KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT FINAL REPORT ON

WECARE-YOUR PERSONAL HEALTH ASSISTANCE

Submitted by:

Abhishek Gurung [91/BCT/072]

Manjil Nepali [105/BCT/072]

Sagar Maharjan [118/BCT/072]

Sovin Shrestha [126/BCT/072]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

August, 2019

WECARE-YOUR PERSONAL HEALTH ASSISTANCE

Submitted by:

Abhishek Gurung [91/BCT/072]

Manjil Nepali [105/BCT/072]

Sagar Maharjan [118/BCT/072]

Sovin Shrestha [126/BCT/072]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

August, 2019

ABSTRACT

According to the World Health Organization, health is “a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity.” Better health is central to human happiness and well-being. Our project ”WeCare” is a prediction system that helps the user to get the information about the disease according to the symptoms they feed to the system. So that it is also called as Health Assistance application because it assists the user about the possible disease but does not give the exact information.

Traditionally, physicians or doctors use a risk calculator to assess the possibility of disease development. These calculators use fundamental information such as age, medical conditions, and more to calculate the probability of developing a certain disease. Main aim of our project is to develop the prediction system which predict the disease with help of data mining algorithm i.e. Feed Forward Back Propagation Neural Network

Keywords– Health Assistance,Data Mining, Back Propagation Neural Network

ACKNOWLEDGMENT

This project is the outcome of inspiration and moral support of many people and for this we are grateful to them all. It's hardly possible to list the names of all but we sincerely acknowledge their incredible contribution during the preparation. We take this opportunity to express our sincere gratitude to all those who have directly or indirectly inspired us for the completion of this project.

We express our gratitude towards the Computer and Electronic Department of Kantipur Engineering College for providing us the learning and working environment which was helpful to work for our team. We are very grateful to our project supervisor Shagar Upadhyay for his support in this project. Special thanks goes to Er. Bishal Thapa and Er. Shiva Gautam for providing us theoretical background, practical advice and valuable suggestions for the development of this design. We would also like to thank the library staff for providing us with the study materials needed for research purposes.

Abhishek Gurung	[91/BCT/072]
Manjil Nepali	[105/BCT/072]
Sagar Maharjan	[118/BCT/072]
Sovin Shrestha	[126/BCT/072]

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Human health is a relative state in which one is able to function well i.e physically, mentally and live socially well-being within the environment in which one is living. The human body is an incredible machine i.e has the ability to adapt, repair itself and manage challenges throughout life. With the on-going development of world over years, human learns the symptoms of affected human and provide remedy for hazardeous diseases.

The health sector is of critical importance for human development, improving living standards in rural areas and for mainstreaming marginalized groups and communities. Despite significant progress in recent years, service delivery in the health sector remains weak. Although an extensive network of primary health care centres has been constructed nationwide, it has not been functioning well in many rural areas due to lack of trained staff, drugs and medicines, etc. The sector's overall performance has suffered due to inadequate funding for essential recurrent expenditures, misallocation of resources and limited capacity for supervision and for co-ordination of the activities of other agencies providing health care services.

This android application entitled "WeCare-Your Personal Health Assistance" aims to be helpful application by predicting the possible diseases as Tele-medicine center. Our proposed system takes symptoms from the user as input and data mining technique is used to do provisional diagnosis as practiced in telemedicine centers. Similar to that in telemedicine center, output generated doesn't provide fully diagnosed results. The user has to consult doctor for further treatment.

In order to reduce the risk of disease, prediction should be done. Discovering of disease is usually based on symptoms, physical examinations and signs of patient body. Normally, doctors are predicting disease by knowledge and experience. Discovering and predicting diseases is a difficult task in medical environment. Discovering disease from

several factors is a multi-layered problem which may lead to negative presumptions and unpredictable effects. As a result, Health-care industry today creates large amounts of complex data about patients, hospitals resources, disease diagnosis, electronic patient records, medical devices etc. The huge amount of data is a key resource to be processed and analyzed for knowledge extraction that enables support for cost-savings and decision making.

1.2 Problem Statement

Taking appointment with the doctors is very difficult in both rural an urban areas of Nepal. Like most developing nations, doctors are geographically maldistributed in Nepal. The Kathmandu valley has one doctor for 850 people but in rural areas the number is one doctor for every 150000 people. The doctor-population density in Kathmandu is estimated to be about 40 times that in rural Nepal.

People often feel reluctant to go to hospital or physician on minor symptoms. However, in many cases, these minor symptoms may trigger major health hazards. If they get some information about the disease they might have been suffering from, it will be helpful when they discuss their problems with the doctor later.

People will not know the possible reasons for the disease and may be mistaken with other disease. It will be helpful to take the preventive measures if people have some idea about the disease they are suffering from. Now a day's people are busy on their daily life works so they forget to take their medicine on time, thus it could be helpful if someone notify them on time to take their medicine. So, Pill Reminder lets you create tasks with a deadline. The reminder simply reminds you when it's time to do it.

1.3 Objectives

The major objectives are listed:

- To give environment to the users to diagnose their symptoms.
- To list out provisional diagnosed diseases.
- To help users take an active role in meaning their health by providing objective healthcare information.

1.4 Applications

This developed application enables users to make sense of symptoms and recognition of disease by decision algorithm. This project has wide use in various fields.

As in rural areas, it might have happened so many times that people may need doctors help immediately, but they are not available due to some reason. This project allows user to get instant guidance on their health related problems. This app is a tool for user to check their symptoms, find trusted information and probable diseases with questionnaire based on the symptoms. This project is not limited to diseases prediction only, it suggests basic preventive measures and overview of the disease. Users can take immediate action. This project has wide area of application use in telemedicine centers. It can be easily access by nursing home staffs, sometimes parents and patient themselves. Users can get quick instant guidance of disease with preventive measures.

This project also provides the functionality of pill reminder, where users can get instant notifications of time and medicine to be taken according to their prescriptive noted schedule. As taking of medicine pill in time, during of sickness has great affect in health, pill reminder can be very useful to users.

1.5 Features

The features are listed:

- Provides suggestions and informs about the possible occurence of disease based on the symptoms ans users information.
- User friendly interface.
- Suggest basic preventive measures and overview of disease.
- Works as pill reminder.

1.6 Feasibility Analysis

The analyses of how feasible the project is in the terms of its technology, its operation, its cost etc in the current context are done in this section. Feasible, in the sense, can be considered as how well it can be continued taking account of the above scenario. That is whether the technology available is appropriate of the software or not, whether the software can be developed within the given cost and time or not and at last whether the software can be developed within the given cost and time or not and at last whether the facility provided by the software is fit for the given operation or not. So the feasibility analysis can be done in terms of the following four topics. These are:

- Economic feasibility
- Technical feasibility
- Operational feasibility
- Schedule feasibility

1.6.1 Economic Feasibility

Economic Feasibility is mainly concerned with the benefits after the implementation of the system. Economic analysis is done by comparing the total cost incurred during the development of the product and integration with the system with the return given by the use of the system. So, here the cost benefit analysis is done. The return should always be greater than the cost incurred for the project to be economically feasible.

1.6.2 Technical Feasibility

Technical Feasibility is concerned mainly with the technology available for the development of the system. Whether the technology needed is available or not. It is mainly concerned with the following issues:

- Is the technology needed for the project practical?
- Do we possess the necessary technology?
- Do we possess the necessary technical expertise?

The technologies needed for the project mainly consist of the hardware devices like a laptop computer, android device for testing which is easily available in the market with the cost which is not so high. The operating system needed is Microsoft windows 2007 or above versions, which is also readily available. The work done in this sector is not for the first time, there has been many work done before. There are many health experts and scientists that have been involved in the work of disease predicting and neural network which are the main aspects of the project. So, there are many findings and algorithm that has already been developed which may easily available in the books or in the internet. The platform required for the development of the project which is the Python. So, the technology needed for the development of the project is practical.

1.6.3 Operational Feasibility

As we know operation means the service provided by the software to the users. The software should be capable of providing all the necessary requirements of the users. So, here, the measure of how well the software can solve the given problem of where it is applied is stated. It also gives the measure of how well the users can use the software to solve their problem and wether the software is acceptable by the users or not.

The improvement of the operations of any domain and the measure of how well the software can solve the problem of a given domain can be analyzed according to PIECES framework which stands for:

P=Performance

I=Information

E=Economy

C=Control

E=Efficiency

S=Services

1.6.4 Schedule feasibility

The time frame of the project is analyzed in this section. The analysis of whether the project will be completed or not within the given time is done in the schedule feasibility. The total time for the completion of the project in about 6 months starting from falgun to shrawan and it is possible to complete the project within this time provided that the required hardware and the software are readily available.

1.7 System Requirements

1.7.1 Software Requirements

- Android Studio
- Java
- Python
- Flask

1.7.2 Hardware Requirements

- Android Mobile

CHAPTER 2
LITERATURE REVIEW

There are different areas in medicine where an expert system has been designed and implemented. Numerous works has been done related to disease diagnosis using different data mining techniques.

2.1 Previous Work On Medical Expert System

2.1.1 MYCIN

MYCIN is the name of a decision support system developed by Stanford University in the early-to mid-seventies, built to assist physicians in the diagnosis of infectious diseases. The system (also known as an ”expert system”) would ask a series of questions designed to emulate the thinking of an expert in the field of infectious disease (hence the ”expert-”), and from the responses to these questions give a list of possible diagnoses, with probability, as well as recommend treatment (hence the ”decision support-”).

MYCIN is an AI program designed:

- to provide expert level solutions to complex problems
- to be understandable
- to be flexible enough to accommodate new knowledge easily

MYCIN was designed to provide advice through a consultative dialogue, which sometimes refer to it as a consultation system.

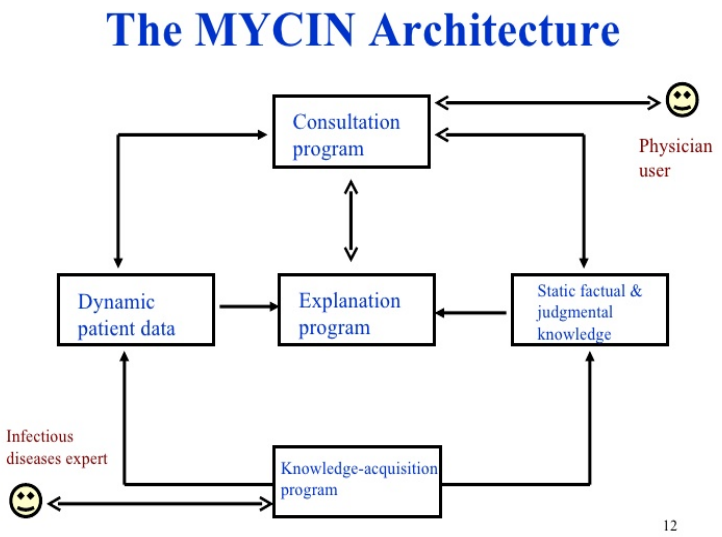

Figure 2.1: MYCIN Architecture(Source: slideshare.net)

2.1.2 Chronic Kidney Disease Prediction Using Back Propagation Neural Network Algorithm

This system of neural network accepts disease-symptoms as input and it is trained according to various training algorithms. Levenberg, Bayesian regularization, Scaled Conjugate and Resilient back propagation algorithm are discussed here. After neural network is trained using back propagation algorithms, this trained neural network system is used for detection of kidney disease in the human body. The back propagation algorithms presented here have capacity for distinguishing amongst infected patients or non-infected person.

2.2 Automated Disease Prediction System (ADPS): A User Input-based Reliable Architecture for Disease Prediction

Automated Disease Prediction System (ADPS) that relies on guided (to be described later) user input. The system takes input from the user and provides a list (topmost diseases have greater likelihood of occurrence) of probable diseases. The accuracy of ADPS has been evaluated extensively. It ensured an average of 14.35% higher accuracy

in comparison with the existing solution.

In this paper, the contribution includes proposing a new disease prediction framework (ADPS) that takes into account symptom names as well as other vital parameters to improve disease prediction accuracy and proposing techniques to allow greater linguistic diversity so that users do not feel uncomfortable while giving input.

It is presumed that the user will give text input in one sentence describing a single symptom at a time (guideline for user input). Subsequent symptoms can be added in new lines. After getting user input, the system will scan through each line and tag each word according to their relevant parameter. Then after performing certain computations (to be described later) the system will return a list of possible diseases ordered according to the likelihood of their occurrences.

2.3 Neural Network Based Intelligent System for Predicting Heart Disease

This paper proposed an intelligent automated system incorporating the techniques of data mining with machine learning in order to make decisions. Medical practitioners are being assisted by the automated systems for providing effective treatment [18]. Data mining techniques involves a combination of statistical methods with machine learning algorithms .Data mining techniques help the system in analyzing the symptoms and machine learning methodshelp in predicting the disease based on the analysis performed [13,20]. The advantage of this automated system is that it predicts the disease in a less amount of time as well in less cost. Therefore, more research is carried out in the field of machine intelligence to improvise the system for an effective prediction. This paper proposed an intelligent system developed using the concept of Multilayer Perceptron Neural Network with Back Propagation Algorithm, as a practitioner needs to make a decision from multiple inputs such as current and previous medical history of a patient.

Neural networks are proved to be effective in making decisions by predicting the data. As the inputs used in predicting the disease are more in number and diagnosis has to be performed at different stages, Multilayer Perceptron based neural networks are used in

this proposed system. Neural Network extends its predictive capability at different hierarchical levels in a multi-layered structure of networks. This multi-layered structure helps in selecting features from the dataset at different scales in order to refine them into more specific features. To facilitate this, the concept of Multi-layer Perceptron Neural Network has been introduced through the implementation of Back-propagation algorithm for efficient diagnosis of heart disease. In this paper, 14 attributes are used as inputs for training the system of neural networks for diagnosing heart disease risk level using multi-layered network.Traditional diagnosing approaches have no proper automated tools use for the purpose of heart disease diagnostic system. The commonly used data mining algorithms for predicting diseases are: Genetic algorithm, K-means algorithm, MAFIA algorithm. Several methods proposed the implementation of classification algorithms in diagnosis of heart disease and resulted with an accuracy of 88.33

CHAPTER 3
METHODOLOGY

3.1 System Process Model

3.1.1 Incremental Software Development Model

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.

The system is put into production when the first increment is delivered. The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the client, there is plan development for the next increment.

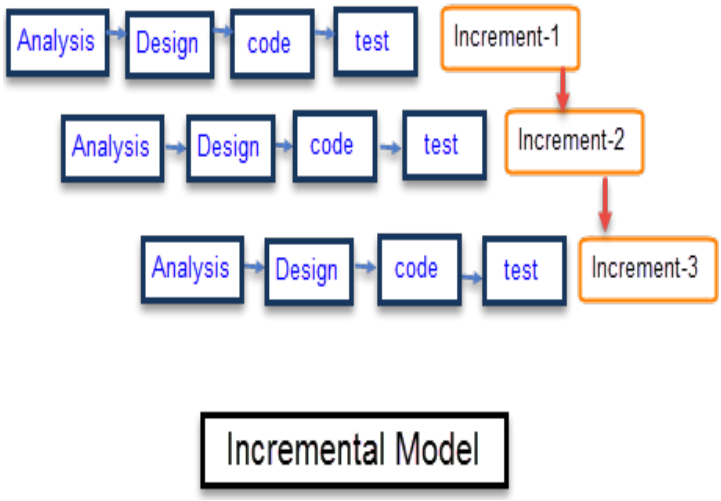

Figure 3.1: Incremental Development Model(Source: researchgate.net)

3.2 Reason of using Python ?

We found out that python was great choice machine learning implementation in our project because of two simple reasons

- Simplicity: When it comes to big data or just very complex decision-making path, the simplicity is the key. Python is well-known for its readability, developer-friendly syntax, and semantics. The less the developer has to worry about the code itself, the more focus and emphasis can be put on finding solutions. Moreover, the Python community is huge and constantly growing, which is also a plus point for a language.
- Libraries and frameworks : What makes Python so suitable is the amount of ready, open source libraries and frameworks. In our project for many data related tasks like importing dataset, formatting dataset, data visualization, statistics about dataset we are using numpy, pandas library. Using these library saves a lot of time and energy.

3.3 Why this project?

As Nepal is a developing country. Health Cares, doctors, hospitals have not reached at every corner of the country specially on the rural areas. So, we came with the idea of doing this project that uses AI to provide health care facilities through our mobile application.

Artificial intelligence in healthcare is the use of complex algorithms and softwares to estimate human cognition in the analysis of complicated medical data. The primary aim of health-related AI applications is to analyze relationships between prevention or treatment techniques and patient outcomes.

AI technology from traditional technologies in health care is the ability to gain information, process it and give a well-defined output to the end-user. AI does this through machine learning algorithms. These algorithms can recognize patterns in behavior and create its own logic. In order to reduce the margin of error, AI algorithms need to be

tested repeatedly. AI algorithms behave differently from humans in two ways:

- Algorithms are literal: If you set a goal, the algorithm can't adjust itself and only understand what is has been told explicitly
- Algorithms are black boxes: Algorithms can predict extremely precise, but not the cause or the why.

The primary aim of health-related AI applications is to analyze relationships between prevention or treatment techniques and patient outcomes.[2] AI programs have been developed and applied to practices such as diagnosis processes, treatment protocol development, drug development, personalized medicine, and patient monitoring and care.

Hence, AI is reshaping the health care industry by helping to predict and scan diseases, detect injuries, and help people maintain good health even on a day-to-day basis with easy-to-use mobile applications.

WeCare-Your Personal Heath Assistance is a AI based project focused on making sure that patients take the right medications at the right time. For that purpose, we use technologies like user friendly interface, preventive measures and overview of the disease, pill reminder which can help people to predict the disease with the ongoing symptoms that they are suffering and help them to maintain good health.

3.4 Procedure for Medical Diagnosis

3.4.1 Data Collection

This is most essential step in machine learning project. Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models. To train our model, we need dataset that contain information about diseases and its respective symptoms.
Here is some information about the dataset which we obtained from UCI machine learning repository.

- Total Number of unique Diseases: 41
- Total Number of symptoms: 132
- Total Number of entities in dataset: 4920

	disease_name	symptoms
0	Fungal infection	itching,skin_rash,nodal_skin_eruptions,dischromic _patches,
1	Allergy	continuous_sneezing,shivering,chills,watering_from_eyes,
2	GERD	stomach_pain,acidity,ulcers_on_tongue,vomiting,cough,chest_pain,
3	Chronic cholestasis	itching,vomiting,yellowish_skin,nausea,loss_of_appetite,abdominal_pain,yellowing_of_eyes,
4	Drug Reaction	itching,skin_rash,stomach_pain,burning_micturition,spotting_ urination,

Figure 3.2: Sample of dataset

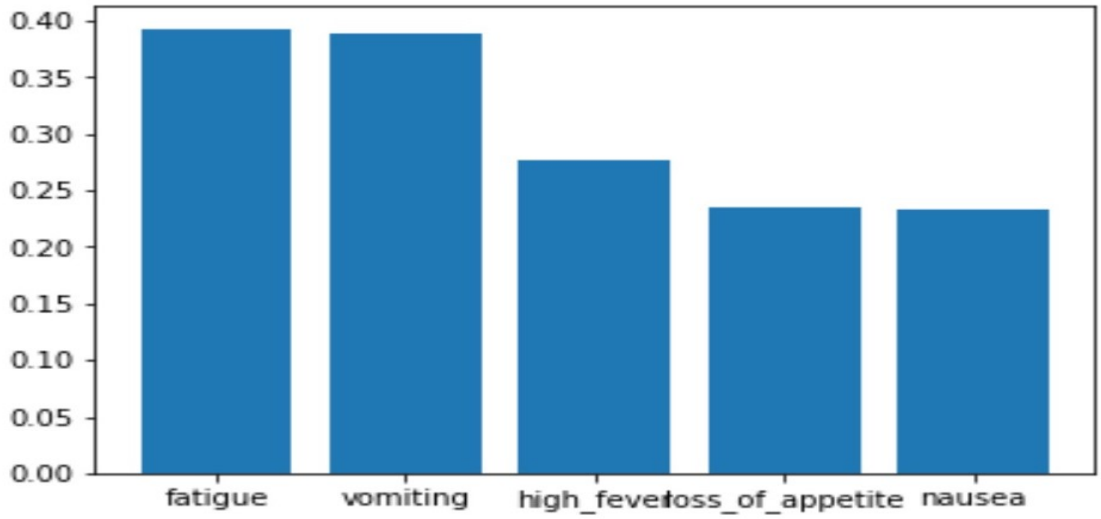

Figure 3.3: Top five most common symptoms

3.4.2 Data Preparation

Most machine learning algorithms require data to be formatted in a very specific way, so datasets generally require some amount of preparation before they can yield useful insights. Some datasets have values that are missing, invalid, or otherwise difficult for an algorithm to process. If data is missing, the algorithm can't use it. If data is invalid, it causes the algorithm to produce less accurate or even misleading outcomes. Good data preparation produces clean and well-curated data that leads to more practical, accurate model outcomes.

Also to train the model we need to split the dataset in train dataset test dataset, the test dataset will be used for model evaluation.

3.4.3 Choosing a Machine Learning Algorithms

When we look at machine learning algorithms, there is no one solution or one approach that fits for all king of problems. There are several factors such as

- Type of problem
- Size of training set
- Accuracy
- Number of features

That can affect your decision to choose a machine learning algorithm. Since our project associated with the supervised learning problem, there are different supervised learning algorithms like KNN, Random Forest, Decision Tree, ANN etc.

3.5 Introduction to Biological neural network

A neural network consists of numbers of non-linear computational elements called neurons or node. These neurons are connected to each other. The concept of neural network developed from way a human behaves. The way neurons work in the human brain is tried to be used in neural network. A neuron's dendrite tree is connected to thousand neighbour neurons. When one neurons fire a positive or negative charge it is received by the dendrites. The strength of all the charges fired is added together through the process of spatial and temporal summation. Spatial summation occurs when several weak signals are converted into a single large one, while the temporal summation converts a rapid series of weak pulses from one source into a large signal.
The dendrites serve as receptors for signals from other neurons, whereas the purpose of an axon is transmission of the generated neural activity to other neurons. The aggregate inputs are then passed to the soma(cell body). If the aggregate is greater then the axon hillock's threshold value, then the neuron fires and an output signal is transmitted down to axon. The strength of the output is constant regardless of whether the input was just above the threshold, or hundred times greater. The output strength is unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it

had at the axon hillock.

Each terminal button is connected to other neurons across a small gap called synapse. The physical and neuro-chemical characteristics of each synapse determine the strength and the polarity of the new input signal. This process repeats itself across all neurons.

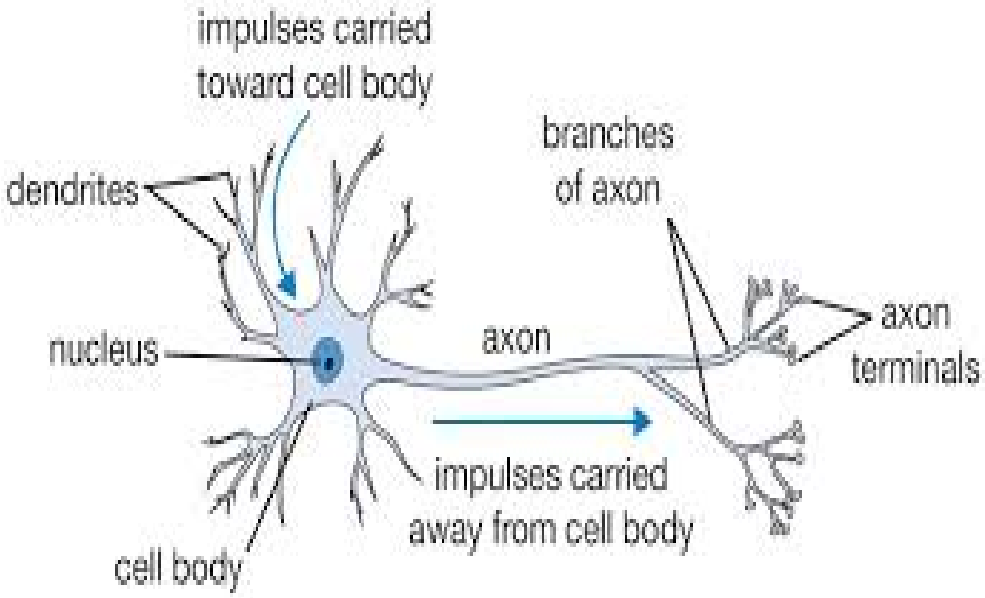

Figure 3.4: Biological Neural Network

Characteristics of Biological Neural Network

Some of the characteristics of the biological neural network that make it superior to the most sophisticated AI computer system for the pattern tasks are the following:

- Robustness and fault tolerance: The decay of the nerve cell does not seem to affect the performance significantly.
- Flexibility: The network automatically adjusts to a new environment without any preprogrammed instruction.
- Ability to deal with various types of data situation: The network can deal with situations that are fuzzy, probablistic, noisy and inconsistence.
- Collective computation: The network performes routinely many operations in parallel and also a given task in a distributed manner.

3.6 Artificial Neural Network

ANN is a mathematical structure composed of highly interconnected elements that are capable of modeling complex nonlinear relationship. A feed-forward multilayer perceptron network which usually adapts error back propagation in training state is the most popular ANN paradigm that is performs well in nonlinear problems. There are two types of ANN namely, supervised ANN and unsupervised ANN. Unsupervised ANN doesn't require output for guiding the weight adjustment whereas for supervised ANN the model is trained based on comparison of the output and target until the model output matches the target.

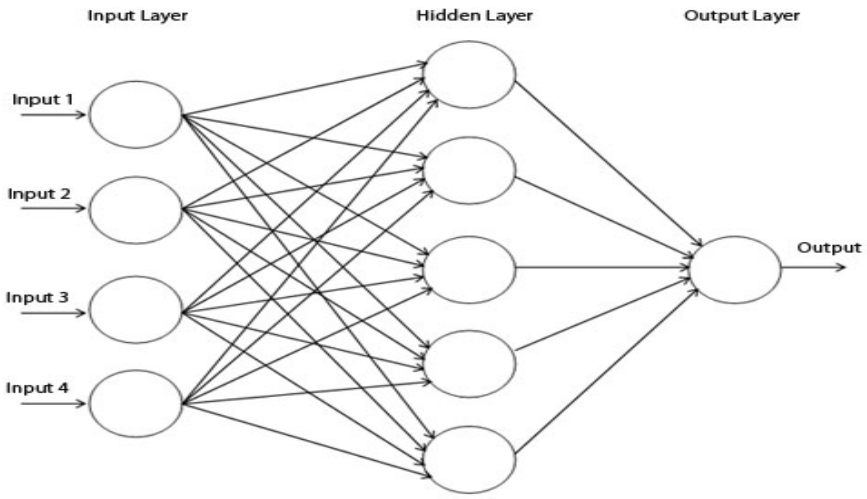

Figure 3.5: Neural Network (Source: researchgate.net)

3.6.1 Elements of a Neural Network

- **Input Layer**: This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.
- **Hidden Layer**: Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by any neural network. Hidden layer performs all sort of computation on the features entered through the input layer and transfer the result to the output layer.
- **Output Layer**: This layer bring up the information learned by the network to the outer world.

- **Neuron**: The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output.
- **Bias**: Bias is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron. Therefore Bias is a constant which helps the model in a way that it can fit best for the given data.
- **Activation Function**: Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.
  Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions you may encounter in practice like sigmoid, reLU, tanh.
- **Learning rate**: learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

3.7 Biological vs Artificial Neural Network

An artificial neural network is basically a mathematical model built from simple functions with changing parameters. Just like a biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons, the artificial neuron has a number of input channels, a processing stage, and one output that can fan out to multiple other artificial neurons. Although artificial neurons and perceptrons were inspired by the biological processes scientists were able to observe in the brain back in the 50s, they are vastly different in terms of both their structure and

workings.

Size: Our brain contains about 86 billion neurons and more than a 100 trillion synapses. The number of "neurons" in artificial networks is much less than that but comparing their numbers this way is misleading. Perceptrons just take inputs on their "dendrites" and generate output on their "axon branches". A single layer perceptron network consists of several perceptrons that are not interconnected: they all just perform this very same task at once. Deep Neural Networks usually consist of input neurons, output neurons and neurons in the hidden layers, in-between. All the layers are usually fully connected to the next layer. Manual feature extraction requires human brain power which is also not taken into account when summing up the number of "neurons" required for Deep Learning tasks. The limitation in size isn't just computational: simply increasing the number of layers and artificial neurons does not always yield better results in machine learning tasks.

Topology: All artificial layers compute one by one, instead of being part of a network that has nodes computing asynchronously. Feedforward networks compute the state of one layer of artificial neurons and their weights, then use the results to compute the following layer the same way. During backpropagation, the algorithm computes some change in the weights the opposing way, to reduce the difference of the feedforward computational results in the output layer from the expected values of the output layer. In biological networks, neurons can fire asynchronously in parallel, have small-world nature with a small portion of highly connected neurons and a large amount of lesser connected ones. Since artificial neuron layers are usually fully connected, this small-world nature of biological neurons can only be simulated by introducing weights that are 0 to mimic the lack of connections between two neurons.

Speed: Certain biological neurons can fire around 200 times a second on average. Signals travel at different speeds depending on the type of the nerve impulse. Action potential frequency carries information for biological neuron networks: information is carried by the firing frequency or the firing mode (tonic or burst-firing) of the output neuron and by the amplitude of the incoming signal in the input neuron in biological systems. Information in artificial neurons is instead carried over by the continuous, floating point

number values of synaptic weights. How quickly feedforward or backpropagation algorithms are calculated carries no information, other than making the execution and training of the model faster. They are functions that can be calculated as many times and as fast as the computer architecture would allow.

Fault-tolerance: Biological neuron networks due to their topology are also fault-tolerant. Information is stored redundantly so minor failures will not result in memory loss. They don't have one "central" part. The brain can also recover and heal to an extent. Artificial neural networks are not modeled for fault tolerance or self regeneration,though recovery is possible by saving the current state (weight values) of the model and continuing the training from that save state. Dropouts can turn on and off random neurons in a layer during training, mimicking unavailable paths for signals and forcing some redundancy. Trained models can be exported and used on different devices that support the framework, meaning that the same artificial neural network model will yield the same outputs for the same input data on every device it runs on. Training artificial neural networks for longer periods of time will not affect the efficiency of the artificial neurons. Another difference is, that all processes (states and values) can be closely monitored inside an artificial neural network.

Power consumption: The brain consumes about 20% of all the human body's energy-despite it's large cut, an adult brain operates on about 20 watts (barely enough to dimly light a bulb) being extremely efficient. For benchmark: a single Nvidia GeForce Titan X GPU runs on 250 watts alone, and requires a power supply instead of beef tallow. Our machines are way less efficient than biological systems. Computers also generate a lot of heat when used, with consumer GPUs operating safely between 50–80 degrees Celsius instead of 36.5–37.5 C. Biological neural networks use very little power compared to artificial networks.

Signal transport and processing: An action potential is either triggered or not — biological synapses either carry a signal or they don't. Perceptrons work somewhat similarly, by accepting binary inputs, applying weights to them and generating binary outputs depending on whether the sum of these weighted inputs have reached a certain threshold. Artificial neurons accept continuous values as inputs and apply a simple non-

linear, easily differentiable function (an activation function) on the sum of its weighted inputs to restrict the outputs' range of values. The human brain works asynchronously, ANNs work synchronously.

Learning: We still do not understand how brains learn, or how redundant connections store and recall information. By learning, we are building on information that is already stored in the brain. Artificial neural networks in the other hand, have a predefined model, where no further neurons or connections can be added or removed. Only the weights of the connections can change during training. The networks start with random weight values and will slowly try to reach a point where further changes in the weights would no longer improve performance. Learning can be understood as the process of finding optimal weights to minimize the differences between the network's expected and generated output: changing weights one way would increase this error, changing them the other way would decrees it. The rate of how artificial neural networks learn can change over time.

Training (backpropagation using an optimization method like stochastic gradient descent, over many layers and examples) is extremely expensive, but using a trained network (simply doing feedforward calculation) is ridiculously cheap. Unlike the brain, artificial neural networks don't learn by recalling information — they only learn during training, but will always "recall" the same, learned answers afterwards, without making a mistake. These are all considerable differences between biological and artificial neurons.

3.8 Back Propagation Neural Network Algorithm

Back Propagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks. From a desired output, the network leans from many inputs, similar to the way a child learns to identify a dog from examples of dogs. Back propagation is a neural network learning algorithm. The feedback forward network structure is most important ANN structure. Design of a feed forward net for any specific application involves many issues, most of which require problem dependent solutions. The overall computational approach comprising of two parts: feed forward

implementation of learned mapping and training of 3 layer network.

Back Propagation Neural is a multilayer neural network consisting of the input layer, at least one hidden layer and output layer. As its name suggests, back propagating will take place in this network. The error which is calculated at the output layer, by comparing the target output and the actual output, will be propagated back towards the input layer.

3.8.1 Architecture of Back propagation

As shown in the diagram, the architecture of Back Propagation Network has three interconnected layers having weights on them. The hidden layer as well as the output layer also has bias, whose weight is always 1, on them. As is clear from the diagram, the working of Back Propagation Network is in two phases. One phase sends the signal from the input layer to the output layer, and the other phase back propagates the error from the output layer to the input layer.

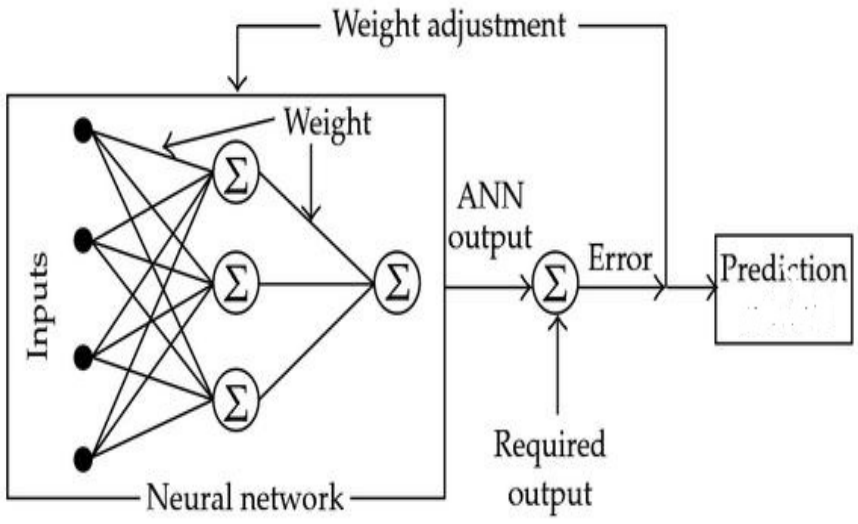

Figure 3.6: ANN architecture with back-propagation algorithm.(Source: researchgate.com)

3.8.2 A Pseudo-Code Algorithm

- Randomly choose the initial weights
- While error is too large

  -For each training pattern (presented in random order)
  - Apply the inputs to the network
  - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
  - Calculate the error at the outputs
  - Use the output error to compute error signals for pre-output layers
  - Use the error signals to compute weight adjustments
  - Apply the weight adjustments

  -Periodically evaluate the network performance

3.8.3 Training/learning Model

The training process of the MLP occurs by continuous adjustment of the weights of the connections after each processing. This adjustment is based on the error in output(which is the different between the expected result and the output). This continuous adjustment of the weights is a supervised learning process called 'backpropagation'.
The backpropagation algorithm consists of two parts:
1.forward pass
2.backward pass
In the forward pass, the expect output corresponding to the given inputs are evaluated. In the backward pass, partial derivatives of the cost function with respects to the different parameters are propagated back through the network.
The process continues until the error is at the lowest value.

3.8.4 Basic Neuron Model In A Feedforward Network

- Inputs x_i arrive through pre-synaptic connections
- Synaptic efficacy is modeled using real weights w_i
- The response of the neuron is a nonlinear function f of its weighted inputs

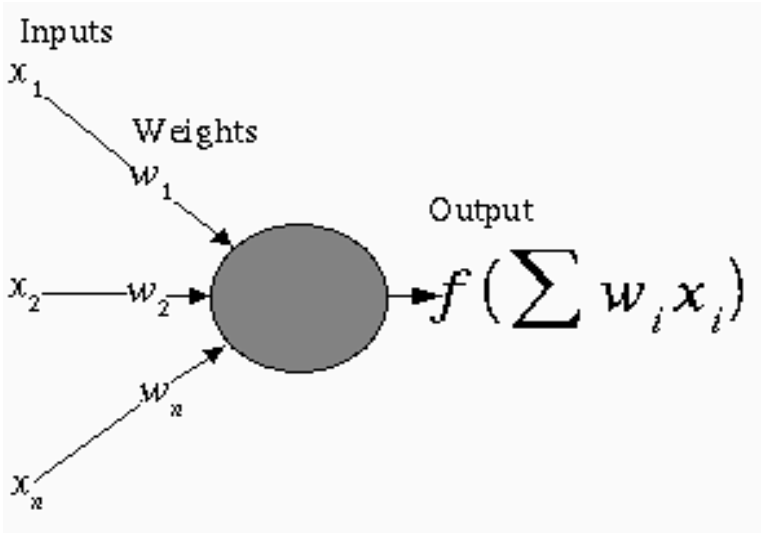

Figure 3.7: Neuron Model Feedforward Network(Source: edureka.co)

3.8.5 Inputs To Neurons

- Arise from other neurons or from outside the network
- Nodes whose inputs arise outside the network are called input nodes and simply copy values
- An input may excite or inhibit the response of the neuron to which it is applied, depending upon the weight of the connection

3.8.6 Weights

- Represent synaptic efficacy and may be excitatory or inhibitory
- Normally, positive weights are considered as excitatory while negative weights are thought of as inhibitory
- Learning is the process of modifying the weights in order to produce a network that performs some function

3.8.7 Outputs

- The response function is normally nonlinear
- Samples include

$$f(x) = 1/1 + exp^{-\sigma x}$$

3.8.8 Network Error

- Total-Sum-Squared-Error (TSSE)

$$TSSE = \sum_{patterns} \sum_{outputs} (desire - actual)^2$$

- Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{\frac{2 * TSSE}{patterns * outputs}}$$

3.8.9 Calculate Outputs For Each Neuron Based On The Pattern

- The output from neuron j for pattern p is O_{pj} where

$$O_{\text{pj}}(net_{\text{j}}) = 1/1 + exp^{-\sigma netj}$$

and

$$(net_{\text{j}}) = bias * W_{\text{bias}} + \sum_{k} O_{\text{pk}} W_{\text{kj}}$$

where k ranges over the input indices and W_{jk} is the weight on the connection from input k to neuron j

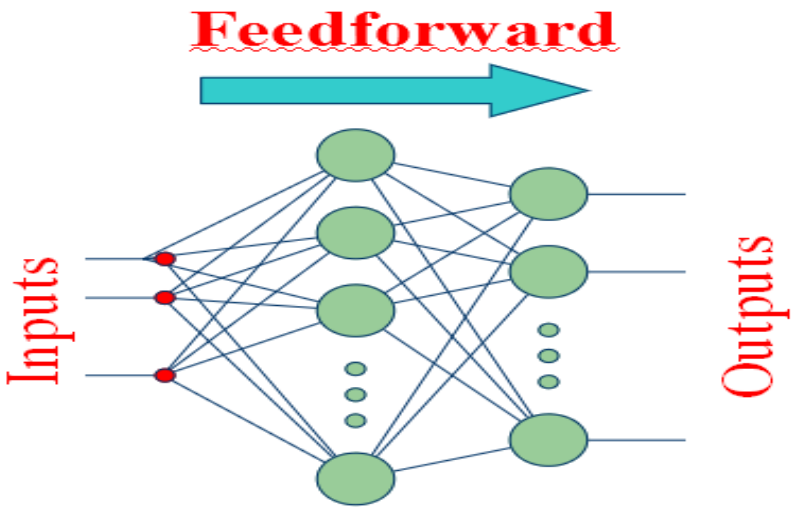

Figure 3.8: Feedforward Network Model(Source: edureka.co)

3.8.10 Calculate The Error Signal For Each Output Neuron

- The output neuron error signal Δ pj is given by

$$\Delta pj = (T_{\text{pj}} - O_{\text{pj}})O_{\text{pj}}(1 - O_{\text{pj}})$$

- T_{pj} is the target value of output neuron j for pattern p
- O_{pj} is the actual output value of output neuron j for pattern p

3.8.11 Calculate The Error Signal For Each Hidden Neuron

- The hidden neuron error signal ∂ pj is given by

$$\partial pj = O_{\text{pj}}(1 - O_{\text{pj}}) \sum_{k} \partial_{\text{pk}} W_{\text{jk}}$$

where ∂ pk is the error signal of a post-synaptic neuron k and W_{jk} is the weight of the connection from hidden neuron j to the post-synaptic neuron k

3.8.12 Calculate And Apply Weight Adjustments

- weight adjustments δ W_{ji} at time t by
  Δ $W_{ji}(t) = O_{pi}$
- Apply weight adjustments according to
  $W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}(t)$
- Some add a momentum term
  α * ΔWji(t-1)

3.8.13 Summarization of Steps

- Calculate the error – How far is your model output from the actual output.
- Minimum Error – Check whether the error is minimized or not.
- Update the parameters – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
- Model is ready to make a prediction – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

3.9 System Analysis

3.9.1 Flowchart Diagram

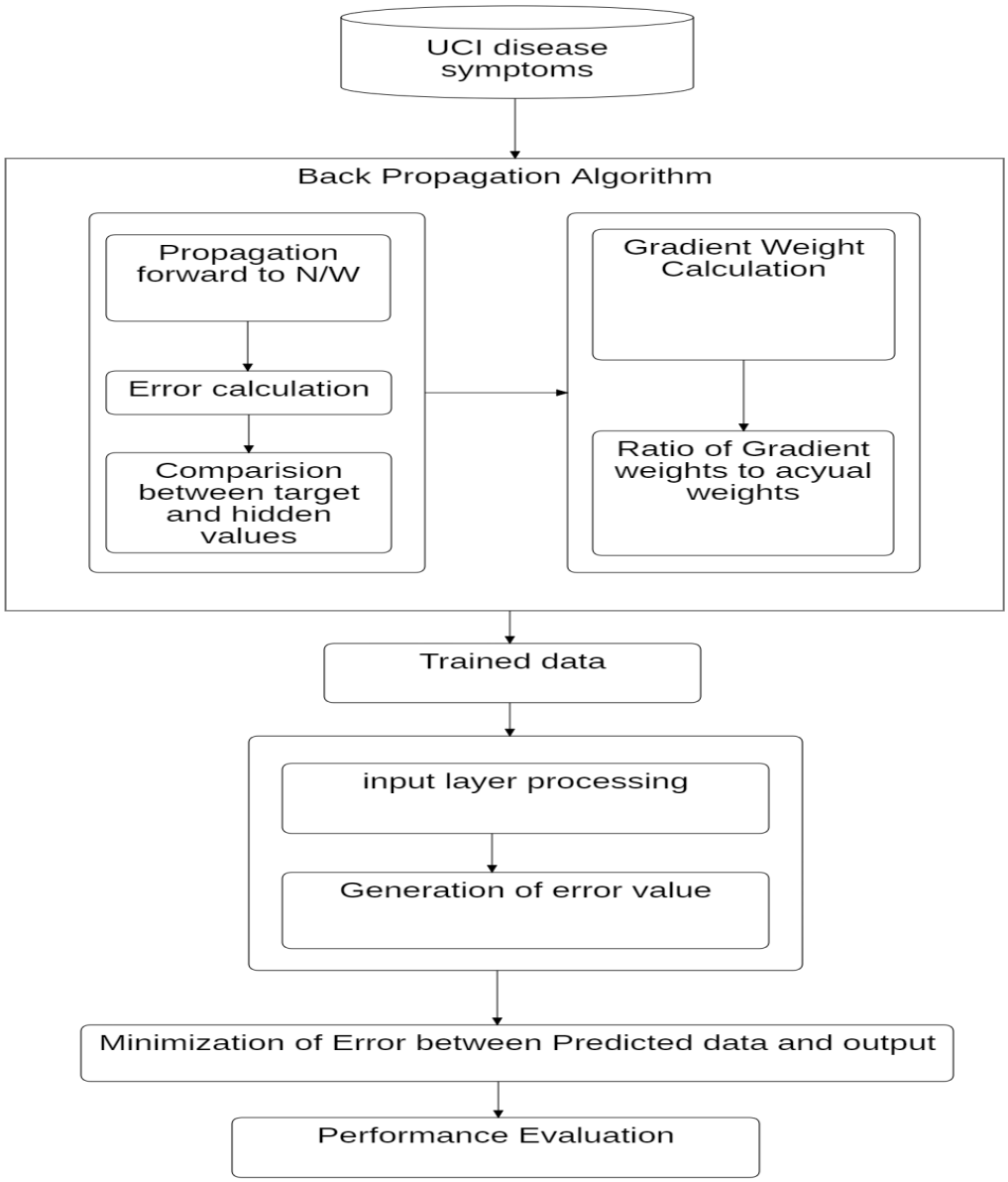

Figure 3.9: Flowchart diagram of Proposed system

3.9.2 Usecase Diagram

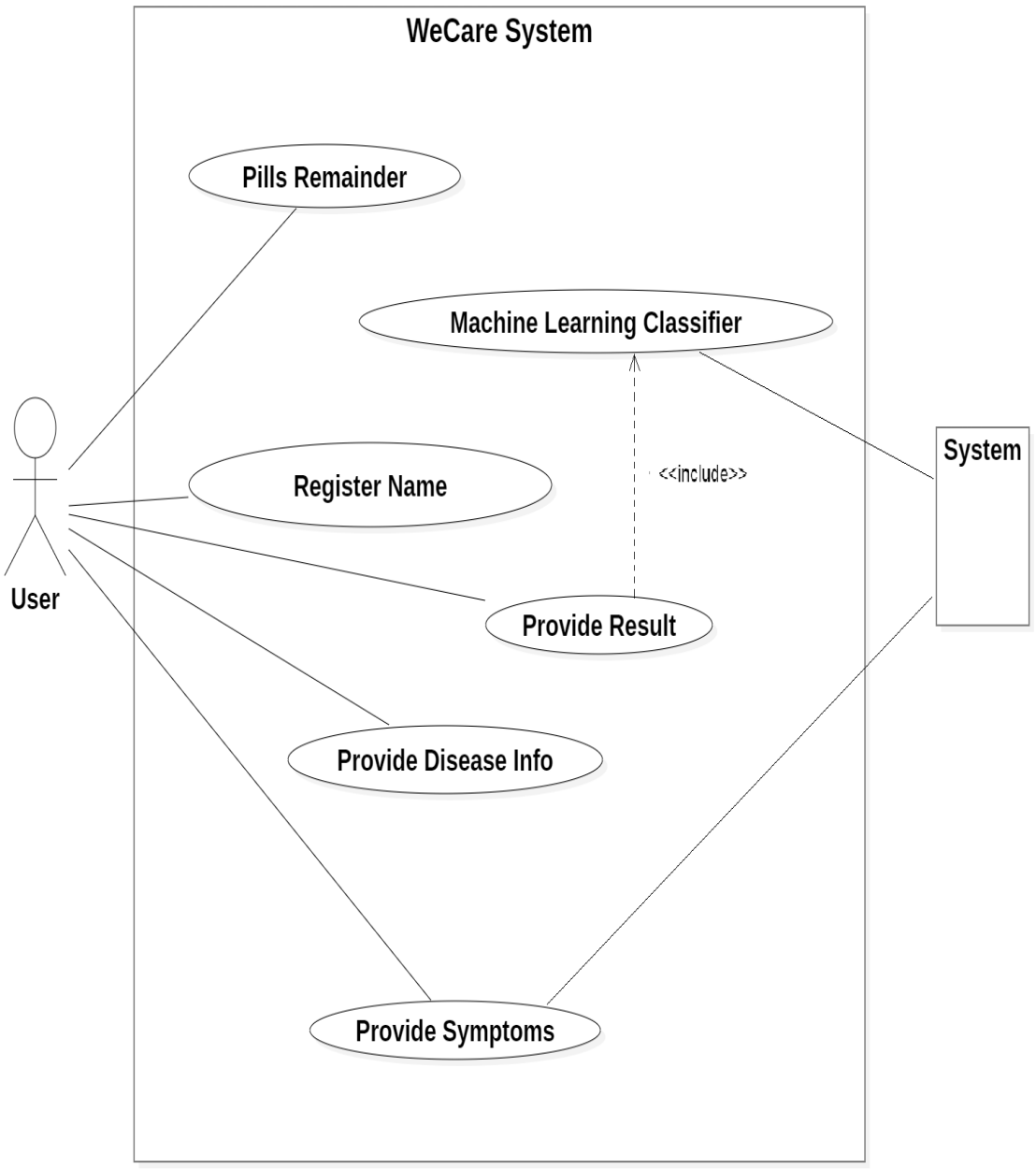

Figure 3.10: UseCase Diagram

3.9.3 Activity diagram

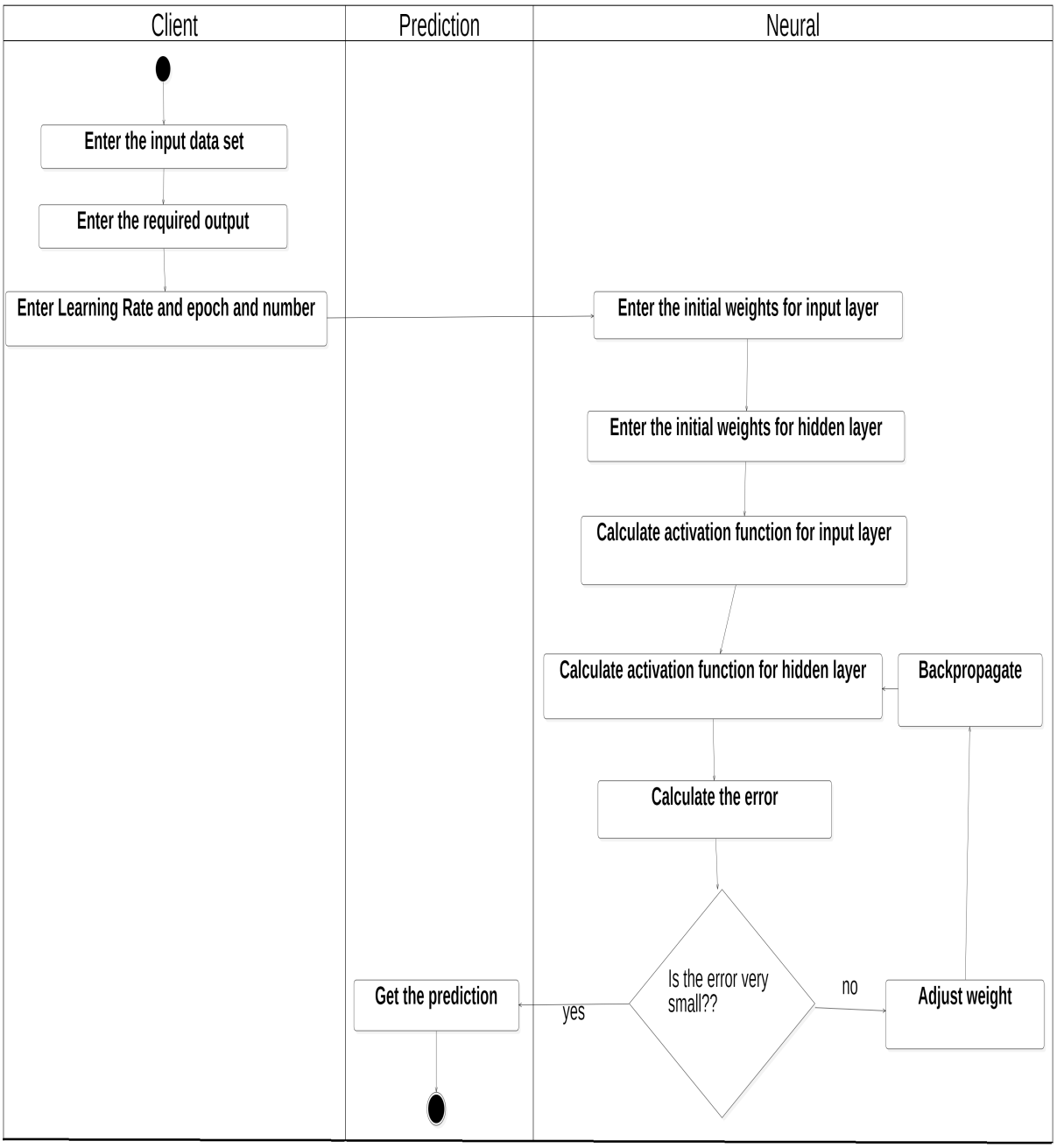

Figure 3.11: Activity Diagram

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Work Completed

4.1.1 Data Collection

Data collection is most crucial phase of any machine learning project because performance of trained model depend upon the quality of collected data sets. Finding the datasets that can be used to trained our model for solving our problem is difficult. Since our proposed system predicate the disease based on the symptoms so, we required dataset that symptoms as features and disease name as predicated class. For our project we got dataset from UCI machine learning repository and since dataset is not in format that we want so we need to format dataset as per our requirement.

4.1.2 Designing UI prototype

Designing UI prototype is essential steps in any project development. Designing UI prototype will gave us mockup of project we wish to build. For our UI prototyping we use Adobe XD software. We designed basic UI where user can enter their symptoms and get their result display.

4.1.3 Selection of optimal Machine learning Algorithm

Since our project is based on Classification problem, we end up with multiple good models to choose from. Each model will have different performance characteristics. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data. Different performance metrics are used to evaluate different Machine Learning Algorithms.

For our project we used Confusion matrix which is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes. The

Confusion matrix in itself is not a performance measure as such, but almost all of the performance metrics are based on Confusion Matrix and the numbers inside it.

Terms associated with Confusion matrix:

- True Positives (TP): True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)
- True Negatives (TN): True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False
- False Positives (FP): False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True)
- False Negatives (FN): False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False).

For each algorithm the accuracy, precision, sensitivity are observed which are described as follows:

1.**Accuracy:** Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.

$$accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

2.**Precision:** This is the fraction of true positives in contrast to the overall correct results is calculated.

$$Precision = \frac{(TP)}{(TP + FP)}$$

3.**Sensitivity:** Sensitivity is the true positive rate and is defined as the number of positive tuples which are correctly classified.

$$Sensitivity = \frac{(TP)}{(TP + FN)}$$

model	accuracy	Precision	Sensitivity
Decision Tree	56.4024390243902	0.577835474112545	0.564024390243902
Back Propagation ANN	84.0243902439024	0.805616287498429	0.840243902439024
Random Forest	67.4186991869919	0.638878424741711	0.674186991869919
Support Vector Machine	54.5615322126126	0.536864506881396	0.520469821721147

Figure 4.1: Performance analysis of different algorithm

The above table depicts the various performance metrics of the classification algorithms Decision tree,Back Propagation ANN, Random Forest and Support Vector Machine on our datasets.

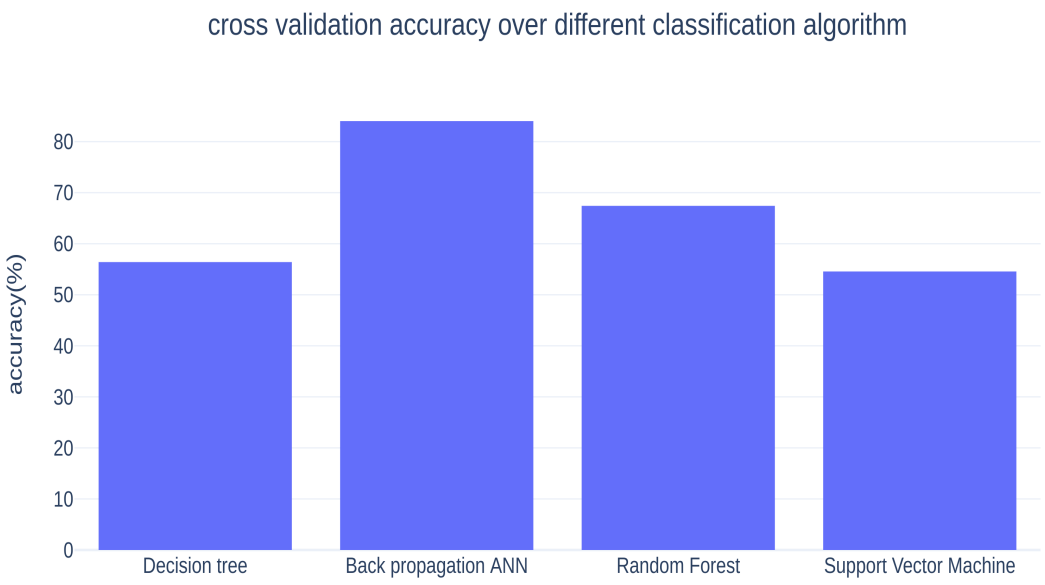

Figure 4.2: Graphical representation of accuracy

The above graph, Figure 4.2 depicts that the back Propagation ANN have the highest accuracy when compared to the other algorithms.

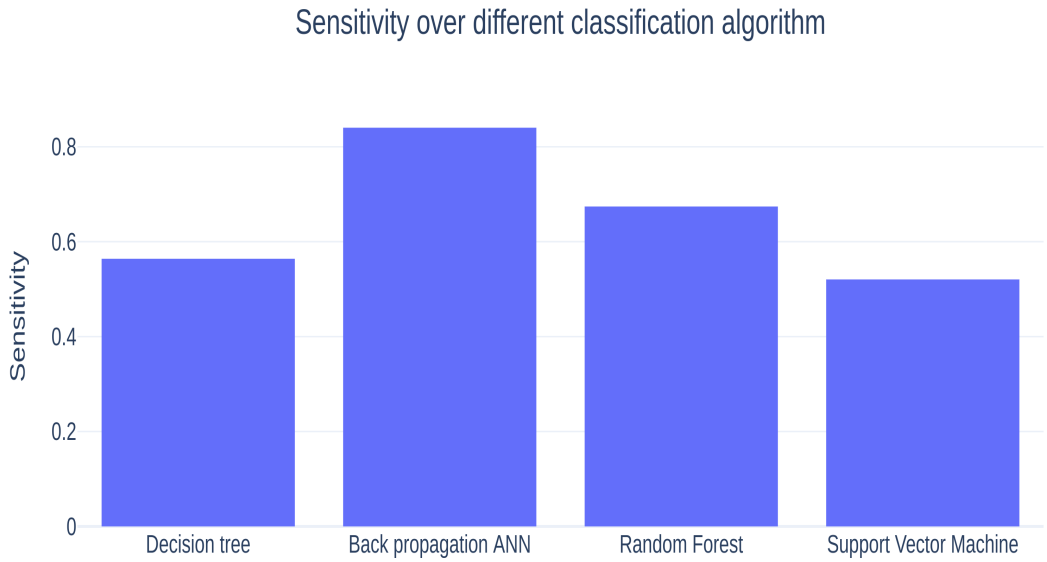

Figure 4.3: Graphical representation of sensitivity

The above graph, Figure 4.3 depicts that back Propagation ANN models have the highest sensitivity.

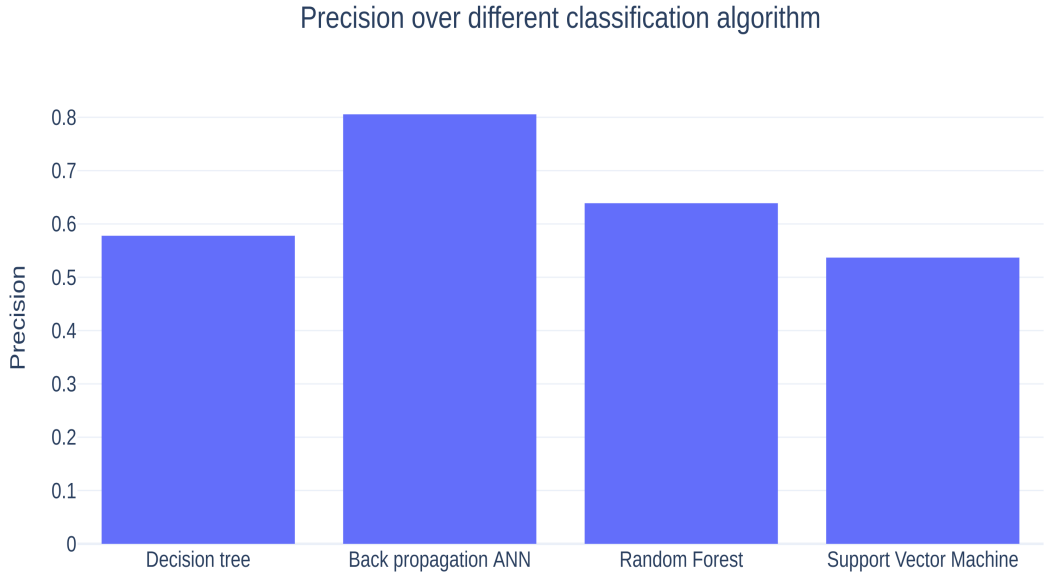

Figure 4.4: Graphical representation of precision

The above graph, Figure 4.4 depicts that back Propagation ANN models have the highest sensitivity.

The proposed system of diseases prediction with appropriate diagnosis has been framed up using Multilayer Perceptron Neural Network. For effective prediction, back propagation algorithm was applied to train the data and compare the parameters iteratively. The propagation algorithm has been repeated until minimum error rate was observed. It is proven from the results that the proposed method effectively predicates the diseases when compared to the other approaches. So our proposed system is able predict the diseases with 84% accuracy using 5 hidden layer. Our predication system gave improved result with high accuracy by using a higher number of hidden layers, but training model with higher hidden layer require a lot of computational power, so due to technical difficulty we are not able to train our model with a higher number of hidden layers.

4.1.4 Selection of optimal hyperparameter value for Modeling

Since we are using Back Propagation Algorithm which is a supervised learning method. Before training this model we need to pass certain hyperparameter such as learning rate, epoch. Right hyper-parameters are crucial to training success.

There is no universal learning rate value which will work for all types of project. So learning rate value is highly dependent on problem at hand and in order to find suitable learning rate value for our project we explore how learning rate affect the model performance.

For this we trained our neural Network with stochastic gradient descent for different value of learning rate and Activation function such as sigmoid, relu, tanh. We used cross validation test method to determine the performance of model on different parameters and we conclude that sigmoid activation function and learning rate=0.6 is optimal for our neural Network model as we got around 88% accuracy while increasing or decreasing learning rate values reduce the accuracy of model.

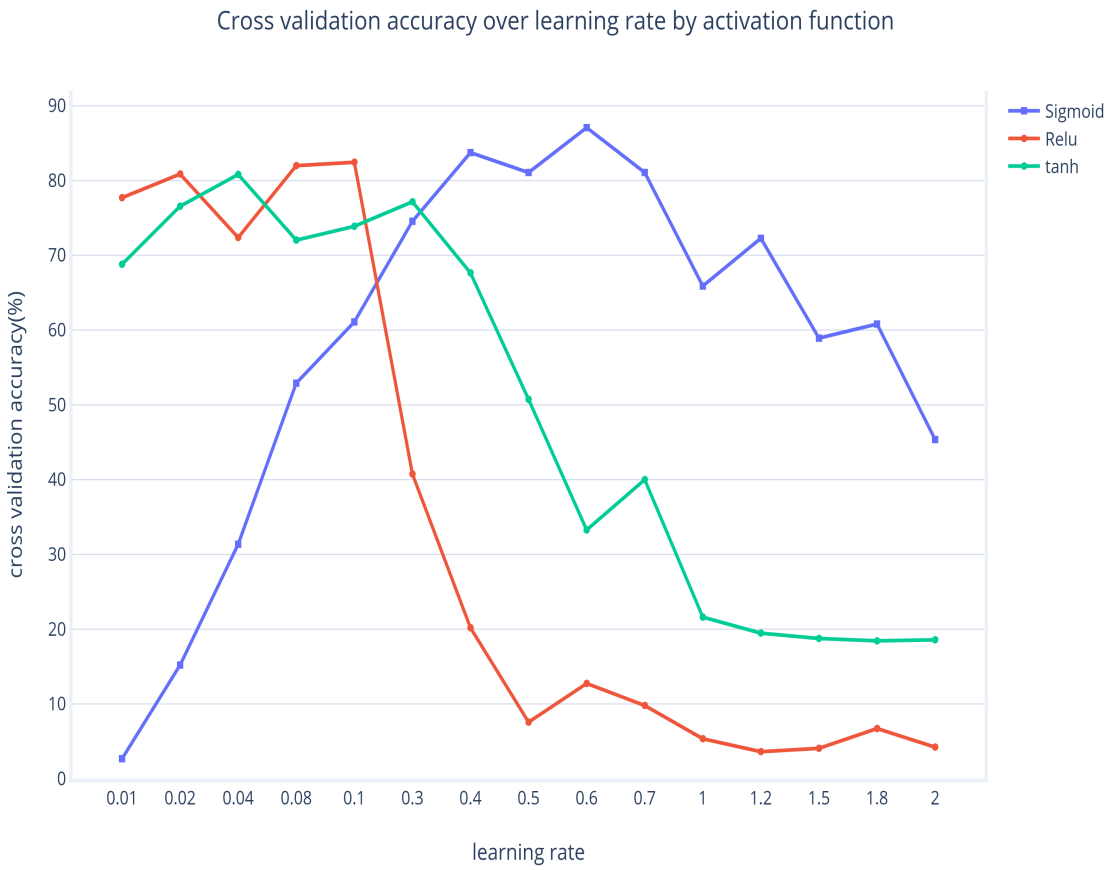

Figure 4.5: Selection of optimal learning rate and activation function

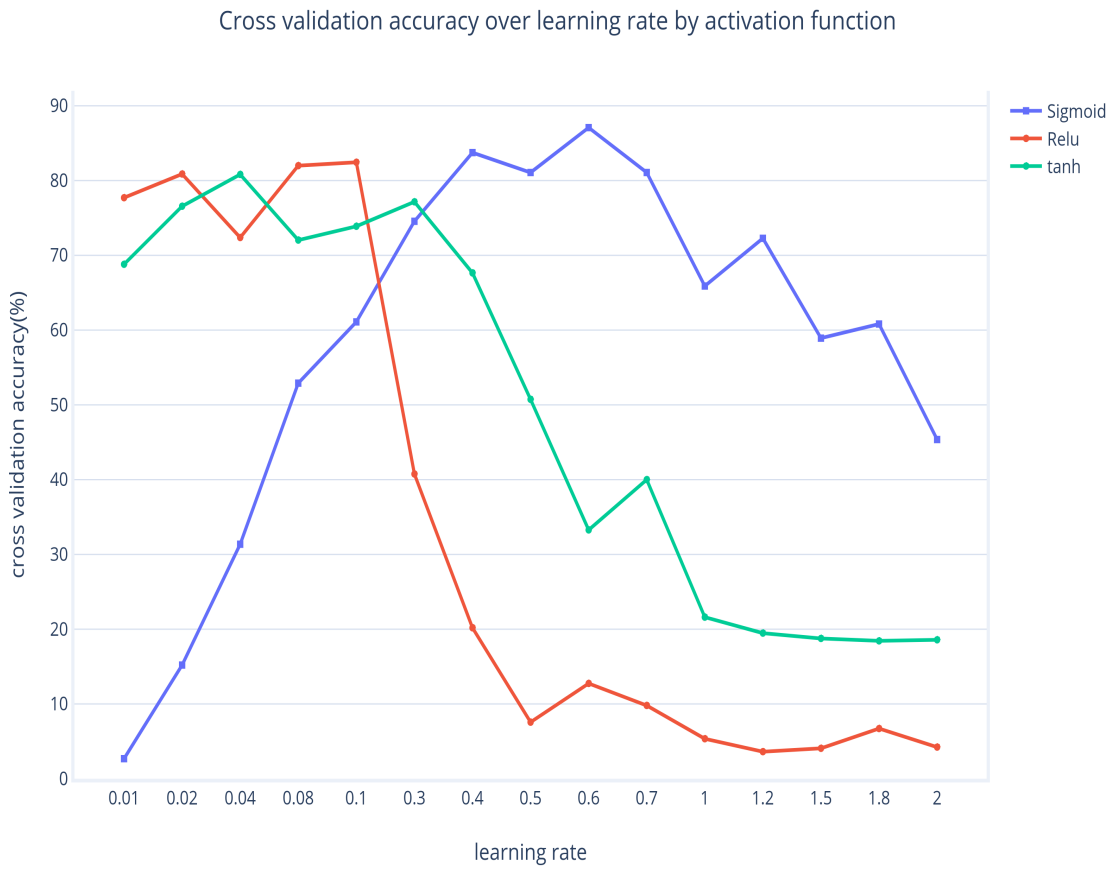

Figure 4.6: Selection of optimal learning rate and activation function

4.1.5 Back Propagation Algorithm implementation using Python

1. **Initialize Network:** Each neuron has a set of weights that need to be maintained. One weight for each input connection and an additional weight for the bias. We will need to store additional properties for a neuron during training, therefore we will use a dictionary to represent each neuron and store properties by names such as ‘weights‘ for the weights. Initially, we randomly set the weights of neuron.

Function named initialize_network that creates a new neural network ready for training. It accepts three parameters, the number of inputs, the number of hidden layer and the number of outputs.

Each neuron has n_inputs + 1 weights, one for each input column in a dataset and an additional one for the bias.

```
def initialize_network(self, n_inputs, n_hidden, n_outputs):
    network = list()
    hidden_layer = [{'weights': [random() for i in range(n_inputs + 1)]} for i in range(n_hidden)]
    network.append(hidden_layer)
    output_layer = [{'weights': [random() for i in range(n_hidden + 1)]} for i in range(n_outputs)]
    network.append(output_layer)
    return network
```

2. **Forward Propagate:** We can calculate an output from a neural network by propagating an input signal through each layer until the output layer outputs its values. We call this forward-propagation.

Forward propagation down into three parts:

1.Neuron Activation.

2.Neuron Transfer.

3.Forward Propagation.

2.1 **Neuron Activation** The first step is to calculate the activation of one neuron on given an input.

Neuron activation is calculated as

activation= sum(weight*input)+bias

```
def activate(self, weights, inputs):
    activation = weights[-1]
    for i in range(len(weights) - 1):
        activation += weights[i] * inputs[i]
    return activation
```

2.2 **Neuron Transfer** Once a neuron is activated, we need to transfer the activation to see what the neuron output actually is. There are different transfer function but for our project we used Sigmoid Activation function.

The sigmoid activation function looks like an S shape, it's also called the logistic function. It can take any input value and produce a number between 0 and 1 on an S-curve. It is also a function of which we can easily calculate the derivative (slope) that we will

need later when backpropagating error.

Output of neuron using sigmoid activation function

output=1/(1+ $\exp^{-activation}$)

Below is a function named transfer that implements the sigmoid equation.

```
def transfer(self, activation):
    return 1.0 / (1.0 + exp(-activation))
```

2.3 **Forward Propagation** output of each neurons is calculate and output of one neurons become input to another neurons.

Function named forward_propagate ( ) that implements the forward propagation for a row of data from our dataset with our neural network. Neuron's output value is stored in the neuron with the name "output". We collect the outputs for a layer in an array named new_inputs that becomes the array inputs and is used as inputs for the following layer.

```
def forward_propagate(self, network, row):
    inputs = row
    for layer in network:
        new_inputs = []
        for neuron in layer:
            activation = self.activate(neuron['weights'], inputs)
            neuron['output'] = self.transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    return inputs
```

3. **Back Propagate Error**

Error is calculated between the expected outputs and the outputs forward propagated from the network. These errors are then propagated backward through the network

from the output layer to the hidden layer, assigning blame for the error and updating weights as they go.

This part is broken down into section

1. Transfer derivative
2. Error backpropagation

3.1 **Transfer derivative**

Given an output value from a neuron, we need to calculate it's slope. We are using the sigmoid transfer function, the derivative of which can be calculated as follows:

derivative = output * (1.0 - output)

Below is a function named transfer_derivative that implements this equation:

```
def transfer_derivative(self, output):
    return output * (1.0 - output)
```

3.2 **Error BackPropagation:**

Here we first calculate the error of each neurons, this will give error signal(input) to propagate backward through the network.

The error of neuron can be calculate as:

error=(expected-output)*transfer_derivatives(output)

where,

expected:Expected value of neuron

output:Output value of neuron

transfer_derivate():calculates the slope of the neuron's output value

This error calculation is used for neuron in output layer and error calculation for hidden layers is bit different and complicated then output layer.

The back-propagated error signal is accumulated and then used to determine the error for the neuron in the hidden layer, as follows:

error = (weight_k * error_j) * transfer_derivative(output)

Where error_j is the error signal from the jth neuron in the output layer, weight_k is the

weight that connects the kth neuron to the current neuron and output is the output for the current neuron.

```
def backward_propagate_error(self, network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network) - 1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] * neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(expected[j] - neuron['output'])
        for j in range(len(layer)):
            neuron = layer[j]
            neuron['delta'] = errors[j] * self.transfer_derivative(neuron['output'])
```

Here the error signal calculated for each neuron is stored with the name ‘delta’. The layers of the network are iterated in reverse order, starting at the output and working backwards. This ensures that the neurons in the output layer have ‘delta’ values calculated first that neurons in the hidden layer can use in the subsequent iteration.

4 **Train Network** Here we train our network will training datasets and for each row data we propagate the data, back propagate error and update the weights. This part is broken down into section

1. Update weights
2. Train network

4.1 **Update weights** Once errors are calculated for each neuron in the network via the back propagation method above, they can be used to update weights.

Network weights are updated as follows:

$$\text{weight} = \text{weight} + \text{learning\_rate} * \text{error} * \text{input}$$

Where weight is a given weight, learning_rate is a hyperparameter we specify, error is the error calculated by the backpropagation procedure for the neuron and input is the

input value that caused the error. The same procedure can be used for updating the bias weight, except there is no input term, or input is the fixed value of 1.0.

4.1.6 RESTFul API using Flask

After training our neural network model we saved our neural network, which is the collection weights value of different layer of the network. Then we create a RESTful API (application program interface) which allow users to send their symptoms in HTTP request and get a response back in JSON format.

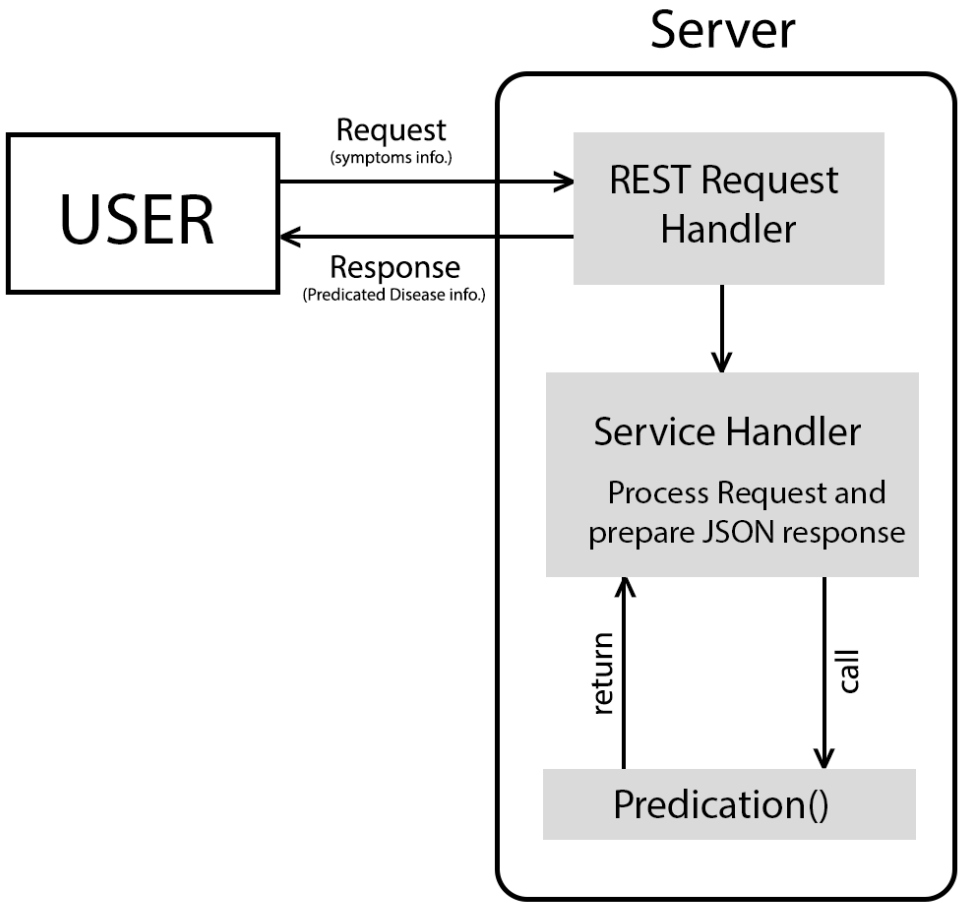

Figure 4.7: Restful API

In order to create RESTFul API we used flask in server side which is micro framework because it does not require particular tools or libraries. After that we have created route which will listen to certain HTTP request. This allowed user to send their symp-

toms information via HTTP request and route function will parse the data (symptoms information) from the HTTP request. Then we call predication function which neural network model, symptoms as parameters and return predicated disease information back to user.

4.1.7 Prototype

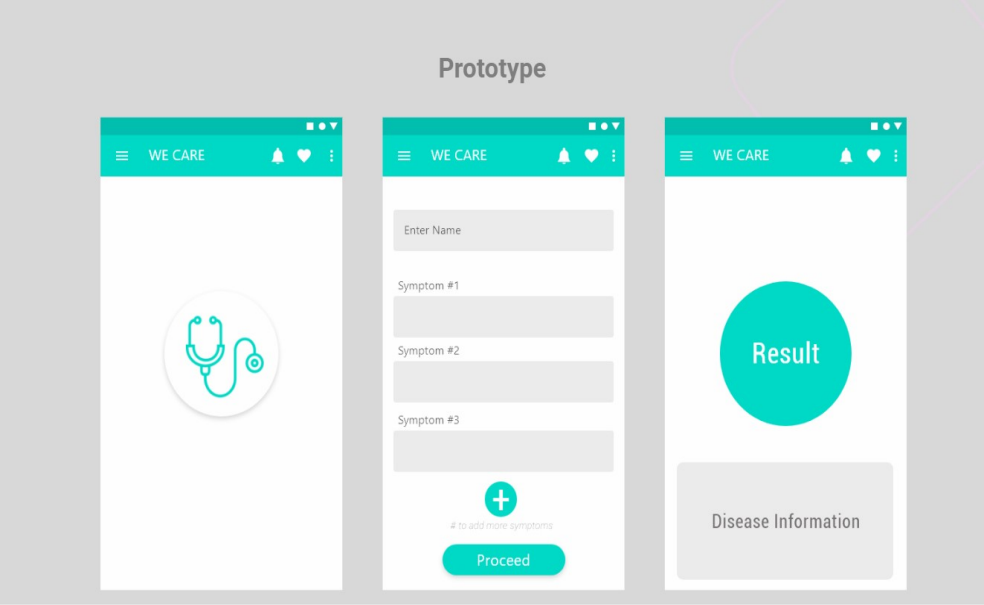

Figure 4.8: Prototype

4.2 Work Remaining

Till now we have completed collecting appropriate dataset containing disease-symptoms data, Designing prototype, implementation of back propagation of our application and simple android application. Still some task is need to be accomplish in order to develop fully functional Application.

- Improving the Accuracy of our back propagation Model
- Creation of interactive UI
- Pill Reminder

4.3 Limitations

In spite of all hardwork, our project consist some limitation. Our project has lack of reliable information. Users cannot describe the detailed information such as if someone is having fever he/she could not describe the temperature and also if someone having pain in any part of the body, how much it is paining. There is only limited number of disease which our project can predict. Users cannot enter their symptoms on their own.

A machine learning model can provide more accurate outcome if the number of hidden layer is increased. But the increase in number of hidden layers leads to computational complexities which cannot be operate by our normal computer. So, we had to use only 5 number of hidden layer. We can use more hidden layer but the complexities and consumption of time will also increases. There is no design guidelines and accuracy depends on training and learning which is not always available. It is hard to maintain degree of meaningfulness and also hard to combine cases together. Predictions are limited to the cases that have been observed. They require accurate details on many past projects. They have large data requirement to learn about various topics which may be time taking and cause various resources.

The actual problem with this inevitable fact is that when they do make errors, diagnosing and correcting them can be difficult because it will require going through the underlying complexities of the algorithms and associated processes. It is impossible to make immediate accurate predictions with a back propagation. It always learns through historical data. The bigger the data and the longer it is exposed to these data, the better it will perform. There is lack of variability. Back propagation deals with statistical truths rather than literal truths. In situations that are not included in the historical data, it will be difficult to prove with complete certainty that the predictions made is suitable in all scenarios. Unlike humans, computers are not good storytellers. Back propagation cannot always provide rational reasons for a particular prediction or decision. They are also limited to answering questions rather than posing them. These systems does not understand context. Depending on the provided data used for training, machine learning is also prone to hidden and unintentional biases. Human input is still important to

better evaluate the outputs of these systems.

4.4 Challenges

During our project, we have faced some challenges which has been overcome. Those challenges are explained below:

- To train a machine learning model, we need large training datasets so that the prediction could be more accurate. It required time to collect a sufficient amount of data. Collecting appropriate data was most difficult task. Hospitals or any Health Organization may unwilling to share their dataset with us or may issue a formal complaint against us if when they realize that we have use their dataset. Even after collecting the appropriate data, there are also problems of a different nature like preparing data for algorithm training is a complicated process.
- For our project, we have to be familiarize with many software like:
  - Pycharm
  - Juypter NoteBook
  - Visual Studio Code
  - Android Studio

  Due to limited knowledge about software, we find very hard to get start with the project. Since our project is based on machine learning, Machine learning comes with a set of predefined recipes called algorithms that are best suited for solving a particular problem. For example, choosing between K-Nearest Neighbor algorithm, Naive Bayes Algorithm, Random forest algorithm, Decision Tree Algorithm and Back Propagation Algorithm can be confusing to a beginner. Like most of the branches in computer science, ML offers multiple techniques to solve the same problem. So, we have to use different evalution techniques for selection of optimal machine learning algorithm for our project. After day-by-day using those softwares, we understood and familiarize with those software and able to the get appropriate results.
- We also find difficult to choose optimal algorithm for prediction and hyperparameter for training of Neural Network. We have used 4 different algorithms

which are: Naive Bayes Algorithm, Random forest algorithm, Decision Tree Algorithm and Back Propagation Algorithm to train our Neural Network. After using these algorithms for training, we have noted the final outcome of each algorithms. Among those algorithms, Back Propagation Algorithm shows the best result. So, we carried our project with Back Propagation Algorithm.

4.5 Work Schedule

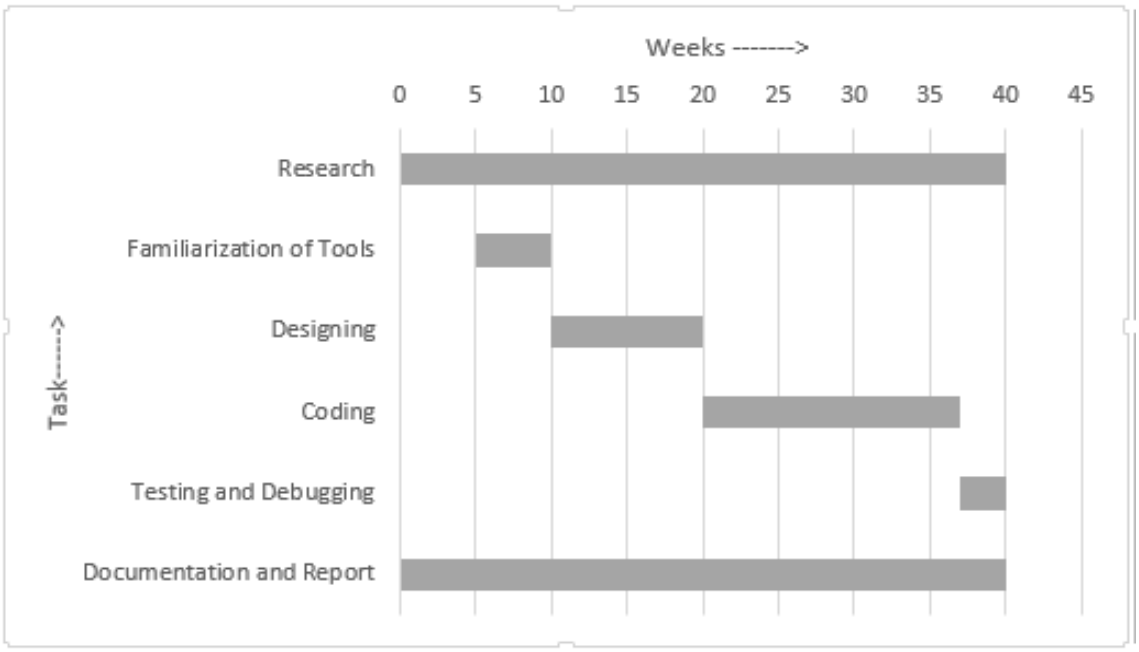

Figure 4.9: Gantt Chart

4.6 Conclusion

Our proposed system of disease prediction with appropriate diagnosis has been framed up using Artificial Neural Network. For effective prediction, back propagation algorithm was applied to train the data and compare the parameters iteratively. The back propagation algorithm has been repeated until minimum error rate was obseved. It is proven from the results that the proposed method, back propagation artificial neural network effectively predicts accuracy around 85.02%. Tuning the hyperparameters like learning rate, epoch etc for increasing the performance model for our project is still in process.

4.7 Future Enhancements

For now our project is capable of only determining/predicting diseases according to provided symptoms only. The same architecture could be extended to predict other diseases as well after analysing the various symptoms of those diseases in consultation with the concerned health experts.

Our project has got a lot of potential; for now due to limitation to our knowledge and time, we have proposed an able system to predict provided diseases only.

In future additional extra features can be added such as;-

- Contacts: Contact number of health experts, doctors, emergency vehicles can be provided for users as they can enquiry about their problems directly.
- Map: Nearby location of health posts, medical facilities, hospitals can be shown around the users.
- User Interface: Good user interface between system and user.
- Chatbot: Our project provides only selecting of symptoms which can predict certain diseases so our project is not so expressful, but the implementation of natural language processings allows user to provide symptoms helps in their preferential language.
- Better Prediction: Past records can be use for better prediction and can be used as references for future prediction as well.

REFERENCES

[1] Shreevastava M. And Gupta A. (2011) " Medical Diagnosis using Back Propagation Algorithm", International Journal of Emerging Technology and advanced Engineering, ISSN 2250-2459, Vol. 1, Issue 1, November 2011.

[2] Taufik, Wan Ab Ghani, Nur Laila Mohd Drus, Sulfeeza. (2019). Data Mining Techniques for Disease Risk Prediction Model: A Systematic Literature Review: Proceedings of the 3rd International Conference of Reliable Information and Communication Technology (IRICT 2018). 10.1007/978-3-319-99007-1_4.

[3] Kadhim Qeethara (2011), "Artificial Neural Networks in Medical Diagnosis". International Journal of Computer Science, Vol. 8, Issue 2, March 2011

[4] Sumathi B., Santhakumaran A. (2011), "Pre-Diagnosis of Hypertension using Artificial Neural Network", Global Journal of Computer Science and Technology, Vol. 11, issue 2, version 1.0, February 2011.

[5] Al-Shayea, Qeethara. (2011). Artificial Neural Networks in Medical Diagnosis. Int J Comput Sci Issues. 8. 150-154.

[6] Escobar GJ, Turk BJ, Ragins A, et al. Piloting electronic medical record-based early detection of inpatient deterioration in community hospitals. J Hosp Med. 2016;11(1):S18–S24.

[7] Dilip Roy Chowdhury, Mridula Chatterjee R.K. Samanta, An Artificial Neural Network Model for Neonatal Disease Diagnosis, International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume (2): Issue(3),2011.

[8] Milan Kumari, Sunila Godara, Comparative Study of Data Mining Classification Methods in Cardiovascular Disease Prediction, IJCST Vol. (2), Issue (2), June 2011.

[9] Vanisree K, Jyothi Singaraju, Decision Support System for Congenital Heart Disease Diagnosis based on Signs and Symptoms using Neural Networks, International Journal of Computer Applications (0975 8887) Volume 19 No.6, April 2011.

[10] Niti Guru, Anil Dahiya, Navin Rajpal, Decision Support System for Disease Diagnosis Using Neural Network, Delhi Business Review, Vol.8, No.1, January-June 2007.

[11] Sellappan Palaniappan, Rafiah Awang, Intelligent Heart Disease Prediction System Using Data Mining Technique, 978-1-4244-1968-5/08/, 2008 IEEE.

[12] Smitha.T,Dr.V.Sundaram"Classification Rules By DecisionTree for disease Prediction", ,International journal of Computer Applications vol-43, No-8, April 2012.