A Mini Project Report on

PREDICTION OF HEART DISEASES USING MACHINE LEARNING

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

EMPATI BHAVANI(17BD1A054D)

GORANTA LAVANYALAHARI(17BD1A054H)

PODDUTURI SANJANA(17BD1A055C)

SAI MOHANA HARINISRI VISSAMSETTY(17BD1A055K)

Under the guidance of

HALEEMA BUSHRA
Assistant Professor
Department of CSE

Department of Computer Science and Engineering

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

Approved by AICTE, Affiliated to JNTUH

3-5-1206, Narayanaguda, Hyderabad – 500029

2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **PREDICTION OF HEART DISEASES USING ML** being submitted by

Empati Bhavani	**(17BD1A054D)**
Goranta Lavanyalahari	**(17BD1A054H)**
Podduturi Sanjana	**(17BD1A055C)**
Sai Mohana Harinisri Vissamsetty	**(17BD1A055K)**

In partial fulfillment for the award of the degree of **Bachelor of Technology** in **Computer Science and Technology** affiliated to the **Jawaharlal Nehru Technological University, Hyderabad.**

Internal Guide **Head of the Department**

(Haleema Bushra) **(Dr. S. Padmaja)**

Submitted for Viva Voce Examination held on _____

External Examiner

Vision of KMIT

Producing quality graduates trained in the latest technologies and related tools and striving to make India a world leader in software and hardware products and services. To achieve academic excellence by imparting in depth knowledge to the students, facilitating research activities and catering to the fast growing and ever- changing industrial demands and societal needs.

Mission of KMIT

- To provide a learning environment that helps students to enhance problem solving skills, be successful in their professional lives and to prepare students to be lifelong learners through multi model platforms and educating them about their professional, and ethical responsibilities
- To establish Industry Institute Interaction to make students ready for the industry.
- To provide exposure to students to the latest tools and technologies hardware and software. the area of hardware and software
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce in the students a spirit of nationalism which will enable the student to develop and understand India's problems and to encourage them to come up with effective solutions for the same.
- To support the faculty in their endeavours to accelerate their learning curve in order to continue to deliver excellent service to students.

Vision & Mission of CSE

Vision of the CSE

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the CSE

- To provide faculty with state of the art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefits the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

PROGRAM OUTCOMES (POs)

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. Design/development of solutions: Design solutions for complex engineering problem and design system component or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to societal, health, safety. Legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyze the common business functions to design and develop appropriate Information Technology solutions for social upliftment.

PSO2: Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

PEO2: Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

PEO3: Graduates will engage in life-long learning and professional development by rapidly adapting changing work environment.

PEO4: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

PROJECT OUTCOMES

P1: It makes us able to predict whether the person has affected with heart disease or not.

P2: It makes us to identify the absolute stage of the heart disease of a patient.

P3: Our project requires only few number of attributes to identify the disease correctly.

P4: Our project has the highest accuracy to predict the disease when compared to laboratory tests .

L – LOW
M – MEDIUM
H – HIGH

PROJECT OUTCOMES MAPPING PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1	M				M	L		L	M	M	M	M
P2	M				M	L		L	M	M	M	M
P3	L	M						L	M	M	M	
P4	M	L	M	L	M			L	M	M	M	

PROJECT OUTCOMES MAPPING PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1	M	M
P2	M	M
P3		
P4		M

PROJECT OUTCOMES MAPPING PROGRAM EDUCATIONAL OBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	M	L	M	M
P2	L	M	M	M
P3				
P4	M	M	L	M

DECLARATION

We hereby declare that the project report entitled "**PREDICTION OF HEART DISEASES USING ML"** is done in the partial fulfillment for the award of the Degree in Bachelor of Technology in Computer Science and Engineering from KMIT affiliated to Jawaharlal Nehru Technological University, Hyderabad. This project has not been submitted anywhere else.

EMPATI BHAVANI(17BD1A054D)

GORANTA LAVANYALAHARI(17BD1A054H)

PODDUTURI SANJANA(17BD1A055C)

SAI MOHANA HARINISRI VISSAMSETTY(17BD1A055K)

ACKNOWLEDGMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. Maheshwar Dutta**, B.E., M Tech., Ph.D., Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Mr. S. Nitin**, Director and **Mrs. Deepa Ganu**, Dean Academics for providing an excellent environment in the college.

We are also thankful to **Dr. S. Padmaja**, Head of the Department for providing us with both time and amenities to make this project a success within the given schedule.

We are also thankful to our guide **Haleema Bushra**, for her valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project. We sincerely thank our friends and family for their constant motivation during the project work.

EMPATI BHAVANI(17BD1A054D)
GORANTA LAVANYALAHARI(17BD1A054H)
PODDUTURI SANJANA(17BD1A055C)
SAI MOHANA HARINISRI VISSAMSETTY(17BD1A055K)

CONTENT

ABSTRACT

Healthcare is a sought after task in the human life. One in four deaths are due to heart disease in India alone. In order to reduce the number of deaths, there is a need to automate the prediction process and alert the patient well in advance. Healthcare industry contains a lot of medical data which aids machine learning algorithms in making decisions accurately in predicting the heart diseases. This project makes use of the heart disease dataset available in Cleveland database of UCI machine learning repository.This project has delved into different algorithms namely Decision tree, k-nearest neighbour algorithm(KNN), Random Forests.The database consists of 303 instances and 76 attributes. Using the decision tree algorithm, we will be able to identify those attributes which are the best one that will lead us to a better prediction of the datasets. The decision tree algorithm works in a way where it tries to solve the problem by the help of tree representation. Here each internal node of the tree represents an attribute, and each leaf node corresponds to a class label.Random Forests consists of multiple decision trees that operate as an ensemble.Random Forests outperform as they are collection of large relatively uncorrelated models.KNN can easily identify and classify people with heart disease from healthy people.The proposed project compares the results using different performance measures,i.e. accuracy,precision,etc. This project delivers the prediction valued from no presence to likely presence. The proposed project's aim is to try and reduce the occurrences of heart diseases in patiens and thus assist doctors in diagnose it effectively.

Keywords: Machine Learning, Heart Diseases, Decision Tree ,K-nearest neighbour, Random Forest,Cleveland Dataset.

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1

1.INTRODUCTION

1.1 Purpose of Project

Heart disease (HD) is one of the most common diseases nowadays, due to number of contributing factors, such as high blood pressure, diabetes, cholesterol fluctuation, exhaustion and many others. An early diagnosis of such disease has been sought for many years, and many data analytics tools have been applied to help health care providers to identify some of the early signs of HD. Many tests can be performed on potential patients to take the extra precautions measures to reduce the effect of having such a disease, and reliable methods to predict early stages of HD, such as the methods proposed in this project, can be a crucial task for saving lives. Number of Machine Learning (ML) algorithms, such as ,K- Nearest Neighbor (K-NN),Decision tree,Random Forest were applied for the purpose of classification and prediction of HD dataset, and many promising results were presented in the literature.

1.2 Problems with Existing System

The existing System was implemented using the Naive Bayes, Support Vector Machine(SVM). Drawbacks of the existing system include prediction of cardiovascular disease results is not accurate. Data Mining techniques does not help to provide effective decision making. It cannot handle the enormous datasets for patient records.

1.3 Proposed System

After evaluating the results from the existing methodologies, we have used python operations to perform heart disease classification for the data obtained from the UCI repository. It provides an easy-to-use visual representation of the dataset, working environment and building the predictive analytics.ML process starts from a preprocessing data phase followed by feature selection based on data cleaning, classification of modelling performance evaluation. KNN technique is used to improve the accuracy of the result.

1.4 Scope of the Project

Here the scope of the project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modeling 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

1.5 Architecture Diagram

The architecture of the proposed system is as displayed in the figure below. The major components of the architecture are as follows: patient database, preprocessing, tokenization, training the model, test the model, design fitness function, application of genetic algorithm, results collection and prediction of heart disease.

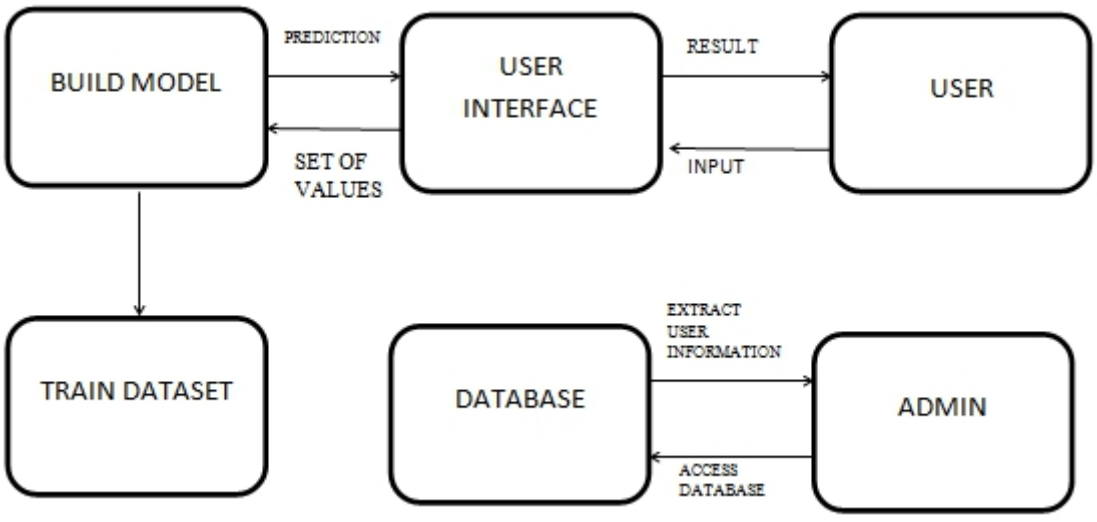


Fig 1.5- Architecture Diagram

CHAPTER -2

2. SYSTEM RERUIREMENT SPECIFICATIONS

2.1 What is SRS?

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.)

The SRS phase consists of two basic activities:

Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

Requirement Specification:

Here, 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.

2.2 Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients 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.

2.3 Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a

manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behaviour of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS.

This section introduces the requirement specification document for Prediction of heart diseases using Machine Learning which enlists functional as well as non-functional requirements.

2.4 Functional Requirements

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data. Functional requirements define specific behaviour or function of the application. Following are the functional requirements:

FR1) The system should allow administrator to monitor and remove inappropriate datasets and code.

FR2) The system allows the users to predict heart disease.

FR3) The system allows users to input attributes.

FR4) Predict disease with the given symptoms.

FR5) Compare the given symptoms with the input datasets.

2.5 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. Especially these are the constraints the system must work within. Following are the non-functional requirements:

NFR 1) The website should provide values used during prediction to the user.

NFR 2) The website should be responsive and consistent.

NFR 3) The model should display the correct condition of the user heart.

Performance:

The performance of the developed model can be calculated by using following methods: Measuring enables you to identify how the performance of your model stands in relation to your defined performance goals and helps you to identify the bottlenecks that affect your model performance. It helps you identify whether your model is moving toward or away from your performance goals. Defining what you will measure, that is, your metrics, and defining the objectives for each metric is a critical part of your testing plan.

Performance objectives include the following:

TP rate, FP rate, F-measure, Accuracy, Precision.

2.6 Software Requirements

Operating System	:	Windows 10 or MAC OS.
Platform	:	Jupyter-Lab
Programming Language	:	Python

2.7 Hardware Requirements

Processor	:	Intel core i3 and above.
Hard Disk	:	100 GB or above.
RAM	:	1 GB or above.
Internet	:	4 Mbps or above (Wireless).

CHAPTER -3

3.LITERATURE SURVEY

The process of testing a software in a well-planned and systematic way is known as software testing lifecycle (STLC). Different organizations have different phases in STLC however generic Software Test Life Cycle (STLC) for waterfall development model consists of the following phases :

1. Requirements Analysis
2. Test Planning
3. Test Analysis
4. Test Design
5. Test Construction and Verification
6. Test Execution and Bug Reporting
7. Final Testing and Implementation
8. Post Implementation

3.1 Requirements Analysis

In this phase testers analyse the customer requirements and work with developers during the design phase to see which requirements are testable and how they are going to test those requirements. It is very important to start testing activities from the requirements phase itself because the cost of fixing defect is very less if it is found in requirements phase rather than in future phases. In this phase all the planning about testing is done like what needs to be tested, how the testing will be done, test strategy to be followed, what will be the test environment, what test methodologies will be followed, hardware and software availability, resources, risks etc. A high level test plan document is created which includes all the planning inputs mentioned above and circulated to the stakeholders.

Usually IEEE 829 test plan template is used for test planning.

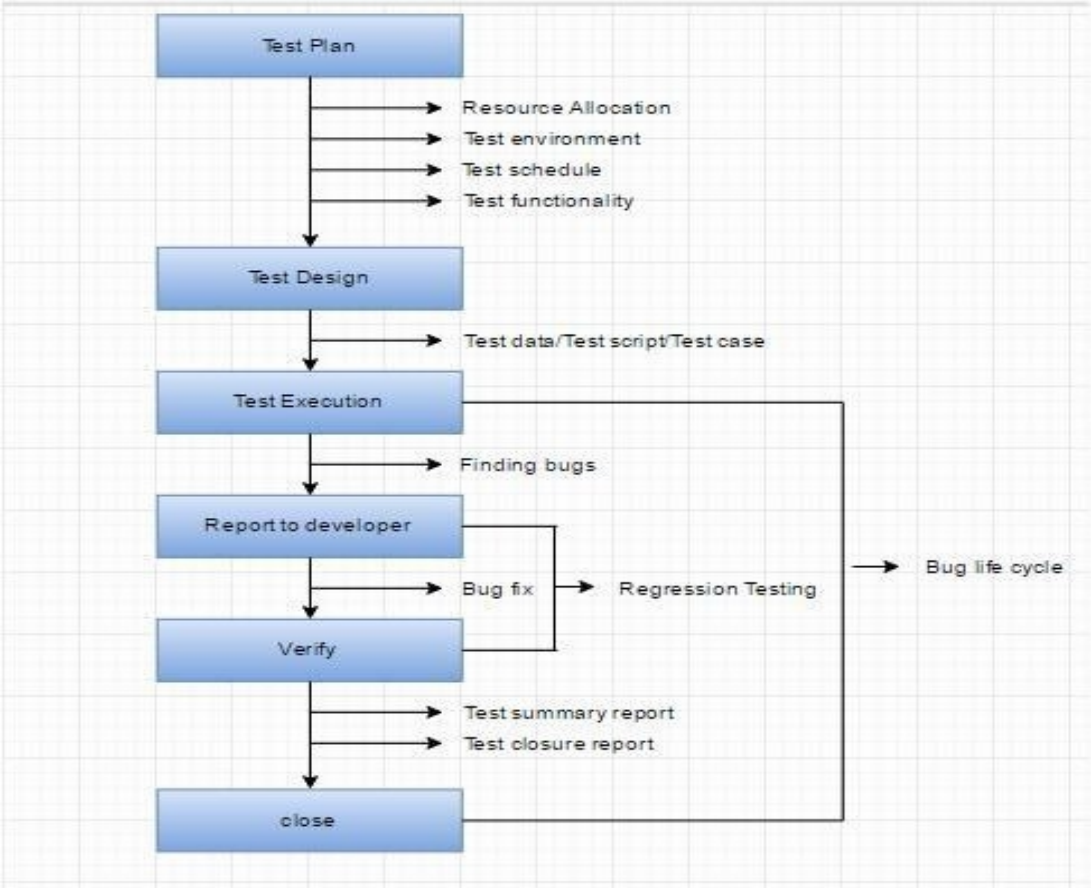


Fig 3.1 - IEEE 829 test plan template

3.1.1 Test Analysis

After test planning phase is over test analysis phase starts, in this phase we need to dig deeper into project and figure out what testing needs to be carried out in each SDLC phase. Automation activities are also decided in this phase, if automation needs to be done for software product, how will the automation be done, how much time will it take to automate and which features need to be automated. Non functional testing areas (Stress and performance testing) are also analysed and defined in this phase.

3.1.2 Test Design

In this phase various black-box and white-box test design techniques are used to design the test cases for testing, testers start writing test cases by following those design

techniques, if automation testing needs to be done then automation scripts also needs to written in this phase.

3.1.3 Test Construction and Verification

In this phase testers prepare more test cases by keeping in mind the positive and negative scenarios, end user scenarios etc. All the test cases and automation scripts need to be completed in this phase and got reviewed by the stakeholders. The test plan document should also be finalized and verified by reviewers.

3.1.4 Test Execution and Bug Reporting

Once the unit testing is done by the developers and test team gets the test build, The test cases are executed and defects are reported in bug tracking tool, after the test execution is complete and all the defects are reported. Test execution reports are created and circulated to project stakeholders. After developers fix the bugs raised by testers they give another build with fixes to testers, testers do re-testing and regression testing to ensure that the defect has been fixed and not affected any other areas of software. Testing is an iterative process i.e. If defect is found and fixed, testing needs to be done after every defect fix. After tester assures that defects have been fixed and no more critical defects remain in software the build is given for final testing.

3.1.5 Final Testing and Implementation

In this phase the final testing is done for the software, non functional testing like stress, load and performance testing are performed in this phase. The software is also verified in the production kind of environment. Final test execution reports and documents are prepared in this phase.

3.1.6 Post Implementation

In this phase the test environment is cleaned up and restored to default state, the process review meetings are done and lessons learnt are documented. A document is prepared to cope up similar problems in future releases.

Phase	Activities	Outcome
Planning	Create high level test plan	Test plan, Refined Specification
Analysis	Create detailed test plan,Functional requirements and Non-Functional requirements	Revised Test Plan, Functional requirements and Non-Functional requirements
Design	UML diagrams and architecture diagram.	Revised UML diagrams and architecture diagrams.
Construction	Building the model using ML algorithms.	Review of the built model.
Testing cycles	Testing the model with the test data.	Test results, Bug reports
Final testing	Execute remaining test data and performance tests, complete Documentation.	Test results and different metrics on test efforts.
Post implementation	Evaluate testing processes	Plan for improvement of testing process

Table 3.1 – Activities and Outcomes of each phase in SDLC

3.2 Technologies Used :

3.2.1 Machine Learning(ML)

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

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

Supervised learning:

The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning:

No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning:

A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

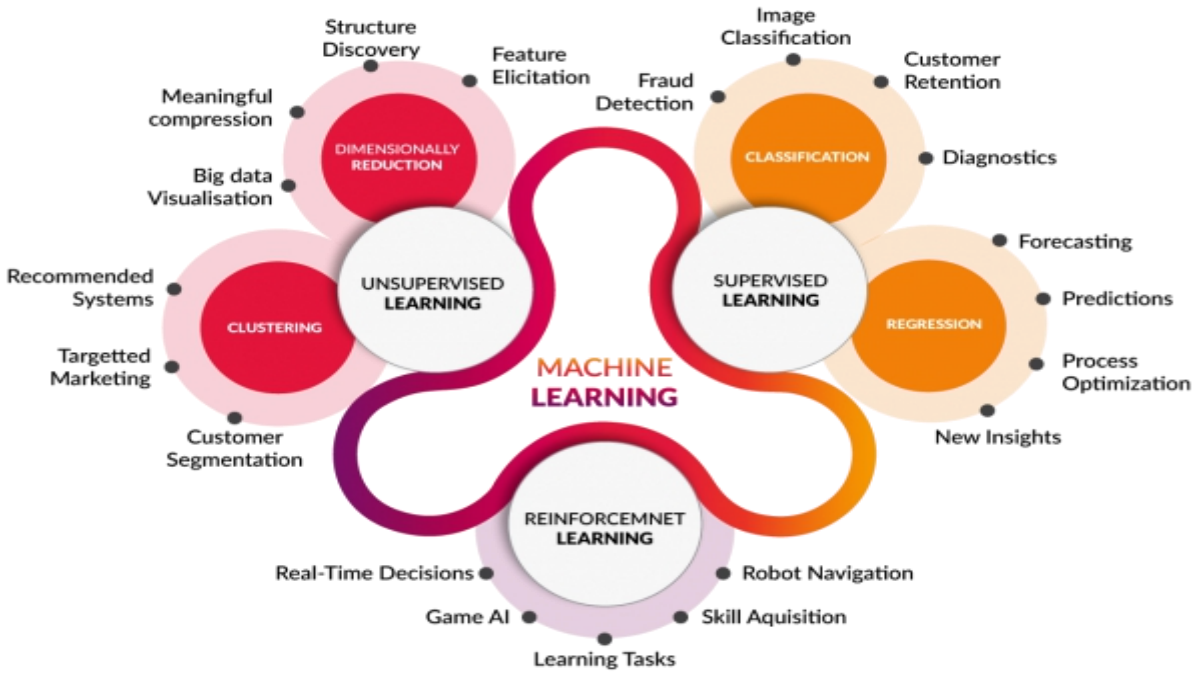


Fig 3.2.1- Machine Learning Categories

3.2.1.1 K Nearest Neighbors

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

A classification problem has a discrete value as its output. A regression problem has a real number (a number with a decimal point) as its output. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

The KNN Algorithm

1.Load the data
2.Initialize K to your chosen number of neighbors
3. For each example in the data
    3.1 Calculate the distance between the query example and the current example from the data.
    3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.As we decrease the value of K to 1, our predictions become less stable.Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far. n cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search.

Disadvantages

- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

KNN's main disadvantage of becoming significantly slower as the volume of data increases makes it an impractical choice in environments where predictions need to be made rapidly. Moreover, there are faster algorithms that can produce more accurate classification and regression results.

3.2.1.2 Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks.

Tree models where the target variable can take a discrete set of values are called classification trees. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Classification And Regression Tree (CART) is general term for this.

Data comes in records of forms i.e (x,Y)=(x1,x2,x3,....,xk,Y). The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the features, x1, x2, x3 etc., that are used for that task.

While making decision tree, at each node of tree we ask different type of questions. Based on the asked question we will calculate the information gain corresponding to it.

Information Gain

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value measures how much information a feature gives us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

Algorithm for constructing decision tree usually works top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrices for measuring best.

Gini Impurity

Pure

Pure means, in a selected sample of dataset all data belongs to same class (PURE).

Impure

Impure means, data is mixture of different classes.

Definition of Gini Impurity

Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set.

If our dataset is Pure then likelihood of incorrect classification is 0. If our sample is mixture of different classes then likelihood of incorrect classification will be high.

Steps for Making decision tree

- Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each nodes).
- Calculate uncertanity of our dataset or Gini impurity or how much our data is mixed up etc.
- Generate list of all question which needs to be asked at that node.
- Partition rows into True rows and False rows based on each question asked.

- Calculate information gain based on gini impurity and partition of data from previous step.
- Update highest information gain based on each question asked.
- Update best question based on information gain (higher information gain).
- Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

Advantage of Decision Tree

- Easy to use and understand.
- Can handle both categorical and numerical data.
- Resistant to outliers, hence require little data preprocessing.

Disadvantage of Decision Tree

- Prone to overfitting.
- Require some kind of measurement as to how well they are doing.
- Need to be careful with parameter tuning.
- Can create biased learned trees if some classes dominate.

3.2.1.3 Random Forest

Random Forest can be used for both classification and regression problems. Random Forest algorithm is a supervised classification algorithm. We can see it from its name, which is to create a forest by some way and make it random. There is a direct relationship between the number of trees in the forest and the results it can get: the larger the number of trees, the more accurate the result. But one thing to note is that creating the forest is not the same as constructing the decision with information gain or gain index approach.

The decision tree is a decision support tool. It uses a tree-like graph to show the possible consequences. If you input a training dataset with targets and features into the decision tree, it will formulate some set of rules. These rules can be used to perform predictions. Through the decision tree algorithm, you can generate the rules. You can then input the features of this movie and see whether it will be liked by your daughter. The process of calculating these nodes and forming the rules is using information gain and Gini

index calculations.

The difference between Random Forest algorithm and the decision tree algorithm is that in Random Forest, the process es of finding the root node and splitting the feature nodes will run randomly.

Overfitting is one critical problem that may make the results worse, but for Random Forest algorithm, if there are enough trees in the forest, the classifier won't overfit the model. The third advantage is the classifier of Random Forest can handle missing values, and the last advantage is that the Random Forest classifier can be modeled for categorical values.

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage.

Random Forest creation pseudocode:

1. Randomly select "K" features from total "m" features where k << m
2. Among the "K" features, calculate the node "d" using the best split point
3. Split the node into daughter nodes using the best split
4. Repeat the a to c steps until "l" number of nodes has been reached
5. Build forest by repeating steps a to d for "n" number times to create "n" number of trees

In the next stage, with the random forest classifier created, we will make the prediction. The random forest prediction pseudocode is shown below:

Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target). Calculate the votes for each predicted target

Consider the high voted predicted target as the final prediction from the random forest algorithm

Advantages of Random Forest algorithm:

- Compared with other classification techniques, there are three advantages as the author mentioned.
- For applications in classification problems, Random Forest algorithm will avoid the

overfitting problem.

- For both classification and regression task, the same random forest algorithm can be used
- The Random Forest algorithm can be used for identifying the most important features from the training dataset, in other words, feature engineering.

3.2.2 Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge.

3.2.3 Jupyter-Lab

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

JupyterLab is a next-generation web-based user interface for Project Jupyter. JupyterLab enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner.

You can arrange multiple documents and activities side by side in the work area using tabs and splitters. Documents and activities integrate with each other, enabling new workflows for interactive computing, for example:

Code Consoles provide transient scratchpads for running code interactively, with full support for rich output. A code console can be linked to a notebook kernel as a computation log from the notebook, for example.

Kernel-backed documents enable code in any text file (Markdown, Python, R, LaTeX, etc.) to be run interactively in any Jupyter kernel.

Notebook cell outputs can be mirrored into their own tab, side by side with the notebook, enabling simple dashboards with interactive controls backed by a kernel.

Multiple views of documents with different editors or viewers enable live editing of documents reflected in other viewers. For example, it is easy to have live preview of Markdown, Delimiter-separated Values, or Vega/Vega-Lite documents.

JupyterLab also offers a unified model for viewing and handling data formats. JupyterLab understands many file formats (images, CSV, JSON, Markdown, PDF, Vega, Vega-Lite, etc.) and can also display rich kernel output in these formats. See File and Output Formats for more information.

Packages

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

Library features

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.

- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation[6] and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

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 for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them

Seaborn is the only library we need to import for this simple example. By convention, it is imported with the shorthand sns.

Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call matplotlib.pyplot.show() when you want to see the plot.

This uses the matplotlib rcParam system and will affect how all matplotlib plots look, even if you don't make them with seaborn. Beyond the default theme, there

are several other options, and you can independently control the style and scaling of the plot to quickly translate your work between presentation contexts (e.g., making a version of your figure that will have readable fonts when projected during a talk). If you like the matplotlib defaults or prefer a different theme, you can skip this step and still use the seaborn plotting functions.

MatPlotLib

Matplotlib is a Sponsored Project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. NumFOCUS provides Matplotlib with fiscal, legal, and administrative support to help ensure the health and sustainability of the project.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

For installation instructions and requirements, see INSTALL.rst or the install documentation.

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.SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, since then it has an active development community,[4] and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has

pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

Several toolkits are available which extend Matplotlib functionality. Some are separate downloads, others ship with the Matplotlib source code but have external dependencies.

Basemap:

map plotting with various map projections, coastlines, and political boundaries

Cartopy:

a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities. (Matplotlib v1.2 and above)

Excel tools:

utilities for exchanging data with Microsoft Excel

GTK tools:

interface to the GTK+ library

Qt interface

Mplot3d: 3-D plots

Natgrid:

interface to the natgrid library for gridding irregularly spaced data.

matplotlib2tikz:

export to Pgfplots for smooth integration into LaTeX documents[14]

Seaborn:

provides an API on top of Matplotlib that offers sane choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas.

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 Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many

contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Existing System

The existing system was built using the Support Vector Machines and Naive Bayes Algorithms.

Naïve Bayes Classifier

Naïve Bayes classifier is based on Bayes theorem. This classifier uses conditional independence in which attribute value is independent of the values of other attributes.

The Bayes theorem is as follows:

Let X= {x1, x2, ..., Xn} be a set of n attributes. In Bayesian, X is considered as evidence

and H be some hypothesis means, the data of X belongs to specific class C. We have to determine P (H|X), the probability that the hypothesis H holds given evidence i.e. data sample X. According to Bayes theorem the P (H|X) is expressed as: P(H|X) = P(X|H) P(H) / P(X).4

Naive Bayes algorithm gives an accuracy of 52.33% with 609m time taken.Using Bayesian classifiers, the system will discover the concealed knowledge associated with diseases from historical records of the patients having heart disease. Bayesian classifiers predict the class membership probabilities, in a way that the probability of a given sample belongs to a particular class statistically. Bayesian classifier is based on Bayes' theorem. We can use Bayes theorem to determine the probability that a proposed diagnosis is correct, given the observation. A simple probabilistic, the naive Bayes classifier is used for classification based on which is based on Bayes' theorem.

According to naïve Bayesian classifier the occurrence or an occurrence of a particular feature of a class is considered as independent in the presence or absence of any other feature. When the dimension of the inputs is high and more efficient result is expected, the chief Naïve Bayes Classifier technique is applicable. The Naïve Bayes model identifies the physical characteristics and features of patients suffering from heart disease. For each input, it gives the possibility of attribute of the expectable state. Naïve Bayes is a statistical classifier which assumes no dependency between attributes. This classifier algorithm uses conditional independence, means it assumes that an attribute value of a given class is independent of the values of other attributes. The advantage of using Naïve Bayes is that one can work with the Naïve Bayes model without. using any Bayesian methods. (Brownlee, 2016). P (Disease|symptom1, symptom2, … … . , symptomn) P(Disease)P(symptom1, … .. , symptomn|Disease) =P(symptom1, symptom2, … . .Symptomn).

The classification tree literally creates a tree with branches, nodes, and leaves that lets us take an unknown data point and move down the tree, applying the attributes of the data point to the tree until a leaf is reached and the unknown output of the data point can be determined. In order to create a good classification tree model, we need to have an existing data set with known output from which we can build our model. We also divide our data set into two parts: a training set, which is used to create the model, and a test set, which is used

to verify that the model is accurate and not over fitted.

Support Vector Machines(SVM)

SVM (Support Vector Machine) is a supervised machine learning algorithm that is mainly used to classify data into different classes. Unlike most algorithms, SVM makes use of a hyperplane, which acts like a decision boundary between the various classes.

SVM can be used to generate multiple separating hyperplanes such that the data is divided into segments and each segment contains only one kind of data.

1. SVM is a supervised learning algorithm. This means that SVM trains on a set of labeled data. SVM studies the labeled training data and then classifies any new input data depending on what it learned in the training phase.
2. A main advantage of SVM is that it can be used for both classification and regression problems. Though SVM is mainly known for classification, the SVR (Support Vector Regressor) is used for regression problems.
3. SVM can be used for classifying non-linear data by using the kernel trick. The kernel trick means transforming data into another dimension that has a clear dividing margin between classes of data. After which you can easily draw a hyperplane between the various classes of data.

The basic principle behind SVM is to draw a hyperplane that best separates the 2 classes. In our case, the two classes are rabbits and wolves. Before we move any further, let's try to understand what a support vector is.

The hyperplane is drawn based on these support vectors and an optimum hyperplane will have a maximum distance from each of the support vectors. And this distance between the hyperplane and the support vectors is known as the margin.

To sum it up, SVM is used to classify data by using a hyperplane, such that the distance between the hyperplane and the support vectors is maximum.

Support vector machine (SVM) proposed by Vapnik and Cortes have been successfully applied for gender classification problems by many researchers. An SVM classifier is a linear classifier where the separating hyper plane is chosen to minimize the expected classification error of the unseen test patterns. SVM is a strong classifier which

can identify two classes.

SVM classifies the test image to the class which has the maximum distance to the closest point in the training. SVM training algorithm built a model that predict whether the test image fall into this class or another. SVM require a huge amount of training data to select an affective decision boundary and computational cost is very high even if we restrict ourselves to single pose (frontal) detection.

The SVM is a learning algorithm for classification. It tries to find the optimal separating hyper plane such that the expected classification error for unseen patterns is minimized. For linearly non-separable data the input is mapped to high-dimensional feature space where they can be separated by a hyper plane. This projection into high dimensional feature space is efficiently performed by using kernels. More precisely, given a set of training samples and the corresponding decision values -1, 1 the SVM aims to find the best separating hyper plane given by the equation WTx+b that maximizes the distance between the two classes.

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.
- Classification of satellite data like SAR data using supervised SVM.
- Hand-written characters can be recognized using SVM.
- The SVM algorithm has been widely applied in the biological and other sciences.
- They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support-vector machine weights have

also been used to interpret SVM models in the past. Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

Kaggle

Our Cleveland Dataset is extracted from Kaggle from UCI repository.Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Kaggle's services:

Machine learning competitions: this was Kaggle's first product. Companies post problems and machine learners compete to build the best algorithm, typically with cash prizes.

Kaggle Kernels: a cloud-based workbench for data science and machine learning. Allows data scientists to share code and analysis in Python, R and R Markdown. Over 150K "kernels" (code snippets) have been shared on Kaggle covering everything from sentiment analysis to object detection.

Public datasets platform: community members share datasets with each other. Has datasets on everything from bone x-rays to results from boxing bouts.

Kaggle Learn: a platform for AI education in manageable chunks.

Kaggle has run hundreds of machine learning competitions since the company was founded. Competitions have ranged from improving gesture recognition for Microsoft Kinect to making an football AI for Manchester City to improving the search for the Higgs boson at CERN.

Competitions have resulted in many successful projects including furthering the state of the art in HIV research, chess ratings and traffic forecasting. Most famously, Geoffrey Hinton and George Dahl used deep neural networks to win a competition hosted by Merck. And Vlad Mnih (one of Hinton's students) used deep neural

networks to win a competition hosted by Adzuna. This helped show the power of deep neural networks and resulted in the technique being taken up by others in the Kaggle community. Tianqi Chen from the University of Washington also used Kaggle to show the power of XGBoost, which has since taken over from Random Forest as one of the main methods used to win Kaggle competitions.

The dataset consists of 303 individuals data. There are 14 columns in the dataset. Our dataset contains the following attributes:

Age:

displays the age of the individual.

Sex:

displays the gender of the individual using the following format :

1 = male

0 = female

Chest-pain type:

displays the type of chest-pain experienced by the individual using the following format :

1 = typical angina

2 = atypical angina

3 = non — anginal pain

4 = asymptotic

Resting Blood Pressure:

displays the resting blood pressure value of an individual in mmHg (unit)

Serum Cholestrol:

displays the serum cholesterol in mg/dl (unit)

Fasting Blood Sugar:

compares the fasting blood sugar value of an individual with 120mg/dl.

If fasting blood sugar > 120mg/dl then : 1 (true)

else : 0 (false)

Resting ECG :

displays resting electrocardiographic results

0 = normal

1 = having ST-T wave abnormality

2 = left ventricular hyperthrophy

Max heart rate achieved :

displays the max heart rate achieved by an individual.

Exercise induced angina :

1 = yes

0 = no

ST depression induced by exercise relative to rest:

displays the value which is an integer or float.

Peak exercise ST segment :

1 = upsloping

2 = flat

3 = downsloping

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

displays the value as integer or float.

Thal :

displays the thalassemia :

3 = normal

6 = fixed defect

7 = reversible defect

Diagnosis of heart disease :

Displays whether the individual is suffering from heart disease or not :

0 = absence

1, 2, 3, 4 = present.

CHAPTER -4

4. SYSTEM DESIGN

4.1 Introduction to UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows:

1. User Model View

   This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.
2. Structural Model View

   In this model, the data and functionality are arrived from inside the system. This model view models the static structures.
3. Behavioural Model View

   It represents the dynamic of behavioural as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.
4. Implementation Model View

   In this view, the structural and behavioural as parts of the system are represented as they are to be built.
5. Environmental Model View

   In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

4.2 UML DIAGRAMS

4.2.1 Use-Case Diagram

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

a. Used to gather requirements of a system.
b. Used to get an outside view of a system.
c. Identify external and internal factors influencing the system.
d. Show the interacting among the requirements are actors.

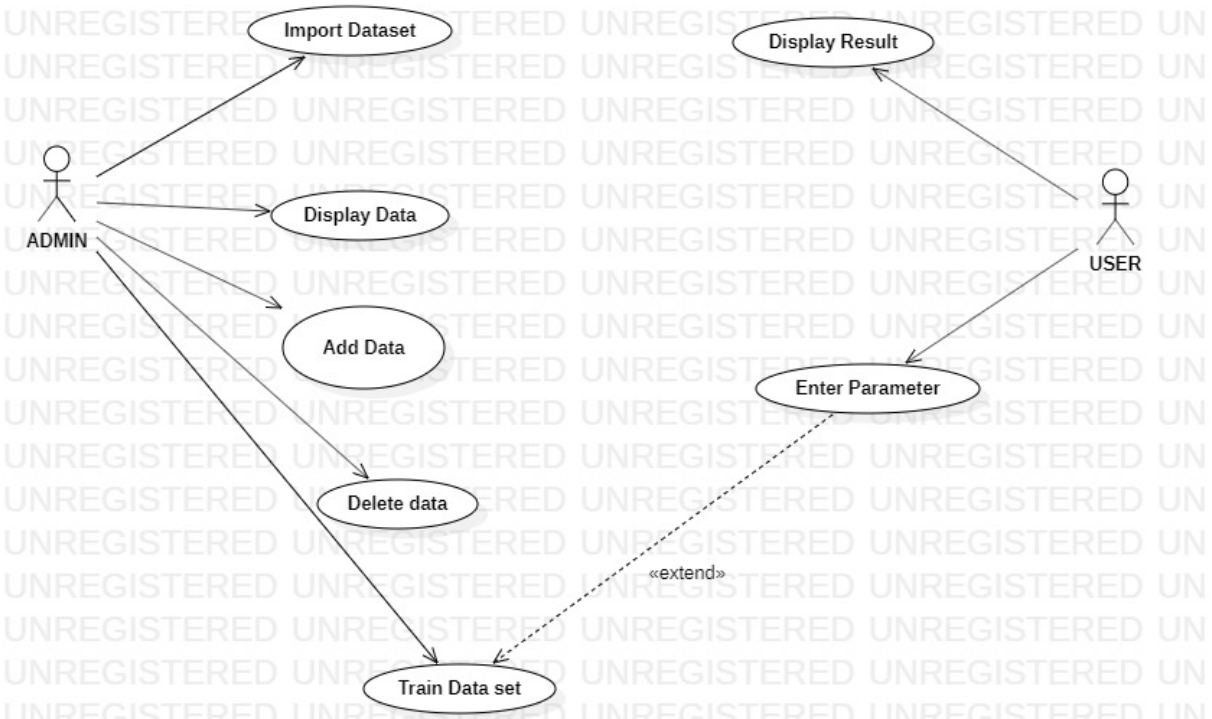


Fig 4.2.1 – Use Case Diagram

4.2.2 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

Basic Sequence Diagram

Class Roles or Participants

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin grey rectangle placed vertically on its lifeline.

Messages

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages.

Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

Lifelines

Lifelines are vertical dashed lines that indicate the object's presence over time.

Destroying Objects

Objects can be terminated early using an arrow labelled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction

occurrence.

Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets []. Guards

When modelling object interactions, there will be times when a condition must be met for a message to be sent to an object. Guards are conditions that need to be used throughout UML diagrams to control flow.

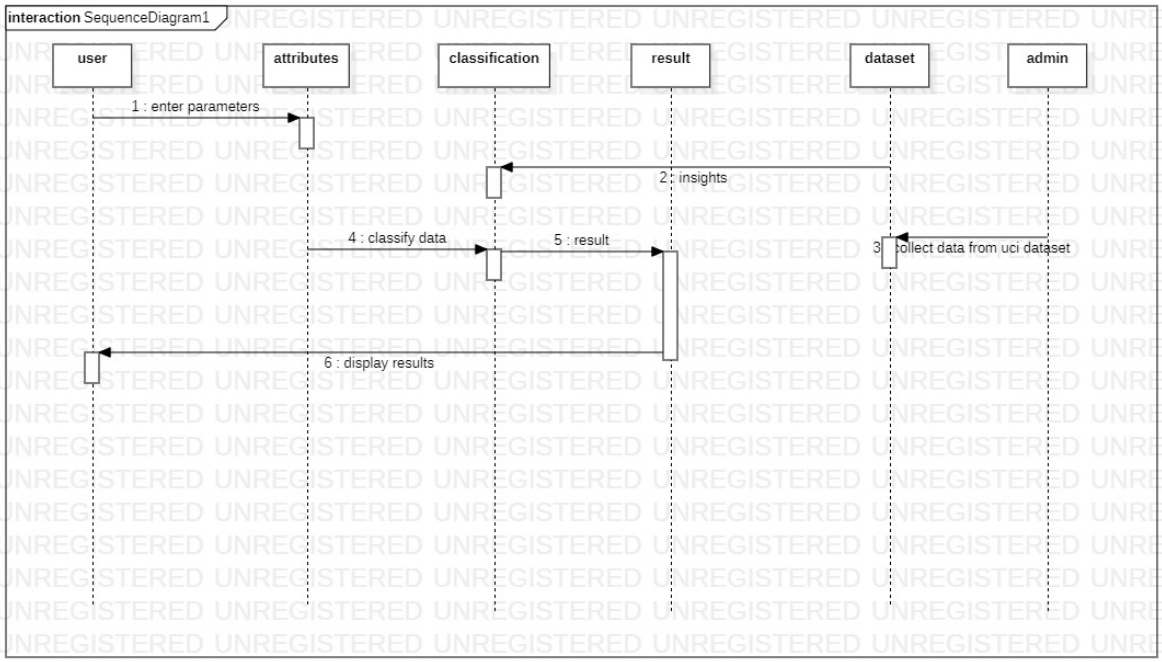


Fig 4.2.2 – Sequence Diagram

4.2.3 Class Diagram

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything

that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.

The main purpose to use class diagrams are:

1. This is the only UML which can appropriately depict various aspects of OOPs concept.
2. Proper design and analysis of application can be faster and efficient.
3. It is base for deployment and component diagram.

Each class is represented by a rectangle having a subdivision of three compartments name,attributes and operation.

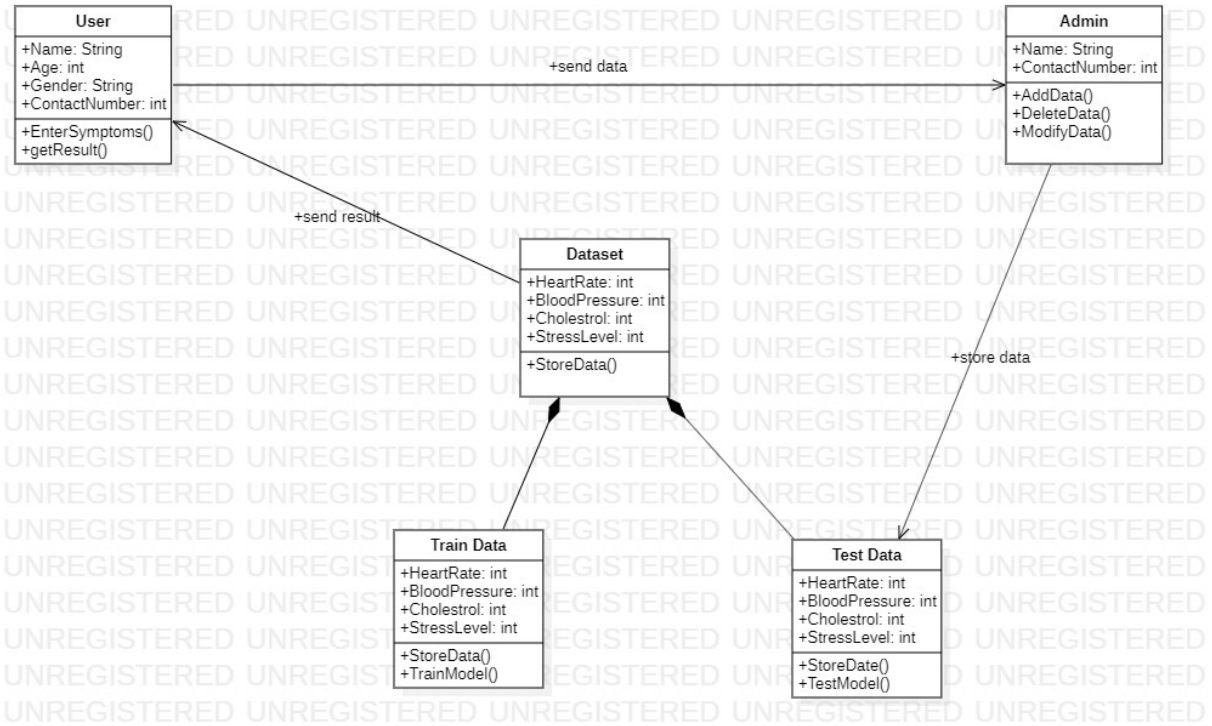


Fig 4.2.3-Class Diagram

4.2.4 Object Diagram

Object is an instance of a class in a particular moment in runtime that can have its own state and data values. Likewise a static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships which may be considered a special case of a class diagram or a communication diagram.

Purpose of Object Diagram

The use of object diagrams is fairly limited, mainly to show examples of data structures.

During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.

Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.

An object diagram shows this relation between the instantiated classes and the defined class, and the relation between these objects in the system. They are be useful to explain smaller portions of your system, when your system class diagram is very complex, and also sometimes modeling recursive relationship in diagram.

Object Names:

Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.

Object Attributes:

Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.

Links:

Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

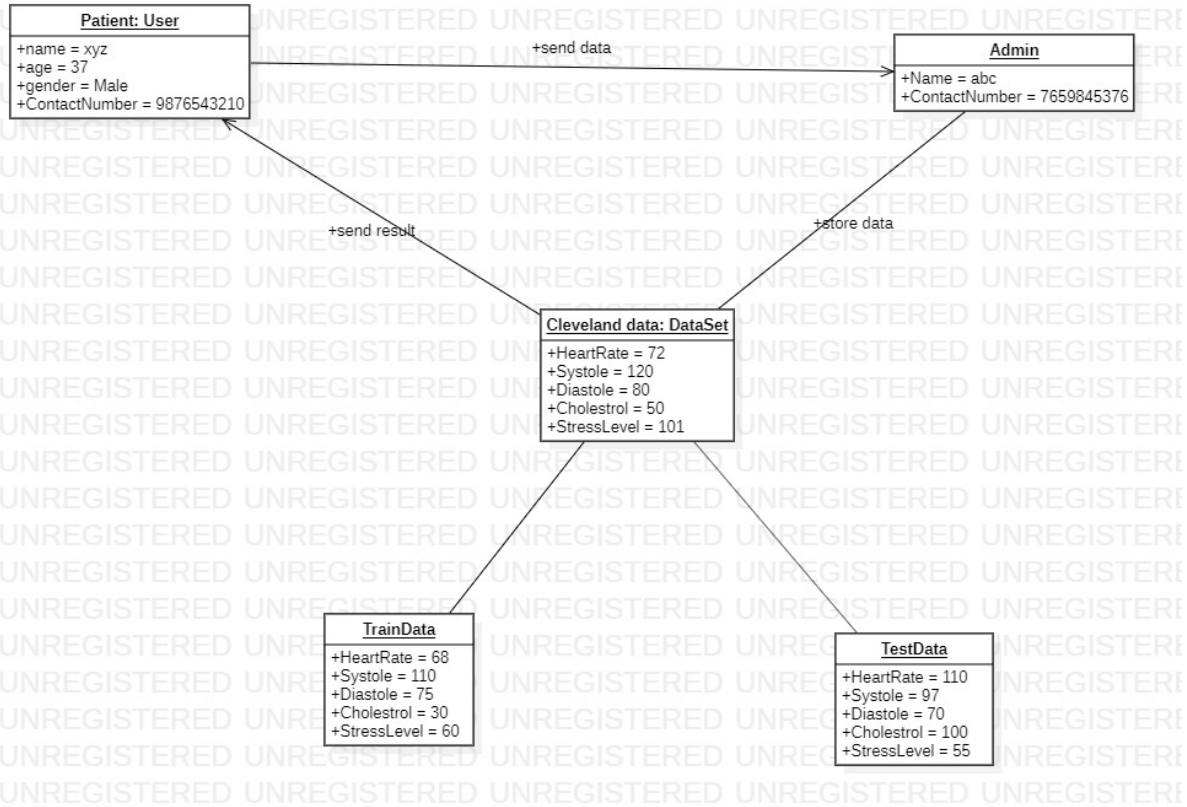


Fig 4.2.4-Object Diagram

CHAPTER -5

5.IMPLEMENTATION

5.1 Pseudo Code

Step 1: Import the required packages.

Step 2: Load the dataset.

Step 3: Summarizing the dataset

Step 4:Applying the datamining techniques like data preprocessing.

Step 5:Applying the feature selection and reduction process to normalize the data.

Step 6: Visualizing the dataset (Correlation matrix)

Step 7: Implementing the algorithms mentioned- K Nearest Neighbours(KNN), Decision Trees, Random Forest.

Step 8: Finding the accuracy using F-measure.

5.2 Code Snippets

Data Pre-Processing

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")


df = pd.read_csv(r"C:\Users\Podduturi sanjana\Downloads\heart.csv")
df.head()

df.info()

df.shape

pd.set_option("display.float", "{:.2f}".format)
df.describe()

df.target.value_counts()


df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])

df.isna().sum()

categorical_val = []
continous_val = []
for column in df.columns:
    print('==============================')
    print(f"{column} : {df[column].unique()}")
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)

categorical_val

plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
```

```
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)

plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)

plt.figure(figsize=(10, 8))

# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")

# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```

Correlation Matrix

```
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
df.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=True, figsize=(12, 8),
                                                   title="Correlation with target")

categorical_val.remove('target')
dataset = pd.get_dummies(df, columns = categorical_val)

dataset.head()

print(df.columns)
print(dataset.columns)

from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])

dataset.head()

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train
Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test
Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")

from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
```

```
y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

K Nearesr Neighbours(KNN)

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)

from sklearn.tree import DecisionTreeClassifier
```

Decision Tree

```
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
```

Random Forest

```
rf_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

CHAPTER-6

6. TESTING

6.1 Introduction to Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, the following professionals are involved in testing a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- Functional Testing
- Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6.2 Software Testing Life Cycle

The process of testing a software in a well planned and systematic way is known as software testing lifecycle (STLC).

Different organizations have different phases in STLC however generic Software Test Life Cycle (STLC) for waterfall development model consists of the following phases.

1. Requirements Analysis
2. Test Planning
3. Test Analysis
4. Test Design

- Requirements Analysis

  In this phase testers analyze the customer requirements and work with developers during the design phase to see which requirements are testable and how they are going to test those requirements.

  It is very important to start testing activities from the requirements phase itself because the cost of fixing defect is very less if it is found in requirements phase rather than in future phases.

- Test Planning

  In this phase all the planning about testing is done like what needs to be tested, how the testing will be done, test strategy to be followed, what will be the test environment, what test methodologies will be followed, hardware and software availability, resources, risks etc. A high level test plan document is created which includes all the planning inputs mentioned above and circulated to the stakeholders.

- Test Analysis

  After test planning phase is over test analysis phase starts, in this phase we need to dig deeper into project and figure out what testing needs to be carried out in each SDLC phase. Automation activities are also decided in this phase, ifautomation needs to be done for software product, how will the automation be

done, how much time will it take to automate and which features need to be automated. Non functional testing areas(Stress and performance testing) are also analyzed and defined in this phase.

- Test Design

  In this phase various black-box and white-box test design techniques are used to design the test cases for testing, testers start writing test cases by following those design techniques, if automation testing needs to be done then automation scripts also needs to written in this phase.

6.3 Test Cases

- The model is tested with test data.
- Accuracy is calculated for each algorithm.

CHAPTER -7

7.SCREENSHOTS

7.1 Importing packages and Data Set

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

```
df = pd.read_csv(r"C:\Users\Podduturi sanjana\Downloads\heart.csv")
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
```

Fig 7.1.1- Importing Packages and read Dataset

7.2 Data Analysis

```
df.shape
```

```
(303, 14)
```

```
pd.set_option("display.float", "{:.2f}".format)
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00
mean	54.37	0.68	0.97	131.62	246.26	0.15	0.53	149.65	0.33	1.04	1.40	0.73	2.31	0.54
std	9.08	0.47	1.03	17.54	51.83	0.36	0.53	22.91	0.47	1.16	0.62	1.02	0.61	0.50
min	29.00	0.00	0.00	94.00	126.00	0.00	0.00	71.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	47.50	0.00	0.00	120.00	211.00	0.00	0.00	133.50	0.00	0.00	1.00	0.00	2.00	0.00
50%	55.00	1.00	1.00	130.00	240.00	0.00	1.00	153.00	0.00	0.80	1.00	0.00	2.00	1.00
75%	61.00	1.00	2.00	140.00	274.50	0.00	1.00	166.00	1.00	1.60	2.00	1.00	3.00	1.00
max	77.00	1.00	3.00	200.00	564.00	1.00	2.00	202.00	1.00	6.20	2.00	4.00	3.00	1.00

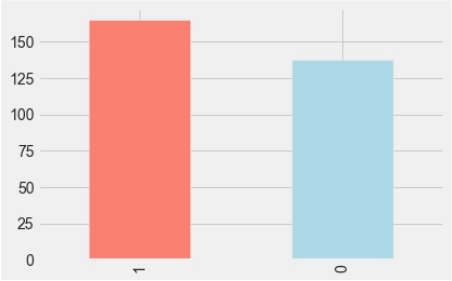
```
df.target.value_counts()
```

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

Fig 7.2.1- Data Analysis

```
df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])
```

```
<AxesSubplot:>
```




```
df.isna().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Fig 7.2.2- Data Analysis(Cont.)

7.3 Data Pre-Processing

```
categorical_val = []
continous_val = []
for column in df.columns:
    print('==============================')
    print(f"{column} : {df[column].unique()}")
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

```
==============================
age : [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
==============================
sex : [1 0]
==============================
cp : [3 2 1 0]
==============================
trestbps : [145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
==============================
chol : [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265 309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284 224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]
==============================
fbs : [1 0]
==============================
restecg : [0 1 2]
==============================
thalach : [150 187 172 178 163 148 153 173 162 174 160 139 171 144 158 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 115 149
 146 175 186 185 159 130 190 132 147 154 202 166 164 184 122 169 138 111
```

Fig 7.3.1- Data Pre-Processing

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

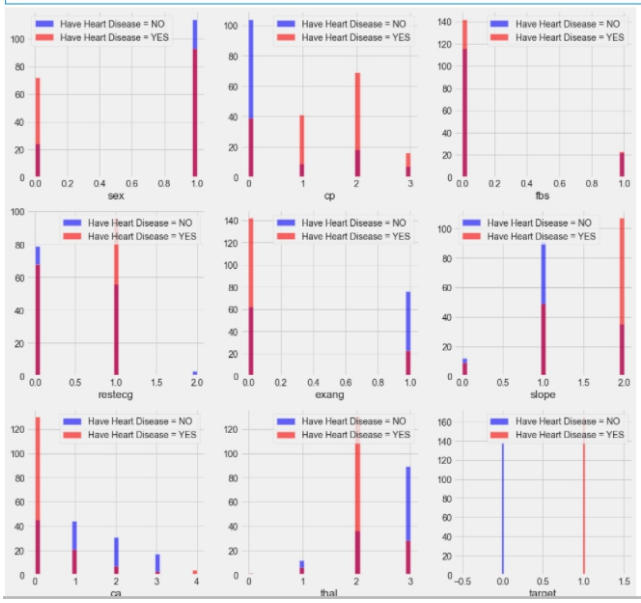


Fig 7.3.2- Data Pre-Processing(Cont.)

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

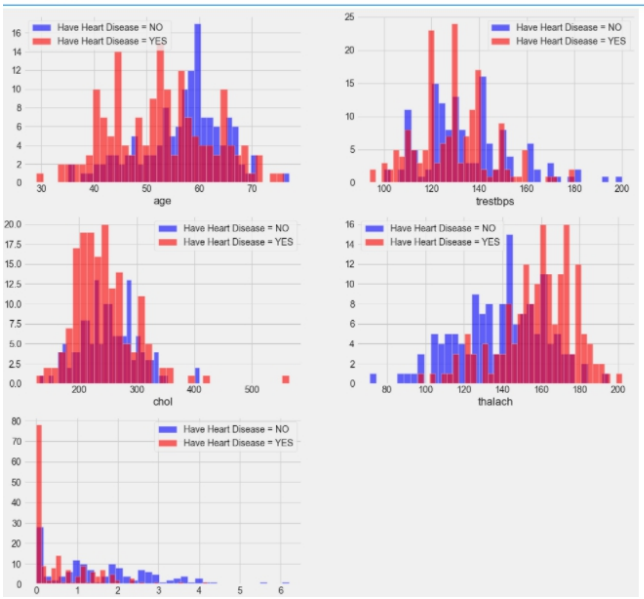


Fig 7.3.3- Data Pre-Processing(Cont.)

```
plt.figure(figsize=(10, 8))

# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")

# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```

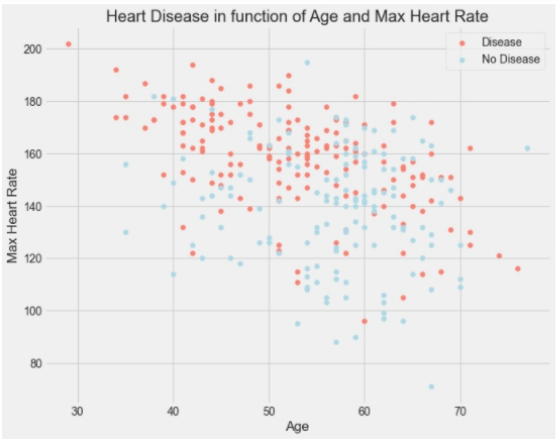


Fig 7.3.4- Data Pre-Processing(Cont.)

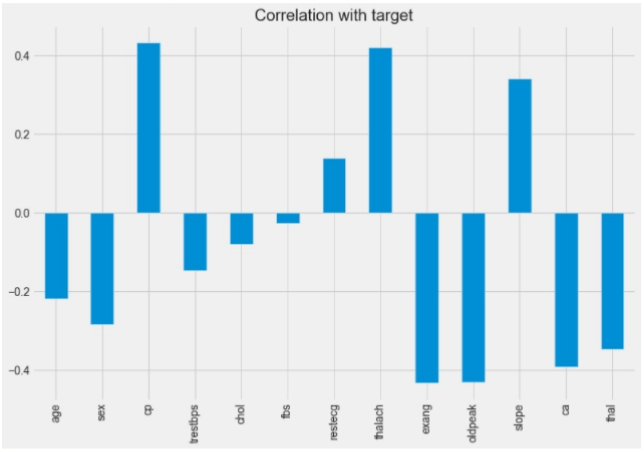
7.4 Correlation matrix

```
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Fig 7.4.1-Correlation Matrix

```
df.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=True, figsize=(12, 8),
                                                   title="Correlation with target")
```

```
<AxesSubplot:title={'center':'Correlation with target'}>
```




```
categorical_val.remove('target')
dataset = pd.get_dummies(df, columns = categorical_val)
```

```
dataset.head()
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	63	145	233	150	2.30	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	37	130	250	187	3.50	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	41	130	204	172	1.40	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	56	120	236	178	0.80	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0

Fig 7.4.2- Correlation with Target

```
print(df.columns)
print(dataset.columns)
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target', 'sex_0',
       'sex_1', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'fbs_0', 'fbs_1', 'restecg_0',
       'restecg_1', 'restecg_2', 'exang_0', 'exang_1', 'slope_0', 'slope_1',
       'slope_2', 'ca_0', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_0', 'thal_1',
       'thal_2', 'thal_3'],
      dtype='object')
```

```
from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])
```

```
dataset.head()
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0.96	0.76	-0.25	0.02	1.09	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	-1.91	-0.09	0.08	1.64	2.13	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	-1.47	-0.09	-0.81	0.98	0.31	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	0.18	-0.67	-0.19	1.24	-0.21	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	0.29	-0.67	2.08	0.58	-0.38	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0

5 rows × 31 columns

Fig 7.4.3-Data Pre-Processing(Cont.)

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_____________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_____________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

```python
from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 7.4.4- Data Classification

7.5 K Nearest Neighbors

```python
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 87.65%
_____________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    0.89   0.87      0.88       0.88          0.88
recall       0.84   0.91      0.88       0.87          0.88
f1-score     0.86   0.89      0.88       0.88          0.88
support    111.00 132.00      0.88     243.00        243.00
_____________________________________________
Confusion Matrix:
 [[ 93  18]
 [ 12 120]]

Test Result:
================================================
Accuracy Score: 91.80%
_____________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.93  0.91      0.92       0.92          0.92
recall       0.89  0.94      0.92       0.92          0.92
f1-score     0.91  0.93      0.92       0.92          0.92
support     28.00 33.00      0.92      61.00         61.00
_____________________________________________
Confusion Matrix:
 [[25  3]
 [ 2 31]]
```

Fig 7.5 -K Nearest Neighbors(KNN)

7.6 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier


tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision   1.00   1.00      1.00       1.00          1.00
recall      1.00   1.00      1.00       1.00          1.00
f1-score    1.00   1.00      1.00       1.00          1.00
support   111.00 132.00      1.00     243.00        243.00
________________________________________________
Confusion Matrix:
 [[111   0]
 [  0 132]]

Test Result:
================================================
Accuracy Score: 81.97%
________________________________________________
CLASSIFICATION REPORT:
              0     1  accuracy  macro avg  weighted avg
precision  0.81  0.82      0.82       0.82          0.82
recall     0.79  0.85      0.82       0.82          0.82
f1-score   0.80  0.84      0.82       0.82          0.82
support   28.00 33.00      0.82     61.00         61.00
________________________________________________
Confusion Matrix:
 [[22  6]
 [ 5 28]]
```

Fig 7.6 -Decision Tree

7.7 Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

rf_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision   1.00   1.00      1.00       1.00          1.00
recall      1.00   1.00      1.00       1.00          1.00
f1-score    1.00   1.00      1.00       1.00          1.00
support   111.00 132.00      1.00     243.00        243.00
________________________________________________
Confusion Matrix:
 [[111   0]
 [  0 132]]

Test Result:
================================================
Accuracy Score: 90.16%
________________________________________________
CLASSIFICATION REPORT:
              0     1  accuracy  macro avg  weighted avg
precision  0.89  0.91      0.90       0.90          0.90
recall     0.89  0.91      0.90       0.90          0.90
f1-score   0.89  0.91      0.90       0.90          0.90
support   28.00 33.00      0.90     61.00         61.00
________________________________________________
Confusion Matrix:
 [[25  3]
 [ 3 30]]
```

Fig 7.7 -Random Forest

CHAPTER -8

8.CONCLUSION

Heart Disease is one of the major concerns for society today.

It is difficult to manually determine the odds of getting heart disease based on risk factors. However, machine learning techniques are useful to predict the output from existing data.

In this project, we proposed a method for heart disease prediction using machine learning techniques, these results showed a great accuracy standard for producing a better estimation result.

We have concluded that by using K Nearest Neighbors we get accuracy of 91% which is highest among random forest(90.16%) and Decision Trees(81.97%) and it is highest among the existing system which is developed using Support Vector Machines and Naive Bayes.

Sample results of heart rate are to be taken at different stages of the same subjects, we find the information from the above input via ML Techniques.

In this what we found is during small datasets in some other cases most of time decision trees direct us to a solution which is not accurate, but when we look at KNN results we are getting more accurate results and random forest results with probabilities of all other possibilities but due to guidance to only one solution decision trees may miss lead.

From the study of various recent research papers written on heart disease prediction using various data mining and machine learning techniques and algorithms. We find that different techniques of data mining and machine learning are used to predict heart disease with the help of different experimental tools such as WEKA, MATLAB etc.

CHAPTER -9

9.REFERENCES

1. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-03626-y#Sec30

2. https://www.sciencedirect.com/science/article/pii/S235291481830217X

3. Khourdifi Y, Bahaj M. Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization. Int J Intell Eng Syst. 2019;12(1):242–52. https://doi.org/10.22266/ijies2019.0228.24.

4. https://towardsdatascience.com/heart-disease-prediction-73468d630cfc

5. https://ieeexplore.ieee.org/abstract/document/9122958

6. Krishnan J Santhana and S Geetha, "Prediction of Heart Disease using Machine Learning Algorithms", ICIICT, 2019

.

7. M. Akhil, B. L. Deekshatulu and P. Chandra, "Classification of Heart Disease Using K-Nearest Neighbor and Genetic Algorithm", Procedia Technol., vol. 10, pp. 85-94, 2013