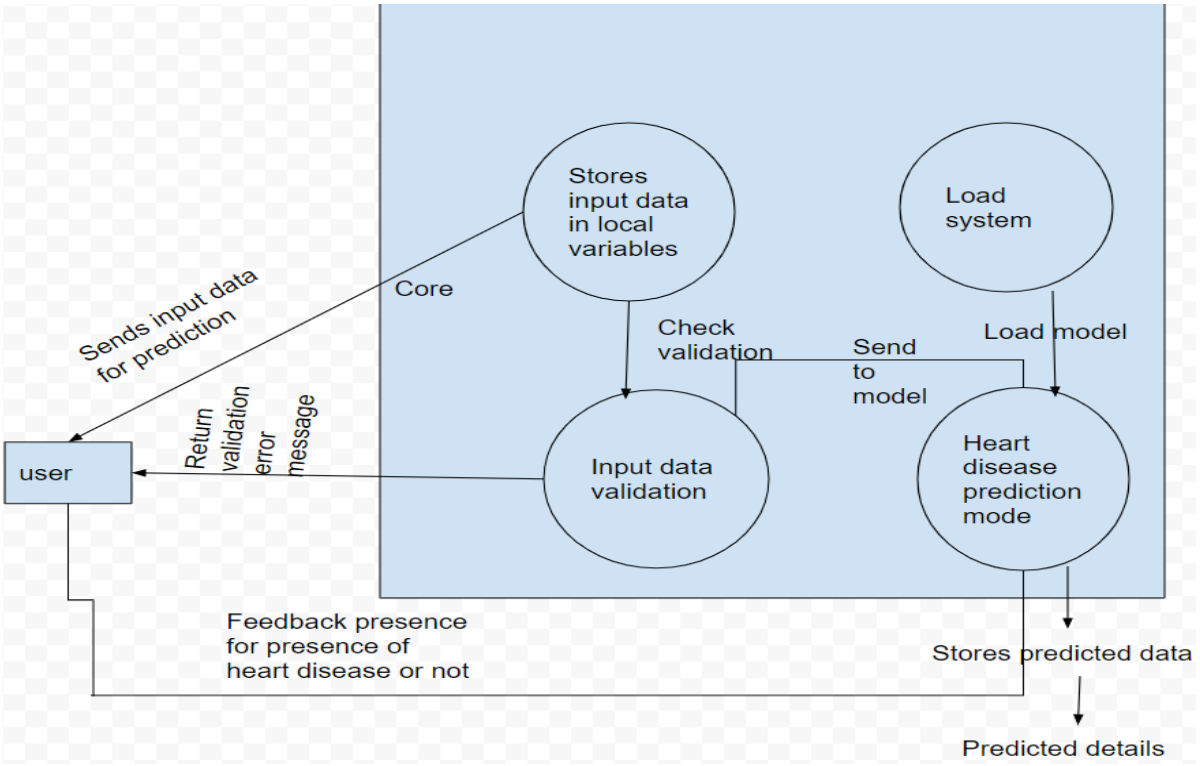


fig4.2:Depicts Data flow diagram

LEVELS OF DFD

DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

0-level DFD

1-level DFD:

2-level DFD:

4.3 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributesand their relation between the classes.

The class diagram is the main building block in object-oriented modeling. It is used both for general conceptual modelling of the semantics of the application and for detail modelling translating the model into programming code. The classes in a class diagram represent both the main objects and or interactionsin the application and the objects to be programmed. In the class diagram these classes are represented by boxes which contain three parts:

1.The upper part holds the name of the class.

2.The middle part contains attributes of a class. Attributes of a class may be public, privateor protected.

3.The bottom part gives the methods or operations the class can take or undertake.

The following fig 4.3 shows the class diagram.

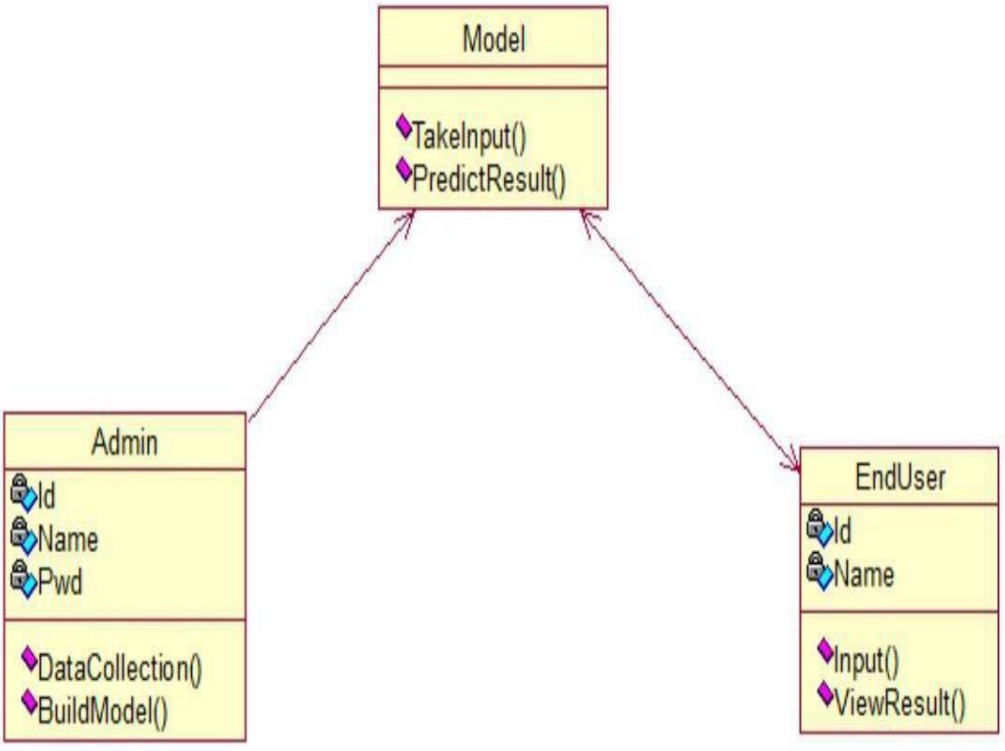


Fig 4.3: Class Diagram

CHAPTER 5

IMPLEMENTATION

5.1 INTRODUCTION TO TECHNOLOGY

5.1.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

5.1.2 Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Pythonis a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python. Here are naming conventions for Python identifiers.

Class names start with an uppercase letter. All other identifiers start with a lowercase letter. Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

5.1.3 Python Lists

1.A list is a collection of elements. These elements may be homogeneous or heterogeneous.

2.A list is a value that contains multiple values in an ordered sequence. The term list value refers to the list itself (which is a value that can be stored in a variable or passed to a function like any other value)

3.Just as string values are typed with quote characters to mark where the string begins and ends, a list begins with an opening square bracket and ends with a closing square bracket, [].

4.Values inside the list are also called items. Items are separated with commas (that is, they are comma-delimited).

5.A list also allows duplicate elements

6.Insertion order is preserved in list.

7.List elements are separated by commas and enclosed within square brackets.

8.Every element in the list has its own unique index number.

9.List supports both forward indexing and backward indexing; forward index starts from 0and backward index starts from -1.

10.We access either specific element by using indexing or set of elements by using slicing from the List. We can create list in different ways. Like by using list () function, by using square brackets This list() allows only one string value with set of characters If we give int type data in the list() functionthen interpreter will throw 'Type Error'errors.

e.g.:

>>> List1=list () #creating empty list

>>>print (List1) []

>>>type (List1) <class 'list'>

5.1.4 Python Tuple

Tuple is used to represent a set of homogeneous or heterogeneous elements into a singleentity. Tuple objects are immutable that means once if we create a tuple later we cannot modifythat All elements are separated by commas (,) and enclosed by parentheses. Parentheses are Optional. Tuple allows duplicate elements. Every element in the tuple has its own index number Tuple supports both forward indexing and also backward indexing, forward indexing starts From 0 and backward indexing starts from -1. If we take only one element in the tuple then we use comma (,) after that single element.

Tuples can be used as keys to the dictionary.

We can create a tuple in different ways, like with tuple(), with () or without () also.

The main difference between lists and tuples is- Lists are enclosed in brackets ()them elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.

Creating a tuple with tuple.

Eg1:

>>>tup=tuple ([10,20,30, True, 'Python'])

>>>print(tup) (10, 20, 30, True, 'Python')

>>>type(tup) <class 'tuple'>

>>>id(tup) 52059

5.1.5 Dictionary:

Each key is separated from its value by a colon (:), the items are separated by commas, andthe whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.Keys are unique within a dictionary while values may not be. The values of a dictionary canbe of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples. Accessing Values in Dictionary. To access dictionary elements, you can use the familiar square brackets along with the key toobtain its value. Following is a simple example –

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result dict['Name']: Zaradict['Age']: 7

5.1.5.1Python is Interpreted:

Python is processed at runtime by the interpreter. You do not need to compile yourprogram before executing it. This is similar to PERL and PHP.

5.1.5.1 Python is Interactive

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python comes with an interactive interpreter. When you type python in your shell or command prompt,the python interpreter open up with a >>> prompt and waiting for your instructions.

>>> says that you are inside the python interpreter

```
$ python

Python 2.7.6 (default, Apr 24 2015, 09:38:35)

[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on Darwin Type "help", "copyright", "credits" or "license" for more information.

    >>>
```

If you want to exit the Python console at any point, just type exit() or use the shortcut Ctrl + Z forWindows and Ctrl + D for Mac/Linux. Then you won't see >>> any longer.

5.1.5.2 Python is Object-Oriented

Python supports Object-Oriented style or technique of programming that encapsulates within objects.

Variables - In Python there are no declarations.

5.1.5.3 Dynamically Typed

Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java.A static typed language requires the programmer to explicitly tell the computer what type of "thing" eachdata value is. For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type. This tells the compiler that the only data that can beused for that variable must be a floating point numberlike etc., a number with a decimal point. If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

5.1.6 Python is a Beginner's Language

Python is a great language for the beginner-level programmers and supports thedevelopment of a wide range of applications from simple text processing.

5.1.7 Running Python Scripts

Open your text editor, type the following text and save it as "hello.py".

```
print "Hello, World!"
```

And run this program by calling "python hello.py". Make sure you change to the directory where you saved the file before doing it.

C:\Users\USER\Desktop> python

5.2 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine Learning focuses on the development of computer programs that can access and use it learn for themselves. The process of learning begins with observations or data, such as examples, direct experience,or instruction, in order to look for patterns in data and make better decisions in the future. Theprimary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

5.2.1 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 aproblem/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 [10]. 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 targetvariable, for which the decision tree uses the tree representation to solve the problemin which the leaf node corresponds to a class label and attributes are represented on theinternal 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 atree-like structure.

In Decision Tree the major challenge is to identify the attribute for the root node in each level.

5.2.1.1 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.

5.2.1.2 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: -

IDichotomiser 3 (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. This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifyingattribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.

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 variation.

5.2.1.5 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 completedataset.

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 bestattributes

Step-4: Generate the Decision Tree node, which contains the bestattribute.

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.

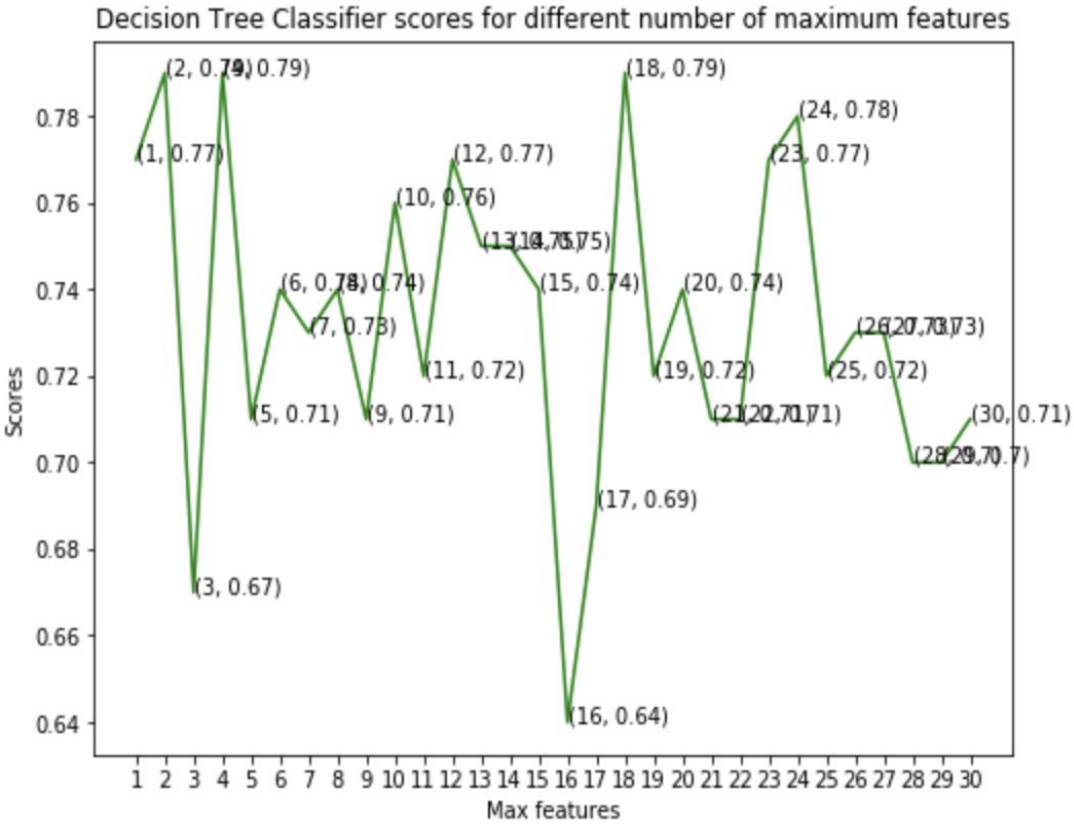


Fig 5.2.1.5:Output for decision tree algorithm

5.2.2 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 pointin 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 spacewhich 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 isconsidered 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 ifa 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 an N-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 supportvectors), 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 Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using anexpensive five-fold cross-validation.

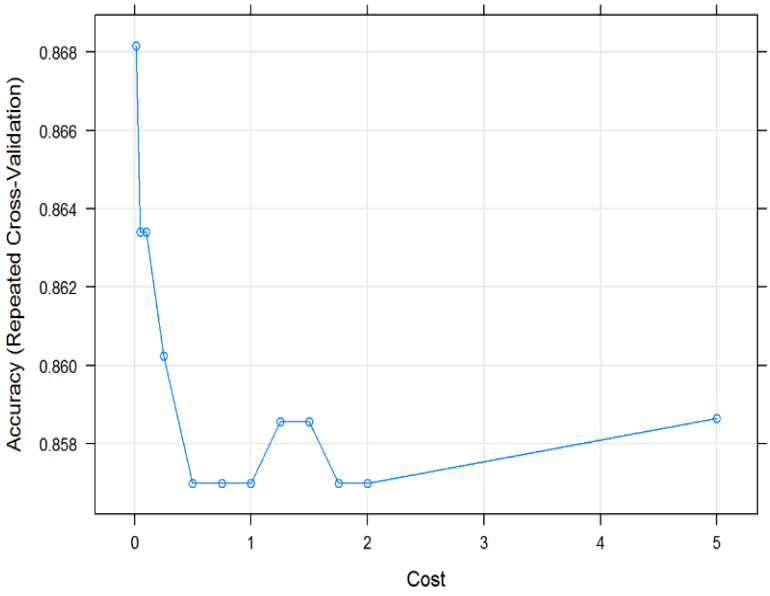


Fig 5.2.2: Output for SVM algorithm

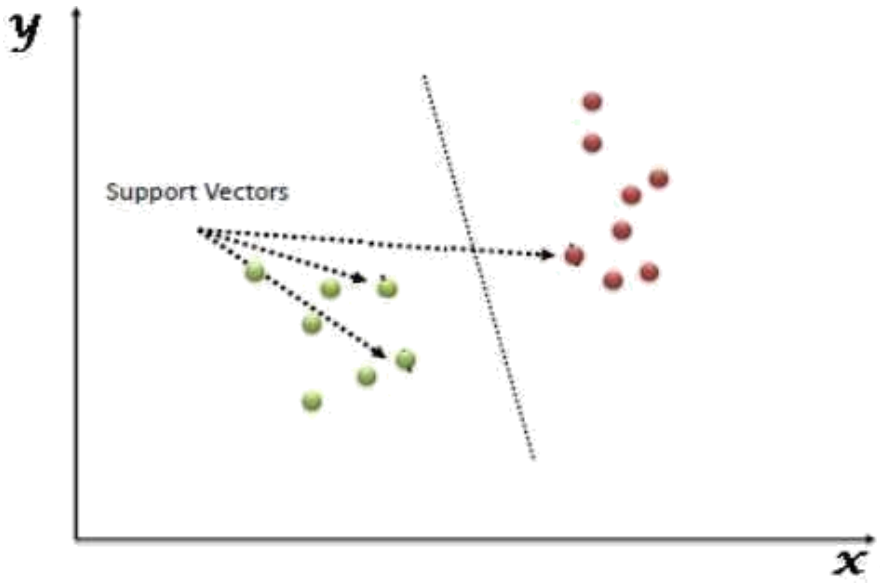


Fig 5.2.2: Support Vector Machine

5.2.3 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 ofindependence 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 isan apple without depending on each other.

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

5.2.4 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

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,

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 probabilityof 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

Then the model assumes that these values are sampled from the Gaussian distribution.

Multinomial: The Multinomial Naïve Bayes classifier is used when the data ismultinomial 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 forthe 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.

5.2.5 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 suchas whether the cells are cancerous or not, a mouse is obese or not based on itsweight, 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 predictprecise 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 casewhen the model is trained on little training data with lots of features. So, on high dimensional datasets, Regularization techniques should be considered to avoid over- fitting (but this makes the model complex). Very high regularization factors may even lead to the model being under-fit on the training data.

Nonl inear problems can't be solved with logistic regression since it has alinear 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 powerfuland complex algorithms such as Neural Networks can easily outperform this algorithm.The following fig 5.1.4 shows a logistic Regression.

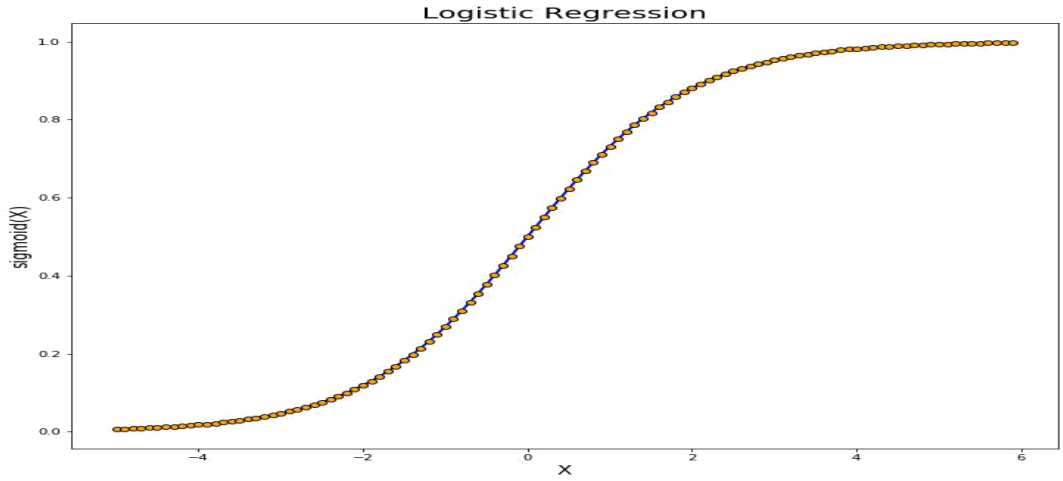


Fig 5.2.5: Logistic Regression

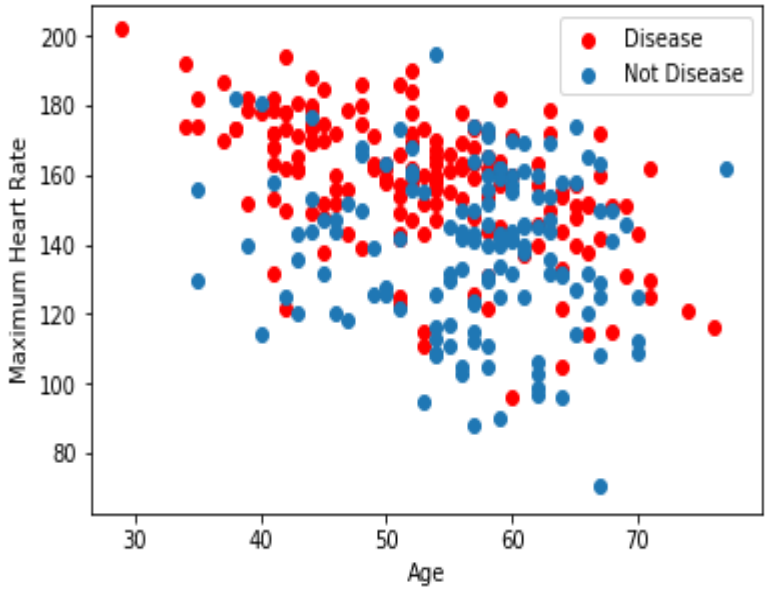


Fig 5.2.5: Output for logistic regression algorithm

5.2.6 XG BOOST 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 fedto 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.

Regularization: XG-boost has in-built L1 (Lasso Regression) and L2 (RidgeRegression) regularization which prevents the model from overfitting. That is why, XG-boost is also called regularized form of GBM (Gradient Boosting Machine).

While using Scikit Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. alpha is used for L1 regularization and lambda isused for L2 regularization.

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 Scikit Learn library, thread hyper-parameter is used for parallel processing. thread represents number of CPU cores to be used. If you want to use allthe available cores, don't mention any value for thread and the algorithm will detect automatically.Handling Missing Values: XG-boost has an in-built capability to handle missingvalues. When XG-boost encounters a missing value at a

node, it tries both the left andright hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

Cross Validation: XG-boost allows user to run a cross-validation at each iteration ofthe 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.Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative

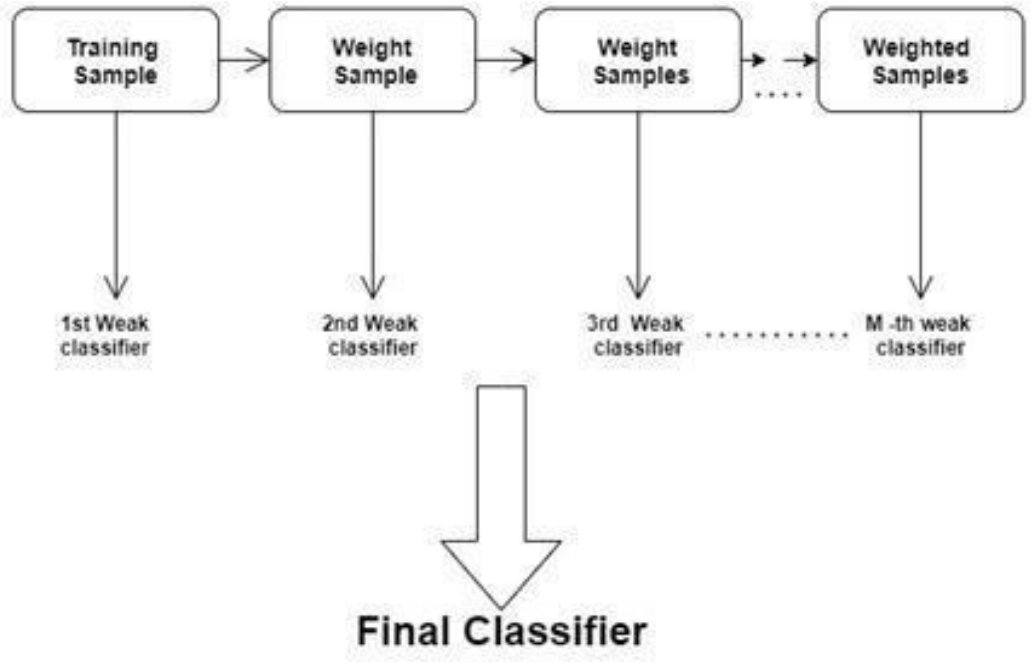


Figure : Xgboost

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. The following fig 5.1.5 shows final classifier.

Fig 5.2.6: XG -boost

5.2.7 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 vectorwhich is independently sampled. The distribution of all trees is 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(nlogn)), where M is the number of growing trees, n is the number of instances, and d is the datadimension.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 randomlyselected 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 ofcombining 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 thatthe classifier can predict accurate results rather than a guessed result.

The predictions from each tree must have very low correlations.

Algorithm Steps:

1. It works in four steps:
2. Select random samples from a given dataset.
3. Construct a Decision Tree for each sample and get a predictionresult from each Decision Tree.
4. Perform a vote for each predicted result.
5. Select the prediction result with the most votes as the finalprediction.

Advantages:

Random Forest is capable of performing both Classification andRegression 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 isnot more suitable for Regression tasks.

5.3 PACKAGES

5.3.1 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

5.3.2 Numpy

NumPy is a library for the python programming language, adding support for large, multi-

dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim with contributions from several other developers. In 2005, Travis created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

5.3.3 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is Discouraged

5.3.4 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is a library in Python predominantly used for making statistical graphics. Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

5.3.5 Pandas

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. Fast and efficient for manipulating and analyzing data. Data from different file objects can be loaded. Easy handling of missing data (represented as NAN) in floating point as well as non-floating point data Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects.

5.4 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():
 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=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:Thalassemla
# Having heart disease
labels= 'Normal', 'Fixed defect', 'Reversable 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('Thalassemla 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('Thalassemla 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:Thalassemla
# Having heart disease
labels= 'Normal', 'Fixed defect', 'Reversable 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('Thalassemla 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('Thalassemla 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:
```

5.4.1 INPUT OUTPUT

Input:

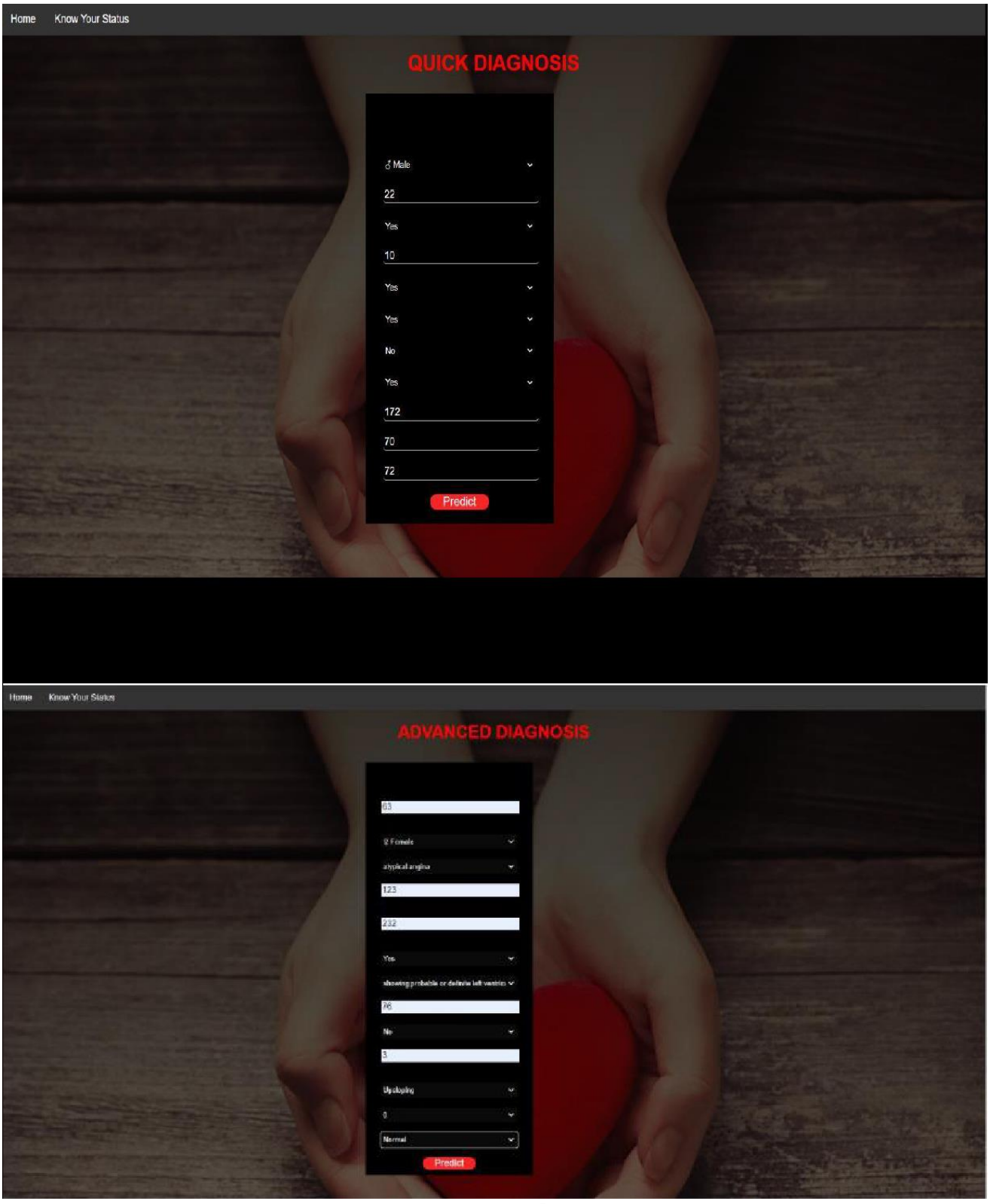
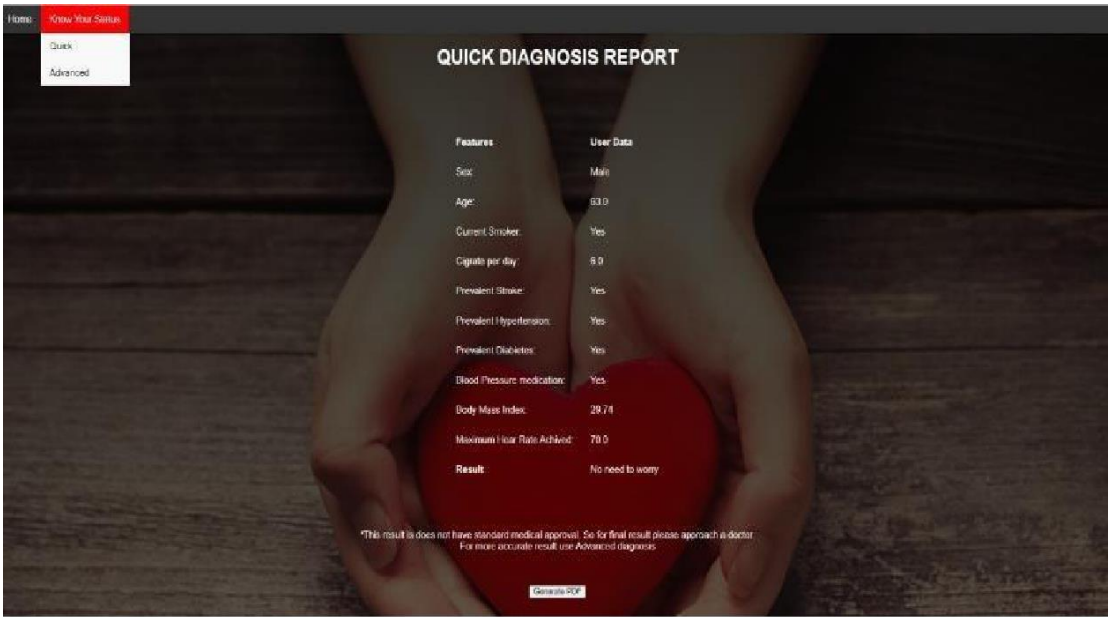


Fig 5.4.1 Input Screenshot

Output



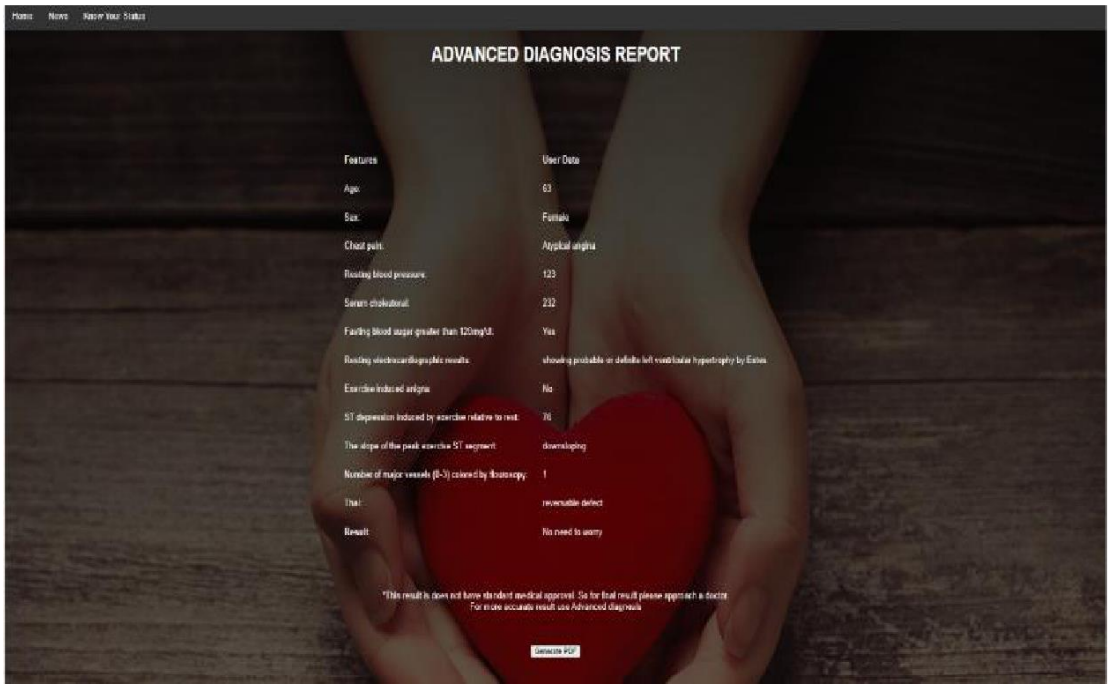
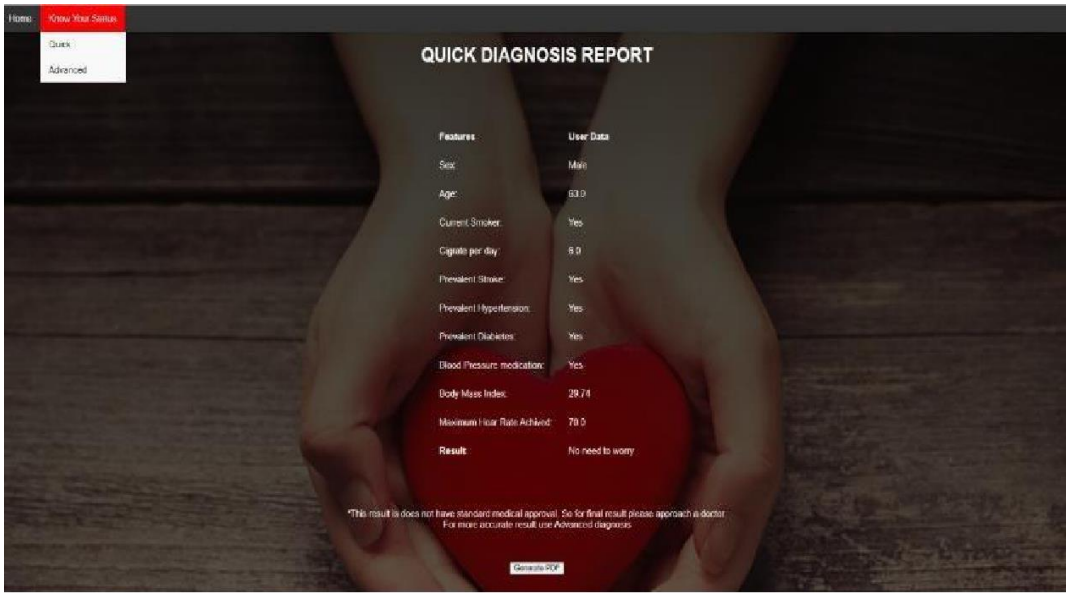


Fig 5.4.1: Output Screenshot



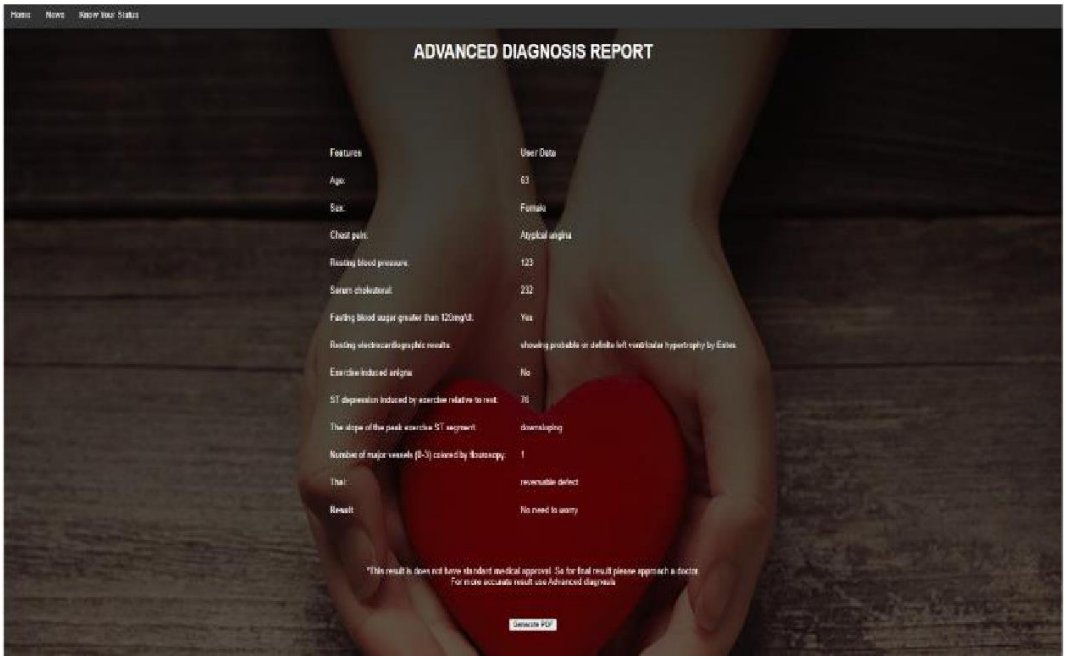


Fig 5.4.1: Web application Screenshot

CHAPTER 6

TESTING

6.1 INTRODUCTION TO TESTING

6.1.1 TESTING TECHNOLOGIES

Testing is the process of detection errors. Testing performs a quality role for assurance and forensuring the ability of software. The results of testing are used later on during maintenance also.

6.1.2 Testing Objectives

The main objective of testing is to uncover a host error, systematically, the minimum effort and timestarting formally, we can say

Test is the process of executing a program with the intent of finding an error.

A successful test is one that uncovers and yet undiscovered error.

A good test case is one that has a high probability of finding errors, if it exits.

6.1.2.1 WHITE BOX TESTING

This is unit testing method where the unit will be taken at a time and tested thoroughly at a statementlevel to find the maximum level errors. We have tested step wise every piece of code, taking care of every statement in the code.

6.1.2.2 BLACK BOX TESTING

This testing method models a single unit and checks the unit at interface and communication with othermodels rather getting into detail levels. Here the model will be treated as a black box that take input andgenerates the output. Output of given input combinations are forwarded to other models.

6.1.2.3 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software that is the model using the detailed design and the process specifications testing is done to uncover errors within the boundary of the model all models must be successful in the unit test before the start of the integration testing. In our project unit testing involves checking each future specified in the component a component performs only small part of the functionality on the system and relies on cooperating with other part of the system.

6.1.2.4 INTEGRATION TESTING

In this project integrating all the modules forms the main system when integrating all the modules we have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the services run perfectly before integration.

6.2 SAMPLE TEST CASES

Table:6.2 testCases

1	Test	Upload
2		Loading the dataset Select the path of the dataset
3.		Loaded dataset should be valid and data should not contain
4.	Test	Load the Dataset. Select the path of the
5.	Expecte	Based on the dataset retrieved from the path should beloaded into the system
6.	Actual Output	Based on the Dataset retrieved from the path should beloaded to the system
7.		Succes

6.2.1 DATASET DETAILS

- Out 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	fbs	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	

Fig 6.2.1: Dataset Attributes

Input dataset attributes

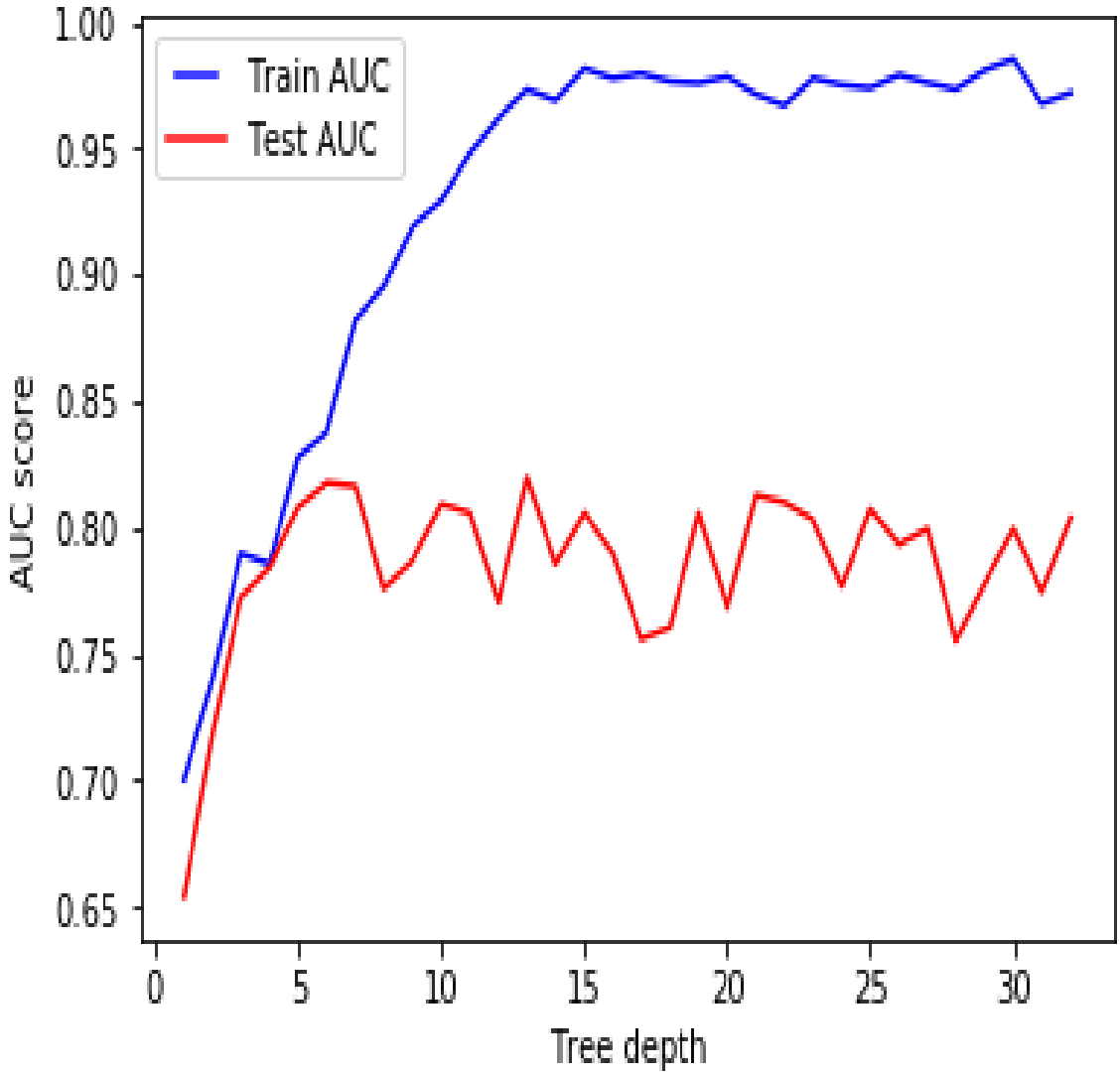
- Gender (value 1: Male; value 0 : Female)
- Chest Pain Type (value 1: typical type 1 angina, value 2: typical type angina, value3: non-angina pain; value 4: asymptomatic)
- Fasting Blood Sugar (value 1: > 120 mg/dl; value 0:< 120 mg/dl)
- Exang – exercise induced angina (value 1: yes; value 0: no)
- 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

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 electrocardiographic 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	Old peak	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

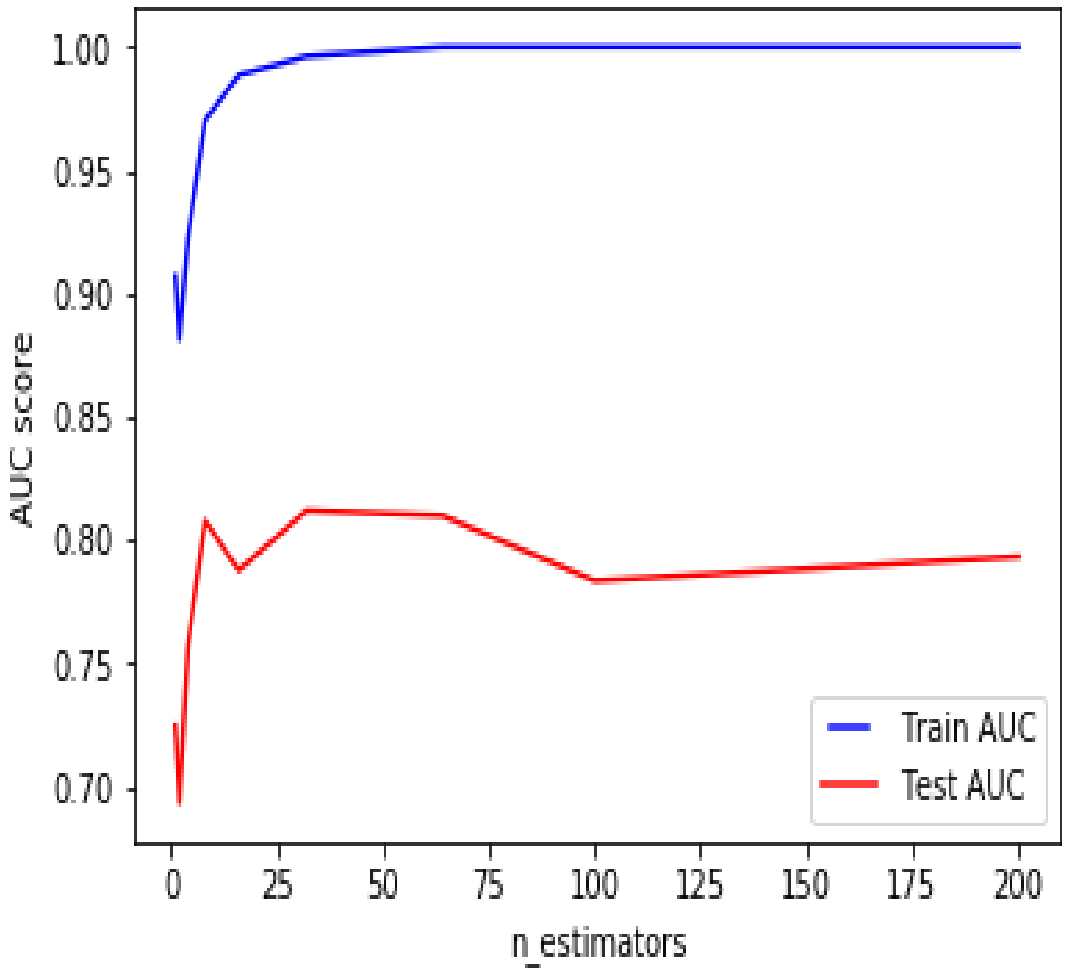
TABLE 2: Attributes of the dataset

6.3 PERFORMANCE ANALYSIS

In the below graph it represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data. We fit each decision tree with depths ranging from 1 to 32 and plot the training and test errors.



The below graph represents the number of trees in the forest. Usually the higher the number of trees the better to learn the data.



PERFORMANCE MEASURES

```
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
```

• The highest accuracy is given by **Random Forest**

CHAPTER 7

CONCLUSION AND FUTURE WORK

Heart diseases are a major killer in India and throughout the world, applicationof 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. Thenumber 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 thedataset 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 inthe 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%.

.

REFERENCES

[1] Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. International Journal of ComputerApplications, 17(8), 43-8

[2] Dangare C S & Apte S S (2012). Improved study of heart disease prediction system using data mining classification techniques. International Journal of ComputerApplications, 47(10), 44-8.

[3] Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. InternationalJournal of Computer Science and Information Technologies, 6(1), 637-9.

[4] Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decisionsupport framework for intelligent heart disease diagnosis. In International Conferenceon Information Society (I-Society 2014) (pp. 259-64). IEEE. ICCRDA 2020 IOP Conf. Series: Materials Science and Engineering 1022 (2021) 012072 IOP Publishing doi:10.1088/1757-899X/1022/1/012072 9

[5] Jee S H, Jang Y, Oh D J, Oh B H, Lee S H, Park S W & Yun Y D (2014). A coronary heart disease prediction model: the Korean Heart Study. BMJ open, 4(5),e005025.

[6] Ganna A, Magnusson P K, Pedersen N L, de Faire U, Reilly M, Ärnlöv J &Ingelsson E (2013). Multilocus genetic risk scores for coronary heart disease prediction. Arteriosclerosis, thrombosis, and vascular biology, 33(9), 2267-72.

[7] Jabbar M A, Deekshatulu B L & Chandra P (2013, March). Heart disease prediction using lazy associative classification. In 2013 International Mutli- Conference on Automation, Computing, Communication, Control and CompressedSensing (iMac4s) (pp. 40- 6). IEEE.

[8] Brown N, Young T, Gray D, Skene A M & Hampton J R (1997). Inpatient deathsfrom acute myocardial infarction, 1982-92: analysis of data in the Nottingham heart attack register. BMJ, 315(7101), 159-64.

[9] Folsom A R, Prineas R J, Kaye S A & Soler J T (1989). Body fat distribution and self-reported prevalence of hypertension, heart attack, and other heart disease in older women. International journal of epidemiologyy, 18(2), 361-7.

[10] Chen A H, Huang S Y, Hong P S, Cheng C H & Lin E J (2011, September). HDPS: Heart disease prediction system. In 2011 Computing in Cardiology (pp. 557-60). IEE