DISEASE PREDICTION FROM SYMPTOMS USING ML

A PROJECT REPORT

Submitted by

AYUSH GUPTA
(21BHI10055)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE (Spl. In Health Informatics)

VIT®
BHOPAL
www.vitbhopal.ac.in

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRI KALAN, SEHORE
MADHYA PRADESH - 466114

SEPTEMBER, 2022

VIT BHOPAL UNIVERSITY, KOTHRI KALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

This project report titled **"DISEASE PREDICTION FROM SYMPTOMS USING ML"** is the bonafide work of **"AYUSH GUPTA""(21BHI10055)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Swagat Kumar Samantaroy,
Assistant Professor Sr-1
School of Computing Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Swagat Kumar Samantaroy,
Assistant Professor Sr-1
School of Computing Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 30th September 2022.

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr.S.Poongundran, Head of the Department, School of Computing Science and Engineering for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Mr. Swagat Kumar Samantroy, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

INDEX

ABSTRACT

Evolution of modern technologies like data science and machine learning has opened the path for healthcare communities and medical institutions, to detect the diseases as earliest as possible and it helps to provide better patient care. Our body shows the symptoms when something wrong is happening within our body, sometimes it may be just a minor problem but sometimes we can have severe illness and if we do not take care of these symptoms at the early stage then it might be too late to cure the disease. So we are proposing a disease prediction system that can predict the possible diseases based on symptoms so it can be cured at the early stage. It saves time that is required to do the complete diagnosis of the patient and based on the suggestions provided by the system we can only get the patient diagnosed for those diseases that are required.

In this project, we are using machine learning algorithms that try to accurately predict possible diseases. The results generated by the proposed system have an accuracy of up to 87%. The system has incredible potential in anticipating the possible diseases more precisely.

INTRODUCTION

It is a system that is made by the use of machine learning algorithms for predicting the possible diseases based on the patient's symptoms. The growth of technology has been improving our lives so far. It provides many tools that can save millions of lives, and machine learning is one of them. Machine Learning is used to develop systems that can help us predict so many diseases based on symptoms. It can suggest to the doctor, probability of possible diseases. And diagnosis can be done based on suggestion, thus cost could be reduced.
We are living in the age of technology and nowadays humans can say that almost anything is possible with the help of technology. Today we have so many tools and methods to access information from any region of this world and Information at this age is so important that without information we would not survive. We have tools that can give us or suggest relevant information at our fingertips and the internet is one of those tools. Today billions of search queries are performed daily and sometimes their given results are relevant and sometimes they are not. In those search queries, thousands of searches are related to medical advice. People often want to know if they have any serious diseases based on their signs and symptoms. But there are no tools available to give them proper information. This project tries to give them tools so that possible disease prediction information can be provided to the end-user.

"Disease Prediction" system based on predictive modeling predicts the disease of the user on the basis of the symptoms that the user provides as an input to the system. The system analyzes the symptoms provided by the user as input and gives the probability of the disease as an output. Disease prediction is done by implementing four techniques such as Naïve Bayes, KNN, Decision Tree, and Random Forest Algorithms. These techniques calculate the probability of the disease. Therefore, average prediction accuracy probability 87% is obtained.

MOTIVATION OF WORK

A key challenge confronting healthcare organizations (hospitals, medical centers) is the facility of quality services at reasonable prices. Quality amenities suggest diagnosing patients accurately and regulating medications that are effective. Poor clinical choices can prompt deplorable results, which are in this manner unsatisfactory. Hospitals should limit the cost of clinical tests. They can accomplish these outcomes by utilizing fitting PC based data and additionally choosing emotionally supportive networks.

DATASET AND MODEL DESCRIPTION

In our proposed system we are using structured datasets that can be created by collecting patient's symptoms and diagnosis from local hospitals and from open source libraries available online. We are using true datasets that give higher accuracy. In the proposed system we utilize machine learning algorithms to predict diseases based on the patient's symptoms.

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 a 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
- Message box

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

2. Numpy: Numpy is the 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 arrays. 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 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 analyzed with 2 ways: Series and Data frames

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

Data frames are two-dimensional data structures used in python to store data consisting of rows and columns.
Pandas data frame is used extensively in this project to use datasets required for training and testing the algorithms. Data frames make 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 algorithms 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
- Gaussian Naive Bayes
- KNN

Decision trees are classified as a very effective and versatile classification technique. It is used in pattern recognition and classification for images. 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 consist of attributes which have the most effect on the outcome, leaf tests for the value of a certain attribute and leaf gives out the output of the tree.

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

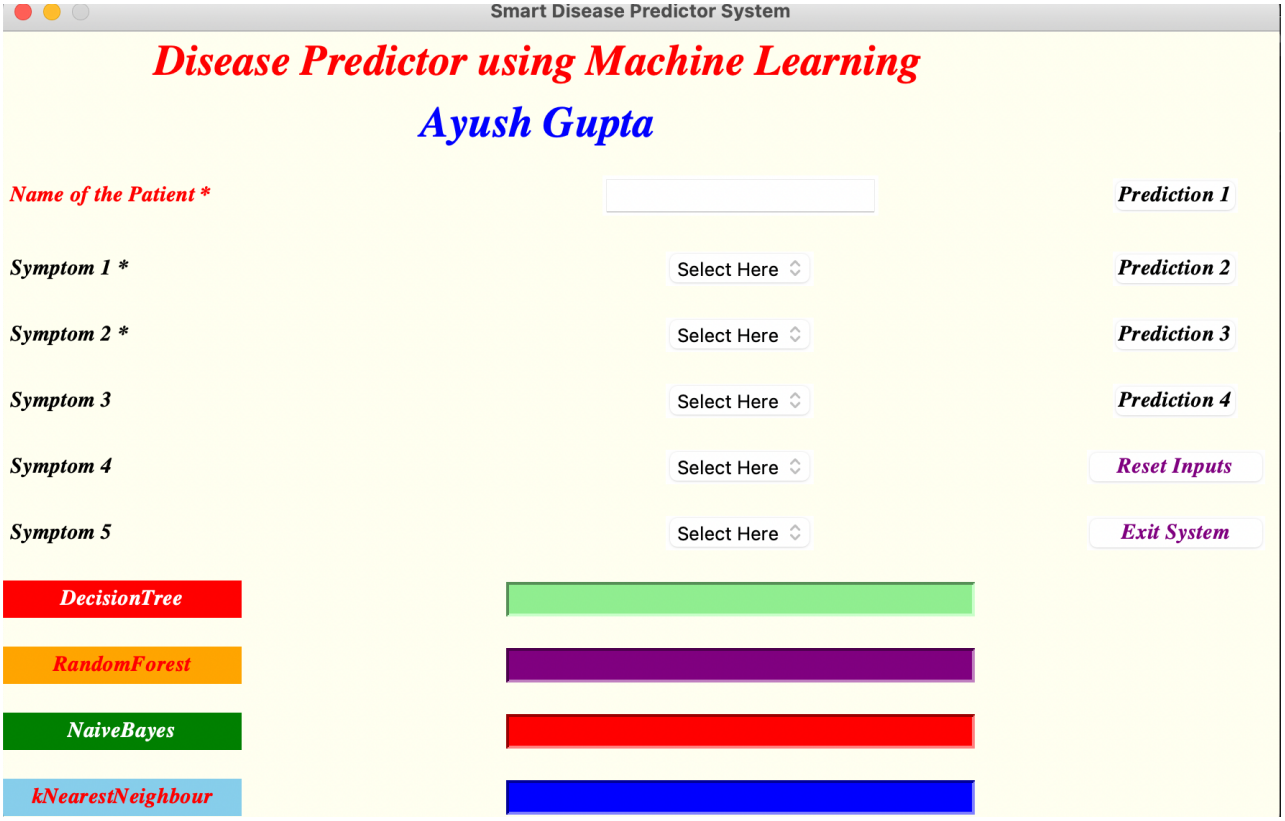
- It chooses random data samples from the dataset.
- It constructs decision trees for every sample dataset chosen.
- At this step every predicted result will be compiled and voted on.
- At last most voted prediction will be selected and be presented as a result of classification.

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.

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

GRAPHIC USER INTERFACE

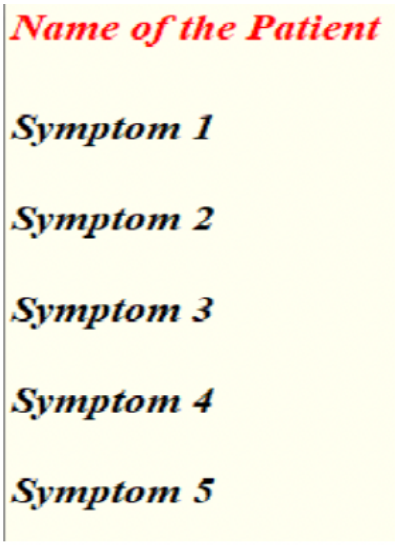
The GUI made for this project is a simple tkinter GUI consisting of labels, message box, button, text, title and option menu.



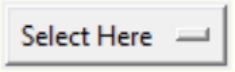
Root.title() is used to set the title as Smart Disease Predictor System.

Label is used to add heading and contributors section.

Labels are further used for different sections.



OptionMenu is used to create a drop down menu.



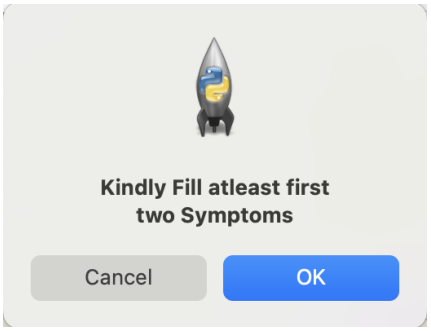
Buttons are used to give functionalities and predict the outcome of models. two utility buttons namely exit and rest are also created.

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

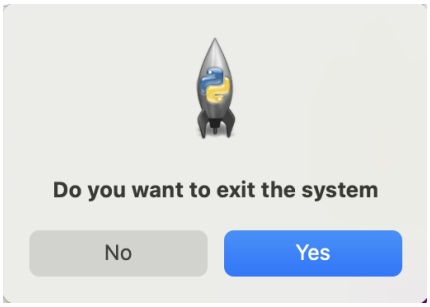
Message box are used at three different places:

1.to restrain the to enter name.

2. to ask for at least two symptoms,



3. to confirm the exit system.

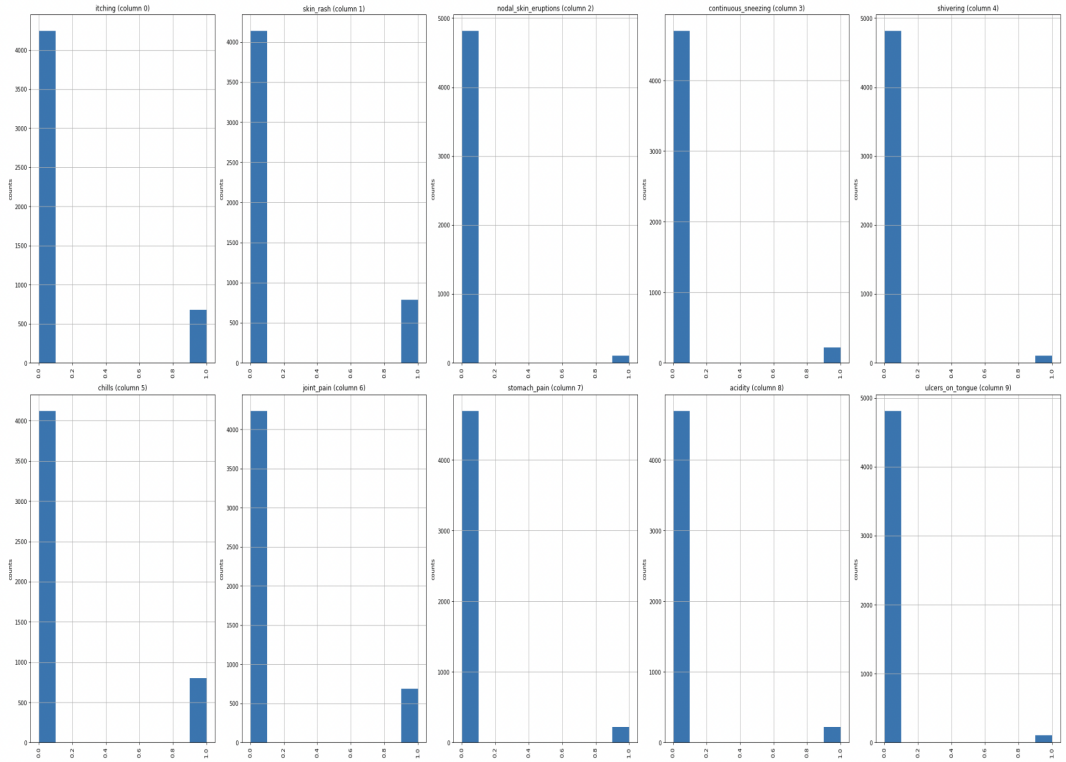


MODULES DESCRIPTION

Functions like plot percolumb distribution() plotScatterMatrix() is used to visualize the data.

```python
# 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
    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()
```

Python



Functions like scatterplot and scattering are used to compare input to training data.

```
#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)#droping symptoms with values 0
    print(x.values)
    y = x.keys()#storing nameof symptoms in y
    print(len(x))
    print(len(y))
    plt.title(disea)
    plt.scatter(y,x.values)
    plt.show()
```

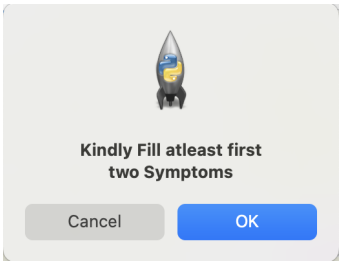
```
def scatterinp(sym1,sym2,sym3,sym4,sym5):
    x = [sym1,sym2,sym3,sym4,sym5]#storing input symptoms in y
    y = [0,0,0,0,0]#creating and giving values to the input symptoms
    if(sym1!='Select Here'):
        y[0]=1
    if(sym2!='Select Here'):
        y[1]=1
    if(sym3!='Select Here'):
        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:
        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]
```

Root=Tk() is used to start working with the tkinter to build the gui. Definition of DecisionTree() function. “pred1” is used to store the predicted disease using a decision tree algorithm.
If the user tries to run the gui without entering the name, then System will prompt the following message.

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



DecisionTreeClassifier() is used to train the model and predict the disease on a testing dataset according to symptoms entered by the user. Final disease form 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.

```
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 using sqlite3
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,Dise
c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2
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())
```

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

RANDOM FOREST ALGORITHM

“pred2” is used to store the predicted disease using a random forest algorithm.

```
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")
 #Creating the database if not exists named as database.db and creating table if not exists named as RandomForest using sqlite3
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(),Symptor
conn.commit()
c.close()
conn.close()
#printing scatter plot of disease predicted vs its symptoms
scatterplt(pred2.get())
```

```
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]
```

RandomForestClassifier() is used to train the model and predict the disease on a 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.metrics
Same database 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”.

K NEAREST NEIGHBOR ALGORITHM

“pred4” is used to store the predicted disease using kNearestNeighbour algorithm.

KNeighboursClassifier() is used to train the model and predict the disease on a 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.metrics.

```
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("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
```

Same database 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”

```
    if (h=='yes'):
        pred4.set(" ")
        pred4.set(disease[a])
    else:
        pred4.set(" ")
        pred4.set("Not Found")
    #Creating the database if not exists named as database.db and creating table if not exists named as KNearestNeighbour using sqlite3
    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(),
    conn.commit()
    c.close()
    conn.close()
    #printing scatter plot of disease predicted vs its symptoms

    scatterplt(pred4.get())
```

NAIVE BAYES ALGORITHM

“pred3” is used to store the predicted disease using 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]
```

GaussianNB() is used to train the model and predict the disease on a 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.metrics.

Same database 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' '.

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

```
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")
 #Creating the database if not exists named as database.db and creating table if not exists named as NaiveBayes using sqlite3
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
conn.commit()
c.close()
conn.close()
#printing scatter plot of disease predicted vs its symptoms
scatterplt(pred3.get())
```

Code of GUI to set initial values of labels.

```
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 message box.

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

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

CONCLUSION

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 used 4 different algorithms to do so. On an average we achieved accuracy of ~87%. 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 a better version of such a system. Our system also has an easy to use interface. It also has various visual representations of data collected and results achieved.

LITERATURE REVIEW

There have been numerous studies done related to predicting the disease using different machine learning techniques and algorithms which can be used by medical institutions. This project reviews some of those studies done in research papers using the techniques and results used by them.

MIN CHEN et al, proposed a disease prediction system in his paper where he used machine learning algorithms. In the prediction of disease, he used techniques like CNN-UDRP algorithm, CNN-MDRP algorithm, Naive Bayes, K-Nearest Neighbor, and Decision Tree. This proposed system had an accuracy of 94.8%.

Lambodar Jena et al, focused on risk prediction for chronic diseases by taking advantage of distributed machine learning classifiers and used techniques like

Naive Bayes and Multilayer Perceptron. This paper tries to predict Chronic-Kidney-Disease and the accuracy of Naïve Bayes and Multilayer Perceptron is 95% and 99.7% respectively

Dhiraj Dahiwade et al, designed a model for prediction of the disease using approaches of machine learning and used techniques like KNN and CNN. This paper suggests disease prediction i.e. based on the patient's symptoms. The accuracy of KNN is 95% and the accuracy of CNN is 98%.

Aditi Gavhane et al, suggested prediction for heart disease that utilizes Machine Learning. A Multi-Layer Perceptron model is used in this system. This system predicts heart disease based on basic symptoms like age, sex, pulse rate, etc. The accuracy of this suggested system is 91%.

Rahulpreet Singh Kohli et al, suggested disease prediction by using applications and methods of machine learning and used techniques like Logistic Regression, Decision Tree, Support Vector Machine, Random Forest and Adaptive Boosting. This paper focuses on predicting Heart disease, Breast cancer, and Diabetes. The highest accuracies are obtained using Logistic Regression that is 95.71% for Breast cancer, 84.42% for Diabetes, and 87.12% for Heart disease.