A PROJECT REPORT

on

"MULTIPLE DISEASE PREDICTION USING ML"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
INFORMATION TECHNOLOGY

BY

ADITI YADAV	2006157
RISHIKA SINGH	2006187
SHIVANGI KUMARI	2006287
KASHISH KAUR	2006467

UNDER THE GUIDANCE OF
ASST.PROF. NALINIPRAVA BEHERA

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

May 2023

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

“MULTIPLE DISEASE PREDICTION USING ML“

submitted by

ADITI YADAV	2006157
RISHIKA SINGH	2006187
SHIVANGI KUMARI	2006287
KASHISH KAUR	2006467

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in Information Technology at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2023, under my guidance.

Date: 04 /05 /23

Asst.Prof. Naliniprava Behera
Project Guide

Acknowledgement

We are profoundly grateful to **Asst.Prof. Naliniprava Behera** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

Aditi Yadav (2006157)
Rishika Singh (2006187)
Shivangi Kumari (2006287)
Kashish Kaur (2006467)

ABSTRACT

ABSTRACT

Multiple disease detection using machine learning is a rapidly growing area of research in healthcare. The availability of large datasets and advances in machine learning algorithms have enabled accurate and efficient detection of multiple diseases simultaneously. This report presents a minor project focused on the development of a machine learning-based approach for the detection of multiple diseases.

The proposed system utilizes a dataset of collected from a study by the University of Columbia performed at New York Presbyterian Hospital in 2004. The first column shows the disease, the second the number of discharge summaries containing a positive and current mention of the disease, and the associated symptom. We successfully created such a system, which uses four different techniques namely, Decision Tree, Random Forest, KNN, Naive Bayes and achieved an average accuracy of approximately 95%. In addition to accurate disease prediction, our system features a user-friendly interface and offers various visual representations of the collected data and results obtained. Our objective was to develop a system that can predict diseases based on the symptoms provided, thereby reducing the burden on hospital OPDs and medical staff.

Keywords: Machine Learning, Multiple disease detection, CSV file, Data preprocessing, Decision Tree, Random Forest, KNN, Naive Bayes

Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Due to the increasing number of patients and diseases every year, the medical system has become overloaded and expensive in many countries. Typically, patients need to visit a doctor for consultations and treatments. However, with sufficient data, predicting diseases through algorithms can be an easy and cost-effective alternative.

Multiple disease prediction using machine learning has the potential to significantly improve patient outcomes. By accurately predicting diseases early on, healthcare professionals can provide appropriate treatment and prevent the development of more serious health conditions.

Furthermore, this method can help identify patterns and trends in the data, leading to new insights into the underlying causes of diseases and potential avenues for treatment. As more data is collected and analyzed, the accuracy of machine learning algorithms is expected to continue to improve, making them an increasingly valuable tool in the medical field.

Our project demonstrates the potential benefits of using machine learning algorithms to accurately predict multiple diseases based on patient symptoms. By achieving an accuracy rate of 92-95%, we have shown that this method is promising and can be implemented in future medical treatments. Through the use of an interactive interface and visualizations, we have made this system more accessible and user-friendly, allowing healthcare professionals to more easily integrate it into their practice.

Chapter 2

Basic Concepts

2.1 Decision Tree:

It is a graphical depiction of all possible solutions to a problem/decision given certain parameters.It is a tree-structured classifier in which internal nodes contain dataset attributes, branches represent decision rules, and each leaf node represents the result. A Decision tree has two nodes: the Decision Node and the Leaf Node. Decision nodes are used to make decisions and have numerous branches, whereas Leaf nodes represent the results of those decisions and do not have any more branches. The judgements or tests are based on the characteristics of the provided dataset.

2.2 Random Forest:

Random Forest is a popular machine learning algorithm used in multiple disease prediction. It is an ensemble learning method that combines multiple decision trees to create a model that is more accurate and robust.

- The algorithm builds multiple decision trees by selecting random subsets of features from the input dataset.
-Each decision tree is trained on a bootstrapped subset of the training data and generates a prediction.
-The final prediction is made by aggregating the predictions from all decision trees.

The Random Forest algorithm is robust against overfitting and is well-suited for high-dimensional datasets. It is trained on input variables that are known to be associated with specific diseases.The trained model can predict the likelihood of having specific diseases based on input variables.

2.3 KNN:

KNN (K-Nearest Neighbors) is a type of machine learning algorithm used for both classification and regression tasks. It works by finding the k number of nearest data points in the training set to the new observation and then assigns the label of the majority of the k neighbors to the new observation. In the case of regression, it takes the average of the k nearest neighbors to make the prediction. KNN is considered simple and easy to understand, but it can be computationally expensive for larger datasets. It is commonly used in various fields including image recognition, medical diagnosis, and text classification.

2.4. Naive Bayes:

The Nave Bayes method is a supervised learning technique that uses the Bayes theorem to solve classification issues. It is mostly utilised in text classification with a large training dataset. The Nave Bayes Classifier is a simple and effective Classification method that aids in the development of rapid machine learning models capable of making quick predictions. It is a probabilistic classifier, which means it predicts based on an object's likelihood.

2.5. GUI:

A GUI (Graphical User Interface) for multiple disease prediction using ML refers to a visual interface that allows users to interact with machine learning models to predict the likelihood of having specific diseases based on input variables. The GUI provides a user-friendly way to input data, visualize the results, and make predictions.

For this project, we have made the GUI using Tkinter which consists of text titles, option menu , a message box, buttons, and labels.

1. The function Root.title() is utilized to establish the title as "Smart Disease Predictor System".
2. Heading and contributor sections are created using Labels, which are also used for other sections.
3. To create a dropdown menu, OptionMenu is implemented.
4. Buttons are utilized to provide functionality and make predictions using the models. Additionally, utility buttons like exit and reset are created.
5. To display the prediction outcomes, a blank space is utilized using the Text function.

Chapter 3

Problem Statement

The way people search for health information is changing globally, and this is influencing their information needs. Many individuals encounter difficulties when seeking online information about diseases, diagnoses, and treatments. However, if a recommendation system that utilizes review mining is developed for doctors and medicines, it could save a considerable amount of time. Nonetheless, users who are not healthcare professionals often struggle to comprehend the diverse medical terminology and find themselves confused due to the abundance of medical information available across different sources. The primary aim of the recommender system is to address the specific needs of users in the health domain.

3.1 Project Planning

Activity	**Starting week**	**Number of weeks**
Literature review	2nd week of January	3
Finalizing problem	2nd week of February	2
Design of overall project	1st week of March	2
Design of sub-modules associated with the system	3rd week of March	1
Implementation starts	1st week of April	3
Preparation of project report	3rd week of April	1
Preparation of project presentation	1st week of May	1/2

3.2 System Architecture

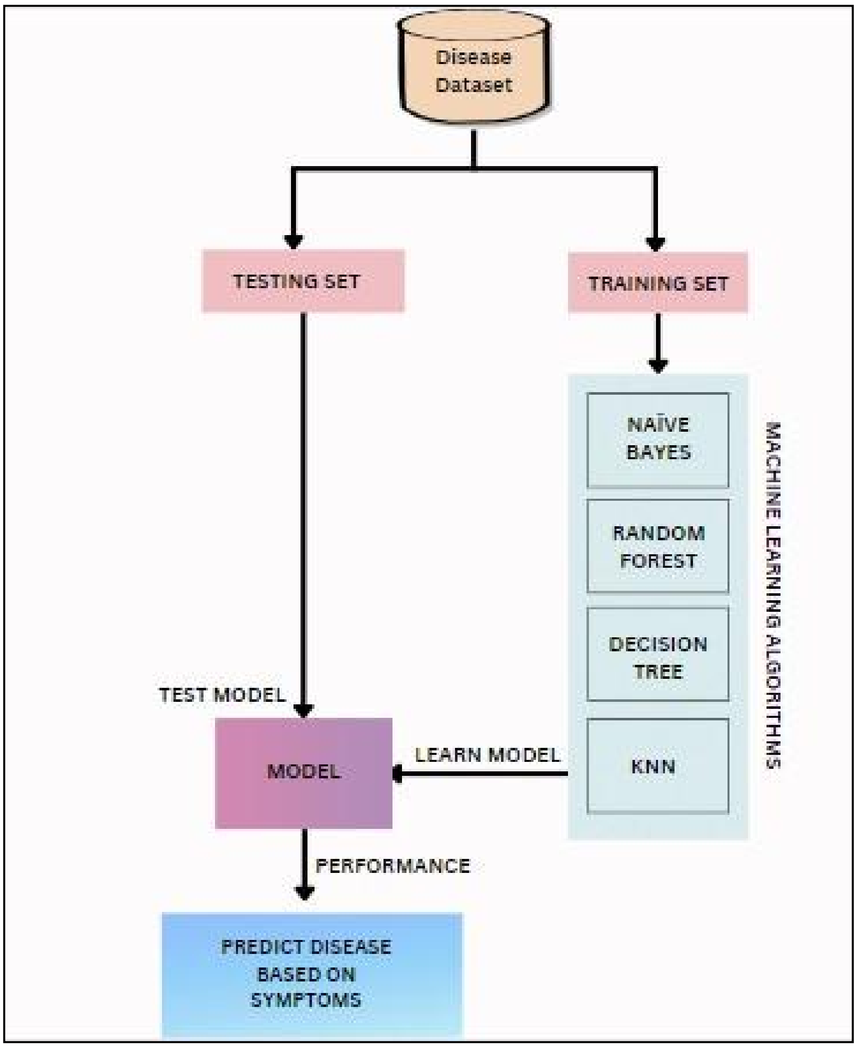


Figure:3.2.1 - Work Flow diagram

Chapter 4

Implementation

In this section, present the implementation done by you during the project development.

4.1 Reading Dataset(CSV File)

To read a CSV file for disease prediction, use Pandas to load the data as a DataFrame, and explore the data using methods such as head(), tail(), and describe(). Use random, stratified, or time-based techniques to split the data into training and testing sets. Preprocess the data by scaling features, selecting features, encoding categorical variables, and filling missing values. The selection of preprocessing and splitting techniques is important to ensure accurate and reliable predictive models for multiple diseases.

4.2 Data Preprocessing

Data preprocessing is essential for preparing data for analysis by cleaning, transforming, and normalizing it. Common steps for multiple disease prediction include data cleaning, feature scaling, encoding, selection, imputation, outlier detection and removal, and sampling. Effective data preprocessing is crucial for developing accurate and reliable predictive models for multiple disease outcomes.

4.3 Exploratory data analysis (EDA)

Exploratory data analysis (EDA) is a process of examining a dataset to identify patterns and relationships between variables using techniques such as visualization, summary statistics, and dimensionality reduction. In multiple disease prediction, EDA is used to analyze the relationships between risk factors and potential predictors by examining distribution and correlation, identifying outliers and missing values, and visualizing data to identify patterns and trends. EDA provides a useful starting point for understanding the data and identifying potential predictors, but more advanced techniques may be necessary.

4.4 Feature selection

Feature selection is the process of identifying informative features to improve accuracy and interpretability of predictive models. It is challenging in multiple disease prediction due to high dimensionality and complexity. Three methods are used: filter, wrapper, and embedded. Common approaches include correlation-based methods, tree-based selection, and regularization. Effective feature selection is critical for developing accurate models and improving generalizability.

4.5 Model Fitting

Model fitting for multiple disease prediction involves selecting a suitable algorithm, data preprocessing, feature selection, model training, validation, and tuning. A large and diverse dataset with relevant input features is necessary, and the model's performance should be validated on an independent dataset.

4.6 Machine Learning Evaluation

Machine learning evaluation is vital in developing accurate predictive models for multiple diseases. Common evaluation metrics include confusion matrix, accuracy, precision, recall, F1 score, ROC curve, and AUC. The choice of evaluation metric depends on the research question and dataset characteristics. The evaluation process also involves determining the appropriate classification threshold. Careful consideration of the evaluation metrics and threshold is critical to developing reliable predictive models for multiple disease outcomes.

4.7 Work Flow

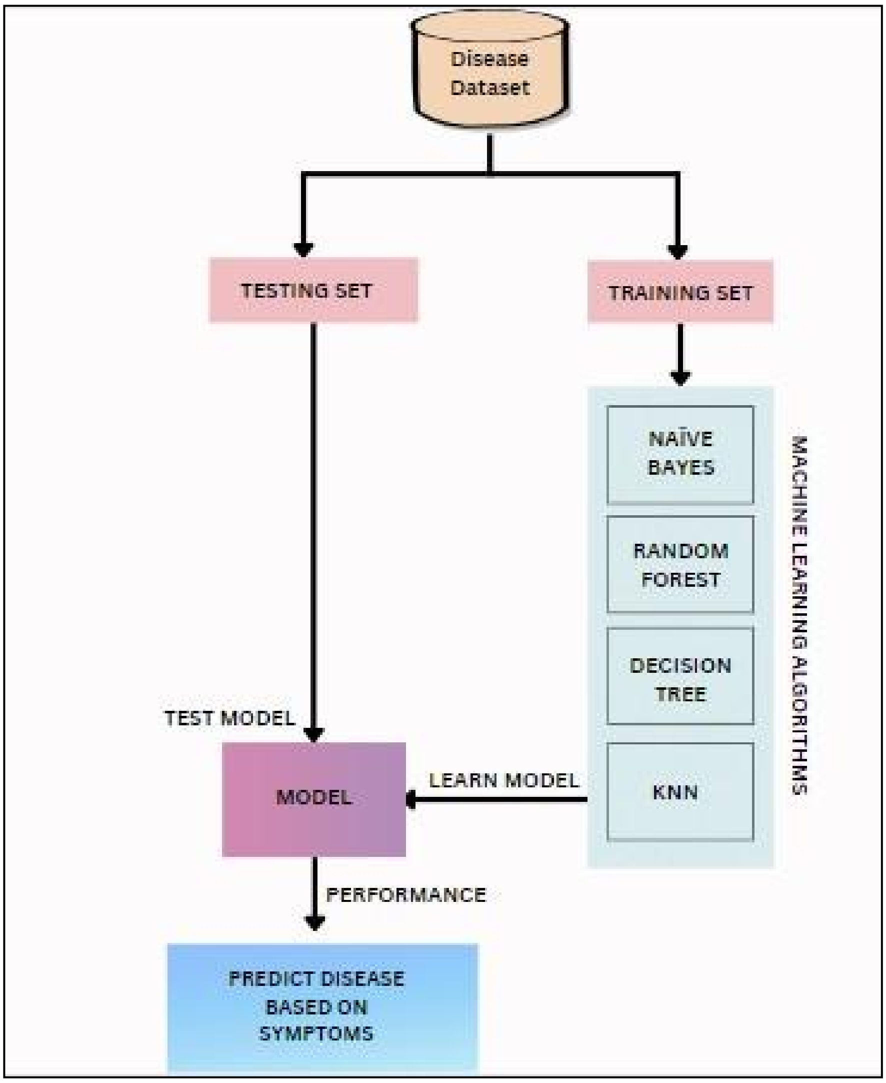
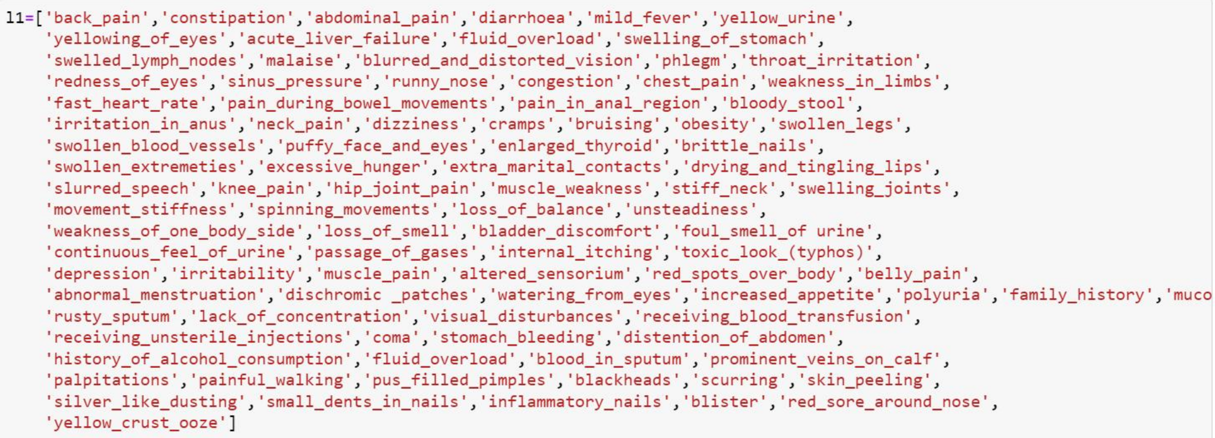


Figure:4.7.1 - Work Flow diagram

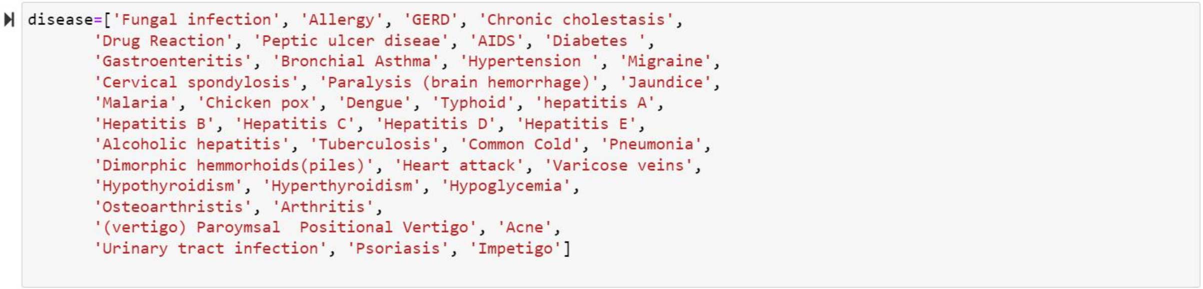
4.8 Code

L1 is a collection of numerous symptoms that are typically present in patients with different diseases.



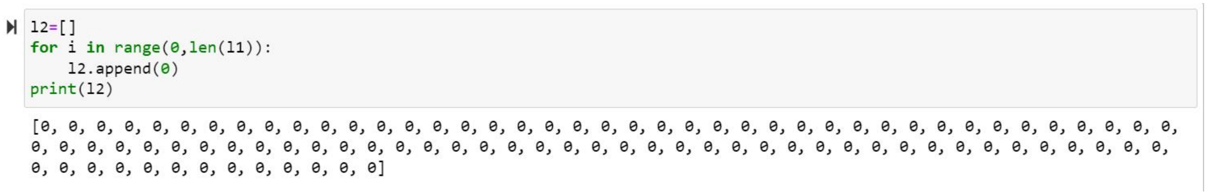
```
l1=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
    'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
    'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',
    'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_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','muco
    'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
    'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',
    'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_calf',
    'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',
    'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose',
    'yellow_crust_ooze']
```

Disease is a list of various diseases that have, for the most part, affected different people.



```
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']
```

The first L2 is the created vacant list. At that moment, L2 is appended in a number of zeroes, which corresponds to a number of diseases in list L1.



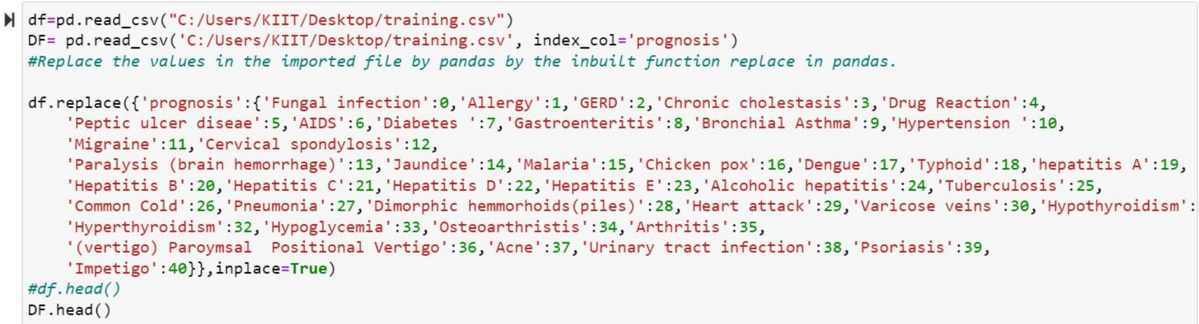
```
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Training.csv is a CSV file that contains diseases and symptoms and is used to prepare the model. The Read_csv() function is used to save the data in the dataframe named df.

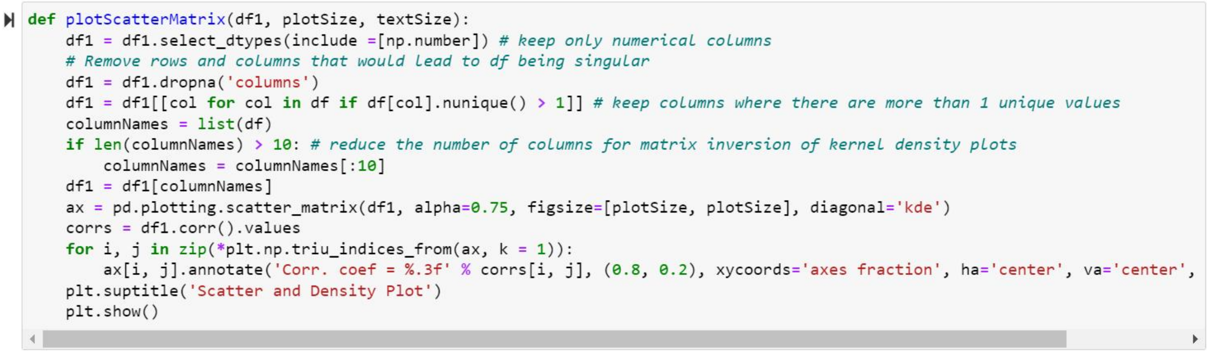
Using the replace() function, the prognosis column, which contains the different diseases, is replaced by numbers ranging from 0 to n-1, where n is the number of different diseases

present in the.csv record. The Head() function is used to display the first five rows of the prepared dataframe.

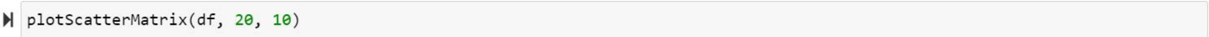


```
df=pd.read_csv("C:/Users/KIIT/Desktop/training.csv")
DF= pd.read_csv('C:/Users/KIIT/Desktop/training.csv', index_col='prognosis')
#Replace the values in the imported file by pandas by the inbuilt function replace in pandas.

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':
    '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()
```

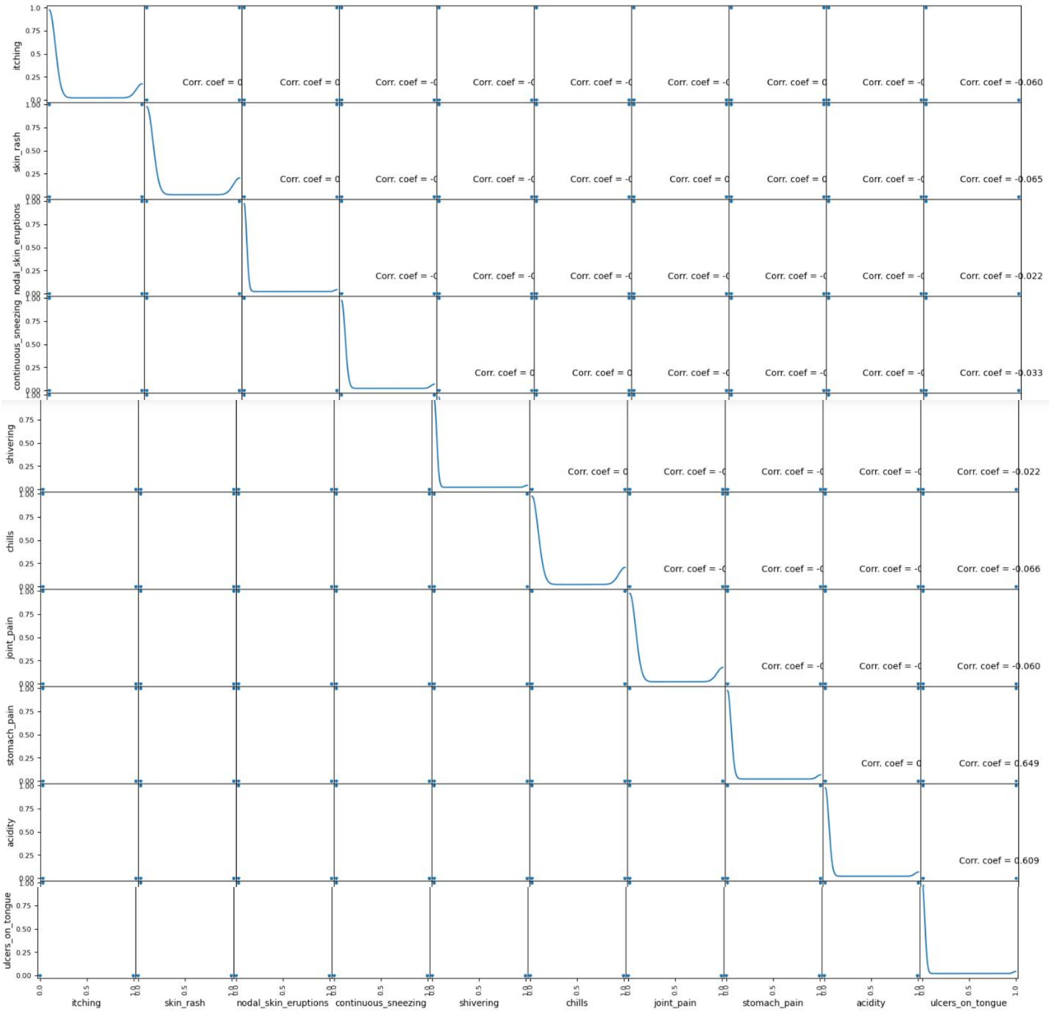
This is the code for the scatter and density plots of the training.csv file's columns.



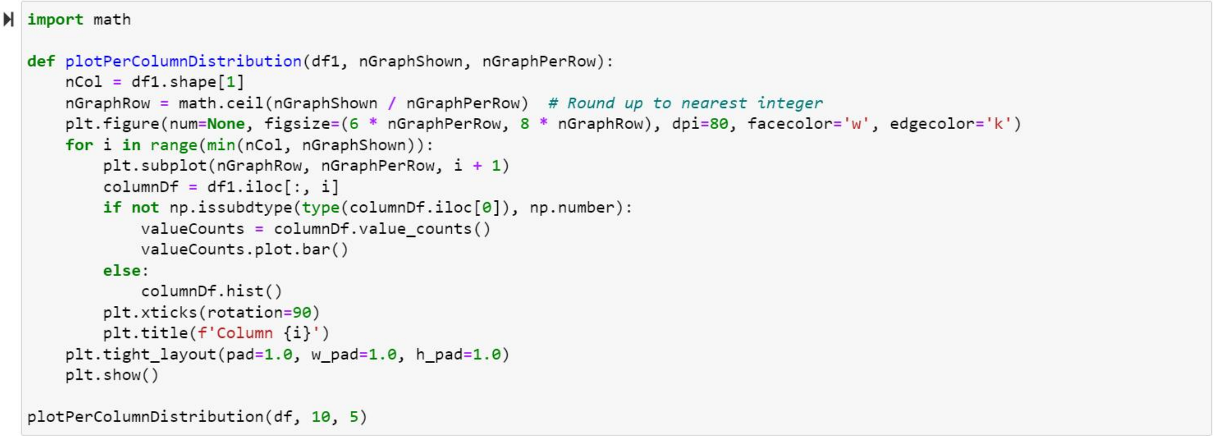
```
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]] # keep columns 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',
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```



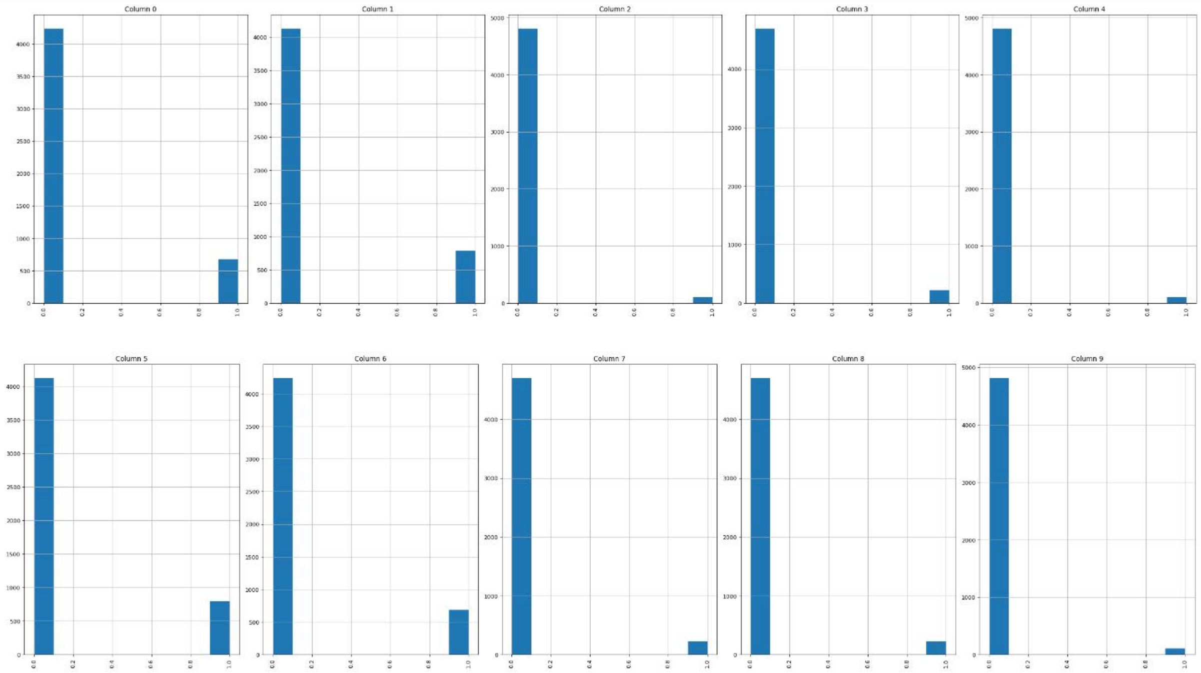
```
plotScatterMatrix(df, 20, 10)
```



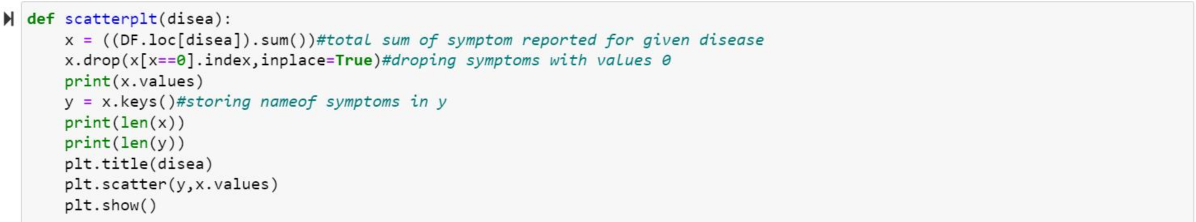
This is the code for the column distribution graph in the training.csv file.



```
import math

def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nCol = df1.shape[1]
    nGraphRow = math.ceil(nGraphShown / nGraphPerRow)  # Round up to nearest integer
    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 = df1.iloc[:, i]
        if not np.issubdtype(type(columnDf.iloc[0]), np.number):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.xticks(rotation=90)
        plt.title(f'Column {i}')
    plt.tight_layout(pad=1.0, w_pad=1.0, h_pad=1.0)
    plt.show()

plotPerColumnDistribution(df, 10, 5)
```

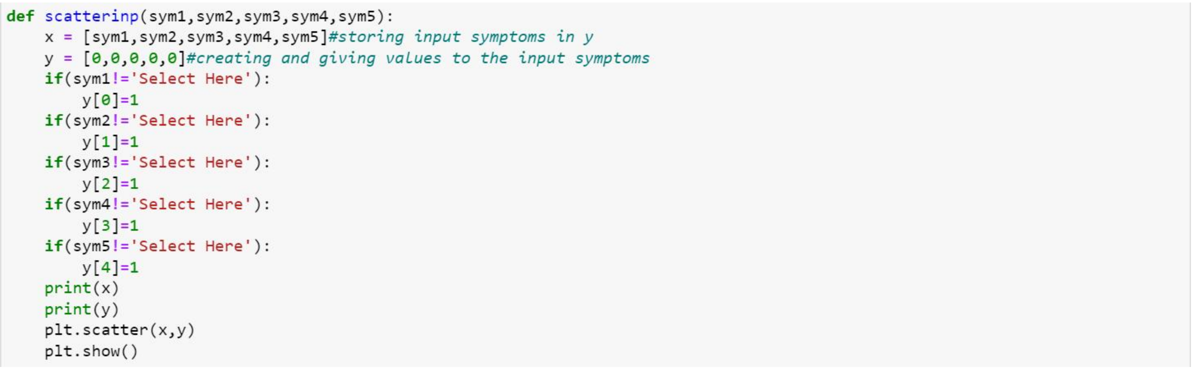


These are the functions for plotting the scatterplot of the user's predicted diseases and symptoms



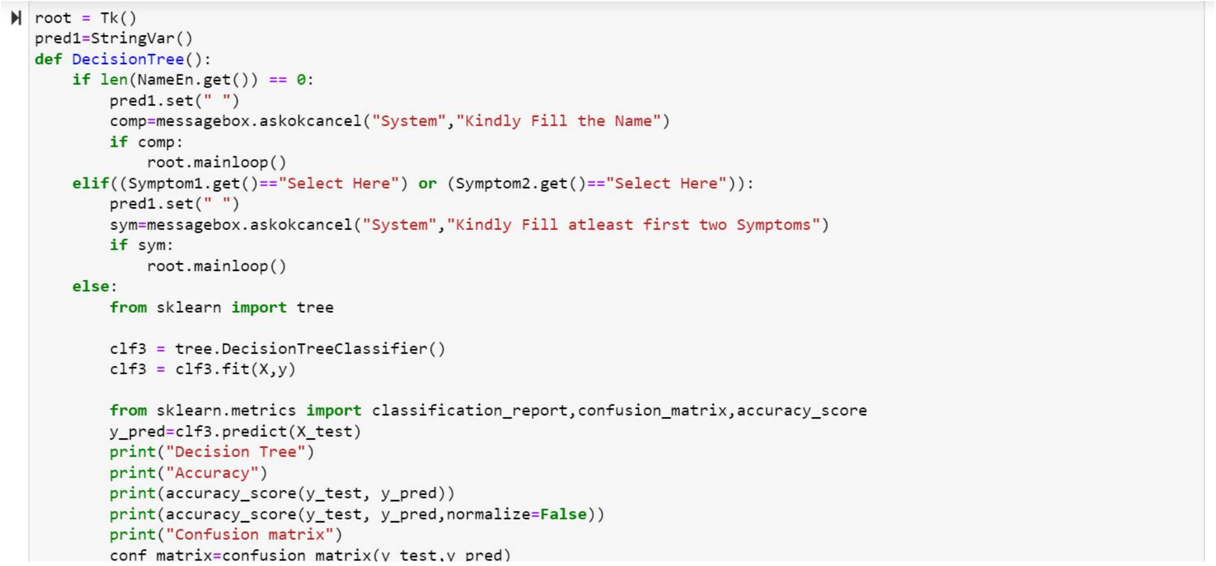
```
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())#total sum of symptom reported for given disease
    x.drop(x[x==0].index,inplace=True)#droping symptoms with values 0
    print(x.values)
    y = x.keys()#storing nameof symptoms in y
    print(len(x))
    print(len(y))
    plt.title(disea)
    plt.scatter(y,x.values)
    plt.show()
```

.

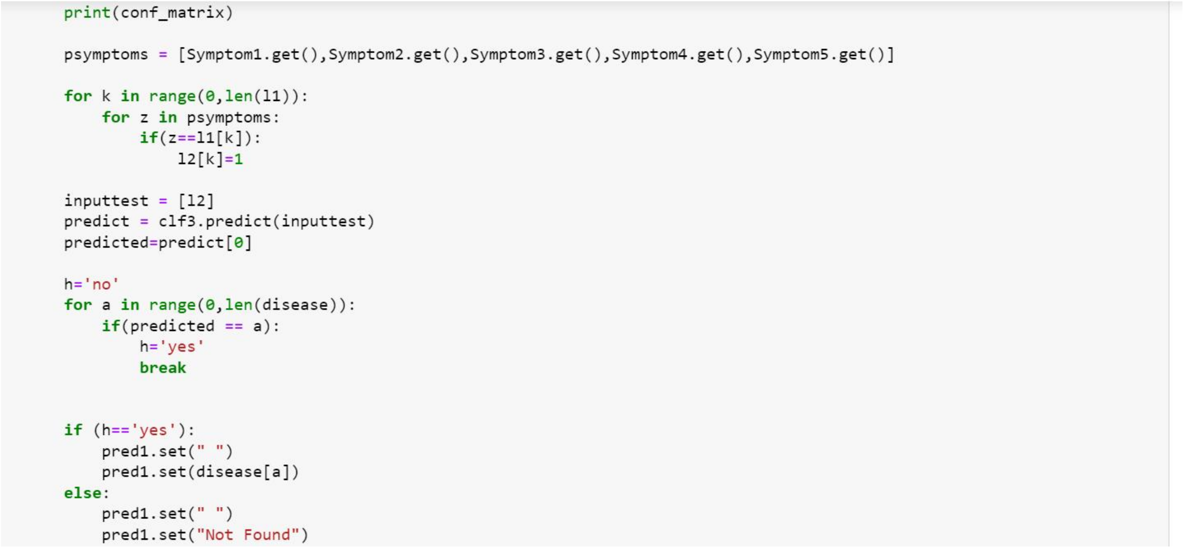


```
def scatterinp(sym1,sym2,sym3,sym4,sym5):
    x = [sym1,sym2,sym3,sym4,sym5]#storing input symptoms in y
    y = [0,0,0,0,0]#creating and giving values to the input symptoms
    if(sym1!='Select Here'):
        y[0]=1
    if(sym2!='Select Here'):
        y[1]=1
    if(sym3!='Select Here'):
     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

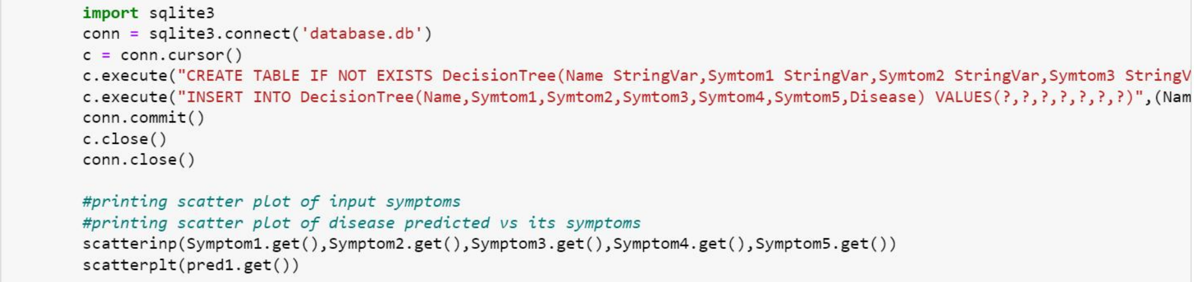
Decision tree function:



```
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)
```

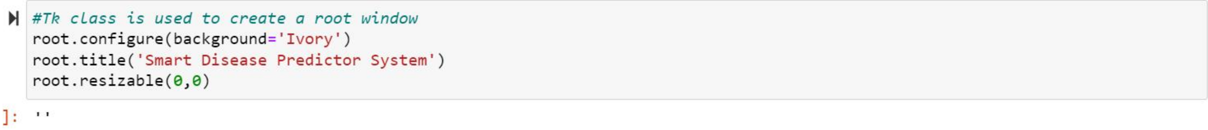


```
        print(conf_matrix)

        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]

        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")
```



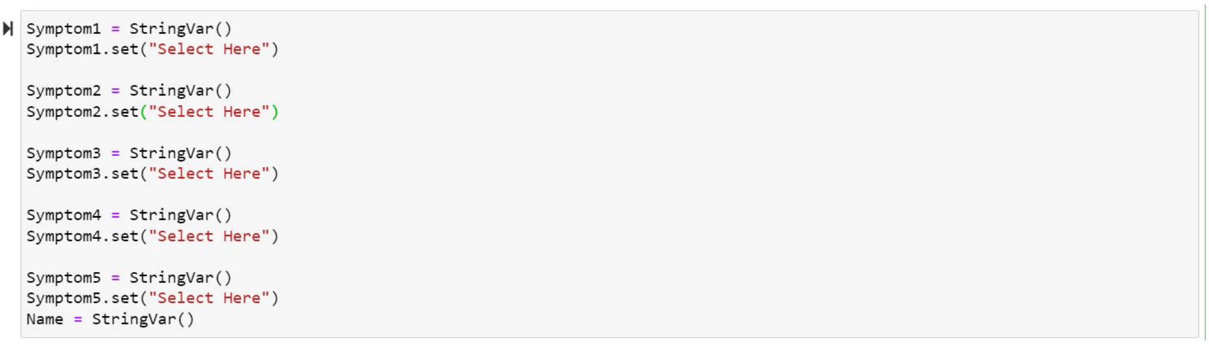
```
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringV
        c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)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

Building GUI:

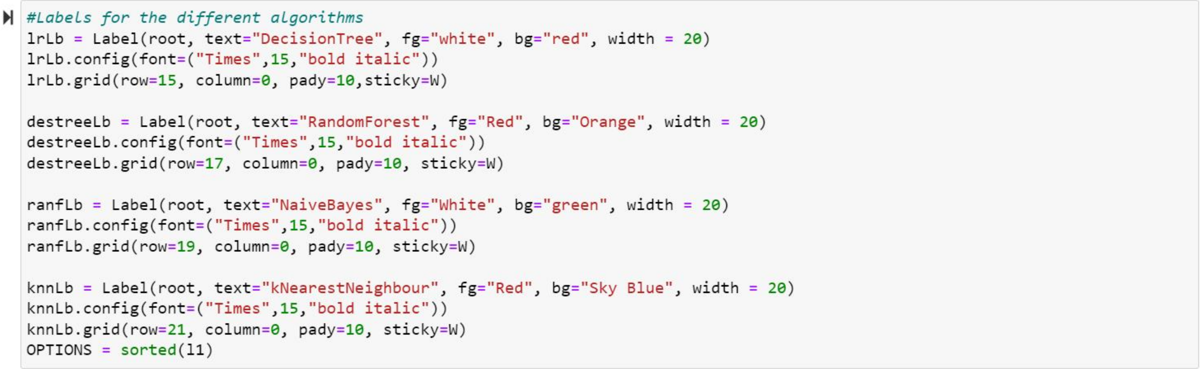


```
#Tk class is used to create a root window
root.configure(background='Ivory')
root.title('Smart Disease Predictor System')
root.resizable(0,0)
```

```
]: ''
```



```
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()
```

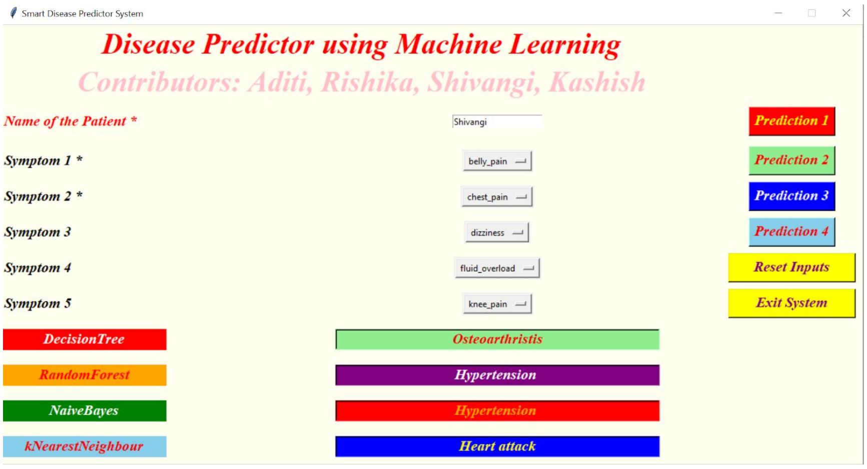


```
#Labels for the different algorithms
lrLb = Label(root, text="DecisionTree", fg="white", bg="red", width = 20)
lrLb.config(font=("Times",15,"bold italic"))
lrLb.grid(row=15, column=0, pady=10,sticky=W)

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)
```

Chapter 5

Result

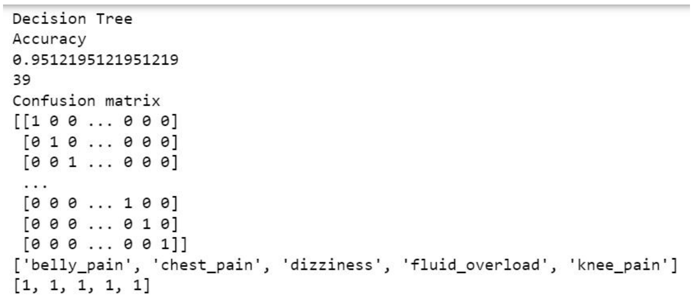
GUI made for this project is a simple tkinter GUI consisting of labels, message box, button, text, title and option menu



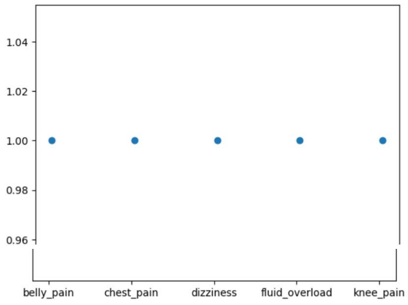
Name of the Patient * Shivangi
Symptom 1 * belly_pain
Symptom 2 * chest_pain
Symptom 3 dizziness
Symptom 4 fluid_overload
Symptom 5 knee_pain

DecisionTree Osteoarthristis
RandomForest Hypertension
NaiveBayes Hypertension
kNearestNeighbour Heart attack

It also prints the accuracy, confusion matrix and the scatter plot of each algorithm as shown below:



```
Decision Tree
Accuracy
0.9512195121951219
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
['belly_pain', 'chest_pain', 'dizziness', 'fluid_overload', 'knee_pain']
[1, 1, 1, 1, 1]
```



```
[114 114 114 114 114 114]
6
6
```

ACCURACY SCORE:

ML Algorithms	Accuracy
Decision Tree	95.12%
Random Forest	95.12%
KNN	95.12%
Naive Bayes	95.12%

Table 5.1 Showing details about Accuracy score

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Our objective was to develop a system that can predict diseases based on the symptoms provided, thereby reducing the burden on hospital OPDs and medical staff. We successfully created such a system, which uses four different algorithms and achieved an average accuracy of approximately 95%.

In addition to accurate disease prediction, our system features a user-friendly interface and offers various visual representations of the collected data and results obtained.

6.2 Future Scope

In the future we can add more diseases in the existing API. We can try to improve the accuracy of prediction in order to decrease the mortality rate.

We can try to make a WebApp using Streamlit an also add a Recommender system for recommending list of hospitals and doctors according to disease.

Try to make the system user-friendly and provide a chatbot for normal queries.

References

[1] K Arumugama, Mohd Navedb, Priyanka P. Shindec, Orlando Leiva-Chaucad, Antonio Huaman-Osorioe, Tatiana Gonzales-Yanac "Multiple disease prediction using Machine learning algorithms" volume 60, Materials Today - Proceedings (2021).

[2] Ankush Singh, Ashish Yadav, Saloni Shah,Prof. Renuka Nagpure "Multiple Disease Prediction System" International Research Journal of Engineering and Technology (IRJET) Volume: 09 Issue: 03 (Mar 2022)

[3] Indukuri Mohit et al 2021 J. Phys.: Conf. Ser. 2089 012009 "An Approach to detect multiple diseases using machine learning algorithm"

[4] https://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

[5] https://techieyantechnologies.com/multiple-disease-prediction-system-using-machine-learning/

[6] https://www.geeksforgeeks.org/disease-prediction-using-machine-learning/

INDIVIDUAL CONTRIBUTION REPORT:

MULTIPLE DISEASE PREDICTION

ADITI YADAV - 2006157
RISHIKA SINGH - 2006187
SHIVANGI KUMARI- 2006287
KASHISH KAUR -2006467

Abstract: Multiple disease detection using machine learning is an area of healthcare research enabled by large datasets and advanced algorithms. Data preprocessing and splitting techniques are important for creating accurate predictive models. Our minor project utilized a dataset from a 2004 study by the University of Columbia and achieved approximately 95% accuracy using four techniques: Decision Tree, Random Forest, KNN, and Naive Bayes. The system has a user-friendly interface and offers visual representations of data and results. The goal is to reduce the burden on hospital OPDs and medical staff by predicting diseases based on symptoms provided.

Individual contribution to project report preparation:

Aditi Yadav - Data Pre-processing, Model training using decision tree, Plotting graph through Column Distribution, Scatterplot, GUI, Project Report Preparation.

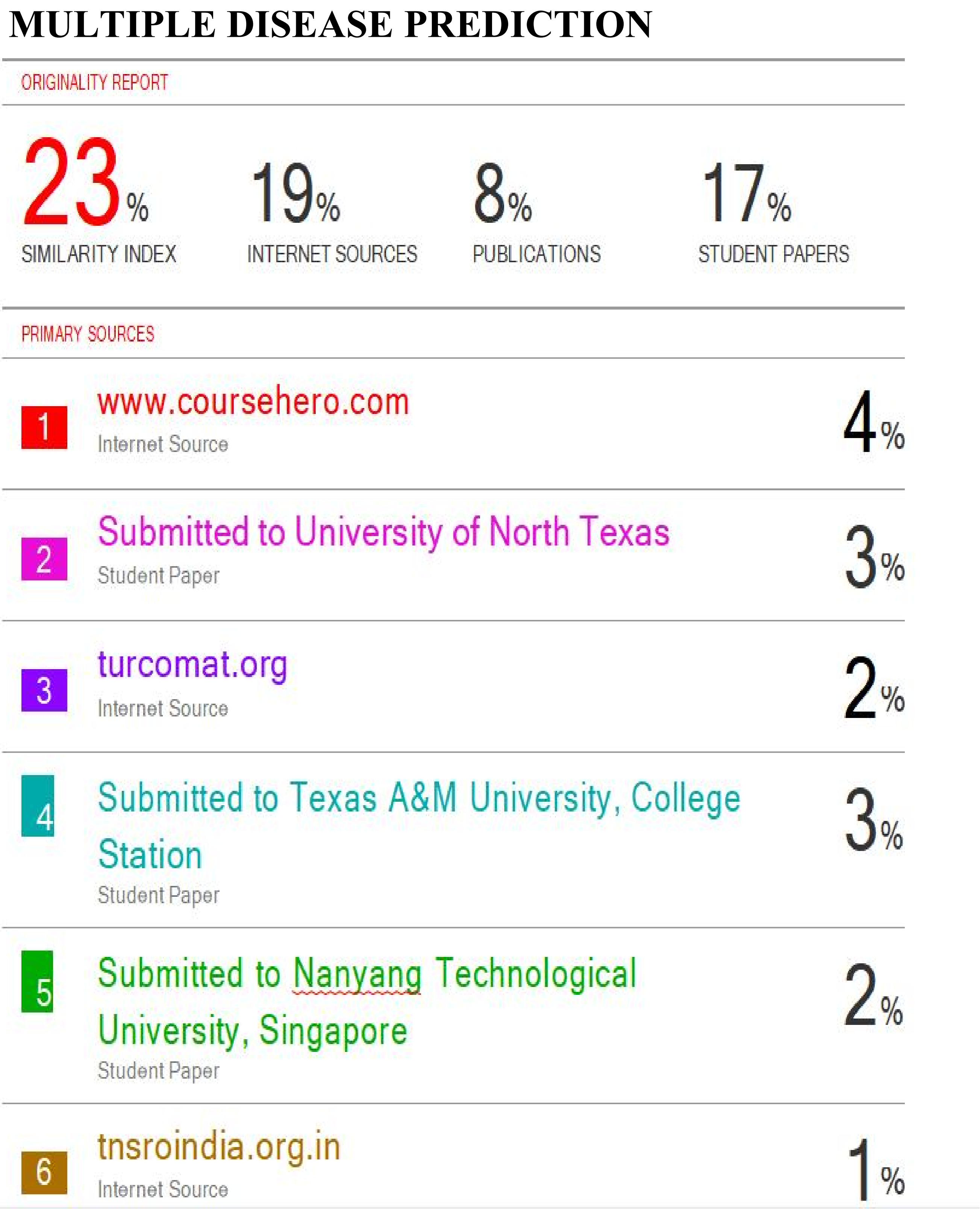
Rishika Singh - Model training using KNN, Plotting graph through Scatter Matrix, Scatterplot, GUI, Project Report Preparation .

Shivangi Kumari - Model training using Random Forest, Plotting graph through Scatter Matrix, Scatterplot, GUI, Project Report Preparation.

Kashish Kaur - Model training using Naive Bayes, Plotting graph through Column Distribution, GUI, Scatterplot, Project Report Preparation.

Full Signature of Supervisor: Full signature of the student:

TURNITIN PLAGIARISM REPORT



MULTIPLE DISEASE PREDICTION

ORIGINALITY REPORT

23%	19%	8%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	4%
2	Submitted to University of North Texas Student Paper	3%
3	turcomat.org Internet Source	2%
4	Submitted to Texas A&M University, College Station Student Paper	3%
5	Submitted to Nanyang Technological University, Singapore Student Paper	2%
6	tnsroindia.org.in Internet Source	1%