**ABSTRACT**

There are variety of options available for cancer treatment. The type of treatment recommended for an individual is influenced by various factors such as cancer-type, the severity of cancer (stage) and most important the genetic heterogeneity. In such a complex environment, the targeted drug treatments are likely to be irresponsive or respond differently. To study anticancer drug response, we need to understand cancerous profiles. These cancerous profiles carry information which can reveal the underlying factors responsible for cancer growth. Hence, there is need to analyse cancer data for predicting optimal treatment options. Analysis of such profiles can help to predict and discover potential drug targets and drugs. In this paper the main aim is to provide machine learning based classification technique for cancerous profiles.

**1. INTRODUCTION**

The Heart is one of the most important organs in the human body. It is the center of the circulatory system. The heart functions as a pump that propels blood to different parts of the human body through a network of blood vessels, supplying a constant supply of oxygen as well as other vital nutritional components. If the heart ever stops functioning and ceases to pump blood, the body will shut down and within very less time a person will expire. The usage of information technology in the health care industry is increasing day by day to aid doctors in decision-making activities. It helps doctors and physicians in disease management, medications, and discovery of patterns and relationships among diagnosis data. Current approaches to predicting cardiovascular risk fail to identify many people who would benefit from preventive treatment, while others receive unnecessary intervention. A classic symptom of heart disease is chest pain. However, with some forms of heart disease, such as atherosclerosis, there may be no symptoms in some people until life-threatening complications develop. Any of a number of conditions that can affect the heart.

Machine-learning offers the opportunity to improve accuracy by exploiting complex interactions between risk factors. Machine learning techniques are used in the diagnosis of heart disease such as naïve Bayes, J48 decision tree, and artificial neural network, k-nearest neighbor algorithm and support vector machine showing different levels of accuracies. With the help of data sets, we predict the presence of heart disease in our body after applying different machine learning techniques on the given input attributes.

**1.1FUNCTIONAL REQUIREMENTS**

**Functional requirements**

In software engineering, a functional requirement defines a software system or its component. A function is defined as a set of inputs the behavior and outputs. The functional requirement may be calculations, technical details, data manipulation, and processing and also specify what a system is supposed to be accomplished.

INPUT: Heart Diseases dataset

OUTPUT: predictive representations of Diseases

PROCESS: pre-process data, implementing algorithms and predicting results

STORAGE: dataset data and training dataset after algorithm implementation

**1.2MODULES:**

**1.2.1 Data Collection**

* + 1. **Data pre-processing**

**1.2.3 Training and Testing Data**

* + 1. **Results and Analysis**

**1.2.1 Data Collection**

We used a dataset from kaggle.com and UCI repository for various diseases based datasets we tested. The diseases which are lower in progress, but higher in duration are termed as non-communicable diseases. There are mainly four classifications among no communicable diseases. They are respiratory diseases (asthma, pulmonary disease), cardiac diseases (heart attacks, strokes, and obesity), diabetes, and cancer. Thus, in order to predict the diseases at an early stage, machine learning algorithms are helping out to give accurate results with respect to the symptoms. Hence this study summarizes the importance of predicting no communicable diseases.

**1.2.2 Data preprocessing**

Data preparation requires approximately 80% of the time. Once data is gathered, it needs to be preprocessed, cleaned, constructed, and formatted in a style that SVM comprehends and is able to work with. DM tools should be used to analyze collected real-time data

**1.2.3Training and Testing Data**

The proposed model needs to be trained and tested under various conditions by altering SVM parameters so that correctness can be obtained. In addition, we consider that the model’s accuracy is maximum. From the collected data, 70:30 will be used to train and test the model, respectively. In case of necessity, there must be provisions to improvise on the algorithm being used.

**1.2.4Results and Analysis**

The prediction results have been evaluated using the following parameters:

**Precision:** It is the fraction of retrieved data that are useful for the query.

**Recall:** It is the fraction of data that are relevant for the query which is effectively retrieved.

**F-measure:** It is a measure that sums up precision and recall.

**Accuracy:** It is the proximity of a computation to the true value which is calculated by taking the true positive and true negative with a fraction of true positive, true negative and false positive with false negative

**1.3Non-Functional Requirements**

A non- functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design.

The major non-functional Requirements of the system are as follows.

**Usability**

The system is designed with a completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the Chosen platform. The code built by using java is more reliable.

**Performance**

This system is developing in the high-level languages and using the advanced front-end and back-end technologies it will give a response to the end user on the client system within very less time.

**Supportability**

The system is designed to be the cross-platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

**Data integrity**

Data integrity is the maintenance of and the assurance of the accuracy and consistency of, data over its entire life-cycle and is a critical aspect to the design, implementation, and usage of any system which stores, processes, or retrieves data. It is at times used as a proxy term for data quality while data validation is a pre-requisite for data integrity. Data integrity is the opposite of data corruption.

**Adaptability**

Adaptability is a feature of a system or of a process. This word has been put to use as a specialized term in different disciplines and in business operations. In ecology, adaptability has been described as the ability to cope with unexpected disturbances in the environment. our project is able to adapt in any environment simple JVM need to install.

**Accessibility**

The main goal of the project is Accessibility. We should design our project to set any device and easy to access. Our simple to access with a minimum bandwidth internet connection

**1.4 SOFTWARE REQUIREMENT SPECIFICATIONS**

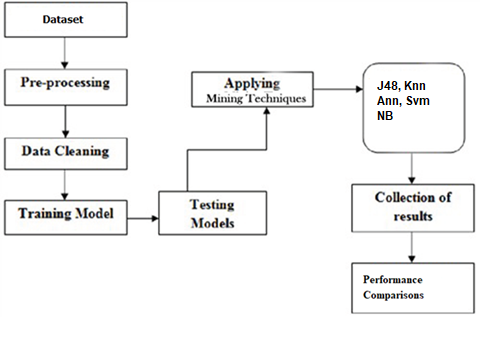
S/W System Configuration:-

Operating system: Windows 7.

Coding Language: python

**2. SYSTEM ANALYSIS**

**2.1 SYSTEM ARCITECTURE**



**2.2 FEASIBILITY ANALYSIS**

A feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, the ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

**2.2.1 TECHNICAL FEASIBILITY**

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in terms of input, output, programs, and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

* Does the existing technology sufficient for the suggested one?
* Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within the latest technology. Through the technology may become obsolete after some period of time, due to the fact that a newer version of the same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

**2.2.2 ECONOMIC FEASIBILITY**

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on the project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during the preliminary investigation:

* The costs conduct a full system investigation.
* The cost of the hardware and software.
* The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend on the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

**2.2.3 BEHAVIORAL FEASIBILITY**

This includes the following questions:

* Is there sufficient support for the users?
* Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviorally feasible.

**3. SYSTEM DESIGN**

**3.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining privacy.

Input Design considered the following things:

1. What data should be given as input?
2. How the data should be arranged or coded.
3. The dialog to guide the operating personnel in providing input.
4. Methods for preparing input validations and steps to follow when an error occurs.

**3.1.1 OBJECTIVES**

* Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
* It is achieved by creating user-friendly screens for the data entry to handle a large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
* When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as for when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**3.2 OUTPUT DESIGN**

Quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to another system through outputs. In output design, it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
* Select methods for presenting information.
* Create a document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**4. SOFTWARE DESCRIPTION**

**4.1 Unified Modelling Language Diagrams**

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

**4.1.1 User Model View**

* This view represents the system from the user’s perspective.
* The analysis representation describes a usage scenario from the end-users perspective.

**4.1.2 Structural model view**

* In this model the data and functionality are arrived from inside the system.
* This model view models the static structures
  + 1. **Behavioural Model View**
* It represents the dynamic of behavioural as parts of the system, depicting the interactions of the collection between various structural elements described in the user model and structural model view.
  + 1. **Implementation Model View**
* In this, the structural and behavioural as parts of the system are represented as they are to be built.

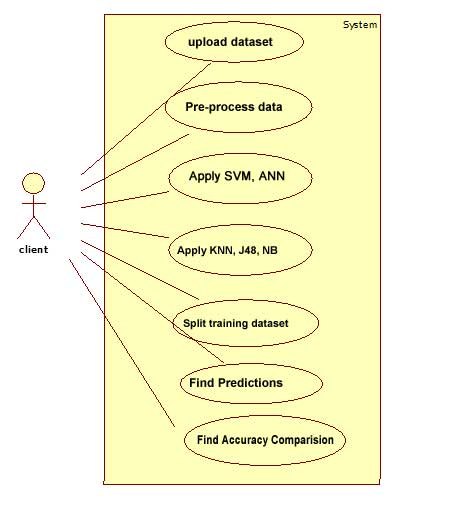
**4.1.5 Environmental Model View**

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

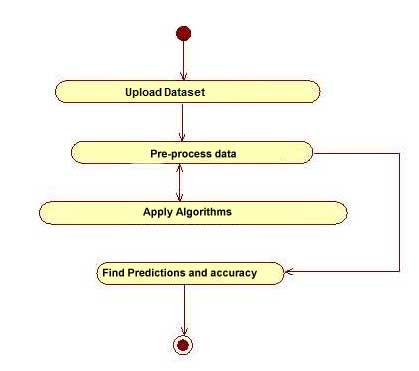
UML is specifically constructed through two different domains they are

* UML Analysis modeling, which focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on behavioral modeling, implementation modeling, and environmental model views.

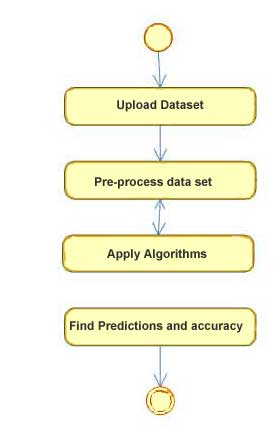
**4.2 Usecase**

****

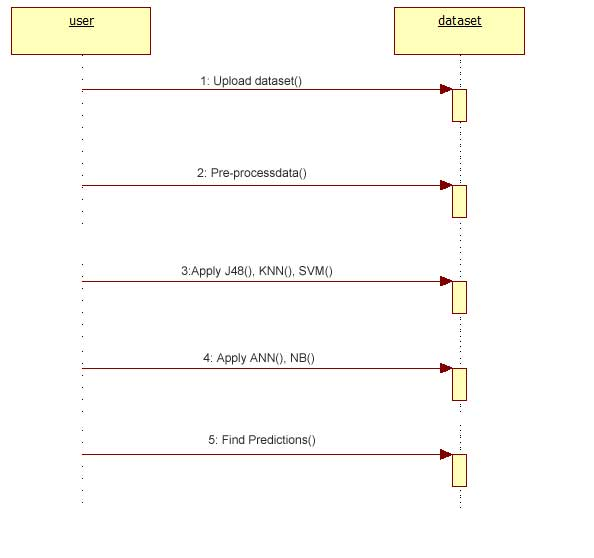
**4.3 Activity**

****

**4.4 State chart**

****

**4.5 Sequence**

****

**5. SYSTEM IMPLEMENTATION**

**5.1 Data cleaning and pre-processing**

Machine learning is the kind of programming which gives computers the capability to automatically learn from data without being explicitly programmed. This means in other words that these programs change their behavior by learning from data.

We will cover various aspects of machine learning in this tutorial. Of course, everything will be related to Python. So it is Machine Learning by using Python. It might well be that you came to this website when looking for an answer to the question: What is the best programming language for machine learning? Python is clearly one of the top players!

**5.2 Introduction to Machine Learning using Python**

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. In this article, we’ll see the basics of Machine Learning and the implementation of a simple machine learning algorithm using python.

**5.3 Machine learning overview**

Machine learning involves a computer to get trained using a given data set and use this training to predict the properties of a given new data. For example, we can train the computer by feeding it 1000 images of cats and 1000 more images which are not of a cat and tell each time to computer whether a picture is a cat or not. Then if we show the computer a new image, then from the above training, the computer should be able to tell whether this new image is a cat or not. Process of training and prediction involves the use of specialized algorithms. We feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. One such algorithm is the K-Nearest-Neighbor classification (KNN classification). It takes a test data and finds k nearest data values to this data from test data set. Then it selects the neighbor of maximum frequency and gives its properties as the prediction result

**5.4 Data Cleaning**

Missing data is always a problem in real life scenarios. Areas like machine learning and data mining face severe issues in the accuracy of their model predictions because of the poor quality of data caused by missing values. In these areas, missing value treatment is a major point of focus to make their models more accurate and valid.

**When and Why Is Data Missed?**

Let us consider an online survey for a product. Many a times, people do not share all the information related to them. Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information. Thus, in some or the other way a part of data is always missing, and this is very common in real time.

**Sources of Missing Values**

Before we dive into code, it’s important to understand the sources of missing data. Here’s some typical reasons why data is missing:

User forgot to fill in a field.

Data was lost while transferring manually from a legacy database.

There was a programming error.

Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

As you can see, some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing.

It’s important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how you deal with filling in the missing values.

Today we’ll learn how to detect missing values, and do some basic imputation. For a detailed statistical approach for dealing with missing data, check out these awesome slides from data scientist Matt Brems.

Keep in mind, imputing with a median or mean value is usually a bad idea, so be sure to check out Matt’s slides for the correct approach.

Let us now see how we can handle missing values (say NA or NaN) using Pandas.

# import the pandas library

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f', 'h'],

columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df

Its **OUTPUT** is as follows −

one two three

a 0.077988 0.476149 0.965836

b NaN NaN NaN

c -0.390208 -0.551605 -2.301950

d NaN NaN NaN

e -2.000303 -0.788201 1.510072

f -0.930230 -0.670473 1.146615

g NaN NaN NaN

h 0.085100 0.532791 0.887415

Using re-indexing, we have created a DataFrame with missing values. In the output, **NaN** means **Not a Number.**

### Check for Missing Values

To make detecting missing values easier (and across different array dtypes), Pandas provides the **isnull()** and **notnull()** functions, which are also methods on Series and DataFrame objects −

### Example

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f','h'],

columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].isnull()

Its **output** is as follows −

a False

b True

c False

d True

e False

f False

g True

h False

Name: one, dtype: bool

## **5.5 Cleaning / Filling Missing Data**

Pandas provides various methods for cleaning the missing values. The fillna function can “fill in” NA values with non-null data in a couple of ways, which we have illustrated in the following sections.

## Replace NaN with a Scalar Value

The following program shows how you can replace "NaN" with "0".

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(3, 3),

index=['a', 'c', 'e'],

columns=['one','two', 'three'])

df = df.reindex(['a', 'b', 'c'])

print df

print ("NaN replaced with '0':")

print df.fillna(0)

Its **output** is as follows −

one two three

a -0.576991 -0.741695 0.553172

b NaN NaN NaN

c 0.744328 -1.735166 1.749580

NaN replaced with '0':

one two three

a -0.576991 -0.741695 0.553172

b 0.000000 0.000000 0.000000

c 0.744328 -1.735166 1.749580

Here, we are filling with value zero; instead we can also fill with any other value.

## Fill NA Forward and Backward

Using the concepts of filling discussed in the Re-Indexing Chapter we will fill the missing values.

|  |  |
| --- | --- |
| **Method** | **Action** |
| pad/fill | Fill methods Forward |
| bfill/backfill | Fill methods Backward |

### Example

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3)),

index=['a', 'c', 'e', 'f','h'],

columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df.fillna(method='pad')

Its **output** is as follows −

one two three

a 0.077988 0.476149 0.965836

b 0.077988 0.476149 0.965836

c -0.390208 -0.551605 -2.301950

d -0.390208 -0.551605 -2.301950

e -2.000303 -0.788201 1.510072

f -0.930230 -0.670473 1.146615

g -0.930230 -0.670473 1.146615

h 0.085100 0.532791 0.887415

## Drop Missing Values

If you want to simply exclude the missing values, then use the **dropna** function along with the **axis** argument. By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded.

### Example

import pandas as pd

import numpy as np

**5.6 Data set:**

We used dataset from kaggle.com and UCI repository for various diseases based datasets we tested. The diseases which are lower in progress, but higher in duration are termed as non-communicable diseases. There are mainly four classifications among no communicable diseases. They are respiratory diseases (asthma, pulmonary disease), cardiac diseases (heart attacks, strokes, and obesity), diabetes, and cancer. Thus, in order to predict the diseases at an early stage, machine learning algorithms are helping out to give accurate results with respective to the symptoms. Hence this study summarizes the importance in predicting no communicable diseases.

**5.7 Algorithms Used**

SVM, Naïve Bayes, J48, ANN, KNN algorithms

|  |  |  |
| --- | --- | --- |
| **K Nearest Neighbours - Classification** |  |  |
| K nearest neighbours is a simple algorithm that stores all available cases and classifies new cases based on a  similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern  recognition already  in the beginning of 1970’s as a non-parametric technique. |  |  |
|  |  |  |
|  |  |  |
| A case is classified by a majority vote of its neighbours, with the case being assigned to the class  most common amongst its K nearest neighbours measured by a distance function. If K = 1, then the case is  simply assigned  to the class of its nearest neighbour. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| It should also be noted that all three distance measures are only valid for continuous variables. In the  instance of categorical variables the Hamming distance must be used. It also brings up the issue of  standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and  categorical variables in the dataset. |  |  |
| https://www.saedsayad.com/images/KNN_hamming.png |  |  |

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

**5.7.1 SUPPORT VECTOR MACHINES**

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

**SVM Algorithm**

**Algorithm: Generate SVM**

Input: Training Data, Testing Data

Output: Decision Value

**Method:**

Step 1: Load Dataset

Step 2: Classify Features (Attributes) based on class labels

Step 3: Estimate Candidate Support Value

While (instances! =null)

Do

Step 4: Support Value=Similarity between each instance in the attribute

Find Total Error Value

Step 5: If any instance < 0

Estimate

Decision value = Support Value\Total Error

Repeat for all points until it will empty

End If

**5.7.2 DECISION TREES**

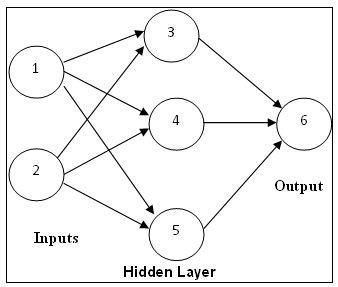
The Decision trees algorithm consists of two parts: nodes and rules (tests). We construct the tree. In which each node reflect a test on an attribute the basic idea of this algorithm is to draw a flowchart diagram that contains a root node on top. All other (non-leaf) nodes represent a test until you reach a leaf node (final result). Decision tree algorithms have been widely used in data mining applications below are some important reasons that why decision trees are used in the area of data mining and classification:

Decision trees create user-friendly rules. They are considered one of easy to understand algorithms to the end user in Data Mining. They show effective association among the dataset attributes and represent in an easy-to-understand form. Decision trees provide a clear indication of important attributes. Decision trees require less computation. They require less computation compared to other classification algorithms. When we implementing decision trees to detect breast cancer then leaf nodes are divided into two categories: Benign or Malignant. Rules will be established among the chosen data set attributes in order to determine if the tumour is benign or malignant.

**5.7.3 NAÏVE BAYES CLASSIFIER**

Naïve Bayes (NB) classifier is a probabilistic classifier based on the Bayes theorem. Rather than predictions, the Naïve Bayes classifier produces probability estimates. For each class value they estimate the probability that a given instance belongs to that class. It assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence Naïve Bayesian Model assumes that all the variables are mutually independent. Let D be the training set of tuples & their associated class labels. Each tuple is represented by N attributes such that a tuple will contain N values. Suppose there are m class labels from 1, C2… for any new tuple X, then the classifier will predict that ∈ the class having highest probability condition

**5.7.4 Multi-layer Perceptron (MLP) ANN**



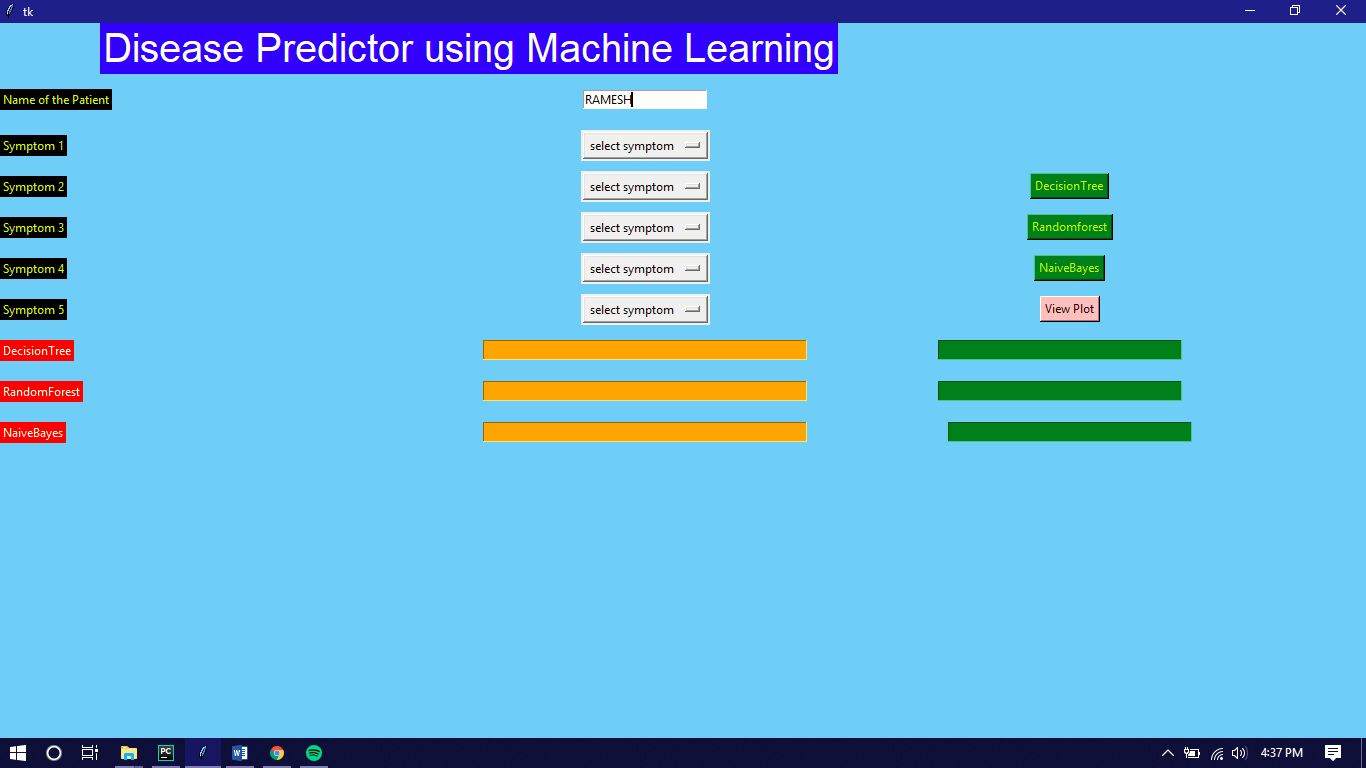
Neural networks are of particular interest because they offer a means of efficiently modeling large and complex problems in which there may be hundreds of predictor variables that have many interactions. (Actual biological neural networks are incomparably more complex.) Neural nets may used in classification problems (where the output is a categorical variable) or for regressions (where the output variable is continuous). The architecture of the neural network shown in figure.4 consists of three layers such as input layer, hidden layer and output layer. The nodes in the input layer linked with a number of nodes in the hidden layer. Each input node joined to each node in the hidden layer. The nodes in the hidden layer may connect to nodes in another hidden layer, or to an output layer. The output layer consists of one or more response variables. A main concern of the training phase is to focus on the interior weights of the neural network, which adjusted according to the transactions used in the learning process. For each training transaction, the neural network receives in addition the expected output. This concept drives us to modify the interior weights while trained neural network used to classify new images.

**Python Code:**

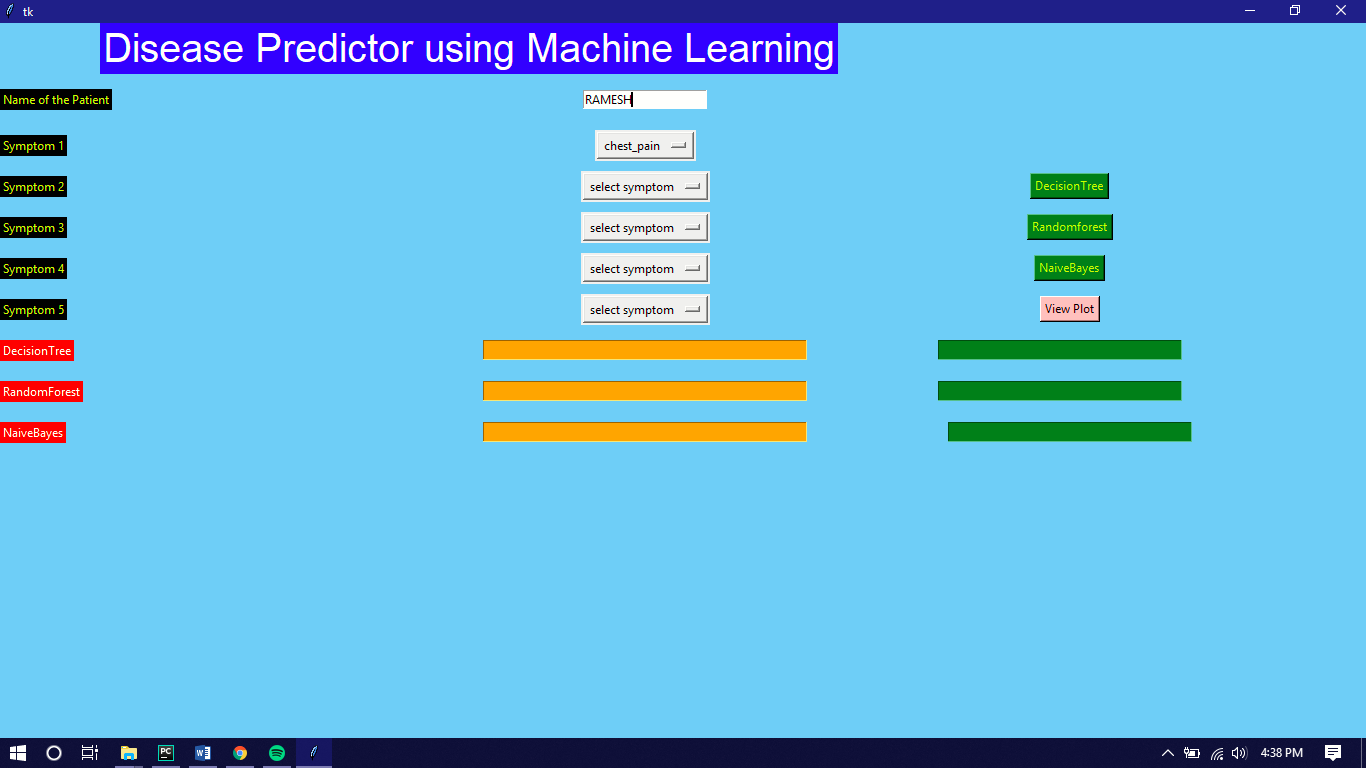
**from** tkinter **import** \*  
**import** numpy **as** np  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.metrics **import** accuracy\_score  
**from** sklearn.tree **import** DecisionTreeClassifier  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.ensemble **import** RandomForestClassifier  
**import** time  
  
**l1=['back\_pain'**,**'constipation'**,**'abdominal\_pain'**,**'diarrhoea'**,**'mild\_fever'**,**'yellow\_urine'**,**'yellowing\_of\_eyes'**,**'acute\_liver\_failure'**,**'fluid\_overload'**,**'swelling\_of\_stomach'**,**'swelled\_lymph\_nodes'**,**'malaise'**,**'blurred\_and\_distorted\_vision'**,**'phlegm'**,**'throat\_irritation'**,  
**'redness\_of\_eyes'**,**'sinus\_pressure'**,**'runny\_nose'**,**'congestion'**,**'chest\_pain'**,**'weakness\_in\_limbs'**,  
**'fast\_heart\_rate'**,**'pain\_during\_bowel\_movements'**,**'pain\_in\_anal\_region'**,**'bloody\_stool'**,  
**'irritation\_in\_anus'**,**'neck\_pain'**,**'dizziness'**,**'cramps'**,**'bruising'**,**'obesity'**,**'swollen\_legs'**,  
**'swollen\_blood\_vessels'**,**'puffy\_face\_and\_eyes'**,**'enlarged\_thyroid'**,**'brittle\_nails'**,  
**'swollen\_extremeties'**,**'excessive\_hunger'**,**'extra\_marital\_contacts'**,**'drying\_and\_tingling\_lips'**,  
**'slurred\_speech'**,**'knee\_pain'**,**'hip\_joint\_pain'**,**'muscle\_weakness'**,**'stiff\_neck'**,**'swelling\_joints'**,  
**'movement\_stiffness'**,**'spinning\_movements'**,**'loss\_of\_balance'**,**'unsteadiness'**,  
**'weakness\_of\_one\_body\_side'**,**'loss\_of\_smell'**,**'bladder\_discomfort'**,**'foul\_smell\_of urine'**,  
**'continuous\_feel\_of\_urine'**,**'passage\_of\_gases'**,**'internal\_itching'**,**'toxic\_look\_(typhos)'**,  
**'depression'**,**'irritability'**,**'muscle\_pain'**,**'altered\_sensorium'**,**'red\_spots\_over\_body'**,**'belly\_pain'**,  
**'abnormal\_menstruation'**,**'dischromic \_patches'**,**'watering\_from\_eyes'**,**'increased\_appetite'**,**'polyuria'**,**'family\_history'**,**'mucoid\_sputum'**,  
**'rusty\_sputum'**,**'lack\_of\_concentration'**,**'visual\_disturbances'**,**'receiving\_blood\_transfusion'**,  
**'receiving\_unsterile\_injections'**,**'coma'**,**'stomach\_bleeding'**,**'distention\_of\_abdomen'**,  
**'history\_of\_alcohol\_consumption'**,**'fluid\_overload'**,**'blood\_in\_sputum'**,**'prominent\_veins\_on\_calf'**,  
**'palpitations'**,**'painful\_walking'**,**'pus\_filled\_pimples'**,**'blackheads'**,**'scurring'**,**'skin\_peeling'**,  
**'silver\_like\_dusting'**,**'small\_dents\_in\_nails'**,**'inflammatory\_nails'**,**'blister'**,**'red\_sore\_around\_nose'**,**'yellow\_crust\_ooze']**  
  
**disease=['Fungal infection'**,**'Allergy'**,**'GERD'**,**'Chronic cholestasis'**,**'Drug Reaction'**,  
**'Peptic ulcer diseae'**,**'AIDS'**,**'Diabetes'**,**'Gastroenteritis'**,**'Bronchial Asthma'**,**'Hypertension'**,  
**' Migraine'**,**'Cervical spondylosis'**,  
**'Paralysis (brain hemorrhage)'**,**'Jaundice'**,**'Malaria'**,**'Chicken pox'**,**'Dengue'**,**'Typhoid'**,**'hepatitis A'**,  
**'Hepatitis B'**,**'Hepatitis C'**,**'Hepatitis D'**,**'Hepatitis E'**,**'Alcoholic hepatitis'**,**'Tuberculosis'**,  
**'Common Cold'**,**'Pneumonia'**,**'Dimorphic hemmorhoids(piles)'**,  
**'Heartattack'**,**'Varicoseveins'**,**'Hypothyroidism'**,**'Hyperthyroidism'**,**'Hypoglycemia'**,**'Osteoarthristis'**,  
**'Arthritis'**,**'(vertigo) Paroymsal Positional Vertigo'**,**'Acne'**,**'Urinary tract infection'**,**'Psoriasis'**,  
**'Impetigo']**  
  
l2=[]  
**for** x **in** range(0,len(l1)):  
 l2.append(0)  
  
  
df=pd.read\_csv(**"Training.csv"**)  
  
df.replace({**'prognosis'**:{**'Fungal infection'**:0,**'Allergy'**:1,**'GERD'**:2,**'Chronic cholestasis'**:3,**'Drug Reaction'**:4,  
**'Peptic ulcer diseae'**:5,**'AIDS'**:6,**'Diabetes '**:7,**'Gastroenteritis'**:8,**'Bronchial Asthma'**:9,**'Hypertension '**:10,  
**'Migraine'**:11,**'Cervical spondylosis'**:12,  
**'Paralysis (brain hemorrhage)'**:13,**'Jaundice'**:14,**'Malaria'**:15,**'Chicken pox'**:16,**'Dengue'**:17,**'Typhoid'**:18,**'hepatitis A'**:19,  
**'Hepatitis B'**:20,**'Hepatitis C'**:21,**'Hepatitis D'**:22,**'Hepatitis E'**:23,**'Alcoholic hepatitis'**:24,**'Tuberculosis'**:25,  
**'Common Cold'**:26,**'Pneumonia'**:27,**'Dimorphic hemmorhoids(piles)'**:28,**'Heart attack'**:29,**'Varicose veins'**:30,**'Hypothyroidism'**:31,  
**'Hyperthyroidism'**:32,**'Hypoglycemia'**:33,**'Osteoarthristis'**:34,**'Arthritis'**:35,  
**'(vertigo) Paroymsal Positional Vertigo'**:36,**'Acne'**:37,**'Urinary tract infection'**:38,**'Psoriasis'**:39,  
**'Impetigo'**:40}},inplace=**True**)  
  
X= df[l1]  
bna=0  
y = df[[**"prognosis"**]]  
tr=pd.read\_csv(**"Testing.csv"**)  
tr.replace({**'prognosis'**:{**'Fungal infection'**:0,**'Allergy'**:1,**'GERD'**:2,**'Chronic cholestasis'**:3,**'Drug Reaction'**:4,  
**'Peptic ulcer diseae'**:5,**'AIDS'**:6,**'Diabetes '**:7,**'Gastroenteritis'**:8,**'Bronchial Asthma'**:9,**'Hypertension '**:10,  
**'Migraine'**:11,**'Cervical spondylosis'**:12,  
**'Paralysis (brain hemorrhage)'**:13,**'Jaundice'**:14,**'Malaria'**:15,**'Chicken pox'**:16,**'Dengue'**:17,**'Typhoid'**:18,**'hepatitis A'**:19,  
**'Hepatitis B'**:20,**'Hepatitis C'**:21,**'Hepatitis D'**:22,**'Hepatitis E'**:23,**'Alcoholic hepatitis'**:24,**'Tuberculosis'**:25,  
**'Common Cold'**:26,**'Pneumonia'**:27,**'Dimorphic hemmorhoids(piles)'**:28,**'Heart attack'**:29,**'Varicose veins'**:30,**'Hypothyroidism'**:31,  
**'Hyperthyroidism'**:32,**'Hypoglycemia'**:33,**'Osteoarthristis'**:34,**'Arthritis'**:35,  
**'(vertigo) Paroymsal Positional Vertigo'**:36,**'Acne'**:37,**'Urinary tract infection'**:38,**'Psoriasis'**:39,  
**'Impetigo'**:40}},inplace=**True**)  
nba=0  
X\_test= tr[l1]  
y\_test = tr[[**"prognosis"**]]  
np.ravel(y\_test)  
**def** DecisionTree():  
  
 **from** sklearn **import** tree  
  
 clf3 = tree.DecisionTreeClassifier()  
 clf3 = clf3.fit(X,y)  
  
  
 **from** sklearn.metrics **import** accuracy\_score  
 start = time.time()  
 y\_pred=clf3.predict(X\_test)  
 end = time.time()  
 *#print( "Run Time: %f" % (end-start))* nba=accuracy\_score(y\_test, y\_pred)  
 end=(end-start)  
 nba=nba+end  
 nba=(nba)\*100  
 print(nba)  
   
  
 psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]  
  
 **for** k **in** range(0,len(l1)):  
 *# print (k,)* **for** z **in** psymptoms:  
 **if**(z==l1[k]):  
 l2[k]=1  
  
 inputtest = [l2]  
 predict = clf3.predict(inputtest)  
   
 predicted=predict[0]  
  
  
  
  
 h=**'no'  
 for** a **in** range(0,len(disease)):  
 **if**(predicted == a):  
 h=**'yes'  
 break  
  
  
 if** (h==**'yes'**):  
 t1.delete(**"1.0"**, END)  
 t1.insert(END, disease[a])  
 ranfLb1.delete(**"1.0"**, END)  
 ranfLb1.insert(END,nba)  
 **else**:  
 t1.delete(**"1.0"**, END)  
 t1.insert(END, **"Not Found"**)  
  
  
**def** randomforest():  
 **from** sklearn.ensemble **import** RandomForestClassifier  
 clf4 = RandomForestClassifier()  
 clf4 = clf4.fit(X,np.ravel(y))  
  
  
 **from** sklearn.metrics **import** accuracy\_score  
 start = time.time()  
 y\_pred=clf4.predict(X\_test)  
 end = time.time()  
 *#print( "Run Time: %f" % (end-start))* nba=accuracy\_score(y\_test, y\_pred)  
 end=(end-start)  
 nba=nba+end  
 nba=(nba)\*100  
 print(nba)  
 *#aaprint(accuracy\_score(y\_test, y\_pred,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  
  
 inputtest = [l2]  
 predict = clf4.predict(inputtest)  
 predicted=predict[0]  
  
 h=**'no'  
 for** a **in** range(0,len(disease)):  
 **if**(predicted == a):  
 h=**'yes'  
 break  
  
 if** (h==**'yes'**):  
 t2.delete(**"1.0"**, END)  
 t2.insert(END, disease[a])  
 ranfLb2.delete(**"1.0"**, END)  
 ranfLb2.insert(END,nba)  
 **else**:  
 t2.delete(**"1.0"**, END)  
 t2.insert(END, **"Not Found"**)  
  
  
**def** NaiveBayes():  
 **from** sklearn.naive\_bayes **import** GaussianNB  
 gnb = GaussianNB()  
 gnb=gnb.fit(X,np.ravel(y))  
  
  
 **from** sklearn.metrics **import** accuracy\_score  
 y\_pred=gnb.predict(X\_test)  
 start = time.time()  
 y\_pred=gnb.predict(X\_test)  
 end = time.time()  
 *#print( "Run Time: %f" % (end-start))* nba=accuracy\_score(y\_test, y\_pred)  
 end=(end-start)  
 nba=nba+end  
 nba=(nba)\*100  
 print(nba)  
   
 *# -----------------------------------------------------* 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  
  
 inputtest = [l2]  
 predict = gnb.predict(inputtest)  
 predicted=predict[0]  
  
 h=**'no'  
 for** a **in** range(0,len(disease)):  
 **if**(predicted == a):  
 h=**'yes'  
 break  
  
 if** (h==**'yes'**):  
 t3.delete(**"1.0"**, END)  
 t3.insert(END, disease[a])  
 ranfLb3.delete(**"1.0"**, END)  
 ranfLb3.insert(END, nba)  
 **else**:  
 t3.delete(**"1.0"**, END)  
 t3.insert(END, **"Not Found"**)  
  
  
**def** grap():  
 results ={**'Accuracy'**: [95.355,95.123,96.12]}  
 index = [**'Decision Tree'**,**'RandomForest'**,**'NB'**]  
 results=pd.DataFrame(results,index=index)  
 fig =results.plot(kind=**'bar'**,title=**'Comaprison of models'**,figsize =(6,6),ylim=[40,100]).get\_figure()  
   
   
root = Tk()  
root.configure(background=**'skyblue'**)  
  
  
Symptom1 = StringVar()  
Symptom1.set(**'select symptom'**)  
Symptom2 = StringVar()  
Symptom2.set(**'select symptom'**)  
Symptom3 = StringVar()  
Symptom3.set(**'select symptom'**)  
Symptom4 = StringVar()  
Symptom4.set(**'select symptom'**)  
Symptom5 = StringVar()  
Symptom5.set(**'select symptom'**)  
Name = StringVar()  
  
  
w2 = Label(root, justify=LEFT, text=**"Disease Predictor using Machine Learning"**, fg=**"white"**, bg=**"blue"**)  
w2.config(font=(**"Elephant"**, 30))  
w2.grid(row=1, column=0, columnspan=2, padx=100)  
  
w2.config(font=(**"Aharoni"**, 30))  
w2.grid(row=2, column=0, columnspan=2, padx=100)  
  
  
NameLb = Label(root, text=**"Name of the Patient"**, fg=**"yellow"**, bg=**"black"**)  
NameLb.grid(row=6, column=0, pady=15, sticky=W)  
  
  
S1Lb = Label(root, text=**"Symptom 1"**, fg=**"yellow"**, bg=**"black"**)  
S1Lb.grid(row=7, column=0, pady=10, sticky=W)  
  
S2Lb = Label(root, text=**"Symptom 2"**, fg=**"yellow"**, bg=**"black"**)  
S2Lb.grid(row=8, column=0, pady=10, sticky=W)  
  
S3Lb = Label(root, text=**"Symptom 3"**, fg=**"yellow"**, bg=**"black"**)  
S3Lb.grid(row=9, column=0, pady=10, sticky=W)  
  
S4Lb = Label(root, text=**"Symptom 4"**, fg=**"yellow"**, bg=**"black"**)  
S4Lb.grid(row=10, column=0, pady=10, sticky=W)  
  
S5Lb = Label(root, text=**"Symptom 5"**, fg=**"yellow"**, bg=**"black"**)  
S5Lb.grid(row=11, column=0, pady=10, sticky=W)  
  
  
lrLb = Label(root, text=**"DecisionTree"**, fg=**"white"**, bg=**"red"**)  
lrLb.grid(row=15, column=0, pady=10,sticky=W)  
  
destreeLb = Label(root, text=**"RandomForest"**, fg=**"white"**, bg=**"red"**)  
destreeLb.grid(row=17, column=0, pady=10, sticky=W)  
  
ranfLb = Label(root, text=**"NaiveBayes"**, fg=**"white"**, bg=**"red"**)  
ranfLb.grid(row=19, column=0, pady=10, sticky=W)  
  
ranfLb1 =Text(root, height=1, width=30, fg=**"white"**, bg=**"green"**)  
ranfLb1.grid(row=15, column=3, pady=10, sticky=W)  
  
ranfLb2 =Text(root, height=1, width=30, fg=**"white"**, bg=**"green"**)  
ranfLb2.grid(row=17, column=3, pady=10, sticky=W)  
  
ranfLb3 = Text(root, height=1, width=30, fg=**"white"**, bg=**"green"**)  
ranfLb3.grid(row=19, column=3, padx=10)  
   
   
OPTIONS = sorted(l1)  
  
NameEn = Entry(root, textvariable=Name)  
NameEn.grid(row=6, column=1)  
  
S1En = OptionMenu(root, Symptom1,\*OPTIONS)  
S1En.grid(row=7, column=1)  
  
S2En = OptionMenu(root, Symptom2,\*OPTIONS)  
S2En.grid(row=8, column=1)  
  
S3En = OptionMenu(root, Symptom3,\*OPTIONS)  
S3En.grid(row=9, column=1)  
  
S4En = OptionMenu(root, Symptom4,\*OPTIONS)  
S4En.grid(row=10, column=1)  
  
S5En = OptionMenu(root, Symptom5,\*OPTIONS)  
S5En.grid(row=11, column=1)  
  
  
gr = Button(root, text=**"View Plot"**, command=grap,bg=**"pink"**,fg=**"Black"**)  
gr.grid(row=11, column=3,padx=10)  
  
dst = Button(root, text=**"DecisionTree"**, command=DecisionTree,bg=**"green"**,fg=**"yellow"**)  
dst.grid(row=8, column=3,padx=10)  
  
rnf = Button(root, text=**"Randomforest"**, command=randomforest,bg=**"green"**,fg=**"yellow"**)  
rnf.grid(row=9, column=3,padx=10)  
  
lr = Button(root, text=**"NaiveBayes"**, command=NaiveBayes,bg=**"green"**,fg=**"yellow"**)  
lr.grid(row=10, column=3,padx=10)  
  
  
t1 = Text(root, height=1, width=40,bg=**"orange"**,fg=**"black"**)  
t1.grid(row=15, column=1, padx=10)  
  
t2 = Text(root, height=1, width=40,bg=**"orange"**,fg=**"black"**)  
t2.grid(row=17, column=1 , padx=10)  
  
t3 = Text(root, height=1, width=40,bg=**"orange"**,fg=**"black"**)  
t3.grid(row=19, column=1 , padx=10)  
  
root.mainloop()

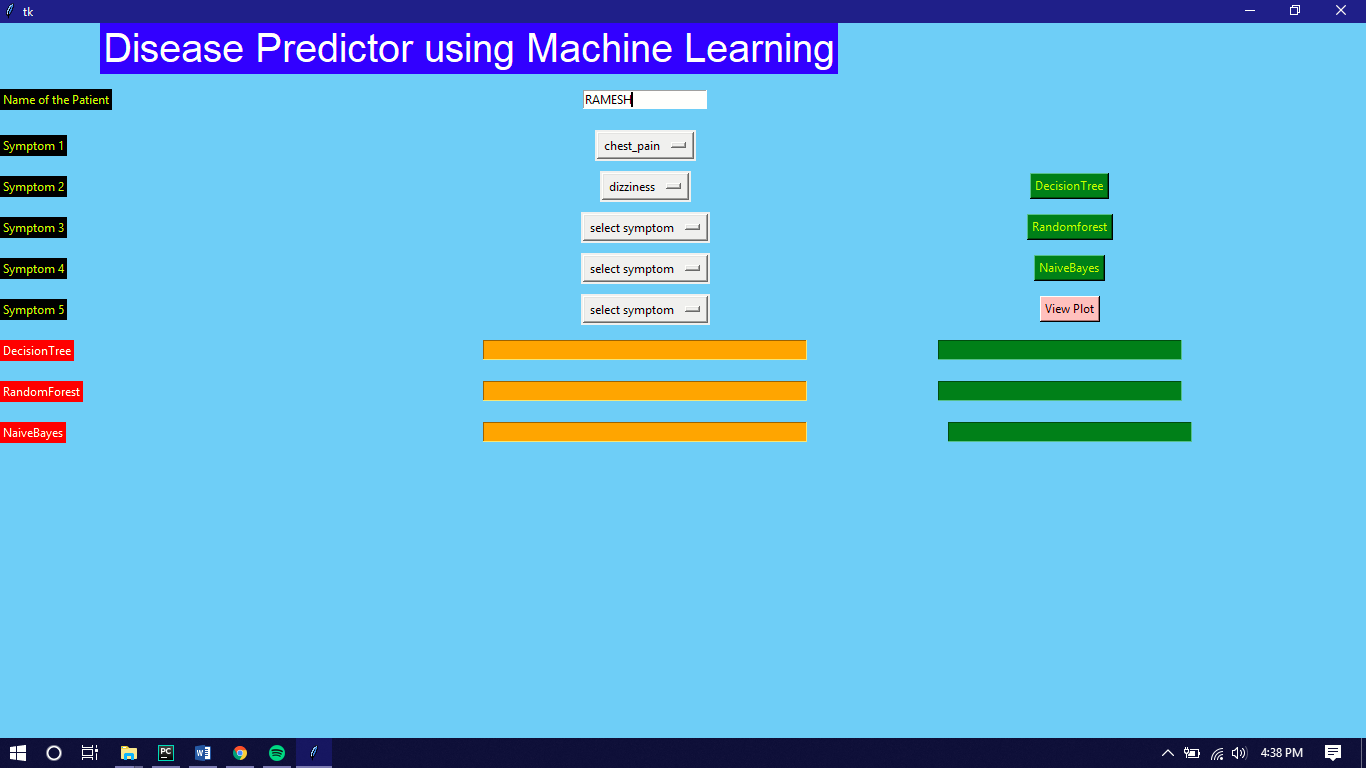
**Output:**

****

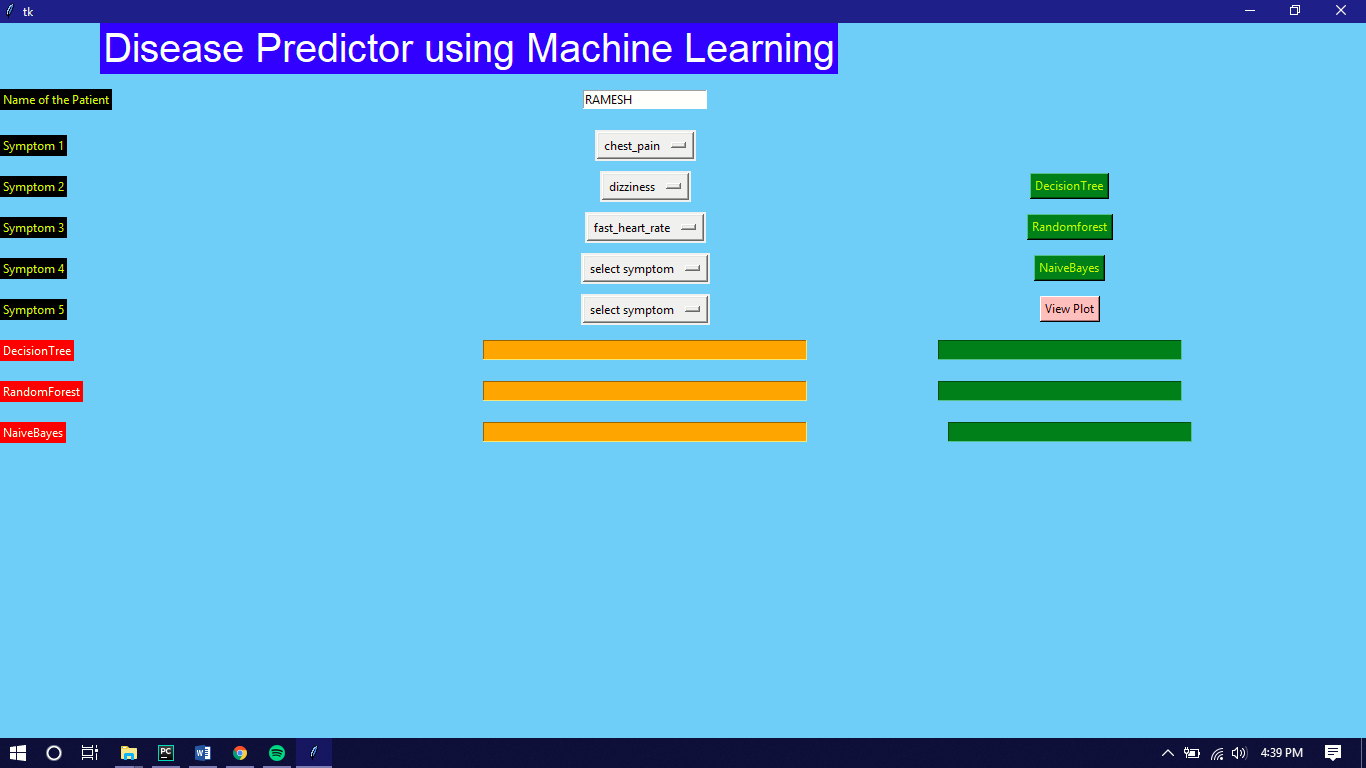
**(5.8.1) Input Name of the Patient**

****

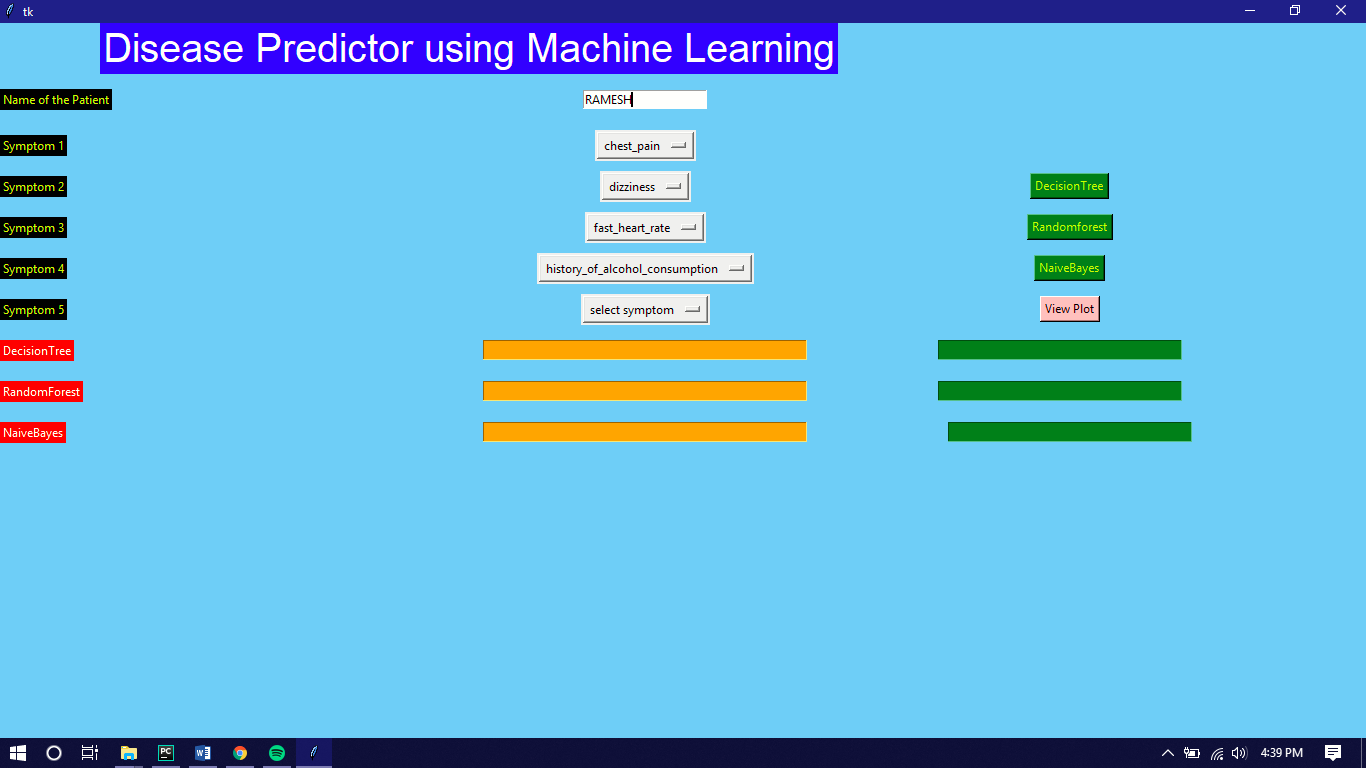
**(5.8.2) Input First Symptom**

****

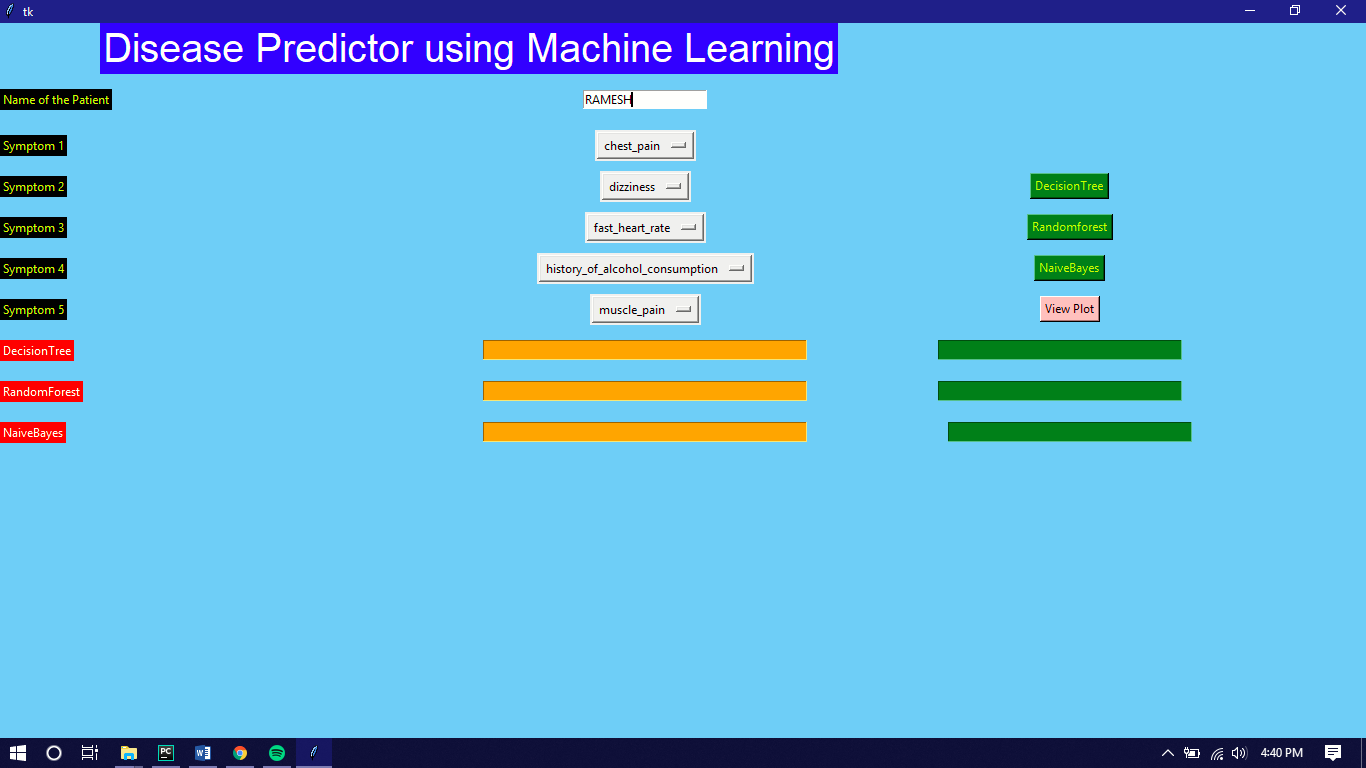
**(5.8.3) Input Second Symptom**

****

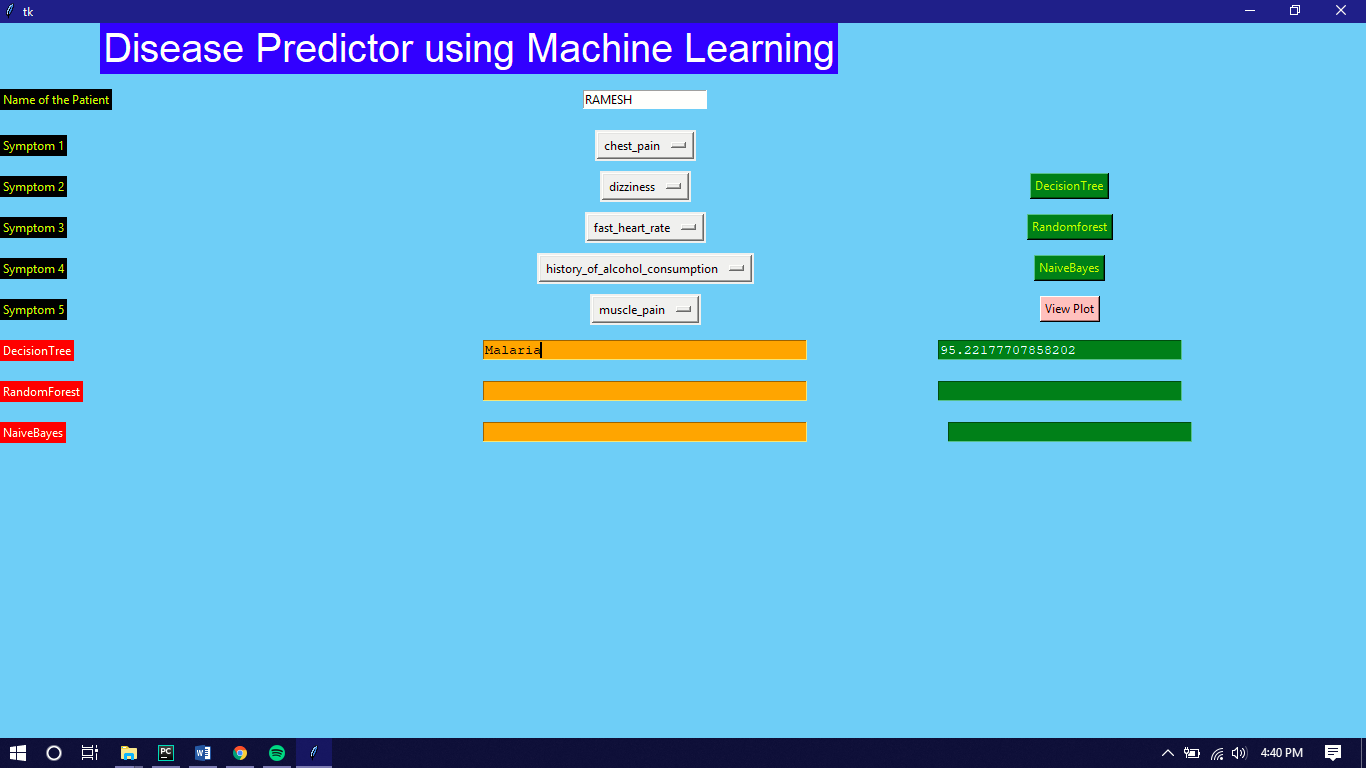
**(5.8.4) Input Third Symptom**

****

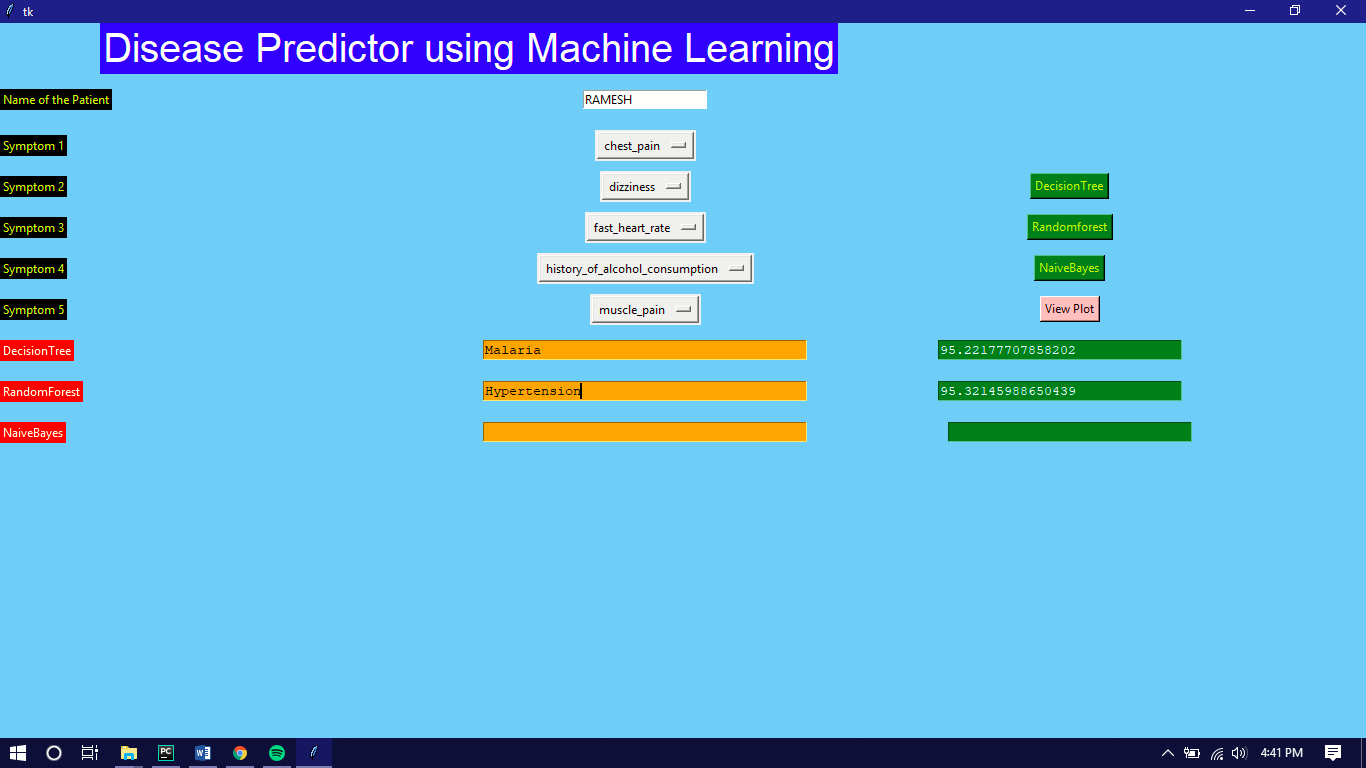
**(5.8.5) Input Fourth Symptom**

****

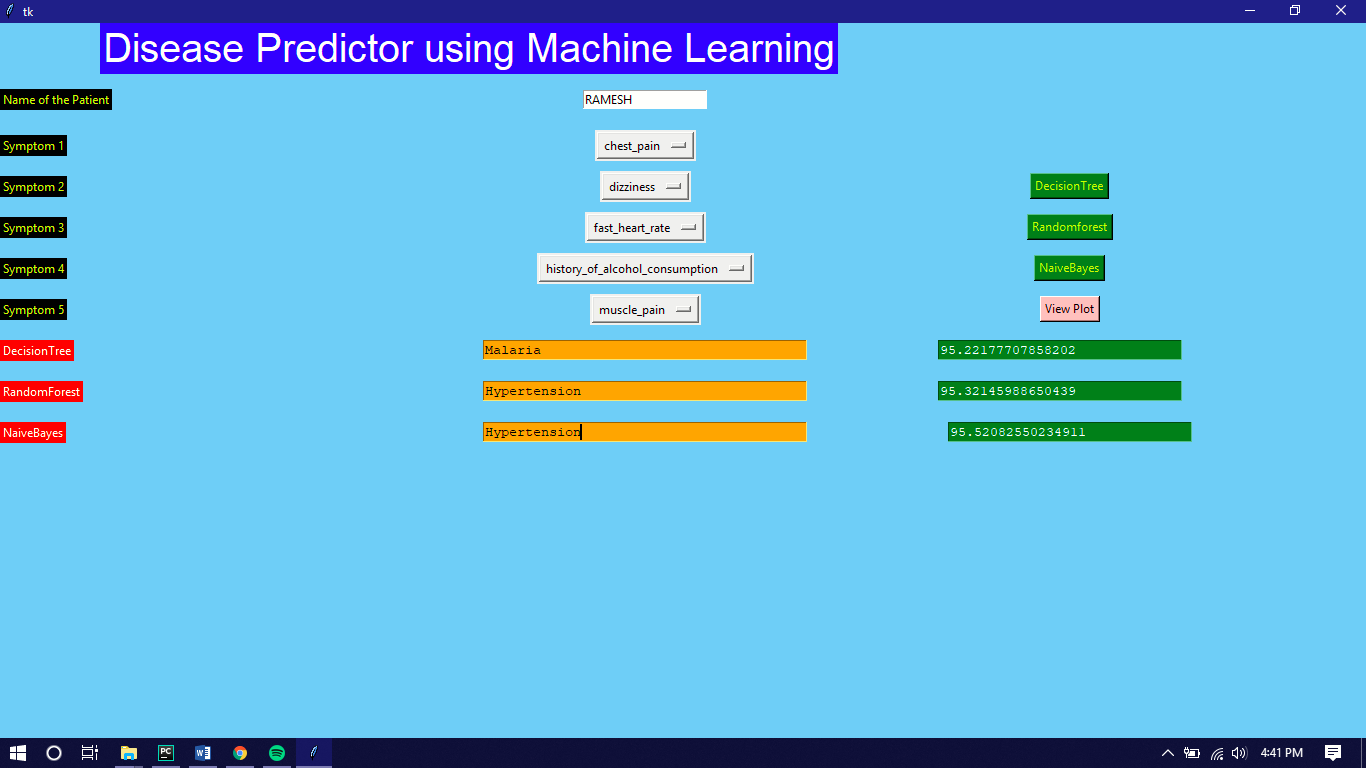
**(5.8.6) Input Fifth Symptom**

****

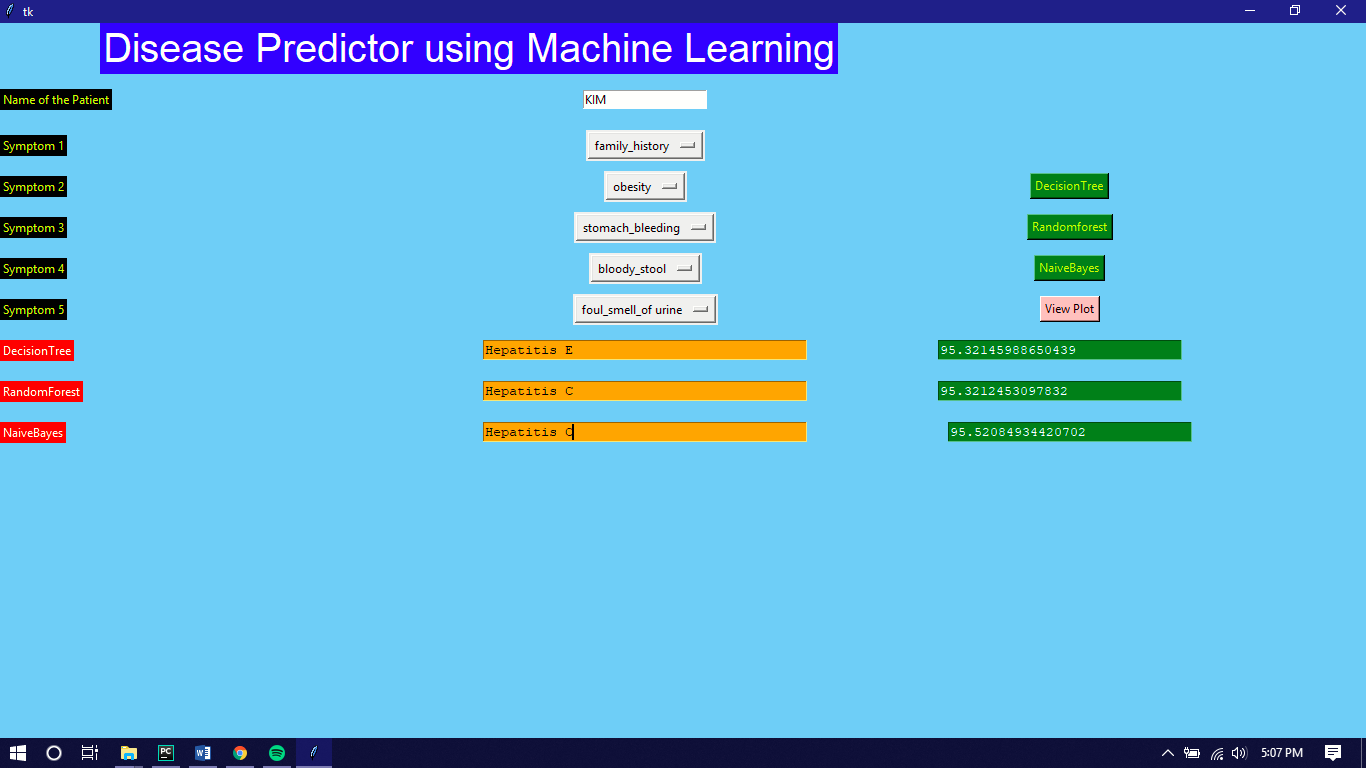
**(5.8.7) Output and Accuracy of Decision Tree Algorithm**

****

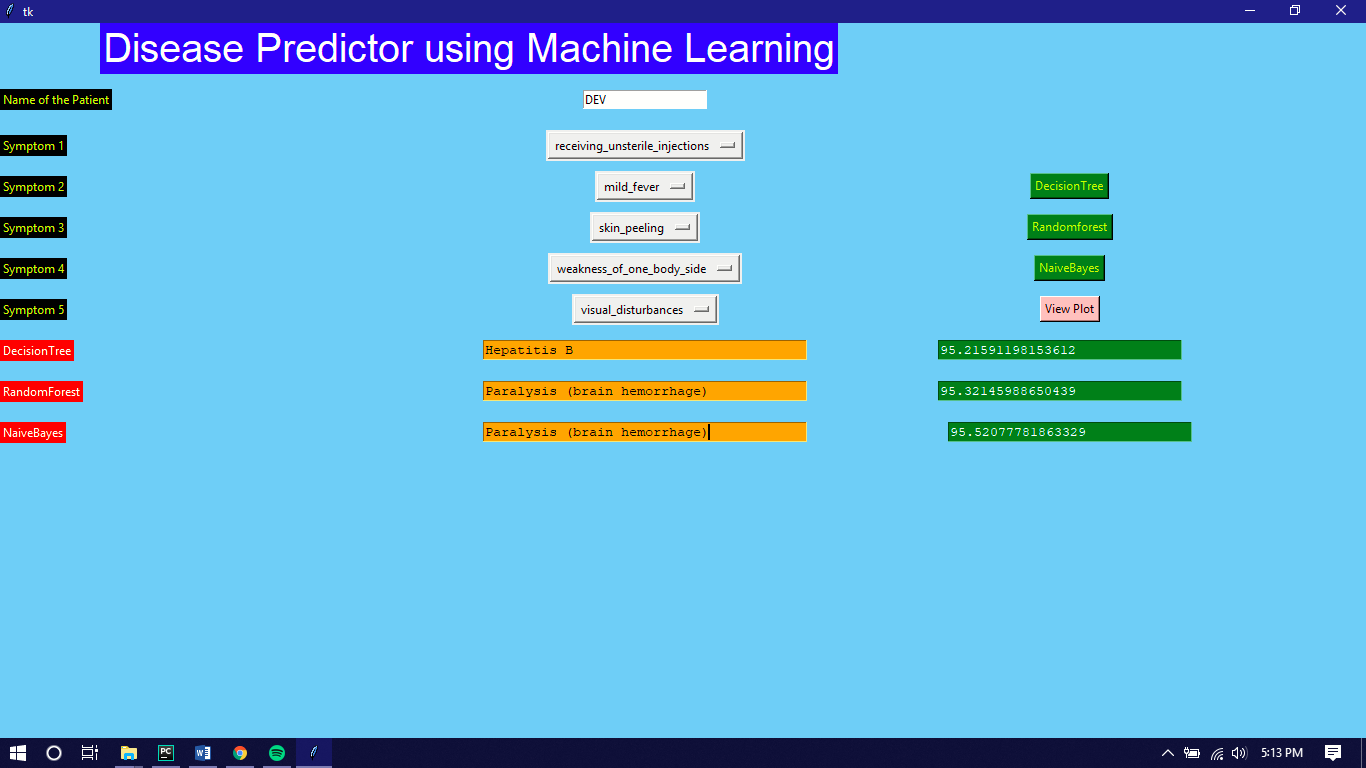
**(5.8.8) Output and Accuracy of Random Forest Algorithm**

****

**(5.8.9) Output and Accuracy of Naïve Bayes Algorithm**

****

**(5.8.10) Another Patient with different Symptoms**

****

**(5.8.11) Another Patient with different Symptoms**

**Project Conclusion**

This project summarizes about the research functionality in establishing huge frameworks in healthcare, using big data and the application of different machine learning algorithms for prognosis of the disease. The contribution made by many researchers for application of different machine algorithms along with accuracy is comparatively shown.

**Future Scope**

In near future we can further investigate on some more deceases by feeding the machine with more number of symptoms. People can easily find the reason for the particular symptoms and can consult the particular specialist without consulting a local clinic.