HEART DISEASE PREDICTION USING MACHINE LEARNING

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

ANYAPU KARTHIK (RA2111003011739)
DACHEPALLI DILEEP (RA2111003011783)
K YAMINI (RA2111003011735)

Under the guidance of

Mrs. Ranjani M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"HEART DISEASE PREDICTION USING MACHINE LEARNING"** is the bona fide work of **ANYAPU KARTHIK(RA2111003011739)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE 07/04/24

Mrs. Ranjani M

Assistant Professor

Department of Computing Technologies Kattankulathur 603 203

M. Pushpalatha

SIGNATURE

Dr. M. Pushpalatha

Head Department of C-Tech

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"HEART DISEASE PREDICTION USING MACHINE LEARNING"** is the bona fide work of **ANYAPU KARTHIK(RA2111003011739)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. Ranjani M

Assistant Professor

SIGNATURE

Dr. M. Pushpalatha

Head Department of C-Tech

TABLE OF CONTENTS

List of Diagrams

List of Tables

INTRODUCTION

The heart is a kind of muscular organ which pumps blood into the body and is the central part of the body's cardiovascular system which also contains lungs. Cardiovascular system also comprises a network of blood vessels, for example, veins, arteries, and capillaries. These blood vessels deliver blood all over the body. Abnormalities in normal blood flow from the heart cause several types of heart diseases which are commonly known as cardiovascular diseases (CVD). Heart diseases are the main reasons for death worldwide. According to the survey of the World Health Organization (WHO), 17.5 million total global deaths occur because of heart attacks and strokes. More than 75% of deaths from cardiovascular diseases occur mostly in middle-income and low-income countries. Also, 80% of the deaths that occur due to CVDs are because of stroke and heart attack . Therefore, prediction of cardiac abnormalities at the early stage and tools for the prediction of heart diseases can save a lot of life and help doctors to design an effective treatment plan which ultimately reduces the mortality rate due to cardiovascular diseases.

Due to the development of advance healthcare systems, lots of patient data are nowadays available (i.e. Big Data in Electronic Health Record System) which can be used for designing predictive models for Cardiovascular diseases. Data mining or machine learning is a discovery method for analyzing big data from an assorted perspective and encapsulating it into useful information. "Data Mining is a non-trivial extraction of implicit, previously unknown and potentially useful information about data". Nowadays, a huge amount of data pertaining to disease diagnosis, patients etc. are generated by healthcare industries. Data mining provides a number of techniques which discover hidden patterns or similarities from data.

Therefore, in this paper, a machine learning algorithm is proposed for the implementation of a heart disease prediction system which was validated on two open access heart disease prediction datasets. Data mining is the computer based process of extracting useful information from enormous sets of databases. Data mining is most helpful in an explorative analysis because of nontrivial information from large volumes of evidence .Medical data mining has great potential for exploring the cryptic patterns in the data sets of the clinical domain.

LITERATURE SURVEY

Machine Learning techniques are used to analyze and predict the medical data information resources. Diagnosis of heart disease is a significant and tedious task in medicine. The term Heart disease encompasses the various diseases that affect the heart. The exposure of heart disease from various factors or symptom is an issue which is not complimentary from false presumptions often accompanied by unpredictable effects. The data classification is based on Supervised Machine Learning algorithm which results in better accuracy. Here we are using the Random Forest as the training algorithm to train the heart disease dataset and to predict the heart disease. The results showed that the medicinal prescription and designed prediction system is capable of prophesying the heart attack successfully. Machine Learning techniques are used to indicate the early mortality by analyzing the heart disease patients and their clinical records (Richards, G. et al., 2001). (Sung, S.F. et al., 2015) have brought about the two Machine Learning techniques, k-nearest neighbor model and existing multi linear regression to predict the stroke severity index (SSI) of the patients. Their study show that k-nearest neighbor performed better than Multi Linear Regression model. (Arslan, A. K. et al., 2016) have suggested various Machine Learning techniques such as support vector machine (SVM), penalized logistic regression (PLR) to predict the heart stroke. Their results show that SVM produced the best performance in prediction when compared to other models.Boshra Brahmi et al, [20] developed different Machine Learning techniques to evaluate the prediction and diagnosis of heart disease. The main objective is to evaluate the different classification techniques such as J48, Decision Tree, KNN and Naïve Bayes. After this, evaluating some performance in measures of accuracy, precision, sensitivity, specificity are evaluated .

SYSTEM ANAYLSIS

1.1 EXISTING SYSTEM

Clinical decisions are often made based on doctors' intuition and experience rather than on the knowledge rich data hidden in the database. This practice leads to unwanted biases, errors and excessive medical costs which affects the quality of service provided to patients. There are many ways that a medical misdiagnosis can present itself. Whether a doctor is at fault, or hospital staff, a misdiagnosis of a serious illness can have very extreme and harmful effects. The National Patient Safety Foundation cites that 42% of medical patients feel they have had experienced a medical error or missed diagnosis. Patient safety is sometimes negligently given the back seat for other concerns, such as the cost of medical tests, drugs, and operations. Medical Misdiagnoses are a serious risk to our healthcare profession. If they continue, then people will fear going to the hospital for treatment. We can put an end to medical misdiagnosis by informing the public and filing claims and suits against the medical practitioners at fault.

Disadvantages:

- Prediction is not possible at early stages.
- In the Existing system, practical use of collected data is time consuming.
- Any faults occurred by the doctor or hospital staff n predicting would lead to fatal incidents.
- Highly expensive and laborious process needs to be performed before treating the patient to find out if he/she has any chances to get heart disease in future.

1.2 PROPOSED SYSTEM

This section depicts the overview of the proposed system and illustrates all of the components, techniques and tools are used for developing the entire system. To develop an intelligent and user-friendly heart disease prediction system, an efficient software tool is needed in order to train huge datasets and compare multiple machine learning algorithms. After choosing the robust algorithm with best accuracy and performance measures, it will be implemented on the development of the smart phone-based application for detecting and predicting heart disease risk level. Hardware components like Arduino/Raspberry Pi, different biomedical sensors, display monitor, buzzer etc. are needed to build the continuous patient monitoring system.

1.3 ALGORITHMS

1.3.1 Logistic Regression

A popular statistical technique to predict binomial outcomes (y = 0 or 1) is Logistic Regression. Logistic regression predicts categorical outcomes (binomial / multinomial values of y). The predictions of Logistic Regression (henceforth, LogR in this article) are in the form of probabilities of an event occurring, i.e. the probability of y=1, given certain values of input variables x. Thus, the results of LogR range between 0-1.

LogR models the data points using the standard logistic function, which is an S-shaped curve also called as sigmoid curve and is given by the equation:

$$\frac{1}{1+e^{-x}} = \frac{e^{x}}{1+e^{x}}$$

Logistic Regression Assumptions:

- Logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other·
- Logistic regression requires quite large sample sizes.
- Even though, logistic (**logit**) regression is frequently used for binary variables (2 classes), it can be used for categorical dependent variables with more than 2 classes.
- In this case it's called Multinomial Logistic Regression.

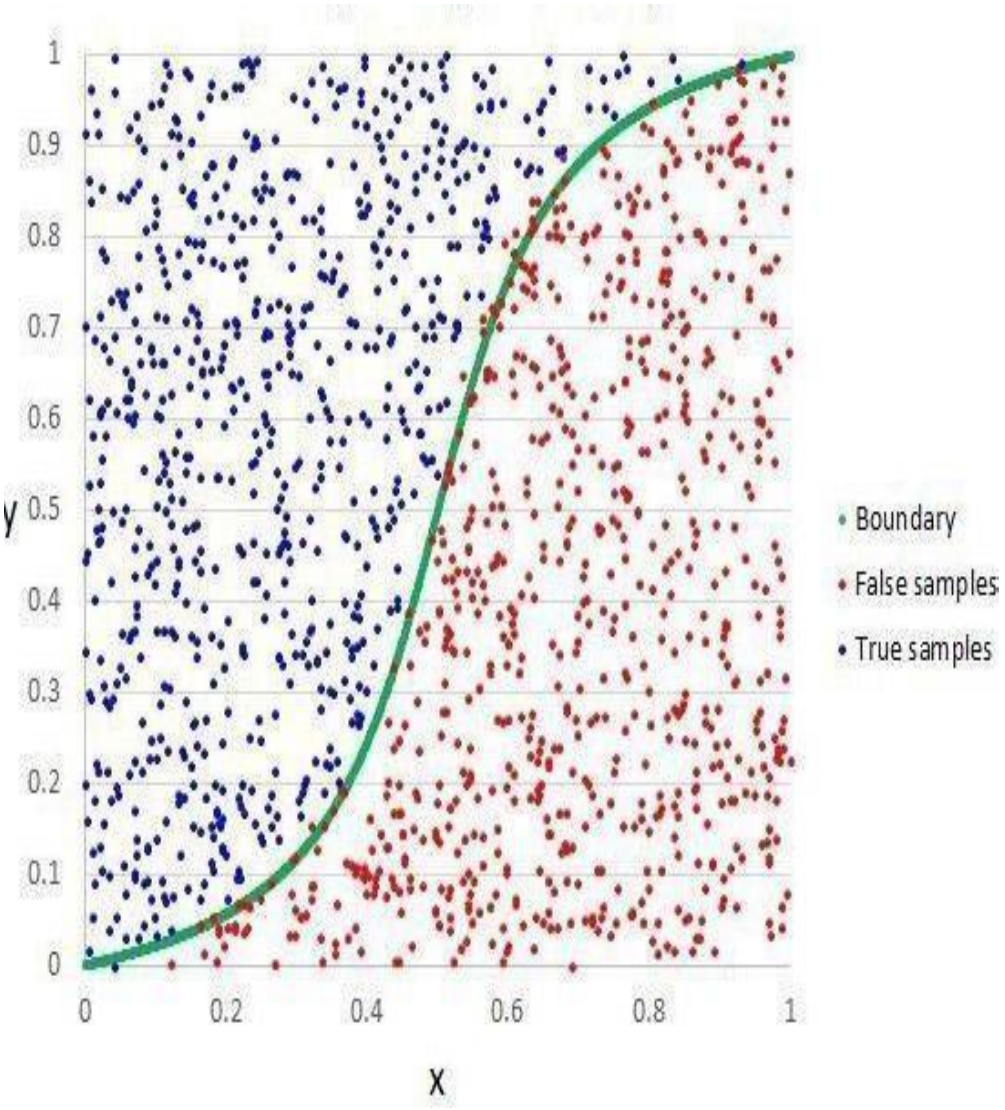

Fig 3.1: logistic regression

1.3.2 Random Forest

Random forest is a supervised learning algorithm which is used for both classification as well as regression .But however ,it is mainly used for classification problems .As we know that a forest is made up of trees and more trees means more robust forest .

Similarly ,random forest creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting .It is ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result .

Working of Random Forest with the help of following steps:

- First ,start with the selection of random samples from a given dataset.
- Next ,this algorithm will construct a decision tree for every sample .Then it will get the prediction result from every decision tree .
- In this step, voting will be performed for every predicted result.
- At last ,select the most voted prediction results as the final prediction result.

The following diagram will illustrates its working-

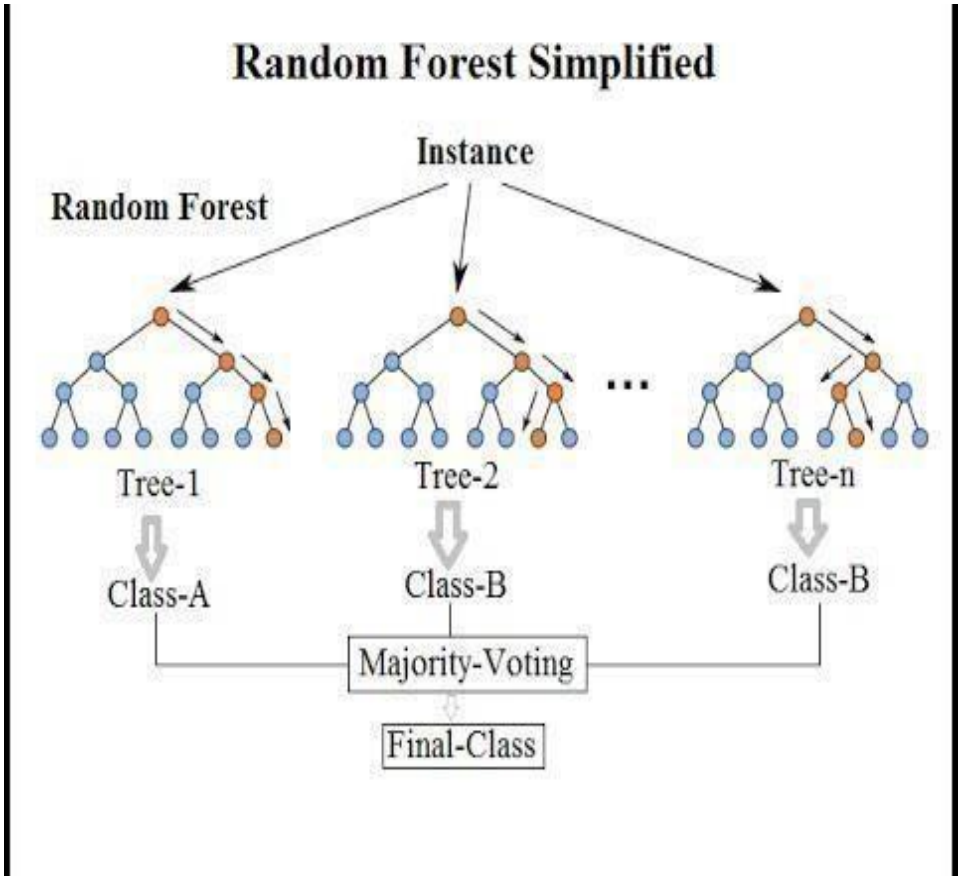

Fig 3.2: Random Forest

1.4 FEASIBILITY STUDY

A Feasibility Study is a preliminary study undertaken before the real work of a project starts to ascertain the likely hood of the projects success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternative.

1.4.1 Economic Feasibility:

It is defined as the process of assessing the benefits and costs associated with the development of project. A proposed system, which is both operationally and technically

feasible, must be a good investment for the organization. With the proposed system the users are greatly benefited as the users can be able to detect the fake news from the real news and are aware of most real and most fake news published in the recent years. This proposed system does not need any additional software and high system configuration. Hence the proposed system is economically feasible.

1.4.2 Technical Feasibility:

The technical feasibility infers whether the proposed system can be developed considering the technical issues like availability of the necessary technology, technical capacity, adequate response and extensibility. The project is decided to build using Python. Jupyter Note Book is designed for use in distributed environment of the internet and for the professional programmer it is easy to learn and use effectively. As the developing organization has all the resources available to build the system therefore the proposed system is technically feasible.

1.4.3 Operational Feasibility:

Operational feasibility is defined as the process of assessing the degree to which a proposed system solves business problems or takes advantage of business opportunities. The system is self-explanatory and doesn't need any extra sophisticated training. The system has built-in methods and classes which are required to produce the result. The application can be handled very easily with a novice user. The overall time that a user needs to get trained is 14 less than one hour. As the software that is used for developing this application is very economical and is readily available in the market. Therefore the proposed system is operationally feasible.

1.5 EFFORT, DURATION AND COST ESTIMATION USING COCOMO MODEL

The Cocomo (Constructive Cost Model) model is the most complete and thoroughly documented model used in effort estimation. The model provides detailed formulas for determining the development time schedule, overall development effort, and effort breakdown by phase and activity as well as maintenance effort.

COCOMO estimates the effort in person months of direct labor. The primary effort factor is the number of source lines of code (SLOC) expressed in thousands of delivered source instructions (KDSI).The model is developed in three versions of different level of detail basic, intermediate, and detailed. The overall modeling process takes into account three classes of systems.

1. **Embedded:** This class of system is characterized by tight constraints, changing environment, and unfamiliar surroundings. Projects of the embedded type are model to the company and usually exhibit temporal constraints.

2. **Organic**: This category encompasses all systems that are small relative to project size and team size, and have a stable environment, familiar surroundings and relaxed interfaces. These are simple business systems, data processing systems, and small software libraries.

3. **Semidetached**: The software systems falling under this category are a mix of those of organic and embedded in nature.

Some examples of software of this class are operating systems, database management system, and inventory management systems.

For basic COCOMO Effort = a*(KLOC) b

Type =c*(effort)d

For Intermediate and Detailed COCOMO Effort = a * (KLOC) b* EAF (EAF = product of cost drivers)

Type of Product	A	B	C	D
Organic	2.4	1.02	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 3.1: Organic, Semidetached and Embedded system values

Intermediate COCOMO model is a refinement of the basic model, which comes in the function of 15 attributes of the product. For each of the attributes the user of the model has to provide a rating using the following six point scale.

VL(Very Low) HI(High)

LO (Low) VH (VeryHigh)

NM(Nominal) XH (ExtraHigh)

The list of attributes is composed of several features of the software and includes product, computer, personal and project attributes as follows.

SOFTWARE REQUIREMENTS SPECIFICATION

4.1 INTRODUCTION TO REQUIREMENT SPECIFICATION

Software Engineering by James F Peters & WitoldPedrycz Head First Java by Ka. A Software Requirements Specification (SRS) is a description of particular software product, program or set of programs that performs a set of functions in a target environment (IEEE Std. 830-1993).

a. Purpose

The purpose of software requirements specification specifies the intentions and intended audience of the SRS.

b. Scope

The scope of the SRS identifies the software product to be produced, the capabilities, application, relevant objects etc. We are proposed to implement Passive Aggressive Algorithm which takes the test and trained data set from the

c. Definitions, Acronyms and Abbreviations Software Requirements Specification

It's a description of a particular software product, program or set of programs that performs a set of function in target environment.

d. References

IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specifications thy Sierra and Bert Bates.

e. Overview

The SRS contains the details of process, DFD's, functions of the product, user characteristics. The non-functional requirements if any are also specified.

f. Overall description

The main functions associated with the product are described in this section of SRS. The characteristics of a user of this product are indicated. The assumptions in this section result from interaction with the project stakeholders.

4.2 REQUIREMENT ANALYSIS

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.) Under requirement specification, the focus is on specifying what has

been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document.

Producing the SRS document is the basic goal of this phase. The purpose of the Software Requirement Specification is to reduce the communication gap between theclients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

4.2.1 Product Perspective:

The application is developed in such a way that any future enhancement can be easily implementable. The project is developed in such a way that it requires minimal maintenance. The software used are open source and easy to install. The application developed should be easy to install and use. This is an independent application which can be easily run on to any system which has Python installed and Jupiter Notebook.

4.2.2 Product Features:

The application is developed in a way that 'Heart disease' accuracy is predicted using Random Forest. The dataset is taken from https://www.datacamp.com/community/tutorials/scikit-learn-credit-card. We can compare the accuracy for the implemented algorithms. User characteristics Application is developed in such a way that its users are v Easy to use v Error free 20 v Minimal training or no training v Patient regular monitor Assumption & Dependencies It is considered that the dataset taken fulfils all the requirements.

4.2.3 Domain Requirements:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also be the bases for validating the final Heart disease system. Any changes made to the requirements in the future will have to go through a formal change approval process. User Requirements User can decide on the prediction accuracy to decide on which algorithm can be used in real-time predictions. Non Functional Requirements ÿ Dataset collected should be in the CSV format ÿ .The column values should be numerical values ÿ Training set and test set are stored as CSV files ÿ Error rates can be calculated for prediction algorithms product.

4.2.4 Requirements Efficiency:

Less time for predicting the Heart Disease Reliability: Maturity, fault tolerance and recoverability. Portability: can the software easily be transferred to another environment, including install ability.

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The below figure shows the process flow diagram or proposed work. First we collected the Cleveland Heart Disease Database from UCI website then pre-processed the dataset and select 16 important features.

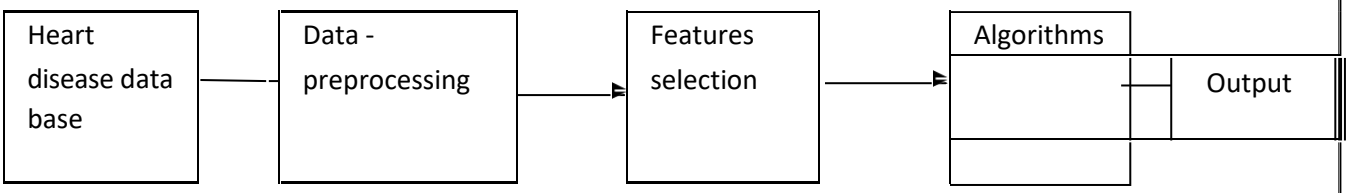

Fig 5.1: System Architecture

For feature selection we used Recursive feature Elimination Algorithm using Chi2 method and get 16 top features. After that applied ANN and Logistic algorithm individually and compute the accuracy. Finally, we used proposed Ensemble Voting method and compute best method for diagnosis of heart disease.

5.2 MODULES

The entire work of this project is divided into 4 modules.

They are:

a. Data Pre-processing
b. Feature
c. Classification
d. Prediction

a. Data Pre-processing:

This file contains all the pre-processing functions needed to process all input documents and texts. First we read the train, test and validation data files then performed some preprocessing like tokenizing, stemming etc. There are some exploratory data analysis is performed like response variable distribution and data quality checks like null or missing values etc.

b. Feature:

Extraction In this file we have performed feature extraction and selection methods from sci-kit learn python libraries. For feature selection, we have used methods like simple bag-of-words and n-grams and then term frequency like tf-tdf weighting. We have also used

word2vec and POS tagging to extract the features, though POS tagging and word2vec has not been used at this point in the project.

c. Classification:

Here we have built all the classifiers for the breast cancer diseases detection. The extracted features are fed into different classifiers. We have used Naive-bayes, Logistic Regression, Linear SVM, Stochastic gradient decent and Random forest classifiers from sklearn. Each of the extracted features was used in all of the classifiers. Once fitting the model, we compared the f1 score and checked the confusion matrix.

After fitting all the classifiers, 2 best performing models were selected as candidate models for heart diseases classification. We have performed parameter tuning by implementing GridSearchCV methods on these candidate models and chosen best performing parameters for these classifier.

Finally selected model was used for heart disease detection with the probability of truth. In Addition to this, we have also extracted the top 50 features from our term-frequency tfidf Vectorizer to see what words are most and important in each of the classes.

We have also used Precision-Recall and learning curves to see how training and test set performs when we increase the amount of data in our classifiers.

d. Prediction:

Our finally selected and best performing classifier was algorithm which was then saved on disk with name final_model.sav. Once you close this repository, this model will be copied to user's machine and will be used by prediction.py file to classify the Heart diseases . It takes a news article as input from user then model is used for final classification output that is shown to user along with probability of truth.

5.3 DATA FLOW DIAGRAM

The data flow diagram (DFD) is one of the most important tools used by system analysis. Data flow diagrams are made up of number of symbols, which represents system components. Most data flow modeling methods use four kinds of symbols: Processes, Data stores, Data flows and external entities.

These symbols are used to represent four kinds of system components. Circles in DFD represent processes. Data Flow represented by a thin line in the DFD and each data store has a unique name and square or rectangle represents external entities.

5.4 UML DIAGRAMS

5.4.1 Use-Case Diagram

A use case diagram is a diagram that shows a set of use cases and actors and their relationships. A use case diagram is just a special kind of diagram and shares the same common properties as do all other diagrams, i.e a name and graphical contents that are a projection into a model. What distinguishes a use case diagram from all other kinds of diagrams is its particular content.

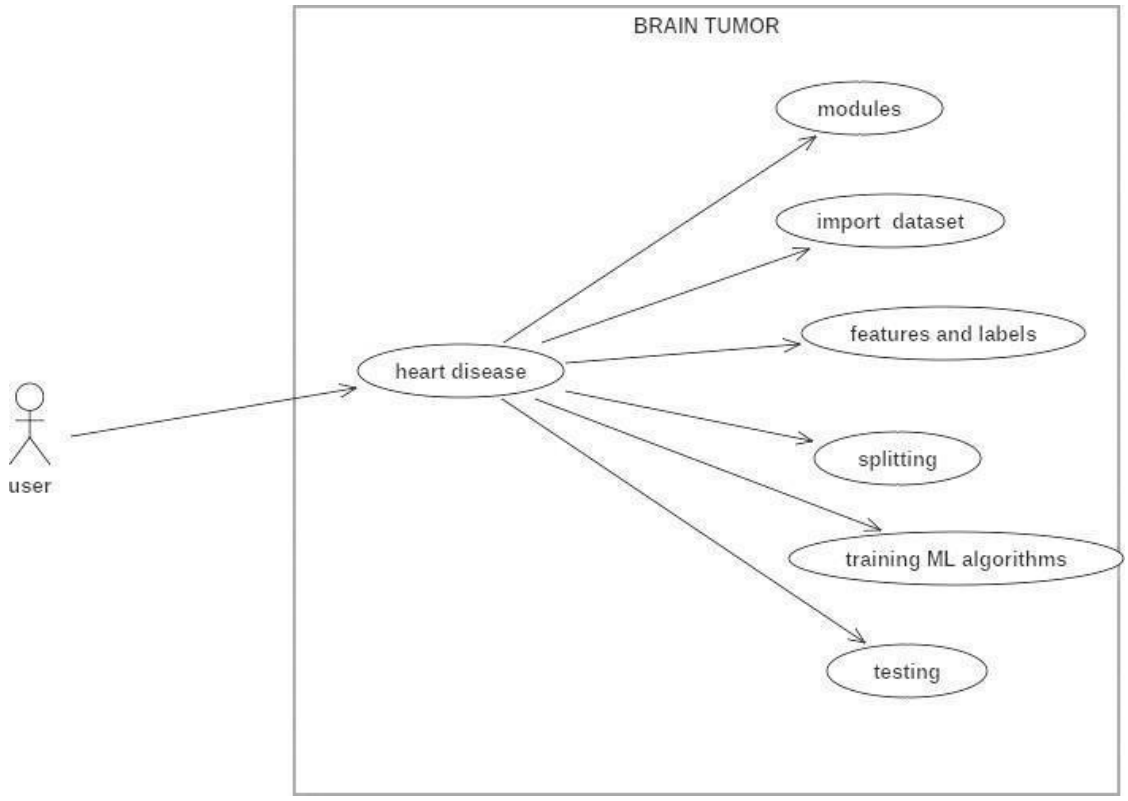

Fig 5.4: Use case Diagram

5.4.2 Activity Diagram

An activity diagram shows the flow from activity to activity. An activity is an ongoing non-atomic execution within a state machine. An activity diagram is basically a projection of the elements found in an activity graph, a special case of a state machine in which all or most states are activity states and in which all or most transitions are triggered by completion of activities in the source.

5.4.4 Class diagram

A Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. It provides a basic notation for other structure diagrams prescribed by UML. It is helpful for developers and other team members too.

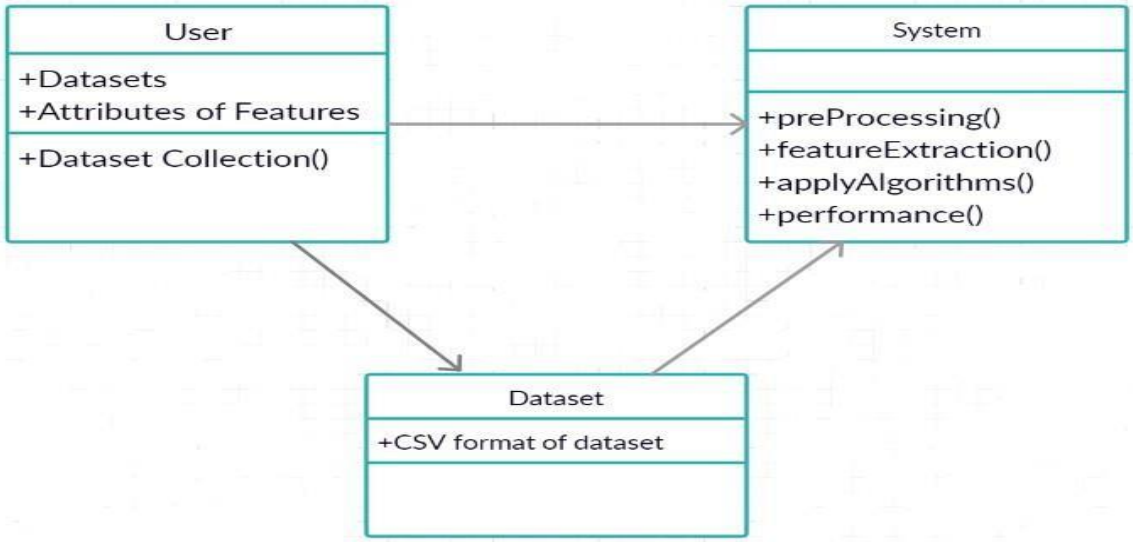

Fig 5.7 Class Diagram

IMPLEMENTATION

STEPS FOR IMPLEMENTATION

1. Install the required packages for building the 'Passive Aggressive Classifier'.
2. Load the libraries into the workspace from the packages.
3. Read the input data set.
4. Normalize the given input dataset.
5. Divide this normalized data into two parts:
    a. Train data
    b. Test data (Note: 80% of Normalized data is used as Train data, 20% of the Normalized data is used as Test data.)

6.2 CODING

Sample code:

```
#importing modules
import pandas as pd
import numpy as np
import seaborn as sns
#Reading dataset(heart.csv)
df=pd.read_csv('heart.csv')

#printing startting instances of data set
df.head()

#visualizing how many persons have heartdisease based on gender using heartdisease dataset
sns.countplot(x='sex',hue='target',data=df)

#visualizing how many persons have heart disease based on type of chest pain
sns.countplot(x='cp',hue='target',data=df)

#preprocessing data i.e. finding null values;
df.isna().sum()
#if we have null values we drop them using following command
#df.dropna()
```

```
x=df[['age','sex','trestbps','chol','cp']]
y=df['target']
x.shape,y.shape

# spliting data for trainig and testing
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3, random_state=100)

#training the data and predicting accuracy using logistic regression
from sklearn.linear_model import LogisticRegression
logreg =LogisticRegression(class_weight='balanced')
sd=logreg.fit(X_train,y_train)
sd.score(X_train,y_train)
#a = sd.predict(X_test)
#d_log = pd.DataFrame(data=a)
#d_log.rename(index=str, columns={0:"y_log"})
logreg.score(X_test,y_test)
#d_log

#testing against new samples
a=np.array([63,1,3,145,233])
b=a.reshape(1,5)
df=pd.DataFrame(b)
df. shape

#predicting whether a person have heart disease or not against new sample
sd.predict(df)
#creating pickle module
import pickle
pickle.dump(sd,open('heart1.pk','wb'))
#training the data and predicting accuracy using Random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier().fit(X_train,y_train)
rf.score(X_train,y_train)
```

SYSTEM TESTING

7.1 WHITE BOX TESTING

It is testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers. It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

a. What do you verify in White Box Testing?

1. White box testing involves the testing of the software code for the following:
2. Internal security holes
3. Broken or poorly structured paths in the coding processes
4. The flow of specific inputs through the code
5. Expected output
6. The functionality of conditional loops
7. Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of white box testing is to verify a working flow for an application. It

involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

b. How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

Step 1: Understand The Source Code

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2: Create Test Cases And Execute

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

c. White Box Testing Techniques

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product. There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques

1. Statement Coverage:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

7.2 Black Box Testing

a. What Is Black Box Testing

It is defined as a testing technique in which functionality of the Application under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In 'BlackBox' Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

The Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

b. How to do Blackbox Testing

- Here are the generic steps followed to carry out any type of Black Box Testing.
- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

c. Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones :-

1. **Functional testing -** This black box testing type is related to the functional requirements of a system; it is done by software testers.
2. **Non-functional testing -** This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.

3. **Regression testing -** Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

d. Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

- For Functional/ Regression Tests you can use - QTP, Selenium
- For Non-Functional Tests, you can use - LoadRunner, JMeter.

e. Black Box Testing Techniques

Following are the prominent Test strategy amongst the many used in Black box Testing.

1. **Equivalence Testing**
2. **Boundary Value Testing**
3. **Decision Table Testing**

1. Equivalence Class Testing:

It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.

2. Boundary Value Testing:

Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.

3. Decision Table Testing:

A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

f. Black Box Testing and Software Development Life Cycle (SDLC)

Black box testing has its own life cycle called Software Testing Life Cycle (STLC) and it is relative to every stage of Software Development Life Cycle of Software Engineering.

1. **Requirement** - This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.

2. **Test Planning & Analysis**

Testing Types applicable to the project are determined. A Test plan is created which determines possible project risks and their mitigation.

3. **Design**

In this stage Test cases/scripts are created on the basis of software requirement documents

4. **Test Execution**

In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

CONCLUSION

In this project, we introduce about the heart disease prediction system with different classifier techniques for the prediction of heart disease. The techniques are Random Forest and Logistic Regression: we have analyzed that the Random Forest has better accuracy as compared to Logistic Regression. Our purpose is to improve the performance of the Random Forest by removing unnecessary and irrelevant attributes from the dataset and only picking those that are most informative for the classification task.

FUTURE SCOPE

As illustrated before the system can be used as a clinical assistant for any clinicians. The disease prediction through the risk factors can be hosted online and hence any internet users can access the system through a web browser and understand the risk of heart disease. The proposed model can be implemented for any real time application .Using the proposed model other type of heart disease also can be determined. Different heart diseases as rheumatic heart disease, hypertensive heart disease, ischemic heart disease, cardiovascular disease and inflammatory heart disease can be identified.

Other health care systems can be formulated using this proposed model in order to identify the diseases in the early stage. The proposed model requires an efficient processor with good memory configuration to implement it in real time. The proposed model has wide area of application like grid computing, cloud computing, robotic modeling, etc. To increase the performance of our classifier in future, we will work on ensembling two algorithms called Random Forest and Adaboost. By ensembling these two algorithms we will achieve high performance.

REFERENCES

[1] P .K. Anooj, ―Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules‖; Journal of King Saud University – Computer and Information Sciences (2012) 24, 27–40. Computer Science & Information Technology (CS & IT) 59

[2] Nidhi Bhatla, Kiran Jyoti"An Analysis of Heart Disease Prediction using Different Data Mining Techniques".International Journal of Engineering Research & Technology

[3] Jyoti Soni Ujma Ansari Dipesh Sharma, Sunita Soni. "Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction".

[4] Chaitrali S. Dangare Sulabha S. Apte, Improved Study of Heart Disease Prediction System using Data Mining Classification Techniques" International Journal of Computer Applications (0975 – 888)

[5] Dane Bertram, Amy Voida, Saul Greenberg, Robert Walker, "Communication, Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams".

[6] M. Anbarasi, E. Anupriya, N.Ch.S.N.Iyengar, ―Enhanced Prediction of Heart Disease with Feature Subset Selection using Genetic Algorithm‖; International Journal of Engineering Science and Technology, Vol. 2(10), 2010.

[7] Ankita Dewan, Meghna Sharma," Prediction of Heart Disease Using a Hybrid Technique in Data Mining Classification", 2nd International Conference on Computing for Sustainable Global Development IEEE 2015 pp 704-706. [2].

[8] R. Alizadehsani, J. Habibi, B. Bahadorian, H. Mashayekhi, A. Ghandeharioun, R. Boghrati, et al., "Diagnosis of coronary arteries stenosis using data mining," J Med Signals Sens, vol. 2, pp. 153-9, Jul 2012.

[9] M Akhil Jabbar, BL Deekshatulu, Priti Chandra," Heart disease classification using nearest neighbor classifier with feature subset selection", Anale. Seria Informatica, 11, 2013