**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No.** |
|  | **ABSTRACT………………………………………………...** | **8** |
|  | **LIST OF FIGURES ……………………………………….** | **9** |
|  | **LIST OF TABLES ..……………………………………….** | **9** |
| **1.** | **Introduction………………………………………………...** | **10** |
|  | * 1.1 Motivation | **10** |
|  | * 1.2 Scope | **11** |
|  | * 1.3 Objective | **11** |
|  | * 1.4 Application | **12** |
| **2.** | **Literature Survey………………………………………….** | **13** |
|  | * 2.1 Literature Survey | **13** |
| **3.** | **Problem Statement……………………………………….** | **14** |
|  | * 3.1 Problem Statement | **15** |
| **4.** | **Software and Hardware Requirements………………….** | **16** |
|  | * 4.1 Software Requirements | **16** |
|  | * 4.2 Hardware Requirements | **16** |
| **5.** | **Methodology…………………………………………….** | **17** |
|  | * 5.1 Method | **17** |
| **6.** | **Design Framework……………………………....................** | **22** |
|  | * 6.1 Data Flow Diagram | **22** |
|  | * 6.2 Use Case Diagram | **23** |
|  | * 6.3 Flowchart | **24** |
|  | * Activity Diagram | **25** |
| **7.** | **Implementation………………………………...** | **26** |
|  | * 7.1 Snapshots | **26** |
|  | * 7.2 System Maintenance | **29** |
| **8.** | **Testing……………………………………….……………...** | **30** |
|  | * 8.1 Testing | **30** |
|  | * 8.2 Test Cases | **31** |
| **9.** | **Chapter 9 Appendices …………………………………** | **32** |
|  | * 9.1 **Coding** | **32** |
| **10.** | **Conclusion and future scope.…………………………** | **54** |
|  | * 10.1 Conclusion | **54** |
|  | * 10.2 Future Scope | **55** |
| **.** | **REFERENCES** | **56** |

**ABSTRACT**

Heart disease is one of the most critical human diseases in the world and affects human life very badly. In heart disease, the heart is unable to push the required amount of blood to other parts of the body. Accurate and on time diagnosis of heart disease is important for heart failure prevention and treatment. The diagnosis of heart disease through traditional medical history has been considered as not reliable in many aspects. To classify the healthy people and people with heart disease, noninvasive-based methods such as machine learning are reliable and efficient. In the proposed study, we developed a machine-learning-based diagnosis system for heart disease prediction by using heart disease dataset. We used seven popular machine learning algorithms, three feature selection algorithms, the cross-validation method, and seven classifiers performance evaluation metrics such as classification accuracy, specificity, sensitivity, Matthews’ correlation coefficient, and execution time. The proposed system can easily identify and classify people with heart disease from healthy people. Additionally, receiver optimistic curves and area under the curves for each classifier was computed. We have discussed all of the classifiers, feature selection algorithms, preprocessing methods, validation method, and classifiers performance evaluation metrics used in this paper. The performance of the proposed system has been validated on full features and on a reduced set of features. The features reduction has an impact on classifiers performance in terms of accuracy and execution time of classifiers. The proposed machine-learning-based decision support system will assist the doctors to diagnosis heart patients efficiently.

**List Of Figures**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE TITLE** | **PAGE NO.** |
| **FIG 5.3** | **Linear Regression** | **18** |
| **Fig 5.7** | **Representation of Features** |  |
| **FIG 6.1** | **Data Flow Diagram** | **22** |
| **FIG 6.2** | **Use Case Model** | **23** |
| **FIG 6.3** | **Flowchart** | **24** |
| **FIG 6.4** | **Activity Diagram** | **25** |
| **FIG 7.1.3** | **Login Window** | **27** |
| **FIG 7.1.4** | **Filling Patient's Details** | **28** |
| **FIG 7.1.5** | **Predicted Output** | **28** |

**List Of Tables**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE TITLE** | **PAGE NO.** |
| **Table 7.1.1** | **Heart Disease Dataset** | **26** |
| **Table 7.1.2** | **Correlation Between Features** | **27** |
| **Table 9.2** | **Test Cases** | **31** |

**1. INTRODUCTION**

The heart disease (HD) has been considered as one of the complex and life deadliest human diseases in the world. In this disease, usually the heart is unable to push the required amount of blood to other parts of the body to fulfill the normal functionalities of the body, and due to this, ultimately the heart failure occurs. The rate of heart disease in the United States is very high. The symptoms of heart disease include shortness of breath, weakness of physical body, swollen feet, and fatigue with related signs, for example, elevated jugular venous pressure and peripheral edema caused by functional cardiac or non cardiac abnormalities. The investigation techniques in early stages used to identify heart disease were complicated, and its resulting complexity is one of the major reasons that affect the standard of life. The heart disease diagnosis and treatment are very complex, especially in the developing countries, due to the rare availability of diagnostic apparatus and shortage of physicians and others resources which affect proper prediction and treatment of heart patients. The accurate and proper diagnosis of the heart disease risk in patients is necessary for reducing their associated risks of severe heart issues and improving security of heart. The European Society of Cardiology (ESC) reported that 26 million adults worldwide were diagnosed with heart disease and 3.6 million were diagnosed every year. Approximately 50% of heart disease people suffering from HD die within initial 1-2 years, and concerned costs of heart disease management are approximately 3% of health-care financial budget.

The invasive-based techniques to the diagnosing of heart disease are based on the analysis of the patient’s medical history, physical examination report, and analysis of concerned symptoms by medical experts. All these techniques mostly cause imprecise diagnosis and often delay in the diagnosis results due to human errors. Moreover, it is more expensive and computationally complex and takes time in assessments.

**1.1 MOTIVATION**

A major challenge facing healthcare organizations (hospitals, medical centers) is the provision of quality treatments that are effective. Poor clinical decisions can lead to disastrous consequences which are therefore unacceptable. Hospitals must also minimize the cost of clinical tests. They can achieve these results by employing support systems. Most hospitals today employ some sort of hospital information systems to manage their healthcare or patient data. They can answer simple queries like? “How many surgeries had resulted in " or “What is the average age of patients who have heart disease hospital stays longer than 10 days?”etc. However, they cannot answer complex queries like “Identify the important preoperative predictors that Increase the length of hospital stay”, “Given patient records on allergies, and “Given patient records, predict the probability of patients getting a heart disease.”

Unfortunately, these data are rarely used to support clinical decision making. There is a wealth of hidden information in these data that is largely untapped. This raises an important question: How can we turn data into useful information that can enable healthcare practitioners to make intelligent clinical decisions? And also how to provide ease and quality of services to patients. This gave motivation to our project to build a system to ease the hospital procedure. This project can allow any patient to check there chances of suffering from a heart disease by simply sitting at their homes for free.

**1.2 SCOPE**

In near future Healthcare systems are moving towards Analytics and Data Science to find the prediction. It is highly useable and has bright future in it. The main reasons for better scope are:

Helps avoid human biasness.

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

•Reduce the cost of medical tests

* Helps avoid human biasness
* To implement Linear Regression Classifier that classifies the disease as per the input of the user.
* Reduce the cost of medical tests

Furthermore, 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 have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions with more accuracy and quick -performance which leads to support and help both doctor and patient.

**1.3 OBJECTIVE**

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

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

•Provides new approach to concealed patterns in the data.

•Helps avoid human biasness.

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

•Reduce the cost of medical tests.

Provides new approach to concealed patterns in the data.

•Helps avoid human biasness.

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

•Reduce the cost of medical

•Helps avoid human biasness.

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

•Reduce the cost of medical testsThe overall objective of my work will be to predict accurately with few tests and attributes the presence of heart disease. Attributes considered form the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with few attributes and faster efficiency the risk of having heart disease. Decisions are often in medical costs which affects the quality of service provided to patients.

made based on doctors’ intuition and experience rather than on the knowledge rich data hidden iin the data set and databases. This practice leads to unwanted biases, errors and excessive

The overall objective of our work will be to predict accurately with few tests and attributes the presence of heart disease. Attributes considered from the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with few attributes and faster efficiency the risk of having heart disease. Decisions are often in medical costs which affects the quality of service provided to patients.

made based on doctors’ intuition and experience rather than on the knowledge rich data hidden iin the data set and databases. This practice leads to unwanted biases, errors and excessive

**1.4 APPLICATION**

**More Accurate Diagnoses**

A major use of practical predictive analytics in medicine has been in the diagnose. Some people have a genetic risk factor and, of course, patients can now be tested for that. However, when a primary care physician can then plug his patient’s data in a predictive analytics algorithm, he/she can receive all of the latest data regarding the potential progression of heart diseases in his patient as well as the latest in treatment protocols.

**Improved Public Health and Preventive Medicine**

We now know how important early intervention is in preventing or reducing the severity of disease, especially in the area of genomics. When patients and their doctors know the treatment/monitoring protocols can begin earlier. This will, in turn, save the medical costs.

**Cost Effective**

There are many prediction systems which are very costly and also hard for doctors to install or get one. The main advantage of this prediction system is that it is free of cost and can be taken by any doctor.

**Quick medication**

Time saving is very important in the field of medical. Along with correct prediction and accuracy of disease. These two are the main factors which are kept in mind in this system. It gives quick and accurate count of disease.

**2. LITERATURE REVIEW**

**2.1 LITERATURE SURVEY**

**2.1.1 "Heart Disease Prediction System Using Data Mining Techniques"**

**- Abhishek Taneja**

In this study, the aim was to design a predictive model for heart disease detection using data mining techniques from Tran thoracic Echocardiography Report dataset that is capable of enhancing the reliability of heart disease diagnosis using echocardiography. Data collected from PGI, Chandigarh from the year 2008 to 2011 containing 7,339 instances was selected and preprocessed for this study. The models were built on the preprocessed Transthoracic Echocardiography dataset with three different supervised machine learning algorithms i.e. J48 Classifier, Naïve Bayes and Multilayer Perception using Weka 3.6.4 machine learning software.

The performances of the models were evaluated using the standard metrics of accuracy, precision, recall and F-measure. 10-Fold Cross Validation was adopted for randomly sampling the training and test data samples. All eight models performed well in predicting heart disease cases. The most effective model to predict patients with heart disease appears to be a J48 classifier implemented on selected attributes with a classification accuracy of 95.56%.

**2.1.2 "An Analysis of Heart Disease Prediction using Different Data Mining Techniques"**

**-Nidhi Bhatla and Kiran Jyoti**

The objective of their work was to provide a study of different data mining techniques that can be employed in automated heart disease prediction systems. Various techniques and data mining classifiers are defined in this work which has emerged in recent years for efficient and effective heart disease diagnosis. The analysis shows that Neural Network with 15 attributes has shown the highest accuracy i.e. 100% so far. On the other hand, Decision Tree has also performed well with 99.62% accuracy by using 15 attributes. Moreover, in combination with Genetic Algorithm and 6 attributes, Decision Tree has shown 99.2% efficiency

**2.1.3 "A Heart Disease Prediction Model using Logistic Regression"**

**- K.Sandhya Rani, M.Sai Manoj and G. Suguna Mani**

As though as identified through the logistic regression, it is more efficient than the data mining techniques as it is combinational and more complex models to increase the accuracy of predicting the early onset of cardiovascular diseases. This paper proposed a framework using combinations of SVM, logistic regression and decision trees to arrive at an accurate prediction of heart disease.

**2.1.4 " Prediction of Heart Disease Using Machine Learning Algorithms"**

**- Rajesh N , T Maneesha, Shaik Hafeez and Hari Krishna**

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 Naïve Bayes results we are getting more accurate results with probabilities of all other possibilities but due to guidance to only one solution decision trees may miss lead. Finally we can say by this experiment that Naïve Bayes is more accurate if the input data is cleaned and well maintained even though ID3 can clean it self it cannot give accurate results every time, and in this same way Naïve Bayes also will not give accurate results every time we need to consider results of different algorithms and by all its results if a prediction is made it will be accurate. But we can use Naïve Bayes consider variables as individual we can use combination of algorithms like Naïve Bayes and K-means to get accuracy

**2.1.5 " Heart Disease Prediction using Logistic Regression Algorithm using Machine Learning"**

**-Reddy Prasad, Pidaparthi Anjali, S.Adil and N.Deepa**

The amount of Heart diseases can exceed the control line and reach to maximum point. Heart disease are complicated and each and every year lots of people are dying with this disease By using this all systems one of the major drawbacks of these works is mainly focus only to the application of classify techniques and algorithms for heart disease prediction, by all these studying various data cleaning and mining techniques that prepare and build a dataset appropriate for data mining. So that I can use this Machine Learning in that logistic regression algorithms by predicting if patient has heart disease or not. Any nonmedical employee can use this software and predict the heart disease and reduce the time complexity of the doctors.

**3. PROBLEM STATEMENT**

**3.1 PROBLEM STATEMENT**

Heart disease can be managed effectively with a combination of lifestyle changes, medicine and,

in some cases, surgery. With the right treatment, the symptoms of heart disease can be reduced

and the functioning of the heart improved. The predicted results can be used to prevent and thus

reduce cost for surgical treatment and other expensive.

The overall objective of our work will be to predict accurately with few tests and attributes the

presence of heart disease. Attributes considered form the primary basis for tests and give

accurate results more or less. Many more input attributes can be taken but our goal is to predict with few attributes and faster efficiency the risk of having heart disease. Decisions are often in the data set and databases. This practice leads to unwanted biases, errors and excessive

medical costs which affects the quality of service provided to patients.

.

made based on doctors’ intuition and experience rather than on the knowledge rich data hidden

made based on doctors’ intuition and experience rather than on the knowledge rich data hidden iin the data set and databases. This practice leads to unwanted biases, errors and excessive

**4.REQUIREMENTS**

**4.1 SOFTWARE REQUIREMENT**

The following software is needed in order to make the model work efficiently:

* Visual Studio Code (v 1.42.0 or above)
* Jupypter Notebook

**4.2 HARDWARE REQUIREMENT**

The following hardware is needed in order to make the model work efficiently:

* A computer needs to be installed for the model to be deployed. .
* Secondary memory to store all the images and database.

**5. METHODOLOGY**

**5.1 METHODS**

The following subsections briefly discuss the methods of the project:

The Cleveland heart disease dataset has a sample size of 303 patients, 76 features, and some missing values. During the analysis, missing values of the dataset were pre processed. The complete information consist of 13 features of the dataset.

The methodology of the proposed system structured into five stages including

1. Preprocessing of dataset
2. Feature selection
3. cross-validation method
4. Machine learning classifiers
5. Classifiers’ performance evaluation methods.

The proposed system has been developed with the aim to classify people with heart disease and healthy people. The performances of different machine learning predictive models for heart disease diagnosis on full and selected features were tested.

**5.1 Pre processing of dataset**

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine learning model needs information in a specified format, for example, random forest algorithm does not support null values, and therefore to execute random forest algorithm null values have to be managed from the original raw data set. Another aspect is that data set should be formatted in such a way that more than one machine learning and deep learning algorithms are executed in one data set and best out of them is chosen.

**5.2 Feature Selection**

Feature selection is necessary for the machine learning process because sometimes irrelevant features affect the classification performance of the machine learning classifier. Feature selection improves the classification accuracy and reduces the model execution time.

**5.3 Machine Learning Classifiers**

In order to classify the heart patients and healthy people, machine learning classification algorithms are used.

**Linear Regression :**

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

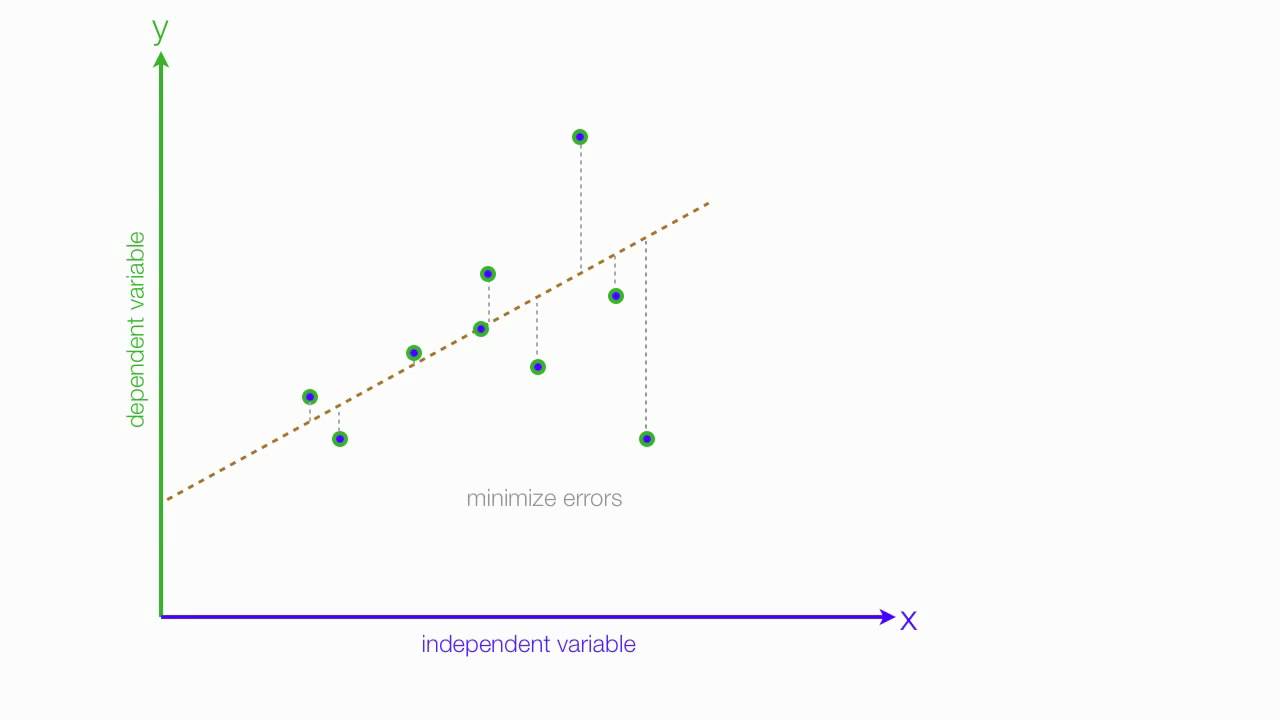


Figure 5.3 : Linear Regression

Linear regression performs the task to predict a dependent variable value(y) based on a given independent variable(x). So, this regression technique finds out a linear relationship between x(input) and y(output). Hence, the name is Linear regression.

**Decision Tree :**

Decision tree is the most powerful and popular tool for classification and prediction. A decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node(terminal node) holds a class label.

Decision tree classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, them moving down the tree branch corresponding to the value of the attribute. This process is then repeated for the sub tree rooted at the new node.

**Random forest classification :**

A random forest is and ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision tree.

Approach:

Pick at random K data points from the training set. Build the decision tree associated with those K data points. Choose the number N tree of trees you want to build and repeat steps. For a new data point, make each one of your Ntree trees predict the value of Y for the data point, and assign the new data point at the average across all of the predicted Y values.

**5.4 Validation Method of Classifiers**

In order to check the performance of the classifiers, various performance evaluation metrics were used in this research. We used confusion matrix, every observation in the testing set is predicted in exactly one box.

* Classification accuracy: accuracy shows the overall performance of the classification
* Classification error: it is the overall incorrect classification of the classification model

**5.4 classifiers’ performance evaluation methods**

Mean and standard deviation of following classifiers are:

**1. Decision Tree:**

* Mean: 0.8697027261593758
* Standard deviation: 0.10993611952914964

**2. Linear Regression:**

* Mean: 0.8697027261593758
* Standard deviation: 0.10993611952914964

**3. Random Forest Regression**

* Mean: 0.9869534947064679
* Standard deviation: 0.11563361371168596

**5.6 Python libraries used**

**5.6.1Pandas :**

Pandas is an open-source library that is built on top of Numpy library. It is a python package that offers various data structures numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance and productivity fir users. In computer programming, pandas is a software library written for the python programming language for data manipulation and analysis. In particular, it offers data structures and operation for manipulating numerical tables and time series.

**5.6.2 NumPy :**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. The NumPy module provides an nd array object using which we can use to perform operations on an array of any dimension. The nd array stands for N-dimensional array where N is any number.

**5.6.3 SKlearn :**

SKlearn of SciKit learn is an open source python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface. Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc. accessible to everybody and reusable in various contexts. It is largely written in python, and uses numpy extensively for high-performance linear algebra and array operations.

**5.6.4 Joblib :**

Joblib is a set of tools to provide light weight pipelining in python. In particular-

1. Transparent disk-catching of function and lazy re-eval000uation .
2. Easy simple parallel computing.

Joblib6 is optimized to be fast and robust on large data in particular and has specific optimization for numpy arrays.

The vision is to provide tools to easily achieve better performance and re-reducibility when working with long running jobs.

**5.7 Representation of Features**

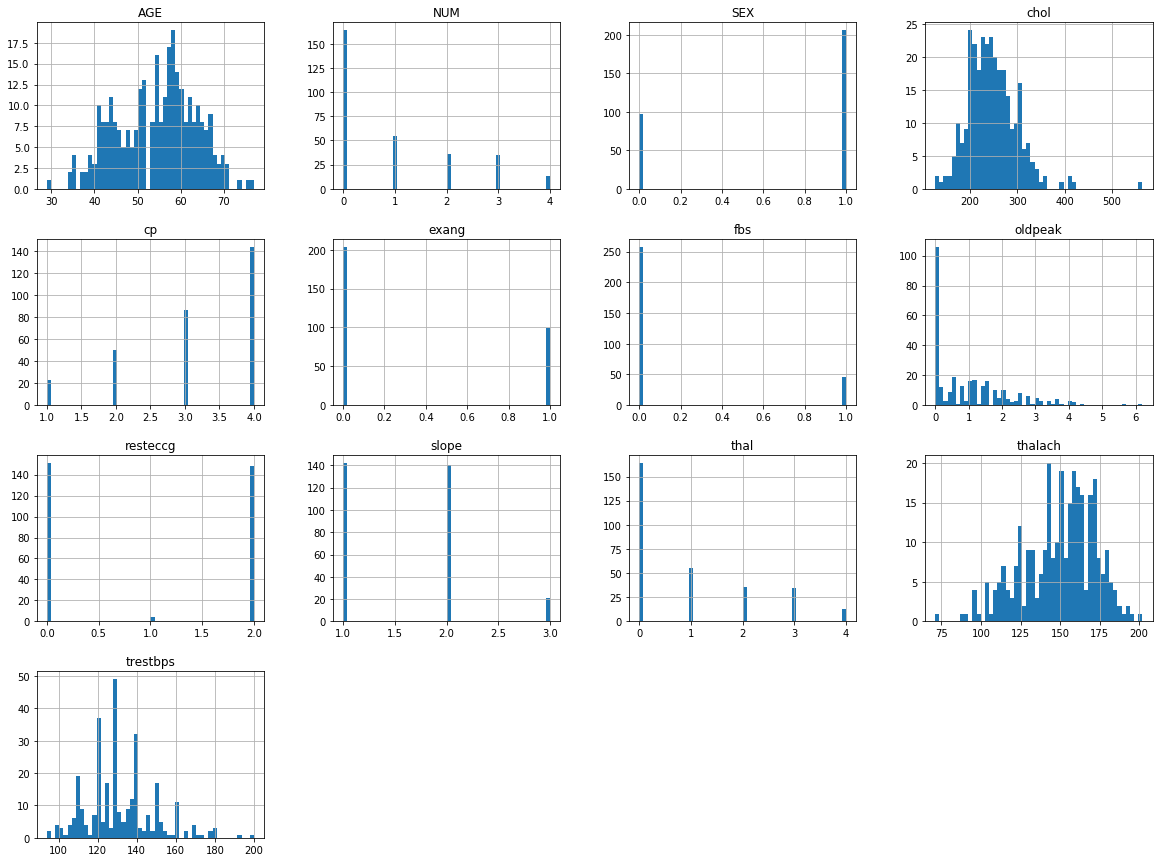


Figure 5.7 : Representation of Features

**6. DESIGN FRAMEWORK**

**6.1 Data Flow Diagram (Dfd):**

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow; there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.  
The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

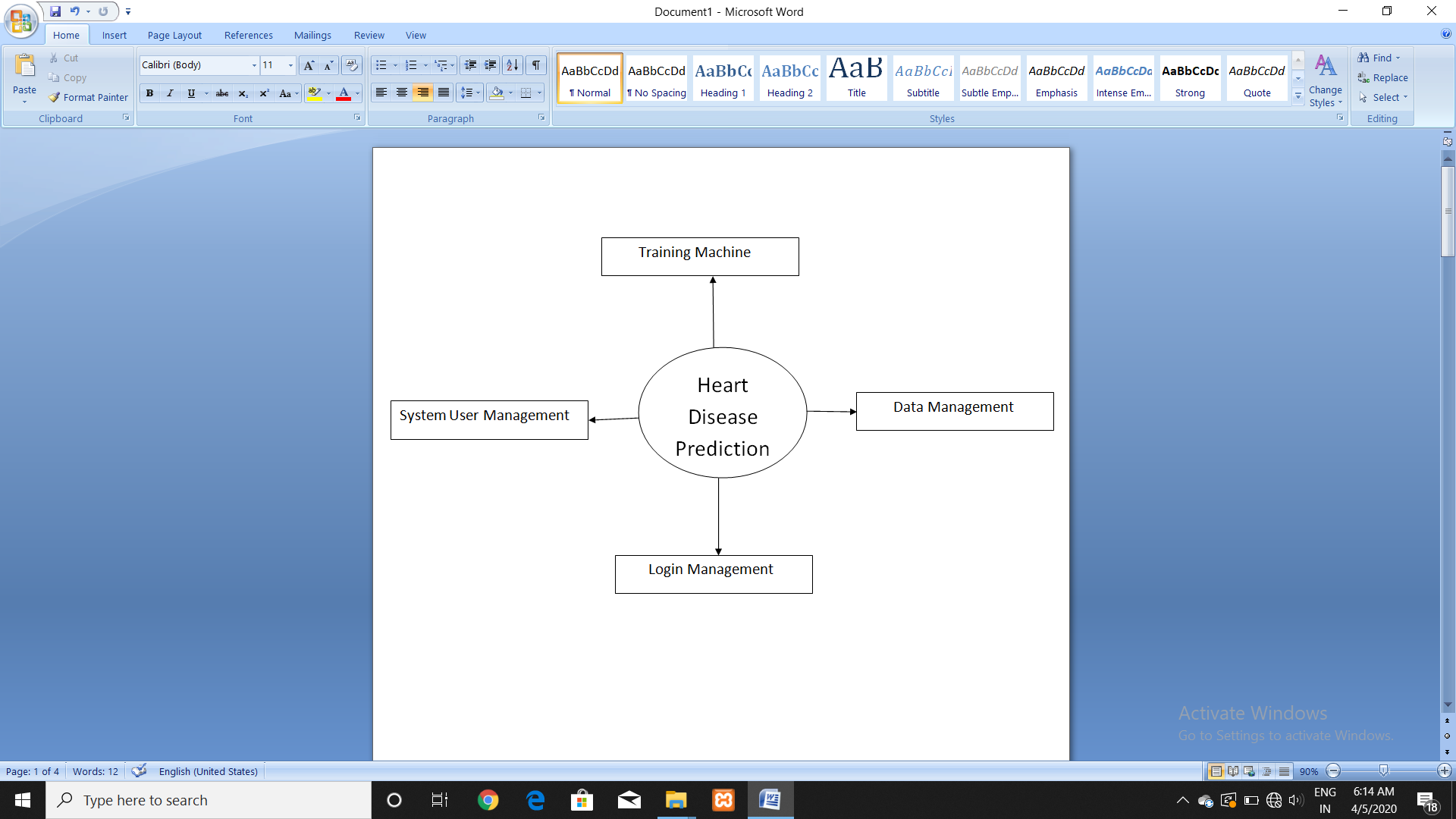


Figure 6.1 : Data Flow Diagram

**6.3 Use Case Diagram:**

In the Unified Modeling Language(UML), a use case diagram can summarize the details of your system’s users (also known as actors) and their interactions with the system. To build one, you’ll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

1. Scenarios in which your system or application interacts with people, organizations, or external systems.
2. Goals that your system or application helps those entities (known as actors) achieve.
3. The scope of your system.

Use case diagrams are idea for defining and organizing functional requirements in a system, specifying the context and requirements of a system and modeling the basic flow of events in a use case.

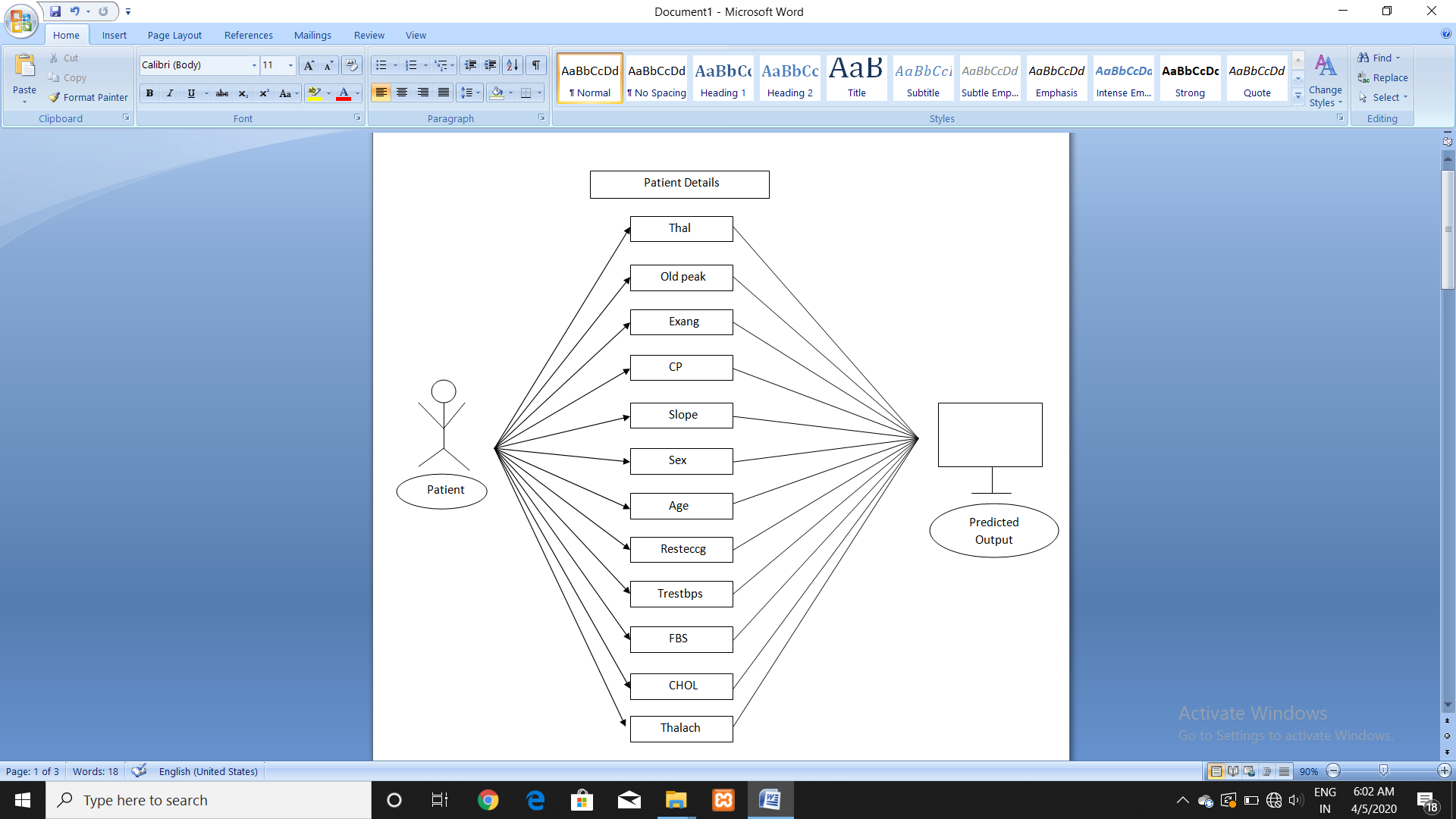


Figure 6.2 : Use Case Diagram

**6.3 FLOWCHART:**

A flowchart is a type of diagram that represents a workflow or process. A flow chart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. The diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields the oval. An End or a beginning. The oval, or terminator, is used to represent the start and end of a process. The rectangle represents a step in the flowcharting process. The rectangle is your go-to symbol once you’ve started flowcharting. The arrow indicates directional flow. The diamond indicates a decision.

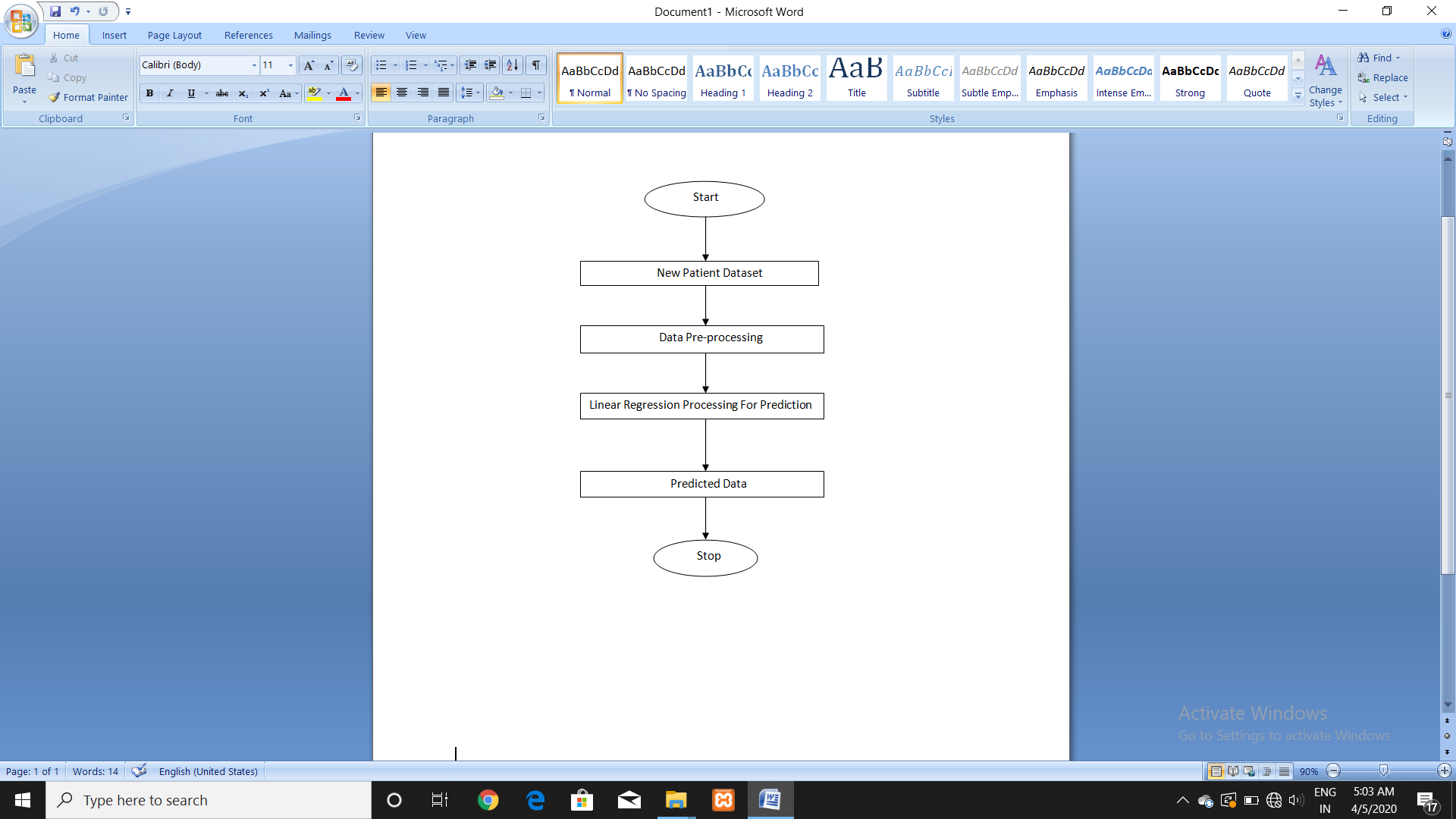


Figure 6.3 : Flowchart

**6.4 Activity Diagrams:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language(UML), activity diagrams are intended to model both computational and organizational processes(i.e. workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stories.

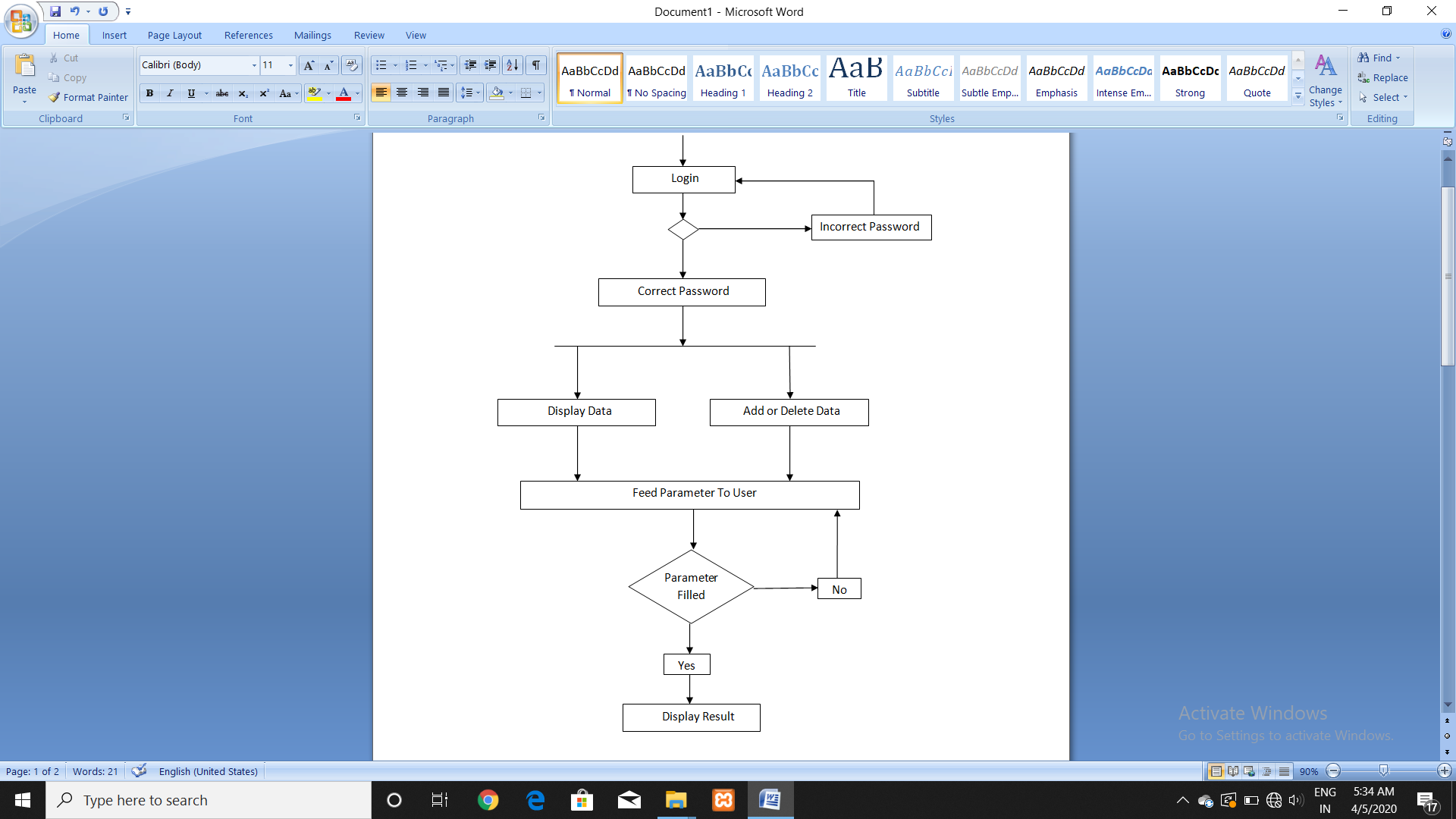


Figure 6.4 : Activity Diagram

**7. IMPLEMENTATION**

**7.1 SNAPSHOTS**

**7.1.1 Heart Disease Dataset (Features) :**

|  |  |  |
| --- | --- | --- |
| S.NO | Attribute Name | Description |
| 1 | Age | Age in years |
| 2 | Sex | Value 1: male value0: female |
| 3 | Chol | Serum cholesterol in mg/di |
| 4 | Slope | The slope of the peak exercise ST segment \* Value 1: upsloping  \*Value 2: flat \* Value 3: downsloping |
| 5 | Thal | 3 = normal; 6 = fixed defect; 7 = reversible defect |
| 6 | Cp | Chest pain type \*Value 1: typical angina \* Value 2: atypical angina \* Value 3: non-angina pain \* Value 4: asymptomatic |
| 7 | Resteccg | Resting electrocardiographic results |
| 8 | Thalach | Maximum heart rate achieved |
| 9 | Fbs | (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) |
| 10 | Oldpeak | ST depression induced by exercise relative to rest |
| 11 | Exang | exercise induced angina (1 = yes; 0 = no) |
| 12 | Trestbps | resting blood pressure (in mm Hg on admission to the hospital) |

Table 7.1.1 : Heart Disease Datasets

**7.1.2 Correlation between the datasets :**

Following figures represents the relation among all the datasets considering NUM to be the predicted output (required).

|  |  |
| --- | --- |
| **Attribute name** | **correlation** |
| **num** | **1.000000** |
| **thal** | **1.000000** |
| **oldspeak** | **0.473498** |
| **Exang** | **0.402233** |
| **Cp** | **0.390256** |
| **Slope** | **0.382420** |
| **Sex** | **0.254034** |
| **Age** | **0.207063** |
| **Resteccg** | **0.180464** |
| **Trestbps** | **0.175432** |
| **Fbs** | **0.071861** |
| **Chol** | **0.009405** |
| **thalach** | **-0.423791** |

Table 7.1.2 : Correlation among Datasets

**7.1.3 Login Window For User :**

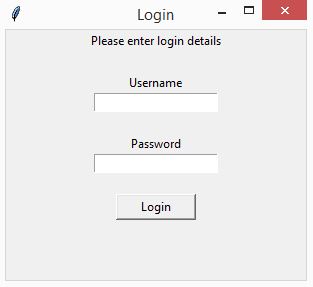


Figure 7.1.3 : Login window

**7.1.4 Filling Patient's Detail :**

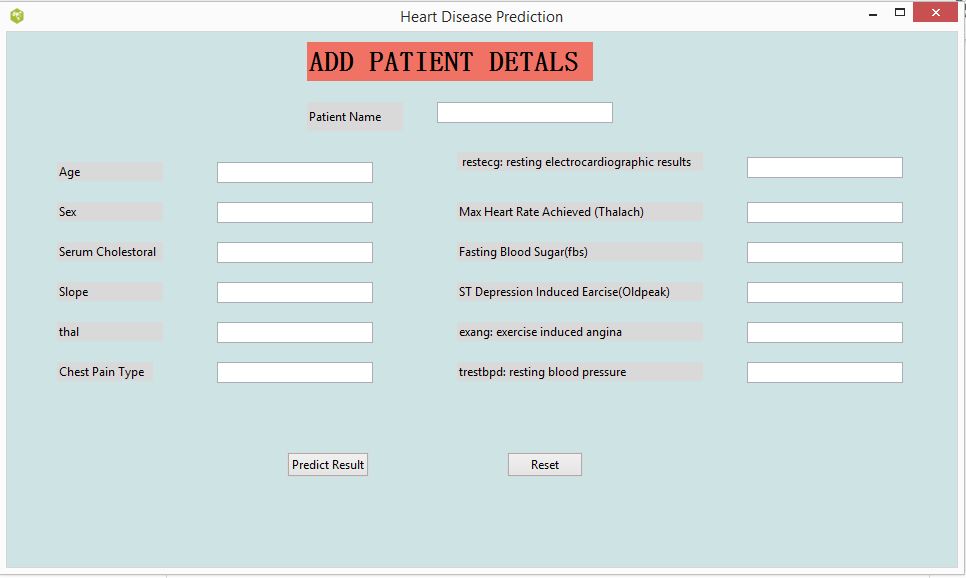


Figure 7.1.4 : Filling Patient's details

**7.1.5 Predicted Output :**



**7.2 SOFTWARE MAINTAINANCE**

It is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software application after delivery to correct faults and to improve performance.

**7.2.1 Need for Maintenance :**

Software Maintenance must be performed in order to:

* Correct faults.
* Improve the design.
* Implement enhancements.
* Interface with other systems.
* Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
* Migrate legacy software.
* Retire software.

**7.2.2 Categories of Software Maintenance :**

**Corrective maintenance:** Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

**Adaptive maintenance:** This includes modifications and updating when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

**Perfective maintenance:** A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

**Preventive maintenance:** This type of maintenance includes modifications and updating to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

**8. TESTING**

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software

**8.1 PRINCIPLES OF TESTING :**

* All the test should meet the customer requirements
* To make our software testing should be performed by third party
* Exhaustive testing is not possible . As we need the optimal amount of testing based on the risk assessment of the application.
* All the test to be conducted should be planned before implementing
* It follows pareto rule (80/20 rule) which states that 80% of errors comes from 20% of program components.
* Start testing with small parts and extend it to large parts.

**8.2 TYPES OF TESTING:**

**8.2.1. Unit Testing :**

It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs.

**8.2.2. Regression Testing :**

Every time new module is added leads to changes in program. This type of testing make sure that whole component works properly even after adding components to the complete program.

**8.2.3. System Testing :**

In this software is tested such that it works fine for different operating system. It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working.

**8.2.4. Stress Testing :**

In this we gives unfavorable conditions to the system and check how they perform in those condition.

**8.2.5. Performance Testing :**

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test speed and effectiveness of program.

**8.1 TEST CASES :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Number** | **Action** | **Input** | **Expected**  **output** | **Actual**  **output** | **Test**  **Result** |
| **1** | **Pre-processing dataset** | **Improper data set** | **Proper data set** | **Proper data set** | **passed** |
| **2** | **Updated result** | **Adding**  **Patient details** | **Predicted**  **Data** | **Predicted**  **Data** | **passed** |
| **3** | **Training machine** | **Dataset** | **Predicted result** | **Predicted**  **Result** | **passed** |
| **4** | **Processing given data** | **Discrete &**  **Continuous data** | **Predicted**  **Result** | **Predicted result** | **Passed** |
| **5** | **Processing given data** | **Either all continuous or all discrete data** | **Predicted**  **Proper result** | **Error or improper output** | **failed** |

Table 8.1 : Test Cases

**9. APPENDICES**

**9.1 CODING**

**9.1.1 Importing libraries and Dataset**

import pandas as pd

Disease = pd.read\_csv("Disease.csv")

Disease = Disease.replace(to\_replace ="?", value ="")

**9.1.2 Training Model**

import numpy as np

def split\_train\_test(data, test\_ratio):

np.random.seed(42)

shuffled = np.random.permutation(len(data))

print(shuffled)

test\_set\_size = int(len(data) \* test\_ratio)

test\_indices = shuffled[:test\_set\_size]

train\_indices = shuffled[test\_set\_size:]

return data.iloc[train\_indices], data.iloc[test\_indices]

from sklearn.model\_selection import train\_test\_split

train\_set, test\_set = train\_test\_split(Disease, test\_size=0.2, random\_state=42)

print(f"Rows in train set: {len(train\_set)}\nRows in test set: {len(test\_set)}\n")

from sklearn.model\_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n\_splits=1, test\_size=0.2, random\_state=42)

for train\_index, test\_index in split.split(Disease, Disease['fbs']):

strat\_train\_set = Disease.loc[train\_index]

strat\_test\_set = Disease.loc[test\_index]

Disease = strat\_train\_set.copy()

Disease = strat\_train\_set.drop("NUM", axis=1)

Disease\_labels = strat\_train\_set["NUM"].copy()

**9.1.3 Processing data by filling the missing values**

Disease= pd.get\_dummies(Disease)

median = Disease["AGE"].median()

**9.1.4 Importing median value to missing data**

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")

imputer.fit(Disease)

X = imputer.transform(Disease)

Disease\_tr = pd.DataFrame(X, columns=Disease.columns)

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

my\_pipeline = Pipeline([

('imputer', SimpleImputer(strategy="median")),

('std\_scaler', StandardScaler()),

])

Disease\_num\_tr = my\_pipeline.fit\_transform(Disease)

**9.1.5 Testing different classifications to implement**

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

model = LinearRegression()

#model = DecisionTreeRegressor()

#model = RandomForestRegressor()

model.fit(Disease\_num\_tr, Disease\_labels)

some\_data = Disease.iloc[:5]

some\_labels = Disease\_labels.iloc[:5]

prepared\_datai = my\_pipeline.transform(some\_data)

**9.1.6 Checking mean squared error**

from sklearn.metrics import mean\_squared\_error

Disease\_predictions = model.predict(Disease\_num\_tr)

mse = mean\_squared\_error(Disease\_labels, Disease\_predictions)

rmse = np.sqrt(mse)

from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(model, Disease\_num\_tr, Disease\_labels, scoring="neg\_mean\_squared\_error", cv=10)

rmse\_scores = np.siqrt(-scores)

**9.1.7 Final Prediction**

from joblib import dump, load

dump(model, 'Disease.joblib')

X\_test = strat\_test\_set.drop("NUM", axis=1)

Y\_test = strat\_test\_set["NUM"].copy()

X\_test = pd.get\_dummies(X\_test)

X\_test\_prepared = my\_pipeline.transform(X\_test)

final\_predictions = model.predict(X\_test\_prepared)

final\_mse = mean\_squared\_error(Y\_test, final\_predictions)

final\_rmse = np.sqrt(final\_mse)

# print(final\_predictions, list(Y\_test))

from joblib import dump, load

import numpy as np

model = load('Disease.joblib')

features = np.array([[-1.54682375, 0.68964466, -0.16363892, -1.14727992, 0.09776352,

-0.41803981, -1.05309766, 1.30786643, -0.70929937, -0.87416525,

-0.94627915, -0.77354494]])

model.predict(features)

**# output = array([2.61351962e-09])**

**9.1.8 GUI For Log-In Window**

from tkinter import \*

def LoginPage():

login\_screen=Tk()

login\_screen.title("Login")

login\_screen.geometry("300x250")

Label(login\_screen, text="Please enter login details").pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Username").pack()

username\_login\_entry = Entry(login\_screen, textvariable="username")

username\_login\_entry.pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Password").pack()

password\_\_login\_entry = Entry(login\_screen, textvariable="password", show= '\*')

password\_\_login\_entry.pack()

Label(login\_screen, text="").pack()

Button(login\_screen, text="Login", width=10, height=1).pack()

login\_screen.mainloop() LoginPage()

**9.1.9 GUI For Filling Patient's Details**

import sys

try:

import Tkinter as tk

except ImportError:

import tkinter as tk

try:

import ttk

py3 = False

except ImportError:

import tkinter.ttk as ttk

py3 = True

def Predict():

print('unknown\_support.Predict')

sys.stdout.flush()

def init(top, gui, \*args, \*\*kwargs):

global w, top\_level, root

w = gui

top\_level = top

root = top

def destroy\_window():

global top\_level

top\_level.destroy()

top\_level = None

if \_\_name\_\_ == '\_\_main\_\_':

import unknown

unknown.vp\_start\_gui()

import sys

try:

import Tkinter as tk

except ImportError:

import tkinter as tk

try:

import ttk

py3 = False

except ImportError:

import tkinter.ttk as ttk

py3 = True

import unknown\_support

def vp\_start\_gui():

'''Starting point when module is the main routine.'''

global val, w, root

root = tk.Tk()

top = Toplevel1 (root)

unknown\_support.init(root, top)

root.mainloop()

w = None

def create\_Toplevel1(root, \*args, \*\*kwargs):

'''Starting point when module is imported by another program.'''

global w, w\_win, rt

rt = root

w = tk.Toplevel (root)

top = Toplevel1 (w)

unknown\_support.init(w, top, \*args, \*\*kwargs)

return (w, top)

def destroy\_Toplevel1():

global w

w.destroy()

w = None

class Toplevel1:

def \_\_init\_\_(self, top=None):

'''This class configures and populates the toplevel window.

top is the toplevel containing window.'''

\_bgcolor = '#d9d9d9' # X11 color: 'gray85'

\_fgcolor = '#000000' # X11 color: 'black'

\_compcolor = '#d9d9d9' # X11 color: 'gray85'

\_ana1color = '#d9d9d9' # X11 color: 'gray85'

\_ana2color = '#ececec' # Closest X11 color: 'gray92'

font9 = "-family {MS Mincho} -size 21 -weight bold -slant " \

"roman -underline 0 -overstrike 0"

self.style = ttk.Style()

if sys.platform == "win32":

self.style.theme\_use('winnative')

self.style.configure('.',background=\_bgcolor)

self.style.configure('.',foreground=\_fgcolor)

self.style.configure('.',font="TkDefaultFont")

self.style.map('.',background=

[('selected', \_compcolor), ('active',\_ana2color)])

top.geometry("950x535+244+129")

top.minsize(124, 1)

top.maxsize(1362, 741)

top.resizable(1, 1)

top.title("Heart Disease Prediction")

top.configure(background="#cde3e4")

self.TLabel1 = ttk.Label(top)

self.TLabel1.place(relx=0.316, rely=0.019, height=39, width=286)

self.TLabel1.configure(background="#f07264")

self.TLabel1.configure(foreground="#000000")

self.TLabel1.configure(font=font9)

self.TLabel1.configure(relief="flat")

self.TLabel1.configure(text='''ADD PATIENT DETALS''')

self.TLabel2 = ttk.Label(top)

self.TLabel2.place(relx=0.316, rely=0.131, height=29, width=96)

self.TLabel2.configure(background="#d9d9d9")

self.TLabel2.configure(foreground="#000000")

self.TLabel2.configure(font="TkDefaultFont")

self.TLabel2.configure(relief="flat")

self.TLabel2.configure(text='''Patient Name''')

self.TLabel2\_1 = ttk.Label(top)

self.TLabel2\_1.place(relx=0.053, rely=0.243, height=19, width=106)

self.TLabel2\_1.configure(background="#d9d9d9")

self.TLabel2\_1.configure(foreground="#000000")

self.TLabel2\_1.configure(font="TkDefaultFont")

self.TLabel2\_1.configure(relief="flat")

self.TLabel2\_1.configure(text='''Age''')

self.TLabel2\_2 = ttk.Label(top)

self.TLabel2\_2.place(relx=0.474, rely=0.617, height=19, width=246)

self.TLabel2\_2.configure(background="#d9d9d9")

self.TLabel2\_2.configure(foreground="#000000")

self.TLabel2\_2.configure(font="TkDefaultFont")

self.TLabel2\_2.configure(relief="flat")

self.TLabel2\_2.configure(text='''trestbpd: resting blood pressure''')

self.menubar = tk.Menu(top,font="TkMenuFont",bg=\_bgcolor,fg=\_fgcolor)

top.configure(menu = self.menubar)

self.TLabel2\_3 = ttk.Label(top)

self.TLabel2\_3.place(relx=0.053, rely=0.393, height=19, width=106)

self.TLabel2\_3.configure(background="#d9d9d9")

self.TLabel2\_3.configure(foreground="#000000")

self.TLabel2\_3.configure(font="TkDefaultFont")

self.TLabel2\_3.configure(relief="flat")

self.TLabel2\_3.configure(text='''Serum Cholestoral''')

self.TLabel2\_4 = ttk.Label(top)

self.TLabel2\_4.place(relx=0.474, rely=0.224, height=19, width=246)

self.TLabel2\_4.configure(background="#d9d9d9")

self.TLabel2\_4.configure(foreground="#000000")

self.TLabel2\_4.configure(font="TkDefaultFont")

self.TLabel2\_4.configure(relief="flat")

self.TLabel2\_4.configure(text='''restecg: resting electrocardiographic results''')

self.TLabel2\_5 = ttk.Label(top)

self.TLabel2\_5.place(relx=0.474, rely=0.542, height=19, width=246)

self.TLabel2\_5.configure(background="#d9d9d9")

self.TLabel2\_5.configure(foreground="#000000")

self.TLabel2\_5.configure(font="TkDefaultFont")

self.TLabel2\_5.configure(relief="flat")

self.TLabel2\_5.configure(text='''exang: exercise induced angina''')

self.TLabel2\_6 = ttk.Label(top)

self.TLabel2\_6.place(relx=0.053, rely=0.542, height=19, width=106)

self.TLabel2\_6.configure(background="#d9d9d9")

self.TLabel2\_6.configure(foreground="#000000")

self.TLabel2\_6.configure(font="TkDefaultFont")

self.TLabel2\_6.configure(relief="flat")

self.TLabel2\_6.configure(text='''thal''')

self.TLabel2\_7 = ttk.Label(top)

self.TLabel2\_7.place(relx=0.053, rely=0.467, height=19, width=106)

self.TLabel2\_7.configure(background="#d9d9d9")

self.TLabel2\_7.configure(foreground="#000000")

self.TLabel2\_7.configure(font="TkDefaultFont")

self.TLabel2\_7.configure(relief="flat")

self.TLabel2\_7.configure(text='''Slope''')

self.TLabel2\_8 = ttk.Label(top)

self.TLabel2\_8.place(relx=0.053, rely=0.617, height=19, width=96)

self.TLabel2\_8.configure(background="#d9d9d9")

self.TLabel2\_8.configure(foreground="#000000")

self.TLabel2\_8.configure(font="TkDefaultFont")

self.TLabel2\_8.configure(relief="flat")

self.TLabel2\_8.configure(text='''Chest Pain Type''')

self.TLabel2\_9 = ttk.Label(top)

self.TLabel2\_9.place(relx=0.053, rely=0.318, height=19, width=106)

self.TLabel2\_9.configure(background="#d9d9d9")

self.TLabel2\_9.configure(foreground="#000000")

self.TLabel2\_9.configure(font="TkDefaultFont")

self.TLabel2\_9.configure(relief="flat")

self.TLabel2\_9.configure(text='''Sex''')

self.TLabel2\_10 = ttk.Label(top)

self.TLabel2\_10.place(relx=0.474, rely=0.393, height=19, width=246)

self.TLabel2\_10.configure(background="#d9d9d9")

self.TLabel2\_10.configure(foreground="#000000")

self.TLabel2\_10.configure(font="TkDefaultFont")

self.TLabel2\_10.configure(relief="flat")

self.TLabel2\_10.configure(text='''Fasting Blood Sugar(fbs)''')

self.TLabel2\_11 = ttk.Label(top)

self.TLabel2\_11.place(relx=0.474, rely=0.467, height=19, width=246)

self.TLabel2\_11.configure(background="#d9d9d9")

self.TLabel2\_11.configiure(foreground="#000000")

self.TLabel2\_11.configure(font="TkDefaultFont")

self.TLabel2\_11.configure(relief="flat")

self.TLabel2\_11.configure(text='''ST Depression Induced Earcise(Oldpeak)''')

self.TLabel2\_12 = ttk.Label(top)

self.TLabel2\_12.place(relx=0.474, rely=0.318, height=19, width=246)

self.TLabel2\_12.configure(background="#d9d9d9")

self.TLabel2\_12.configure(foreground="#000000")

self.TLabel2\_12.configure(font="TkDefaultFont")

self.TLabel2\_12.configure(relief="flat")

self.TLabel2\_12.configure(text='''Max Heart Rate Achieved (Thalach)''')

self.TButton1 = ttk.Button(top)

self.TButton1.place(relx=0.295, rely=0.785, height=25, width=82)

self.TButton1.configure(command=unknown\_support.Predict)

self.TButton1.configure(takefocus="")

self.TButton1.configure(text='''Predict Result''')

self.TButton2 = ttk.Button(top)

self.TButton2.place(relx=0.526, rely=0.785, height=25, width=76)

self.TButton2.configure(takefocus="")

self.TButton2.configure(text='''Reset''')

self.TEntry1 = ttk.Entry(top)

self.TEntry1.place(relx=0.221, rely=0.243, relheight=0.039

, relwidth=0.164)

self.TEntry1.configure(takefocus="")

self.TEntry1.configure(cursor="ibeam")

self.TEntry1\_13 = ttk.Entry(top)

self.TEntry1\_13.place(relx=0.221, rely=0.318, relheight=0.039

, relwidth=0.164)

self.TEntry1\_13.configure(takefocus="")

self.TEntry1\_13.configure(cursor="ibeam")

self.TEntry1\_14 = ttk.Entry(top)

self.TEntry1\_14.place(relx=0.221, rely=0.393, relheight=0.039

, relwidth=0.164)

self.TEntry1\_14.configure(takefocus="")

self.TEntry1\_14.configure(cursor="ibeam")

self.TEntry1\_15 = ttk.Entry(top)

self.TEntry1\_15.place(relx=0.221, rely=0.467, relheight=0.039

, relwidth=0.164)

self.TEntry1\_15.configure(takefocus="")

self.TEntry1\_15.configure(cursor="ibeam")

self.TEntry1\_16 = ttk.Entry(top)

self.TEntry1\_16.place(relx=0.221, rely=0.542, relheight=0.039

, relwidth=0.164)

self.TEntry1\_16.configure(takefocus="")

self.TEntry1\_16.configure(cursor="ibeam")

self.TEntry1\_17 = ttk.Entry(top)

self.TEntry1\_17.place(relx=0.221, rely=0.617, relheight=0.039

, relwidth=0.164)

self.TEntry1\_17.configure(takefocus="")

self.TEntry1\_17.configure(cursor="ibeam")

self.TEntry1\_18 = ttk.Entry(top)

self.TEntry1\_18.place(relx=0.453, rely=0.131, relheight=0.039

, relwidth=0.185)

self.TEntry1\_18.configure(takefocus="")

self.TEntry1\_18.configure(cursor="ibeam")

self.TEntry1\_19 = ttk.Entry(top)

self.TEntry1\_19.place(relx=0.779, rely=0.234, relheight=0.039

, relwidth=0.164)

self.TEntry1\_19.configure(takefocus="")

self.TEntry1\_19.configure(cursor="ibeam")

self.TEntry1\_20 = ttk.Entry(top)

self.TEntry1\_20.place(relx=0.779, rely=0.318, relheight=0.039

, relwidth=0.164)

self.TEntry1\_20.configure(takefocus="")

self.TEntry1\_20.configure(cursor="ibeam")

self.TEntry1\_21 = ttk.Entry(top)

self.TEntry1\_21.place(relx=0.779, rely=0.393, relheight=0.039

, relwidth=0.164)

self.TEntry1\_21.configure(takefocus="")

self.TEntry1\_21.configure(cursor="ibeam")

self.TEntry1\_22 = ttk.Entry(top)

self.TEntry1\_22.place(relx=0.779, rely=0.467, relheight=0.039

, relwidth=0.164)

self.TEntry1\_22.configure(takefocus="")

self.TEntry1\_22.configure(cursor="ibeam")

self.TEntry1\_23 = ttk.Entry(top)

self.TEntry1\_23.place(relx=0.779, rely=0.542, relheight=0.039

, relwidth=0.164)

self.TEntry1\_23.configure(takefocus="")

self.TEntry1\_23.configure(cursor="ibeam")

self.TEntry1\_24 = ttk.Entry(top)

self.TEntry1\_24.place(relx=0.779, rely=0.617, relheight=0.039

, relwidth=0.164)

self.TEntry1\_24.configure(takefocus="")

self.TEntry1\_24.configure(cursor="ibeam")

if \_\_name\_\_ == '\_\_main\_\_':

vp\_start\_gui()

**9.1.10 GUI For Predicted Result**

import sys

try:

import Tkinter as tk

exceipt ImportError:

import tkinter as tk

try:

import ttk

py3 = False

except ImportError:

import tkinter.ttk as ttk

py3 = True

def init(top, gui, \*args, \*\*kwargs):

global w, top\_level, root

w = gui

top\_level = top

root = top

def destroy\_window():

global top\_level

top\_level.destroy()

top\_level = None

if \_\_name\_\_ == '\_\_main\_\_':

import 1

1.vp\_start\_gui()

import sys

try:

import Tkinter as tk

except ImportError:

import tkinter as tk

try:

import ttk

py3 = False

except ImportError:

import tkinter.ttk as ttk

py3 = True

import 1\_support

def vp\_start\_gui():

'''Starting point when module is the main routine.'''

global val, w, root

root = tk.Tk()

top = Toplevel1 (root)

1\_support.init(root, top)

root.mainloop()

w = None

def create\_Toplevel1(root, \*args, \*\*kwargs):

'''Starting point when module is imported by another program.'''

global w, w\_win, rt

rt = root

w = tk.Toplevel (root)

top = Toplevel1 (w)

1\_support.init(w, top, \*args, \*\*kwargs)

return (w, top)

def destroy\_Toplevel1():

global w

w.destroy()

w = None

class Toplevel1:

def \_\_init\_\_(self, top=None):

'''This class configures and populates the toplevel window.

top is the toplevel containing window.'''

\_bgcolor = '#d9d9d9' # X11 color: 'gray85'

\_fgcolor = '#000000' # X11 color: 'black'

\_compcolor = '#d9d9d9' # X11 color: 'gray85'

\_ana1color = '#d9d9d9' # X11 color: 'gray85'

\_ana2color = '#ececec' # Closest X11 color: 'gray92'

font10 = "-family {Segoe UI Light} -size 20 -weight bold " \

"-slant roman -underline 1 -overstrike 0"

font11 = "-family Gadugi -size 12 -weight bold -slant roman " \

"-underline 0 -overstrike 0"

self.style = ttk.Style()

if sys.platform == "win32":

self.style.theme\_use('winnative')

self.style.configure('.',background=\_bgcolor)

self.style.configure('.',foreground=\_fgcolor)

self.style.configure('.',font="TkDefaultFont")

self.style.map('.',background=

[('selected', \_compcolor), ('active',\_ana2color)])

top.geometry("727x450+305+131")

top.minsize(124, 1)

top.maxsize(1362, 741)

top.resizable(1, 1)

top.title("Prediction")

top.configure(background="#cae8e4")

self.TLabel1 = ttk.Label(top)

self.TLabel1.place(relx=0.028, rely=0.089, height=49, width=646)

self.TLabel1.configure(background="#bec8f3")

self.TLabel1.configure(foreground="#cd5277")

self.TLabel1.configure(font=font10)

self.TLabel1.configure(relief="flat")

self.TLabel1.configure(text='''Your Changes Of Suffering From a Heart Disease Is:''')

self.TButton1 = ttk.Button(top)

self.TButton1.place(relx=0.33, rely=0.4, height=45, width=176)

self.TButton1.configure(takefocus="")

self.TButton1.configure(text='''Predicted Result''')

self.TLabel2 = ttk.Label(top)

self.TLabel2.place(relx=0.371, rely=0.222, height=19, width=106)

self.TLabel2.configure(background="#d9d9d9")

self.TLabel2.configure(foreground="#000000")

self.TLabel2.configure(font="TkDefaultFont")

self.TLabel2.configure(relief="flat")

self.TLabel2.configure(text='''On a scale of 0-4''')

self.TLabel3 = ttk.Label(top)

self.TLabel3.place(relx=0.069, rely=0.622, height=23, width=596)

self.TLabel3.configure(background="#e4cde2")

self.TLabel3.configure(foreground="#c6592b")

self.TLabel3.configure(font=font11)

self.TLabel3.configure(relief="flat")

self.TLabel3.configure(text='''If Predicted Result< 2 : you are not suffering from any heart disease!!''')

self.TLabel3\_1 = ttk.Label(top)

self.TLabel3\_1.place(relx=0.065, rely=0.764, height=23, width=596)

self.TLabel3\_1.configure(background="#e4cde2")

self.TLabel3\_1.configure(foreground="#c6592b")

self.TLabel3\_1.configure(font="-family {Gadugi} -size 12 -weight bold")

self.TLabel3\_1.configure(relief="flat")

self.TLabel3\_1.configure(text='''If Predicted Result> 2 : you have chances of suffering from heart disease!!''')

if \_\_name\_\_ == '\_\_main\_\_':

vp\_start\_gui()

**10. CONCLUSION AND FUTURE SCOPE**

**10.1 CONCLUSION:**

Heart Diseases is one of the major concerns for our society today. It is difficult to manually determine the chance or rate of  heart disease in people and that’s why then our system or , machine learning techniques are useful to predict the rate or chances of this .and the output from existing data.

Our system analyses the accuracy of prediction of heart disease using

Several algorithm such as :-

**1**. **Decision tree**

**2. linear regression and**

**3. random forest regression**

The heart disease dataset from the UCI machine learning repository was utilized for training and testing purposes. The ensemble algorithms to produce result for different cases and the accuracy rate of our system is maximum up to 72.6%.

As though as identified through the linear regression, it is more efficient than the data mining techniques as it is combinational and more complex models to increase the accuracy of predicting the early onset of cardiovascular diseases. Heart disease are complicated and each and every year lots of people are dying with this disease By using this all systems one of the major drawbacks of these works is mainly focus only to the application of classify techniques and algorithms for heart disease prediction, by all these studying various data cleaning techniques that prepare and build a dataset appropriate for linear regression. So that we can use this Machine Learning in that linear regression algorithms by predicting if patient has heart disease or not. Any non-medical employee can use this software and predict the heart disease and reduce the time complexity of the doctors.

**10.2 FUTURE SCOPE :**

In this we have studied various classification algorithms that can be used for classification of heart disease databases, also we have seen different techniques that can be used for classification and analyze the data set and the accuracy obtained by them.

In future we will be working on further improvement of the model such as:

**1**.The accurateness of the structure can be further upgraded by creating various combinations of data mining techniques and by parameter tuning also.

**2.** The future scope of the project is the prediction of heart diseases by using advanced techniques and algorithms in less time complexity.

**3**.For predicting the heart disease we can use other techniques like data mining and different type of other algorithm of machine learning.

**4**.In future in order to upgrade the efficiency , we use large number of data set for better predicting.

**REFERENCES**

1. <http://archive.ics.uci.edu/ml/datasets/Heart+Disease>
2. <https://www.hindawi.com/journals/misy/2018/3860146/>
3. <https://www.geeksforgeeks.org>
4. <https://jamanetwork.com/journals/jama/fullarticle/193997>
5. Heart Disease Prediction System Using Data Mining Techniques Abhishek Taneja Department of Computer Science, S.A. Jain College, Ambala City, India. (Received: November 15, 2013; Accepted: November 25, 2013)
6. An Analysis of Heart Disease Prediction using Different Data Mining Techniques Nidhi Bhatla Kiran Jyoti GNDEC, Ludhiana, India GNDEC, Ludhiana, India International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 8, October - 2012
7. A Heart Disease Prediction Model using Logistic Regression, K. Sandhya Rani, Asst Prof, M. Sai Mani and G. Suguna Mani, Dhanekula Institute of Engineering and Technology, Vijaywada, Andhra Pradesh, India
8. Prediction of Heart Disease Using Machine Learning Algorithms Rajesh N, T Maneesha, Shaik Hafeez , Hari Krishna, Computer Science Engineering, K L E F, Guntur, India
9. Heart Disease Prediction using Logistic Regression Algorithm using Machine Learning Reddy Prasad,Pidaparthi Anjali, S.Adil, N.Deepa , International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019