MACHINE LEARNING PROJECT

On

Disease Prediction based on Symptoms

Submitted by:

P. Damodhar Sai – 1700303C204

D. Surya Praneeth Reddy- 1700305C204

M. Phani Sri Ram- 1700340C204

EV Karthik Reddy – 1600397C204

Subject: Machine Learning

Submitted to:

Dr. Shuvabrata Bandopadhaya

BML Munjal University

December, 2020

Project Description:

Health information needs are also changing the information seeking behaviour and can be observed around the globe. Challenges are faced by many people who 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 it will save a lot of time. In this type of system, the user is generally gets confused because of 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 tools 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

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 for 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 array processing using methods like sum, mean, std, max, min, all, etc

Array created with numpy also behave differently than 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 which 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.

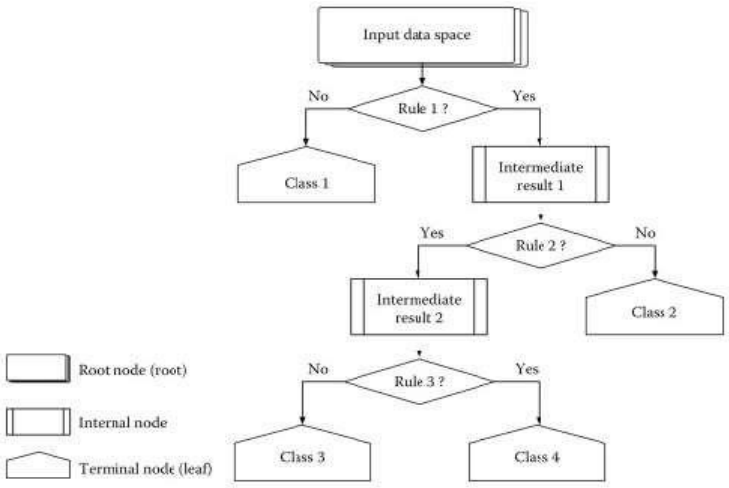
Models

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 due 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 pattern 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.

Modules

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from tkinter import *
import numpy as np
import pandas as pd
import os
```

These are the imported libraries that are utilized to use various tools that are available in that specific library. tkinter is used to build a Graphical User Interface in Python.

```
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',
    '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']
```

L1 is the list made for various symptoms which are generally showed up in people for various diseases.

```
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)', 'Heart attack', 'Varicose veins',
       'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
       'Osteoarthristis', 'Arthritis',
       '(vertigo) Paroymsal  Positional Vertigo', 'Acne',
       'Urinary tract infection', 'Psoriasis', 'Impetigo']
```

Disease is the list made for different diseases which are for the most part appeared in different individuals.

```
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
```

First L2 is the vacant list made. At that point, equivalent to a number of diseases in list L1, L2 is appended in a number of zeroes.

```
df=pd.read_csv("C:/Users/Damodhar Sai Pesay/Desktop/training.csv")
DF= pd.read_csv('C:/Users/Damodhar Sai Pesay/Desktop/training.csv', index_col='prognosis')
df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reacti
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertensi
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid'
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuber
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins
    '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)
#df.head()
DF.head()
```

There is a CSV document containing diseases and symptoms, named training.csv, which is utilized to prepare the model. read_csv() function is utilized to store the information in the dataframe, named df. Utilizing replace() function, prognosis column that are the different diseases, it is replaced by the numbers from 0 to n-1, where n is the number of different diseases present in .csv record. head() function is utilized to print the initial five rows of the preparation dataframe.

Out[5]:

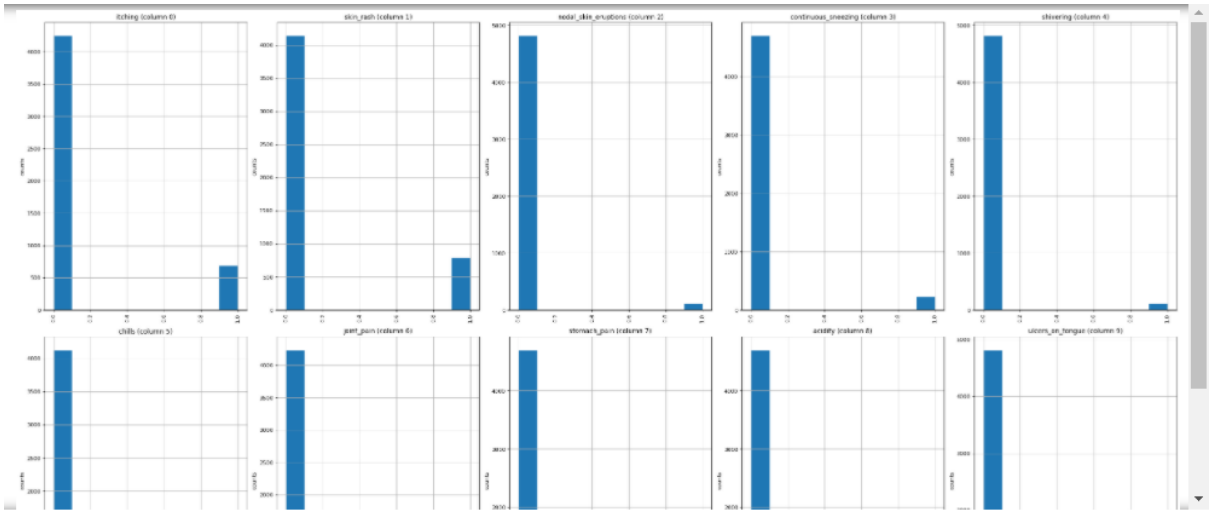
prognosis	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity
Fungal infection	1	1	1	0	0	0	0	0	0
Fungal infection	0	1	1	0	0	0	0	0	0
Fungal infection	1	0	1	0	0	0	0	0	0
Fungal infection	1	1	0	0	0	0	0	0	0
Fungal infection	1	1	1	0	0	0	0	0	0

5 rows × 132 columns

This is the output produced which contains the initial five rows of the dataframe df

```
def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]]
    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', edge
    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()
```

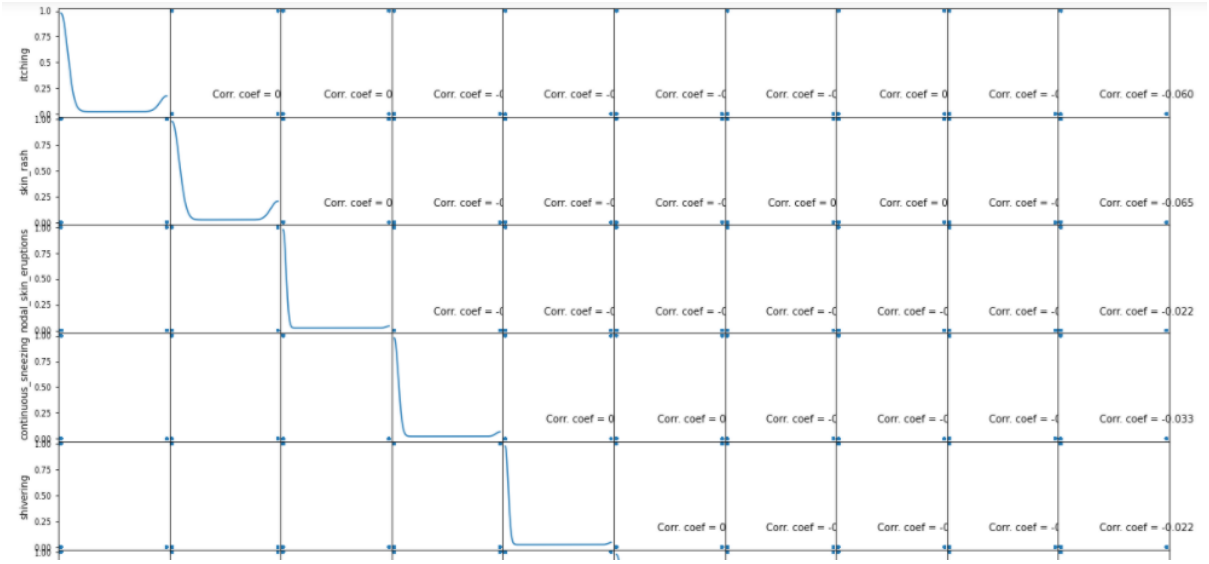
This is the code for the distribution graph of the columns of training.csv file.



Output for the distribution graph of the columns of training.csv file.

```
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number])

    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]]
    columnNames = list(df)
    if len(columnNames) > 10:
        columnNames = columnNames[:10]
    df1 = df1[columnNames]
    ax = pd.plotting.scatter_matrix(df1, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df1.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

This is the code for the scatter and the density plots of the columns of training.csv file



Output for the distribution graph of the columns of training.csv file.

```
X= df[l1]
y = df[["prognosis"]]
np.ravel(y)
print(X)
```

Putting the Symptoms in X and prognosis/diseases in y for training the model.

```
      back_pain  constipation  abdominal_pain  diarrhoea  mild_fever  \
0             0             0               0          0           0
1             0             0               0          0           0
2             0             0               0          0           0
3             0             0               0          0           0
4             0             0               0          0           0
...         ...           ...             ...        ...         ...
4915          0             0               0          0           0
4916          0             0               0          0           0
4917          0             0               0          0           0
4918          0             0               0          0           0
4919          0             0               0          0           0

      yellow_urine  yellowing_of_eyes  acute_liver_failure  fluid_overload  \
0                0                  0                    0               0
1                0                  0                    0               0
2                0                  0                    0               0
3                0                  0                    0               0
4                0                  0                    0               0
...            ...                ...                  ...             ...
4915             0                  0                    0               0
4916             0                  0                    0               0
4917             0                  0                    0               0
4918             0                  0                    0               0
4919             0                  0                    0               0
```

Output for the print(x) in which different symptoms has the values ‘0’ or ‘1’ according to their presence in the particular diseases.

```
print(y)
```

```
      prognosis
0             0
1             0
2             0
3             0
4             0
...         ...
4915         36
4916         37
4917         38
4918         39
4919         40

[4920 rows x 1 columns]
```

Output for the print(y) in which different disease has values according to their symptoms.

To build the precision of the model, we utilized four distinctive algorithms which are as per the following

- Decision Tree algorithm
- Random Forest algorithm
- KNearestNeighbour algorithm
- Naive Bayes algorithm

```
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
    print(x)
    print(y)
    plt.scatter(x,y)
    plt.show()
```

These are the function to plot the scatterplot of the predictions of the diseases and the symptoms entered by the user.

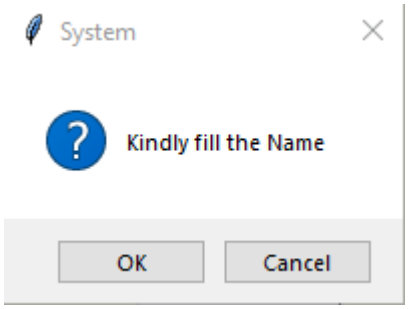
Decision Tree Function:

```
root = Tk()
pred1=StringVar()
def DecisionTree():
```

root=Tk() is used to for start working with the tkinter to build the GUI. Definition of DecisionTree() function. “pred1” is used to store the predicted disease using decision tree algorithm.

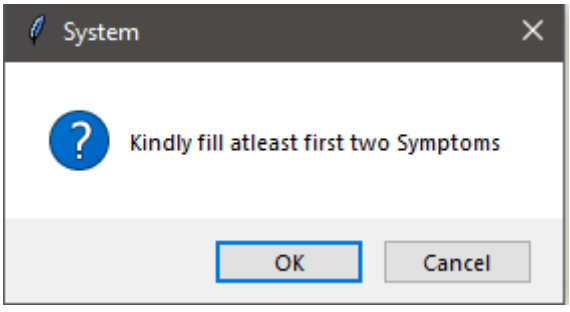
```
if len(NameEn.get()) == 0:
    pred1.set(" ")
    comp=messagebox.askokcancel("System","Kindly fill the Name")
    if comp:
        root.mainloop()
```

If user tries to run the GUI without entering the name, then system will prompt the following message.




```
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()
```

After filling the name, user have to fill five symptoms and out of which first two are compulsory. If user will not select atleast two symptoms, then following message will be prompt from the system



```
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)

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 = 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")
```

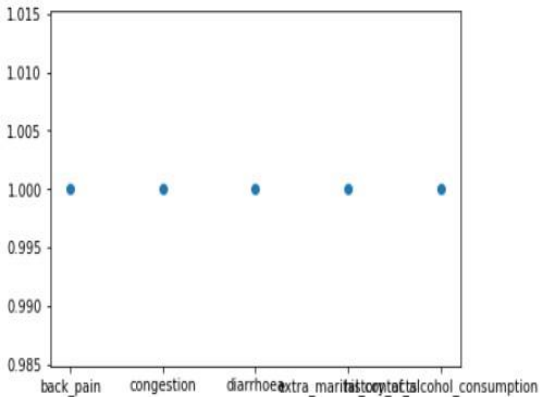
DecisionTreeClassifier() is used to train the model and predict the disease on testing dataset according to symptoms entered by the user. Final disease for decision tree is stored in a variable named “pred1”. Accuracy of predicting the disease is printed using accuracy score and confusion matrix is created using confusion matrix which are imported from sklearn metrices.

```
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 StringVar,S
c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.
conn.commit()
c.close()
conn.close()
```

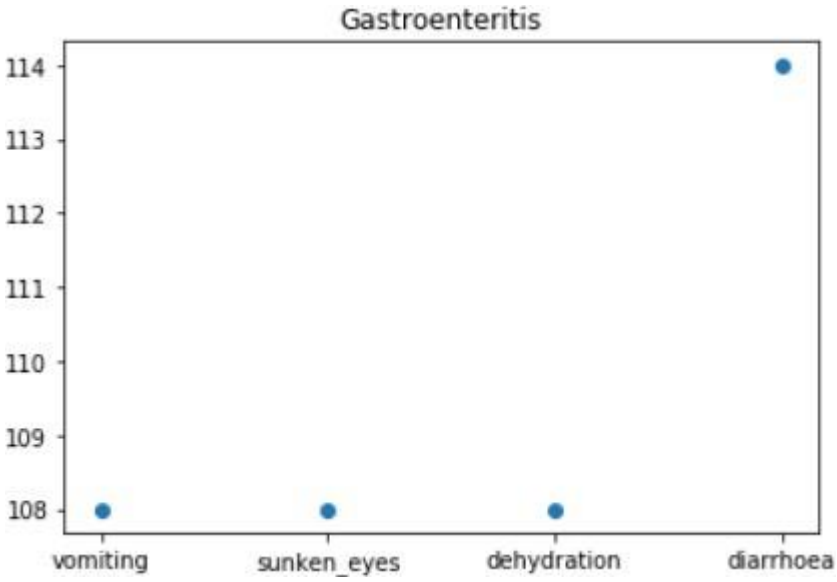
Creating a database named “database.db”, if not exists, using “sqlite3” database. For storing data for decision tree algorithm "DecisionTree" table is created, if not exists, in database.db using “CREATE” function in sqlite. Values are inserted in DecisionTree table using “INSERT” function in sqlite.

```
scatterinp(Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get())
scatterplt(pred1.get())
```

```
['back_pain', 'congestion', 'diarrhoea', 'extra_marital_contacts', 'history_of_alcohol_consumption']
[1, 1, 1, 1, 1]
```



The scatterplot for the symptoms which are given by the user as input



The scatter plot for the disease on the basis of symptoms which given by the user as input

Random Forest Function:

```
pred2=StringVar()
def randomforest():
```

Definition of randomforest() function. “pred2” is used to store the predicted disease using random forest algorithm.

```
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)

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'):
    pred2.set(" ")
    pred2.set(disease[a])
else:
    pred2.set(" ")
    pred2.set("Not Found")
```

RandomForestClassifier() is used to train the model and predict the disease on testing dataset according to symptoms entered by the user. Final disease for random forest is stored in a variable named “pred2”. Accuracy of predicting the disease is calculated using accuracy_score and confusion matrix is created using confusion_matrix which are imported from sklearn.metrices.

```
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,S
c.execute("INSERT INTO RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.
conn.commit()
c.close()
conn.close()
```

Same databse is used i.e.,databse.db that is used in decision tree algorithm with different table for random forest algorithm which is name as “RandomForest”.

KNearest Neighbour:

```
pred4=StringVar()
def KNN():
    if len(NameEn.get()) == 0:
```

Definition of KNN() function. “pred4” is used to store the predicted disease using kNearestNeighbour algorithm.

```
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("kNearest Neighbour")
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()]

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

inputtest = [l2]
predict = knn.predict(inputtest)
predicted=predict[0]

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


if (h=='yes'):
    pred4.set(" ")
    pred4.set(disease[a])
else:
    pred4.set(" ")
    pred4.set("Not Found")
```

KNeighboursClassifier() is used to train the model and predict the disease on testing dataset according to symptoms entered by the user. Final disease for kNearestNeighbour is stored in a variable named “pred4”. Accuracy of predicting the disease is calculated using accuracy_score and confusion matrix is created using confusion_matrix which are imported from sklearn.metrices

```
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name StringVar,Symtom1 StringVar,Symtom
c.execute("INSERT INTO KNearestNeighbour(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)
conn.commit()
c.close()
conn.close()
```

Same databse is used i.e.,databse.db that is used in all previous algorithms with different table for kNearest Neighbour algorithm which is name as “KNearestNeighbour”.

Naïve Bayes Function:

```
pred3=StringVar()
def NaiveBayes():
```

Defination of NaïveBayes() function. “pred3” is used to store the predicted disease using Naïve Bayes algorithm.

```
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()]
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'):
    pred3.set(" ")
    pred3.set(disease[a])
else:
    pred3.set(" ")
    pred3.set("Not Found")
```

GaussianNB() is used to train the model and predict the disease on testing dataset according to symptoms entered by the user. Final disease for Naïve Bayes is stored in a variable named "pred3". Accuracy of predicting the disease is calculated using accuracy_score and confusion matrix is created using confusion_matrix which are imported from sklearn.metrices.

```
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symtom1 StringVar,Symtom2 Strir
c.execute("INSERT INTO NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(
conn.commit()
c.close()
conn.close()
```

Same databse is used i.e.,databse.db that is used in all previous algorithms with different table for Naïve Bayes algorithm which is name as "NaiveBayes".

Building the Graphical User Interface

```
#Tk class is used to create a root window
root.configure(background='Ivory')
root.title('Smart Disease Predictor System')
root.resizable(0,0)
```

Graphical User Interface is build using tkinter library in Python. Root is used to start the GUI. It is configured with the background that is set to “Ivory”. GUI title is given as “Smart Disease Predictor System” using title() function in tkinter library. Resizable function is used to fix the size GUI.

```
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()
```

Here, variables are defined like Name, Symptom1, Symptom2, etc and they initialized to “Select Here” using set() function in tkinter library.

This is how the above variables are looking like in the GUI

```
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(" ")
    try:
        prev_win.destroy()
        prev_win=None
    except AttributeError:
```

It is the definition of the function “Reset()” which is used to reset the GUI inputs which are given by the user. It is called when user click on the button “Reset Inputs” from the GUI.

“Reset Inputs” button in GUI

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

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

It is the definition of the function “Exit()” which is used to come out from the GUI. It is called when user click on the button “Exit System” from the GUI.

“Exit System” button in GUI

```
#Headings for the GUI written at the top of GUI
w2 = Label(root, justify=LEFT, text="Disease Predictor using Machine Learning", fg="Red", bg="Ivory")
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2 = Label(root, justify=LEFT, text="Contributors: Damodhar,Surya,Phani,Karthik", fg="Pink", bg="Ivory"
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=2, column=0, columnspan=2, padx=100)
```

"w2" is the label created for showing the headings in the GUI using label() function from tkinter library. Two text are written under label w2 in row1 and row2 with font features as "Times", "30", "bold italic".

This is how w2 label is available in the GUI

```
#Label for the name
NameLb = Label(root, text="Name of the Patient *", fg="Red", bg="Ivory")
NameLb.config(font=("Times",15,"bold italic"))
NameLb.grid(row=6, column=0, pady=15, sticky=W)
```

"NameLb" is the label created for showing the "Name of the Patient *" using label() function in tkinter library. It is configured using config() function and the grid of the label is set using grid() function.

NameLb label in GUI. '*' shows that it is compulsory for the user to give his/her name.

```
#Creating Labels for the symtoms
S1Lb = Label(root, text="Symptom 1 *", fg="Black", bg="Ivory")
S1Lb.config(font=("Times",15,"bold italic"))
S1Lb.grid(row=7, column=0, pady=10, sticky=W)

S2Lb = Label(root, text="Symptom 2 *", fg="Black", bg="Ivory")
S2Lb.config(font=("Times",15,"bold italic"))
S2Lb.grid(row=8, column=0, pady=10, sticky=W)

S3Lb = Label(root, text="Symptom 3", fg="Black",bg="Ivory")
S3Lb.config(font=("Times",15,"bold italic"))
S3Lb.grid(row=9, column=0, pady=10, sticky=W)

S4Lb = Label(root, text="Symptom 4", fg="Black", bg="Ivory")
S4Lb.config(font=("Times",15,"bold italic"))
S4Lb.grid(row=10, column=0, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5", fg="Black", bg="Ivory")
S5Lb.config(font=("Times",15,"bold italic"))
S5Lb.grid(row=11, column=0, pady=10, sticky=W)
```

These are the labels for showing the symptoms of the disease. It is created using label() function from tkinter library. Its features are configured using config() function and their grid is set by using grid() function from tkinter library.

```
#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)
```

NameEn is the entry box created for getting the name of the patient using Entry() function in tkinter library. S1, S2, S3, S4, S5 are the option menu used to get symptoms from the user which is created using Optionmenu in tkinter library. *OPTIONS is the list of unique symptoms.

List of symptoms available for user

```
#Labels for the different algorithms
lrLb = Label(root, text="DecisionTree", fg="white", bg="red", width = 20)
lrLb.config(font=("Times",15,"bold italic"))
lrLb.grid(row=15, column=0, pady=10,sticky=W)

destreeLb = Label(root, text="RandomForest", fg="Red", bg="Orange", width = 20)
destreeLb.config(font=("Times",15,"bold italic"))
destreeLb.grid(row=17, column=0, pady=10, sticky=W)

ranfLb = Label(root, text="NaiveBayes", fg="White", bg="green", width = 20)
ranfLb.config(font=("Times",15,"bold italic"))
ranfLb.grid(row=19, column=0, pady=10, sticky=W)

knnLb = Label(root, text="kNearestNeighbour", fg="Red", bg="Sky Blue", width = 20)
knnLb.config(font=("Times",15,"bold italic"))
knnLb.grid(row=21, column=0, pady=10, sticky=W)
OPTIONS = sorted(l1)
```

These are the labels created for showing the texts of different algorithms

```
#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)
```

Buttons created for predicting the disease using different algorithms

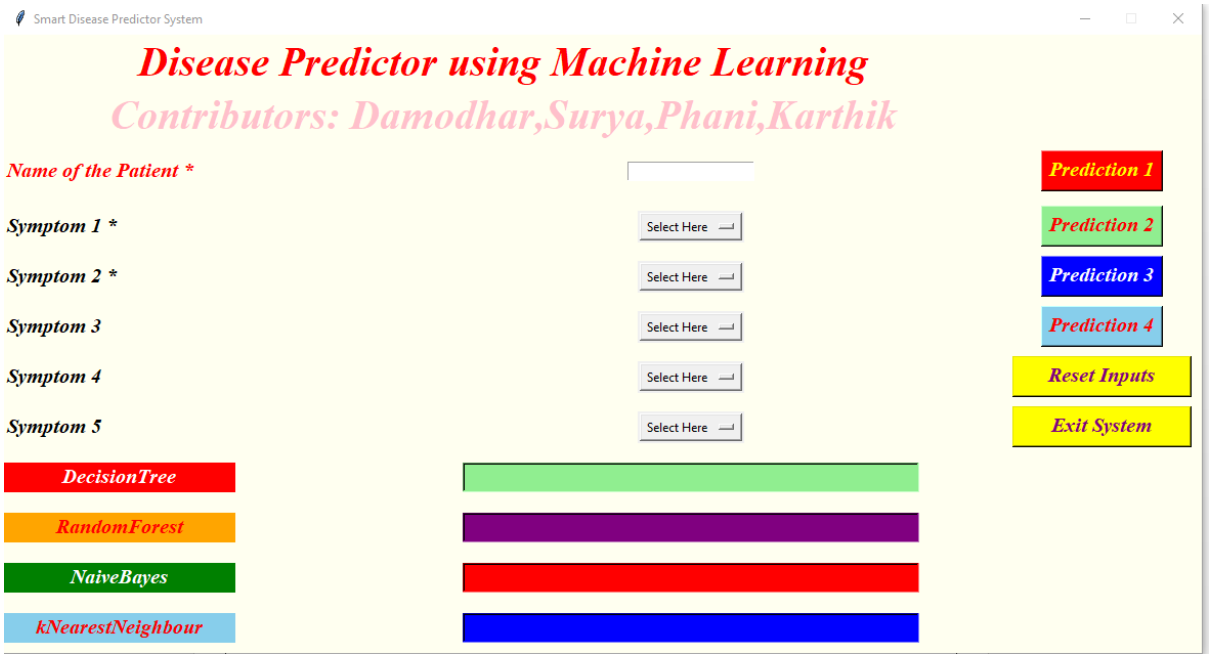
- Press Prediction 1 for Decision tree algorithm.
- Press Prediction 2 for Random Forest algorithm.
- Press Prediction 3 for Naïve Bayes algorithm.
- Press Prediction 4 for K-Nearest Neighbour algorithm.
- Press Reset Inputs for resetting the inputs
- Press Exit System for Exiting from the System


```
#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)
```

These are label created for showing the predicted disease using different algorithm.

```
#calling this function because the application is ready to run
root.mainloop()
```

This is calling the GUI.



The Final GUI presented to the user.

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. Our system also has an easy to use interface. It also has various visual representation of data collected and results achieved.

References:

1. http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html
2. https://ieeexplore.ieee.org/document/8819782
3. https://link.springer.com/chapter/10.1007/978-981-15-5089-8_55#:~:text=Machine%20Learning%20is%20an%20emerging,dataset%20and%20predict%20the%20disease