Disease Prediction Based on Symptoms Using Machine Learning

A PROJECT REPORT

Submitted by

ABHINAYA R (715517104003)

AISHWARYA S (715517104005)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH, COIMBATORE 641 062

ANNA UNIVERSITY: CHENNAI - 600 025

JUNE 2021

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DISEASE PREDICTION BASED ON SYMPTOMS USING MACHINE LEARNING"** is the bonafide work of **"ABHINAYA R (715517104003), AISHWARYA S(715517104005)"** who carried out the project work under my supervision.

----------	**----------**
SIGNATURE	**SIGNATURE**
Dr. R. Manimegalai	Ms. G. Niranjani
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Assistant Professor(Senior Grade)
Computer Science and Engineering	Computer Science and Engineering
PSG Institute of Technology and	PSG Institute of Technology and
Applied Research,	Applied Research
Coimbatore – 641 062	Coimbatore – 641 062

Submitted for the project viva-voce Examination held on _____

----------	**----------**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I express my sincere thanks to **Shri. L Gopalakrishnan, Managing Trustee, PSG & Son's Charities.**

I am very grateful to **Dr. G. Chandramohan**, **B.E(Hons), M.Tech, Ph.D, Principal** for providing me with an environment to complete our project successfully.

I also take the privilege of expressing my gratitude to **Dr. P.V. Mohanram, B.E(Hons), M.Tech, Ph.D, Secretary** in extending his support to carry out my study.

I am greatly indebted to **Dr. R. Manimegalai, M.E, Ph.D, Head of the Department, Computer Science and Engineering** for her guidance which was instrumental in the completion of my project.

I express my sincere thanks to **ProjectGuide Ms. G. Niranjani, M.Tech, Assistant Professor(Senior Grade)** for her constant encouragement and support throughout the course.

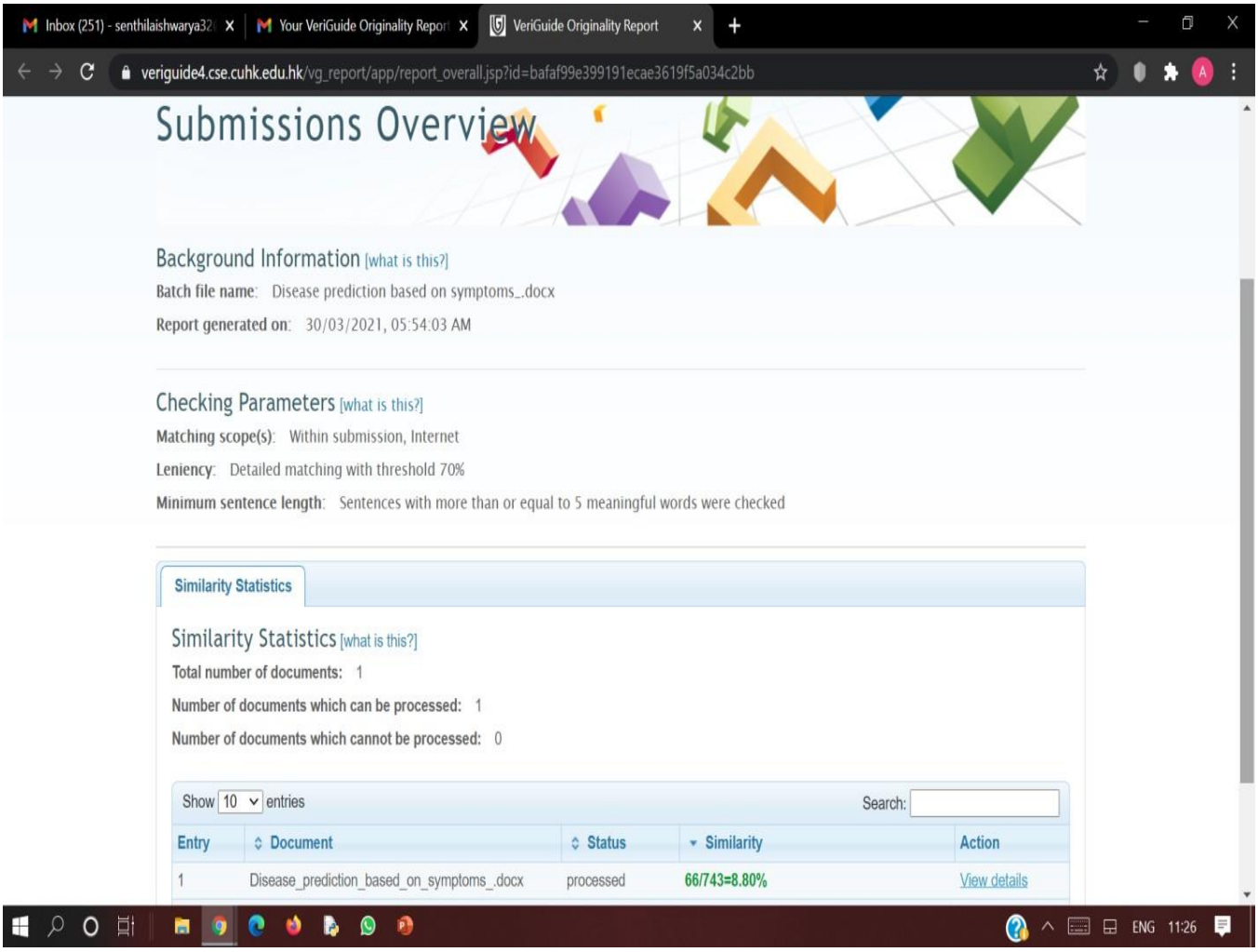
Also, I earnestly thank my **Project Coordinator Mr. S. Thivaharan, M.Tech, Assistant Professor (Selection Grade)** for his consistent support throughout the course.

Finally, I take this opportunity to extend my deepest appreciation to my family and friends, who supported me during the crucial times of my project.

- ABHINAYA R

AISHWARYA S

VERIGUIDE ANALYSIS REPORT



ABSTRACT

The general day to day health of a person is vital for efficient functioning of the human body. Considering certain prominent symptoms and their diseases to build a Machine learning model that can predict common diseases based on real symptoms is the objective of the project. Using the dataset of the most commonly exhibited diseases, we built a relation to predict possible diseases based on the symptoms. The proposed model utilizes the capabilities of different machine learning algorithms to achieve accurate prediction. In health industry, it provides several benefits such as detection of diseases and faster diagnosis. With the rise in number of patients and diseases every year, medical system is overloaded and have become overpriced in many countries. Most of these diseases involve a consultation with doctors to get treatment. With sufficient data, prediction of a disease by an algorithm can be very easy and cheap. Prediction of a disease using the symptoms is an integral part of treatment. This project accurately predicts a disease by looking at the symptoms of the patient. Such a system can have a very large potential in medical industry. This project also has an interactive interface to facilitate interaction with the system and display the result of our study.

Keywords : Machine learning, Disease prediction, Health care.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
CNN	CONVOLUTION NEURAL NETWORKS
CSV	COMMA SEPARATED VALUES
DT	DECISION TREE
GUI	GRAPHICAL USER INTERFACE
KNN	K – NEAREST NEIGHBOUR
LR	LOGISTIC REGRESSION
ML	MACHINE LEARNING
NB	NAÏVE BAYES
RF	RANDOM FOREST
SVM	SUPPORT VECTOR MACHINES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO DISEASE PREDICTION BASED ON SYMPTOMS

Healthcare is one of the most important research fields with the rapid improvement of technology and increase in data. It is difficult to handle huge amount of data of the patients. Big Data Analytics can be used to handle such data. There are a lot of procedures for the treatment of multiple diseases across the world. Machine Learning is a prominent approach that helps in prediction and diagnosis of a disease. The project presents the idea of a prediction of a disease based on symptoms using machine learning. Machine Learning algorithms such as Naive Bayes, KNN, Decision Tree and Random Forest are employed to predict the disease. Its implementation is done in Python programming language.

1.2 OBJECTIVE

Accurate and on-time analysis of any health-related problem is important for the prevention and treatment of a disease. The traditional way of diagnosis may not be sufficient in the case of a serious ailment. Developing a medical diagnosis system based on machine learning algorithms for prediction of any disease can help in a more accurate diagnosis than the conventional method. The objective of this project is to build a machine learning model to predict the disease based on symptoms using multiple machine learning algorithms.

1.3 PROBLEM STATEMENT

There are many people who search online for health related information, diagnosis and different treatments. A recommendation system will be useful for this purpose. The objective of this project is to build a machine learning model to predict

a disease based on its symptoms. This proposed model utilizes the capabilities of different machine learning algorithms to achieve accurate prediction.

1.4 PROJECT OVERVIEW

This project presents the idea of prediction of a disease based on symptoms using machine learning. Machine Learning algorithms such as Naive Bayes, KNN, Decision Tree and Random Forest are employed on the provided dataset to build the model and predict the disease.

1.5 SCOPE AND MOTIVATION

The number of patients and diseases are increasing every year. The disease can be predicted with sufficient data using Machine Learning algorithms. This project accurately predicts a disease based on the symptoms given by the user. This system can have large potential in the medical industry. This project also has an interactive interface to facilitate user interaction with the system and display the result of our study.

1.6 INTRODUCTION TO MACHINE LEARNING

Machine learning allows software applications to become more accurate in predicting outcomes without being explicitly programmed. Machine learning is of three types - Supervised Learning, Unsupervised Learning, Reinforcement Learning. Supervised learning is training the model by providing training, input and output patterns to the systems. Unsupervised learning is a self-learning technique in which system has to discover the features of the input population on its own and no prior set of categories are used. Reinforcement learning is the training of machine learning models to make a sequence of decisions. Some applications of machine

learning are medical diagnosis, image recognition, traffic prediction, product recommendations, self driving cars, speech recognition.

1.7 MACHINE LEARNING IN DISEASE PREDICTION BASED ON SYMPTOMS

Machine Learning is a prominent approach that helps in prediction and diagnosis of a disease. Machine Learning algorithms such as Naive Bayes, K–Nearest Neighbour, Decision Tree and Random Forest are employed on the dataset to build the model for the prediction of disease.

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is a classification algorithm for binary (two-class) and multi-class classification problems.

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

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

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and considers the average to improve the predictive accuracy of the model. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING SYSTEMS

The number of papers dealing with disease prediction based on symptoms in literature is growing exponentially. Several researchers have played a significant role in the development of disease prediction algorithms.

D. Dahiwade et al., [1] in 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, designed a disease prediction model using machine learning approach. Due to increased amount of data growth in medical and healthcare field the accurate analysis on medical data has benefits on early patient care. Data mining finds hidden pattern information in the huge amount of medical data. In this literature, the aim is to recognize trends across various types of supervised Machine learning models in disease detection through the examination of performance metrics. The most prominently discussed ML algorithms are Convolution Neural Network(CNN), K-Nearest Neighbour (KNN). It is a general disease prediction based on symptoms of the patient. The accuracy of general disease prediction by using CNN is 84.5% which is more than KNN algorithm.

S. Grampurohit et al., [2] in International Conference for Emerging Technology (INCET), Belgaum, India, 2020, proposed accurate analysis of medical database benefits in early disease prediction, patient care and community services. The techniques of machine learning are successfully employed in assorted applications including disease prediction. The aim of developing classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage.

In this literature a sample data of 4920 patients records diagnosed with 41 diseases was selected for analysis. A dependent variable composed of 41 diseases. 95 of 132 independent variables (symptoms) closely related to diseases were selected and optimized. This research work carried out that demonstrates the disease prediction system developed using Machine learning algorithms such as Decision Tree classifier, Random forest classifier, and Naïve Bayes classifier. The paper presents the comparative study of the results of the above algorithms used.

Hong Qing Yu [3] in IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2019, introduced Experimental Disease Prediction on Combining Natural Language Processing and Machine Learning. He proposed a framework to evaluate the efficiency of applying both Machine Learning and Natural Language Processing technologies for disease prediction system. He used modern computational methods to develop and analyse new approaches that can efficiently predict the disease with reasonable accuracy.

S. Vijava Shetty et al., [4] in International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, designed symptom based health prediction system using data mining. Taking certain prominent symptoms and their diseases to build a Machine learning model to predict common diseases based on real symptoms is the objective of this research.

Feixiang Huang et al., [5] in IEEE International Conference on Granular Computing, Hangzhou, China, 2012 proposed a model to predict a disease by using data mining based on healthcare information system. This paper applies the data mining process to predict hypertension from patient medical records with eight other diseases.

A. Gavhane et al., [6] in Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018 has discussed about prediction of heart disease using machine learning. The problem is

solved using emerging technologies like deep learning to get good results in terms of speed and Accuracy. This work investigated and showed the potential of using DNN-based data analysis for detecting heart disease based on routine clinical data. DNN data analysis techniques can yield very high accuracy.

M. Shankar et al., [7] in Second International Conference on Advances in Computing and Communication Engineering, Dehradun, India, 2015 introduced a method for disease recognition and cure time prediction based on symptoms and proposed a novel method for recognition of diseases and prediction of their cure time based on the symptoms. For predicting the cure time of a disease, reinforcement learning is used. The algorithm takes into account the similarity between the condition of the current user and other users who have suffered from the same disease.

M. Chen et al., [8] in IEEE, 2017 developed a disease prediction model based on machine learning over big data from healthcare communities using convolutional neural network (CNN)-based multimodal disease risk prediction algorithm using structured and unstructured data from hospital. To overcome the difficulty of incomplete data, we use a latent factor model to reconstruct the missing data. Regional chronic disease of cerebral infarction were experimented. The prediction accuracy of our proposed algorithm reaches 94.8%.

With the dataset of the most commonly exhibited diseases, they built a relation to predict the possible disease based on the symptoms which were given as input. The proposed model utilizes the capability of different Machine learning algorithms combined with text processing to achieve accurate prediction. In health industry, it provides several benefits such as pre-emptive detection of diseases, faster diagnosis, medical history for review of patients.

2.2 DRAWBACKS IN THE EXISTING SYSTEM

- The system involves complex processing methods.
- Lot of steps are involved and all these steps are repeatedly used in a loop to identify the disease.
- The size of the software is usually large.
- A generalised approach is not seen in some of the above projects. They have been designed only for a particular disease.

2.3 PROPOSED SYSTEM

The proposed system uses machine learning algorithm to predict the disease based on their symptoms. A machine learning model is built by taking certain prominent symptoms and their diseases. The dataset of the most commonly exhibited diseases is collected to build the model. The model is built to predict the possible disease based on the symptoms which are given as input. The proposed model utilizes the capability of different Machine learning algorithms to achieve accurate prediction. The system uses Naive Bayes, K – Nearest Neighbor, Decision Tree and Random Forest algorithm. This system does not require hardware components and interfacing. Hence it reduces the cost and also code can be reused. The total size of the software is very small i.e. it can be installed in any system.

2.4 BENEFITS OF THE PROPOSED SYSTEM

- The project eliminates the use of an extra hardware.
- This system can be effectively used anywhere.
- The code is very simple and can be enhanced with additional parameters to make the system more effect

CHAPTER 3
SYSTEM DESCRIPTION

The language used to code the algorithm is python. Machine learning models are trained using dataset which contains symptoms and its diseases and tested using testing dataset. The user enters the symptoms as input. The symptoms are passed to the machine learning models to detect the disease. The detected diseases are displayed on the screen.

3.1 PYTHON

Python is an interpreted, high – level programming language. Integrated Development and Learning Environment (IDLE) is an integrated development environment (IDE) for Python. It allows the programmers to write Python code. Like Python Shell, IDLE can be used to execute a single statement. It gives the ability to create, modify, and execute Python scripts. IDLE provides text editor to create Python scripts that has features like smart indent, syntax highlighting and,autocompletion. It also has a debugger with stepping and breakpoints features.

3.2 NUMPY

NumPy is a Python programming language library, that adds support for multi-dimensional arrays and large matrices. It has huge collection of high-level mathematical functions to work on the arrays. It is used to perform mathematical operations on arrays such as algebraic, statistical and trigonometric routines.

3.3 PANDAS

Pandas is a Python programming language library which provides data manipulation and analysis tools for Python programming language. It is free

software released under BSD license. Pandas allows us to import data from various file formats such as Microsoft Excel , comma-separated values (csv), SQL and JSON, . Pandas provides various data manipulation operations such as reshaping, selecting, merging, as well as data cleaning and wrangling features.

3.4 TKINTER

Tkinter is the commonly used library for creating Graphical User Interface in Python. This framework allows users with a simple way to create GUI elements using the widgets found which is found in the Tk toolkit. Tk widgets can be used to construct menus, data fields, buttons etc. in the Python application.

3.5 SKLEARN

Scikit-learn (Sklearn) is a software machine learning library developed for python programming language. The sklearn library contains many tools for statistical modelling and machine learning including clustering, regression, classification, and dimensionality reduction. It features various algorithms like k-neighbours, random forests and support vector machines. It also supports Python scientific and numerical libraries like SciPy and NumPy.

3.6 SQLITE3

The sqlite3 module is a part of the Python standard library. It allows us work with a fully featured on-disk SQL database without installing any additional software. To connect python with sqlite3, establish a connection to the SQLite database by creating a Connection object. Next, a Cursor object is created using cursor method of the connection object. Then, SQL queries are executed.

CHAPTER 4
SYSTEM DESIGN

4.1 COLLECTION OF DATA

The dataset for this project has been collected from various open-source websites. The dataset collected were divided into training set and a testing set. A training set is a dataset used to train the model. In training the model, specific features are picked out from the training set. These features are then incorporated into the model. The test set is a dataset used to measure how well the model performs at making predictions on that test set.

4.1.1 Attributes Required for Disease Prediction based on Symptoms Using Machine Learning

Various symptoms and its diseases were collected from open source. Attributes are symptoms and its diseases

4.1.2 Description of Dataset

For the base version of this project, 261 diseases and 500+ symptoms are downloaded from open source. The number of symptoms and diseases in training and testing are listed as a table 4.1 below:

Table 4.1 Description of dataset

Attributes	**Training Set**	**Test Set**
Symptoms and its Diseases	4921	1121

4.2 MODEL DEVELOPMENT

The model is developed using Machine Learning Algorithms. Machine Learning is the research field where the computers have the capability to learn without being explicitly programmed. It is based on the idea that systems can learn from data, identify patterns in the data and make predictions with minimal human intervention. Machine learning is used in websites to make personalized recommendations , email filters to sort out spam, web search engines, banking software and lots of apps such as voice recognition.

4.3 CHOICE OF MACHINE LEARNING ALGORITHM

The Machine Learning algorithms used in this project are Naive Bayes, K-Nearest Neighbour, Decision Tree and Random Forest Algorithm. These algorithms have proven to be very effective in prediction of diseases. In this project, the proposed model utilizes the capabilities of these machine learning algorithm to achieve accurate prediction.

4.3.1 Naive Bayes Algorithm

Naive Bayes algorithm is a supervised machine learning algorithm, based on Bayes theorem. It is mainly used in text classification which includes a high-dimensional training dataset. Naive Bayes Classifier is one of the simple and most effective algorithms which helps in building models that can make quick predictions**.** It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some examples of Naive Bayes Algorithm are Sentimental analysis, classifying articles and spam filtration**.**

4.3.1.1 Bayes Theorem

Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It serves as a way to figure out conditional probability.

Given a Hypothesis H and evidence E, Bayes' Theorem states that the relationship between the probability of Hypothesis before getting the evidence P(H) and the probability of the hypothesis after getting the evidence P(H|E) is :

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Figure 4.1 Bayes Theorem

This relates the probability of the hypothesis before getting the evidence P(H), to the probability of the hypothesis after getting the evidence, P(H|E). P(H) is called the prior probability, and P(H|E) is called the posterior probability. The factor that relates the two, P(H|E) / P(E), is called the likelihood ratio.

4.3.1.2 Steps involved in Naive Bayes Algorithm

1) Calculate prior probability for given class labels.
2) Calculate conditional probability with each attribute for each class
3) Multiply same class conditional probability
4) Multiply prior probability with same class conditional probability
5) See which class has higher probability, higher probability class belongs to given input set.

4.3.1.3 Types of Naive Bayes Model:

There are three types of Naive Bayes Model, which are given below:

Gaussian: The Gaussian model assumes that the features follows normal distribution.

Multinomial: The Multinomial Naive Bayes classifier is used when the features are multinomial distributed.

Bernoulli: The Bernoulli classifier is similar to the Multinomial classifier, but the predictor variables are independent Boolean variables.

4.3.1.4 Applications of Naive Bayes

Medical Diagnosis:

Figure 4.2 Application of Naive Bayes

[Source : https://www.edureka.co/blog/naive-bayes-tutorial/**]**

Nowadays modern hospitals are equipped with monitoring and other data collection devices resulting in huge amount of data which are collected through health examination and medical treatment. Naive Bayes classifier takes into account evidence from many attributes to make the final prediction and provides transparent explanations of its decisions.

Weather Prediction :

Figure 4.3 Application of Naive Bayes

[Source : https://www.edureka.co/blog/naive-bayes-tutorial/**]**

Weather is one of the most influential factors in our daily life, to an extent that it may affect the economy of a country that depends on occupation like agriculture. Weather prediction has been a challenging problem in the meteorological department for years. Even after the technological and scientific advancement, the accuracy in prediction of weather has never been sufficient.

4.3.1.5 Advantages of Naive Bayes Classifier:

- Naive Bayes is one of the fast and easy Machine Learning algorithms to predict a class of datasets.
- It is effective in Multi-class predictions as compared to the other Algorithms.
- It is the most popular for text classification problems.

4.3.1.6 Disadvantages of Naive Bayes Classifier:

Naive Bayes assumes that all features are independent, it cannot learn the relationship between features.

4.3.2 K-Nearest Neighbour Algorithm

KNN which stands for K Nearest Neighbour is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending upon the features of its neighbouring data points. An example of KNN Algorithm is shown in figure 4.4, the features such as pointy ears can be used to identify cats and similarly we can identify dogs based on their long ears.

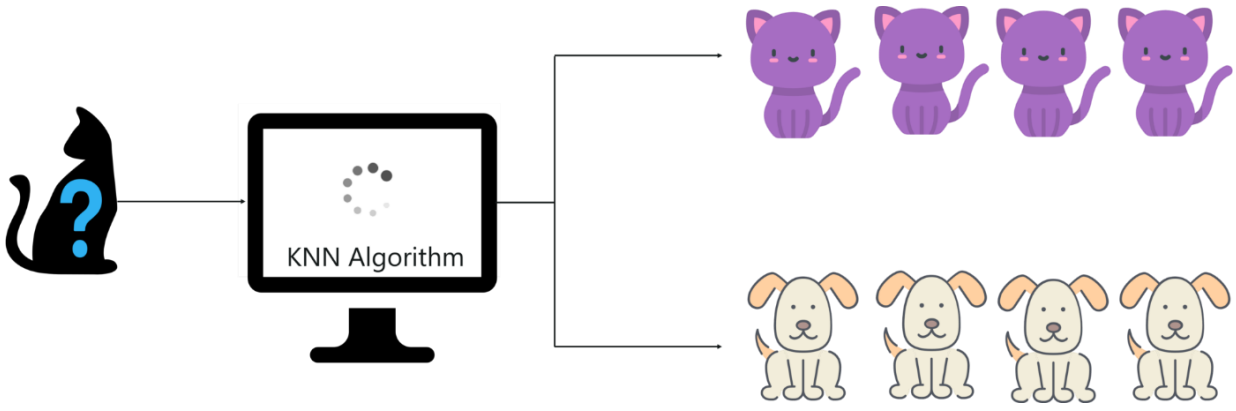


Figure 4.4 Example of KNN
[Source : https://medium.com/edureka/knn-algorithm-in-r-a2d657bca691**]**

The KNN algorithm will classify into either cats or dogs depending on the similarity of features. So if the new image has pointy ears, it will classify the image as a cat because it is similar to the cat images. The KNN algorithm classifies data points based on their similarity.

4.3.2.1 Features of KNN Algorithm

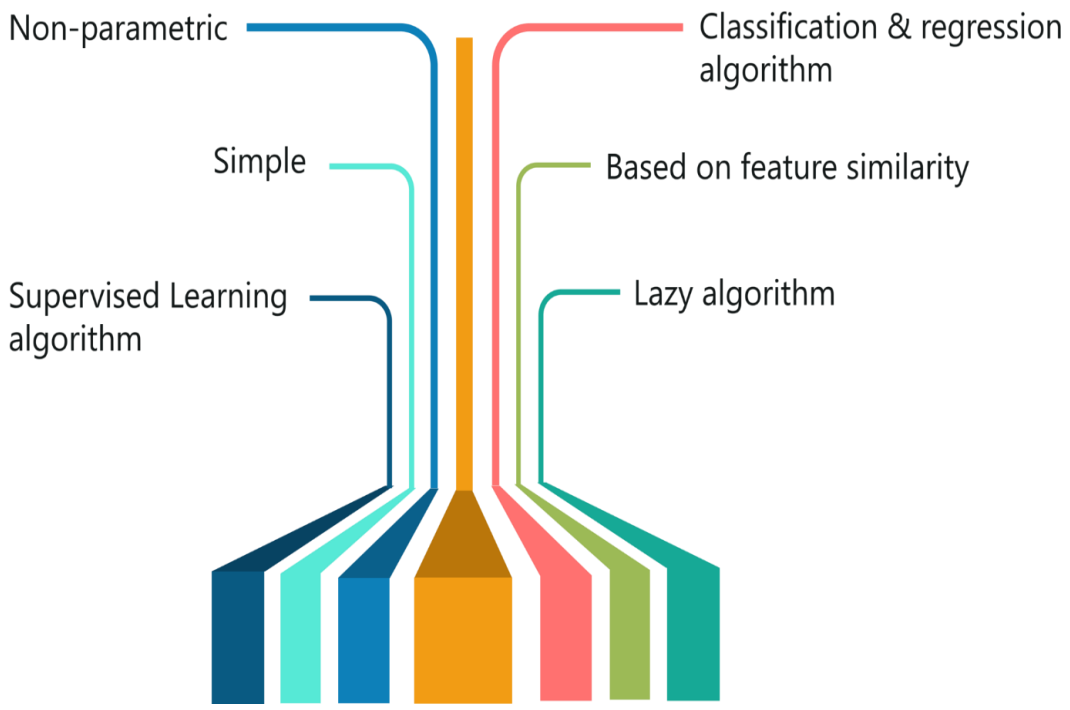


Figure 4.5 Features of KNN

[Source : https://medium.com/edureka/knn-algorithm-in-r-a2d657bca691**]**

The KNN algorithm has the following features:

KNN is a Supervised Learning algorithm that uses labelled data to predict the output.

It is one of the simple Machine learning algorithms that can be implemented for various set of problems.

It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbour and classifies the data point into the class it is most similar to.

KNN does not make any assumptions about the data set. This makes the algorithm more effective as it can handle realistic data.

KNN is a lazy algorithm, it memorizes the training data set

KNN can be used for both classification and regression problems.

4.3.2.2 K in KNN Algorithm

The k-nearest neighbour algorithm uses a very simple approach to perform classification. When tested with a new example, it looks through the training data and finds the k training examples that are closest to the new example. It then assigns the most common class label (among those k-training examples) to the test example.

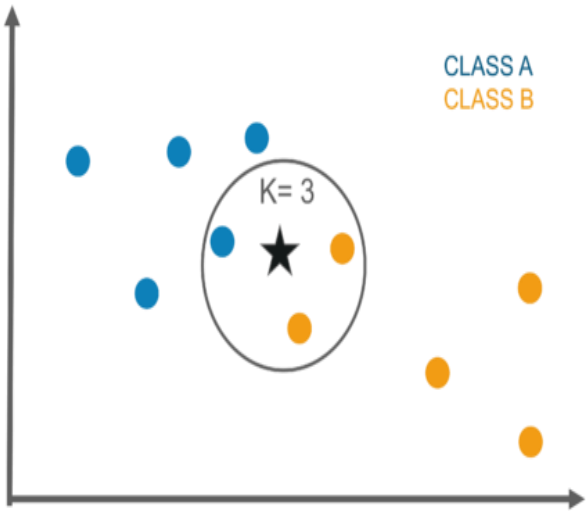


Figure 4.6 K in KNN Algorithm
[Source : https://www.edureka.co/blog/k-nearest-neighbors-algorithm/**]**

An example of k=3 is illustrated in Figure 4.6. K in KNN algorithm represents the number of nearest neighbours. If k=3, the labels of the three closest classes are checked and the most common label is assigned.

4.3.2.3 Advantages of KNN

- New data can be added seamlessly
- No Training Period
- KNN is very easy to implement

4.3.2.4 Disadvantages of KNN

- Does not work well with large dataset
- Need feature scaling
- Does not work well with high dimensions
- Sensitive to noisy data, missing values and outliers

4.3.2.5 Applications of KNN

- KNN is used for Recommendation Systems. Although in the real world, more sophisticated algorithms are used for the recommendation system. KNN is not suitable for high dimensional data, but KNN is an excellent baseline approach for the systems. Many companies make a personalized recommendation for its consumers, such as Netflix, Amazon, YouTube, and many more.
- KNN can search for semantically similar documents. Each document is considered as a vector. If documents are close to each other, that means the documents contain identical topics.
- KNN can be effectively used in detecting outliers. For example, Credit Card fraud detection.

4.3.3 DECISION TREE ALGORITHM

Decision tree is the most powerful tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. An example for decision tree algorithm is illustrated in Figure 4.7.

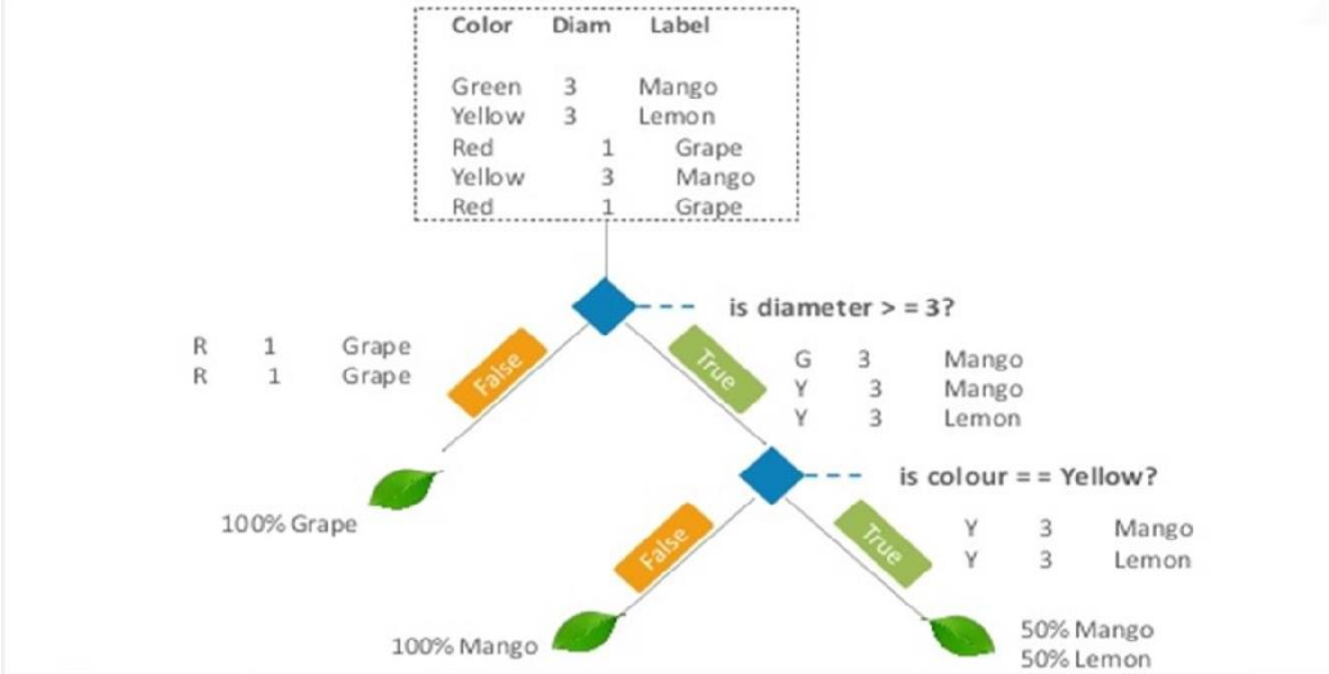


Figure 4.7 Decision tree example

[Source : https://www.slideshare.net/EdurekaIN/decision-tree-algorithm-decision-tree-in-python-machine-learning-algorithms-edureka**]**

4.3.3.1 Decision Tree Terminology:

Root node: Represents entire population

Splitting: Process of dividing sample

Decision Node: Node splits into further sub nodes

Leaf / Terminal Node: Last stage of node (output label)

Pruning: Opposite to splitting (to reduce size of tree)

4.3.3.2 Types of Decision Tree Algorithm

Iterative Dichotomiser 3(ID3) creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets.

C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-

then rules. This accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

Classification and Regression Trees (CART) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets.

4.3.3.3 Advantages of Decision Tree

- Decision trees generate understandable rules.
- Decision trees can handle both continuous and categorical data.
- Decision trees perform classification without much computation.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

4.3.3.4 Disadvantages of Decision Tree

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous variable.
- Decision trees are prone to errors in classification problems with many class and small number of training examples.
- Decision trees can be computationally expensive

4.3.4 Random Forest Algorithm

Random forest algorithm is a supervised classification and regression algorithm. It creates a forest with several trees.

Generally, the more trees in the forest the more robust the forest looks like. Similarly, in the random forest classifier, the higher the number of trees in the forest, greater is the accuracy of the results.

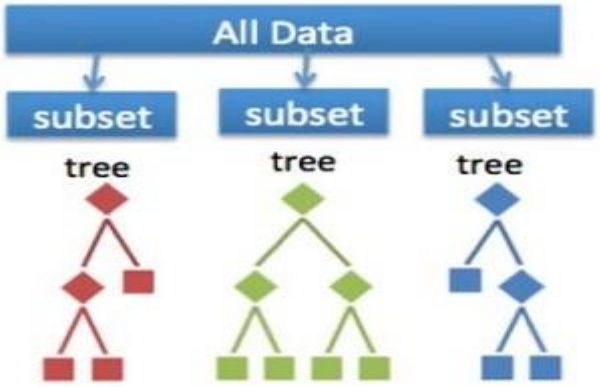


Figure 4.7 Random forest algorithm

[Source : https://www.edureka.co/blog/random-forest-classifier/**]**

Random forest builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction. The forest it builds is a collection of Decision Trees, trained with the bagging method.

4.3.4.4 Difference between Random Forest And Decision Trees

Random forest is an ensemble of decision trees, it randomly selects a set of parameters and creates a decision tree for each set of chosen parameters.

After creating multiple Decision trees using this method, each tree selects or votes the class, and the class receiving the most votes by a simple majority is termed as the predicted class.

Decision trees are built on the entire data set using all the predictor variables, whereas Random Forests are used to create multiple decision trees, such that each decision tree is built only on a part of the data set.

4.3.4.5 Uses of Random Forest

Decision trees are convenient and easily implemented, they lack accuracy. Decision trees work very effectively with the training data that was used to build them, but they're not flexible when it comes to classifying the new sample which means that the accuracy during testing phase is very low.

This happens due to a process called Over-fitting. Over-fitting occurs when a model studies the training data to such an extent that it negatively influences the performance of the model on new data.

This means that the disturbance in the training data is recorded and learned as concepts by the model. But the problem here is that these concepts do not apply to the testing data and negatively impact the model's ability to classify the new data, hence reducing the accuracy on the testing data.

4.3.4.6 Advantages Of Random Forest

- Random Forest is less prone to overfitting than Decision Tree and other algorithms
- Random Forest outputs the importance of features.

4.3.4.7 Disadvantages Of Random Forest

- Random Forest may change considerably by a small change in the data.
- Random Forest computations may go far more complex compared to other algorithms.

4.4 MODEL ALGORITHM

The overall training process using the algorithm may be summarized as below

Step1: The required libraries are imported.

Step2: The file is in CSV format, pandas read_csv method is used to read our CSV data file.

Step3: Load the data in pandas dataframe.

Step4: Replace the disease with the values in the file using inbuilt function replace in pandas.

Step5: Scikit-Learn contains the tree library, which contains built-in classes and methods for various decision tree algorithms. We use the DecisionTreeClassifier. The fit method of this classifier is called to train the algorithm on the training data. The training data is passed as the parameter to the fit method. The classifier is trained to make predictions on the test data. To make predictions, the predict method of the DecisionTreeClassifier class is used.

Step6: Import the KNeighborsClassifier class from the sklearn.neighbors library. The class is initialized with one parameter, i.e, n_neighbors. This is basically the value for the K. There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for KNN algorithm.

Step7: Initialize the Naive Bayes Classifier and fit the data. In this, the Guassian Naive Bayes Classifier is used.

Step8: Initialize the Random Forest Classifier class of the sklearn.ensemble library. The important parameter of the Random Forest Classifier class is the n_estimators parameter. This parameter defines the number of trees in the random forest. The value of the n_estimator is set to 100.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 SYSTEM FLOW

Initially the model for Naïve bayes, K-Nearest Neighbour, Decision tree and random forest is created. The model is trained, tested and optimized to give the best accuracy.

When the application is initiated, there are two modules – user module and admin module. The user has to enter his valid details and login into the system. If he is a new user the user has to signup and then login. The user has to select among the three options – Check disease, History and feedback.

The user can check the disease by entering his symptoms. The higher the symptoms he enter, higher will be the accuracy. The detected disease will be displayed on the screen. The user can check his history in the history module. The user can also give his feedback in the feedback module. In the admin module, the admin can login with his credentials. The admin can see the history and feedback of all the users. The Figure shows a simple representation of the system architecture.

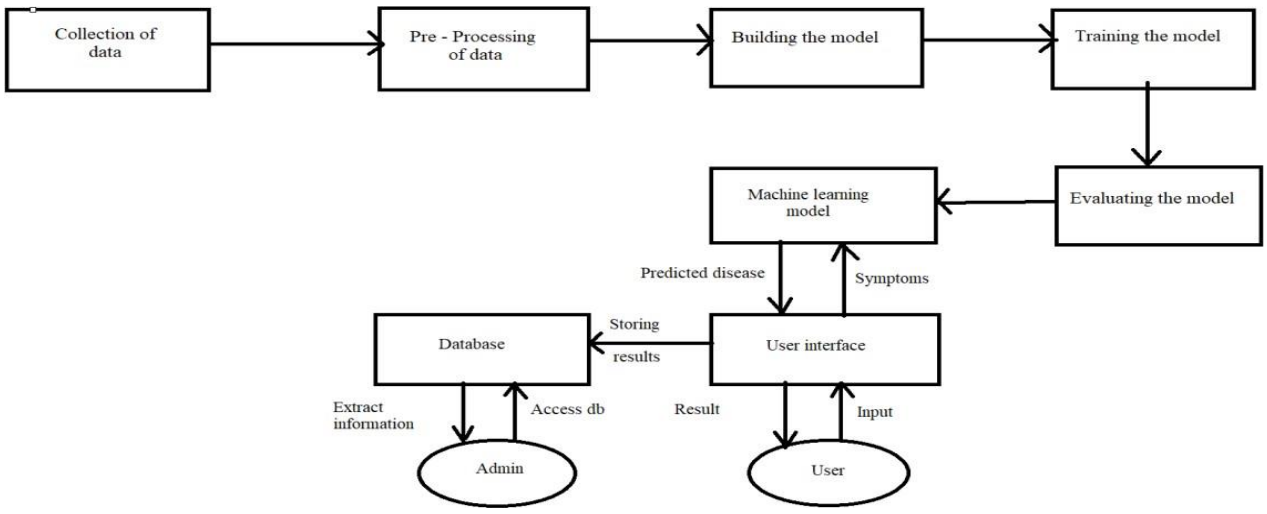


Figure 5.1 System Architecture

5.2 SETTING UP ENVIRONMENT

Python 3.6 is installed. The Pandas, NumPy, Tkinter and Sklearn, sqlite3 libraries are installed in the system.

5.3 MODEL IMPLEMENTATION

The Machine learning algorithms used for training the model are Naive Bayes, K – Nearest Neighbour, Decision tree and Random forest algorithm.

Step1: The required libraries are imported.

Step2: The file is in CSV format, pandas read_csv method is used to read our CSV data file.

Step3: Load the data in pandas dataframe.

Step4: Replace the disease with the values in the file using inbuilt function replace in pandas.

Step5: Scikit-Learn contains the tree library, which contains built-in classes and methods for various decision tree algorithms. We use the DecisionTreeClassifier**.** The fit method of this classifier is called to train the algorithm on the training data. The training data is passed as the parameter to the fit method. The classifier has been trained to make predictions on the test data. To make predictions, the predict method of the DecisionTreeClassifier class is used.

Step6: ScikitLearn's metrics library contains the classification_report and confusion_matrix which is used for evaluating the algorithm.

Step7: Import the KNeighborsClassifier class from the sklearn.neighbors library. The class is initialized with one parameter, i.e n_neighbors. This is basically the value for the K. There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for KNN algorithm.

Step8: ScikitLearn's metrics library contains the classification_report and confusion_matrix.

Step9: Initialize the Naive Bayes Classifier and fit the data. Guassian Naive Bayes Classifier is used.

Step10: Then, we evaluate its algorithm using the test set and calculate the accuracy

Step11: Initialize the Random Forest Classifier class of the sklearn.ensemble library. The important parameter of the Random Forest Classifier class is the n_estimators parameter. This parameter defines the number of trees in the random forest. The value of the n_estimator is set to 100.

Step12: The final step of solving a machine learning problem is to evaluate the performance of the algorithm. ScikitLearn's metrics library contains the classification_report and confusion_matrix which is used for evaluating the algorithm.

5.4 SYSTEM IMPLEMENTATION

In the proposed system, there is a diseases with its symptoms dataset which consists of two folders, Train and Test. These data are fed to train the model. The test data is used to evaluate the performance of the algorithm.

Apart from the datasets, there are ten python files. They are

Home_page.py – This is the first page of the system. This page contains the description of the system. When the user clicks Get Started button, the next page is initiated.

Login page.py – Here there are two modules user module and admin module. The user has to select the user module.

Login_form.py – This displays the login form for the user on the screen. If the user is the new user then he has to signup and then login the form. The form asks for email id and password which the user has to enter and login.

User_module.py – After login the user has to select between the three options check disease, history and feedback.

CheckDisease.py – If the user clicks check disease button, this page is opened. The user has to select the symptoms from the drop down menu to check the disease. The higher the symptoms he enter, higher will be the accuracy. Then he has to click the buttons to check the disease.

History.py – In this page the user can check his history.

Feedback.py – In this page the user can give his feedback.

Admin_login.py – Admin has to enter his credentials and login.

Admin_history.py – Here the admin can check the history of all the users.

Admin_feedack.py – Here the admin can check the feedback of the users.

CHAPTER 6
RESULT AND ANALYSIS

6.1 TESTING THE MACHINE LEARNING ALGORITHM

System testing presents an interesting anomaly for the software engineer. The engineer creates a series of test cases that are intended to “demolish” the software that has been built. Testing requires that the developer discards the preconceived notions of the “correctness” of software just developed and overcome a conflict of interest that occurs when errors are uncovered.

There are several rules that can serve well as testing objectives. Testing is a process of executing a program with the intent of finding an error. A good testcase is one that has a high probability of finding an as-yet undiscovered error. A successful test is one that uncovers an as-yet undiscovered error.

Initially the environment checking is done to see if all the necessary requirements are installed or not.

Installation of Python 3.6, Pandas, Numpy, Tkinter, Sklearn, sqlite3 are confirmed. The model is trained and tested with different symptoms. The results are obtained and the accuracy is checked. The model is able to predict reasonable diseases and hence the model is tested successfully.

Now the model is tested after loading it into the application’s script. On successful compilation, the loading of the model is confirmed. Now, the symptoms are tested for different diseases and the output is displayed.

6.2 RESULT ANALYSIS

The system has been tested several times, improvising the result and accuracy at each test. The training accuracy tends to be 95% and the model results coped up with the accuracy.

Disease	Count of Disease Occurrence	Symptom
UMLS:C0020538_hypertensive disease	3363	UMLS:C0008031_pain chest
		UMLS:C0392680_shortness of breath
		UMLS:C0012833_dizziness
		UMLS:C0004093_asthenia
		UMLS:C0085639_fall
		UMLS:C0039070_syncope
		UMLS:C0042571_vertigo
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0030252_palpitation
		UMLS:C0027497_nausea
		UMLS:C0002962_angina pectoris
		UMLS:C0438716_pressure chest
UMLS:C0011847_diabetes	1421	UMLS:C0032617_polyuria
		UMLS:C0085602_polydypsia
		UMLS:C0392680_shortness of breath
		UMLS:C0008031_pain chest
		UMLS:C0004093_asthenia
		UMLS:C0027497_nausea
		UMLS:C0085619_orthopnea
		UMLS:C0034642_rale
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0241526_unresponsiveness
		UMLS:C0856054_mental status changes
		UMLS:C0042571_vertigo
		UMLS:C0042963_vomiting
		UMLS:C0553668_labored breathing
UMLS:C0011570_depression mental^UMLS:C0011581_depressive disorder	1337	UMLS:C0424000_feeling suicidal
		UMLS:C0438696_suicidal

Figure 6.1 Dataset screenshot

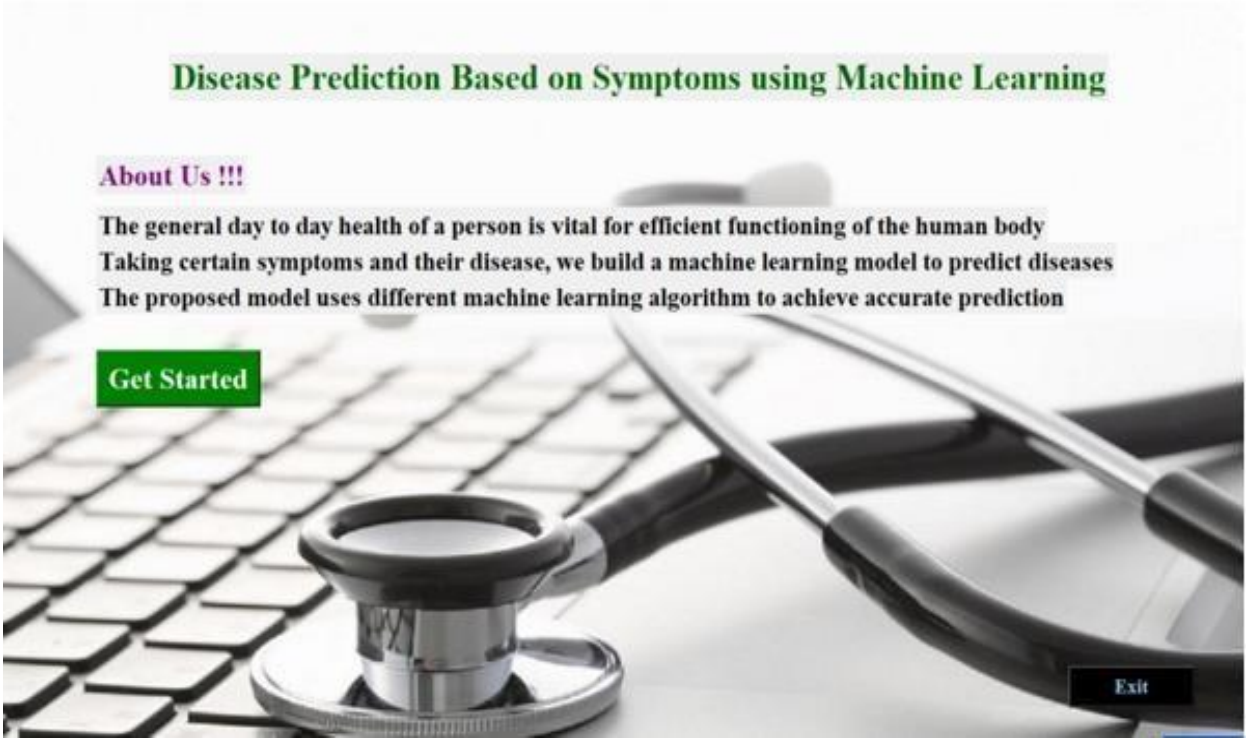


Figure 6.2 Screenshot of home page

Figure 6.3 Screenshot of user / admin module

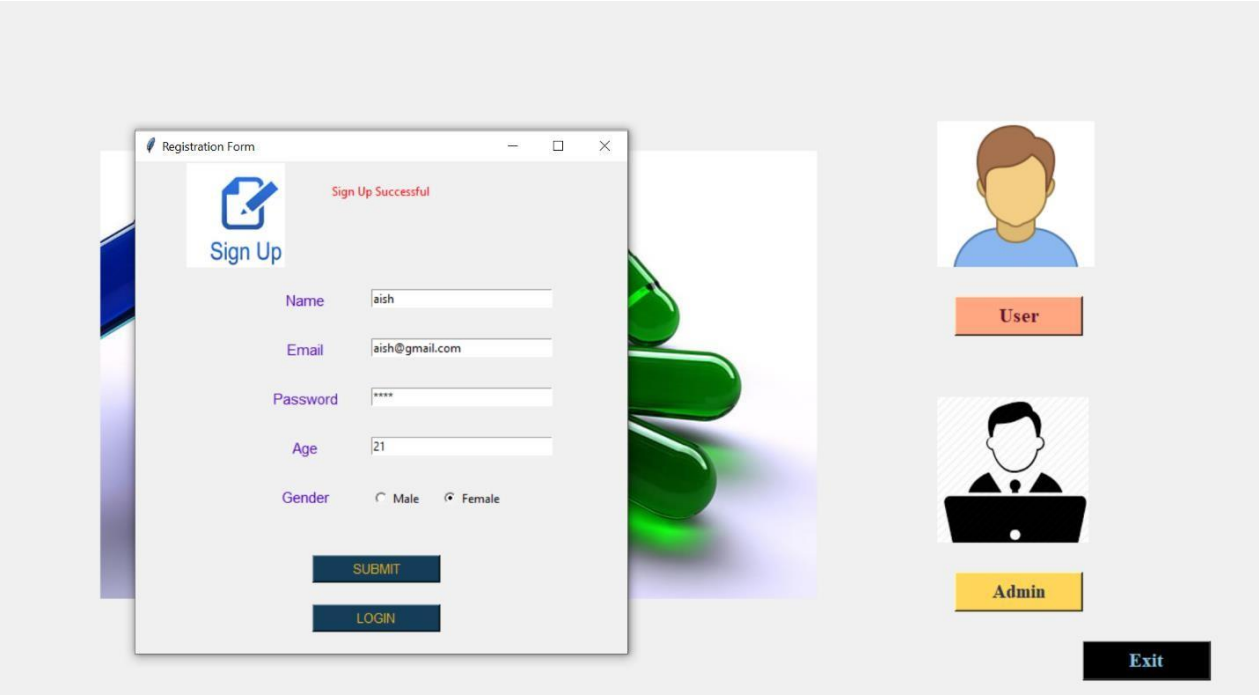


Figure 6.4 Screenshot of user signup module

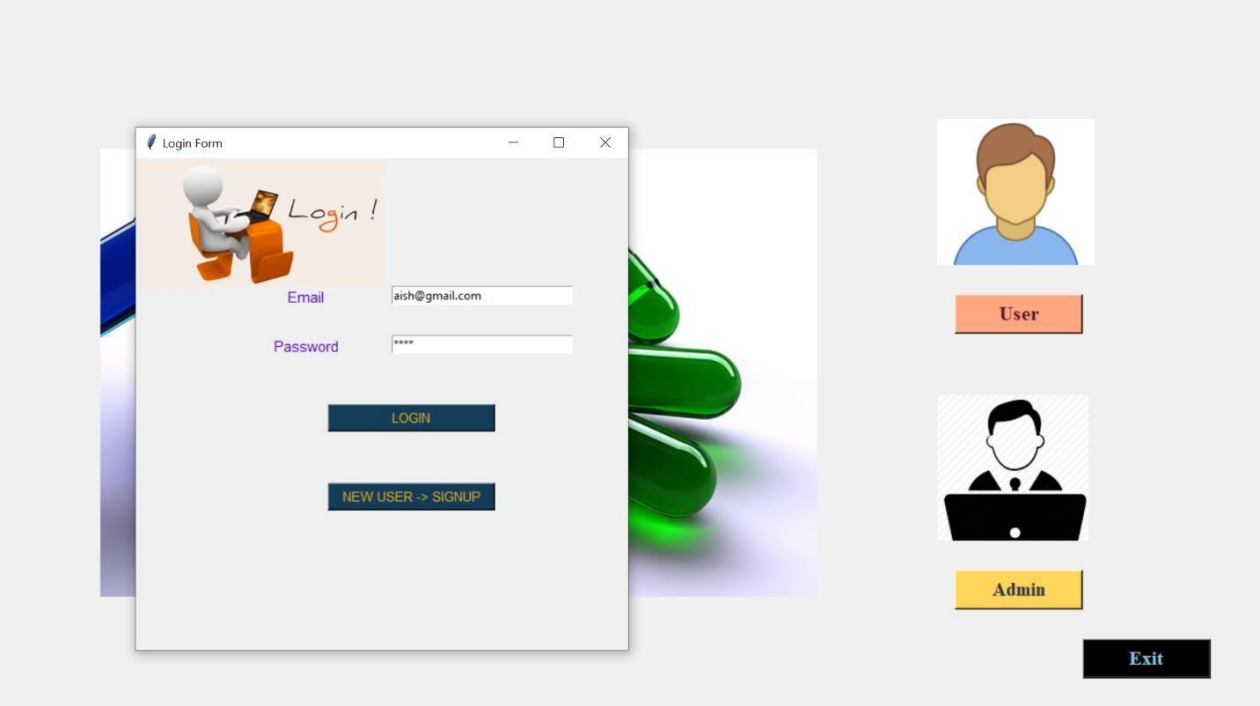


Figure 6.5 Screenshot of user login module

Figure 6.6 Screenshot of user module

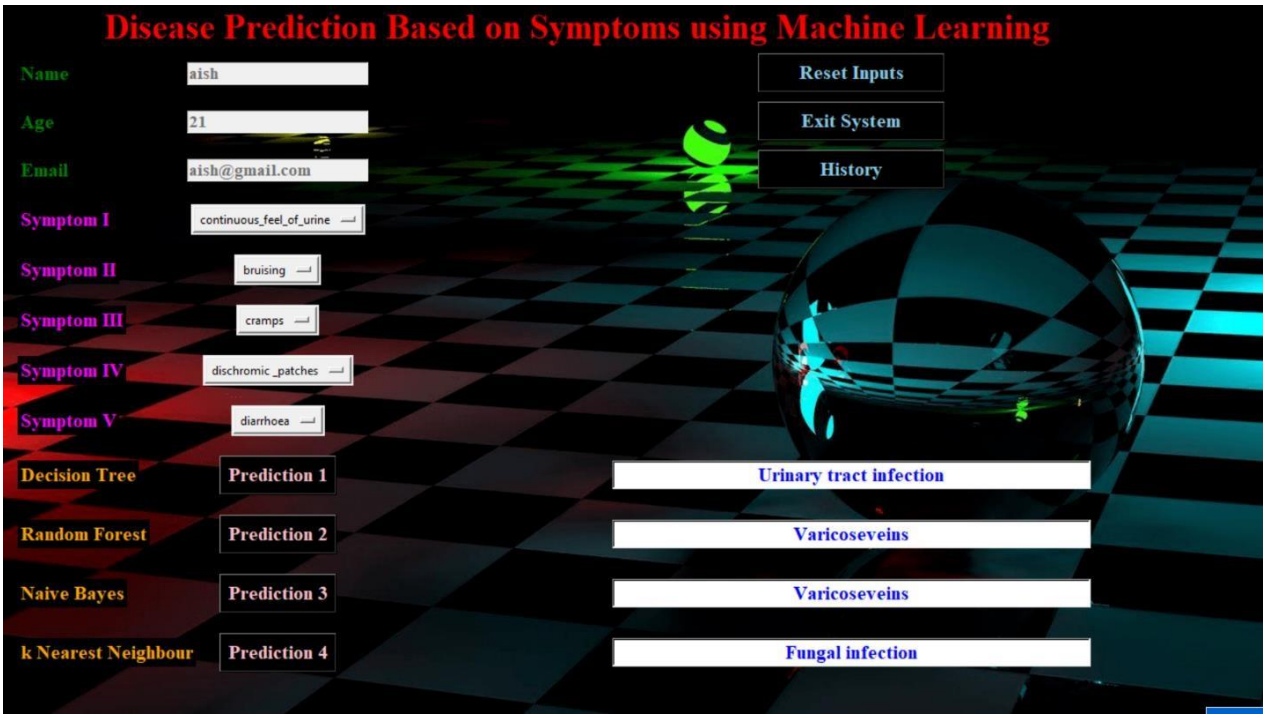


Figure 6.7 Screenshot of disease prediction page

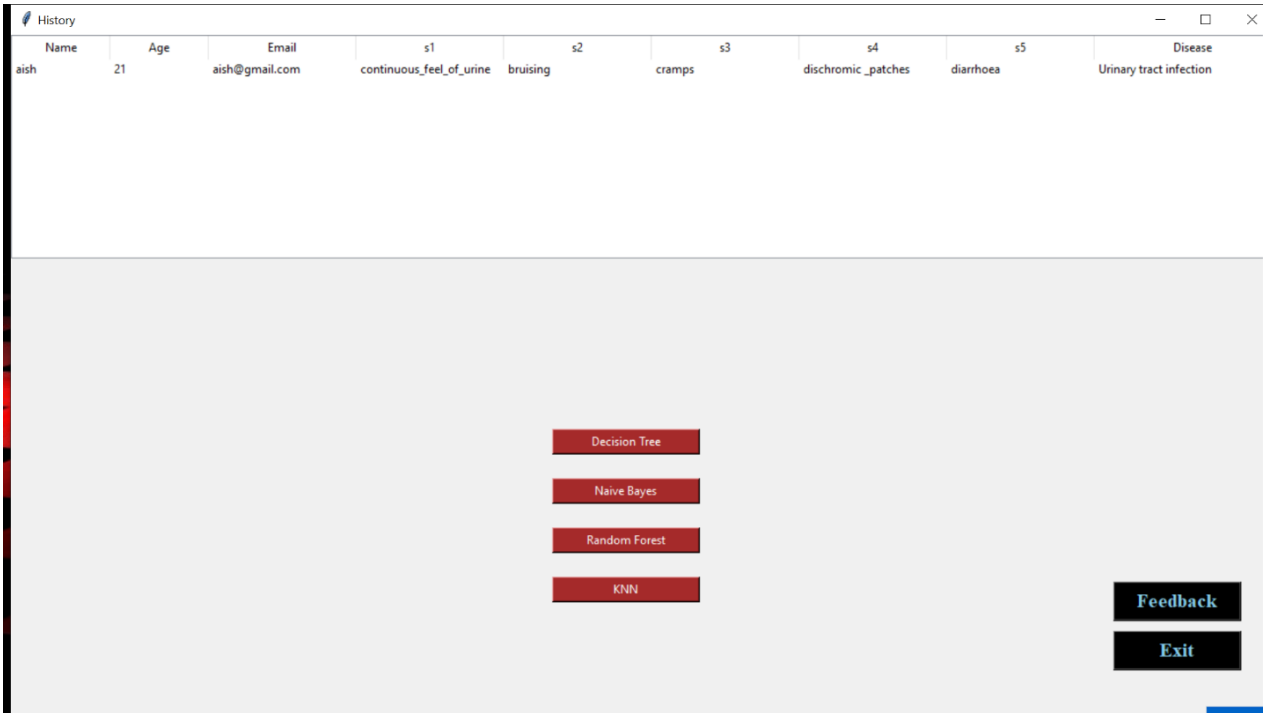


Figure 6.8 Screenshot of history page

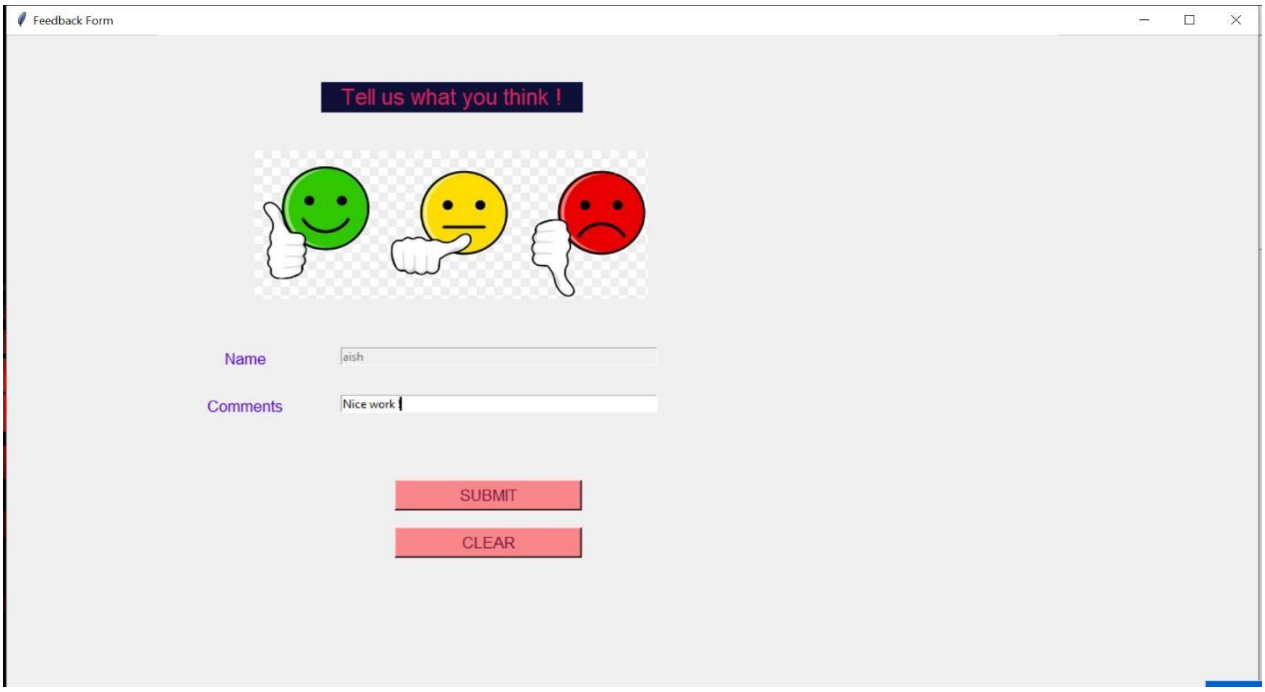


Figure 6.9 Screenshot of feedback page

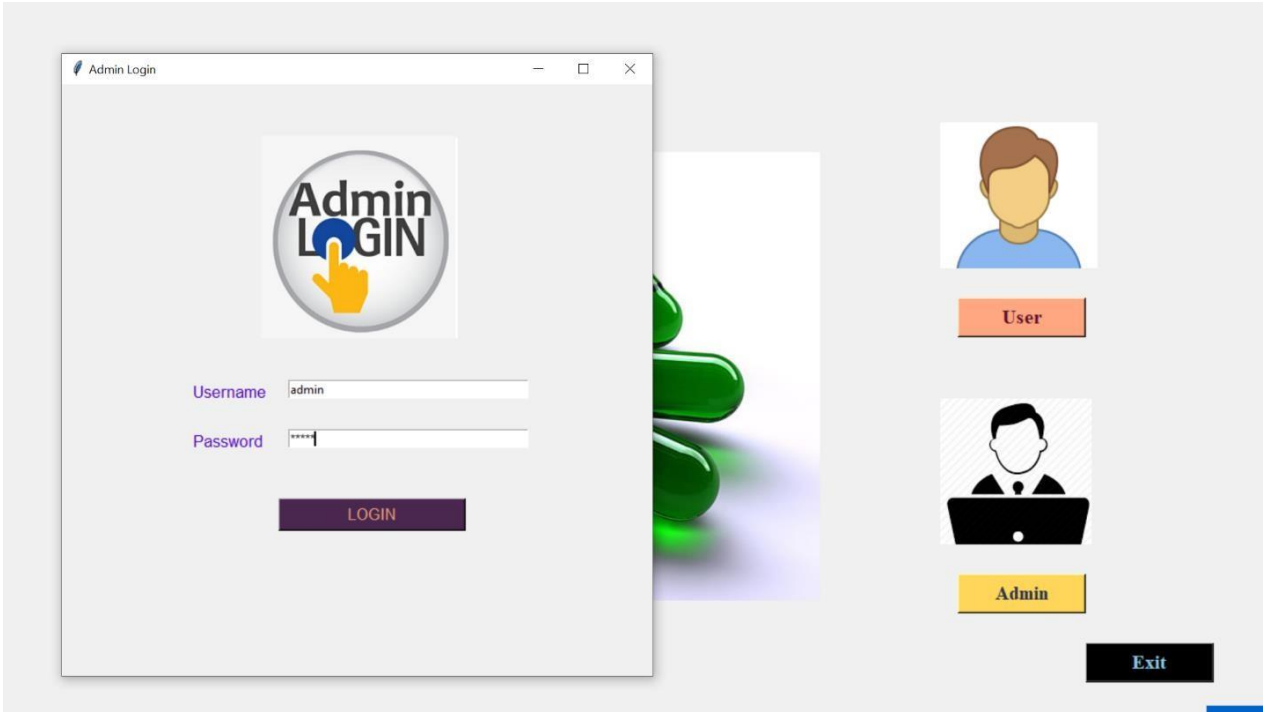


Figure 6.10 Screenshot of admin login page

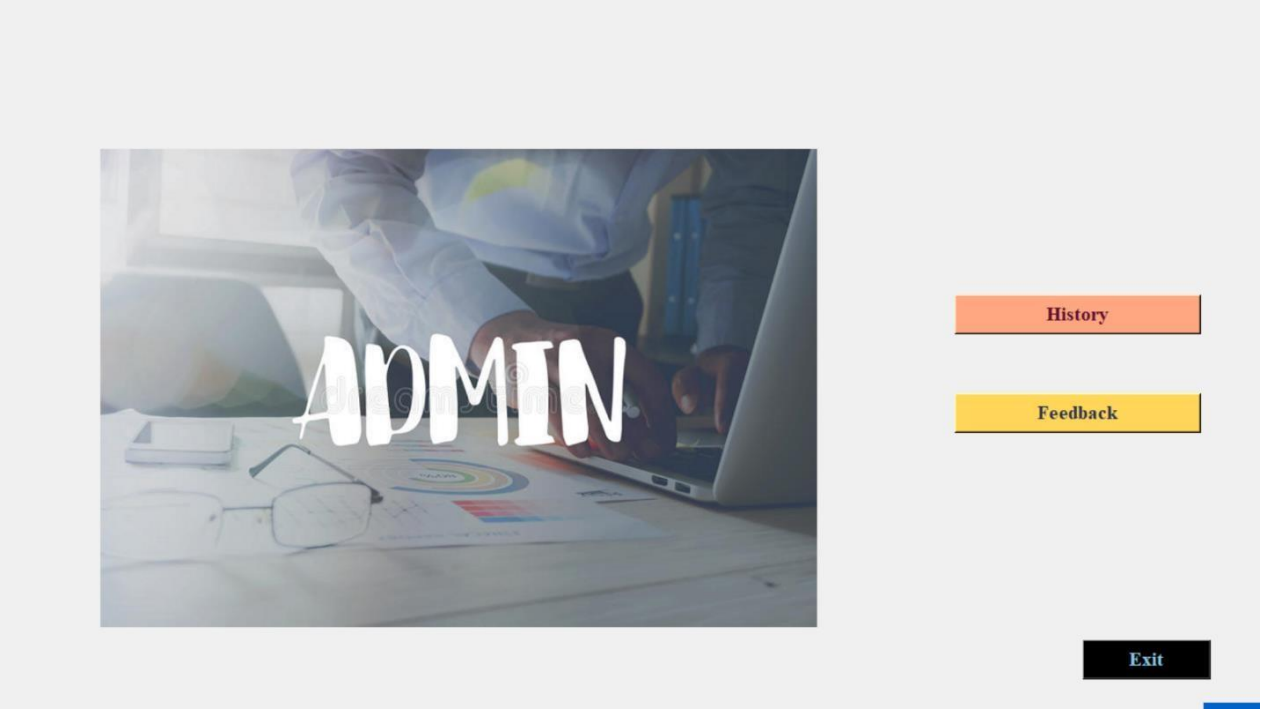


Figure 6.11 Screenshot of admin module

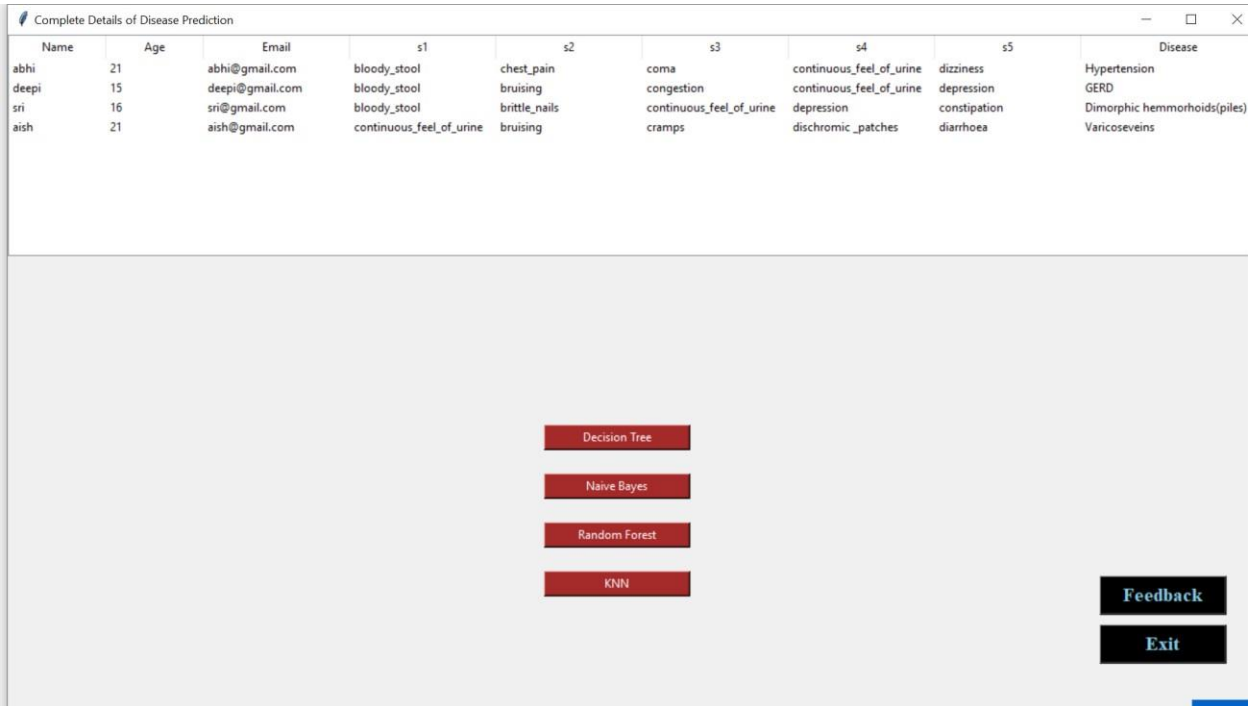


Figure 6.12 Screenshot of admin history page

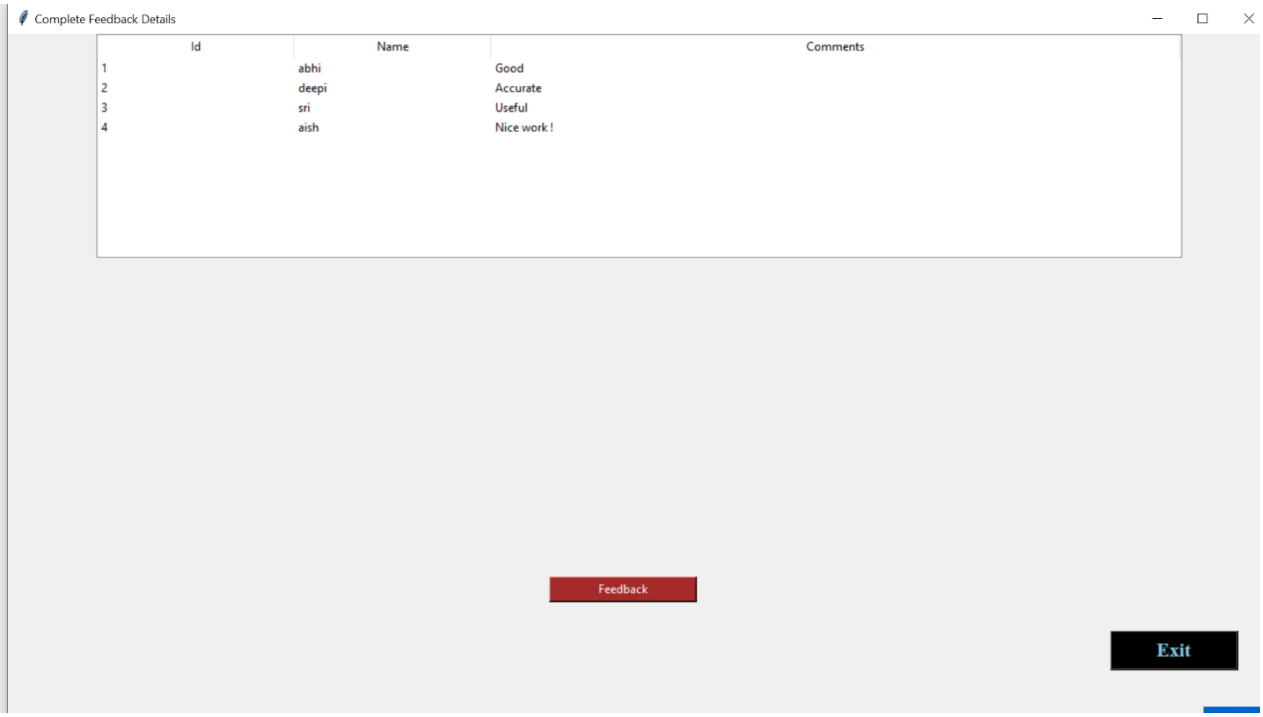


Figure 6.13 Screenshot of admin feedback page

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The project Disease prediction based on symptoms using Machine Learning is very much useful in everyone's day to day life and it is mainly more important for the healthcare sector to predict the diseases of patients based on the symptoms that they have specified. The Disease Prediction system is to provide prediction for the various and generally occurring diseases that when unchecked and sometimes ignored can turn into fatal diseases and cause a lot of problem to the patient and as well as their family members and can consult the concerned specialist.

7.2 FUTURE WORK

- The system can be trained with more number of datasets and it can further be optimised.
- The more interactive user interface can be created. The latest diseases can be updated.
- Doctors specific to the predicted disease can be referred.

APPENDIX

The code of modules are as follows:

Homepage.py

```
import tkinter
from tkinter import *
import os
root = Tk()
root.minsize(800,800)
root.title("Disease Prediction Based on Symptoms using Machine Learning")
root.configure(background='black')
root.wm_attributes("-fullscreen", True)
from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")
    if qExit:
        root.destroy()
        exit()
def login():
    import login_page
C = Canvas(root, bg="black", height=250, width=300)
filename = PhotoImage(file="C:\\Users\\Abinaya\\Desktop\\fmp1\\b.png")
background_label = Label(root, image = filename)
```

```
background_label.place(x=0, y=0, relwidth=1, relheight=1)
w2 = Label(root, justify=CENTER, text="Disease Prediction Based on Symptoms using Machine Learning", fg="darkgreen")
w2.config(font=("Times",26,"bold"))
w2.place(x=175, y=50)
w2 = Label(root, justify=CENTER, text="About Us !!!", fg="purple")
w2.config(font=("Times",22,"bold"))
w2.place(x=100, y=150)
w2 = Label(root, justify=CENTER, text="The general day to day health of a person is vital for efficient functioning of the human body", fg="Black")
w2.config(font=("Times",19,"bold"))
w2.place(x=100, y=200)

w2 = Label(root, justify=CENTER, text="Taking certain symptoms and their disease, we build a machine learning model to predict diseases",fg="Black")
w2.config(font=("Times",19,"bold"))
w2.place(x=100, y=235)
w2 = Label(root, justify=CENTER, text="The proposed model uses different machine learning algorithm to achieve accurate prediction", fg="Black")
w2.config(font=("Times",19,"bold"))
w2.place(x=100, y=270)
dst = Button(root, text="Get Started",command=login, fg="white",bg="green")
dst.config(font=("Times",22,"bold"))
dst.place(x=100, y=340)
ex = Button(root,text="Exit", command=Exit,bg="Black",fg="sky blue",width=10)
ex.config(font=("Times",15,"bold"))
ex.place(x=1100,y=650)
```

Login_page.py

```
import tkinter as tk
from tkinter import *
import PIL
root = Toplevel()
root.wm_attributes("-fullscreen", True)
from tkinter import messagebox
def Exit():
qExit=messagebox.askyesno("System","Do you want to exit the system")
if qExit:
root.destroy()
exit()
```

```
def usermodule():
   import login_form
def adminmodule():
   import admin_module
bg = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\e.png")
bg_label = Label(root, image = bg)
bg_label.place(x=100, y=150)
bg_label.image = bg
dst=Button(root,text="User",command=usermodule,bg="#FFA781",fg="#5B0E2D
",width=10)
dst.config(font=("Times",15,"bold"))
dst.place(x=970, y=300)
rnf=Button(root,text="Admin",command=adminmodule,bg="#FFD55A",fg="#293
250",
width=10)
rnf.config(font=("Times",15,"bold"))
rnf.place(x=970, y=580)
ex = Button(root,text="Exit", command=Exit,bg="Black",fg="sky blue",width=10)
ex.config(font=("Times",15,"bold"))
ex.place(x=1100,y=650)
photo = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\admin.png")
photo_label = Label(root, image = photo)
photo_label.place(x=950, y=400)
photo_label.image = photo
user = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\person-male.png")
user_label = Label(root, image = user)
user_label.place(x=950, y=120)
user_label.image = user
root.mainloop()
```

Login_form.py

```
from tkinter import *
import sqlite3
root = Toplevel()
root.geometry('500x500')
root.title("Login Form")
email=StringVar()
password=StringVar()
```

```
def database():
  global conn, cursor
  conn = sqlite3.connect('database.db')
  cursor = conn.cursor()
  email1 = email.get()
  password1 = password.get()
  if email1 == "" or password1 == "":
      lbl_text.config(text="Please complete the required field!", fg="red")
  else:
     cursor.execute("SELECT * FROM `PatientDetails` WHERE `Email` = ? AND
                                    `Password` = ?", (email1, password1))
      if cursor.fetchone() is not None:
         lbl_text.config(text="Login Successful", fg="red")
         #HomeWindow()
         email.set("")
         password.set("")
         #lbl_text.config(text="")
         f = open('email.txt', 'w')
         f.truncate(0)
         f.write(email1)
         f.close()
         import user_module
      else:
         lbl_text.config(text="Invalid username or password", fg="red")
         email.set("")
         password.set("")
  cursor.close()
  conn.close()
         def sign_up():
import sign_up_form
bg = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\login.png")
bg_label = Label(root, image = bg)
bg_label.place(x=0, y=0)
bg_label.image = bg
label_1 = Label(root, text="Email",width=20,font=("bold", 11),fg='#5300C6')
label_1.place(x=80,y=130)

entry_1 = Entry(root,textvar=email,width=30)
entry_1.place(x=260,y=130)
label_2 = Label(root, text="Password",width=20,font=("bold", 11),fg='#5300C6')
```

```
label_2.place(x=80,y=180)
entry_2 = Entry(root,textvar=password, show="*",width=30)
entry_2.place(x=260,y=180)
Button(root,text='LOGIN',width=20,font=("bold",10),bg='#143D59',fg='#F4B41A'
,com   mand=database).place(x=195,y=250)
Button(root,text='NEWUSER>SIGNUP',width=20,font=("bold",10),bg='#143D59'
,fg='#   F4B41A',command=sign_up).place(x=195,y=330)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)
lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2)
root.mainloop()
```

Signup_form.py

```
from tkinter import *
import sqlite3
root = Toplevel()
root.geometry('500x500')
root.title("Registration Form")
def login():
   import login_form
def database_s():
      name1=name.get()
      email1=email.get()
      password1=password.get()
      age1=age.get()
      gender1=gender.get()
      if(gender1 == 1):
            g = "Male"
      else:
            g = "Female"
      conn = sqlite3.connect('database.db')
      if email1 == "" or password1 == "" or name1 == "" or age1 == "" or gender1
== "":
         lbl_text.config(text="Please complete the required field!", fg="red")
      elif not age1.isnumeric():
         lbl_text.config(text="Age must be an Integer", fg="red")
      else:
         with conn:
```

```
            cursor=conn.cursor()
cursor.execute('CREATE TABLE IF NOT EXISTS PatientDetails (Name
TEXT,Email TEXT,Password TEXT,Age INTEGER,Gender TEXT)')
cursor.execute('INSERTINTOPatientDetails(Name,Email,Password,Age,Gender)
VALUES(?,?,?,?,?)',(name1,email1,password1,age1,g))
        conn.commit()
        lbl_text.config(text="Sign Up Successful", fg="red")
        login()
name=StringVar()
email=StringVar()
password=StringVar()
age=StringVar()
gender=IntVar()
g = StringVar()

bg = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\signup.png")
bg_label = Label(root, image = bg)
bg_label.place(x=50, y=0)
bg_label.image = bg

label_1 = Label(root, text="Name",width=20,font=("bold", 11),fg='#5300C6')
label_1.place(x=80,y=130)
entry_1 = Entry(root,textvar=name,width=30)
entry_1.place(x=240,y=130)
label_2 = Label(root, text="Email",width=20,font=("bold", 11),fg='#5300C6')
label_2.place(x=80,y=180)
entry_2 = Entry(root,textvar=email,width=30)
entry_2.place(x=240,y=180)
label_3 = Label(root, text="Password",width=20,font=("bold", 11),fg='#5300C6')
label_3.place(x=80,y=230)
entry_3 = Entry(root,textvar=password,show="*",width=30)
entry_3.place(x=240,y=230)
label_5 = Label(root, text="Age",width=20,font=("bold", 11),fg='#5300C6')
label_5.place(x=80,y=280)
entry_5 = Entry(root,textvar=age,width=30)
entry_5.place(x=240,y=280)
label_4 = Label(root, text="Gender",width=20,font=("bold", 11),fg='#5300C6')
label_4.place(x=80,y=330)
Radiobutton(root, text="Male",padx = 5, variable=gender,
value=1).place(x=235,y=330)
```

```
Radiobutton(root,text="Female",padx=20,variable=gender,
value=2).place(x=290,y=330)
b1=Button(root,text='SUBMIT',width=15,font=("bold",10),bg='#143D59',fg='#F4
B41A', command=database_s).place(x=180,y=400)
b1=Button(root,text='LOGIN',width=15,font=("bold",10),bg='#143D59',fg='#F4B
41A',
command=login).place(x=180,y=450)
#root.bind("<Return>",database_s)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)
lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2)
root.mainloop()
```

User_module.py

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
root = Toplevel()
root.wm_attributes("-fullscreen", True)
from tkinter import messagebox
def Exit():
   qExit=messagebox.askyesno("System","Do you want to exit the system")
   if qExit:
      root.destroy()
      exit()

def checkdisease():
   import PythonCodeOfAlgorithm
def history():
   import history
def feedback():
   import feedback
bg = PhotoImage(file = "C:\\Users\\Abinaya\\Desktop\\fmp1\\h.png")
bg_label = Label(root, image = bg)
bg_label.place(x=100, y=50)
bg_label.image = bg
dst=Button(root,text="CheckDisease",command=checkdisease,bg="#FFA781",fg=
"#5B0E2D",width=20)
dst.config(font=("Times",15,"bold"))
```

```
dst.place(x=1000, y=200)
rnf=Button(root,text="History",command=history,bg="#FFD55A",fg="#293250",
width=20)
rnf.config(font=("Times",15,"bold"))
rnf.place(x=1000, y=300)
ex=Button(root,text="Feedback",command=feedback,bg="#00E1D9",fg="#5E001
F",
width=20)
ex.config(font=("Times",15,"bold"))
ex.place(x=1000,y=400)
ex = Button(root,text="Exit", command=Exit,bg="Black",fg="sky blue",width=10)
ex.config(font=("Times",15,"bold"))
ex.place(x=1100,y=650)
```

CheckDisease.py

```
import tkinter
from tkinter import *
import sqlite3
import numpy as np
import pandas as pd
import sklearn
import os
pred1=StringVar()
def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
    root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two
Symptoms")
        if sym:
            root.mainloop()
    else:
        print("NAME: "+NameEn.get())
        from sklearn import tree
```

```
        clf3 = tree.DecisionTreeClassifier()
        clf3 = clf3.fit(X,y)
        from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
        y_pred1=clf3.predict(X_test)
        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get
()]

        for p in range(0,len(l1)):
            for x in psymptoms:
                if(x==l1[p]):
                    l2[p]=1

        input= [l2]
        predict = clf3.predict(input)
        predicted=predict[0]

        b='no'
        for i in range(0,len(disease)):
            if(predicted == i):
                b='yes'
                break

        if (b=='yes'):
            pred1.set(" ")
            pred1.set(disease[a])
        else:
            pred1.set(" ")
            pred1.set("Not Found")
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name
StringVar,Age INTEGER, Email StringVar,Symtom1 StringVar,Symtom2
StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
DecisionTree(Name,Age,Email,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5
,Disease)
```

```
VALUES(?,?,?,?,?,?,?,?,?,?)",(NameEn.get(),AgeEn.get(),EmailEn.get(),Symptom1.
get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred1.get(
)))
        conn.commit()
        c.close()
        conn.close()

pred2=StringVar()
def randomforest():
    if len(NameEn.get()) == 0:
        pred2.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred2.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two
Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.ensemble import RandomForestClassifier
        clf4 =  RandomForestClassifier(n_estimators=100)
        clf4 = clf4.fit(X,np.ravel(y))

        from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
        y_pred2=clf4.predict(X_test)
        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get
()]

        for p in range(0,len(l1)):
            for x in psymptoms:
                if(x==l1[p]):
                    l2[p]=1

        input = [l2]
        predict = clf4.predict(input)
        predicted=predict[0]
```

```
        b='no'
        for i in range(0,len(disease)):
            if(predicted == i):
                b='yes'
                break
        if (b=='yes'):
            pred2.set(" ")
            pred2.set(disease[a])
        else:
            pred2.set(" ")
            pred2.set("Not Found")
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name
StringVar,Age INTEGER, Email StringVar,Symtom1 StringVar,Symtom2
StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
RandomForest(Name,Age,Email,Symtom1,Symtom2,Symtom3,Symtom4,Symtom
5,Disease)
VALUES(?,?,?,?,?,?,?,?,?)",(NameEn.get(),AgeEn.get(),EmailEn.get(),Symptom1.
get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred2.get(
)))
        conn.commit()
        c.close()
        conn.close()

pred4=StringVar()
def KNN():
    if len(NameEn.get()) == 0:
        pred4.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred4.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two
Symptoms")
        if sym:
```

```
        root.mainloop()
    else:
        from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
        knn=knn.fit(X,np.ravel(y))

        from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
        y_pred3=knn.predict(X_test)
        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get
()]

        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        input = [l2]
        predict = clf4.predict(input)
        predicted=predict[0]

        b='no'
        for i in range(0,len(disease)):
            if(predicted == i):
                b='yes'
                break
        if (b=='yes'):
            pred4.set(" ")
            pred4.set(disease[a])
        else:
            pred4.set(" ")
            pred4.set("Not Found")
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name
StringVar,Age INTEGER, Email StringVar,Symtom1 StringVar,Symtom2
StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
```

```
c.execute("INSERTINTOKNearestNeighbour(Name,Age,Email,Symtom1,Symto
m2,Symtom3,Symtom4,Symtom5,Disease)VALUES(?,?,?,?,?,?,?,?,?)",(NameEn.g
et(),AgeEn.get(),EmailEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),
Symptom4.get(),Symptom5.get(),pred4.get()))
conn.commit()
c.close()
conn.close()

pred3=StringVar()
def NaiveBayes():
   if len(NameEn.get()) == 0:
      pred3.set(" ")
      comp=messagebox.askokcancel("System","Kindly Fill the Name")
      if comp:
         root.mainloop()
   elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
      pred3.set(" ")
      sym=messagebox.askokcancel("System","Kindly Fill atleast first two
Symptoms")
      if sym:
         root.mainloop()
   else:
      from sklearn.naive_bayes import GaussianNB
      gnb = GaussianNB()
      gnb=gnb.fit(X,np.ravel(y))

      from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
      y_pred4=gnb.predict(X_test)
      print("Naive Bayes")
      print("Accuracy : ",end="")
      print(int(accuracy_score(y_test, y_pred4)*100),end="")
      print("%")
      #print(accuracy_score(y_test, y_pred4,normalize=False))

      psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get
()]
      for k in range(0,len(l1)):
         for z in psymptoms:
```

```
            if(z==l1[k]):
                l2[k]=1

        input = [l2]
        predict = clf4.predict(input)
        predicted=predict[0]

        b='no'
        for i in range(0,len(disease)):
            if(predicted == i):
                b='yes'
                break
        if (b=='yes'):
            pred3.set(" ")
            pred3.set(disease[a])
        else:
            pred3.set(" ")
            pred3.set("Not Found")
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name
StringVar,Age INTEGER, Email StringVar,Symtom1 StringVar,Symtom2
StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
NaiveBayes(Name,Age,Email,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,
Disease)
VALUES(?,?,?,?,?,?,?,?,?,?)",(NameEn.get(),AgeEn.get(),EmailEn.get(),Symptom1.
get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred3.get(
)))
        conn.commit()
        c.close()
        conn.close()
root.configure(background='Black')
root.title('Disease Prediction Based on Symptoms using Machine Learning')
root.wm_attributes("-fullscreen", True)
Symptom1 = StringVar()
Symptom1.set("Select Here")
Symptom2 = StringVar()
```

```
Symptom2.set("Select Here")
Symptom3 = StringVar()
Symptom3.set("Select Here")
Symptom4 = StringVar()
Symptom4.set("Select Here")
Symptom5 = StringVar()
Symptom5.set("Select Here")
f = open("email.txt")
e = f.read()                #email
f.close()
con1 = sqlite3.connect("database.db")
cur1 = con1.cursor()
cur1.execute("SELECT Name FROM PatientDetails where `Email`= ?",(e,))
n = cur1.fetchone()         #name
cur1.execute("SELECT Age FROM PatientDetails where `Email`= ?",(e,))
a = cur1.fetchone()         #age
cur1.close()
con1.close()

Name = StringVar()
Name.set(n)
Age = StringVar()
Age.set(a)
Email = StringVar()
Email.set(e)

prev_win=None
def Reset():
    global prev_win
   Symptom1.set("Select Here")
   Symptom2.set("Select Here")
   Symptom3.set("Select Here")
   Symptom4.set("Select Here")
   Symptom5.set("Select Here")
   #NameEn.delete(first=0,last=100)
   #AgeEn.delete(first=0,last=100)
   #EmailEn.delete(first=0,last=100)
   pred1.set(" ")
    pred2.set(" ")
    pred3.set(" ")
```

```
    pred4.set(" ")
    try:
        prev_win.destroy()
        prev_win=None
    except AttributeError:
        pass
#Exit button to come out of system
from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")
    if qExit:
        root.destroy()
        exit()
def History():
    import history
```

History.py

```
from tkinter import ttk
import tkinter as tk
from tkinter import *
import sqlite3
from tkinter import messagebox
f = open('email.txt', 'r')
email = f.read()
f.close()
#print(email)
def dec():
    tree.delete(*tree.get_children())
    con1 = sqlite3.connect("database.db")
    cur1 = con1.cursor()
    cur1.execute("SELECT * FROM DecisionTree where `Email`= ?",(email,))
    rows = cur1.fetchall()
    for row in rows:
        #print(row)
        tree.insert("", tk.END, values=row)
    con1.close()
def naive():
    tree.delete(*tree.get_children())
    con1 = sqlite3.connect("database.db")
    cur1 = con1.cursor()
```

```
    cur1.execute("SELECT * FROM NaiveBayes where `Email`= ?",(email,))
    rows = cur1.fetchall()
    for row in rows:
        #print(row)
        tree.insert("", tk.END, values=row)
    con1.close()
def knn():
    tree.delete(*tree.get_children())
    con1 = sqlite3.connect("database.db")
    cur1 = con1.cursor()
    cur1.execute("SELECT * FROM KNearestNeighbour where `Email`=
?",(email,))
    rows = cur1.fetchall()
    for row in rows:
        #print(row)
        tree.insert("", tk.END, values=row)
    con1.close()
def random():
    tree.delete(*tree.get_children())
    con1 = sqlite3.connect("database.db")
    cur1 = con1.cursor()
    cur1.execute("SELECT * FROM RandomForest where `Email`= ?",(email,))
    rows = cur1.fetchall()
    for row in rows:
        #print(row)
        tree.insert("", tk.END, values=row)
    con1.close()

def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")
    if qExit:
        root.destroy()
        exit()
def feedback():
    import feedback
root = tk.Tk()
root.geometry("1290x800")
root.title("History")
style = ttk.Style()
style.configure("Treeview.Heading", font=("Times",15,"bold"))
```

```
tree = ttk.Treeview(root, column=("c1", "c2", "c3","c4","c5","c6","c7","c8","c9"),
show='headings')
tree.column("#1",minwidth=0, width=100, stretch="NO")
tree.heading("#1", text="Name")
tree.column("#2",minwidth=0, width=100, stretch="NO")
tree.heading("#2", text="Age")
tree.column("#3",minwidth=0, width=150, stretch="NO")
tree.heading("#3", text="Email")
tree.column("#4",minwidth=0, width=150, stretch="NO")
tree.heading("#4", text="s1")
tree.column("#5",minwidth=0, width=150, stretch="NO")
tree.heading("#5", text="s2")
tree.column("#6",minwidth=0, width=150, stretch="NO")
tree.heading("#6", text="s3")
tree.column("#7",minwidth=0, width=150, stretch="NO")
tree.heading("#7", text="s4")
tree.column("#8", minwidth=0, width=150, stretch="NO")
tree.heading("#8", text="s5")
tree.column("#9", minwidth=0, width=200, stretch="NO")
tree.heading("#9", text="Disease")
tree.pack()
tk.Button(root,text='DecisionTree',width=20,bg='brown',fg='white',command=dec)
.place(x=550,y=400)
tk.Button(root,text='NaiveBayes',width=20,bg='brown',fg='white',command=naive
).place(x=550,y=450)
tk.Button(root,text='RandomForest',width=20,bg='brown',fg='white',command=ran
dom).place(x=550,y=500)
tk.Button(root,
text='KNN',width=20,bg='brown',fg='white',command=knn).place(x=550,y=550)
tk.Button(root,text='Feedback',bg="Black",fg="Blue",font=("Times",15,"bold"),wi
dth=10,command=feedback).place(x=1120,y=555)
tk.Button(root,text='Exit',bg="Black",fg="Blue",font=("Times",15,"bold"),width=
10,command=Exit).place(x=1120,y=605)
root.mainloop()
```

Feedback.py

```
from tkinter import *
from tkinter import messagebox
import sqlite3
from tkinter import messagebox
```

```
root = Toplevel()
root.geometry('900x900')
root.title("Feedback Form")
f = open('email.txt', 'r')
email = f.read()
f.close()

def database_s():
      conn = sqlite3.connect('database.db')
      if Name == "" or Comments == "":
         lbl_text.config(text="Please complete the required field!", fg="red")
      else:
         with conn:
           cursor=conn.cursor()
         cursor.execute("CREATE TABLE IF NOT EXISTS Feedback (Id
INTEGER, Name StringVar,Comments StringVar)")
cursor.execute("INSERTINTOFeedback(Name,Comments)VALUES(?,?)",(entry_
1.get(),entry_2.get()))
conn.commit()
cursor.close()
conn.close()
messagebox.showinfo(title = "Feedback", message = "Feedback Submitted!")

def clear():
   Name.set(" ")
   Comments.set(" ")

def Exit():
   qExit=messagebox.askyesno("System","Do you want to exit the system")
   if qExit:
      root.destroy()
      exit()

f = open("email.txt")
e = f.read()              #email
f.close()
con2 = sqlite3.connect("database.db")
cur2 = con2.cursor()
cur2.execute("SELECT Name FROM PatientDetails where `Email`= ?",(e,))
```

```
n = cur2.fetchone()         #name
cur2.close()
con2.close()
print(n)


Name=StringVar()
Name.set(n)

Comments=StringVar()
bg = PhotoImage(file ="C:\\Users\\Abinaya\\Desktop\\fmp1\\feedback.png")
bg_label = Label(root, image = bg)
bg_label.place(x=250, y=120)
bg_label.image = bg

label_1 = Label(root, text="Tell us what you think !",width=20,font=("bold",
17),fg='#e52165',bg='#0d1137')
label_1.place(x=320,y=50)

label_2 = Label(root, text="Name",width=20,font=("bold", 13),fg='#5300C6',)
label_2.place(x=150,y=330)
entry_1 = Entry(root,textvariable=Name, state=DISABLED,width=53)
entry_1.place(x=340,y=330)
label_2 = Label(root, text="Comments",width=20,font=("bold", 13),fg='#5300C6')
label_2.place(x=150,y=380)
entry_2 = Entry(root,textvariable=Comments, width=53)
entry_2.place(x=340,y=380)
Button(root,text='SUBMIT',width=20,font=("bold",12),bg='#F9868B',fg='#761137
',command=database_s).place(x=395,y=470)
Button(root,text='CLEAR',width=20,font=("bold",12),bg='#F9868B',fg='#761137',
command=clear).place(x=395,y=520)
Button(root,text='Exit',bg="Black",fg="SkyBlue",font=("Times",15,"bold"),width
=10,command=Exit).place(x=1120,y=605)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)
lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2)
root.mainloop()
```

REFERENCES

1. D. Dahiwade, G. Patle and E. Meshram, "Designing Disease Prediction Model Using Machine Learning Approach," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 1211-1215, doi: 10.1109/ICCMC.2019.8819782.

2. S. Vijava Shetty, G. A. Karthik and M. Ashwin, "Symptom Based Health Prediction using Data Mining," 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, pp. 744-749, doi: 10.1109/ICCES45898.2019.9002132.

3 S. Grampurohit and C. Sagarnal, "Disease Prediction using Machine Learning Algorithms," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-7, doi:10.1109/INCET49848.2020.9154130.

4. R. Lee and C. Chitnis, "Improving Health-Care Systems by Disease Prediction," 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2018, pp. 726-731, doi: 10.1109/CSCI46756.2018.00145.

5. F. Huang, S. Wang and C. Chan, "Predicting disease by using data mining based on healthcare information system," 2012 IEEE International Conference on Granular Computing, 2012, pp. 191-194, doi: 10.1109/GrC.2012.646869

6. A. Gavhane, G. Kokkula, I. Pandya and K. Devadkar, "Prediction of Heart Disease Using Machine Learning," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018, pp. 1275-1278, doi: 10.1109/ICECA.2018.8474922.

7. M. Shankar, M. Pahadia, D. Srivastava, T. S. Ashwin and G. R. M. Reddy, "A Novel Method for Disease Recognition and Cure Time Prediction Based on Symptoms," 2015 Second International Conference on Advances in Computing and Communication Engineering, Dehradun, India, 2015, pp. 679-682, doi: 10.1109/ICACCE.2015.66.

8. M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, "Disease Prediction by Machine Learning Over Big Data From Healthcare Communities," in IEEE Access, vol. 5, pp. 8869-8879, 2017, doi: 10.1109/ACCESS.2017.2694446.

9. H. Q. Yu, "Experimental Disease Prediction Research on Combining Natural Language Processing and Machine Learning," 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2019, pp. 145-150, doi: 10.1109/ICCSNT47585.2019.8962507.

10. P. Hamsagayathri and S. Vigneshwaran, "Symptoms Based Disease Prediction Using Machine Learning Techniques," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 747-752, doi: 10.1109/ICICV50876.2021.9388603.

11. P. Zhang, X. Huang and M. Li, "Disease Prediction and Early Intervention System Based on Symptom Similarity Analysis," in IEEE Access, vol. 7, pp. 176484-176494, 2019, doi: 10.1109/ACCESS.2019.29578

12. Al-Aidaroos, K., Bakar, A., & Othman, Z. (2012). Medical Data Classification with Naive Bayes Approach. Information Technology Journal.

13. Ashish Chhabbi, LakhanAhuja, SahilAhir, and Y. K. Sharma,19 March 2016,“Heart Disease Prediction Using Data Mining Techniques”, International Journal of Research in Advent Technology,E-ISSN:2321-9637,Special Issue National Conference “NCPC-2016”, pp. 104-106.

14. H. Yin and N. K. Jha, "A Health Decision Support System for Disease Diagnosis Based on Wearable Medical Sensors and Machine Learning Ensembles," in IEEE Transactions on Multi-Scale Computing Systems, vol. 3, no. 4, pp. 228-241, 1 Oct.-Dec. 2017, doi: 10.1109/TMSCS.2017.2710194.

15. B. Nithya and V. Ilango, "Predictive analytics in health care using machine learning tools and techniques," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), 2017, pp. 492-499, doi: 10.1109/ICCONS.2017.8250771.

16. F. Rustam et al., "COVID-19 Future Forecasting Using Supervised Machine Learning Models," in IEEE Access, vol. 8, pp. 101489-101499, 2020, doi: 10.1109/ACCESS.2020.2997311.

17. M. A. Khan, "An IoT Framework for Heart Disease Prediction Based on

MDCNN Classifier," in IEEE Access, vol. 8, pp. 34717-34727, 2020, doi: 10.1109/ACCESS.2020.2974687.

18. F. Ahamed and F. Farid, "Applying Internet of Things and Machine-Learning for Personalized Healthcare: Issues and Challenges," 2018 International Conference on Machine Learning and Data Engineering (iCMLDE), 2018, pp. 19-21, doi: 10.1109/iCMLDE.2018.00014.

19. W. H. S. D. Gunarathne, K. D. M. Perera and K. A. D. C. P. Kahandawaarachchi, "Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD)," 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), 2017, pp. 291-296, doi: 10.1109/BIBE.2017.00-39.

20. K. Shailaja, B. Seetharamulu and M. A. Jabbar, "Machine Learning in Healthcare: A Review," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 910-914, doi: 10.1109/ICECA.2018.8474918.