Disease prediction using Machine Learning

SREENITHI KM

Abstract:

In recent years, the field of healthcare has witnessed a significant transformation due to the advent of machine learning technologies. This paper explores the application of three machine learning algorithms—Random Forest, Naive Bayes, and Decision Tree—for disease prediction. By leveraging clinical datasets, we aim to evaluate the effectiveness of these algorithms in predicting the onset of various diseases. The study provides insights into the strengths and weaknesses of each algorithm and offers a comparative analysis based on their performance metrics. The goal is to identify the most suitable algorithm for accurate and reliable disease prediction, which could aid healthcare professionals in early diagnosis and treatment planning.

Introduction

Disease prediction is a critical component in modern healthcare, aiming to identify potential health issues before they manifest into serious conditions. Integrating machine learning (ML) techniques in medical diagnostics can enhance predictive accuracy and provide personalized treatment options. Traditional diagnostic methods often rely heavily on the expertise and experience of healthcare professionals, which can be subjective and inconsistent. Machine learning, on the other hand, offers a data-driven approach, ensuring more consistent and objective results.

This paper investigates the use of three prominent machine learning algorithms—Random Forest, Naive Bayes, and Decision Tree—in the context of disease prediction. These algorithms were chosen due to their widespread application and proven efficacy in various domains of classification problems. The primary focus is to evaluate their performance using clinical datasets, thus providing a comparative study that highlights their respective advantages and limitations.

The healthcare landscape has undergone a significant transformation, with the number of patients and the prevalence of diseases steadily rising across the globe. This surge in demand has put an immense strain on medical systems, leading to widespread overload and, in many countries, a concerning increase in healthcare costs. A significant contributor to this challenge is the reliance on in-person consultations with doctors as the primary means of disease diagnosis and treatment.

However, the advent of advanced data-driven algorithms holds the promise of revolutionizing this landscape. By leveraging sufficient data, these algorithms can accurately predict diseases based on the patient's symptoms, potentially eliminating the need for costly and time-consuming doctor visits in many cases. This shift towards data-driven disease prediction represents a pivotal moment in the future of medical treatment. In this project, we have explored the potential of this approach, aiming to accurately predict diseases by examining the symptoms presented by the patient. Through the implementation of four distinct algorithms, we have achieved an impressive accuracy ranging from 92% to 95%. This remarkable level of precision highlights the transformative power of this technology and its ability to reshape the medical treatment landscape.

To further enhance the user experience and accessibility of this system, we have also designed an interactive interface to facilitate seamless interaction. Additionally, we have made concerted efforts to thoroughly analyze and visualize the results of our study, ensuring a comprehensive understanding of the project's outcomes.

The development of this disease prediction system marks a significant stride towards a more efficient, equitable, and accessible healthcare system. By empowering patients with accurate and affordable diagnoses, we can pave the way for a future where medical treatment is more streamlined, cost-effective, and centered on the individual's needs.

Database Collection

Dataset for this project was collected from a study of university of Columbia performed at New York Presbyterian Hospital during 2004. Link of dataset is given below. http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

Need for the Project

The healthcare industry faces numerous challenges, including the early detection and accurate diagnosis of diseases. Early intervention is crucial in managing chronic conditions and improving patient outcomes. However, existing diagnostic methods often suffer from limitations such as:

- **Subjectivity:** Diagnostic accuracy can vary significantly among healthcare professionals.
- **Time Constraints:** Manual diagnosis is time-consuming, potentially delaying treatment.
- **Data Overload:** The vast amount of medical data available today can overwhelm healthcare providers, making it difficult to extract relevant insights.

Machine learning offers a solution to these challenges by automating the diagnostic process, thereby reducing human error and improving efficiency. By developing reliable disease prediction models, we can provide healthcare professionals with powerful tools to make informed decisions, ultimately enhancing patient care and reducing healthcare costs.

Methodology:

The project utilizes a range of standard libraries for database analysis and model creation, including:

Tkinter: A standard GUI library of Python, Tkinter provides a fast and easy way to create graphical user interfaces (GUIs). It offers a powerful object-oriented tool for creating GUIs, with various widgets such as buttons, canvas, labels, entry fields, checkbuttons, listboxes, and more. In this project, Tkinter was used to create an interactive GUI for the disease prediction model.

NumPy: A core library for scientific computing in Python, NumPy provides powerful tools for dealing with multi-dimensional arrays. It serves as a general-purpose array processing package, making it highly suitable for handling and manipulating the data required for disease prediction.

Pandas: A popular Python library for data analysis, Pandas offers highly optimized performance with backend code written in C or Python. It provides two main data structures: Series (one-dimensional array) and DataFrames (two-dimensional data structure with rows and columns). In this project, Pandas DataFrames were extensively used to work with the datasets required for training and testing the algorithms.

Scikit-learn (sklearn): An open-source machine learning library for Python, Scikit-learn provides a wide range of algorithms for classification, regression, and clustering tasks. In this project, the inbuilt classification algorithms from Scikit-learn, such as Decision Tree, Random Forest Classifier, and Naive Bayes, were utilized for disease prediction.

The synergistic use of these libraries allowed the researchers to efficiently handle data preprocessing, model development, and the creation of an interactive GUI for the disease prediction system.

Algorithm Descriptions

Random Forest Algorithm

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes of the individual trees (Breiman, 2001). It improves predictive accuracy and controls over-fitting by averaging the results of multiple decision trees. The algorithm works as follows:

Data Preparation: The data is divided into subsets.
Tree Construction: For each subset, a decision tree is constructed using a random subset of features.
Voting: Each tree provides a classification, and the final output is determined by majority voting.

Random Forest is robust to overfitting, especially when there are many trees, and it can handle large datasets with higher dimensionality effectively.

Naive Bayes Algorithm

Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming strong independence between the features. Despite the simplicity of its assumptions, it performs surprisingly well in various real-world situations (Zhang, 2004). The steps include:

Calculate Prior Probability: Determine the probability of each class from the dataset.
Calculate Likelihood: For each feature given a class, calculate the likelihood using the feature distribution.
Apply Bayes' Theorem: Compute the posterior probability for each class.
Classification: Assign the class with the highest posterior probability.

Naive Bayes is particularly effective for large datasets and provides good results in applications where the assumption of feature independence holds.

Decision Tree Algorithm

Decision Tree is a non-parametric supervised learning method used for classification and regression. It splits the dataset into subsets based on the most significant attribute. The process is as follows:

Tree Building: The tree is constructed by recursively splitting the dataset into subsets based on attribute values (Quinlan, 1986).
Splitting Criteria: The best split is determined using criteria like Gini impurity or information gain.
Tree Pruning: The tree is pruned to avoid overfitting by removing branches that have little importance.

Decision Trees are easy to interpret and visualize, making them a popular choice for understanding the decision-making process.

GUI

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

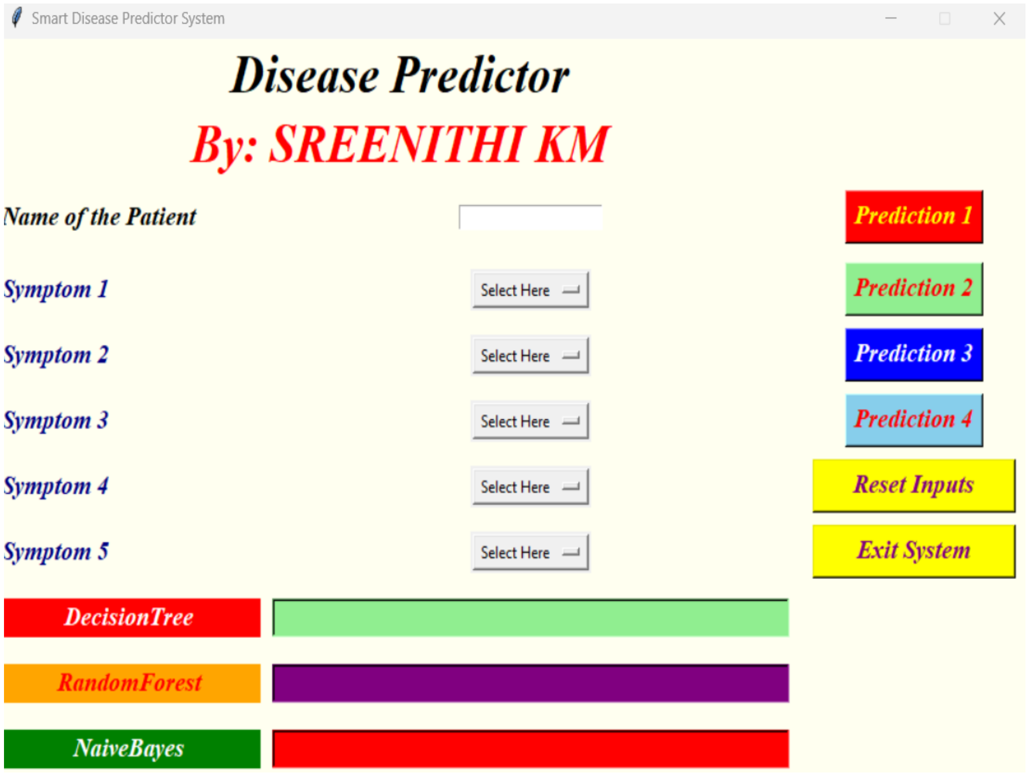


Figure 1: Disease Predictor System

Label is used to add heading.

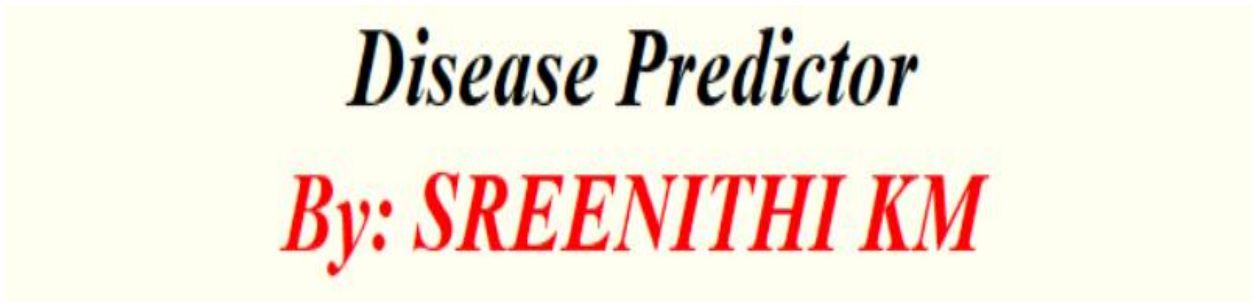


Figure 2: Title

Labels are further used for different sections

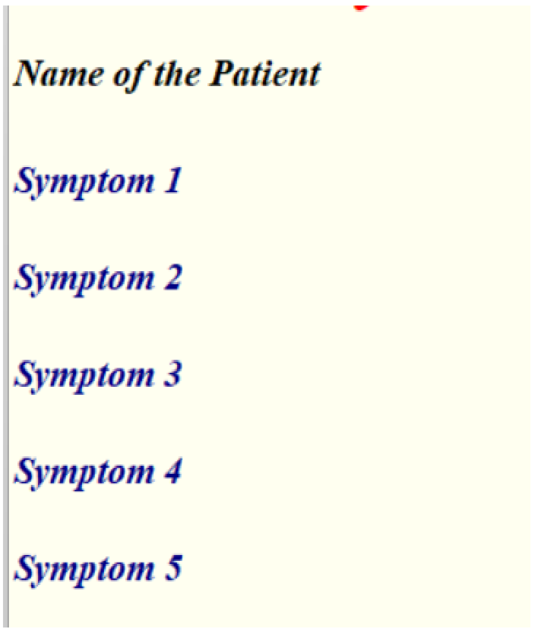


Figure 3: Sections

OptionMenu is used to create drop down menu

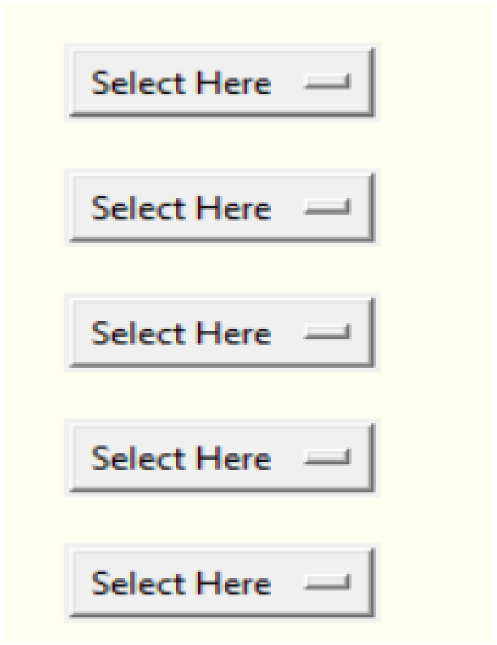


Figure 4: Selection place

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

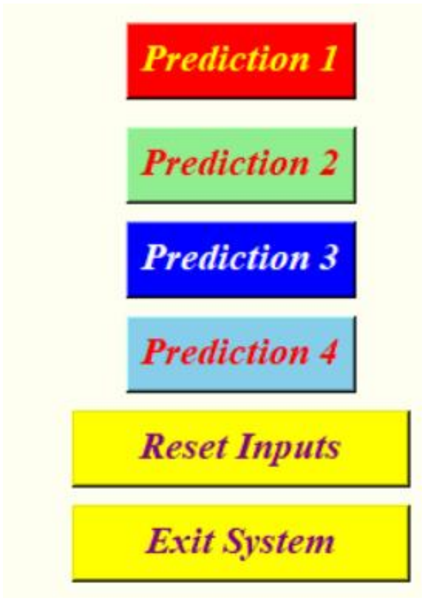
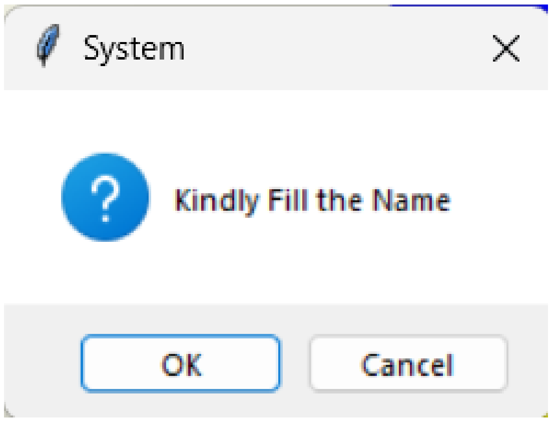


Figure 5: Prediction Botton

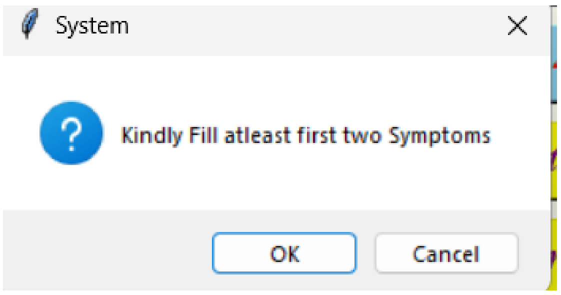
Messagebox are used at three different places, one- to restrain the to enter name


System

Kindly Fill the Name

Figure 6: Error Botton

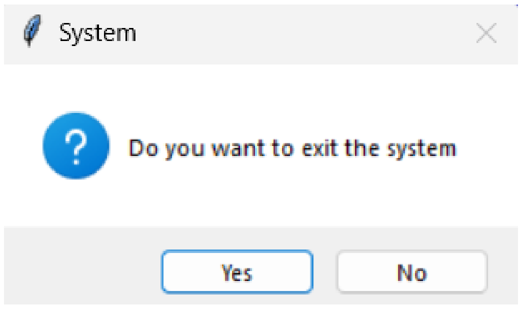
two- to ask for at least two symptoms


System

Kindly Fill atleast first two Symptoms

Figure 7: Error Botton

three- to confirm to exit system

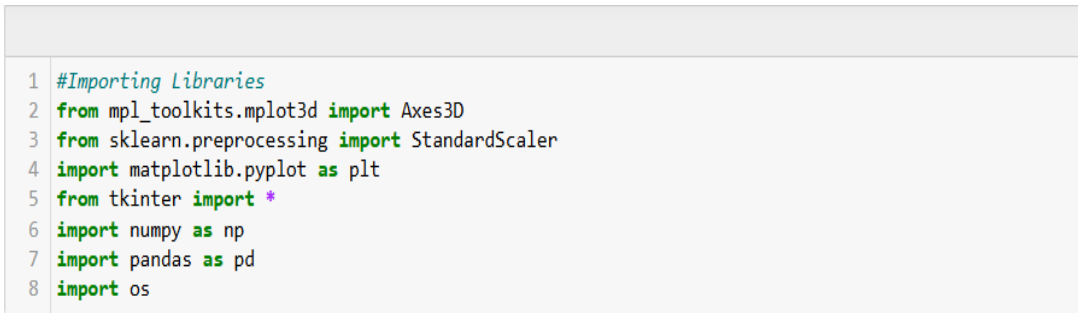


Do you want to exit the system

Figure 8: Error Botton

Modules

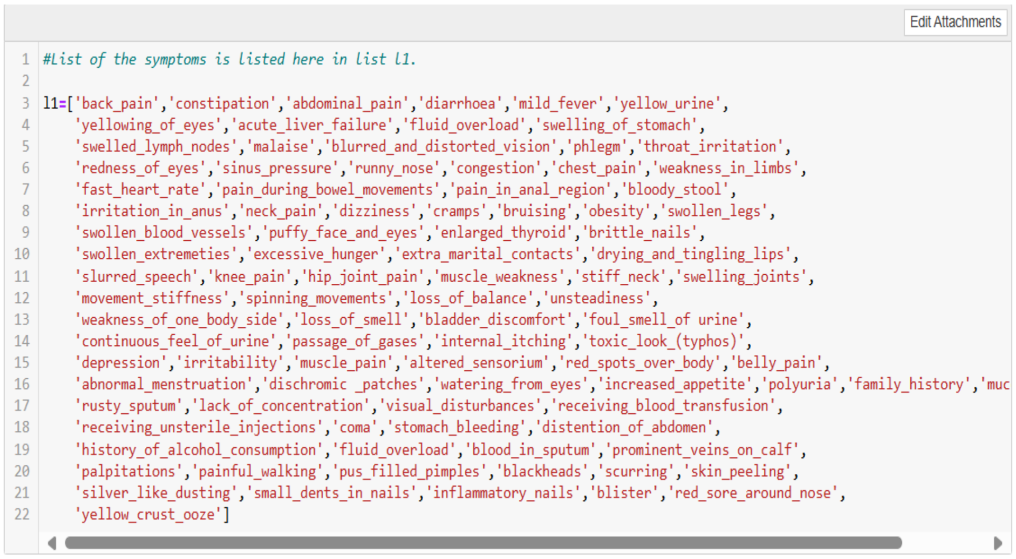
Importing Libraries



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

Figure 9: Libraries

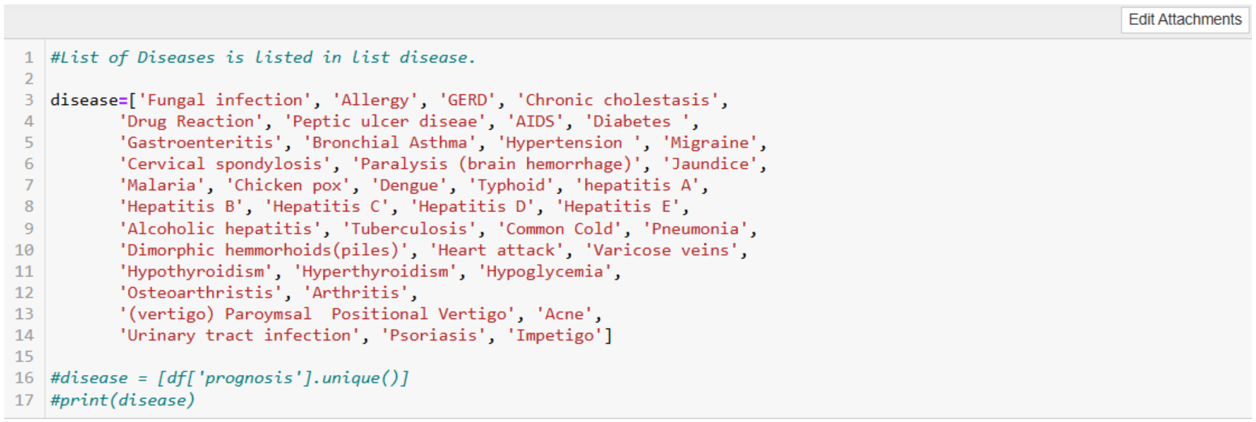
List of the symptoms is listed here in list l1.



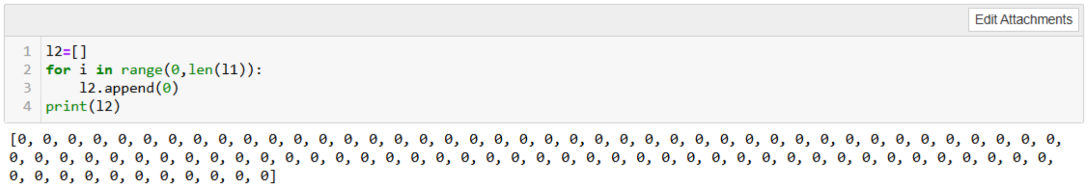
```python
#List of the symptoms is listed here in list l1.

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','muc
    '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']
```

Figure 10: Symptoms is listed

List of Diseases is listed in list disease.



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

#disease = [df['prognosis'].unique()]
#print(disease)
```

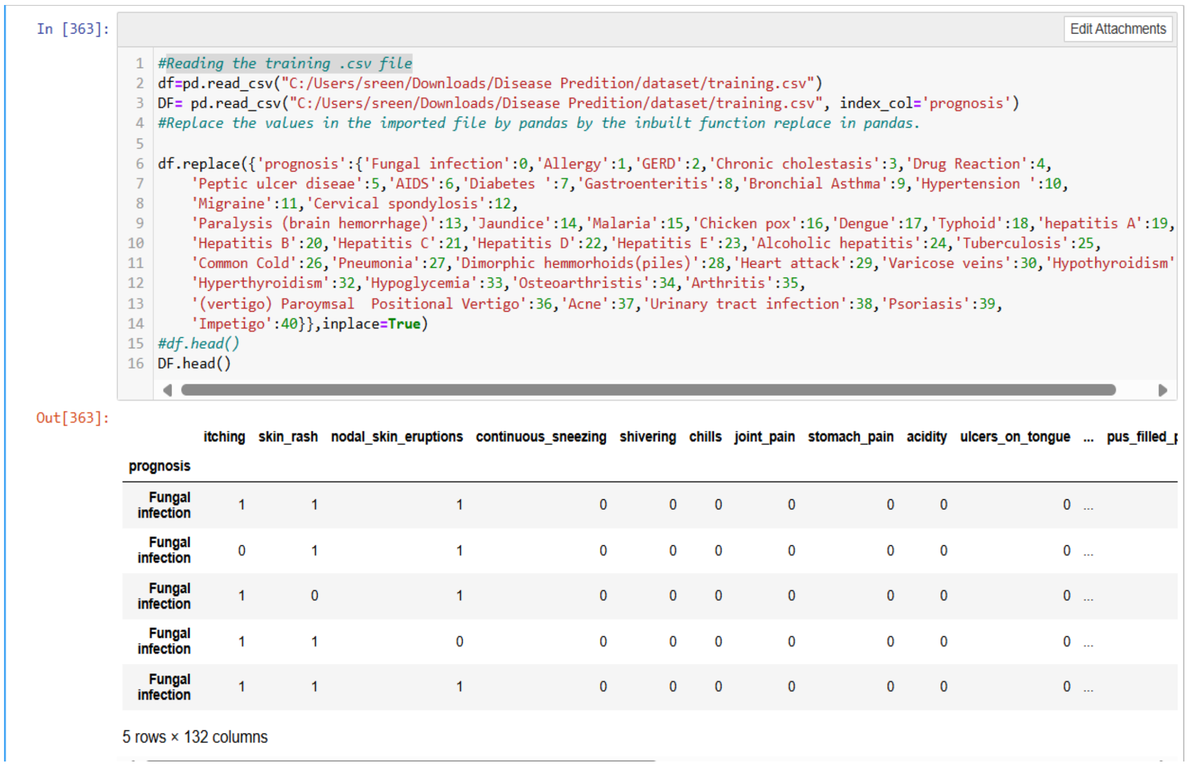


```
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
```

```
[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]
```

Figure 11: List of Diseases

Reading the training .csv file



In [363]:

```
#Reading the training .csv file
df=pd.read_csv("C:/Users/sreen/Downloads/Disease Predition/dataset/training.csv")
DF= pd.read_csv("C:/Users/sreen/Downloads/Disease Predition/dataset/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()
```

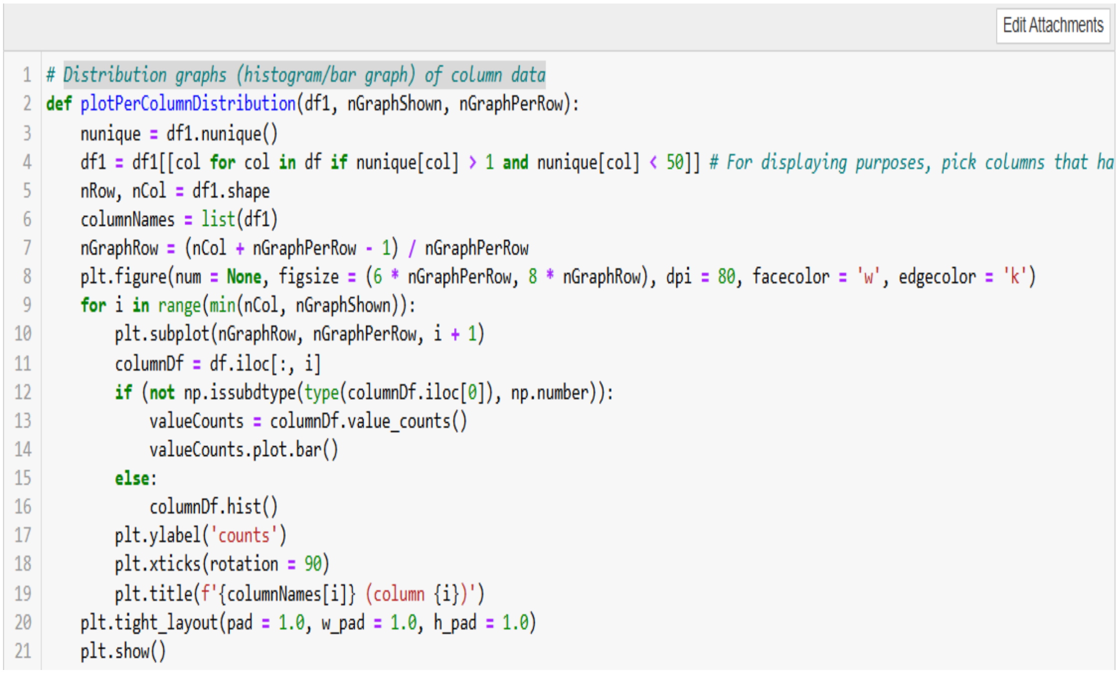
Out[363]:

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

5 rows × 132 columns

Figure 12: Reading the training

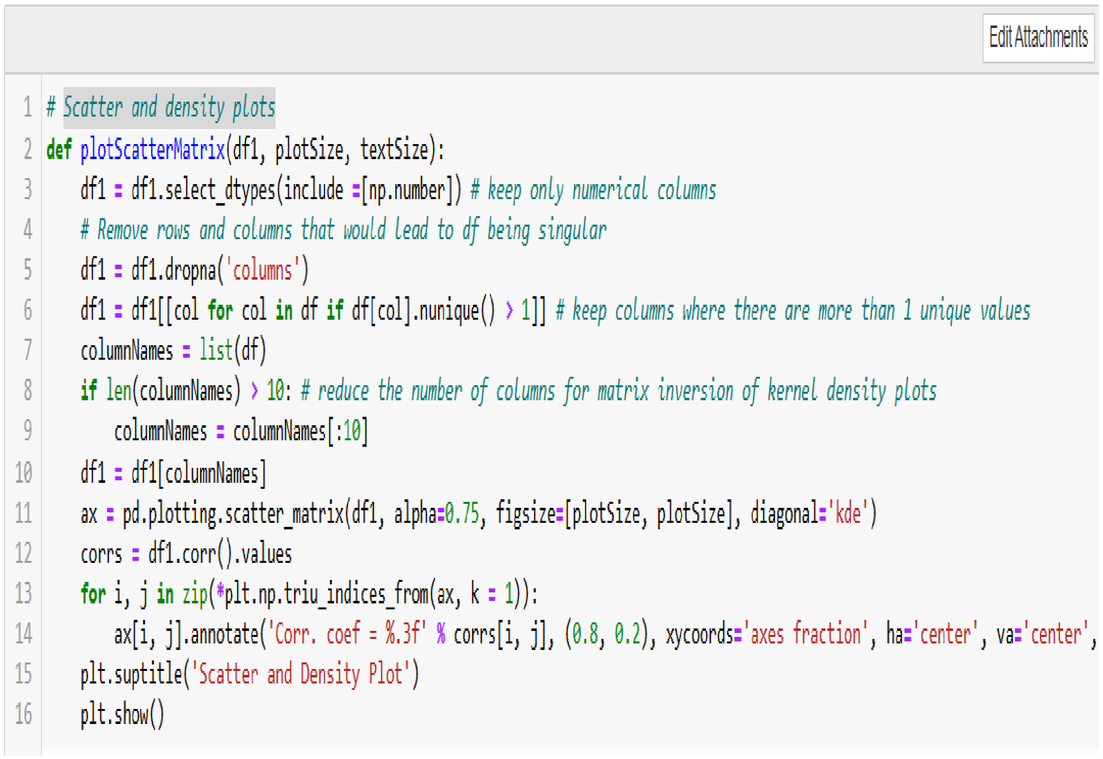
Distribution graphs (histogram/bar graph) of column data



```
# 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 ha
    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()
```

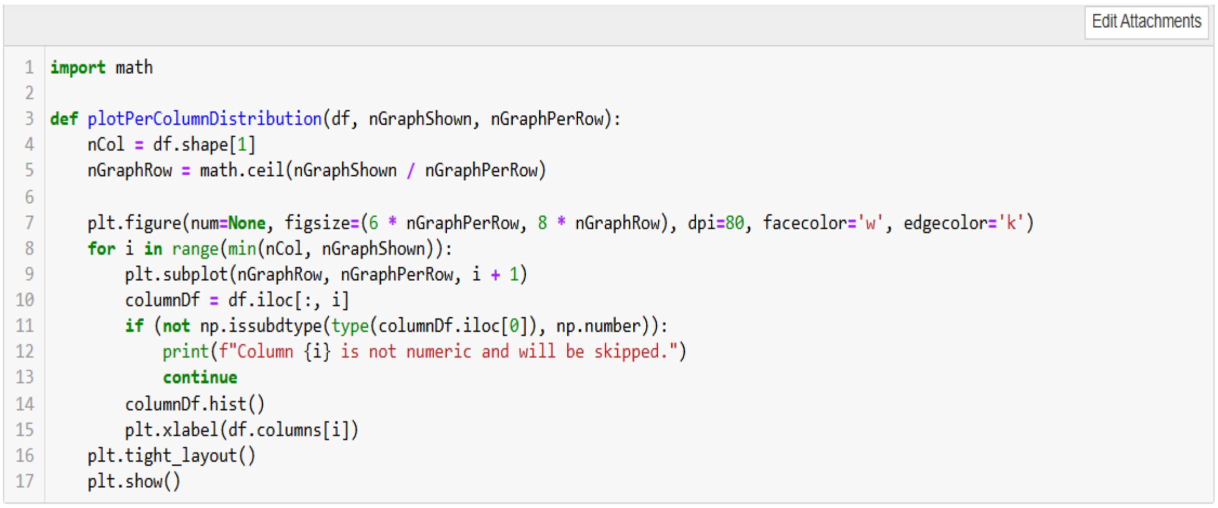
Figure 13: Distribution graphs

Scatter and density plots



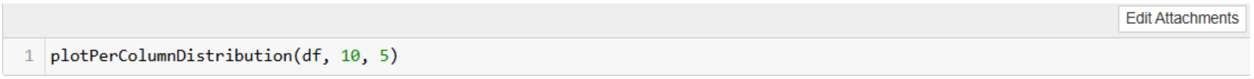
```
# Scatter and density plots
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique()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

Figure 14: Scatter and density plots

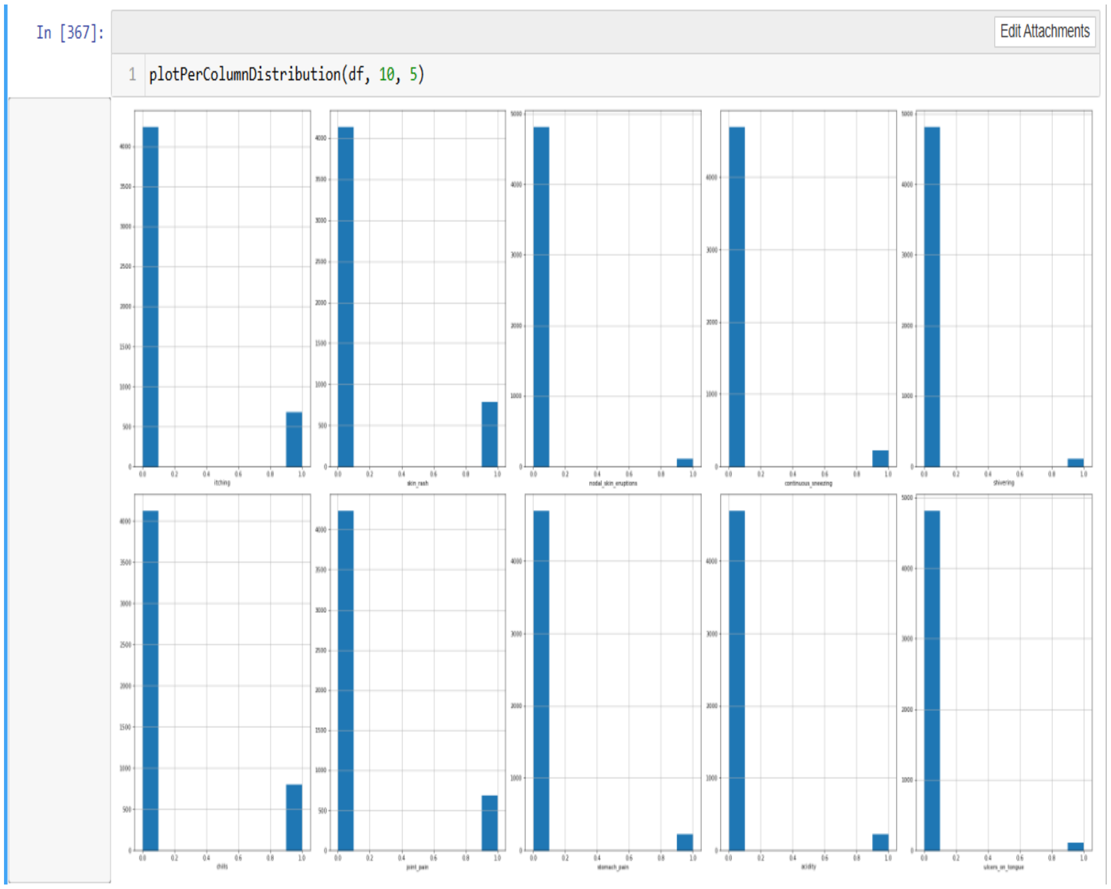


```python
import math

def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nCol = df.shape[1]
    nGraphRow = math.ceil(nGraphShown / 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)):
            print(f"Column {i} is not numeric and will be skipped.")
            continue
        columnDf.hist()
        plt.xlabel(df.columns[i])
    plt.tight_layout()
    plt.show()
```

Figure 15: Plot

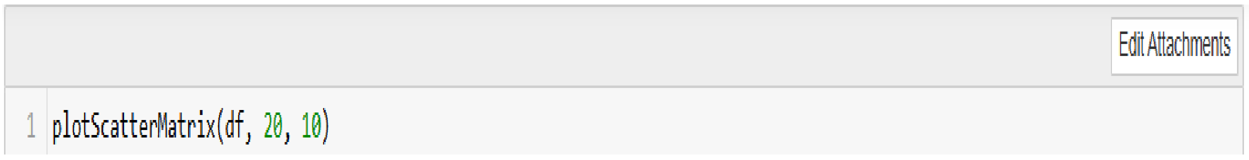


```python
plotPerColumnDistribution(df, 10, 5)
```



```
In [367]:  plotPerColumnDistribution(df, 10, 5)
```

Figure 16: Graph



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

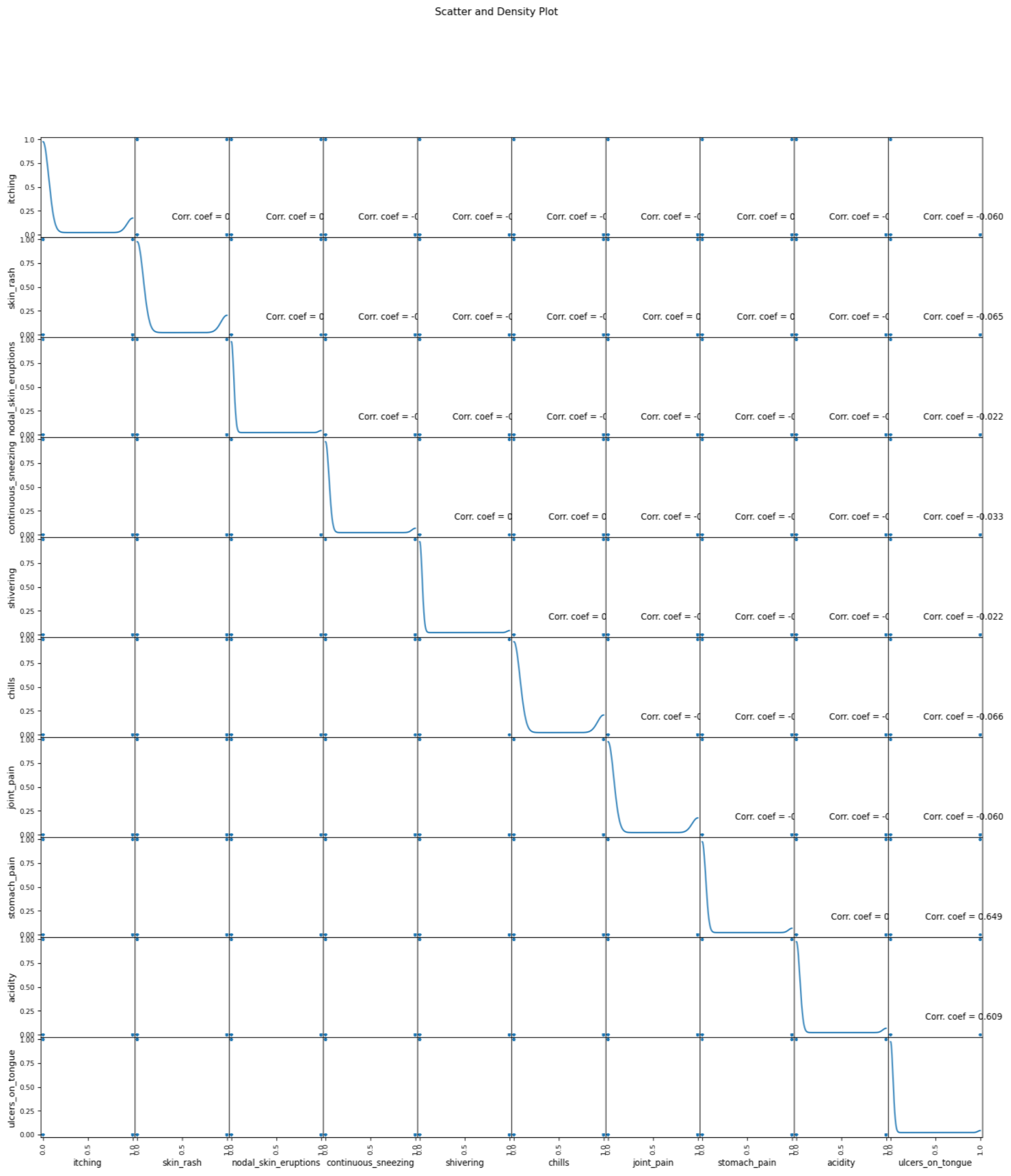
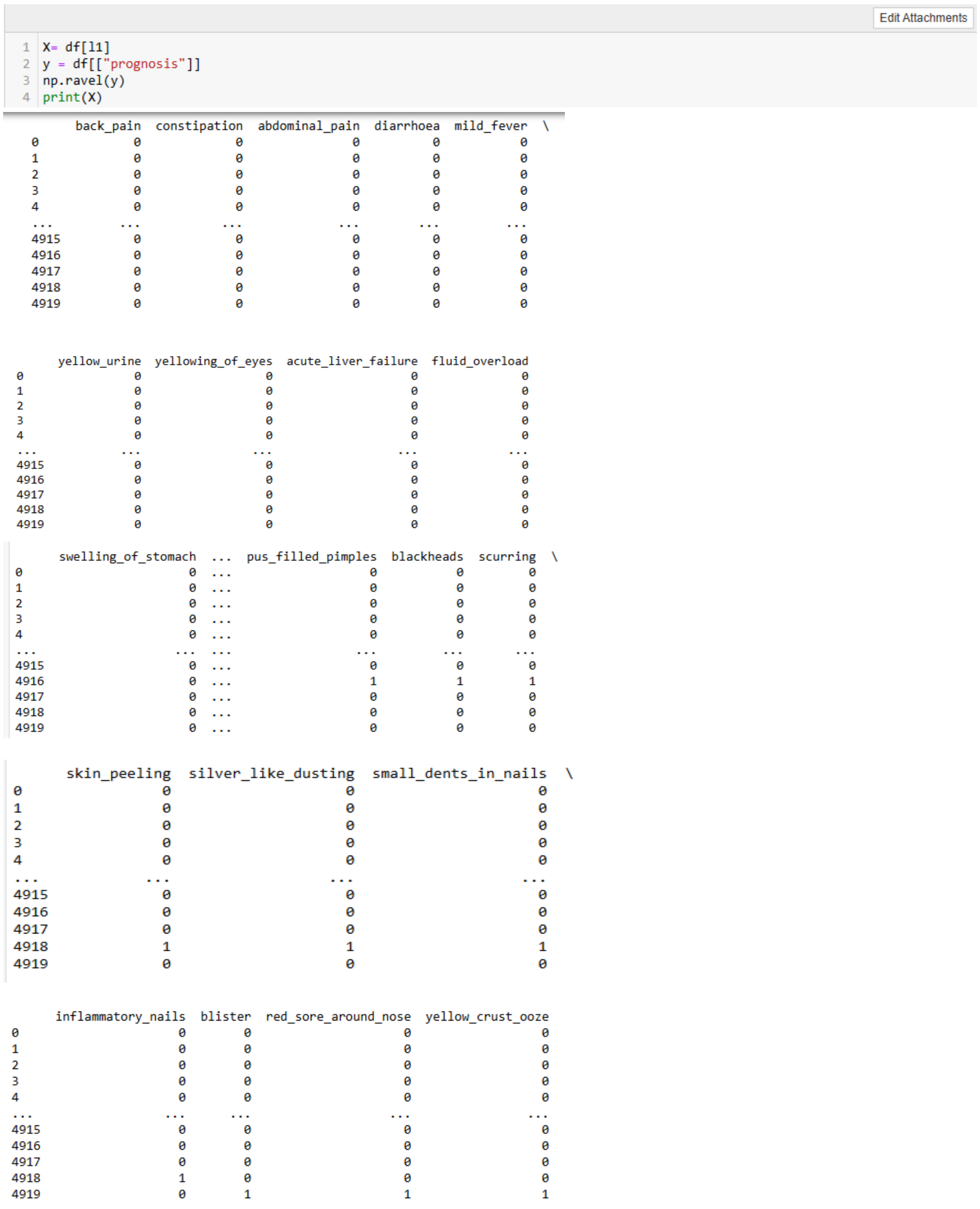


Figure 17: Matrix



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

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

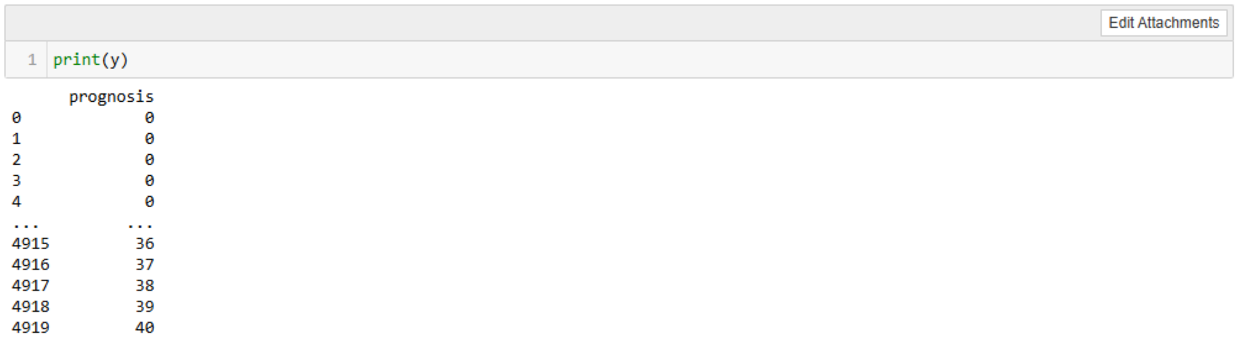
```
      yellow_urine  yellowing_of_eyes  acute_liver_failure  fluid_overload
0                0                  0                    0               0
1                0                  0                    0               0
2                0                  0                    0             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

```
      swelling_of_stomach  ...  pus_filled_pimples  blackheads  scurring  \
0                       0  ...                   0           0         0
1                       0  ...                   0           0         0
2                       0  ...                   0           0         0
3                       0  ...                   0           0         0
4                       0  ...                   0           0         0
...                   ...  ...                 ...         ...       ...
4915                    0  ...                   0           0         0
4916                    0  ...                   1           1         1
4917                    0  ...                   0           0         0
4918                    0  ...                   0           0         0
4919                    0  ...                   0           0         0
```

```
      skin_peeling  silver_like_dusting  small_dents_in_nails  \
0                0                    0                     0
1                0                    0                     0
2                0                    0                     0
3                0                    0                     0
4                0                    0                     0
...            ...                  ...                   ...
4915             0                    0                     0
4916             0                    0                     0
4917             0                    0                     0
4918             1                    1                     1
4919             0                    0                     0
```

```
      inflammatory_nails  blister  red_sore_around_nose  yellow_crust_ooze
0                      0        0                     0                  0
1                      0        0                     0                  0
2                      0        0                     0                  0
3                      0        0                     0                  0
4                      0        0                     0                  0
...                  ...      ...                   ...                ...
4915                   0        0                     0                  0
4916                   0        0                     0                  0
4917                   0        0                     0                  0
4918                   1        0                     0                  0
4919                   0        1                     1                  1
```

Figure 18: Prognosis

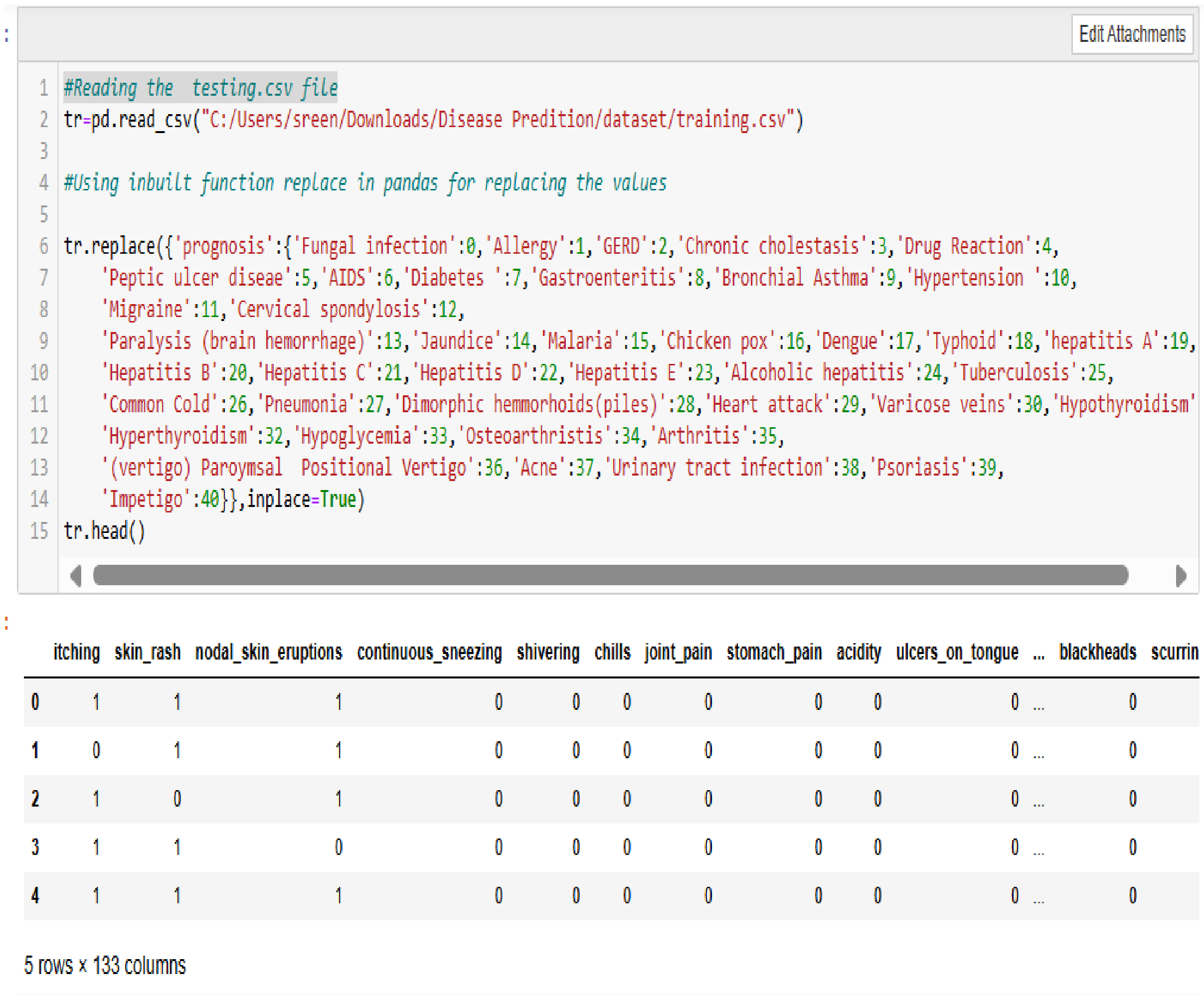

Edit Attachments

```
print(y)
```

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

Figure 19: Y axis

Reading the testing.csv file

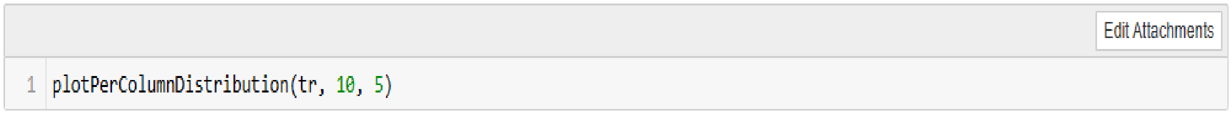

Edit Attachments

```
#Reading the  testing.csv file
tr=pd.read_csv("C:/Users/sreen/Downloads/Disease Predition/dataset/training.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'
    '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)
tr.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads	scurrin
0	1	1	1	0	0	0	0	0	0	0	...	0	
1	0	1	1	0	0	0	0	0	0	0	...	0	
2	1	0	1	0	0	0	0	0	0	0	...	0	
3	1	1	0	0	0	0	0	0	0	0	...	0	
4	1	1	1	0	0	0	0	0	0	0	...	0	

5 rows × 133 columns

Figure 20: Reading the Test.csv



```
plotPerColumnDistribution(tr, 10, 5)
```

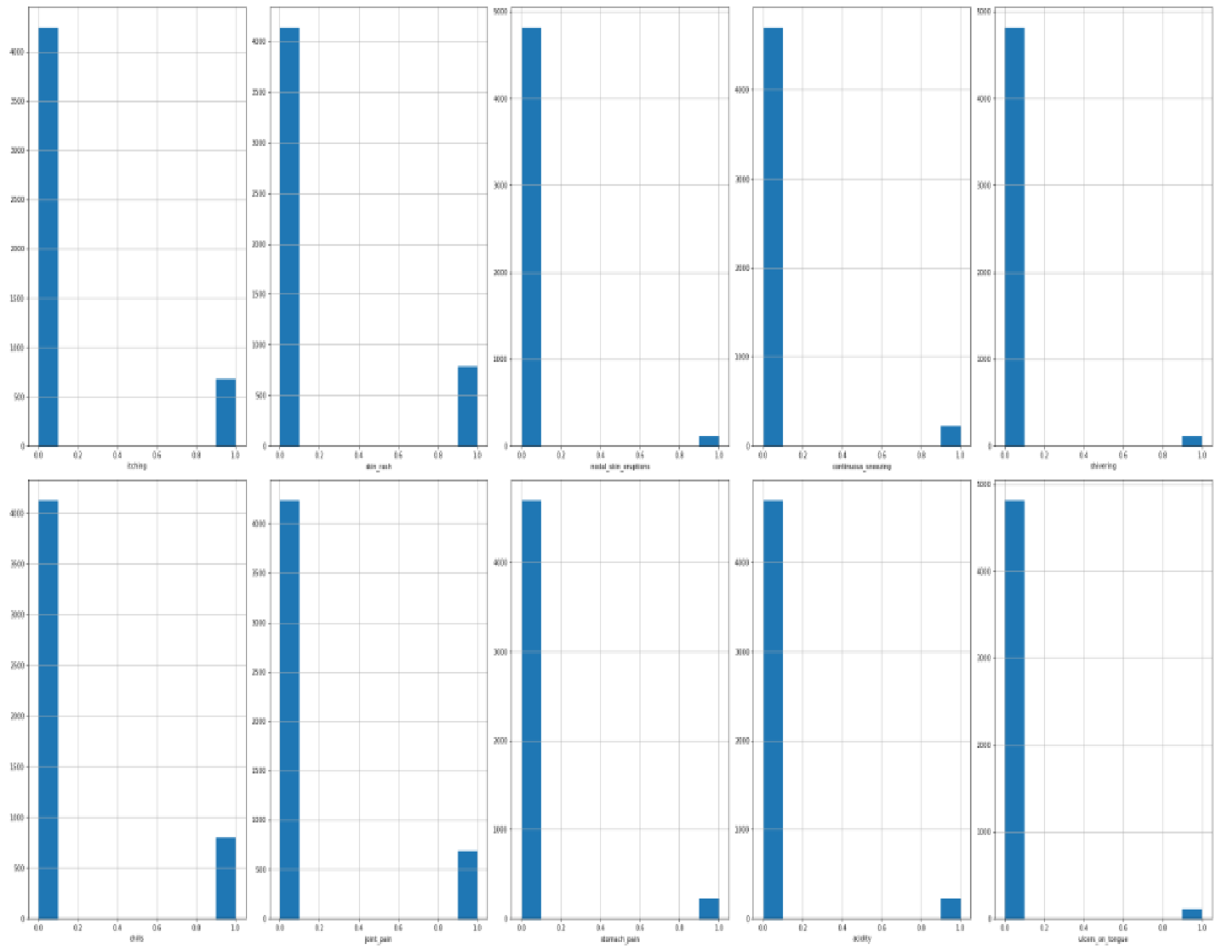
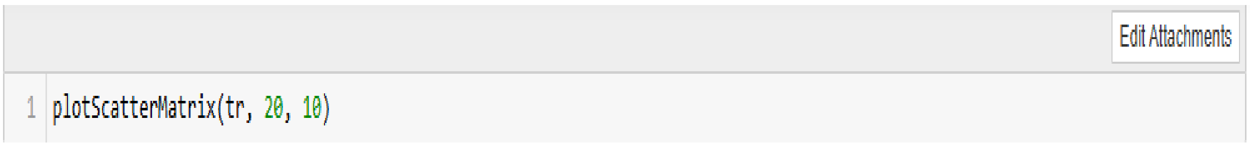


Figure 21: Graph



```
plotScatterMatrix(tr, 20, 10)
```

Figure 22: To Plot Matrix

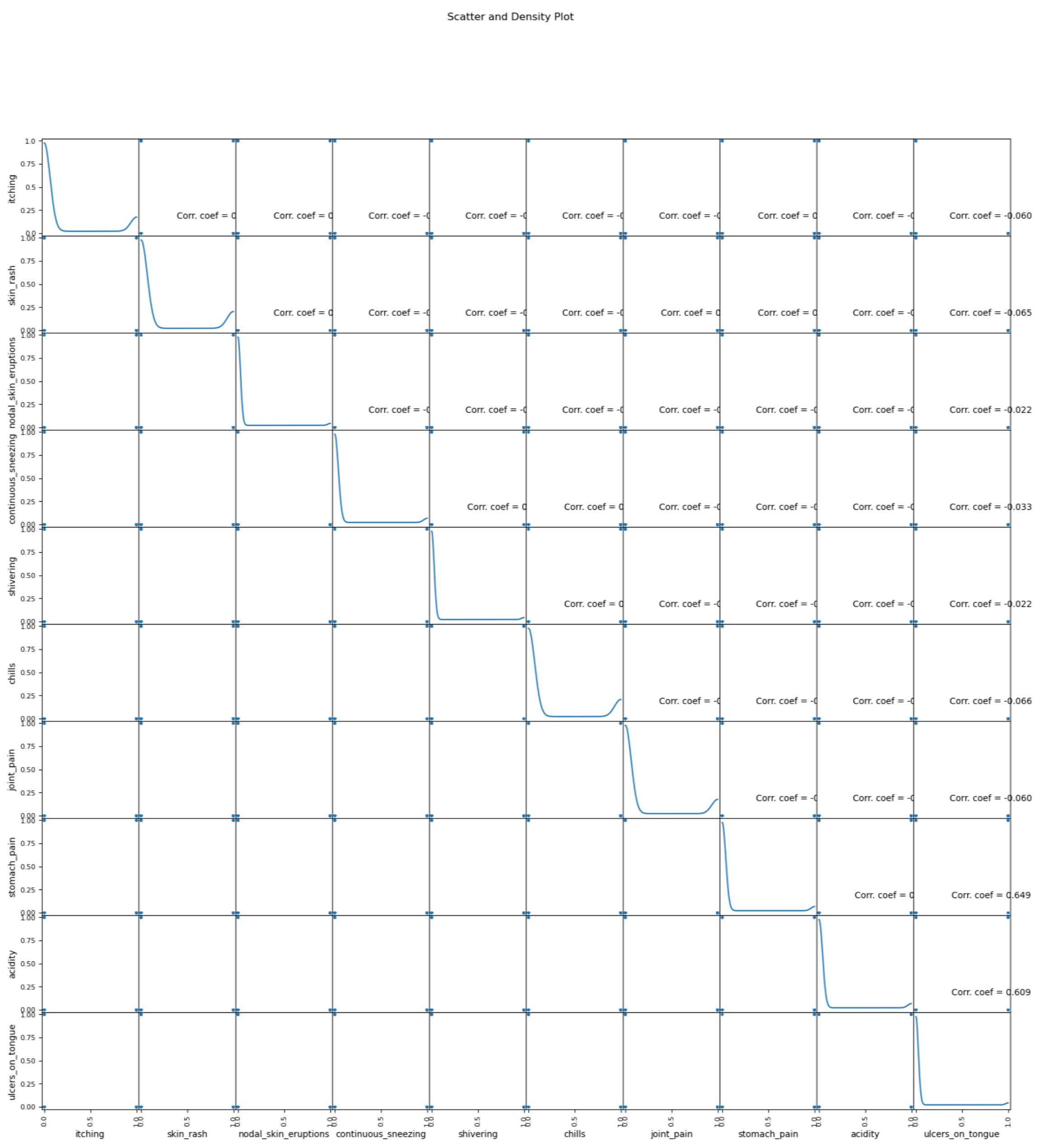
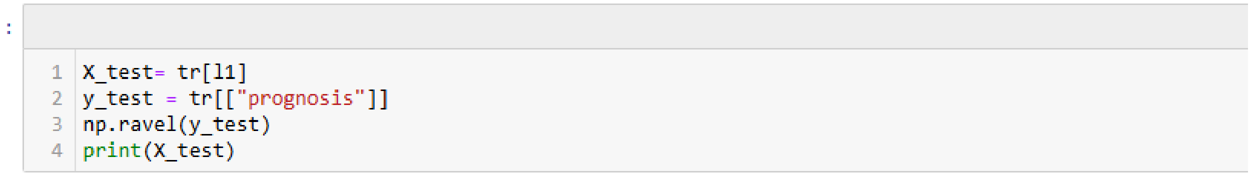
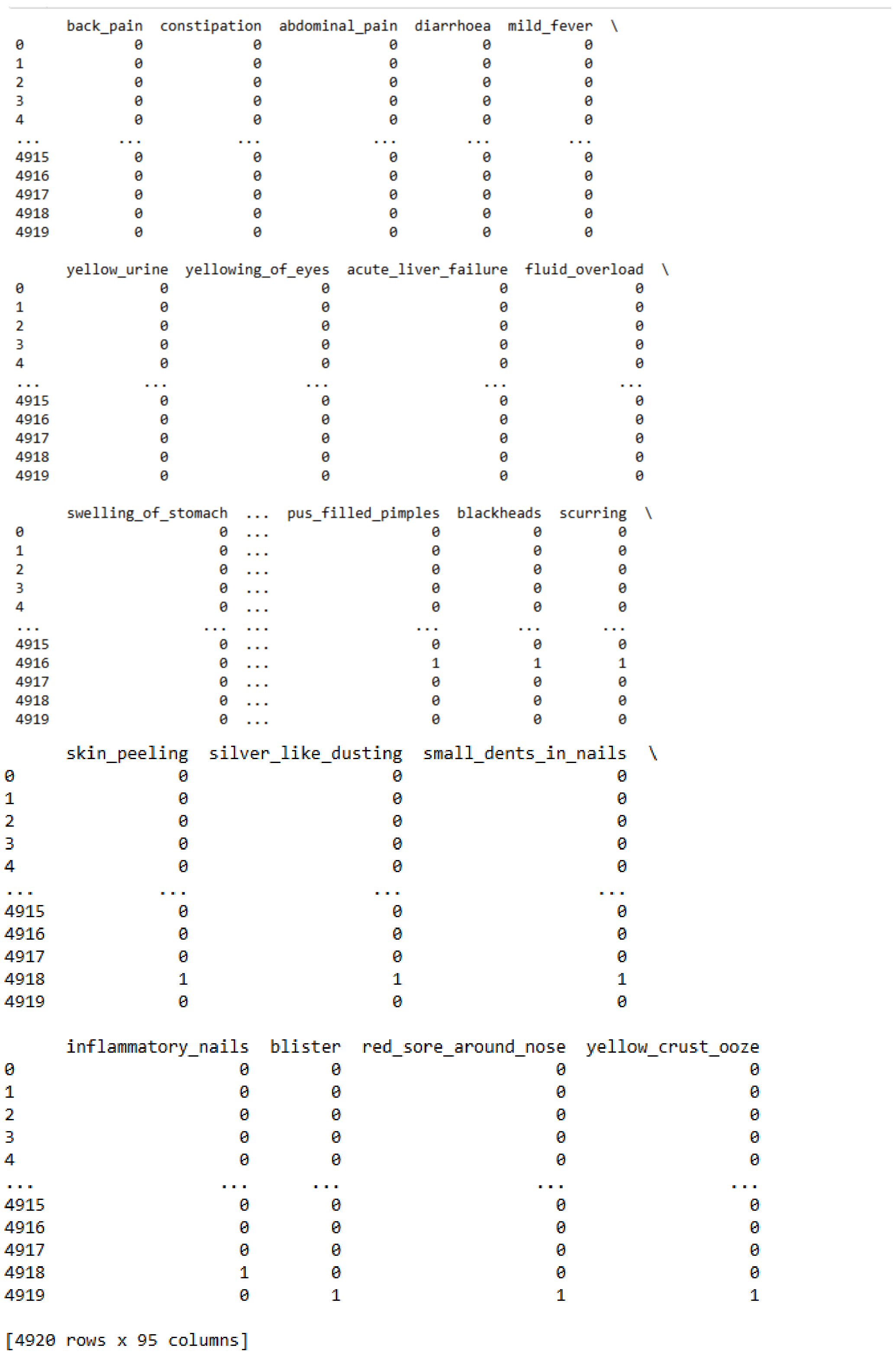


Figure 23: Matrix



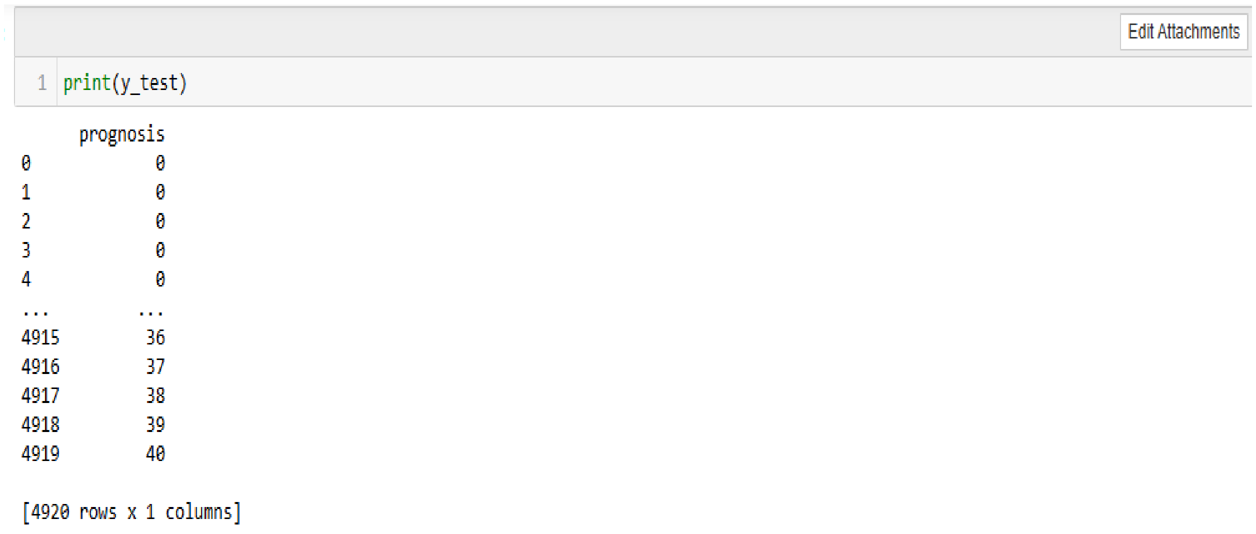
```
X_test= tr[l1]
y_test = tr[["prognosis"]]
np.ravel(y_test)
print(X_test)
```

Figure 24: Prognosis



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

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

      swelling_of_stomach  ...  pus_filled_pimples  blackheads  scurring  \
0                       0  ...                   0           0         0
1                       0  ...                   0           0         0
2                       0  ...                   0           0         0
3                       0  ...                   0           0         0
4                       0  ...                   0           0         0
...                   ...  ...                 ...         ...       ...
4915                    0  ...                   0           0         0
4916                    0  ...                   1           1         1
4917                    0  ...                   0           0         0
4918                    0  ...                   0           0         0
4919                    0  ...                   0           0         0

      skin_peeling  silver_like_dusting  small_dents_in_nails  \
0                0                    0                     0
1                0                    0                     0
2                0                    0                     0
3                0                    0                     0
4                0                    0                     0
...            ...                  ...                   ...
4915             0                    0                     0
4916             0                    0                     0
4917             0                    0                     0
4918             1                    1                     1
4919             0                    0                     0

      inflammatory_nails  blister  red_sore_around_nose  yellow_crust_ooze
0                      0        0                     0                  0
1                      0        0                     0                  0
2                      0        0                     0                  0
3                      0        0                     0                  0
4                      0        0                     0                  0
...                  ...      ...                   ...                ...
4915                   0        0                     0                  0
4916                   0        0                     0                  0
4917                   0        0                     0                  0
4918                   1        0                     0                  0
4919                   0        1                     1                  1

[4920 rows x 95 columns]
```

Figure 25: Prognosis



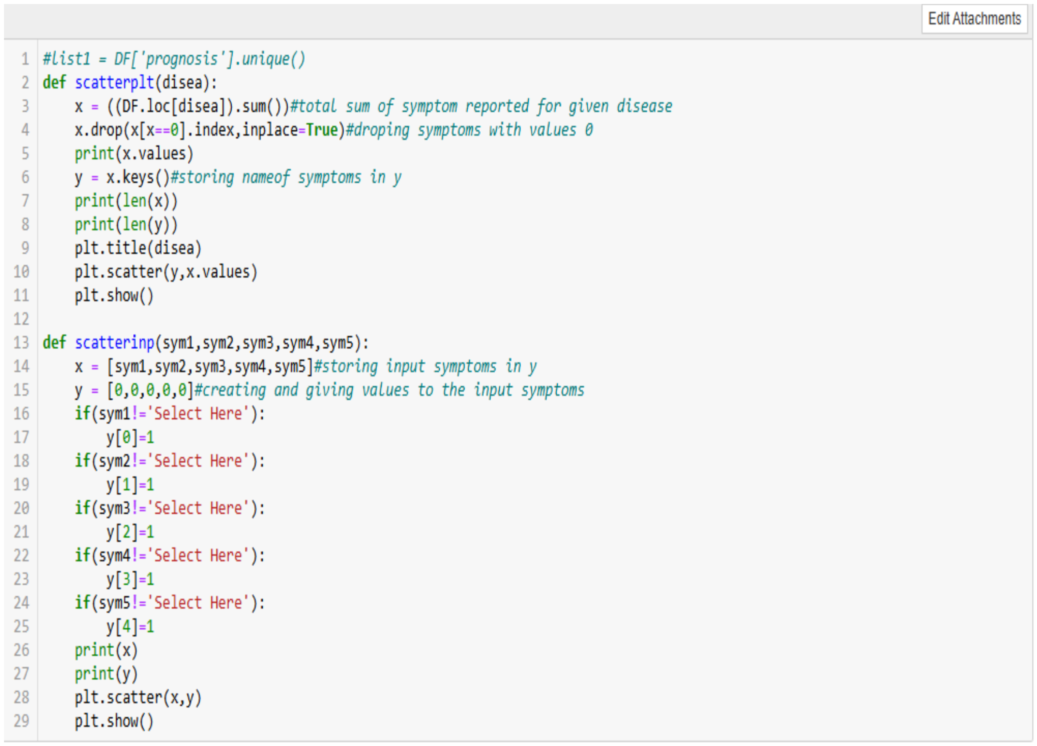
```
print(y_test)
```

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

[4920 rows x 1 columns]
```

Figure 26: Test Y axis

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

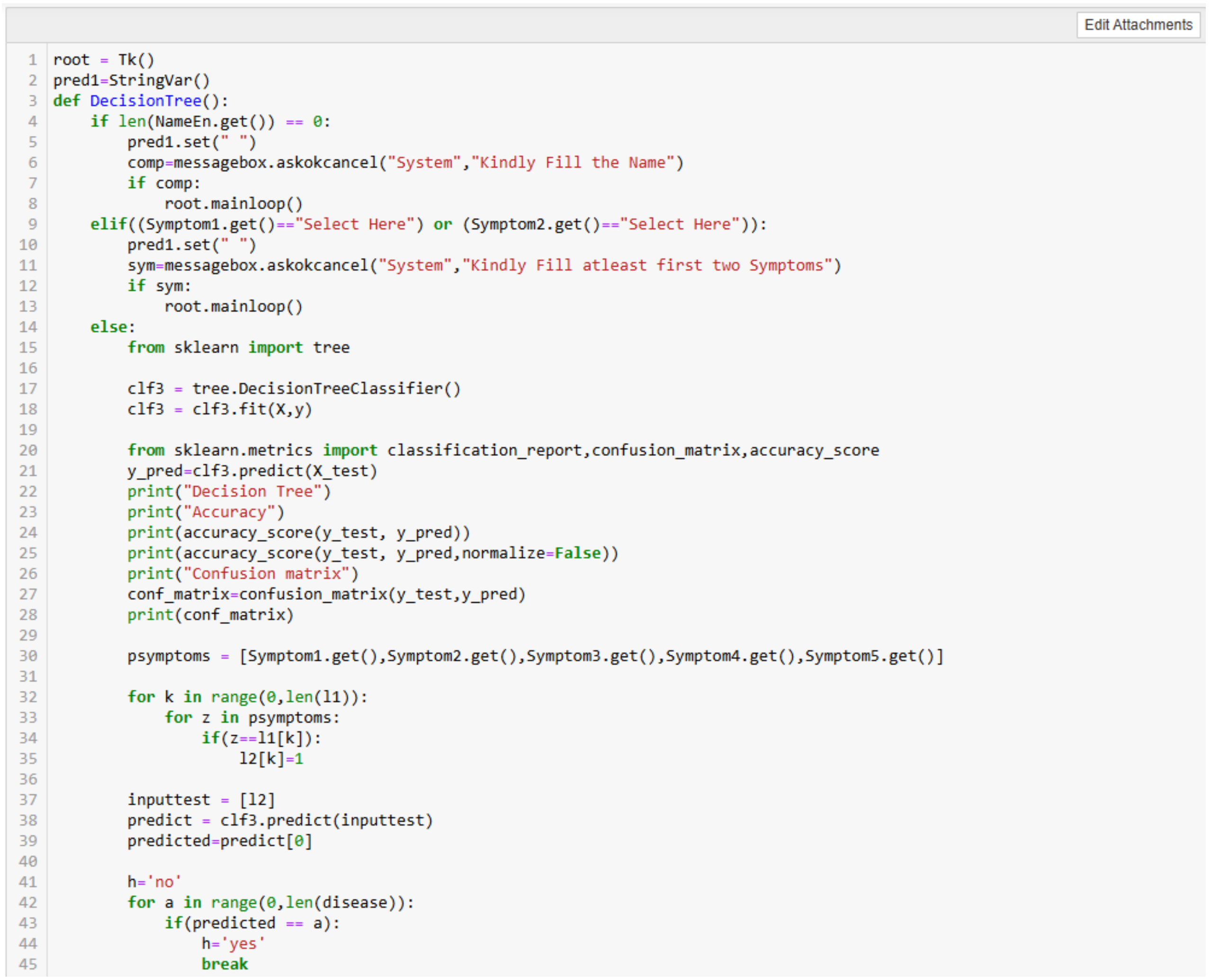
- Decision Tree algorithm
- Random Forest algorithm
- Naive Bayes algorithm



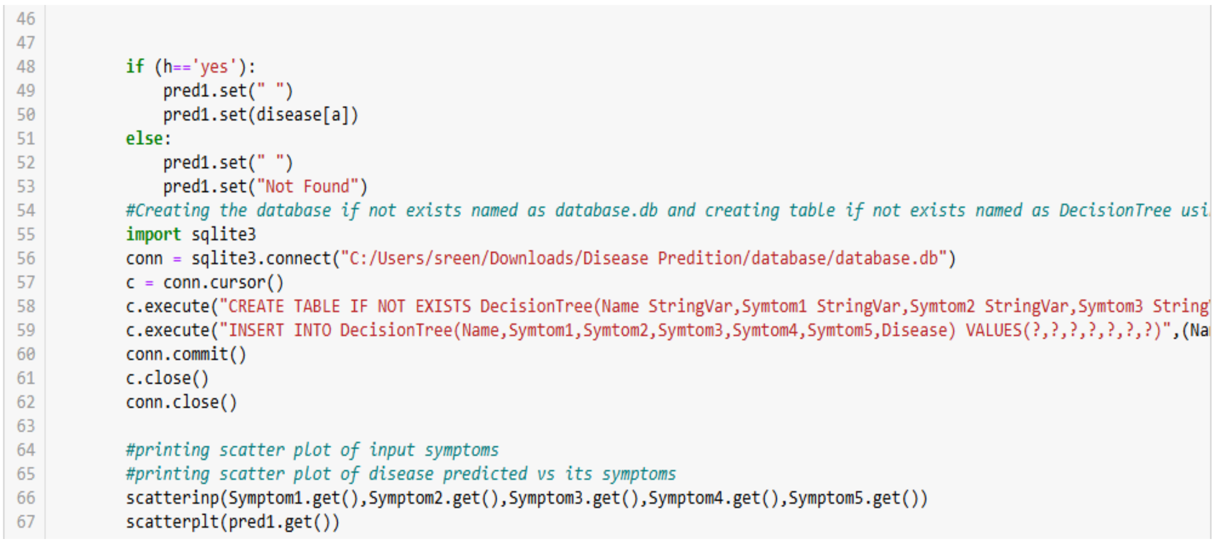
```python
#list1 = DF['prognosis'].unique()
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())#total sum of symptom reported for given disease
    x.drop(x[x==0].index,inplace=True)#droping symptoms with values 0
    print(x.values)
    y = x.keys()#storing nameof symptoms in y
    print(len(x))
    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()
```

Figure 27: Algorithm

DecisionTree



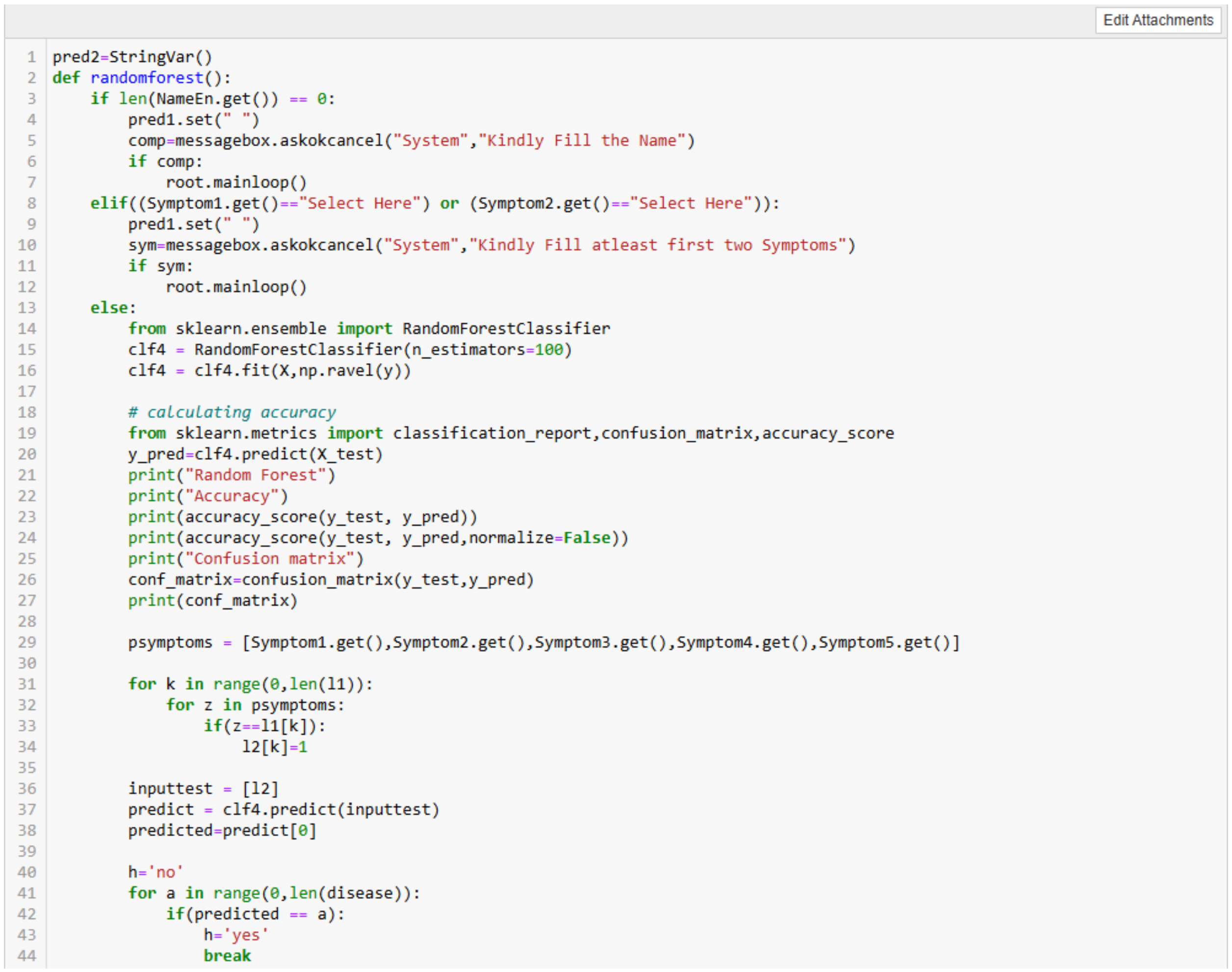
```python
root = Tk()
pred1=StringVar()
def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        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
```



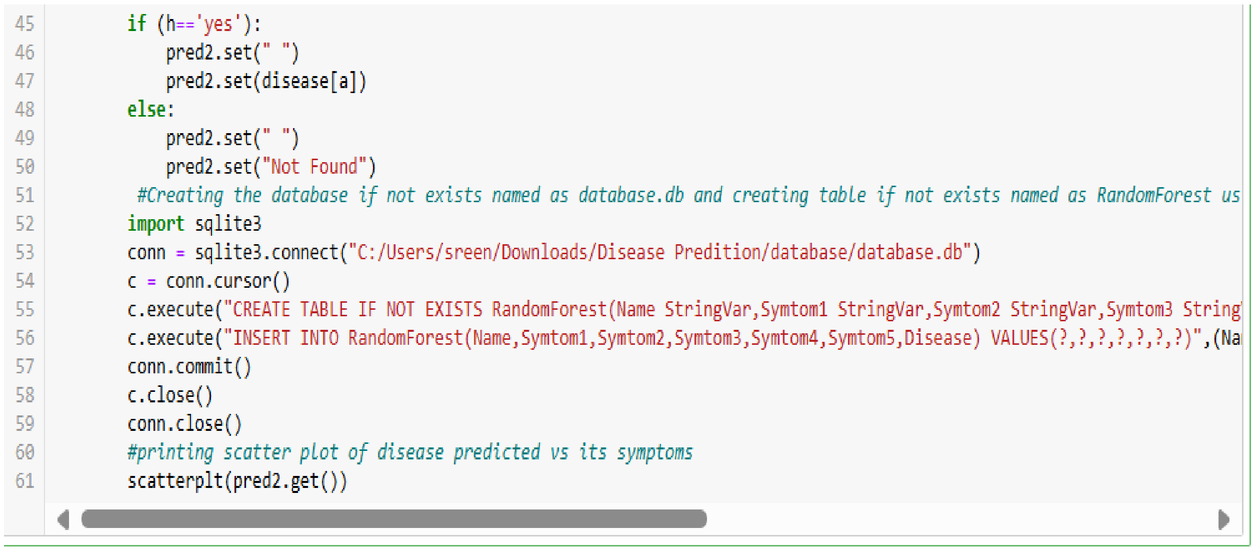
```python


        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 usi
        import sqlite3
        conn = sqlite3.connect("C:/Users/sreen/Downloads/Disease Predition/database/database.db")
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 String
        c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(Na
        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())
```

Figure 28: Decision Tree Algorithm

Randomforest



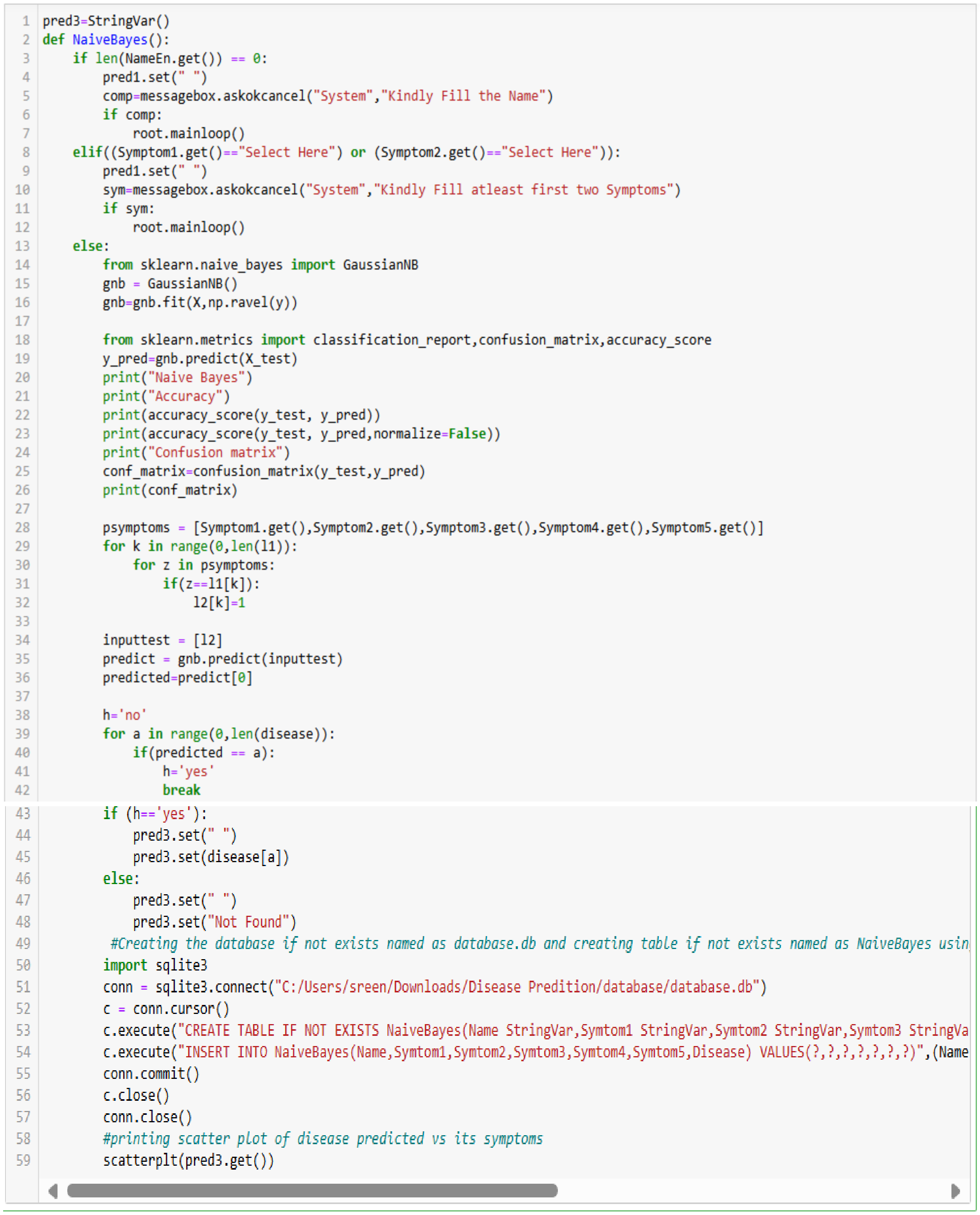
```python
pred2=StringVar()
def randomforest():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        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
```



```python
        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 us
        import sqlite3
        conn = sqlite3.connect("C:/Users/sreen/Downloads/Disease Predition/database/database.db")
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 String
        c.execute("INSERT INTO RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(Na
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms
        scatterplt(pred2.get())
```

Figure 29: Randomforest Algorithm

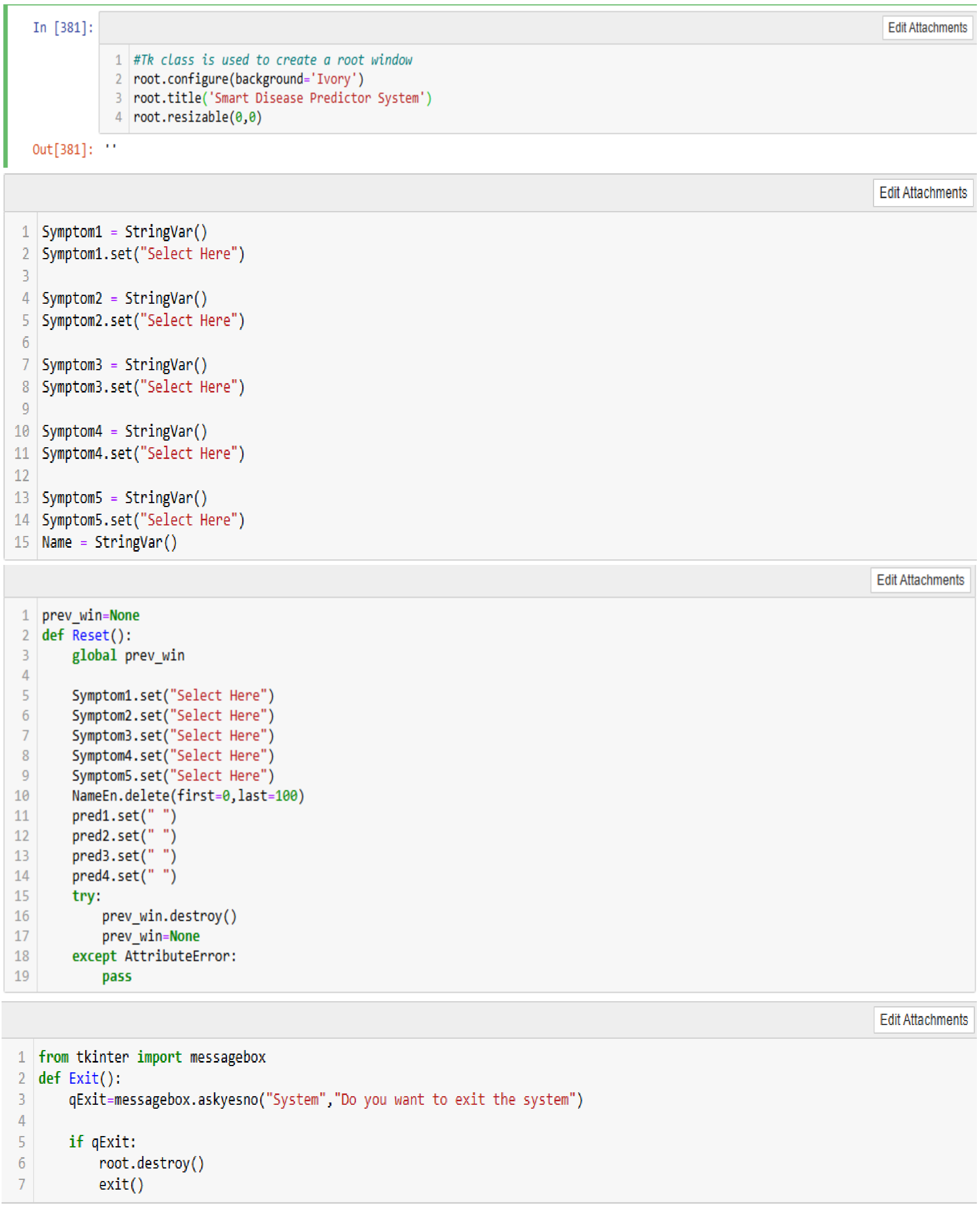
NaiveBayes



```python
pred3=StringVar()
def NaiveBayes():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.naive_bayes import GaussianNB
        gnb = GaussianNB()
        gnb=gnb.fit(X,np.ravel(y))

        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=gnb.predict(X_test)
        print("Naive Bayes")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
        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")
         #Creating the database if not exists named as database.db and creating table if not exists named as NaiveBayes usin
        import sqlite3
        conn = sqlite3.connect("C:/Users/sreen/Downloads/Disease Predition/database/database.db")
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVa
        c.execute("INSERT INTO NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(Name
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms
        scatterplt(pred3.get())
```

Figure 30: Decision NaiveBayes Algorithm

Tk class is used to create a root window



In [381]:

Edit Attachments

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

Out[381]: ''

Edit Attachments

```
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()
```

Edit Attachments

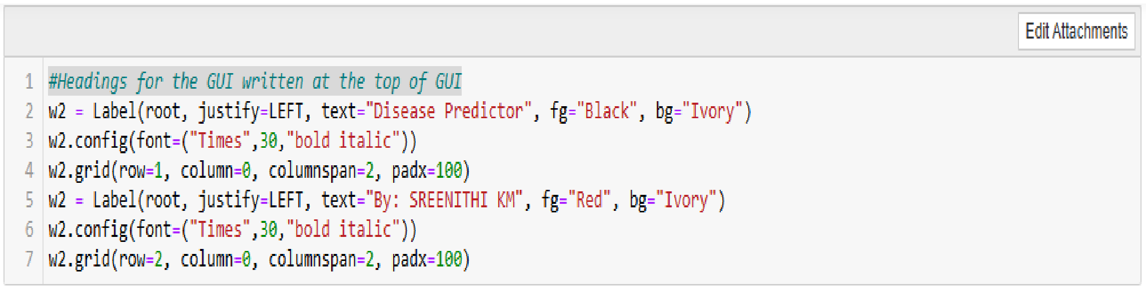
```
prev_win=None
def Reset():
    global prev_win

    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
```

Edit Attachments

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

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

Figure 31: Tk class

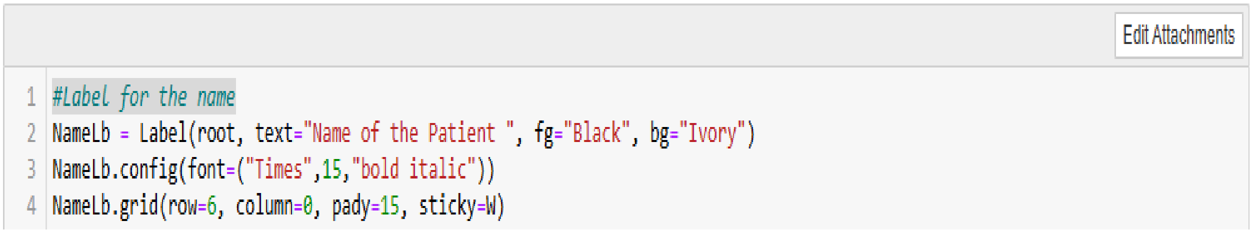
Headings for the GUI written at the top of GUI


Edit Attachments

```python
#Headings for the GUI written at the top of GUI
w2 = Label(root, justify=LEFT, text="Disease Predictor", fg="Black", bg="Ivory")
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2 = Label(root, justify=LEFT, text="By: SREENITHI KM", fg="Red", bg="Ivory")
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=2, column=0, columnspan=2, padx=100)
```

Figure 32: Headings for the GUI

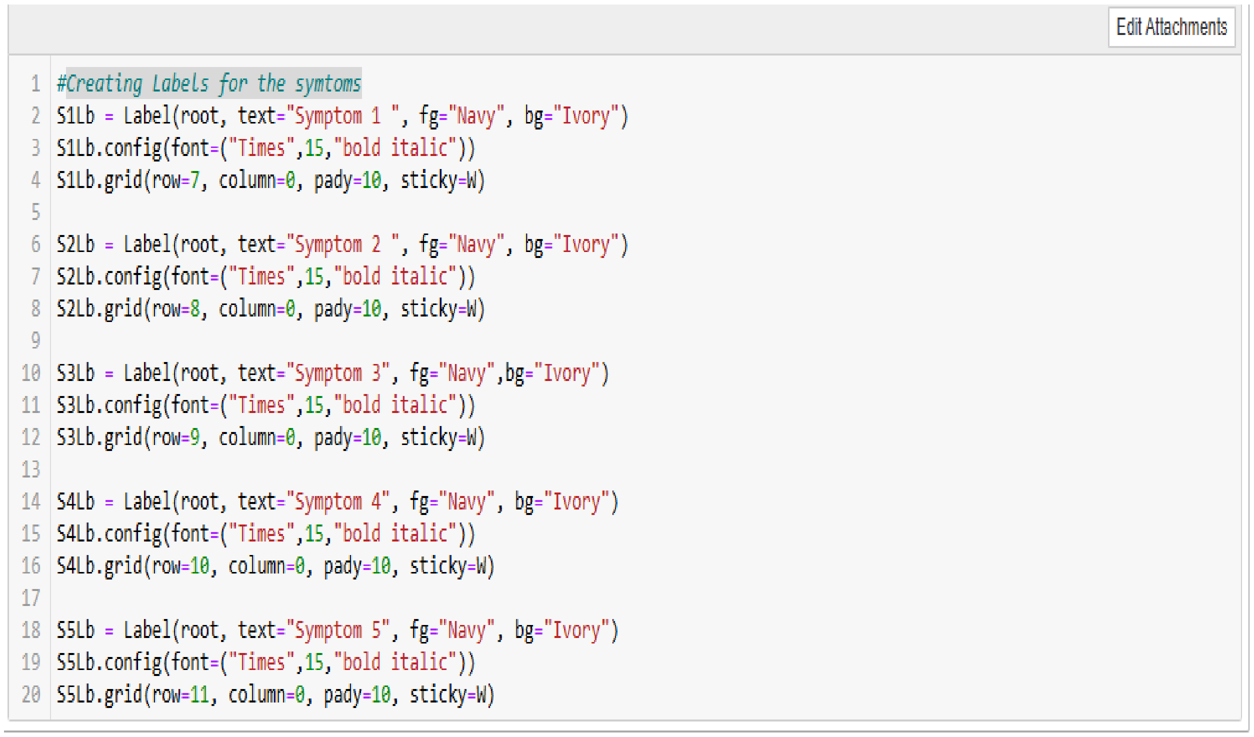
Label for the name


Edit Attachments

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

Figure 33: Label for the name

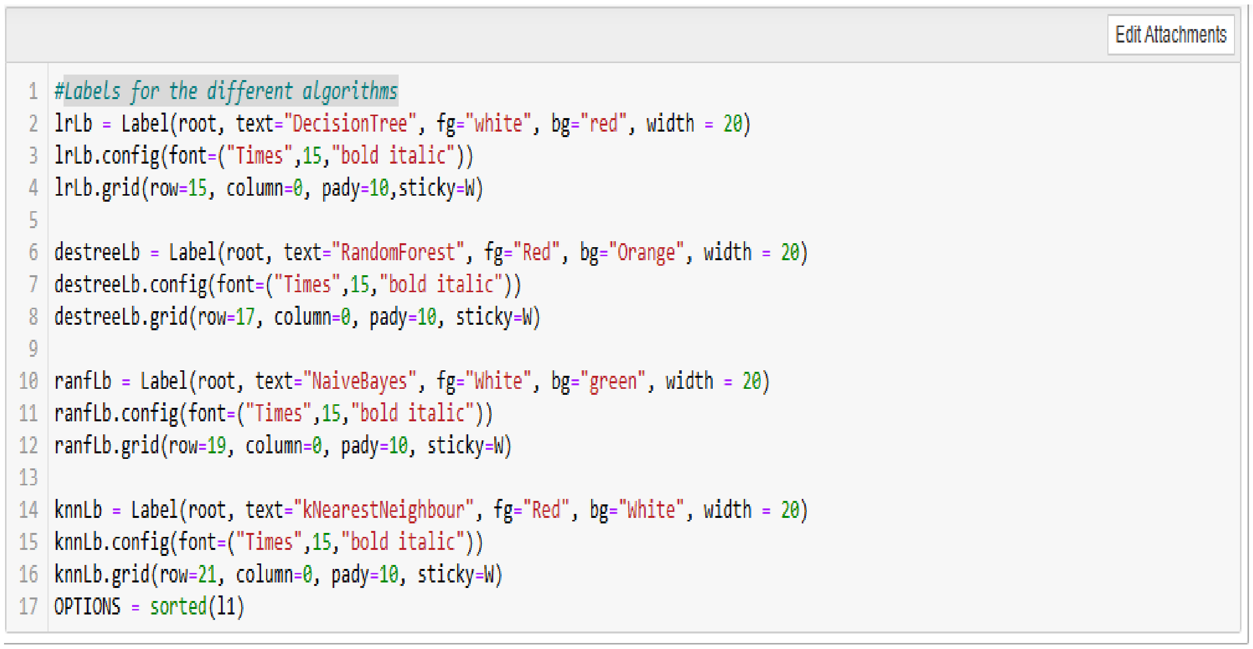
Creating Labels for the symtoms


Edit Attachments

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

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

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

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

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

Figure 34: Creating Labels for the symptoms

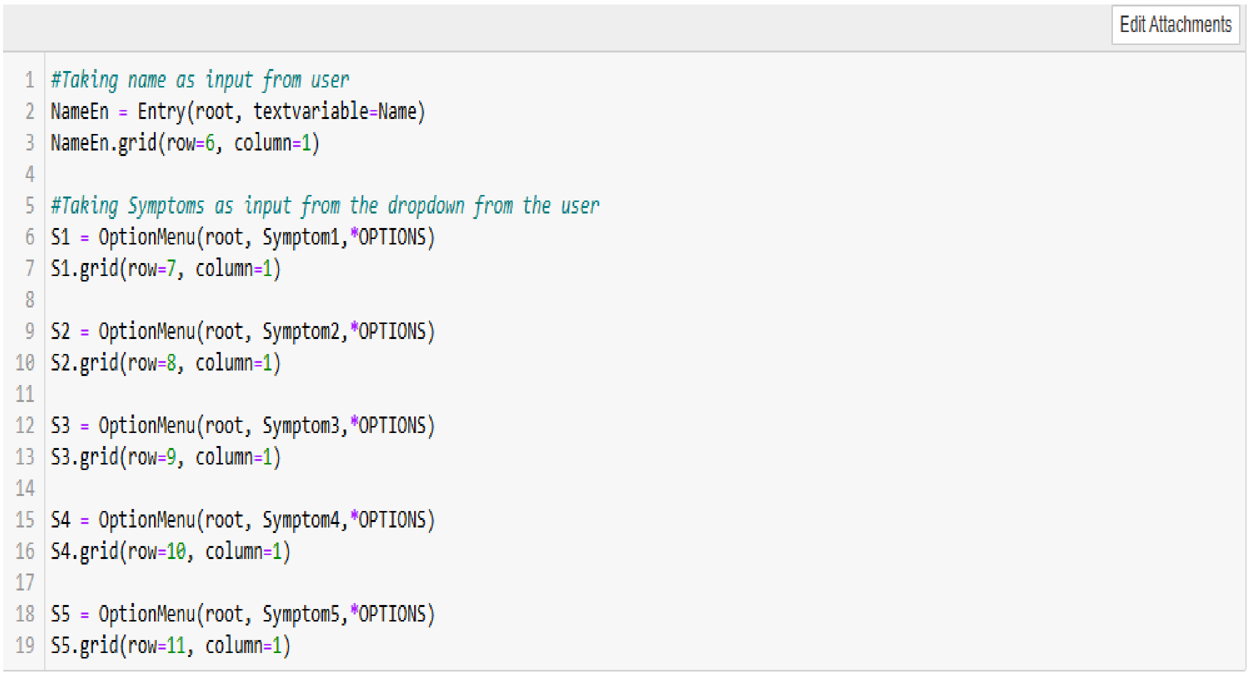
Labels for the different algorithms



```python
#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="White", width = 20)
knnLb.config(font=("Times",15,"bold italic"))
knnLb.grid(row=21, column=0, pady=10, sticky=W)
OPTIONS = sorted(l1)
```

Figure 35: Labels for the different algorithms

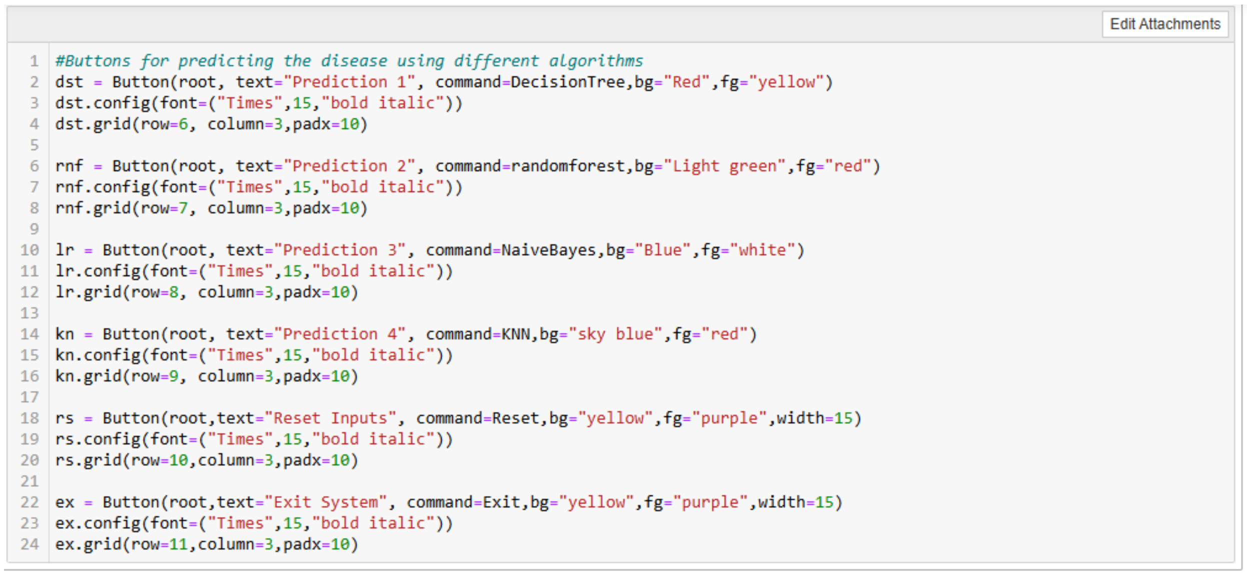
Taking name as input from user



```python
#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)
```

Figure 36: Taking name as input from user

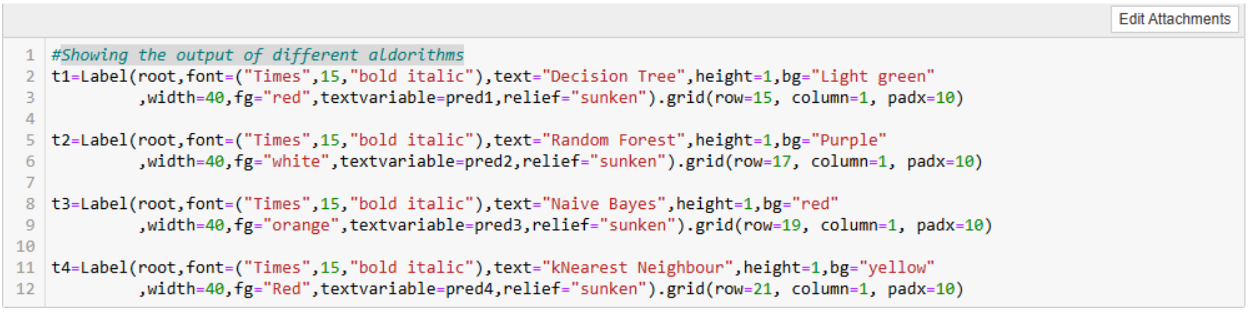
Buttons for predicting the disease using different algorithms


Edit Attachments

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

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

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

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

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

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

Figure 37: Buttons for predicting the disease

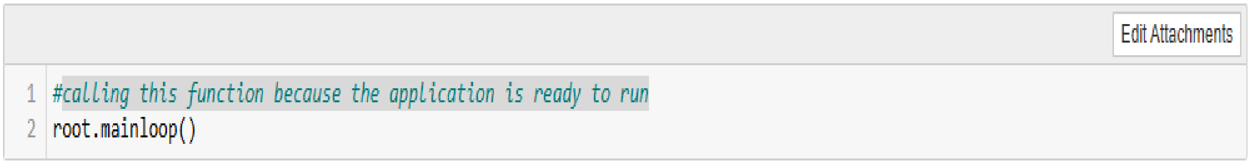
Showing the output of different algorithms


Edit Attachments

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

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

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

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

Figure 38: Showing the output of different algorithms

calling this function because the application is ready to run


Edit Attachments

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

Figure 39: calling this function

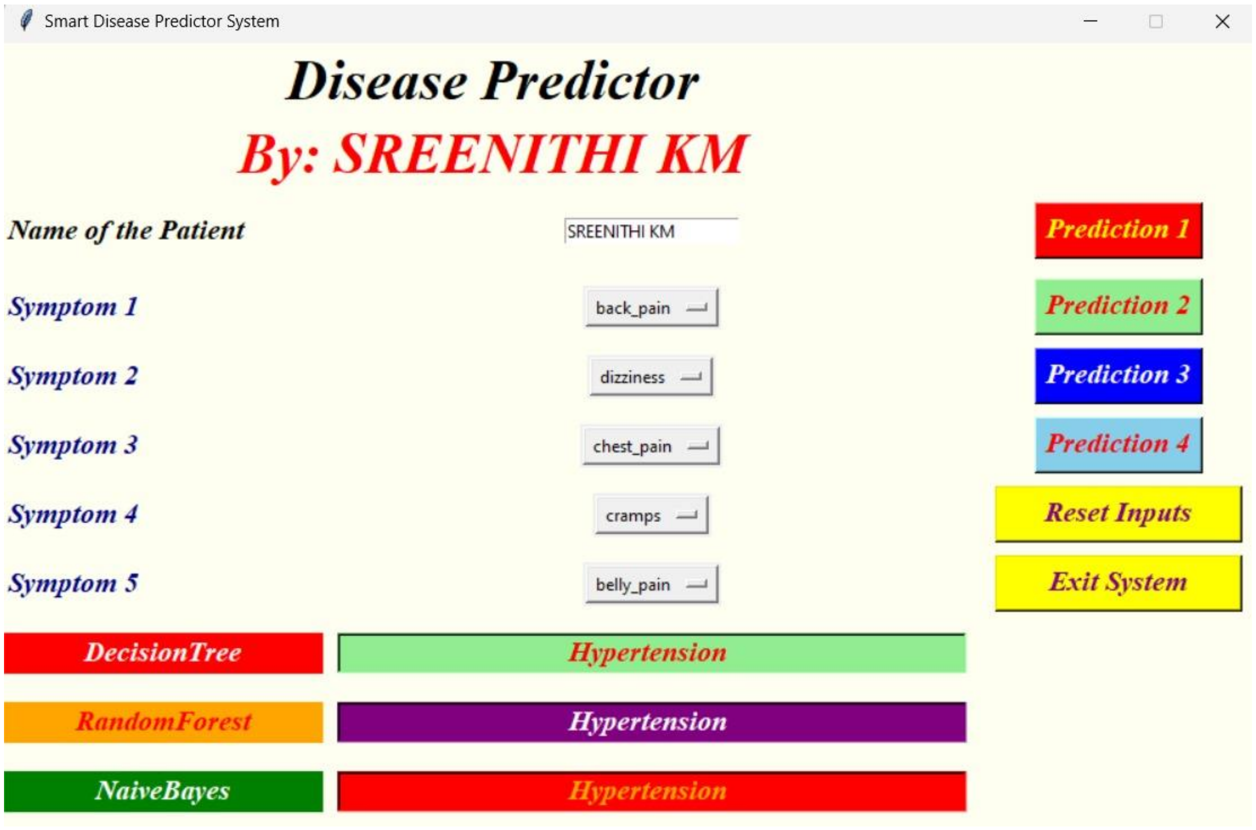
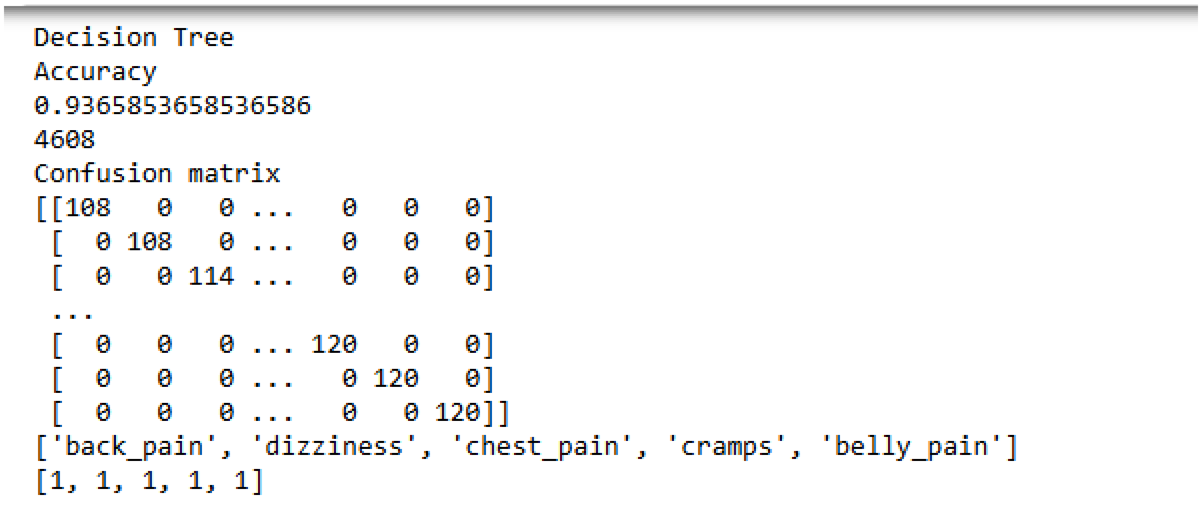


Figure 40: Disease Prediction



```
Decision Tree
Accuracy
0.9365853658536586
4608
Confusion matrix
[[108   0   0 ...   0   0   0]
 [  0 108   0 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ... 120   0   0]
 [  0   0   0 ...   0 120   0]
 [  0   0   0 ...   0   0 120]]
['back_pain', 'dizziness', 'chest_pain', 'cramps', 'belly_pain']
[1, 1, 1, 1, 1]
```

Figure 41: Confusion Matrix

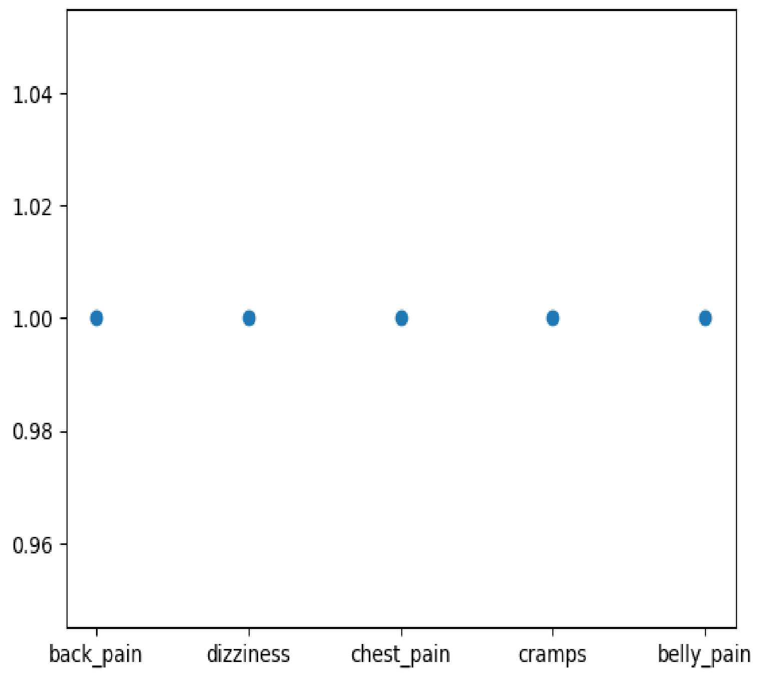


Figure 42: Graph

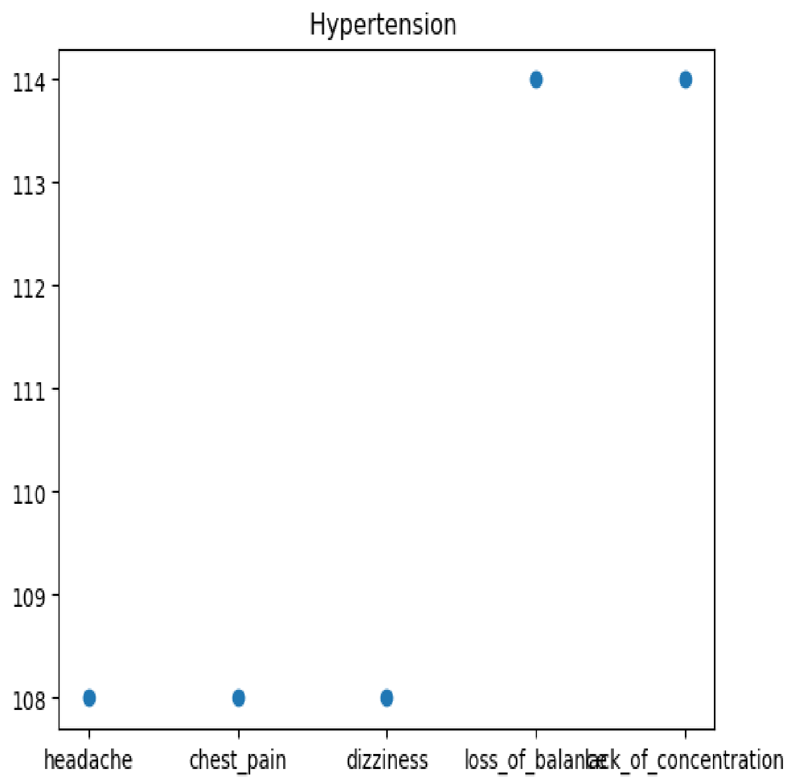
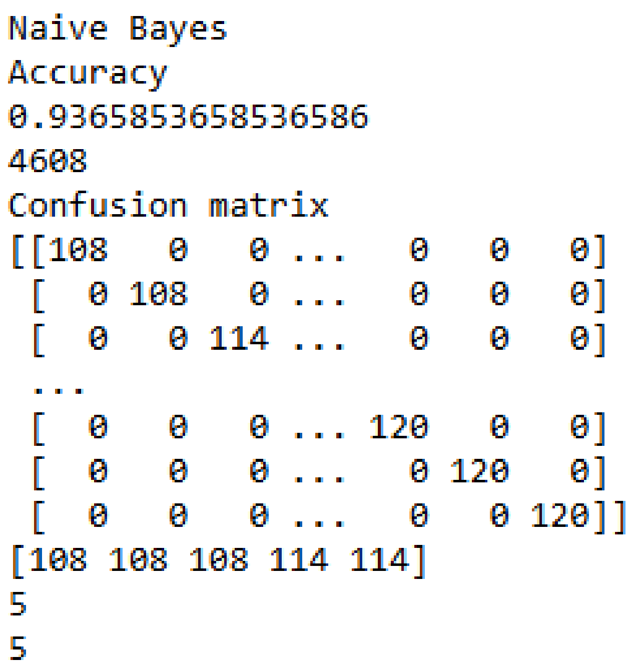


Figure 43: Graph



```
Naive Bayes
Accuracy
0.9365853658536586
4608
Confusion matrix
[[108   0   0 ...   0   0   0]
 [  0 108   0 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ... 120   0   0]
 [  0   0   0 ...   0 120   0]
 [  0   0   0 ...   0   0 120]]
[108 108 108 114 114]
5
5
```

Figure 44: Confusion Matrix

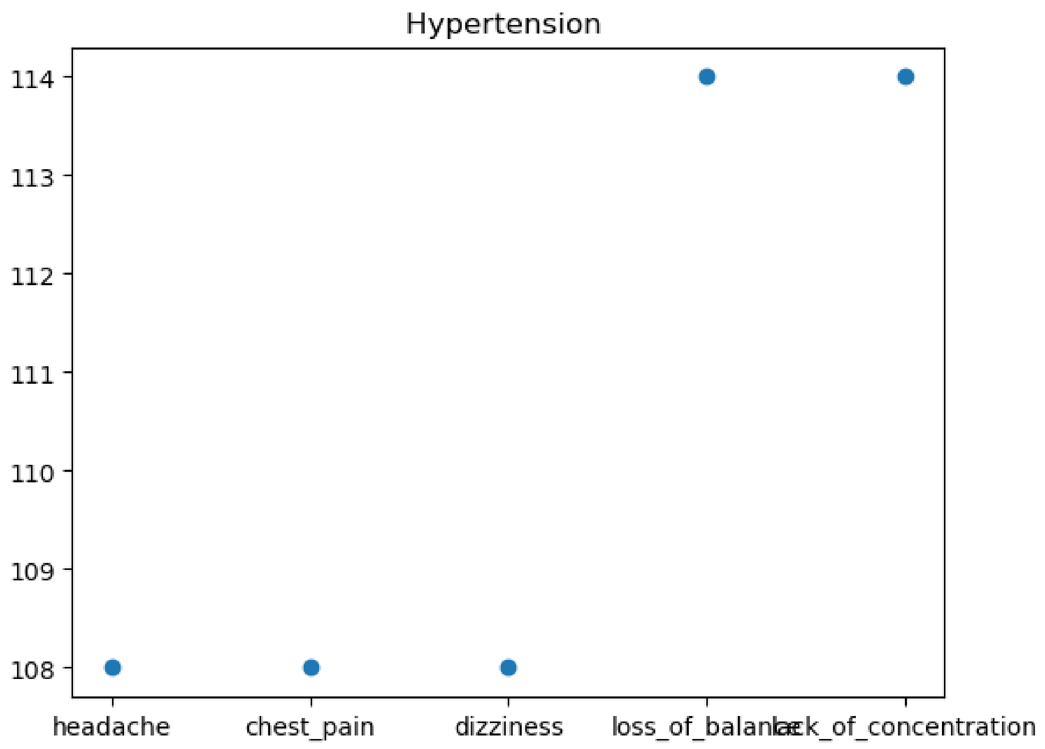
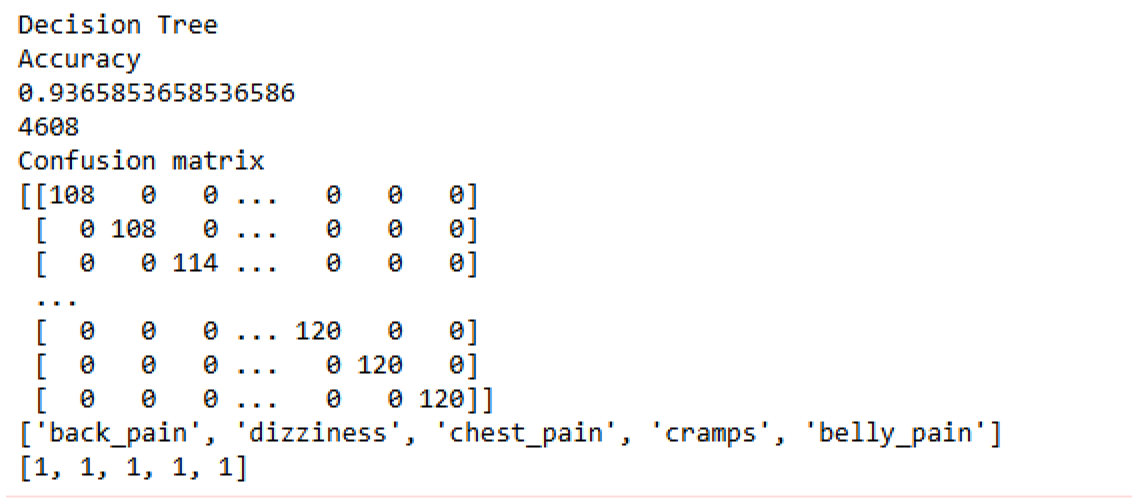


Figure 45: Confusion Matrix



```
Decision Tree
Accuracy
0.9365853658536586
4608
Confusion matrix
[[108   0   0 ...   0   0   0]
 [  0 108   0 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ... 120   0   0]
 [  0   0   0 ...   0 120   0]
 [  0   0   0 ...   0   0 120]]
['back_pain', 'dizziness', 'chest_pain', 'cramps', 'belly_pain']
[1, 1, 1, 1, 1]
```

Figure 46: Confusion Matrix

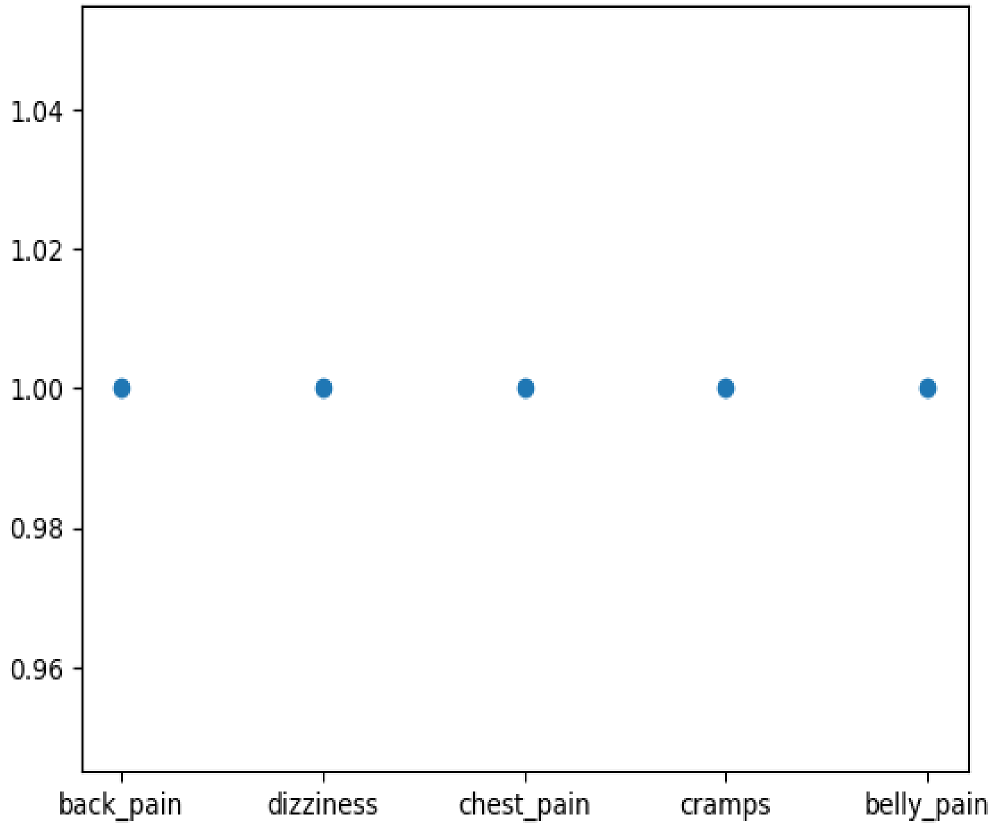
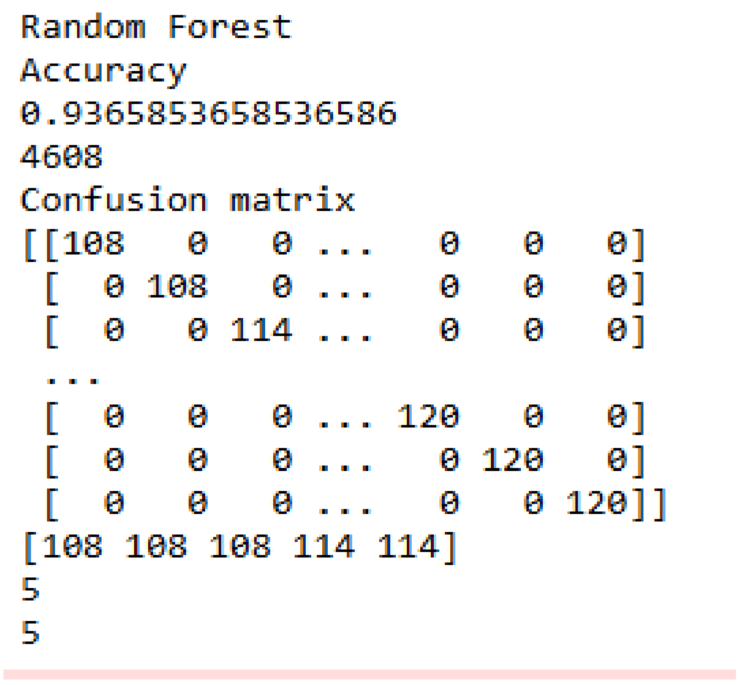


Figure 47: Graph



```
Random Forest
Accuracy
0.9365853658536586
4608
Confusion matrix
[[108   0   0 ...   0   0   0]
 [  0 108   0 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ... 120   0   0]
 [  0   0   0 ...   0 120   0]
 [  0   0   0 ...   0   0 120]]
[108 108 108 114 114]
5
5
```

Figure 48: Confusion Matrix

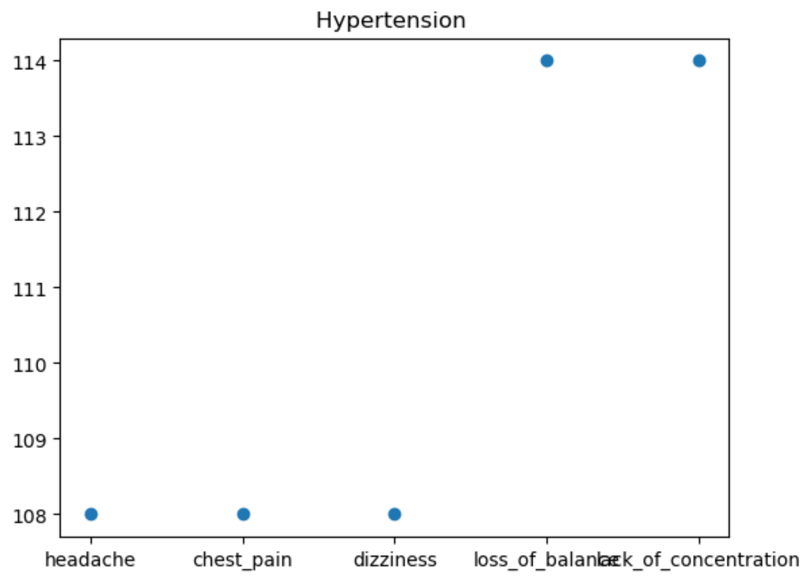
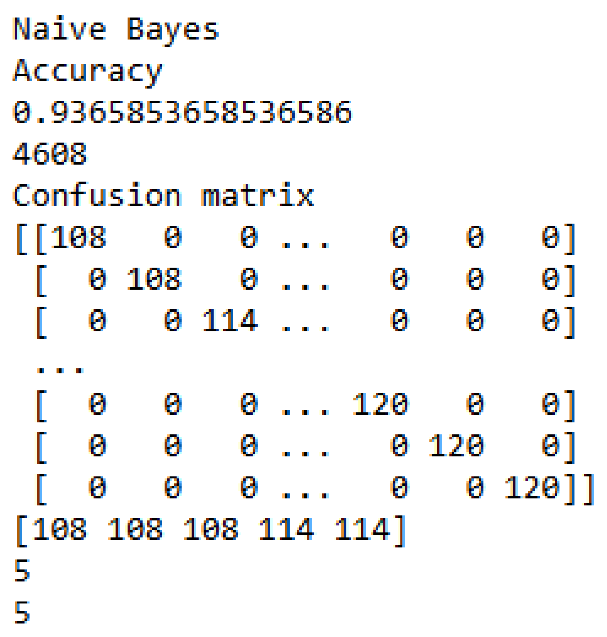


Figure 49: Graph



```
Naive Bayes
Accuracy
0.9365853658536586
4608
Confusion matrix
[[108   0   0 ...   0   0   0]
 [  0 108   0 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ... 120   0   0]
 [  0   0   0 ...   0 120   0]
 [  0   0   0 ...   0   0 120]]
[108 108 108 114 114]
5
5
```

Figure 50: Confusion Matrix

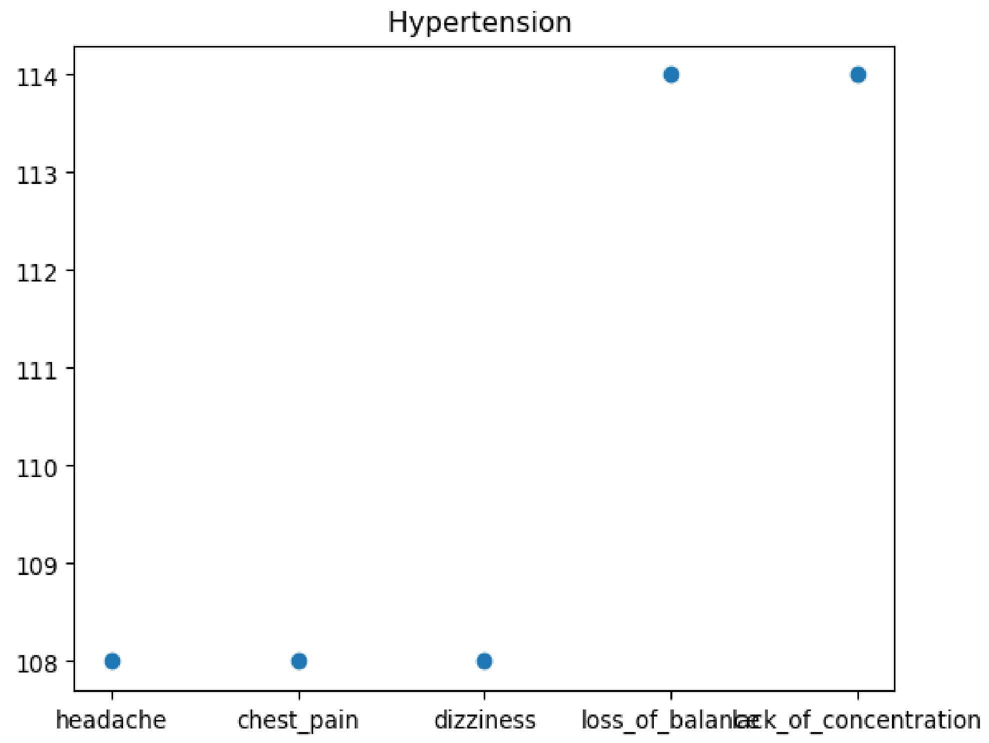


Figure 51: Graph

Conclusion

This study compares the performance of Random Forest, Naive Bayes, and Decision Tree algorithms for disease prediction using clinical data. The findings suggest that while all three algorithms have their merits, Random Forest generally provides better accuracy and robustness due to its ensemble nature (Breiman, 2001). Naive Bayes, with its simplicity and speed, is useful for initial insights and scenarios with strong feature independence (Zhang, 2004). Decision Trees offer interpretability and are beneficial when a clear visualization of decision rules is needed (Quinlan, 1986). Future work could focus on hybrid models and further optimization to enhance predictive accuracy and reliability in clinical settings.

References

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. doi:10.1023/A:1010933404324

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1), 81-106. doi:10.1023/A:1022643204877

Zhang, H. (2004). The Optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, 562-567.