A PROJECT REPORT ON

**DISEASE PREDICTION USING**

**MACHINE LEARNING**

IV SEMESTER PROJECT

B.TECH IN COMPUTER

SCIENCE AND ENGINEERING

**Submitted by:**

Sagar Negi

University Roll No: 2016976

Batch: 2020-2024

**Under the Supervision of:**

Dr.Manoj Diwakar

Associate Professor

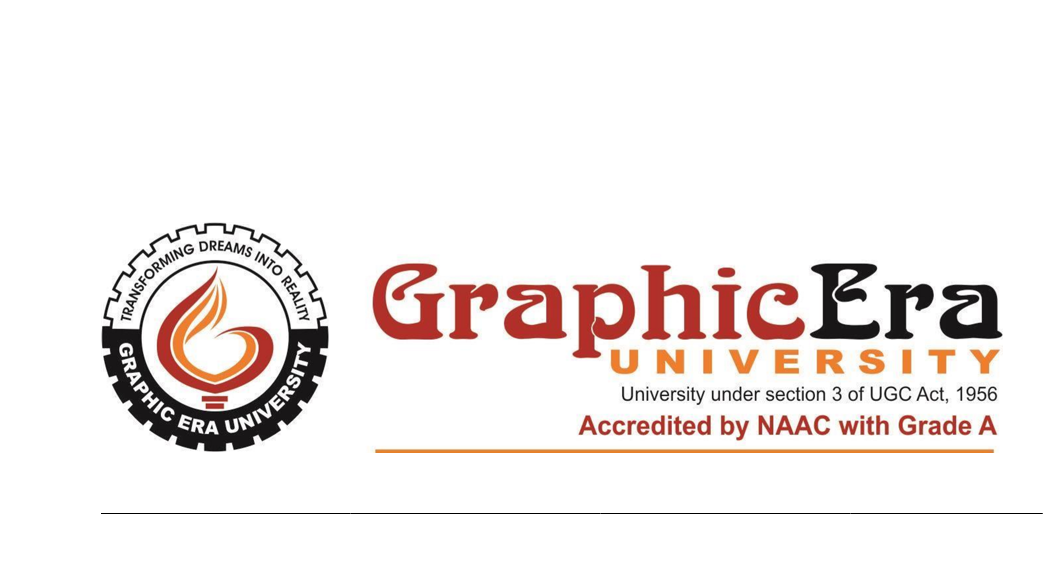
****

**DEPARTMENT OF**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA DEEMED TO BE UNIVERSITY

DEHRADUN, UTTARAKHAND

****

**ACKNOWLEDGEMENT:**

I would like to take this opportunity to express my gratitude to entire faculty at Department of Computer Science and Engineering, Graphic Era Deemed to be University, who evaluated the project from time to time and gave me valuable suggestions as to how to improve the project.

I am grateful to Dr.Manoj Diwakar, Associate Professor, for his supervision, encouragement, inspiration and guidance. Working under him is being an enriched experience. In all, I found congenial work environment in Graphic Era Deemed to be University and this project completion will mark a new beginning for me in the coming days.

I am highly indebted to Graphic Era Deemed to be University for providing me the required infrastructure and facilities to accomplish the given task.

Sagar Negi  
B.Tech CSE  
2020-2024  
Graphic Era University

**CONTENTS**

* **INTRODUCTION**
* **PROBLEM STATEMENT**
* **AIM**
* **METHODOLOGY**
* **LIBRARIES USED**
* **MODELS USED**
* **CODE**
* **IMAGES FROM THE PROJECT**
* **CONCLUSION**
* **REFERENCES**

**DISEASE PREDICTION SYSTEM**

**INTRODUCTION:**

This report is about the project chosen i.e., DISEASE PREDICTION SYSTEM USING MACHINE LEARNING. This report tries to give a brief overview of the objective of the project, Models and the libraries used, code, application and the Methodology followed to achieve this project.

**PROBLEM STATEMENT:**

Health information needs are 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, it will save a lot of time. In the current 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. 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. So in order to increase the efficiency of the current medical system by reducing the workload on them, we have designed a disease prediction System.

**AIM:**

To implement a Disease Prediction System that can efficiently predict the disease of a human, based on the symptoms that he/she possess, using four different algorithms which will enhance the working of the medical System.

**METHODOLOGY:**

Prediction of disease by looking at the symptoms is an integral part of treatment. In our project, DISEASE PREDICTION USING MACHINE LEARNING, we have tried to predict the 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.

**DATA COLLECTION:**

Data Collection Dataset for this project was collected from a study of university of Columbia performed at New York Presbyterian Hospital during 2004.

**LIBRARIES 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.

1. **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 in 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.

1. **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.

1. **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 USED:**

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

• Decision tree

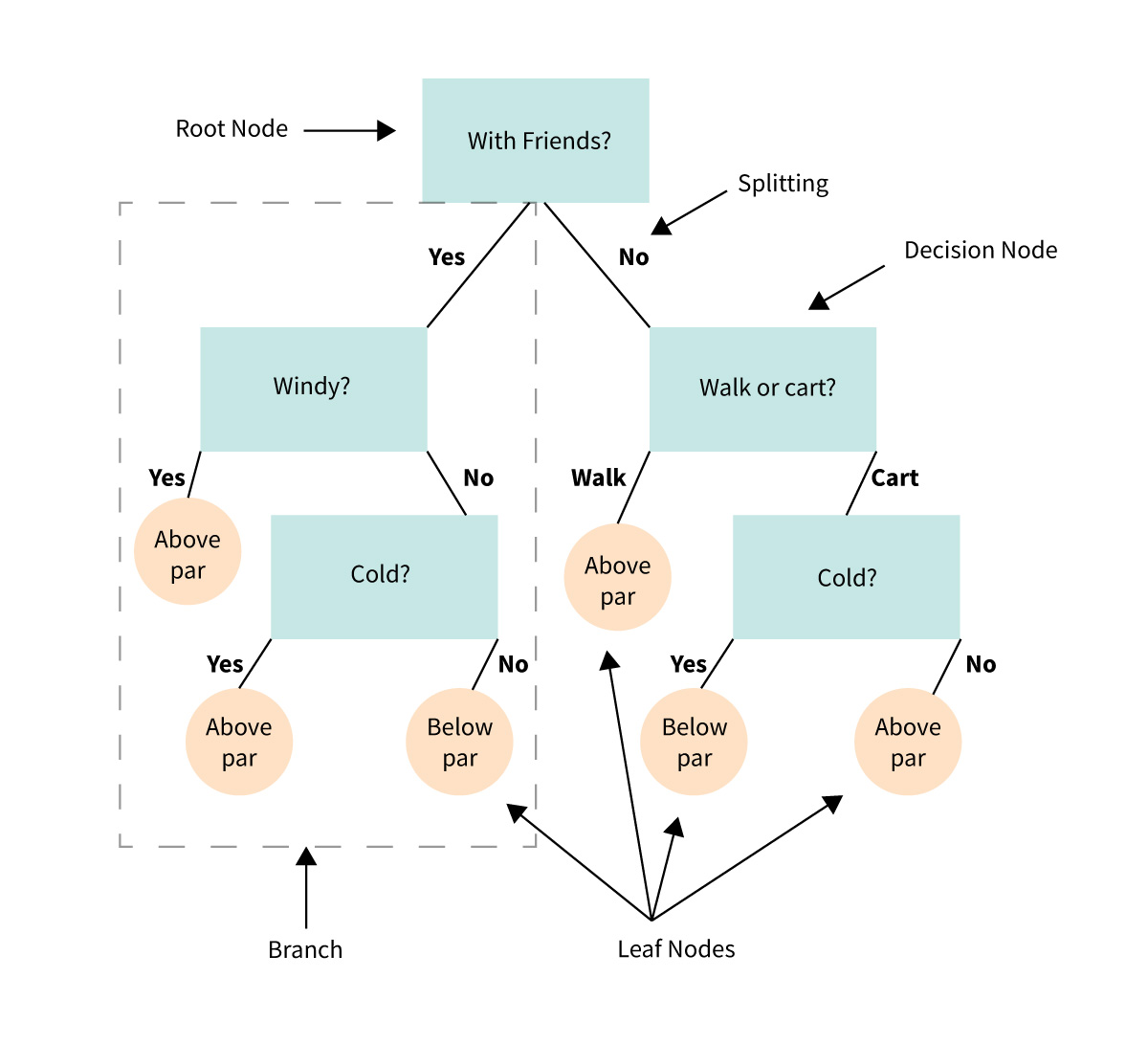
• Random forest tree

• Gaussian Naïve Bayes

• KNN

1. **Decision tree:** It 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%.



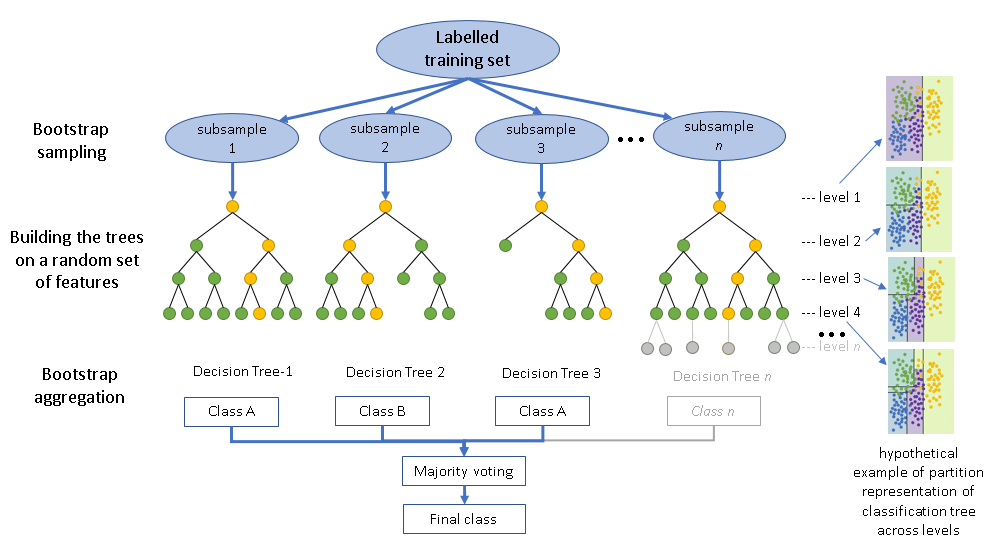
1. **Random Forest Algorithm**: It is a supervised learning algorithm used for both classification and regression. This algorithm works on 4 basic steps –   
    1. It chooses random data samples from dataset.

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.

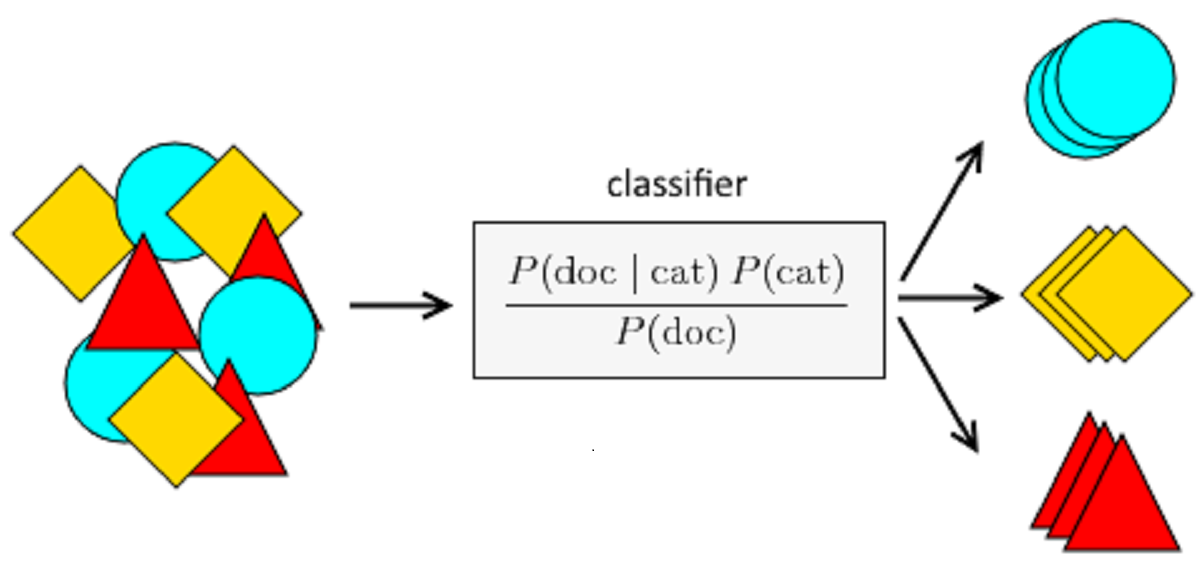
In this project we have used random forest classifier with 100 random samples and the result given is ~95% accuracy.



1. **K Nearest Neighbour:** It 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.



1. **Naïve Bayes algorithm**: It 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.



**CODE:**

**#Importing Libraries from matplotlib to visualize the data**

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

**#Importing Libraries to create GUI**

from tkinter import \*

**#Importing Libraries to perform calculations**

import numpy as np

import pandas as pd

import os

**#List of the symptoms is listed here in list l1.**

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']

**#List of Diseases is listed in list disease.**

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 i in range(0,len(l1)):

l2.append(0)

print(l2)

**#Reading the training.csv file**

df=pd.read\_csv("training.csv")

**#Replace the values in the imported file by pandas by the inbuilt function replace in pandas.**

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)

**#printing the top 5 rows of the training dataset**

df.head()

**# 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 between 1 and 50 unique values**

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()

**# Scatter and density plots**

def plotScatterMatrix(df1, plotSize, textSize):

df1 = df1.select\_dtypes(include =[np.number]) # keep only numerical columns

**# Remove rows and columns that would lead to df being singular**

df1 = df1.dropna('columns')

df1 = df1[[col for col in df if df[col].nunique() > 1]] **# keep columns where there are more than 1 unique values**

columnNames = list(df)

if len(columnNames) > 10: **# reduce the number of columns for matrix inversion of kernel density plots**

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='center', va='center', size=textSize)

plt.suptitle('Scatter and Density Plot')

plt.show()

plotScatterMatrix(df, 20, 10)

X= df[l1]

y = df[["prognosis"]]

np.ravel(y)

print(X)

print(y)

**#Reading the testing.csv file**

tr=pd.read\_csv("testing.csv")

**#Using inbuilt function replace in pandas for replacing the values**

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)

**#printing the top 5 rows of the testing data**

tr.head()

plotScatterMatrix(tr, 20, 10)

X\_test= tr[l1]

y\_test = tr[["prognosis"]]

np.ravel(y\_test)

print(X\_test)

print(y\_test)

#list1 = DF['prognosis'].unique()

def scatterplt(disea):

x = ((DF.loc[disea]).sum()) **#total sum of symptom reported for given disease**

x.drop(x[x==0].index,inplace=True) **#dropping symptoms with values 0**

print(x.values)

y = x.keys() **#storing names of symptoms in y**

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] **#storing input symptoms in y**

y = [0,0,0,0,0] **#creating and giving values to the input symptoms**

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()

**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:

print(NameEn.get())

from sklearn import tree

clf3 = tree.DecisionTreeClassifier()

clf3 = clf3.fit(X,y)

from sklearn.metrics import classification\_report,confusion\_matrix,accuracy\_score

y\_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")

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,Symtom4 TEXT,Symtom5 TEXT,Disease StringVar)")

c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred1.get()))

conn.commit()

c.close()

conn.close()

**#printing scatter plot of input symptoms**

**#printing scatter plot of disease predicted vs its symptoms** scatterinp(Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get())

scatterplt(pred1.get())

**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)

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")

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,Symtom4 TEXT,Symtom5 TEXT,Disease StringVar)")

c.execute("INSERT INTO RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred2.get()))

conn.commit()

c.close()

conn.close()

**#printing scatter plot of disease predicted vs its symptoms**

scatterplt(pred2.get())

**KNN Algorithm**

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()]

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")

import sqlite3

conn = sqlite3.connect('database.db')

c = conn.cursor()

c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease StringVar)")

c.execute("INSERT INTO KNearestNeighbour(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred4.get()))

conn.commit()

c.close()

conn.close()

**#printing scatter plot of disease predicted vs its symptoms**

scatterplt(pred4.get())

**Naïve Bayes Algorithm**

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()]

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")

import sqlite3

conn = sqlite3.connect('database.db')

c = conn.cursor()

c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease StringVar)")

c.execute("INSERT INTO NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get(),pred3.get()))

conn.commit()

c.close()

conn.close()

**#printing scatter plot of disease predicted vs its symptoms**

scatterplt(pred3.get())

**#Tk class is used to create a root window**

root.configure(background='Lightgrey')

root.title('Disease Prediction System')

root.resizable(0,0)

**#taking first input as symptom**

Symptom1 = StringVar()

Symptom1.set("Select Here")

**#taking second input as symptom**

Symptom2 = StringVar()

Symptom2.set("Select Here")

**#taking third input as symptom**

Symptom3 = StringVar()

Symptom3.set("Select Here")

**#taking fourth input as symptom**

Symptom4 = StringVar()

Symptom4.set("Select Here")

**#taking fifth input as symptom**

Symptom5 = StringVar()

Symptom5.set("Select Here")

Name = StringVar()

**#function to Reset the given inputs to initial position**

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:

pass

**#Exit button to come out of system**

from tkinter import messagebox

def Exit():

qExit=messagebox.askyesno("System","Do you want to exit the system")

if qExit:

root.destroy()

exit()

**#Headings for the GUI written at the top of GUI**

w2 = Label(root, justify=LEFT, text="Disease Prediction System using Machine Learning", fg="Black", bg="Lightgrey")

w2.config(font=("Times",30,"bold"))

w2.grid(row=2, column=0, columnspan=4, padx=100)

**#Label for the name**

NameLb = Label(root, text="Name of the Patient", fg="Red", bg="Lightgrey")

NameLb.config(font=("Times",15,"bold"))

NameLb.grid(row=6, column=0, pady=15,padx=50, sticky=W)

**#Creating Labels for the symptoms**

S1Lb = Label(root, text="Symptom 1", fg="Black", bg="Lightgrey")

S1Lb.config(font=("Times",15,"bold"))

S1Lb.grid(row=7, column=0, pady=10,padx=60, sticky=W)

S2Lb = Label(root, text="Symptom 2", fg="Black", bg="Lightgrey")

S2Lb.config(font=("Times",15,"bold"))

S2Lb.grid(row=8, column=0, pady=10,padx=60, sticky=W)

S3Lb = Label(root, text="Symptom 3", fg="Black",bg="Lightgrey")

S3Lb.config(font=("Times",15,"bold"))

S3Lb.grid(row=9, column=0, pady=10,padx=60, sticky=W)

S4Lb = Label(root, text="Symptom 4", fg="Black", bg="Lightgrey")

S4Lb.config(font=("Times",15,"bold"))

S4Lb.grid(row=10, column=0, pady=10,padx=60, sticky=W)

S5Lb = Label(root, text="Symptom 5", fg="Black", bg="Lightgrey")

S5Lb.config(font=("Times",15,"bold"))

S5Lb.grid(row=11, column=0, pady=10,padx=60, sticky=W)

#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,padx=30,sticky=W)

destreeLb = Label(root, text="RandomForest", fg="Black", bg="Yellow", width = 20)

destreeLb.config(font=("Times",15,"bold italic"))

destreeLb.grid(row=17, column=0, pady=10,padx=30, 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,padx=30, sticky=W)

knnLb = Label(root, text="kNearestNeighbour", fg="Black", bg="Sky Blue", width = 20)

knnLb.config(font=("Times",15,"bold italic"))

knnLb.grid(row=21, column=0, pady=10,padx=30, sticky=W)

OPTIONS = sorted(l1)

**#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)

**#Buttons for predicting the disease using different algorithms**

dst = Button(root, text="Prediction 1", command=DecisionTree,bg="Red",fg="White")

dst.config(font=("Times",15,"bold italic"))

dst.grid(row=6, column=3,padx=10)

rnf = Button(root, text="Prediction 2", command=randomforest,bg="Yellow",fg="Black")

rnf.config(font=("Times",15,"bold italic"))

rnf.grid(row=7, column=3,padx=10)

lr = Button(root, text="Prediction 3", command=NaiveBayes,bg="Green",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="Black")

kn.config(font=("Times",15,"bold italic"))

kn.grid(row=9, column=3,padx=10)

rs = Button(root,text="Reset Inputs", command=Reset,bg="Black",fg="White")

rs.config(font=("Times",15,"bold italic"))

rs.grid(row=10,column=3,padx=10)

ex = Button(root,text="Exit System", command=Exit,bg="Black",fg="White")

ex.config(font=("Times",15,"bold italic"))

ex.grid(row=11,column=3,padx=10)

**#Showing the output of different algorithms**

t1=Label(root,font=("Times",15,"bold italic"),text="Decision Tree",height=1,bg="Black"

,width=40,fg="White",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="Black"

,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="Black"

,width=40,fg="White",textvariable=pred3,relief="sunken").grid(row=19, column=1, padx=10)

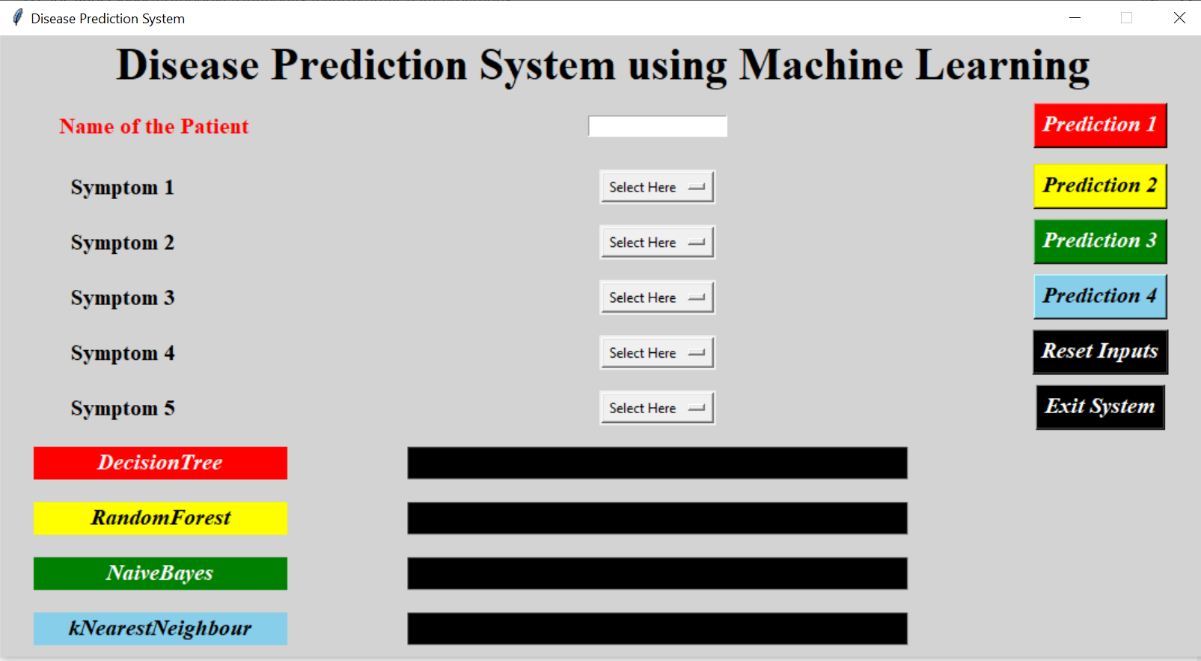
t4=Label(root,font=("Times",15,"bold italic"),text="kNearest Neighbour",height=1,bg="Black"

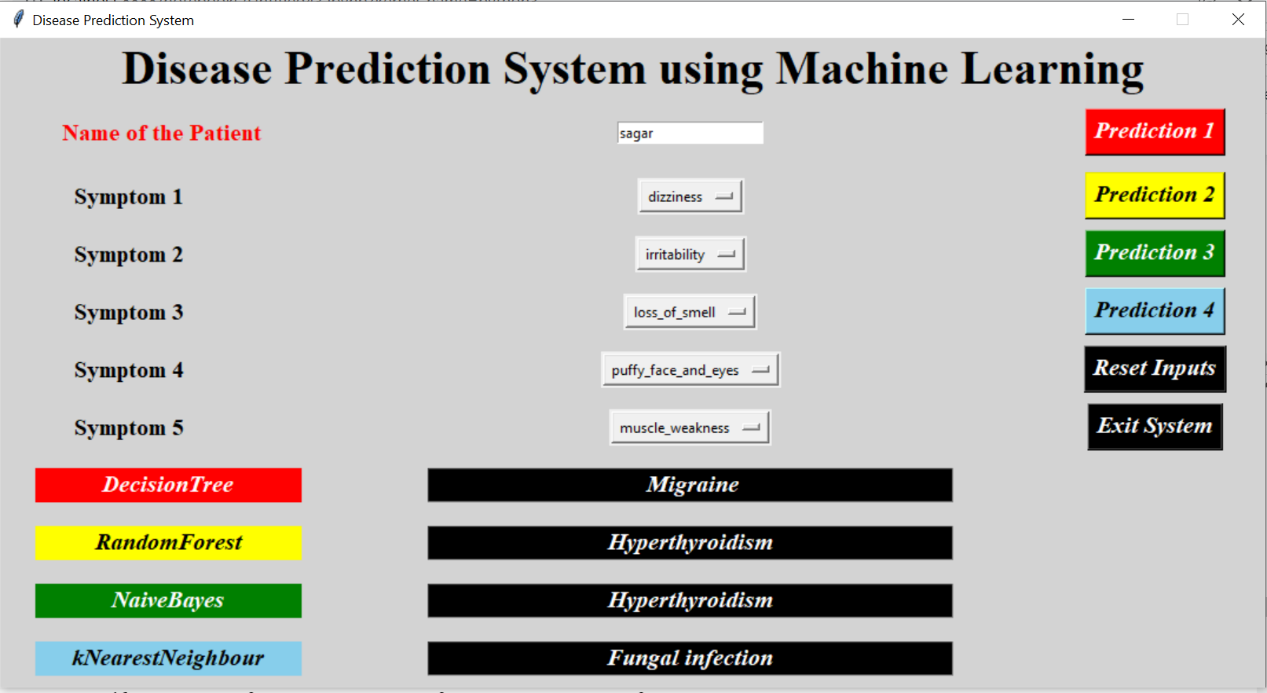
,width=40,fg="White",textvariable=pred4,relief="sunken").grid(row=21, column=1, padx=10)

**#calling this function because the application is ready to run**

root.mainloop()

**IMAGES FROM THE PROJECT:**

****

****

**CONCLUSION:**

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

**REFERENCES:**

♣ columbia.edu (Disease prediction Database)

♣ Google

♣ YouTube