Disease prediction based on symptoms.

Aditya Arya 11716665, Sudhanshu 11716071, Rohan Agarwal 11716106

KM031, School of Computer Science & Engineering,

Lovely Professional University

.

Project Description

Health information needs are also changing the information seeking behaviour and can be observed around the globe. Challenges faced by many people are looking online for health information regarding diseases, diagnoses and different treatments. If a recommendation system can be made for doctors and medicine while using review mining will save a lot of time. In this type of system, the user face problem in understanding the heterogeneous medical vocabulary as the users are laymen. User is confused because a large amount of medical information on different mediums are available.The idea behind recommender system is to adapt to cope with the special requirements of the health domain related with users.

Introduction

With the rise in number of patient and disease every year medical system is overloaded and with time have become overpriced in many countries. Most of the disease involves a consultation with doctors to get treated. With sufficient data prediction of disease by an algorithm can be very easy and cheap. Prediction of disease by looking at the symptoms is an integral part of treatment. In our project we have tried accurately predict a disease by looking at the symptoms of the patient. We have used 4 different algorithms for this purpose and gained an accuracy of 92-95%. Such a system can have a very large potential in medical treatment of the future. We have also designed an interactive interface to facilitate interaction with the system. We have also attempted to show and visualized the result of our study and this project.

Database Collection

Dataset for this project was collected from a study of university of Columbia performed at New York Presbyterian Hospital during 2004. Link of dataset is given below.

http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

Library Used

In this project standard libraries for database analysis and model creation are used. The following are the libraries used in this project.

1. tkinter: It's a standard GUI library of python. Python when combined with tkinter provides fast and easy way to create GUI. It provides powerful object-oriented tool for creating GUI.

It provides various widgets to create GUI some of the prominent ones being:

- Button
- Canvas
- Label
- Entry
- Check Button
- List box
- Message
- Text
- Messagebox

Some of these were used in this project to create our GUI namely messagebox, button, label, Option Menu, text and title. Using tkinter we were able to create an interactive GUI for our model.

2. Numpy: Numpy is core library of scientific computing in python. It provides powerful tools to deal with various multi-dimensional arrays in python. It is a general purpose array processing package.

Numpy's main purpose is to deal with multidimensional homogeneous array. It has tools ranging from array creation to its handling. It makes it easier to create a n dimensional array just by using np.zeros() or handle its contents using various other methods such as replace, arrange, random, save, load it also helps I array processing using methods like sum, mean, std, max, min, all, etc

Array created with numpy also behave differently then arrays created normally when they are operated upon using operators such as +,-,*,/.

All the above qualities and services offered by numpy array makes it highly suitable for our purpose of handling data. Data manipulation occurring in arrays while performing various operations need to give the desired results while predicting outputs require such high operational capabilities.

3. pandas : it is the most popular python library used for data analysis. It provides highly optimized performance with back-end source code purely written in C or python.

Data in python can be analysed with 2 ways

- Series
- Dataframes

Series is one dimensional array defined in pandas used to store any data type.

Dataframes are two-dimensional data structure used in python to store data consisting of rows and columns.

Pandas dataframe is used extensively in this project to use datasets required for training and testing the algorithms. Dataframes makes it easier to work with attributes and results. Several of its inbuilt functions such as replace were used in our project for data manipulation and preprocessing.

4. sklearn: Sklearn is an open source python library with implements a huge range of machine-learning, pre-processing, cross-validation and visualization algorithms. It features various simple and efficient tools for data mining and data processing. It features various classification, regression and clustering algorithm such as support vector machine, random forest classifier, decision tree, gaussian naïve-Bayes, KNN to name a few.

In this project we have used sklearn to get advantage of inbuilt classification algorithms like decision tree, random forest classifier, KNN and naïve Bayes. We have also used inbuilt cross validation and visualization features such as classification report, confusion matrix and accuracy score.

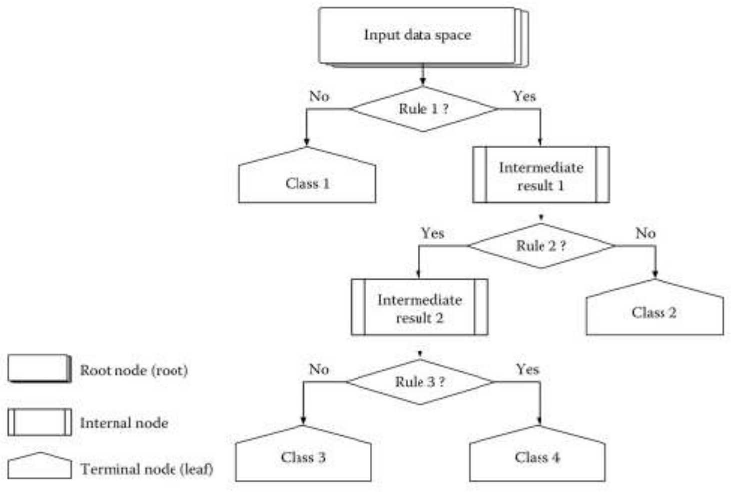
Models

There are four different kind of models present in our project to predict the disease these are

- Decision tree
- Random forest tree
- Gaussian Naïve Bayes
- KNN

Decision tree is classified as a very effective and versatile classification technique. It is used in pattern recognition and classification for image. It is used for classification in very complex problems dew to its high adaptability. It is also capable of engaging problems of higher dimensionality. It mainly consists of three parts root, nodes and leaf.

Roots consists of attribute which has most effect on the outcome, leaf tests for value of certain attribute and leaf gives out the output of tree.



Decision tree is the first prediction method we have used in our project. It gives us an accuracy of ~95%.

Random Forest Algorithm is a supervised learning algorithm used for both classification and regression. This algorithm works on 4 basic steps –

1. It chooses random data samples from dataset.
2. It constructs decision trees for every sample dataset chosen.
3. At this step every predicted result will be compiled and voted on.
4. At last most voted prediction will be selected and be presented as result of classification.

In this project we have used random forest classifier with 100 random samples and the result given is ~95% accuracy.

K Nearest Neighbour is a supervised learning algorithm. It is a basic yet essential algorithm. It finds extensive use in pattern finding and data mining.

It works by finding a pattern in data which links data to results and it improves upon the patter recognition with every iteration.

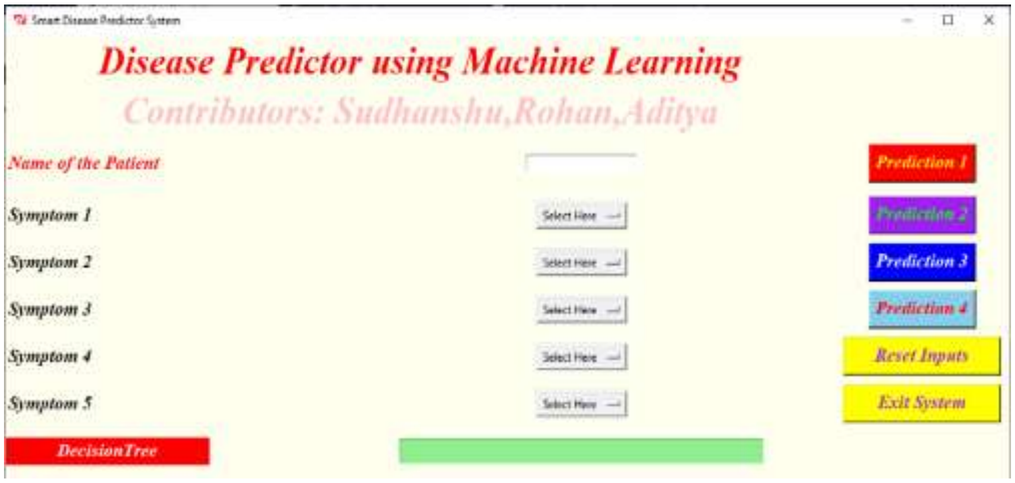
We have used K Nearest Neighbour to classify our dataset and achieved ~92% accuracy.

Naïve Bayes algorithm is a family of algorithms based on naïve bayes theorem. They share a common principle that is every pair of prediction is independent of each other. It also makes an assumption that features make an independent and equal contribution to the prediction.

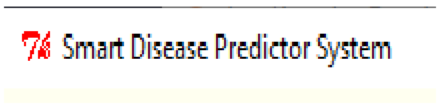
In our project we have used naïve bayes algorithm to gain a ~95% accurate prediction.

GUI

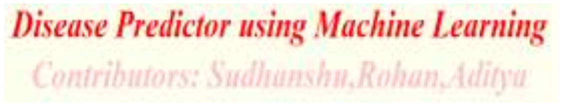
GUI made for this project is a simple tkinter GUI consisting of labels, messagebox, button, text, title and option menu



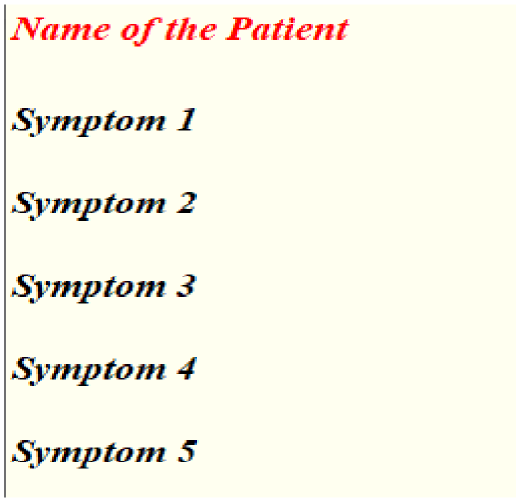
Root.title() is used to set the the title as Smart Disease Predictor System



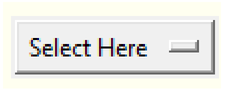
Label is used to add heading and contributors section.


Disease Predictor using Machine Learning
Contributors: Sudhanshu,Rohan,Aditya

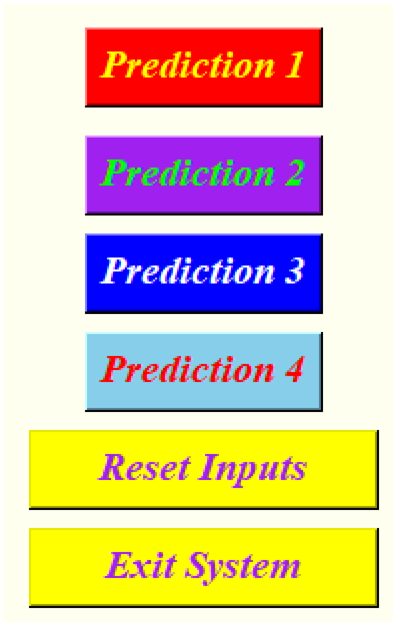
Labels are further used for different sections



OptionMenu is used to create drop down menu

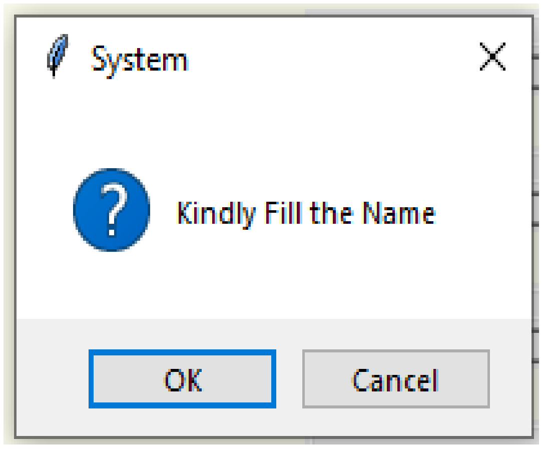


Buttons are used to give functionalities and predict the out come of models also two utility buttons namely exit and rest are also created.



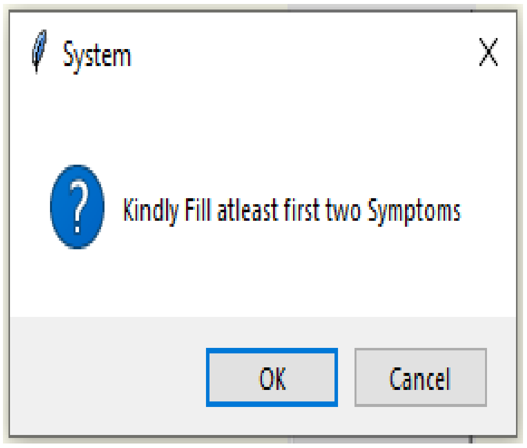
Text is used to show output of the prediction using blank space.

Messagebox are used at three different places, one- to restrain the to enter name


System

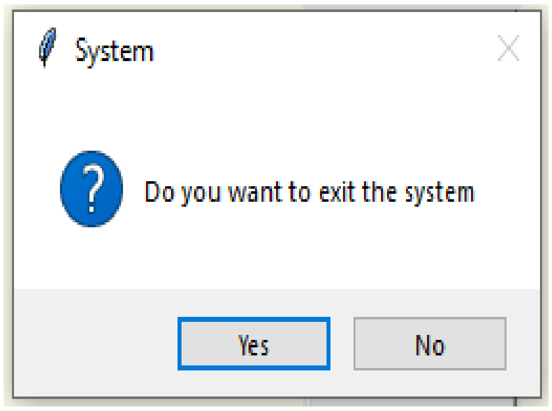
Kindly Fill the Name

two- to ask for at least two symptoms,


System

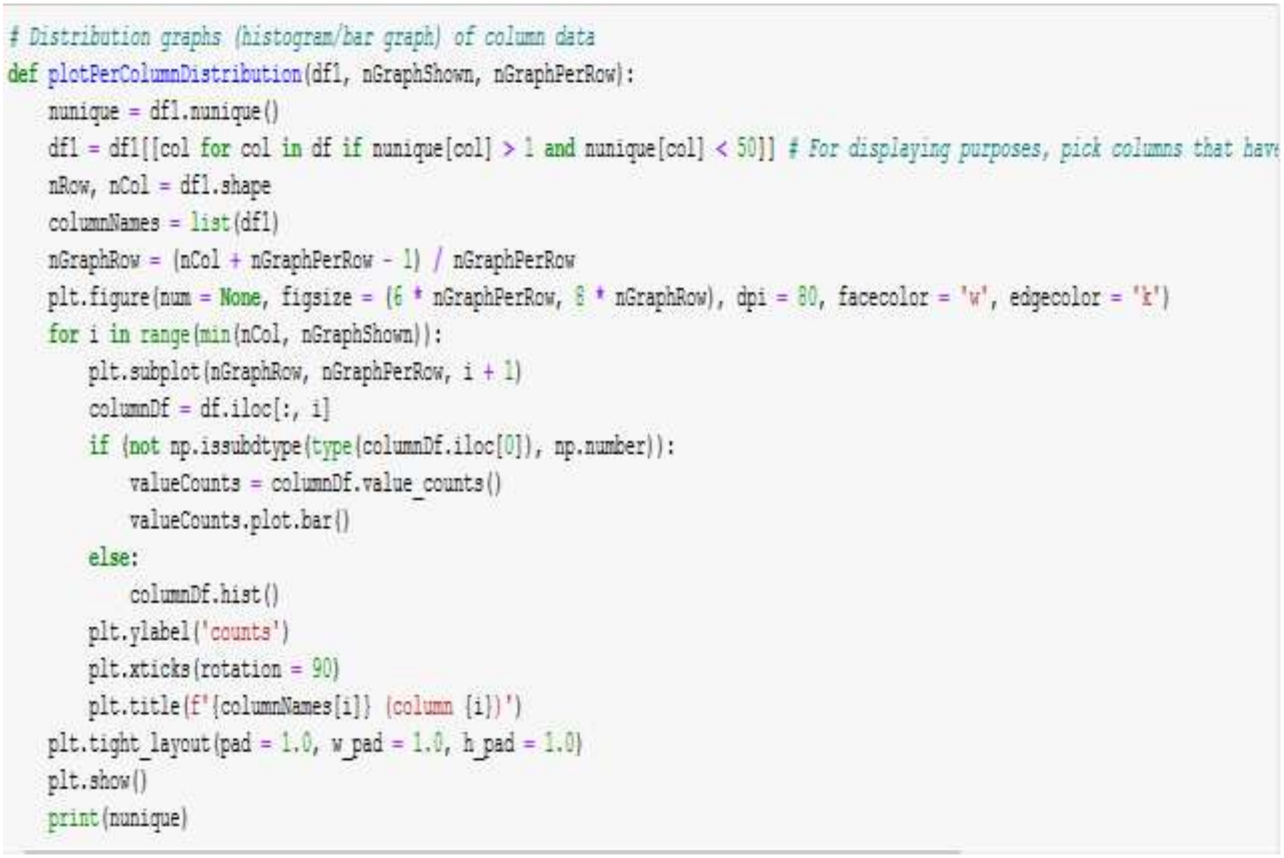
Kindly Fill atleast first two Symptoms

three- to confirm to exit system.

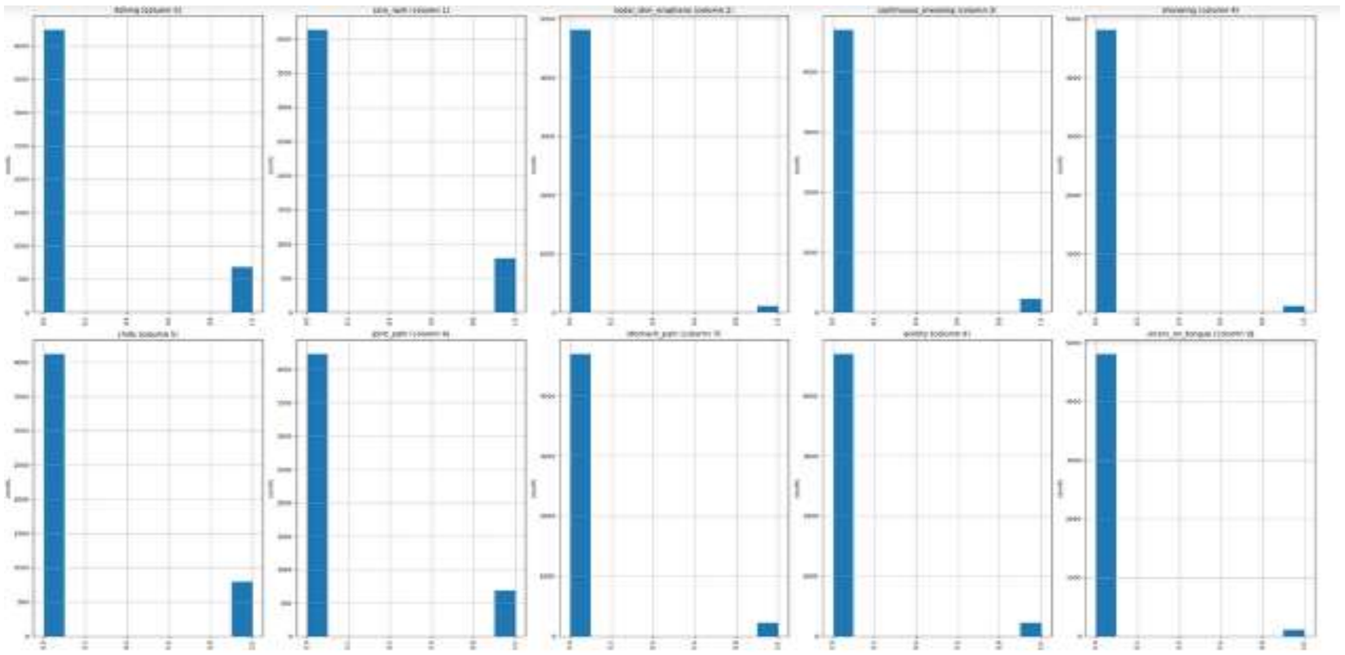

System

Do you want to exit the system

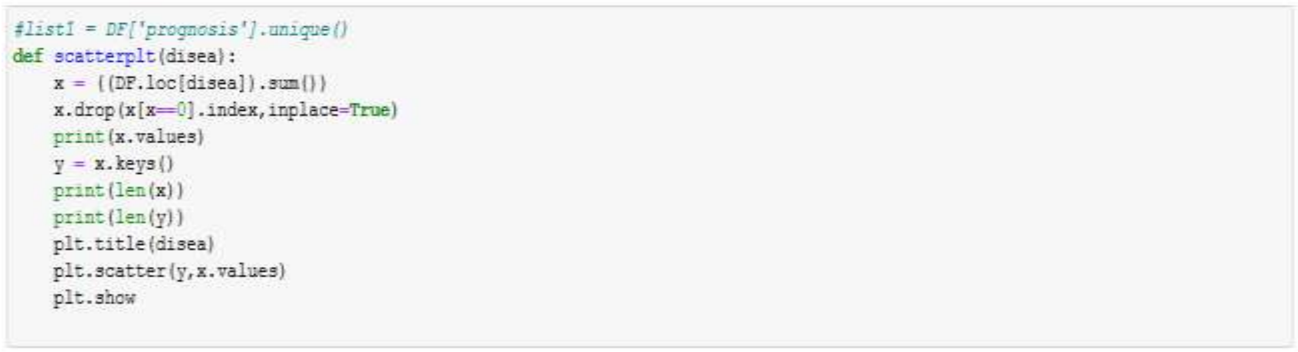
Modules



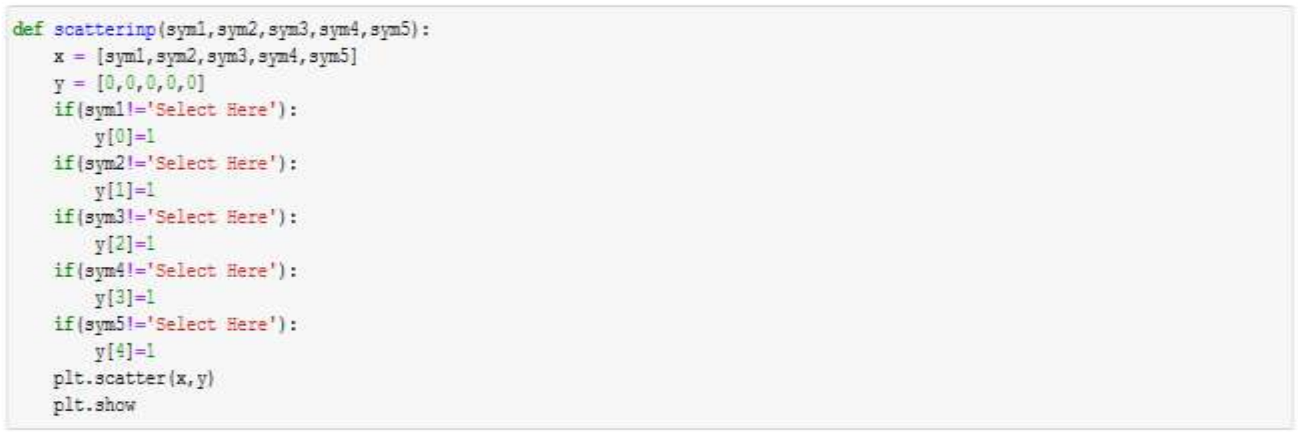
```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have
    nRow, nCol = df1.shape
    columnNames = list(df1)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
    print(nunique)
```



Functions like plotpercolumbdistribution() plotScatterMatrix() is used to viaualize the data.

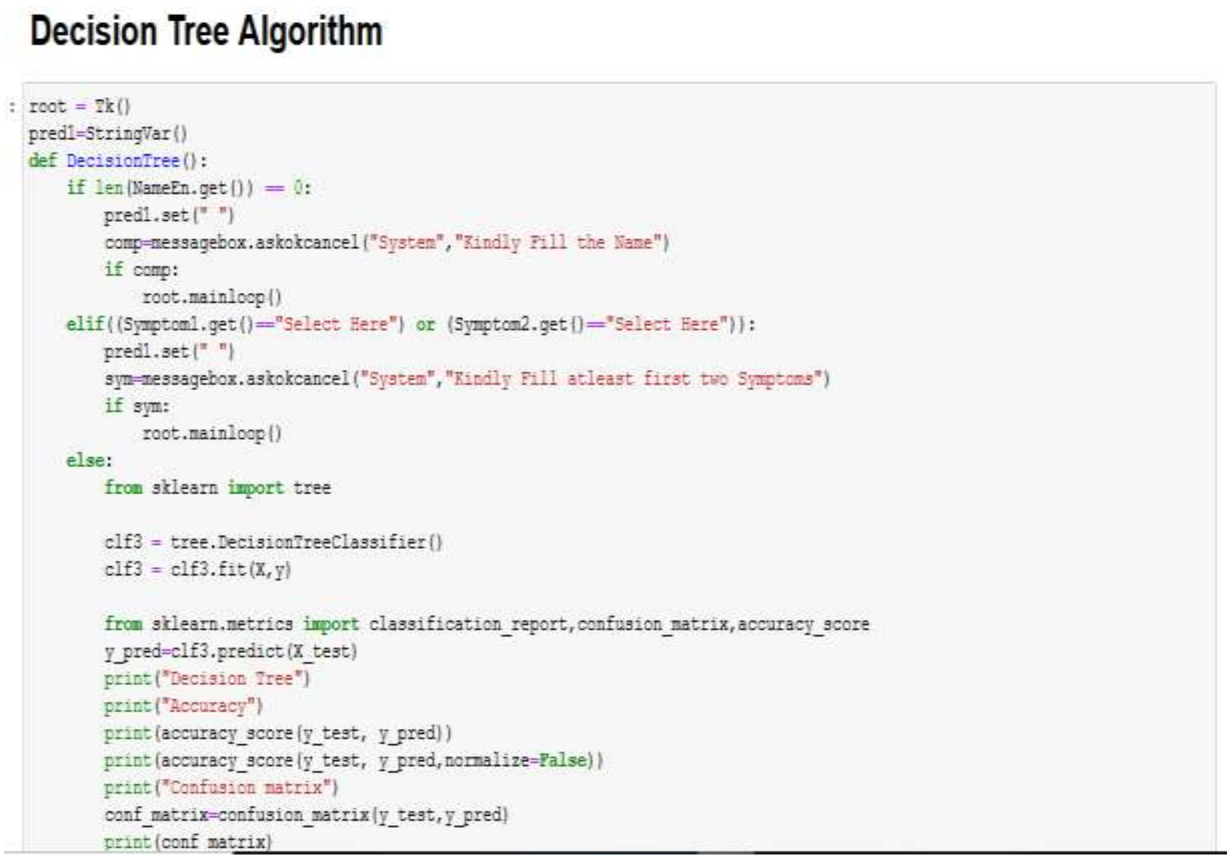


```
#list1 = DF['prognosis'].unique()
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())
    x.drop(x[x==0].index,inplace=True)
    print(x.values)
    y = x.keys()
    print(len(x))
    print(len(y))
    plt.title(disea)
    plt.scatter(y,x.values)
    plt.show
```



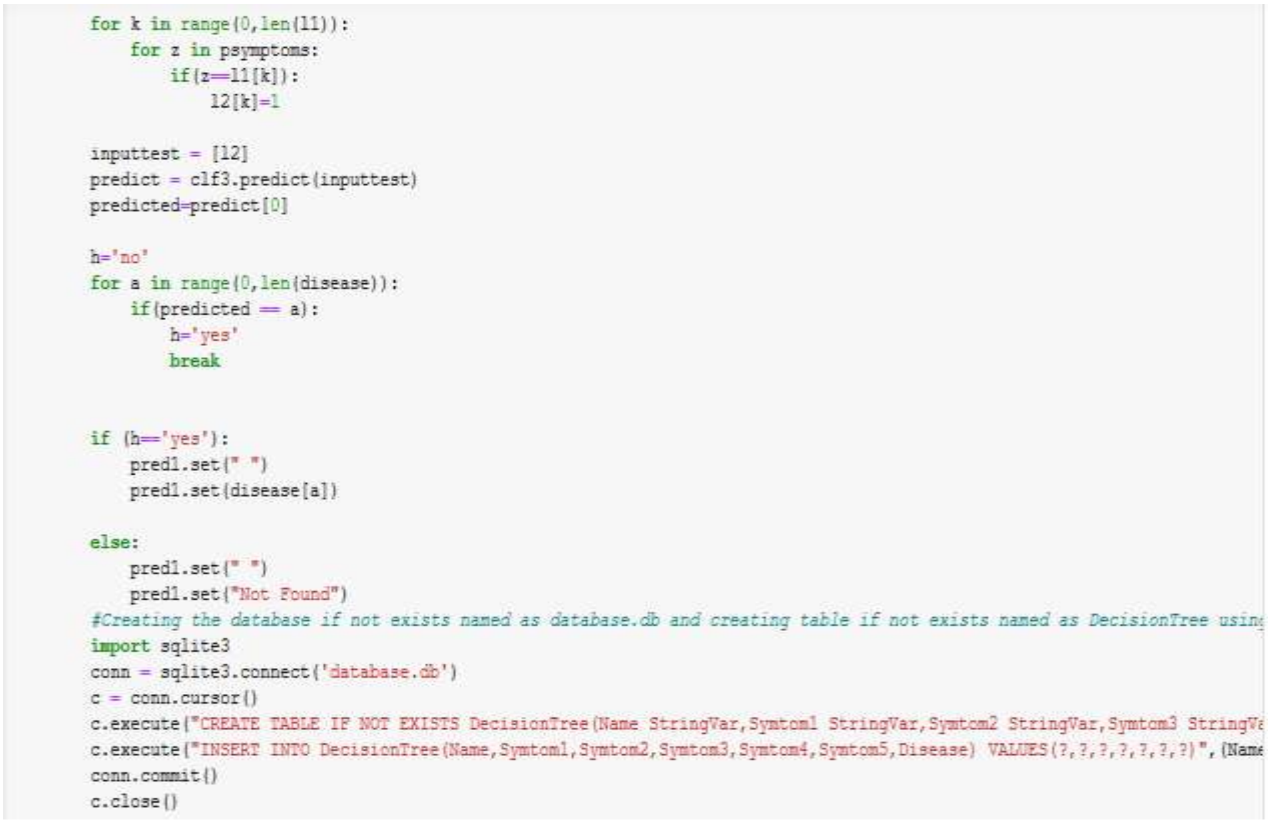
```
def scatterinp(sym1,sym2,sym3,sym4,sym5):
    x = [sym1,sym2,sym3,sym4,sym5]
    y = [0,0,0,0,0]
    if(sym1!='Select Here'):
        y[0]=1
    if(sym2!='Select Here'):
        y[1]=1
    if(sym3!='Select Here'):
        y[2]=1
    if(sym4!='Select Here'):
        y[3]=1
    if(sym5!='Select Here'):
        y[4]=1
    plt.scatter(x,y)
    plt.show
```

Function like scatterplt and scatterinp are used to compare input to training data.



Decision Tree Algorithm

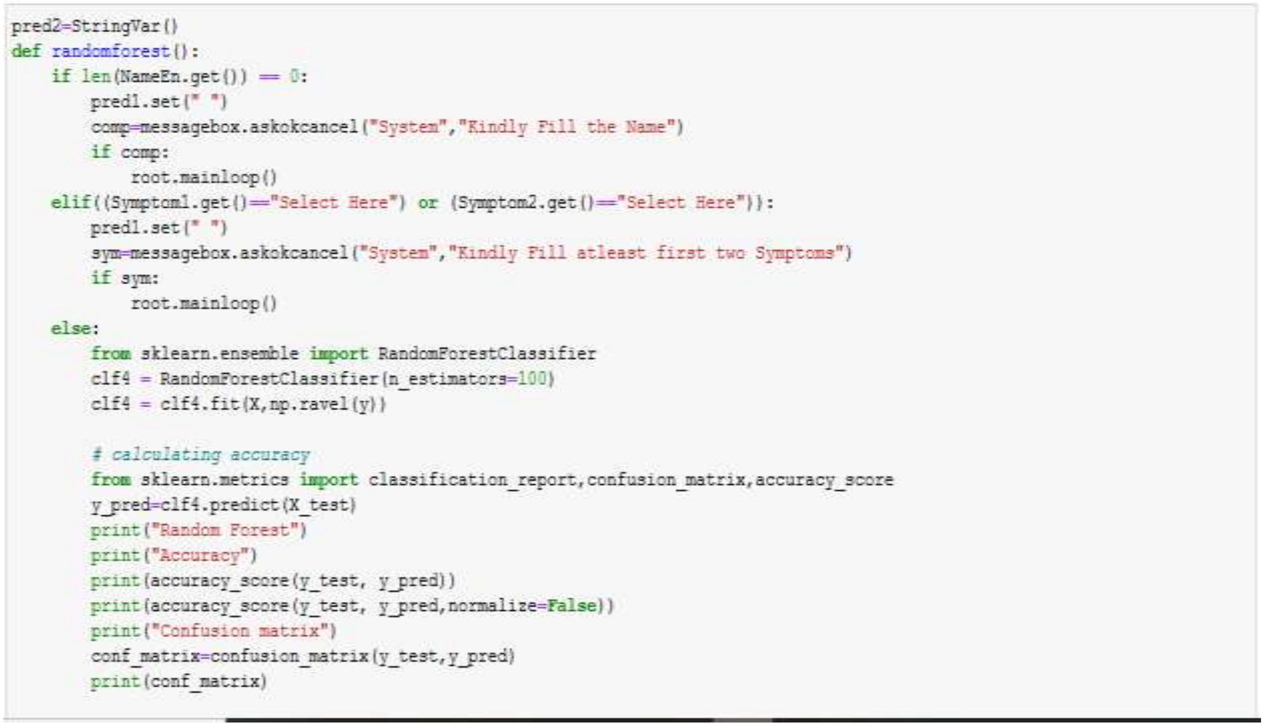
```
root = Tk()
pred1=StringVar()
def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn import tree

        clf3 = tree.DecisionTreeClassifier()
        clf3 = clf3.fit(X,y)

        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=clf3.predict(X_test)
        print("Decision Tree")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)
```



```
        for k in range(0,len(l1)):
            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'):
            pred1.set(" ")
            pred1.set(disease[a])

        else:
            pred1.set(" ")
            pred1.set("Not Found")
        #Creating the database if not exists named as database.db and creating table if not exists named as DecisionTree usin
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVa
        c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(Name
        conn.commit()
        c.close()
```

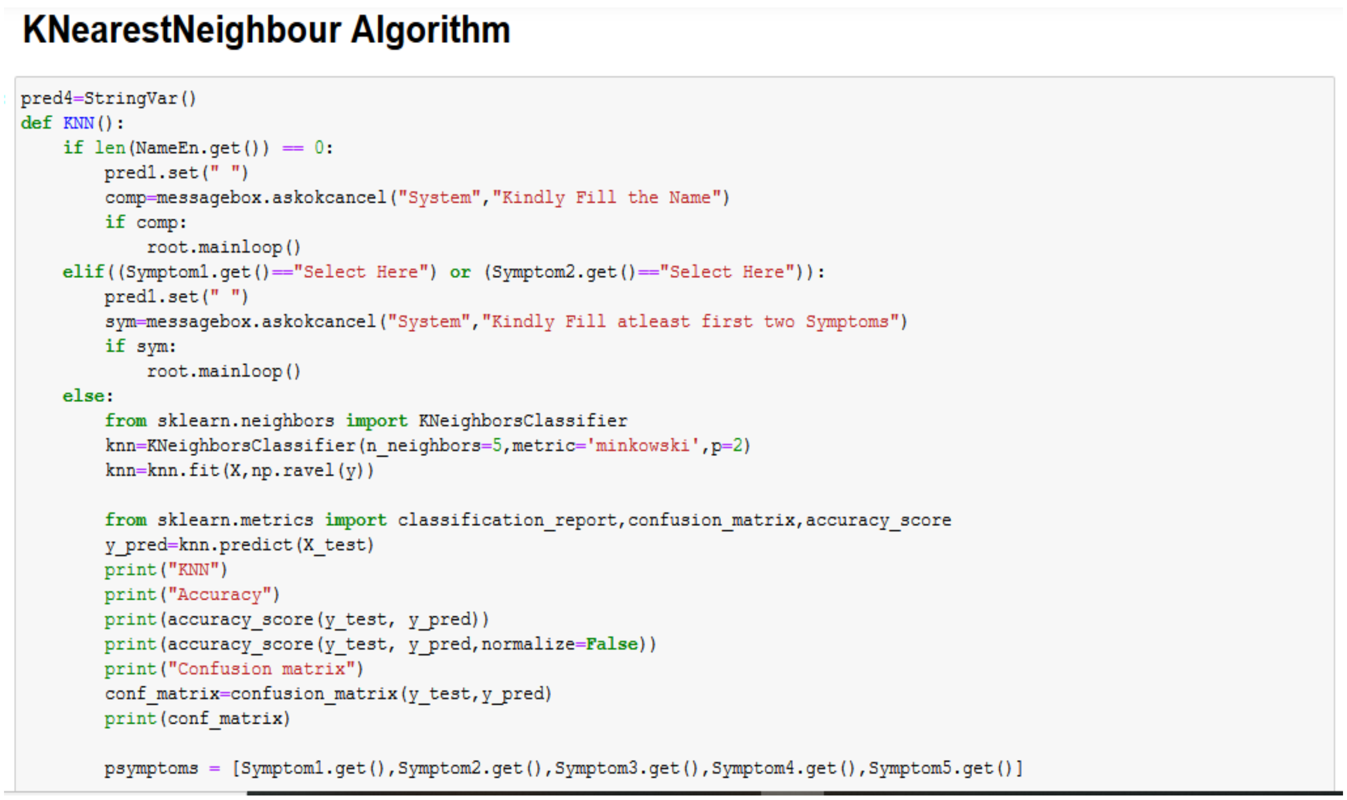
Algorithm of decision tree and database storage.

Random Forest Algorithm



```
pred2=StringVar()
def randomforest():
    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:
        from sklearn.ensemble import RandomForestClassifier
        clf4 = RandomForestClassifier(n_estimators=100)
        clf4 = clf4.fit(X,np.ravel(y))

        # calculating accuracy
        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=clf4.predict(X_test)
        print("Random Forest")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)
```

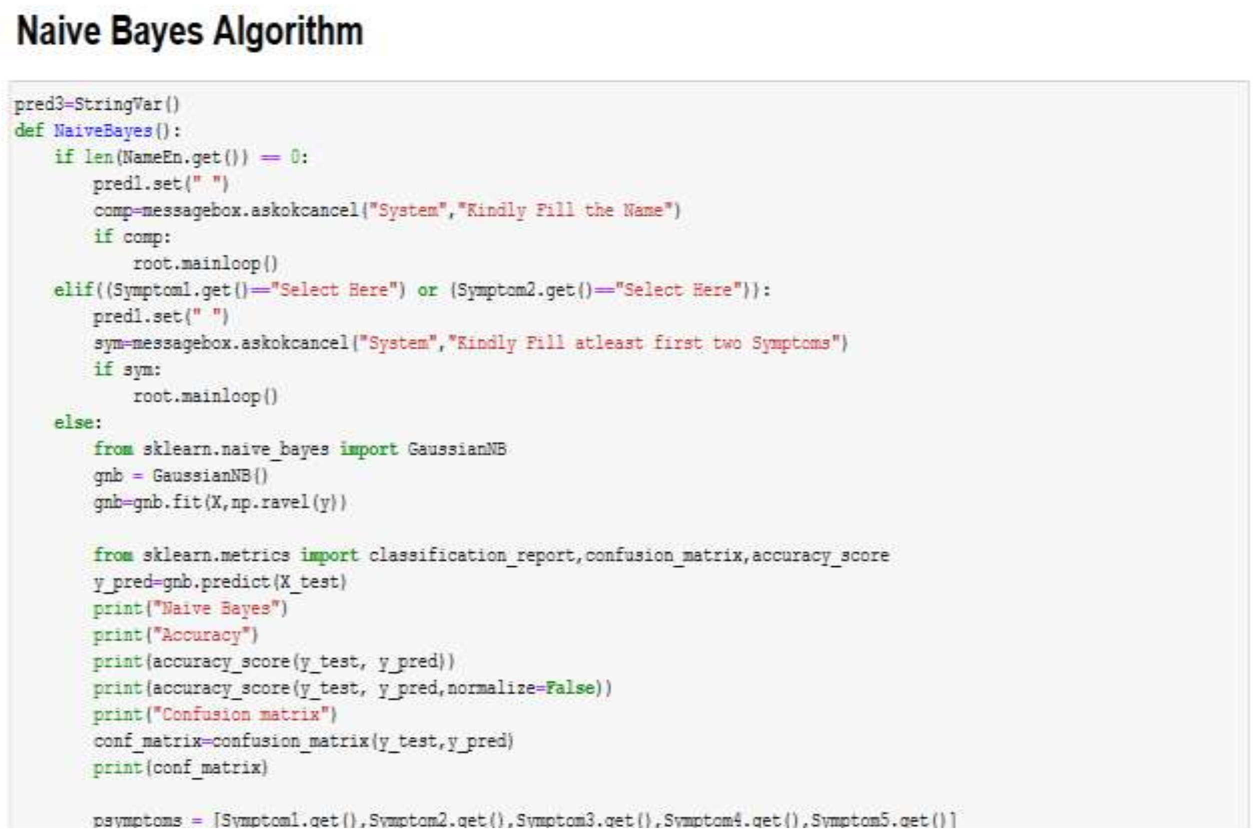
Algorithm of random forest classifier.



KNearestNeighbour Algorithm

```python
pred4=StringVar()
def KNN():
    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:
        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_pred=knn.predict(X_test)
        print("KNN")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
```

Algorithm of K nearest neighbour.

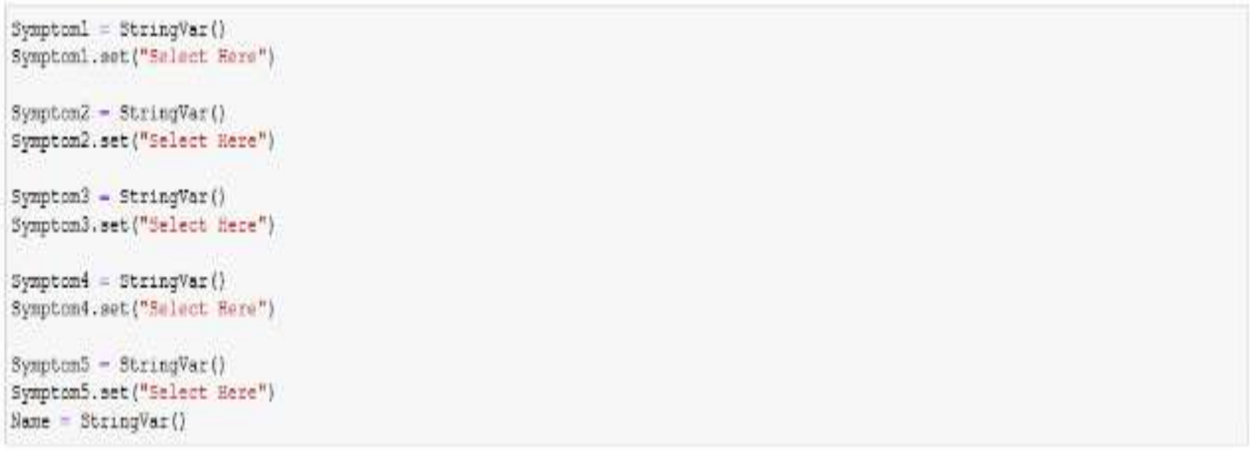


Naive Bayes Algorithm

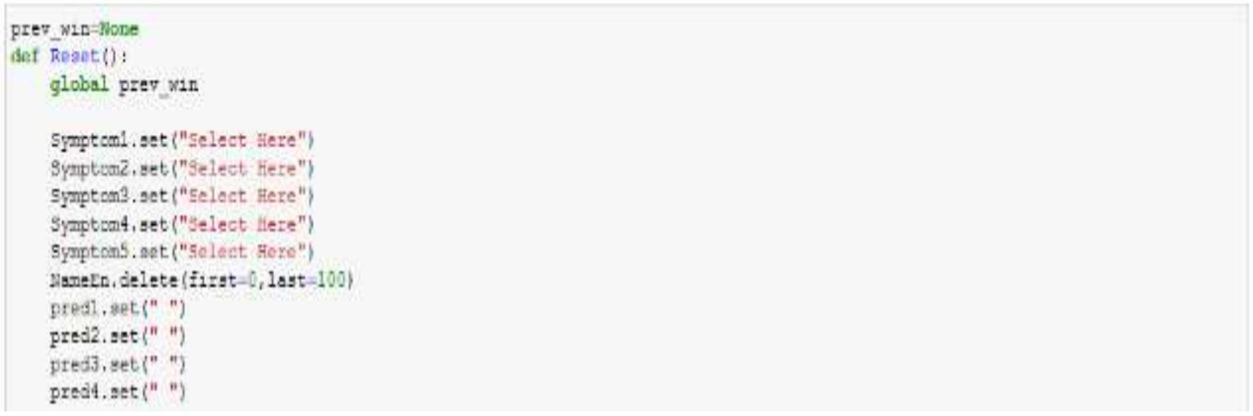
```python
pred3=StringVar()
def NaiveBayes():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        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_pred=gnb.predict(X_test)
        print("Naive Bayes")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
```

Algorithm of naïve bayes classifier

All these classifier is connected to database and GUI to function seamlessly.

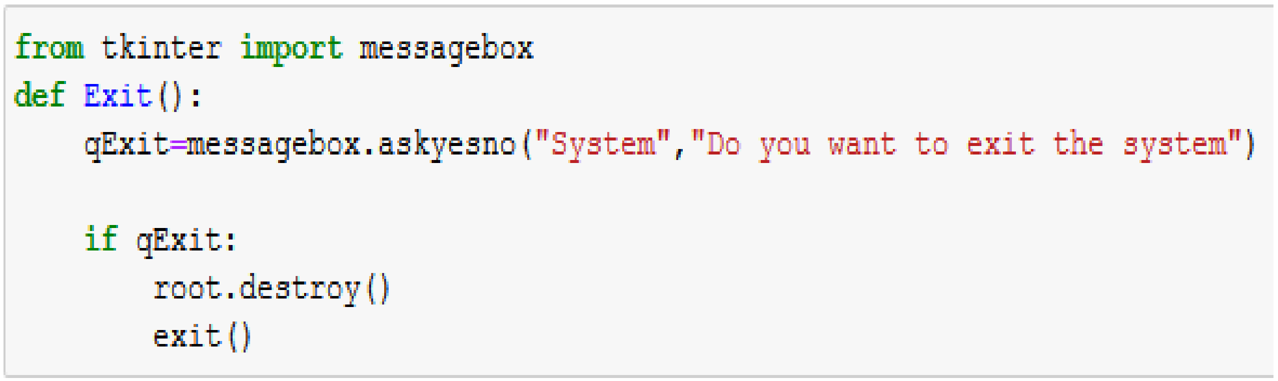


```
Symptom1 = StringVar()
Symptom1.set("Select Here")

Symptom2 = StringVar()
Symptom2.set("Select Here")

Symptom3 = StringVar()
Symptom3.set("Select Here")

Symptom4 = StringVar()
Symptom4.set("Select Here")

Symptom5 = StringVar()
Symptom5.set("Select Here")
Name = StringVar()
```



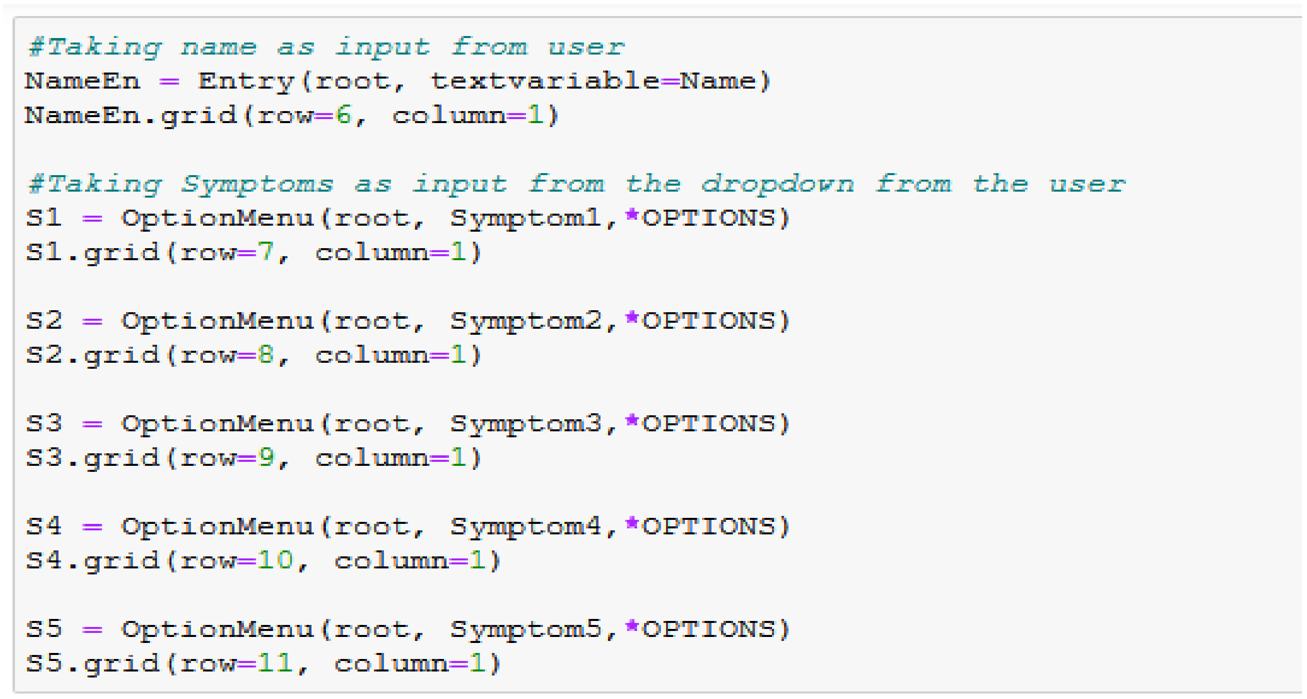
```
prev_win=None
def Reset():
    global prev_win

    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)
    pred1.set(" ")
    pred2.set(" ")
    pred3.set(" ")
    pred4.set(" ")
```

Code of GUI to set initial values of labels.



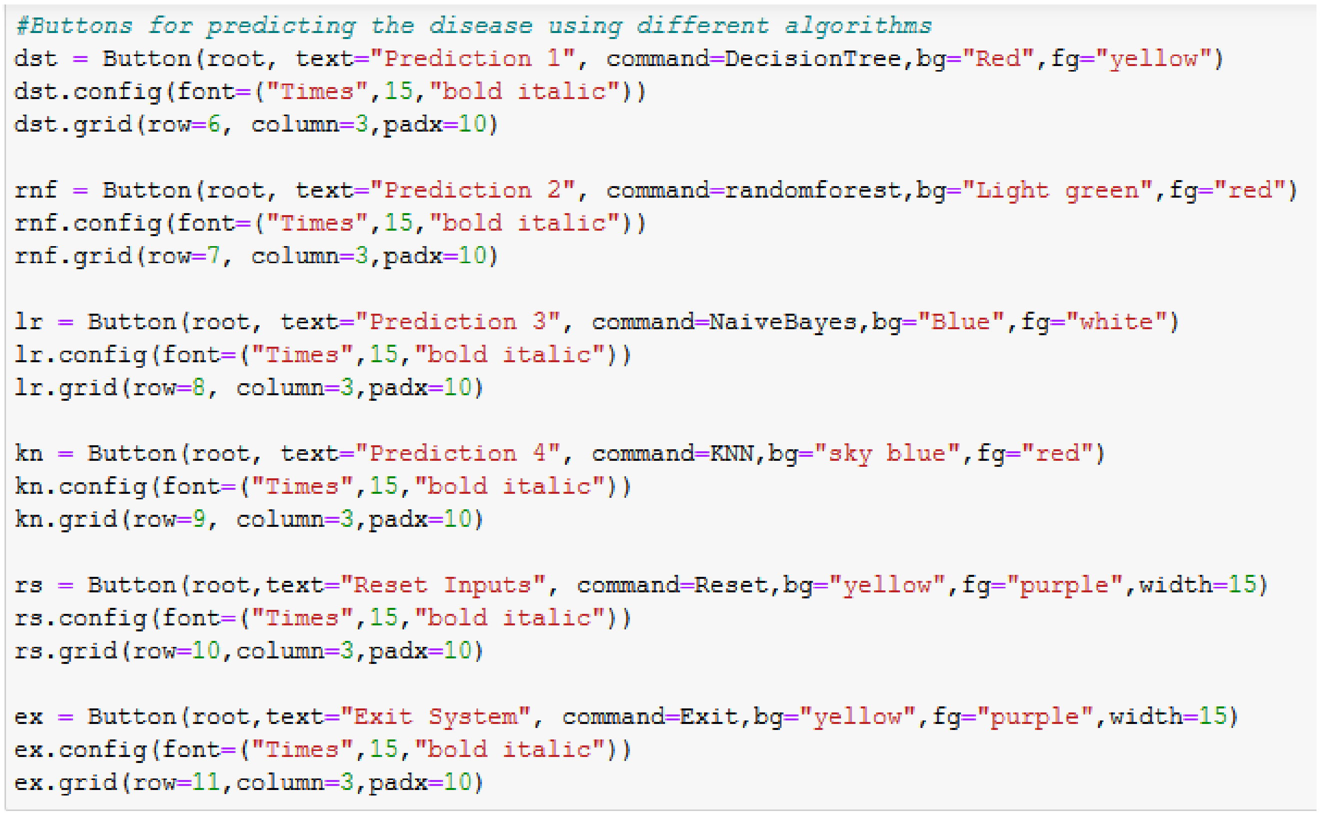
```
from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")

    if qExit:
        root.destroy()
        exit()
```

Code of message box.



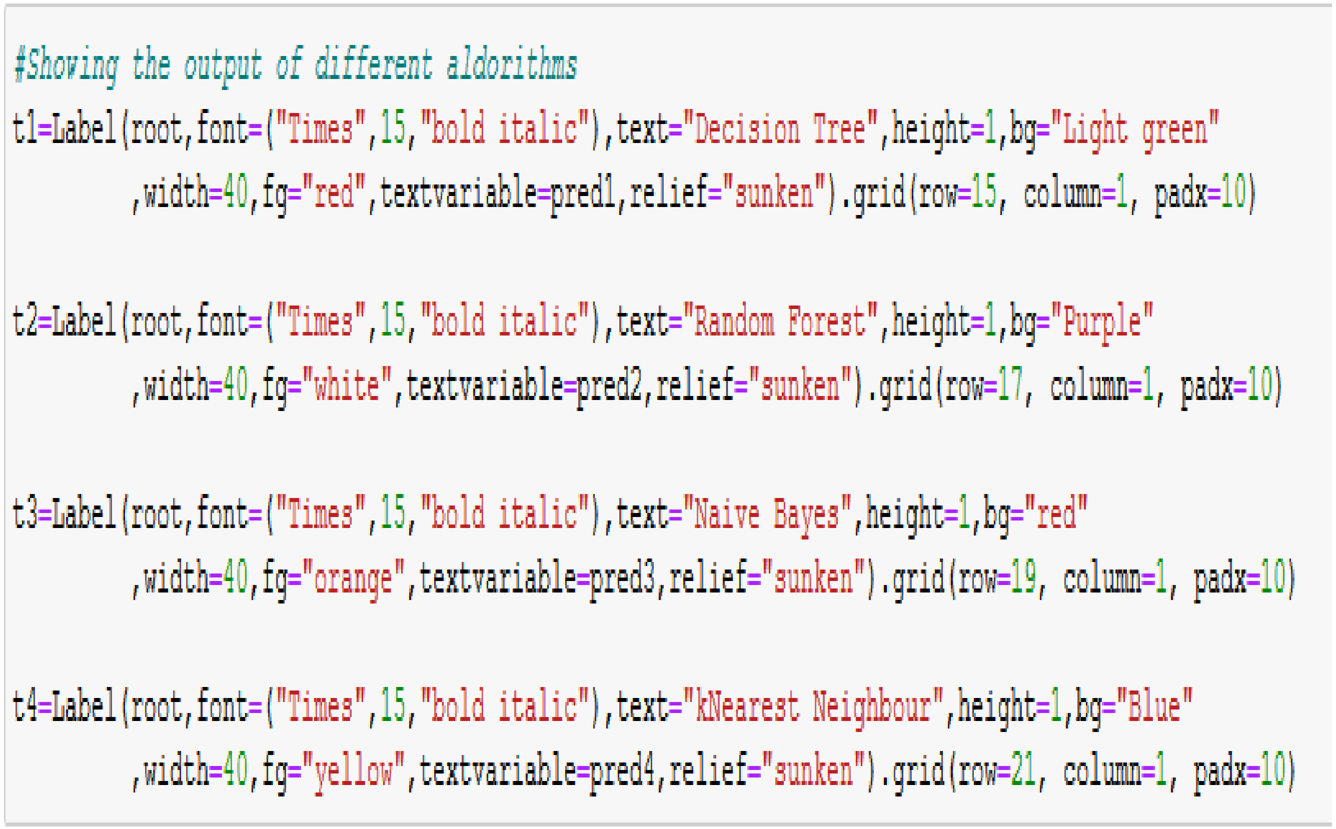
```
#Taking name as input from user
NameEn = Entry(root, textvariable=Name)
NameEn.grid(row=6, column=1)

#Taking Symptoms as input from the dropdown from the user
S1 = OptionMenu(root, Symptom1,*OPTIONS)
S1.grid(row=7, column=1)

S2 = OptionMenu(root, Symptom2,*OPTIONS)
S2.grid(row=8, column=1)

S3 = OptionMenu(root, Symptom3,*OPTIONS)
S3.grid(row=9, column=1)

S4 = OptionMenu(root, Symptom4,*OPTIONS)
S4.grid(row=10, column=1)

S5 = OptionMenu(root, Symptom5,*OPTIONS)
S5.grid(row=11, column=1)
```

Code of option menu



```
#Buttons for predicting the disease using different algorithms
dst = Button(root, text="Prediction 1", command=DecisionTree,bg="Red",fg="yellow")
dst.config(font=("Times",15,"bold italic"))
dst.grid(row=6, column=3,padx=10)

rnf = Button(root, text="Prediction 2", command=randomforest,bg="Light green",fg="red")
rnf.config(font=("Times",15,"bold italic"))
rnf.grid(row=7, column=3,padx=10)

lr = Button(root, text="Prediction 3", command=NaiveBayes,bg="Blue",fg="white")
lr.config(font=("Times",15,"bold italic"))
lr.grid(row=8, column=3,padx=10)

kn = Button(root, text="Prediction 4", command=KNN,bg="sky blue",fg="red")
kn.config(font=("Times",15,"bold italic"))
kn.grid(row=9, column=3,padx=10)

rs = Button(root,text="Reset Inputs", command=Reset,bg="yellow",fg="purple",width=15)
rs.config(font=("Times",15,"bold italic"))
rs.grid(row=10,column=3,padx=10)

ex = Button(root,text="Exit System", command=Exit,bg="yellow",fg="purple",width=15)
ex.config(font=("Times",15,"bold italic"))
ex.grid(row=11,column=3,padx=10)
```

Code of buttons.



```
#Showing the output of different aldorithms
t1=Label(root,font=("Times",15,"bold italic"),text="Decision Tree",height=1,bg="Light green"
         ,width=40,fg="red",textvariable=pred1,relief="sunken").grid(row=15, column=1, padx=10)

t2=Label(root,font=("Times",15,"bold italic"),text="Random Forest",height=1,bg="Purple"
         ,width=40,fg="white",textvariable=pred2,relief="sunken").grid(row=17, column=1, padx=10)

t3=Label(root,font=("Times",15,"bold italic"),text="Naive Bayes",height=1,bg="red"
         ,width=40,fg="orange",textvariable=pred3,relief="sunken").grid(row=19, column=1, padx=10)

t4=Label(root,font=("Times",15,"bold italic"),text="kNearest Neighbour",height=1,bg="Blue"
         ,width=40,fg="yellow",textvariable=pred4,relief="sunken").grid(row=21, column=1, padx=10)
```

Code of result display.

Conclusions

We set out to create a system which can predict disease on the basis of symptoms given to it. Such a system can decrease the rush at OPDs of hospitals and reduce the workload on medical staff. We were successful in creating such a system and use 4 different algorithm to do so. On an average we achieved accuracy of ~94%. Such a system can be largely reliable to do the job. Creating this system we also added a way to store the data entered by the user in the database which can be used in future to help in creating better version of such system. Our system also has an easy to use interface. It also has various visual representation of data collected and results achieved.

References

a. http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html