

Chapter 1

INTRODUCTION

With the increasing capacity to control irresistible, and dietary ailments there has come the acknowledgment that hereditary illnesses are a noteworthy reason for handicap, demise, and human catastrophe. By identifying the genes underlying the human disease can contribute to developing better treatment and prevention strategies for disease and is critical to biomedical progress.

1.1 PURPOSE

In most cases, diagnosis of a disease occurs after the onset of it. It is simply based on symptoms, physical observation and biometric readings. In many cases, diagnosis occurs after the worst has occurred, and in many cases, especially In case of Genetic Diseases, mathematical prediction can be used to foresee a disease even before it has occurred.

A predictive analysis not only helps aware a person of possible disease occurrence but also helps in prescribing preventive medical treatment or simple lifestyle changes to prevent the havoc from occurring. Several chronic diseases like Diabetes, Alzheimer, Arrhythmia, and several Cardiovascular diseases are Genetic in nature.

In our project, we intend to build a framework which can Predict the hazard factor of having coronary illness based on candidate gene, and family history of an individual.

1.2 SCOPE

The machine learning model developed in this project mainly finds its scope in medical diagnosis sciences where it aids in predicting and treating a disease even before it has occurred. It will help medical practitioners arrive at a diagnostic conclusion by simple and accurate mathematical prediction, and save efforts gone into rigorous biomedical lab testing and physical observation.

It also finds its scope in many other systems where the identity of people is stored using their genetic data, and a general population analysis can be done by health services of a government.

1.3 LITERATURE SURVEY

Researchers have been applying distinctive information mining strategies to help health care professionals with enhanced precision in the diagnosis of coronary illness. Neural Networks, Naïve Bayes, Genetic algorithm, Decision Tree, classification via clustering etc. are a few methods utilized as a part of the finding the diagnosis of heart disease. jabbar[1] , Dr. Priti Chandra , Dr. B. L. Deekshatuluc Heart Disease Prediction System utilizing Associated Classification and Genetic Algorithm Cooperative characterization is a current and fulfilling method which coordinates association rule mining and classification to a model for forecast and accomplishes most extreme exactness. Associative classifiers are particularly fit to applications where maximum accuracy is wanted to demonstrate a forecast. There are numerous domains, for example, medicals where the greatest precision of the model is wanted. Mrs .G. Subbalakshmi[2] and Mr. K. Ramesh Proposed "Decision Support in Heart Disease Prediction system utilizing Naive Bayes ". They proposed a Decision Support in Heart Disease Prediction System (DSHDPS) utilizing data mining modeling procedure, in particular, Naïve Bayes. Utilizing medical profiles, for example, age, sex, blood pressure and blood sugar it can anticipate the probability of patients getting a coronary illness.

Sitair-Taut et al. used the weka tool to examine applying Naïve Bayes and J4.8 Decision Trees for the detection of coronary heart disease. Andreeva utilized C4.5 Decision Tree in the diagnosis of heart disease showing exactness of 75.73% (Andreeva 2006).

1.4 EXISTING SYSTEMS

Although there exist several systems such as Intelligent and effective heart attack prediction system using data mining and AINN for prediction. They employed the multilayer perceptron neural network with back propagation as the training algorithm. A graph based way to deal with coronary illness forecast was proposed. However,they lack the genetic parameter to be taken into consideration for prediction and hence are not accurate as ours. They can't provide prediction based on the history of candidate gene.

1.5 PROPOSED SYSTEM

The proposed model takes 19 features into account and produces a more accurate prediction system which will enable an individual to determine whether he/she is at higher risk of having disease or not. Out of these 19 features, 2 features represents the Risk Allele Frequency of the disease causative gene taken from each of his parents and himself. It will help them in taking precautions and avoid the chances of having disease beforehand.

The proposed system uses Bagging methods with Decision Trees as base estimator for prediction. The model is trained on the labeled data and is stored as a pickle object. The pickled object is loaded on the application server.

1.6 STATEMENT OF THE PROBLEM

Clinical decisions are regularly made based on doctor's instinct and experience instead of the knowledge-rich data covered up in the database. This practice leads to undesirable biases, errors and exorbitant therapeutic costs which influences the quality of treatment provided to patients. To reduce therapeutic errors and enhance patient safety machine learning models are integrated with available clinical decision support systems. Quality of clinical decisions can be improved with the help of a knowledge-rich environment generated by machine learning algorithms.

1.7 SUMMARY

This chapter gave a brief introduction on what exactly the proposed system is. It also provides a glimpse of the features of the product to be developed, the need for such documentation, some applications, and the comparison between the existing methods and systems. It even marked the main features of the device which helps in overcoming the drawbacks of the existing system.

Chapter 2

SYSTEM REQUIREMENTS SPECIFICATIONS

2.1 SOFTWARE REQUIREMENTS SPECIFICATIONS

Requirement specification gives a depiction of the planned condition for the development of the system under consideration. It is a complete depiction of the conduct of a system to be developed. A good software requirements specifications characterizes how an application will interface with system components, different projects and human clients in a wide assortment of certifiable circumstances. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from unfavorable events are assessed. A software requirement specification ought to be right, unambiguous, complete, reliable, ranked for importance as well as stability, verifiable, modifiable and traceable. One job of the system specification is to define the full functionality of the system. The sections in this chapter deal with the various kinds of software, hardware and other functional and non functional requirements of the project. A complete description of the various users of the system is also mentioned.

2.1.1 Operating Environment

The environment in which the model operates (client side) may consist of any Operating System with a browser capable of handling HTML5 pages. The server side should be able to handle multiple clients connected to it at the same time and should have sufficient memory.

Hardware Requirements

- **Processor:** 2.2GHz or higher.
- **RAM:** 2GB(64 bit).
- **Storage:** 50GB of available hard disk space.
- Other general hardwares such as a mouse and keyboard for inputs and a monitor for display.

Software Requirements

Should have Python installed with following modules:

- Flask
- Panda
- NumPy
- Scikit-Learn

Other requirements include:

- **Operating system:** Ubuntu 14.04 or above
- **Programming languages:** Python, JavaScript, HTML, CSS

2.1.2 Functional Requirements

Expected services of a project are expressed in the form of functional requirements. Our model should be able to accept incoming connections from clients to establish a 1-1 communication link between the server and the client. Subsequently, users need to provide necessary information to process the result. The system should be able to check for correctness of data entered by user prior to prediction.

2.1.3 Non-Functional Requirements

Non-functional requirements are the various capabilities offered by the system. They force restrictions on the design or implementation such as performance engineering requirements, quality benchmarks, or design constraints. These have nothing to do with the expected results, but focus on how well the outcomes are achieved.

- **Usability**

The system must not be so complex that an individual avoids it. The customer should be okay with the interfaces and should not have troubles while moving to another structure with another

condition. The menus and other components should be named in a way that they give clear understandability of the usefulness of the system.

- **Reliability**

The system should be tried and tested and able to fulfill its functionalities. Once a client makes a few improvements, the progressions should be made faultlessly by the system. The alterations made by the Programmer should be visible both to all the Project Members.

- **Security**

Bug tracking delivers the most extreme security accessible at the most astounding execution rate conceivable, guaranteeing that unapproved clients can't get to crucial issue data without authorization. System must provide vital security and must secure the entire process from crashing.

- **Portability**

This is required when the web server, which is facilitating the framework stalls out because of a few issues, which requires their framework to be taken to another framework.

- **Re-usability**

The system should be modularised such that it could be used as a part of another system requiring minimal or insignificant work.

2.1.4 User Characteristics

User characteristics define the different users who can use the application and benefit from it. User characteristics will help us in understanding our target customers and give us a fair idea as to whom we need to develop our system. As of now our system can be used by any individual who is above thirty years of age and wanted to know his risk of having the disease. Users must enter relevant information to the system.

2.1.5 Applications

Applications lie at Medical help centers. A medical practitioner may use their knowledge to perform a genetic test although they cannot process a classification algorithm. This project eases the process of analyzing their results and providing a highly accurate report to the Health-Check seeker.

The model could be trained on data for different disease and can be used to predict the risk of developing those diseases. This tool can be used to know beforehand the risk of a disease and can

help in save lot of money and man hours and in turn lead to better productivity.

2.2 SUMMARY

This chapter talks about the Software Requirement Specification that introduces the hardware and software requirements and the operating environment needed to run the system. It also describes the different users that interact with the application, along with its application in particular fields. It gives an outline that is needed to design, develop and test the application.

Chapter 3

HIGH LEVEL DESIGN

High level design explains the architecture of the product. Architecture gives an idea about the structure of the system and the subsystems. It mainly focuses on making the end users understand some technical aspects of the project. Design process is not done completely at once, but rather built and developed during the whole project lifetime, adding details and corrections with each iteration. Backtracking processes are used to revise any defects from prior designs.

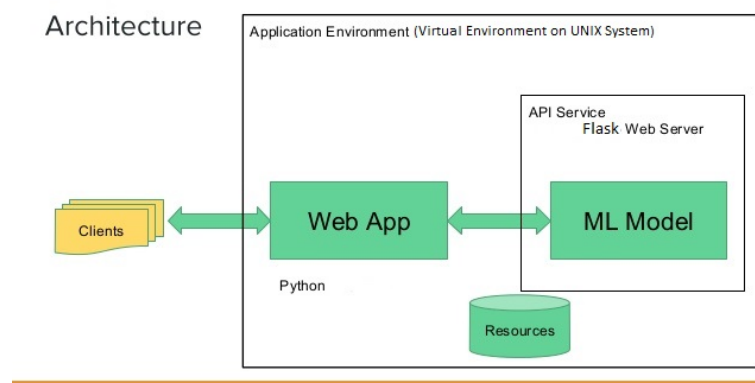


Figure 3.1: High Level Design

3.1 ASSUMPTIONS

Few assumptions were made to analyze and synthesize the data:

- The first assumption made was of the nuclear family. We have assumed that candidates belong to a family having only him as a single child, he has no siblings. This assumption is reasonable for the problem we are trying to solve as it is not expected for a candidate to get affected by his siblings' genes. Each individual carries genes of his parents.
- The second assumption is that a candidate using the system must be over thirty years of age. As Genetic variation occurs only to a certain age and afterwards no significant changes are observed.

3.2 DESIGN APPROACH

Here are two non-specific methodology for software designing:

- Top-down Design: takes the entire software system as one entity and then decomposes it to accomplish in excess of one subsystem or component based on some characteristics.
- Bottom-up Design: model starts with most specific and basic components. It continues with integrating higher level of components by using essential or lower level components.

As mentioned above the project requires two main modules to be implemented. Each module has its own components to be developed. We use bottom-up design strategy in this product design phase as we start designing the basic components in each module and finally we interlink both the modules to get the final product.

3.3 SYSTEM ARCHITECTURE

The system architecture gives an expansive review of how the system should function. It gives a point by point perspective of the task of the system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. It can include system parts, the remotely noticeable properties of those parts, the connections between them.

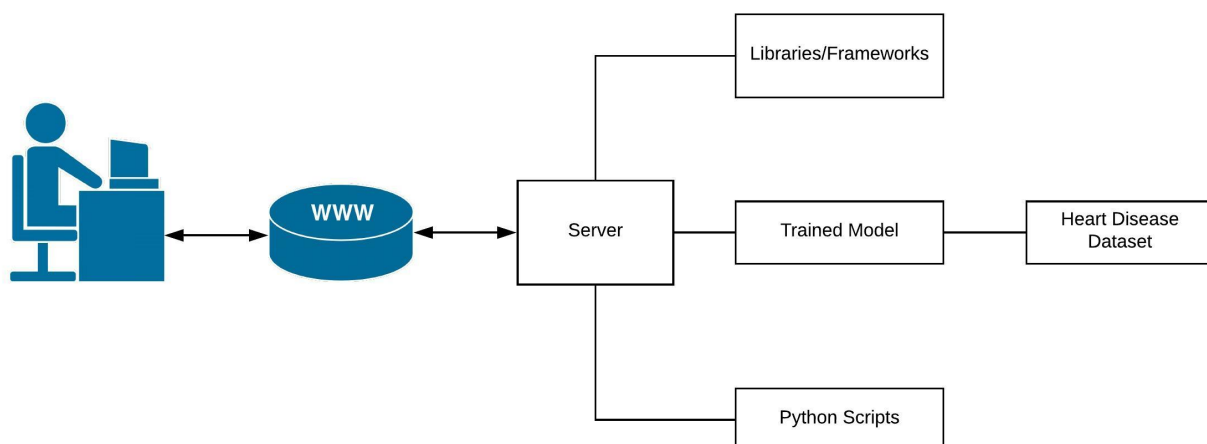


Figure 3.2: System Architecture

3.4 DATA FLOW DIAGRAM

It is a basic graphical formalism that can be utilized to represent a system in terms of the information to the system, different processing carried out on these input data and the output data is generated by the system.

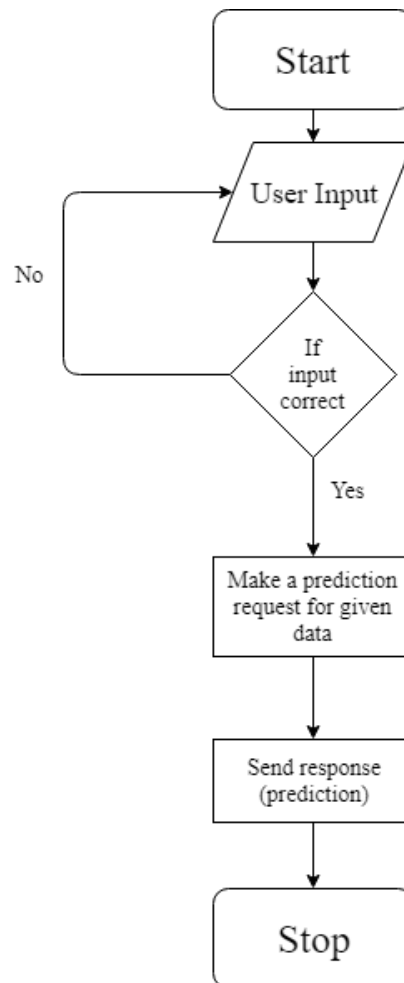


Figure 3.3: Data Flow Diagram

3.5 SEQUENCE DIAGRAM

A Sequence diagram is a graphical representation that shows how system work with co-operation and in what order. It is a develop of a Message Sequence Chart. A sequence diagram demonstrates object interaction in time sequence. It portrays the items and classes associated with the situation and the succession of messages traded between the objects expected to complete the functionality of the situation.

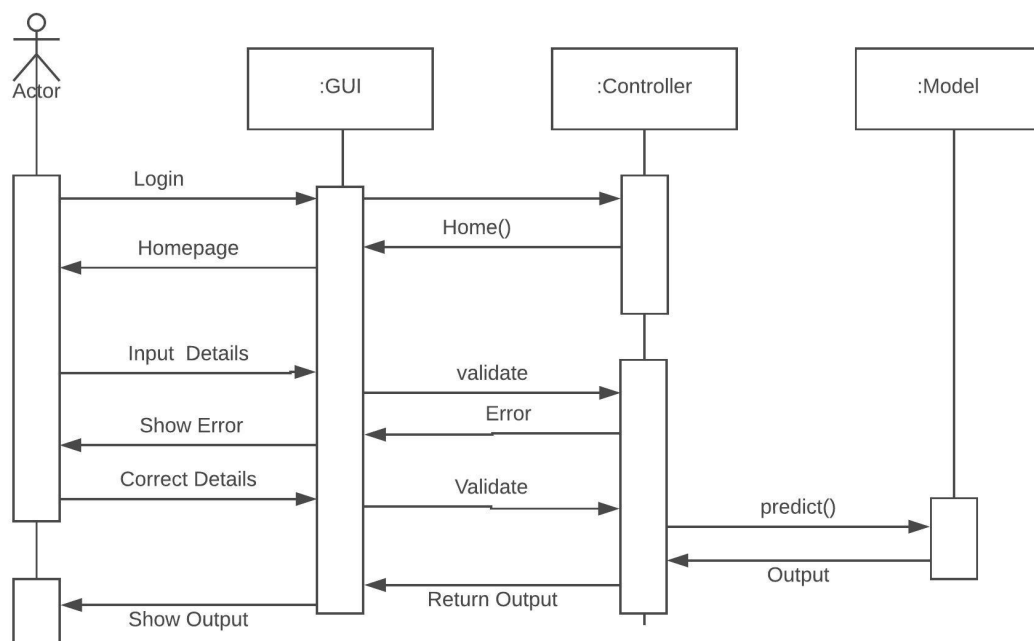


Figure 3.4: Sequence Diagram

3.6 SUMMARY

In this chapter we portray the high level design of the classification and prediction system. System architecture gives an overview of the whole system. The data flow diagram show how the data is processed in the system. The sequence diagram depicts the time based interaction between the different objects involved in the project.

Chapter 4

DETAILED DESIGN

4.1 PURPOSE

The Detailed design is a deliberation when contrasted with source code, however ought to be sufficiently detailed to guarantee that interpretation to source is an exact mapping rather than a rough understanding. The purpose of detailed design is to portray every details about the project and steps in making of the project.

4.2 PREDICTION

The dataset we used is obtained from UCI Machine Learning repository. Obtained dataset is Pre-processed to remove missing values and eliminate non-relevant features. However, we realized upon completion of processing that the data collected was inadequate, and we modeled the data based on distributions. Processed dataset is further split into training and test data. Training dataset is used for training the model and is tested against the Test dataset. The accuracy achieved by doing so is of about 84% . The model classifies input values into class labels. Class labels represents the presence or absence of disease. This ends the prediction phase.

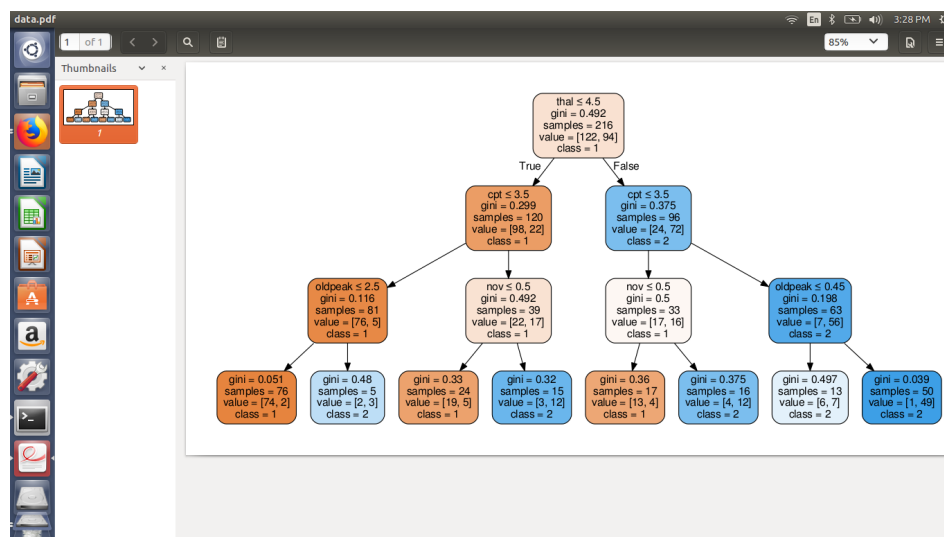


Figure 4.1: Tree form during training process

4.3 CLASSIFICATION

Classification is the process of grouping of entities with the similar attribute under one class label. This is necessary because it validates our hypothesis on the efficiency of our model. Classification in our case is a binary classification showing the risk factor of disease by predicting which class the candidate belongs to, that is whether he is at the risk of having the disease or not. Decision trees classifiers are used for the classification. Type 1 represents the samples in which the

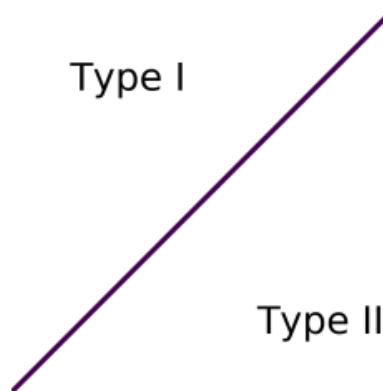


Figure 4.2: Binary Classification

candidate is at the risk of having the heart disease and Type 2 represents the remaining samples in which candidate will not be at the risk of having the disease.

4.4 DATASET

Cleveland Clinic Foundation Heart disease data set is used in this project. Dataset consist of both continuous and discrete values. 76 raw attributes are presented in the data. However, Only 13 of these attributes are included in all of the published experiments. Subsequently, to permit correlation with the writing, we limited testing to these same attributes. Additional genetic features are modeled using underlying distribution present in the data. The data set contains 303 rows of which 297 are complete. Six rows are incomplete and hence are expelled.

Table 4.1: Dataset Attributes

Name	Type	Description
Age	Continuous	Age in years
Sex	Discrete	1 = male 0 = female
Cp	Discrete	Chest pain type 1 = typical angina 2 = atypical angina 3 = non-anginal pain 4 = asymptomatic
Trestbps	Continuous	Resting blood pressure (in mm Hg)
Chol	Continuous	Serum cholesterol in mg/dl
Fbs	Discrete	Fasting blood sugar >120 mg/dl 1 = true 0 = false
Restecg	Discrete	Resting electrocardiographic results: 0 = normal 1 = having ST-T wave abnormality 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria
Thalach	Continuous	Maximum heart rate achieved
EIA	Discrete	Exercise induced angina 1 = yes 0 = no
Old peak ST	Continuous	Depression induced by exercise relative to rest The slope of the peak exercise segment
Slope	Discrete	1 = up sloping 2 = flat 3 = down sloping

Table 4.1: Dataset Attributes

Name	Type	Description
Ca	Discrete	Number of major vessels colored by fluoroscopy that ranged between 0 and 3. 3 = normal
Thal	Discrete	6= fixed defect 7= reversible defect
raf (rs10757278)	Discrete	Risk Allele frequency of rs10757278 in child mother and father lies between 0 and 1
raf (rs2383207)	Discrete	Risk Allele frequency of rs2383207 in child mother and father lies between 0 and 1
Diagnosis	Discrete	Diagnosis classes 0 = healthy 1= patient who is subject to possible heart disease

This dataset can be used as a training grounds to best type of fit for all the further incoming data of similar nature.

4.5 DECISION TREE CLASSIFIER

Once the Data Assessment is done the selection of appropriate classifiers is the major task pending. There are multiple ways we can approach this problem such as SVM, Naive Bayes, Random Forest etcetera. But as we tried and tested we got the best scores for the decision tree classifier(CART) implemented with bagging .

Decision tree is one of the most used supervised learning algorithm because of it's simplicity and the fact that it can be used for for both classification and regression tasks. Decision tree constructs classification or regression models as a tree structure. It separates a dataset into smaller subsets while in the meantime a related choice tree is incrementally created. The final outcome is a tree with decision nodes and leaf nodes.

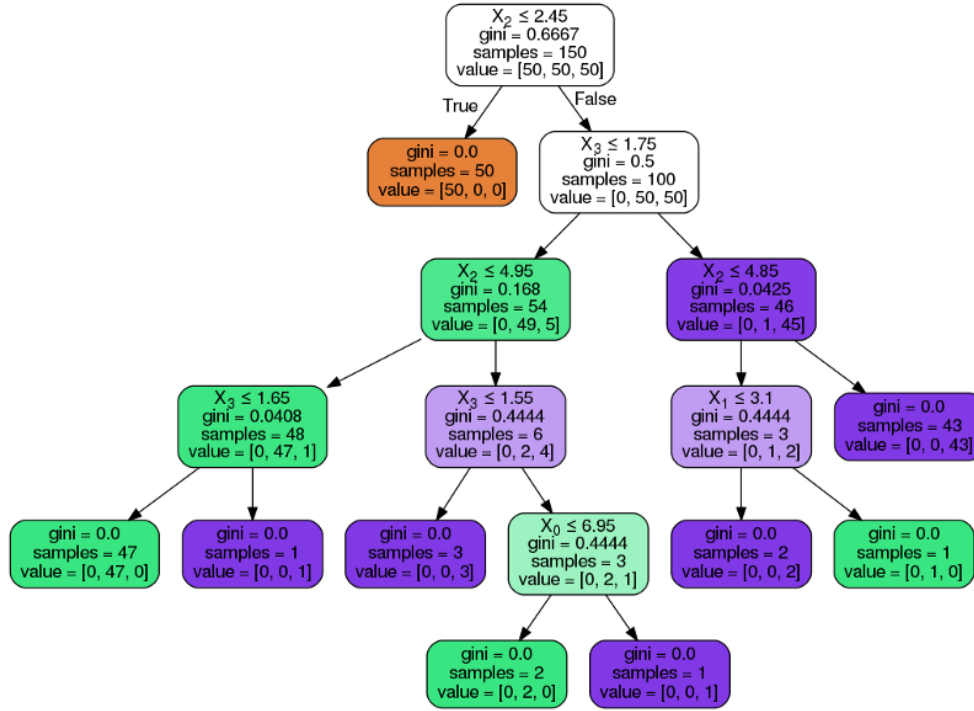


Figure 4.3: Visualization using Decision Trees

4.5.1 Gini Index

Gini Index measures the impurity of data. It is used to determine the root node in a decision tree. The Gini Index is calculated for each attribute in the data set. If there are c classes of the target attribute, with the probability of the i th class being P_i , the Gini Index is defined as shown below.

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

Figure 4.4: Gini Index

The training algorithm for decision trees with bagging applies the general technique of bootstrap aggregating to tree learners. Given a training set $X = x_1, \dots, x_n$ with responses $Y = y_1, \dots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, \dots, B$:

- Sample, with replacement, n training examples from X, Y ; call these X_b, Y_b .
- Train a classification or regression tree f_b on X_b, Y_b .

So to say in simple words, Bagging creates multiple decision trees and merges them together to get a more accurate and stable prediction. This method ensures that the results are usually better or equal to decision trees for most cases.

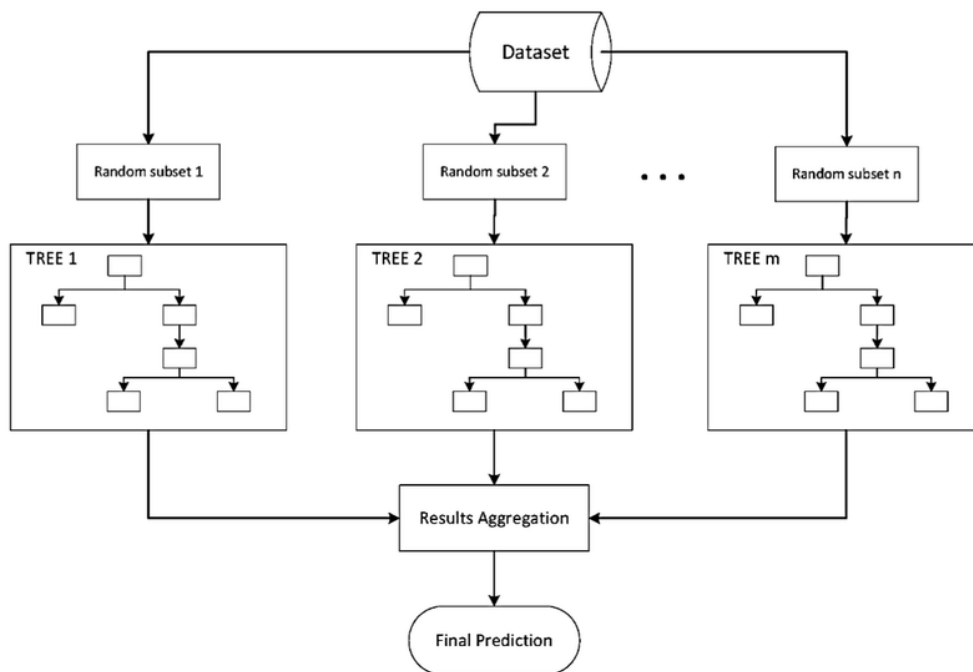


Figure 4.5: Visualization of bagging algorithm

Chapter 5

IMPLEMENTATION

Implementation is the phase of the undertaking when the hypothetical plan is turned out into a working system. This stage is the interpretation of the system necessities and specifications into a working model to satisfy the real time services. Key functionalities identified from the design stage are converted into functions that are executable using appropriate programming languages.

Implementation of any software is constantly gone before by imperative choices with respect to selection of platform, the language utilized, etc. These choices are often influenced by a couple of components, for instance, genuine condition in which the system works, the speed that is required, the security concerns, and other implementation particular points of interest.

5.1 PROGRAMMING LANGUAGE SELECTION

This project is implemented in Python, JavaScript and Flask as they are:

- Simple: The languages was designed to be easy for the professional programmer to learn and use effectively. They inherits the C/C++ style and many object oriented features of C++.
- Object-oriented: The object model in Flask is simple and easy to extend, while primitive types are kept as non-objects for performance reasons.
- Robust: Python frees you from having to worry about some of the common programming errors. It is a strictly typed language and checks the code both at compile time and run time.
- Multi threaded: JavaScript was designed to meet the real world requirement of creating interactive network programs. To accomplish this, it supports server-side programming on the client side.
- Interpreted and High-performance: Web development enables the creation of cross-platform programs by running on the browser.

5.2 PLATFORM SELECTION

5.2.1 Linux(Ubuntu 16.10)

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Use of Virtual Environment

Virtual environments have the advantage that they never install the required dependencies system wide so we have a better control over the environment in which our application is running. We can choose only to install the required libraries and packages and keep the environment clean.

Virtual environment is also important as we may have multiple applications on one system with conflicting requirements.

5.2.2 Flask

Flask is a Python framework, based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD license. Some of the important features of Flask are:

- built-in development server and fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Jinja2 template
- support for secure cookies (client side sessions)
- WSGI 1.0 compliant
- Unicode based

5.2.3 Python

A prominent component of Python is its indenting of source proclamations to make the code less demanding to peruse. Python offers dynamic data write, instant class, and interfaces to numerous framework calls and libraries. It can be broadened, utilizing the C or C++ language.

5.3 Libraries Required

- **scikit-learn**: scikit learn is a wide python library which practices machine learning, cross validation of data, and preprocessing of data. It is built on NumPy and SciPy.
- **NumPy**: NumPy's main object is an multidimensional array, it holds an array data structure. NumPy is a core python library which contains a collection of tools and techniques. One of these tools is multi dimensional object.
- **Matplotlib**: It is a library extensively used for visualizing the data.
- **SciPy**: It is a library which contains functions to perform more complex calculations, particularly scientific computations.
- **Pandas**: Pandas is an open source, Python library. It provides high-performance, easily useable data structures and data analysis tools for Python programming language.

5.4 IMPLEMENTATION STEPS

i. Front-end implementations

- Create a simple web page with a text input field, a button to transmit the text to a server and a popup box which can show the user the output processed by the system after prediction.
- Establish a web-socket communication between the web page and the server.
- Validation mechanism is embedded in web page itself for checking the correctness of value entered by user.
- Once data is validated, it is passed to server and model is invoked.

ii. Deployment

- collect data as in step 1.
- Using the synthetic data, train a decision tree classifier to classify between labels.
- Using using the input data as test data, classify into one of the two classes.
- Provide the prediction outcome to the user.

5.5 Summary

This chapter dealt with the various techniques used in the development of the project, starting with the language and platform selection to finally explain the entire process of implementation steps.

Chapter 6

TESTING

Testing is the method of evaluate and analyse a system or its component(s) with the conjecture to discover whether it fulfills the predecided requirements or not. It is done in order to identify any missing requirements, errors, or gaps in contrary to the actual requirements. Finally the entire is tested to make sure that faults in previous faces is eradicated and the project works as specified.

6.1 SOFTWARE TESTING

It is essential to declare the nature of the product before it is put into real use. Programming testing is a procedure of assessing the framework and additionally its individual components with the point of finding any bugs or mistakes. The real reason for programming testing is to discover out whether the item fulfills its prerequisites and particulars. It can be utilized to discover any gaps or missing necessities in contrast with the normal prerequisites.

A good way to reduce the cost incurred from fixing the bugs is to start the testing process in parallel to the implementation. In the software test cycle testing can be started once the specifications are clearly lied out and go on till the deployment of the product. Testing can be done in each phase of the life cycle with the intent of improving the phase. Besides debugging, testing also involves verification and validation of the product to ensure both functional and non functional requirements are met.

6.1.1 White Box Testing

White box testing is a method of software testing that tests the working of the program or an application rather than functionality. It Check whether all independent paths within a module have been exercised at least once. It exercise all logical decisions on their false sides. Execution of all loops and their boundaries and within their boundaries occurs. Path testing is an obvious example.

It ensure whether all possible validity checks and validity lookups have been provided to validate data entry. To eliminate redundancy that occurs in regression of our project, white box testing is used. Since errors can occur anywhere in classification code ,redundancy tests are done.

WHITE BOX TESTING APPROACH

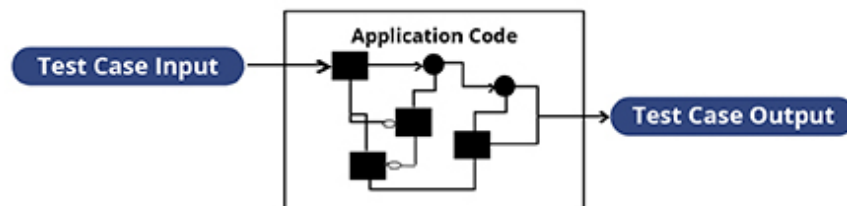


Figure 6.1: White box testing

6.1.2 Black Box Testing

In black box testing the function of the black box is understood completely in terms of inputs and outputs in spite of not knowing the content or implementation of black box. Specification testing is also called as functional testing because the program is considered as a function that maps values from its input domain to values in its output range.

Test cases of Specification based testing as 2 specific advantages:

- They are not dependent on how the software is implemented. In case if the implementation changes, the test cases are still useful.
- Reduction in the overall project development interval because both the test case development and implementation can occur in parallel.

6.2 LEVELS OF TESTING

There are normally 3 levels of testing: unit testing, integration testing and system testing.

From the point of view of customers there are 2 different levels: low-level testing, high-level testing.

- low level testing is a set of tests for different level components of software application.

- High level testing is a set of tests for the application as a whole.

6.2.1 Unit Testing

In the method individual units of source code, program module associated control data, other procedures are tested to find their fit for use. Specific functionality of the developed subunits are tested. The more appropriate testing at unit testing is structural testing. This testing is followed by integration testing. Integration testing basically clubs all the modules tested for unit testing and perform the tests defined in integration test plan to them. This is then followed by system testing.

6.2.2 Integration Testing

Integration testing is the second level of testing. As mentioned before this combines all the modules of unit testing and tests them. Its main goal is to test whether there are any problems in the integration of different modules.

6.2.3 System Testing

System testing basically evaluates whether a system meets the requirements specified before. For example, a system testing may include inputting values in the input fields, printing results, format of the results etc.

It also tests the behaviour of system as specified by the customer. It not only tests the requirements in software/hardware specification but also tests beyond that. This is then followed by acceptance testing.

6.2.4 Acceptance Testing

As the name itself suggests acceptance testing is used to test the system compliance for the requirements. This is the final stage of testing process before the system is accepted for operational use. The system is tested within the data supplied from the system procurer rather than simulated data.

6.3 UNIT TESTING OF MODULE

Execution of the prediction system is tested for various conditions and the test cases are tabulated as follows:

Unit Testing of validation of data

Test Case ID	Unit Test Case P1
Description	Check Whether input data is correct
Input	Relevant Data
Expected Output	All Data entered should must be correct
Actual Output	correct input type
Remarks	Pass

Unit Testing of server working

Test Case ID	Unite Test Case P 2
Description	Check Whether input data is passed on to server
Input	Data entered by user
Expected Output	Model should be invoked
Actual Output	Model is invoked
Remarks	pass

Integration Testing

Test Case ID	Unite Test Case P 3
Description	To test the predictor with input values
input	Related Information
Expected Output	Outcome based on prediction
Actual Output	Result shown to user based on his input values
Remarks	Pass

6.4 RESULTS

A classifier accuracy is given in terms of sensitivity, specificity and Recall. Sensitivity tells about how good a test is at detecting the positives. Specificity express how good a test is at avoiding false alarms. while precision tells about how many positively classified were relevant.

When using single decision tree classifier for making predictions the accuracy achieved was of about 79%.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 6.2: Formulae

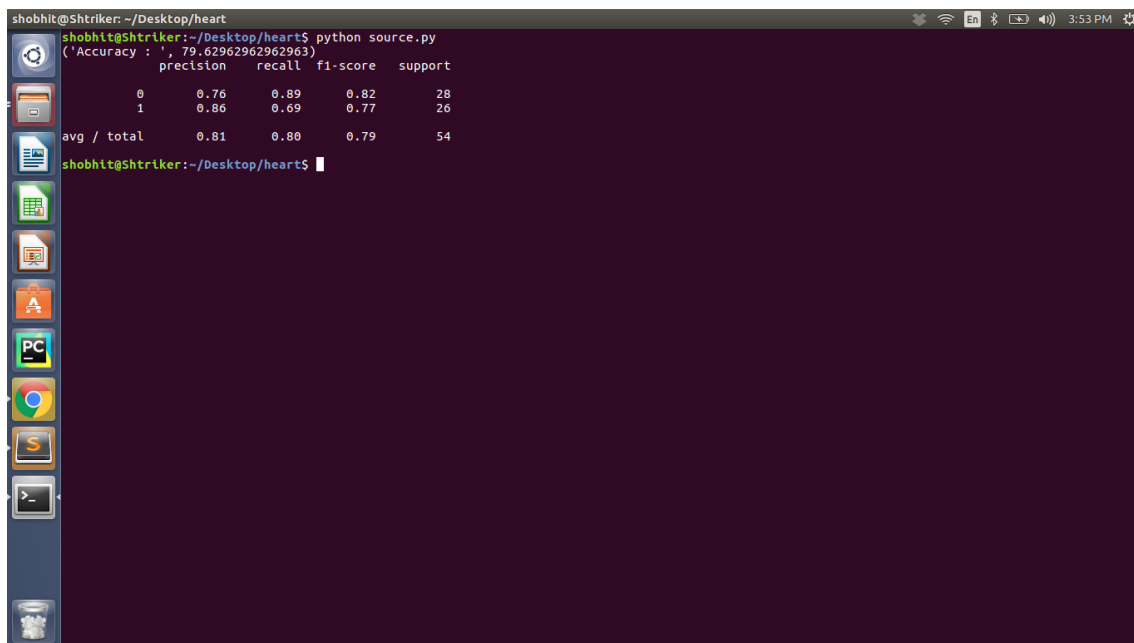
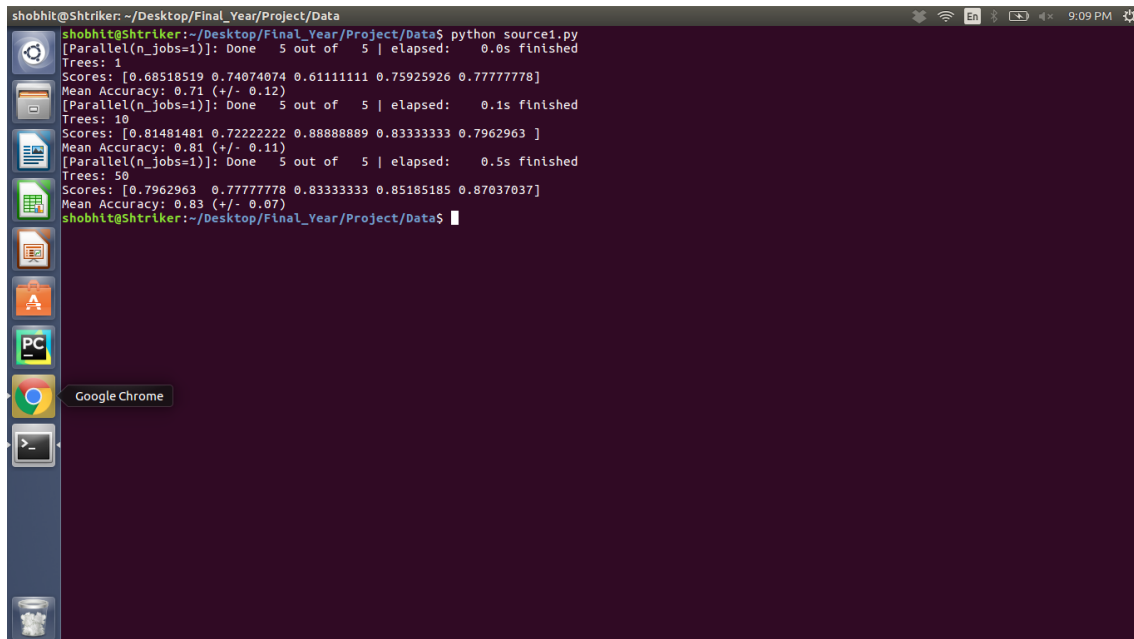


Figure 6.3: Result

while when bootstrap aggregation with replacement or bagging is implemented by taking decision trees as base estimator the accuracy becomes more stable. The accuracy achieved after implementing the bagging algorithm is of about 84%.



```
shobhit@Shtriker: ~/Desktop/Final_Year/Project/Data
shobhit@Shtriker:~/Desktop/Final_Year/Project/Data$ python source1.py
[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.0s finished
Trees: 1
Scores: [0.68518519 0.74074074 0.61111111 0.75925926 0.77777778]
Mean Accuracy: 0.71 (+/- 0.12)
[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.1s finished
Trees: 10
Scores: [0.81481481 0.72222222 0.88888889 0.83333333 0.7962963 ]
Mean Accuracy: 0.81 (+/- 0.11)
[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.5s finished
Trees: 50
Scores: [0.7962963 0.77777778 0.83333333 0.85185185 0.87037037]
Mean Accuracy: 0.83 (+/- 0.07)
shobhit@Shtriker:~/Desktop/Final_Year/Project/Data$
```

Figure 6.4: Final Result

6.5 SUMMARY

Software testing is a process to show the customers the quality of the product. Unit testing is a method used to test individual modules. Integration testing aggregates all modules that are unit tested and tests them using Integration testing methods. System testing basically tests the system to check whether the system meets all the specified requirements.

Chapter 7

USER INTERFACE

7.1 GRAPHICAL USER INTERFACE

A web based user interface was implemented for this project. The Web UI is an HTML-based application used to configure and manage the server appliance from a remote client. The Website provides a clear description of all the work done in the project. Users will feed the data to the trained model and the results of the prediction are presented on the webpage.

Following are the technologies used for implementing the UI:

- Bootstrap
- HTML & CSS
- Javascript
- Flask

Bootstrap is used for front-end development, it contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as Javascript extensions.

Following are the advantages of using Bootstrap:

- It already has predefined design templates and classes, which saves a lot of time.
- All bootstrap components share a consistent design throughout.
- It is easy to use and compatible with all browsers.

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and jinja2 template engine. Flask is considered and used for the following advantages:

- It's easy to set up
- It's well documented
- It's very simple and minimalistic, and doesn't include anything you won't use
- It's flexible enough that you add extensions, if you need more functionality

Jinja is a template engine for the python programming language and is licensed under a BSD License. It provides Python like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any mark-up as well as source Code.

As the below figure 7.1 would be the home page of the project and on clicking predict button on the home page, a form is shown to the user. User input the related information through this form. Figure 7.2 is showing the valid forms values that are entered.

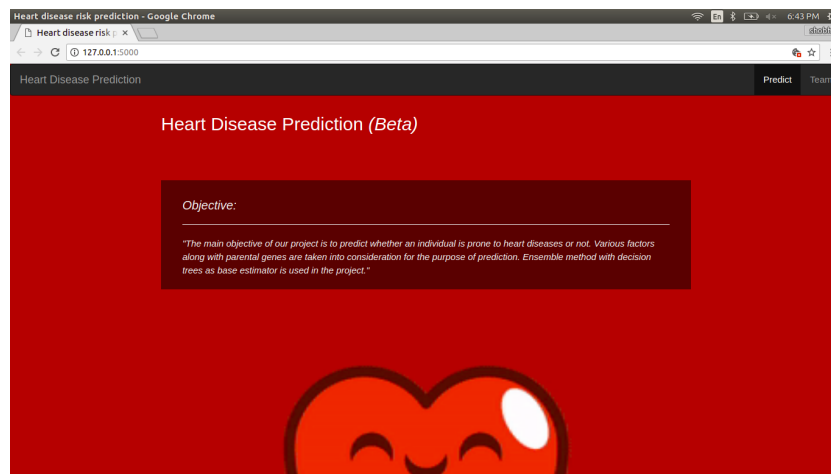


Figure 7.1: Home Page

On inputting the appropriate and acceptable form values would redirect the page to the below figure 7.3

A screenshot of the input form for the 'Heart Disease Prediction' application. The form is titled 'Please Enter the following Details:' and is set against a red background. It contains several input fields and dropdown menus for user data entry. The fields are: Age (with a hint 'between 30 to 80 years'), Sex (a dropdown menu), Chest Pain Type (a dropdown menu), Blood Pressure (with a hint 'Resting Blood Pressure (50-300 only)'), Cholesterol level (with a hint 'Serum Cholesterol in mg/dl(<200 healthy)'), Fasting Blood Pressure (a dropdown menu), ECG Value (a dropdown menu), Heart Rate (with a hint 'Maximum heart rate achieved'), and Exercise Induced Angina (a dropdown menu). The 'Predict' button is visible in the top right corner of the form area.

Figure 7.2: Input Form

Figure 7.3 as shown below include the prediction button. After entering values and clicking on prediction button will redirect to the server. The model is invoked and the data is hence passed to it.

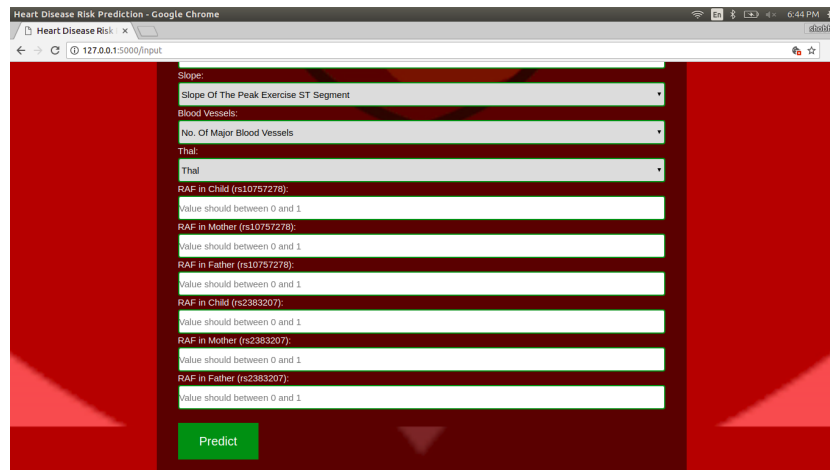
The screenshot shows a web browser window titled "Heart Disease Risk Prediction - Google Chrome". The address bar shows "127.0.0.1:5000/input". The page has a red background. It contains several input fields: "Slope:" with a dropdown menu showing "Slope Of The Peak Exercise ST Segment"; "Blood Vessels:" with a dropdown menu showing "No. Of Major Blood Vessels"; "Thal:" with a dropdown menu showing "Thal"; and three pairs of input fields for "RAF in Child (rs10757278)", "RAF in Mother (rs10757278)", and "RAF in Father (rs10757278)", each with a note "Value should be between 0 and 1". Below these fields is a green "Predict" button.

Figure 7.3: Predict Button

Figure 7.4 shows the outcome of prediction made by model based on the data input by user. This outcome tends to change according to the input data.

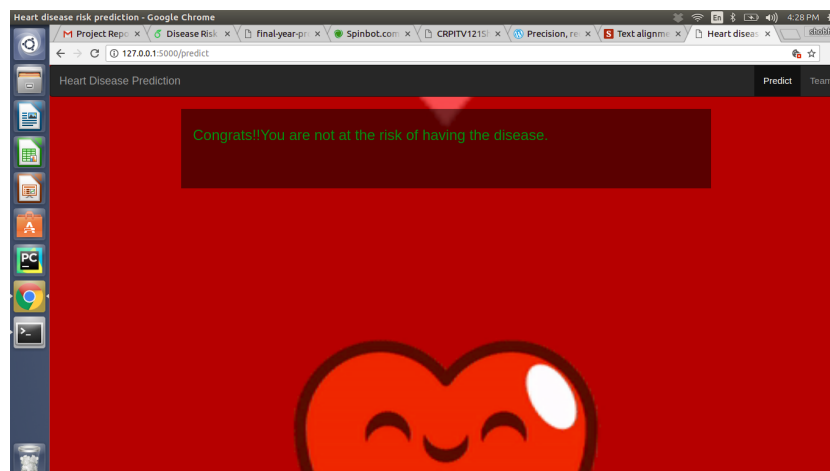


Figure 7.4: Prediction

Chapter 8

ANALYSIS & CONCLUSION

8.1 ANALYSIS

As a part of the prediction stage in our system, we initially split the dataset obtained from UCI Machine Learning repository to training and test data following the 0.7:0.3 ratio i.e., 70% of the data was used as training data and the latter 30% was used as test data. In the ensemble classifier we fed the data into the Decision tree algorithm and modeled the classifier. To this modeled classifier we fed in the test data and the class labels for the test data are predicted. We obtained an accuracy of 84%. The Summary of our analysis as shown below:

- An individual has high risk of having the disease when Risk Allele Frequency of rs10757278 and rs2383207 exceeds threshold value of 0.60.
- Risk of having disease is also increased when resting heart rate increases.
- Individual with smoking habits are more prone to develop the disease.
- Individual with angina chest pain type is at the verge of developing disease.

8.2 CONCLUSION

This project outlines the different analysis done on the heart disease dataset and the usage of bagging method with decision tree classifiers as model to make predictive insights on the probability of having the heart disease. This model can be applied throughout the various disease and can be used as an aid to help make better decisions by knowing the chances of having the disease. The model should be updated periodically and include more additional features for it to make more accurate predictions.

8.3 LIMITATIONS

- absence of existing data.
- The results of the study were limited to the area in which research data was collected.
- The model is valid only for the individuals in age group of 30-80 years.
- Model is also applicable only on family having single child.

8.4 FUTURE WORK

- Future researchers can examine the same correlations by carrying out longitudinal research study i.e, doing research on data which is gathered over a long span of time.
- Future work can also consider including more variables from the database that could have more impact on determining cause of disease .

8.5 SUMMARY

In this project we see the analysis of the project which we created. The analysis is done by taking in the heart disease data set and splitting it into 70:30 ratio as training and test data respectively. On this data bagging method with decision tree classifiers is applied. The thus modeled classifier gives an accuracy of 84%. In the next section precision, recall, F-score, accuracy are shown.

The limitations of the project are mentioned in the above section and future scope shows what development can be made to get better results.