VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

Disease Prediction System Using Machine Learning

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology: Electronics and Communication Engineering

By

Ayushman Mookherjee

18BEC0888

Under the guidance of

Prof Sanjay Kumar Singh, SENSE

VIT, Vellore.

DECLARATION:

I hereby declare that the thesis entitled “Disease Prediction Using Machine Learning" submitted by me, for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering to VIT is a record of bonafide work carried out by me under the supervision of Prof. Sanjay Kumar Singh.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 01/05/2022

Ayushman Mookherjee

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “Disease Prediction Using Machine Learning” submitted by **Ayushman Mookherjee 18BEC0888, SENSE**, VIT, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2021 to 30.04.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Vellore

Date: 01/05/2022

Prof Sanjay Kumar Singh

Signature of the Guide

Internal Examiner **External Examiner**

Dr. Prakasam P

Head of the Department

Electronics and Communication Engineering, SENSE

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to all those who gave me the possibility to complete this project. Special thanks to my guide Prof Sanjay Kumar Singh whose help, stimulating suggestions and encouragement helped me in the fabrication process and also in writing this report. I also thank my team-mate Abhishek Harikrishnan for his valuable inputs to this project.

I would like to also acknowledge with much appreciation the crucial role that VIT and its staff has played throughout this journey. I would also like to acknowledge our Head of the Department, Dr Prakasam P for encouraging me and being a guiding light throughout this journey.

Ayushman Mookherjee

Executive Summary

Health-related information demands are also influencing information-seeking behaviour around the world. Many people's biggest issue is finding information on diseases, diagnostics, and treatments for certain diseases on the internet. If a recommendation system for doctors and users can be created, then review mining will save a lot of time and aid the people involved. The user has difficulty understanding the varied medical terminology in this type of system since the users are ordinary people who are unfamiliar with medical jargon. Because a vast amount of medical information is available on numerous platforms, the user is perplexed. The goal of this project is to adapt to and cope with the unique needs of each user in the health industry.

A web-based recommendation system which serves as a multi-utility tool for normal users and medical professionals alike is what is intended. After all the basic objective of this project is to show that technologies like Machine-Learning and Artificial Intelligence can also help in the healthcare sector.

Index

INTRODUCTION

Abstract:

The healthcare sector is one of the most important research issues in the modern era, because of rapid advances in technology and data. It's difficult to keep track of such a large amount of patient data. The use of Big Data Analytics makes it easier to manage this data. Various ailments can be treated in a variety of ways all throughout the world. Machine Learning is a novel way for predicting and diagnosing diseases.

This study demonstrates how symptoms can be used to predict disease using machine learning. Machine learning approaches such as Gaussian Naive Bayes, Decision Tree, Random Forest, and KNN are used to predict sickness on the supplied dataset. The programming language python is used to implement it. The research reveals which algorithm is the most accurate based on its accuracy. The accuracy of an algorithm is determined by how well it performs on a particular dataset. We'll then create a user interface that allows the user to submit disease symptoms and have the system predict the ailment using various machine learning methods.

Motivation:

The need of an easy to use, easily accessible and reliable source for disease prediction which is low cost, low maintenance and can help a large amount of people. The objective of this prediction system is for those users who do not have testing centers available nearby and to those who don’t have access to immediate medical attention. This prediction system can be used as the first stop for diagnosis and can be used as an able aid for medical professionals in helping them and making their lives easier as well.

Background:

Diagnosis and prognosis of diseases take time, and when it comes to the healthcare sector, time is the most valuable aspect. Sometimes it takes medical professionals time to arrive at their prognosis, and it may prove futile. In places where medical facilities are not that great like in lower tiered villages or towns, a system like this can greatly help and can act as the first line of help. This system can help medical professionals arrive at their prognosis faster and can also help in detecting diseases at an earlier stage.

LITERATURE REVIEW

A disease prediction system can provide basic information to the majority of a country's population at little or no cost. Using real-world data, this disease prediction system focuses on general disease prediction and specialized covid disease prediction. This prediction system can provide a centralized system where everyone can easily access information.

1) Web based disease prediction and recommender system:
The prediction in this paper is done using four different machine learning algorithms (Random Forest, Nave Bayes, KNN, and Ensemble). The source of the data is not specified. For various algorithms, the accuracy achieved in this paper ranges between 84 and 93.5.

2) An Overview on Disease Prediction for Preventive Care of Health Deterioration:
This paper discusses the need for a low-cost disease prediction system capable of detecting diseases at an early stage. It also includes a list of the techniques that are currently being used to predict disease. Logistic regression, Support Vector Machine (SVM), Decision Tree, and Clustering techniques are the most popular, accounting for 27.5 percent, 25 percent, 22.5 percent, and 20 percent, respectively. Logistic regression dominates the list due to the binary nature of medical data in most cases.

3) Use of Electronic Health Data for Disease Prediction:
This paper discusses the various methods currently in use for specific disease prediction, as well as their accuracies. The term 'Electronic Health Data' refers to digitised health data that contains disease, diagnostic, and treatment information for patients. In this comprehensive literature review, all papers with the keywords 'Electronic Health data' and 'Disease prediction' were compared and listed.

4) PREDICTION OF DISEASES USING SUPERVISED LEARNING:
This paper contains disease predictions gleaned from a variety of sources, including hospitals, patient discharge slips, and the UCI repository. The model is then trained using supervised machine learning algorithms such as Decision Tree, Random Forest, SVM (Support Vector Machine), and Naive-Bayes. The accuracies in SVM range from 54 to 95 in Random Forest. The precise source of the data is not specified in this paper.

5) Disease Prediction by Machine Learning Over Big Data from Healthcare Communities:
The use of a new convolutional neural network-based multimodal disease risk prediction method is proposed in this paper (CNN-MDRP). The information was gathered from real-life hospital data from central China between 2013 and 2015. The proposed algorithm has a prediction accuracy of 94.8 percent.

6) Prediction of Disease Using Machine Learning and Deep Learning:
This paper is primarily concerned with predicting the occurrence of a specific disease. Because we all know that prevention is better than cure, it is critical to identify a specific disease and follow the necessary guidelines before its severity worsens. As a result, we devised a method for predicting disease before it occurs. There is already a system in place for predicting disease, but it is limited to datasets available in local healthcare communities (structured data). The proposed system focuses primarily on big data, which is widely used nowadays. It employs the multi-model CNN algorithm to process both unstructured and structured data. This algorithmic model is made up of three layers i.e. the input layer, hidden layer (can be multiple) and the output layer.

7) Disease Prediction using Machine Learning:
The purpose of this work is to explore the idea that supervised ML algorithms can improve health care by detecting diseases accurately and early. We analyse studies that use more than one supervised ML model for each disease recognition challenge in this article. This method provides greater comprehensiveness and precision because evaluating the effectiveness of a single algorithm across multiple study conditions introduces bias, resulting in imprecise conclusions. The examination of ML models will be carried out on a few disorders affecting the heart, kidney, breast, and brain. Numerous techniques, including KNN, NB, DT, CNN, SVM, and LR, will be assessed for disease identification.

8) Designing Disease Prediction Model Using Machine Learning Approach:
This paper predicts disease in three cases: diabetes, cerebral infraction, and heart disease. Disease prediction is based on structured data. Prediction of heart disease, diabetes, and cerebral infraction is done using several machine learning algorithms such as nave bayes, decision trees, and the KNN algorithm. The Decision tree algorithm outperforms the Nave Bayes and KNN algorithms. They also indicate whether a patient would suffer from a high risk of cerebral infarction or a low risk of cerebral infarction. They used CNN-based multi-model illness risk prediction using text data to estimate the risk of cerebral infraction. The accuracy of CNN-based uni-model illness risk predictions is compared to CNN-based multi-model disease risk prediction system. The accuracy of disease prediction reaches up to the 94.8% with faster speed than CNN based unimodal disease risk prediction algorithm.

9) Disease prediction from various symptoms using machine learning:
A disease prediction system is proposed. The doctor may not always be available when needed. However, in the present era, this prediction method can be used at any time if necessary. Individual symptoms, as well as age and gender, can be fed into the ML model for additional processing. Following basic data processing, the ML model uses the current input to train and test the algorithm, resulting in the anticipated disease.

10) Development of machine learning model for diagnostic disease prediction based on laboratory tests:
Deep learning and machine learning (ML) are increasingly being used in medical science, notably in the realms of visual, audio, and linguistic data. We wanted to create a novel optimised ensemble model for disease prediction utilising laboratory test findings by combining a DNN (deep neural network) model with two ML models. Based on value counts, clinical importance-related features, and missing values, 86 attributes (laboratory tests) were chosen from datasets. For the five most frequent diseases, the optimised ensemble model had an F1-score of 81 percent and a prediction accuracy of 92 percent. The prediction power and disease categorization patterns of the deep learning and ML models differed.

Existing System:

There is a need of an easily accessible system that focuses on the identification of common diseases and chronic diseases to meet the demands of a large number of people in all existing systems.

The source of data is not stated in the majority of studies, which reduces the system's reliability for practical usage. None of the articles provide a system that is simple to use and comprehend for disease detection, as well as a credible dataset to make the model more reliable.

Furthermore, there is no framework in place to assist medical practitioners in making educated decisions using technology, resulting in late disease identification and misdiagnosis in many situations.

Proposed System:

A GUI-based patient diagnostic system that is simply accessible and simple to use, the diagnostic system will be able to forecast the disease based on the user's symptoms. The proposed web-based disease prediction system is simple to use and will provide users with an idea of what they may be experiencing. The platform can be an effective source for gathering data on common symptoms and the underlying condition.

Novelty:

The vast majority of projects available on the internet seek to predict one sort of disease, such as heart disease, diabetes, and so on. All of these projects use one form of machine learning method, often Linear Regression or Support Vector Machines, in which the user enters symptoms and returns a prediction, however the accuracy is not always adequate.

We are attempting to construct a robust system that will make predictions using four different machine learning techniques. As a result, the types of diseases discovered at an early stage are increasing, while the margin of error is decreasing. The key goal is to help medical professionals rather than replace them; technology helps with that goal by helping those professionals to make better decisions. This also contributes to the early diagnosis of the specific disease.

Aside from that, we will undertake an in-depth analysis and comparison of the results acquired from the four algorithms, commenting on each of their accuracies, modes of operation, learning methods, and, lastly, which can be utilized where. Essentially, we will examine how each algorithm is applied and, ideally, explain how Machine Learning may aid in the healthcare business.

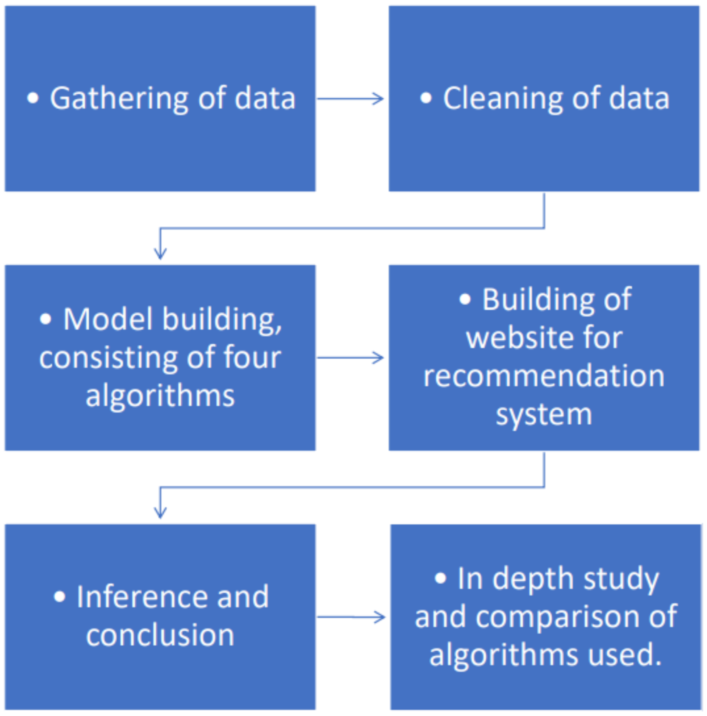
TECHNICAL SPECIFICATIONS

Libraries Used:

- **Tkinter**: It's a standard Python GUI library. Python, when paired with tkinter, provides a quick and straightforward approach to construct a graphical user interface. It provides a sophisticated object-oriented tool for designing graphical user interfaces. Message-box, button, label, Option Menu, text, and title were some of these that were employed in this project to develop our GUI. We were able to construct an interactive GUI for our model using tkinter.
- **Numpy**: Numpy is the Python core library for scientific computing. It provides sophisticated Python utilities for dealing with diverse multi-dimensional arrays. It is a package for general-purpose array processing. The basic aim of Numpy is to deal with multidimensional homogenous arrays. It includes tools for everything from array construction to array management. It simplifies the creation of an n-dimensional array with np.zeros() and the manipulation of its contents with methods such as replace, arrange, random, save, and load. It also aids in array processing when utilizing methods such as sum, mean, standard deviation, max, min, all, and so on.
- **Pandas**: It is the most widely used Python package for data analysis. It offers highly optimized performance with back-end source code written entirely in C or Python. Python data can be analyzed in two ways namely series and dataframes. A series is a one-dimensional array defined in pandas that may store any form of data. Dataframes are two-dimensional data structures in Python that are used to store data in the form of rows and columns. In this project, Pandas dataframes are widely employed to use datasets required for training and testing the algorithms. Dataframes enable working with attributes and results easier. Several of its built-in functions, such as replace, were employed for data modification and preprocessing in our research.
- **Sklearn**: Sklearn is a Python open-source library that provides a wide variety of machine learning, pre-processing, cross-validation, and visualization techniques. It includes a number of straightforward and effective tools for data mining and processing. It includes a variety of classification, regression, and clustering algorithms, including support vector machine, random forest classifier, decision tree, gaussian nave-Bayes, and KNN, to mention a few. We used sklearn in this project to take advantage of built-in classification techniques such as decision trees, random forest classifiers, KNN, and naive Bayes. We also made use of built-in cross validation and visualization tools such as the classification report, confusion matrix, and accuracy score.

DESIGN APPROACH AND DETAILS

APPROACH:



• Gathering of data
• Cleaning of data
• Model building, consisting of four algorithms
• Building of website for recommendation system
• Inference and conclusion
• In depth study and comparison of algorithms used.

- Gathering Data: The very first step in any machine learning task is to prepare the data. There are two CSV files in this dataset: one for training and one for testing.
- Cleaning the Data: The most critical stage in a machine learning project is cleaning the data. Our machine learning model's quality is determined on the quality of our data. As a result, data must always be cleaned before being fed to the model for training. All of the columns in our dataset are numerical, except for the goal column, prognosis, which is a textual type that is encoded to numerical form using a label encoder.
- Model Building: Once the data has been gathered and cleaned, it is ready to be utilized to train a machine learning model. The Decision Tree, Naive Bayes Classifier, KNN, Random Forest Classifier, and others will be trained using this cleaned data. The quality of the models will be determined using a confusion matrix.
- Inference: After training the four models, we will predict the disease based on the input symptoms by combining the predictions of all three models. This increases the robustness and accuracy of our overall prediction.

CODES:

Iteration-1

Includes data preprocessing, cleaning, validation, testing and a dry run.

```
#Importing all libraries
import pickle
import numpy as np
import pandas as pd
from scipy.stats import mode
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

%matplotlib inline
```

```
# Reading the dataset
```

```
# Reading the train.csv by removing the
# last column since it's an empty column
DATA_PATH = "Training.csv"
data = pd.read_csv(DATA_PATH).dropna(axis = 1)

# Checking whether the dataset is balanced or not
disease_counts = data["prognosis"].value_counts()
temp_df = pd.DataFrame({"Disease": disease_counts.index,"Counts": disease_counts.values})

plt.figure(figsize = (18,8))
sns.barplot(x = "Disease", y = "Counts", data = temp_df)
plt.xticks(rotation=90)
plt.show()
```

```
# Encoding the target value into numerical
# value using LabelEncoder
encoder = LabelEncoder()
data["prognosis"] = encoder.fit_transform(data["prognosis"])
```

```
# Splitting the data for training and testing the model
```

```
X = data.iloc[:,:-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test =train_test_split(
X, y, test_size = 0.2, random_state = 24)

print(f"Train: {X_train.shape}, {y_train.shape}")
print(f"Test: {X_test.shape}, {y_test.shape}")
```

```
Train: (3936, 132), (3936,)
Test: (984, 132), (984,)
```

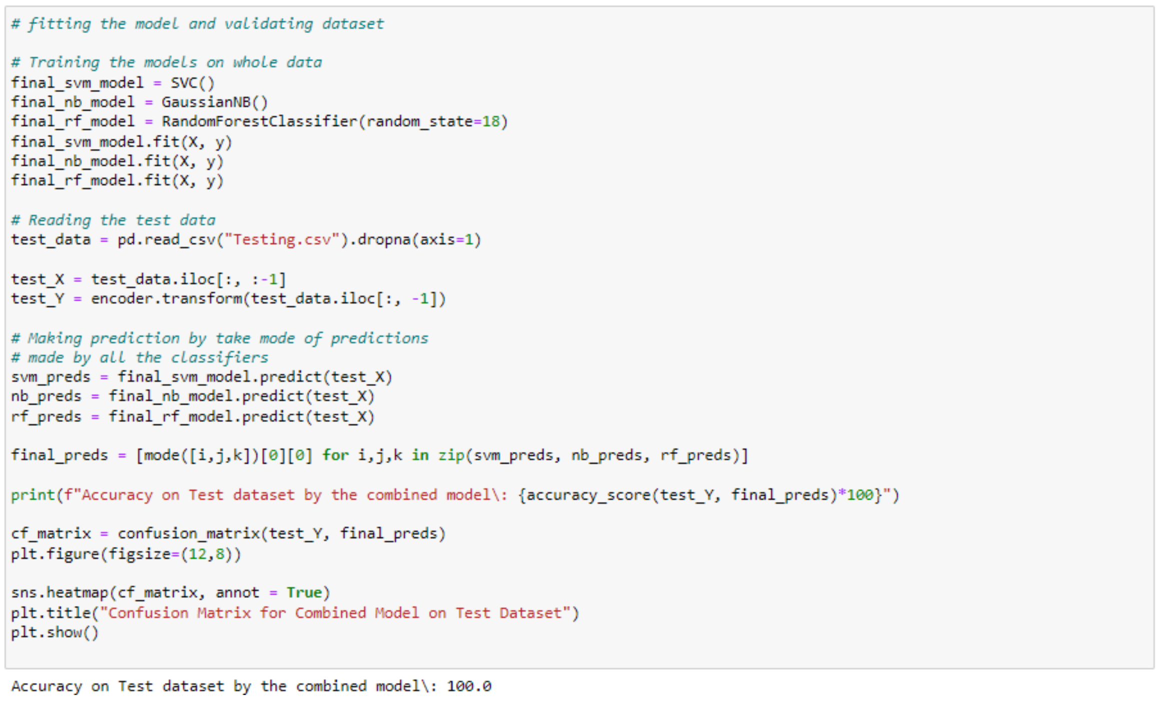
```python
# Model Building
```

```python
# Defining scoring metric for k-fold cross validation
def cv_scoring(estimator, X, y):
    return accuracy_score(y, estimator.predict(X))

# Initializing Models
models = {
    "SVC":SVC(),
    "Gaussian NB":GaussianNB(),
    "Random Forest":RandomForestClassifier(random_state=18)
}

# Producing cross validation score for the models
for model_name in models:
    model = models[model_name]
    scores = cross_val_score(model, X, y, cv = 10,n_jobs = -1,scoring = cv_scoring)
    print("=="*30)
    print(model_name)
    print(f"Scores: {scores}")
    print(f"Mean Score: {np.mean(scores)}")
```

```
============================================================
SVC
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Gaussian NB
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Random Forest
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
```

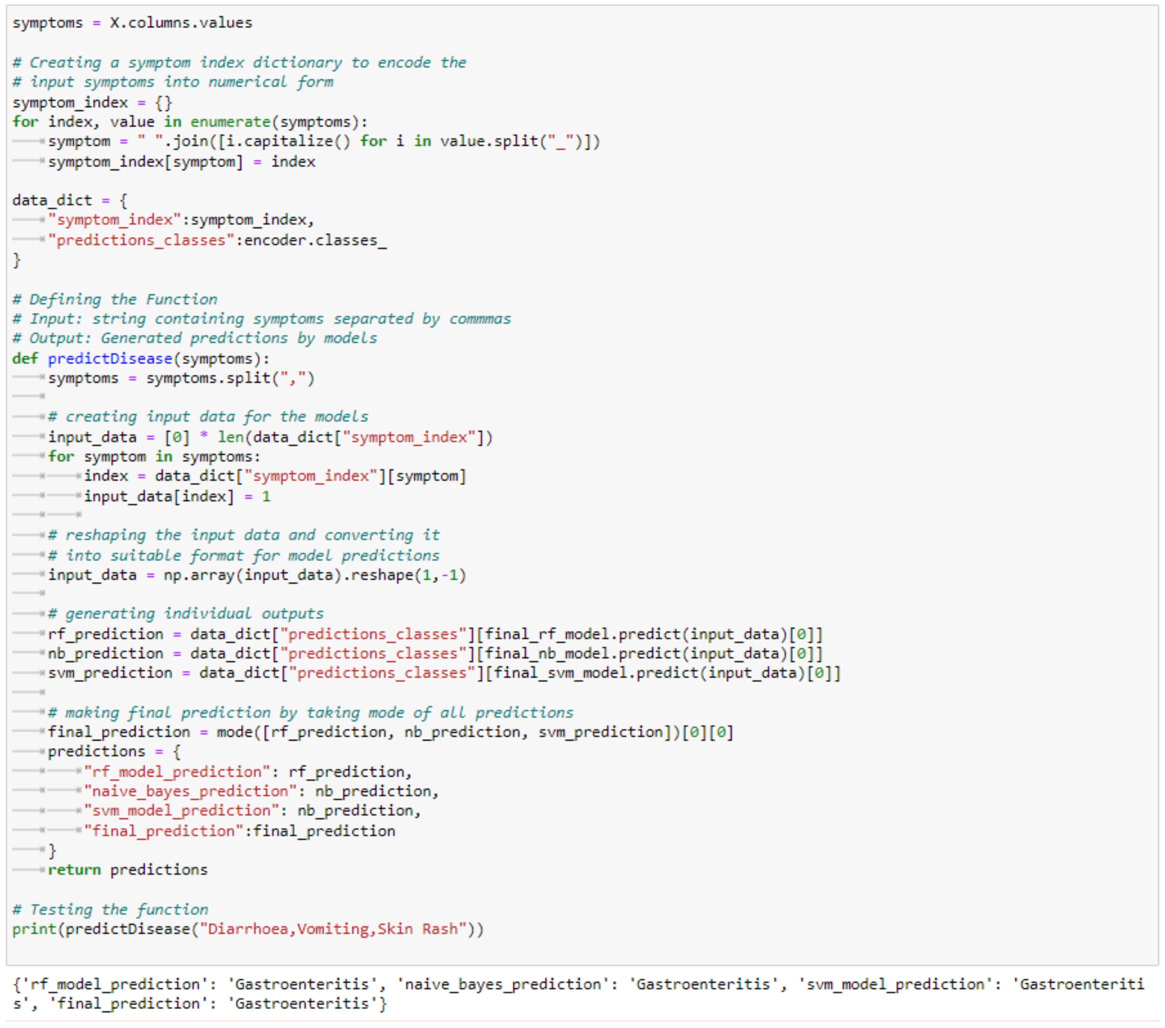
```python
# Training and testing SVM Classifier
svm_model = SVC()
svm_model.fit(X_train, y_train)
preds = svm_model.predict(X_test)

print(f"Accuracy on train data by SVM Classifier\: {accuracy_score(y_train, svm_model.predict(X_train))*100}")

print(f"Accuracy on test data by SVM Classifier\: {accuracy_score(y_test, preds)*100}")
cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for SVM Classifier on Test Data")
plt.show()

# Training and testing Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
preds = nb_model.predict(X_test)
print(f"Accuracy on train data by Naive Bayes Classifier\: {accuracy_score(y_train, nb_model.predict(X_train))*100}")

print(f"Accuracy on test data by Naive Bayes Classifier\: {accuracy_score(y_test, preds)*100}")
cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Naive Bayes Classifier on Test Data")
plt.show()

# Training and testing Random Forest Classifier
rf_model = RandomForestClassifier(random_state=18)
rf_model.fit(X_train, y_train)
preds = rf_model.predict(X_test)
print(f"Accuracy on train data by Random Forest Classifier\: {accuracy_score(y_train, rf_model.predict(X_train))*100}")

print(f"Accuracy on test data by Random Forest Classifier\: {accuracy_score(y_test, preds)*100}")

cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Random Forest Classifier on Test Data")
plt.show()
```

```
Accuracy on train data by SVM Classifier\: 100.0
Accuracy on test data by SVM Classifier\: 100.0
```



```python
# fitting the model and validating dataset

# Training the models on whole data
final_svm_model = SVC()
final_nb_model = GaussianNB()
final_rf_model = RandomForestClassifier(random_state=18)
final_svm_model.fit(X, y)
final_nb_model.fit(X, y)
final_rf_model.fit(X, y)

# Reading the test data
test_data = pd.read_csv("Testing.csv").dropna(axis=1)

test_X = test_data.iloc[:, :-1]
test_Y = encoder.transform(test_data.iloc[:, -1])

# Making prediction by take mode of predictions
# made by all the classifiers
svm_preds = final_svm_model.predict(test_X)
nb_preds = final_nb_model.predict(test_X)
rf_preds = final_rf_model.predict(test_X)

final_preds = [mode([i,j,k])[0][0] for i,j,k in zip(svm_preds, nb_preds, rf_preds)]

print(f"Accuracy on Test dataset by the combined model\: {accuracy_score(test_Y, final_preds)*100}")

cf_matrix = confusion_matrix(test_Y, final_preds)
plt.figure(figsize=(12,8))

sns.heatmap(cf_matrix, annot = True)
plt.title("Confusion Matrix for Combined Model on Test Dataset")
plt.show()
```

```
Accuracy on Test dataset by the combined model\: 100.0
```



```python
symptoms = X.columns.values

# Creating a symptom index dictionary to encode the
# input symptoms into numerical form
symptom_index = {}
for index, value in enumerate(symptoms):
    symptom = " ".join([i.capitalize() for i in value.split("_")])
    symptom_index[symptom] = index

data_dict = {
    "symptom_index":symptom_index,
    "predictions_classes":encoder.classes_
}

# Defining the Function
# Input: string containing symptoms separated by commmas
# Output: Generated predictions by models
def predictDisease(symptoms):
    symptoms = symptoms.split(",")
    
    # creating input data for the models
    input_data = [0] * len(data_dict["symptom_index"])
    for symptom in symptoms:
        index = data_dict["symptom_index"][symptom]
        input_data[index] = 1
        
    # reshaping the input data and converting it
    # into suitable format for model predictions
    input_data = np.array(input_data).reshape(1,-1)
    
    # generating individual outputs
    rf_prediction = data_dict["predictions_classes"][final_rf_model.predict(input_data)[0]]
    nb_prediction = data_dict["predictions_classes"][final_nb_model.predict(input_data)[0]]
    svm_prediction = data_dict["predictions_classes"][final_svm_model.predict(input_data)[0]]
    
    # making final prediction by taking mode of all predictions
    final_prediction = mode([rf_prediction, nb_prediction, svm_prediction])[0][0]
    predictions = {
        "rf_model_prediction": rf_prediction,
        "naive_bayes_prediction": nb_prediction,
        "svm_model_prediction": nb_prediction,
        "final_prediction":final_prediction
    }
    return predictions

# Testing the function
print(predictDisease("Diarrhoea,Vomiting,Skin Rash"))
```

```
{'rf_model_prediction': 'Gastroenteritis', 'naive_bayes_prediction': 'Gastroenteritis', 'svm_model_prediction': 'Gastroenteriti
s', 'final_prediction': 'Gastroenteritis'}
```

```
data = {"model": final_svm_model,  "encoder": encoder}
with open('svm.pkl', 'wb') as file:
    pickle.dump(data, file)
```

```
data = {"model": final_nb_model,  "encoder": encoder}
with open('nb.pkl', 'wb') as file:
    pickle.dump(data, file)
```

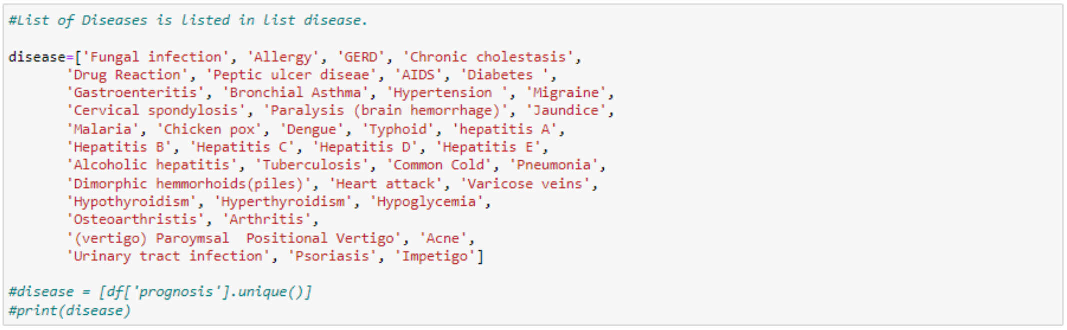
```
data = {"model": final_rf_model,  "encoder": encoder}
with open('rf.pkl', 'wb') as file:
    pickle.dump(data, file)
```

Iteration-2

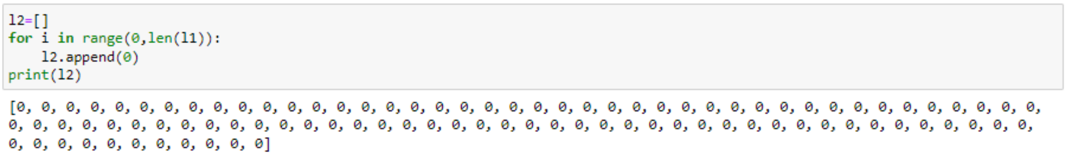
Consists of actual model building, training, fitting, GUI building and algorithm analysis.

```
#Importing Libraries

from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from tkinter import *
import tkinter as tk
from tkinter import LabelFrame, Label, Tk#, Canvas
import numpy as np
import pandas as pd
import os
```

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

l1 = ['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
    'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
    'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',
    'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_in_limbs',
    'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','bloody_stool',
    'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesity','swollen_legs',
    'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittle_nails',
    'swollen_extremeties','excessive_hunger','extra_marital_contacts','drying_and_tingling_lips',
    'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_neck','swelling_joints',
    'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness',
    'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_smell_of urine',
    'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_look_(typhos)',
    'depression','irritability','muscle_pain','altered_sensorium','red_spots_over_body','belly_pain',
    'abnormal_menstruation','dischromic _patches','watering_from_eyes','increased_appetite','polyuria','family_history','mucoid_s
    'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
    'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',
    'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_calf',
    'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',
    'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose',
    'yellow_crust_ooze']
```

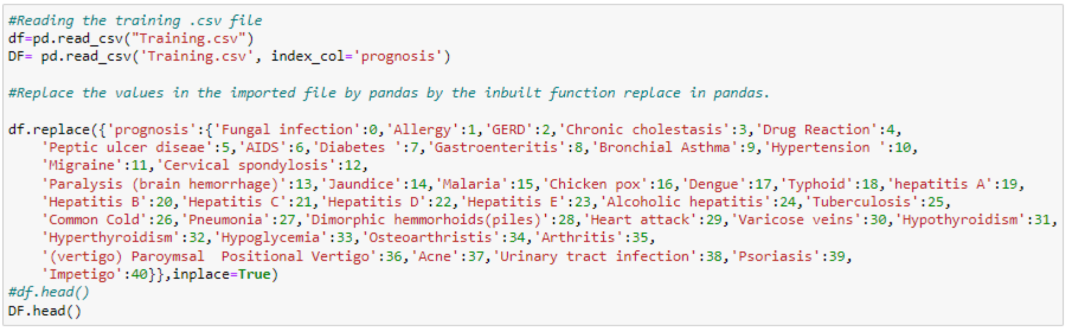


```
#List of Diseases is listed in list disease.

disease=['Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis',
       'Drug Reaction', 'Peptic ulcer diseae', 'AIDS', 'Diabetes ',
       'Gastroenteritis', 'Bronchial Asthma', 'Hypertension ', 'Migraine',
       'Cervical spondylosis', 'Paralysis (brain hemorrhage)', 'Jaundice',
       'Malaria', 'Chicken pox', 'Dengue', 'Typhoid', 'hepatitis A',
       'Hepatitis B', 'Hepatitis C', 'Hepatitis D', 'Hepatitis E',
       'Alcoholic hepatitis', 'Tuberculosis', 'Common Cold', 'Pneumonia',
       'Dimorphic hemmorhoids(piles)', '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)
```

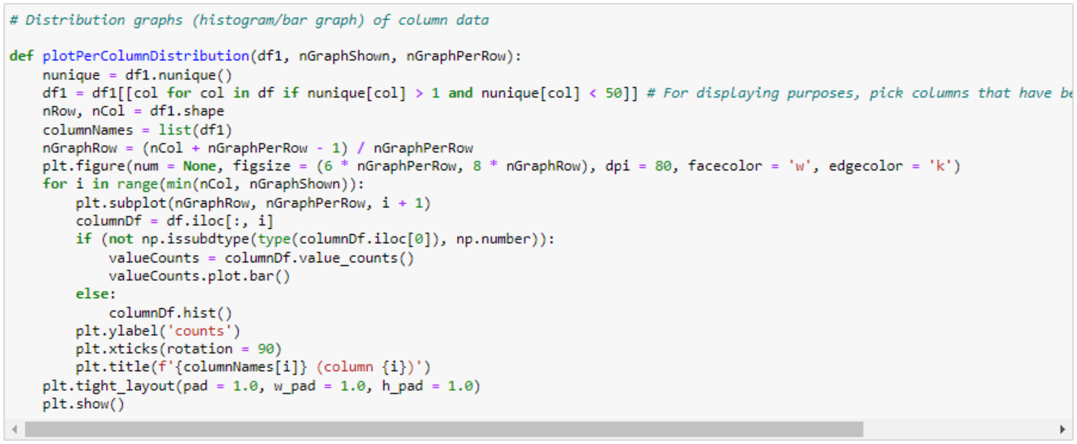


```
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
```

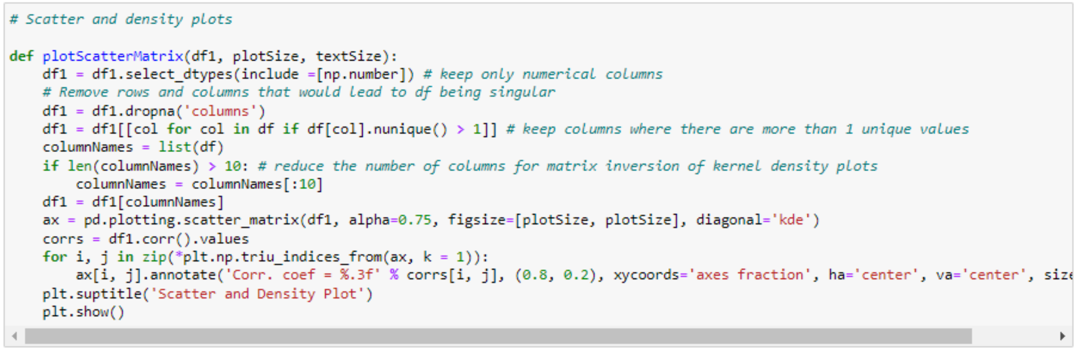
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



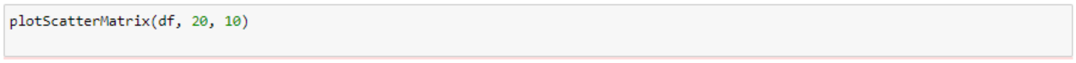
```
#Reading the training .csv file
df=pd.read_csv("Training.csv")
DF= pd.read_csv('Training.csv', index_col='prognosis')

#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':31,
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
#df.head()
DF.head()
```



```
# Distribution graphs (histogram/bar graph) of column data

def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have be
    nRow, nCol = df1.shape
    columnNames = list(df1)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```



```
# Scatter and density plots

def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df1 = df1[columnNames]
    ax = pd.plotting.scatter_matrix(df1, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df1.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```



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

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

```
print(y)
```

```
      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]
```

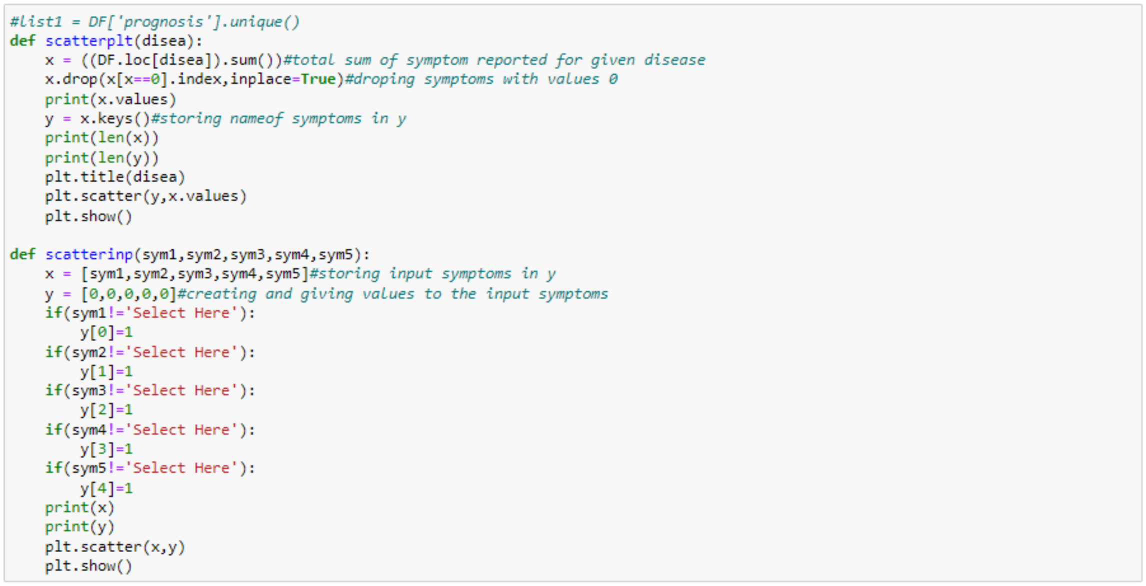
```
#Reading the  testing.csv file
tr=pd.read_csv("Testing.csv")

#Using inbuilt function replace in pandas for replacing the values

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
tr.head()
```

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

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

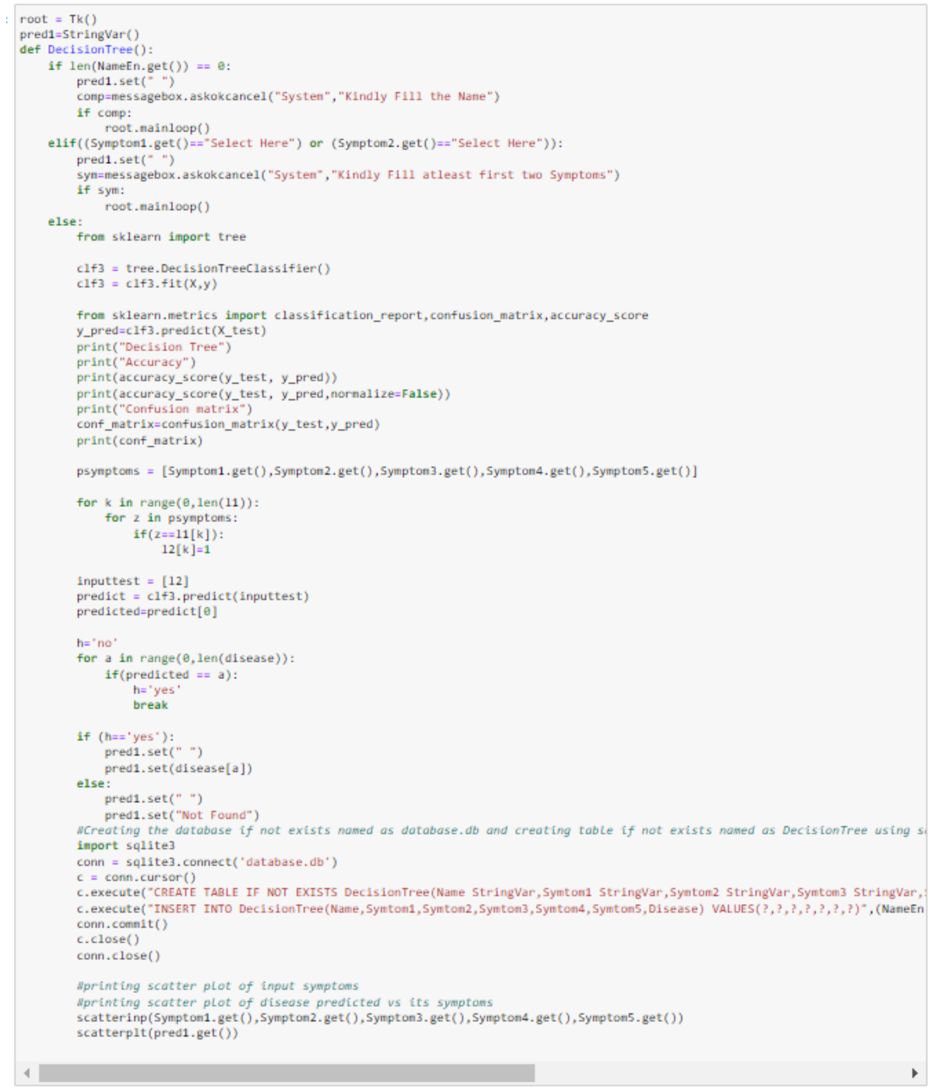
```
print(y_test)
```

```
    prognosis
0           0
1           1
2           2
3           3
4           4
5           5
6           6
7           7
8           8
9           9
10         10
11         11
12         12
13         13
14         14
15         15
16         16
```



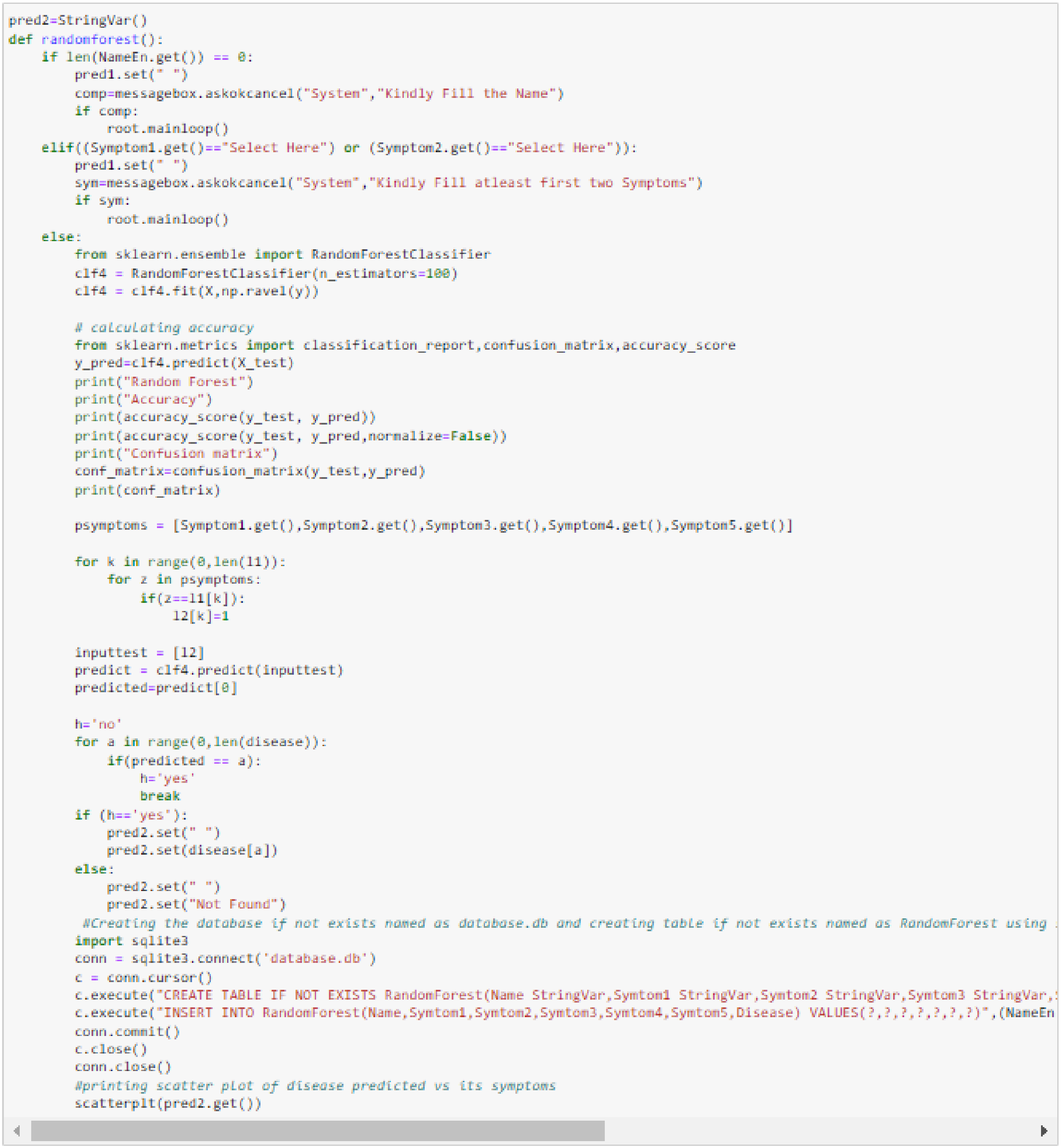
```
#list1 = DF['prognosis'].unique()
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())#total sum of symptom reported for given disease
    x.drop(x[x==0].index,inplace=True)#droping symptoms with values 0
    print(x.values)
    y = x.keys()#storing nameof symptoms in y
    print(len(x))
    print(len(y))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



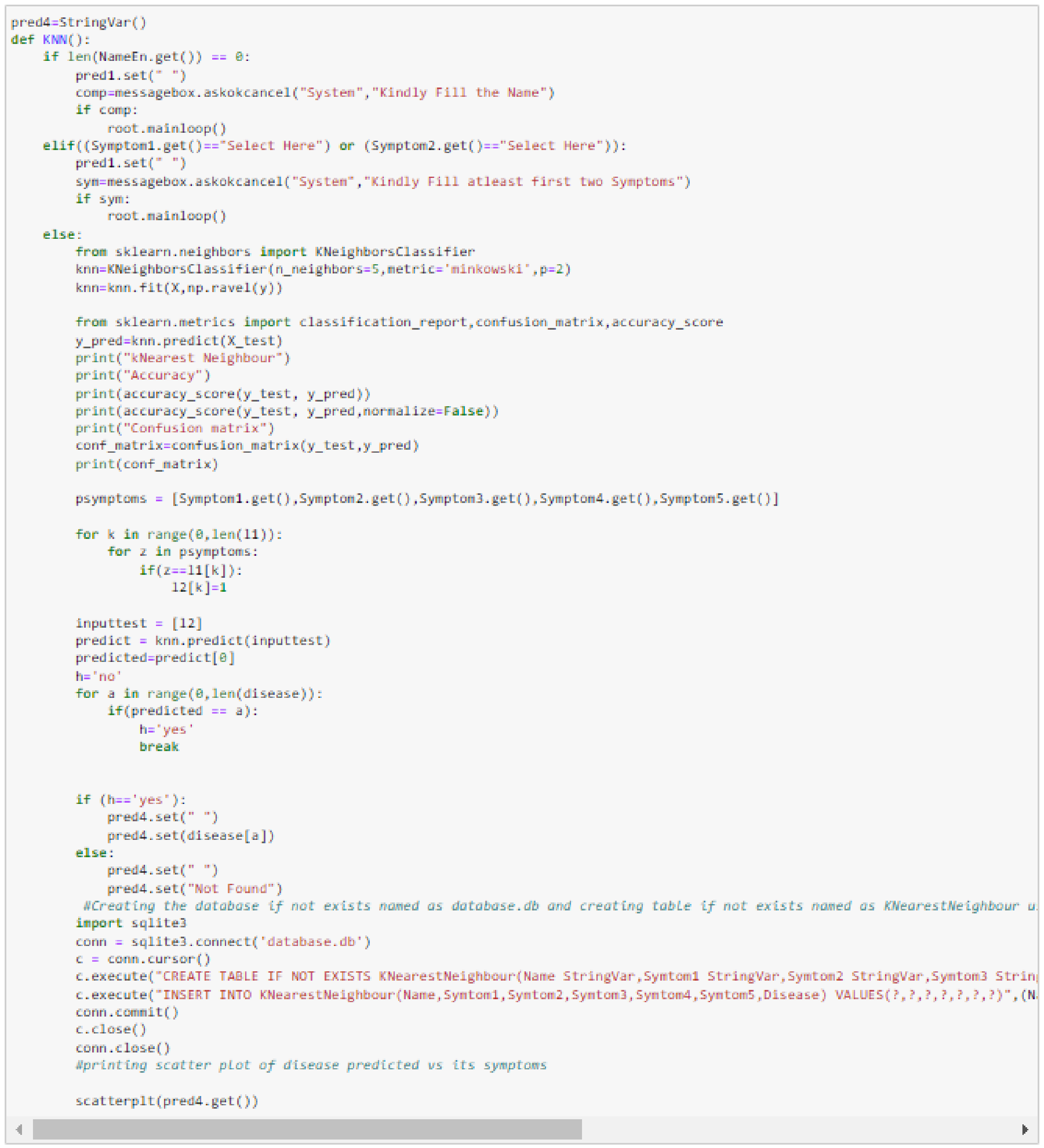
```
root = Tk()
pred1=StringVar()
def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
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

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

        #printing scatter plot of input symptoms
        #printing scatter plot of disease predicted vs its symptoms
        scatterinp(Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get())
        scatterplt(pred1.get())
```

RANDOM FOREST ALGORITHM



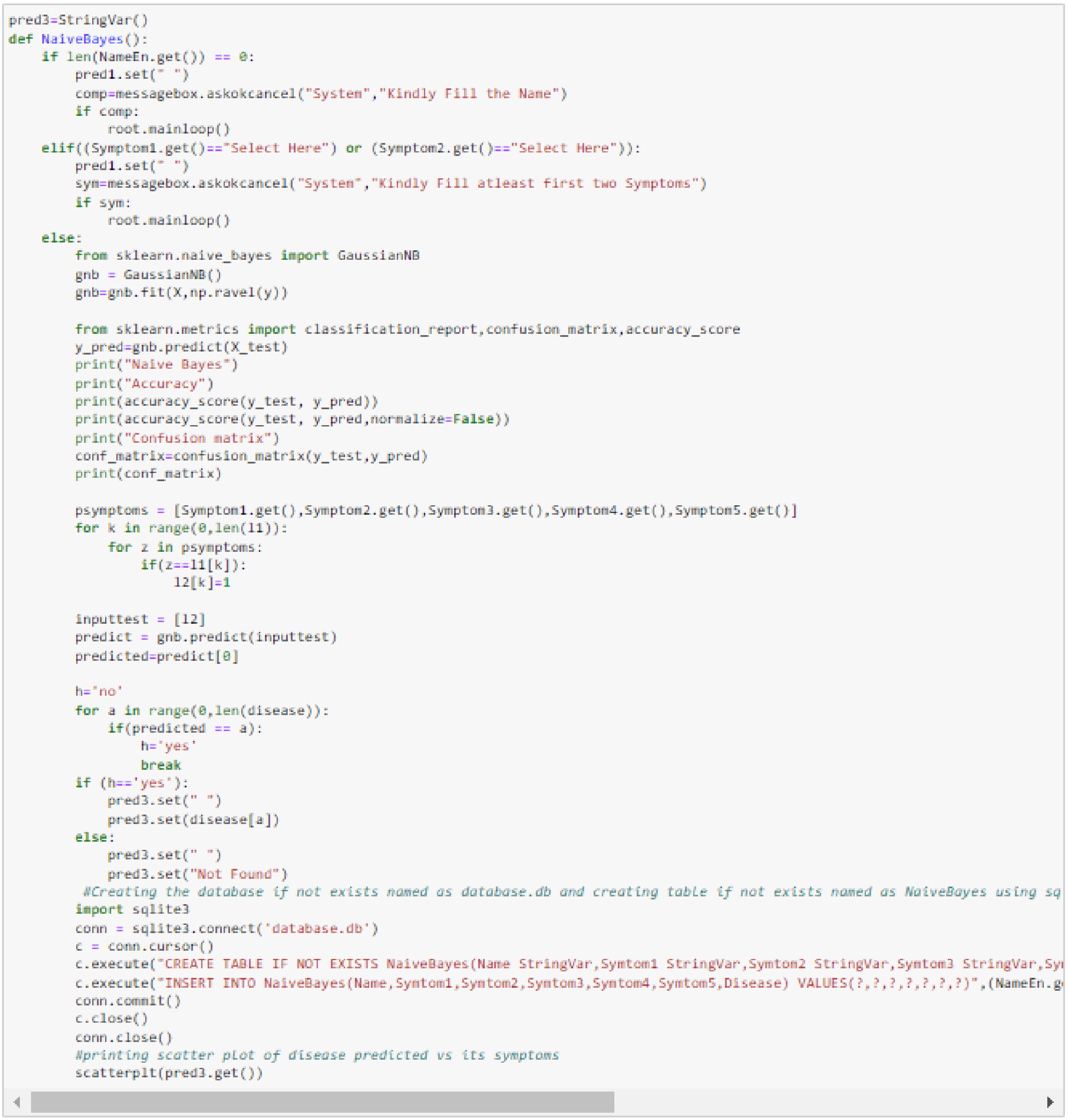
```
pred2=StringVar()
def randomforest():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.ensemble import RandomForestClassifier
        clf4 = RandomForestClassifier(n_estimators=100)
        clf4 = clf4.fit(X,np.ravel(y))

        # calculating accuracy
        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=clf4.predict(X_test)
        print("Random Forest")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]

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

        inputtest = [l2]
        predict = clf4.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break
        if (h=='yes'):
            pred2.set(" ")
            pred2.set(disease[a])
        else:
            pred2.set(" ")
            pred2.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists named as RandomForest using 
        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,
        c.execute("INSERT INTO RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms
        scatterplt(pred2.get())
```

KNN ALGORITHM

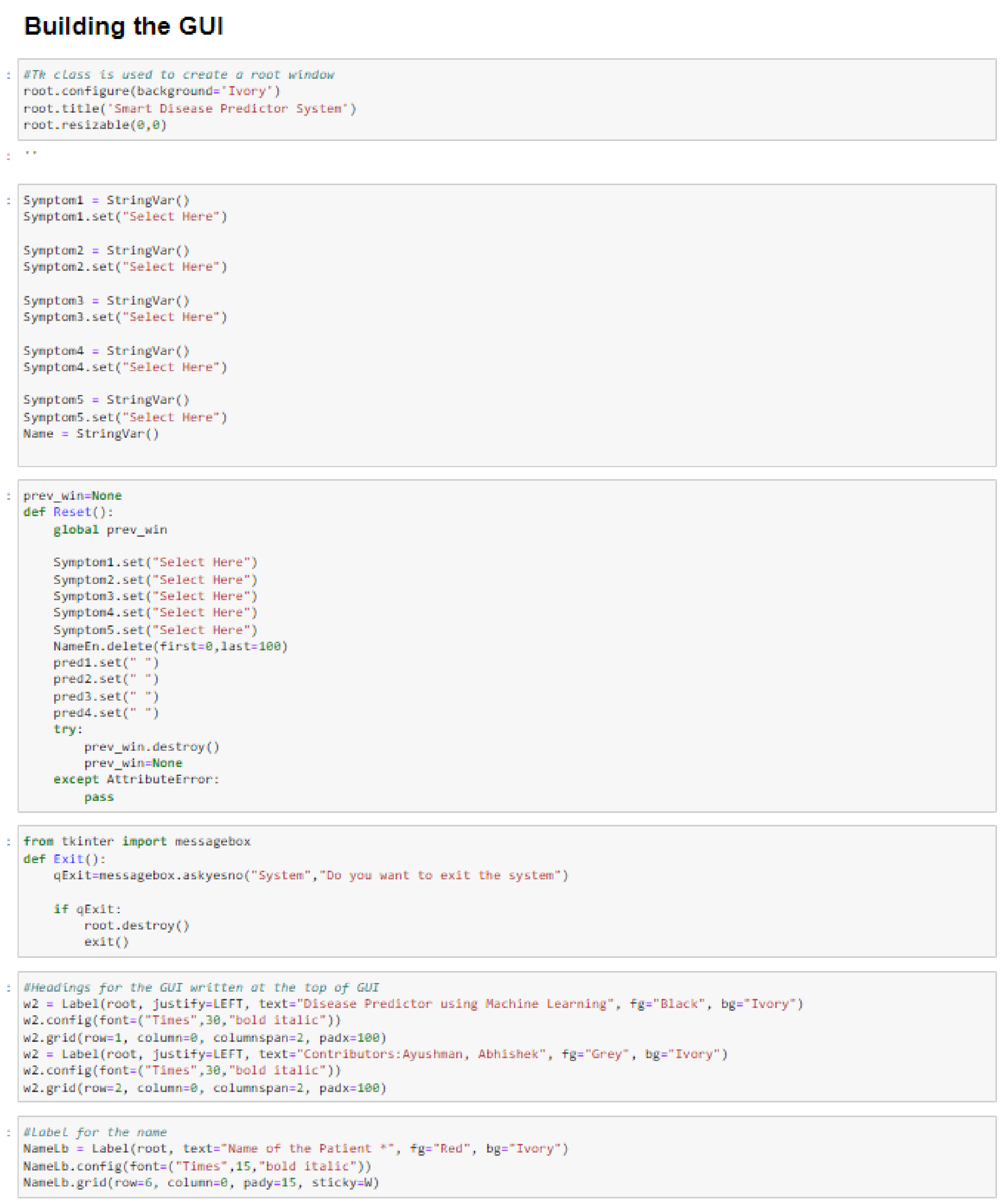


```
pred4=StringVar()
def KNN():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
        knn=knn.fit(X,np.ravel(y))

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

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]

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

        inputtest = [l2]
        predict = knn.predict(inputtest)
        predicted=predict[0]
        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break


        if (h=='yes'):
            pred4.set(" ")
            pred4.set(disease[a])
        else:
            pred4.set(" ")
            pred4.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists named as KNearestNeighbour u
        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 Strin
        c.execute("INSERT INTO KNearestNeighbour(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(N
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms

        scatterplt(pred4.get())
```

NAIVE-BAYES ALGORITHM



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

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

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

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

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break
        if (h=='yes'):
            pred3.set(" ")
            pred3.set(disease[a])
        else:
            pred3.set(" ")
            pred3.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists named as NaiveBayes using sq
        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,Sy
        c.execute("INSERT INTO NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?,?,?,?,?)",(NameEn.g
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms
        scatterplt(pred3.get())
```



Building the GUI

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

```
''
```

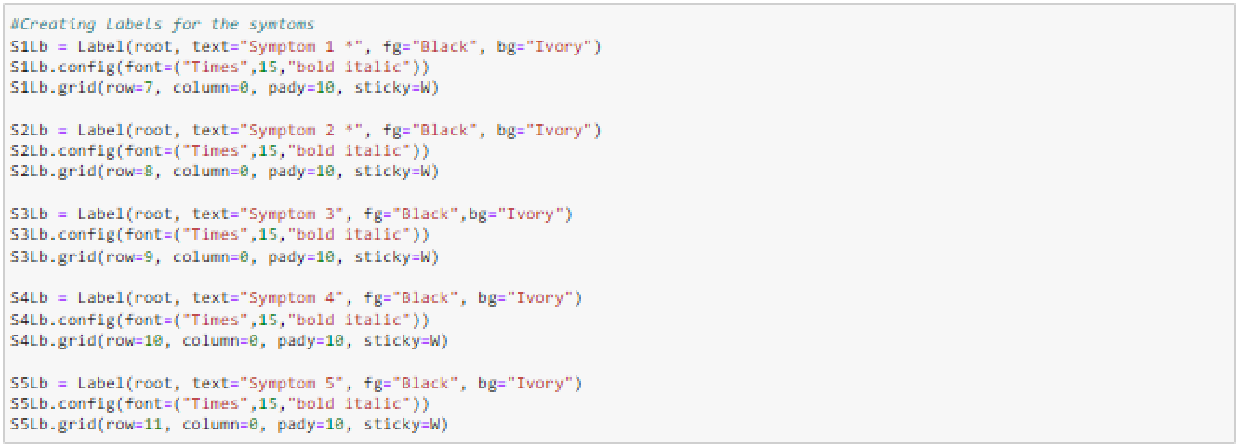
```
Symptom1 = StringVar()
Symptom1.set("Select Here")

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()
```

```
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
```

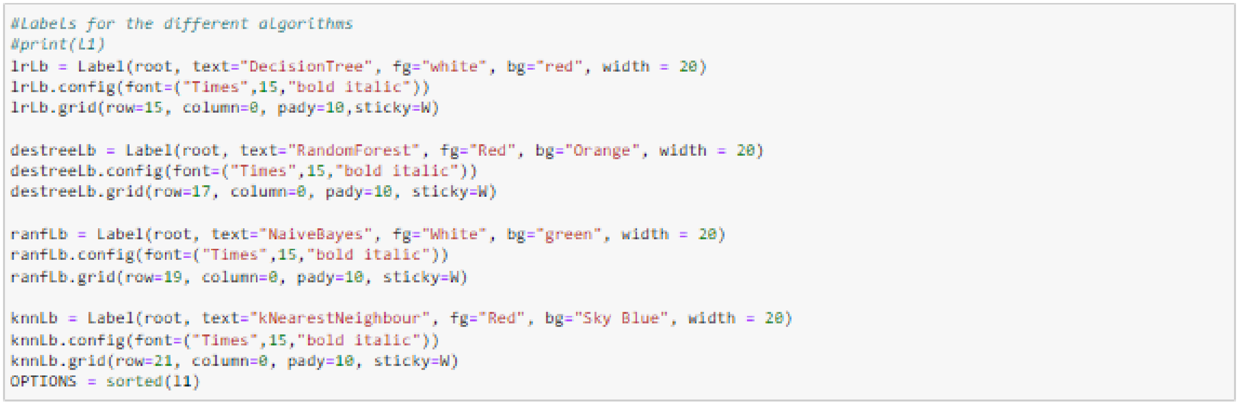
```
from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")

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

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

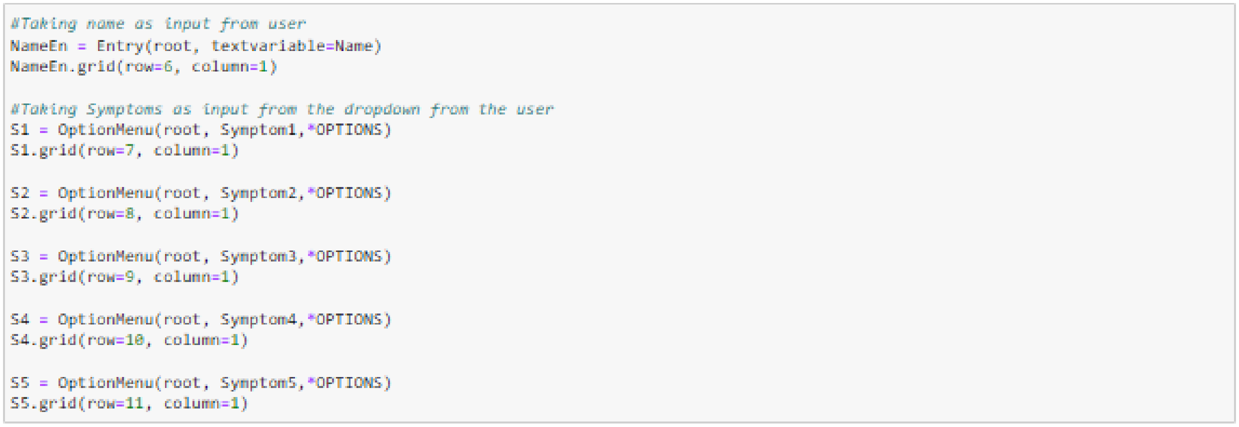
```
#Label for the name
NameLb = Label(root, text="Name of the Patient *", fg="Red", bg="Ivory")
NameLb.config(font=("Times",15,"bold italic"))
NameLb.grid(row=6, column=0, pady=15, sticky=W)
```



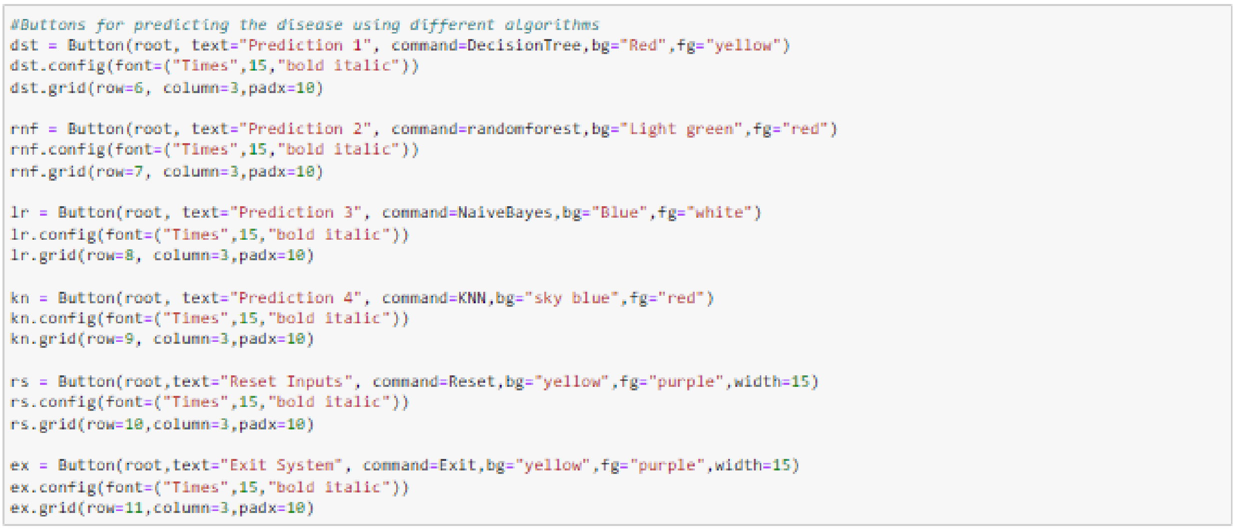
```
#Creating Labels for the symtoms
S1Lb = Label(root, text="Symptom 1 *", fg="Black", bg="Ivory")
S1Lb.config(font=("Times",15,"bold italic"))
S1Lb.grid(row=7, column=0, pady=10, sticky=W)

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

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

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

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



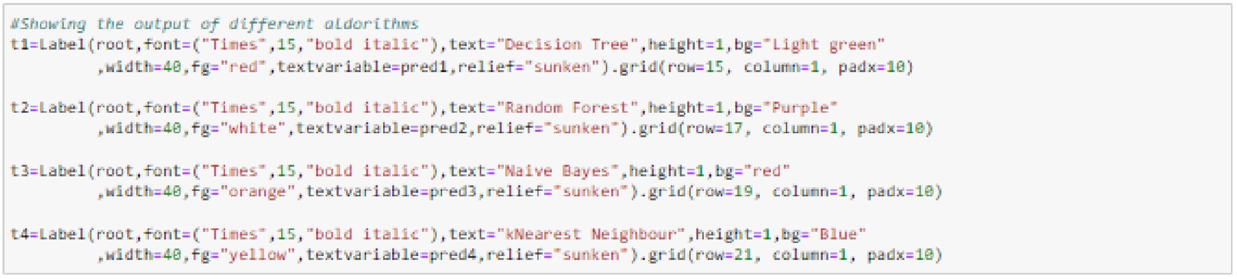
```
#Labels for the different algorithms
#print(l1)
lrLb = Label(root, text="DecisionTree", fg="white", bg="red", width = 20)
lrLb.config(font=("Times",15,"bold italic"))
lrLb.grid(row=15, column=0, pady=10,sticky=W)

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)
```



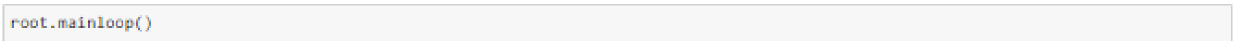
```
#Taking name as input from user
NameEn = Entry(root, textvariable=Name)
NameEn.grid(row=6, column=1)

#Taking Symptoms as input from the dropdown from the user
S1 = OptionMenu(root, Symptom1,*OPTIONS)
S1.grid(row=7, column=1)

S2 = OptionMenu(root, Symptom2,*OPTIONS)
S2.grid(row=8, column=1)

S3 = OptionMenu(root, Symptom3,*OPTIONS)
S3.grid(row=9, column=1)

S4 = OptionMenu(root, Symptom4,*OPTIONS)
S4.grid(row=10, column=1)

S5 = OptionMenu(root, Symptom5,*OPTIONS)
S5.grid(row=11, column=1)
```



```
#Buttons for predicting the disease using different algorithms
dst = Button(root, text="Prediction 1", command=DecisionTree,bg="Red",fg="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)
```



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

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

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

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



```
root.mainloop()
```

CONSTRAINTS:

Working with a huge dataset was a big task. Cleaning the unwanted data from the dataset and testing it took some time. There are codes available for different models on the internet, but since our project depends on a real and a huge dataset, the models had to be coded respectively and trained on our dataset.

The GUI and website that was built for the project took some time and was challenging as we have worked very less with tkinter and web development tools.

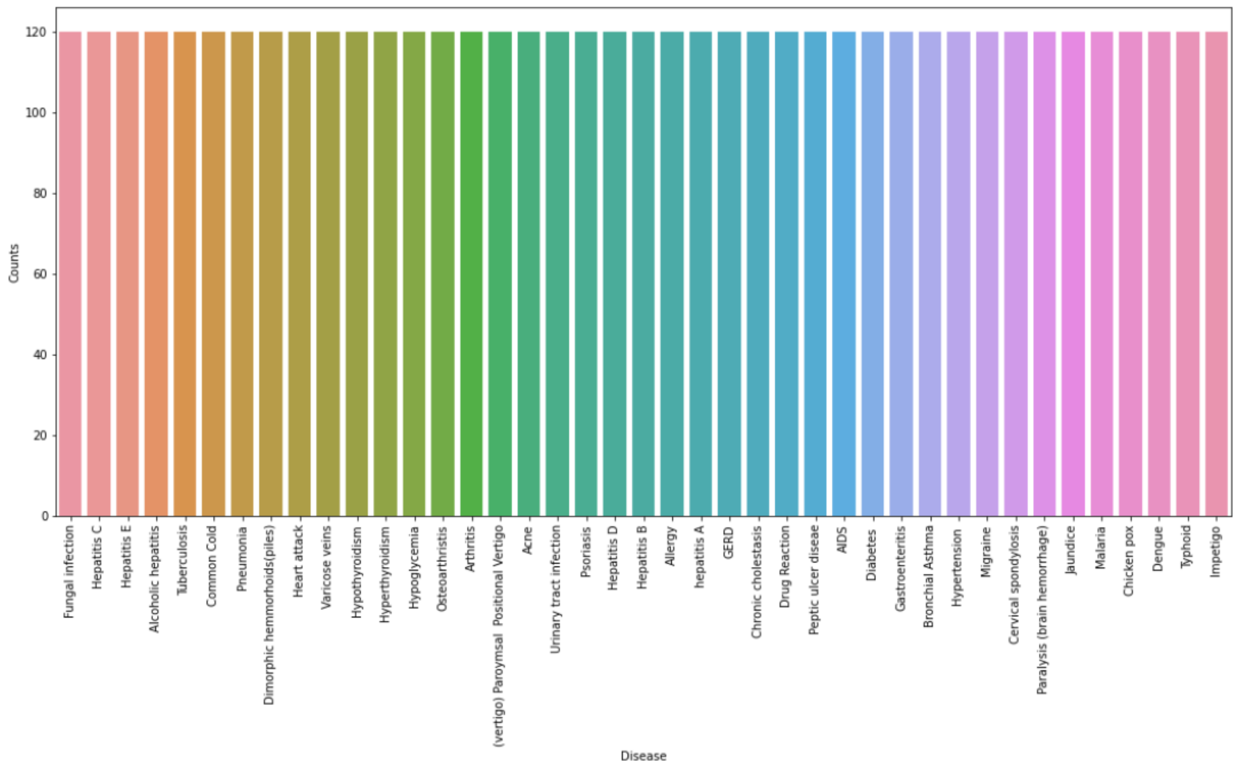
TIMELINE

	January	February	March	April	May
Project Synopsis	⬛				
Research	🟦	🟦			
Building Model		🟧	🟧		
Comparing the various algorithm used			🟨		
Implementing the Website			🟦	🟦	
Inference and Conclusion				🟩	🟩

Implementation, Results and Analysis

Reading the dataset:

To begin, we will load the dataset from the folders using the pandas library. While reading the dataset, we will remove the null column. This is a clean dataset that contains no null values and only 0s and 1s as features. We must determine whether or not our target column is balanced when performing a classification task. A bar plot will be used to determine whether or not the dataset is balanced.



We can see from the above plot that the dataset is balanced, meaning that each illness has precisely 120 samples, and no additional balancing is necessary. We can see that our goal column, the prognosis column, has an object datatype, which is incompatible with machine learning training.

To transform the prognosis column to a numerical datatype, we'll use a label encoder. Label Encoder assigns a unique index to each label, converting it to numerical form. If there are n labels altogether, the numbers allocated to each label will range from 0 to n-1.

Splitting the data for training and testing the model:

Now that we've cleaned our data by removing Null values and converting the labels to numerical format, we can split it up to train and test the model. We will divide the data in an 80:20 ratio, which means that 80 percent of the data will be used to train the model and 20 percent will be used to evaluate the models' performance.

Model Building:

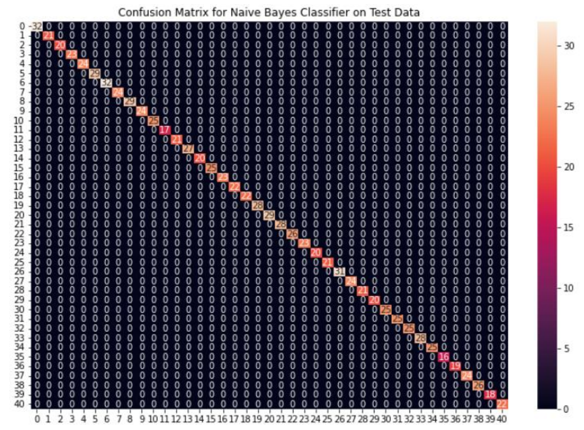
We will now begin working on the modelling portion of the project after we have split the data. To evaluate the machine learning models, we will use K-Fold cross-validation. For cross-validation, we will use the Decision Tree, the Gaussian Naive Bayes Classifier, and the Random Forest Classifier and the KNN classifier. Before we get into the implementation, let's review k-fold cross-validation and the mentioned machine learning models.

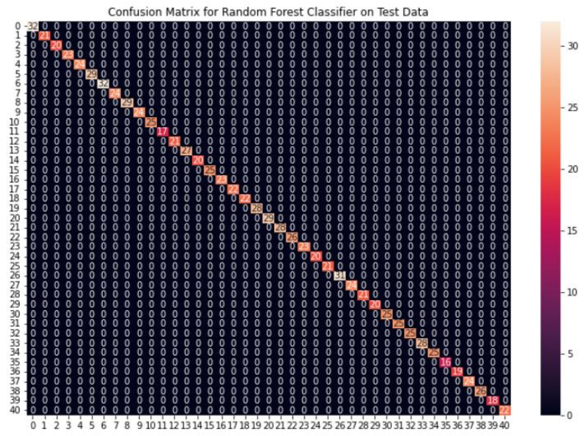
- K-Fold Cross-Validation: Cross-validation is a resampling technique used to evaluate machine learning models on a small sample of data. The procedure has a single parameter called k that specifies the number of groups into which a given data sample should be divided. As a result, the procedure is frequently referred to as k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the model's reference, for example, k=10 resulting in 10-fold cross-validation. Cross-validation is generally used in applied machine learning to measure the skill of a machine learning model on unseen data. That is, to use a small sample to estimate how the model is likely to perform in general when used to generate predictions on data that was not utilized during the model's training. It is a popular method because it is straightforward to grasp and produces a less biased or optimistic estimate of model competence than other methods, such as a simple train/test split.
- Decision Tree: A decision tree is a highly successful and versatile classification approach. It is utilized in visual pattern recognition and categorization. Because of its versatility, it is utilized for categorization in exceedingly complex problems. It is also capable of dealing with higher-dimensional challenges. It is made up of three parts: the root, the nodes, and the leaves. The root consists of the attribute that has the greatest influence on the outcome, the leaf tests for the value of a specific attribute, and the leaf outputs the tree's output.
- KNN: The K Nearest Neighbour algorithm is a supervised learning technique. It is a simple yet necessary algorithm. It is widely used in pattern recognition and data mining. It works by identifying a pattern in data that connects data to results, and it improves pattern recognition with each iteration.
- Gaussian Naïve Bayes: The Gaussian Naive Bayes algorithm is a series of algorithms that are based on the Naive Bayes theorem. They all agree on one thing, that is every pair of predictions is independent of each other. It also assumes that features contribute independently and equally to the prediction.
- Random Forest Algorithm: The Random Forest Technique is a supervised learning algorithm that can be used for classification as well as regression. This method is comprised of four fundamental phases -

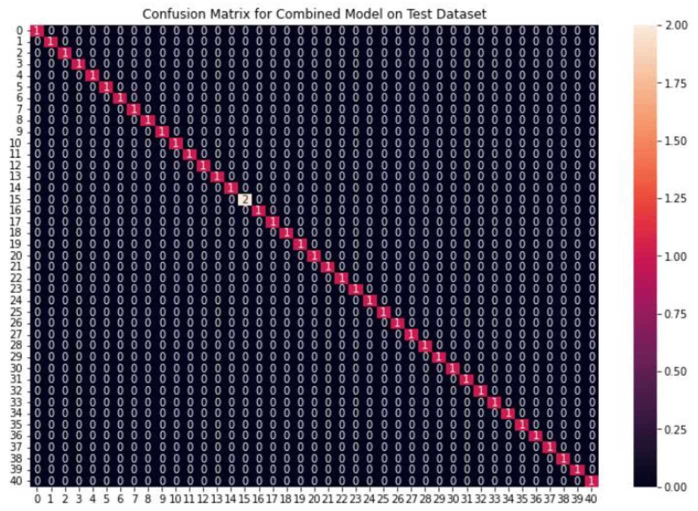
1. It selects data samples at random from a dataset.
2. It creates decision trees for each sample dataset that is selected.
3. Each anticipated result will be compiled and voted on at this stage.
4. Finally, the most popular forecast will be chosen and presented as the categorization result.

Evaluation Methods:

For the performance evaluation in this project we have used Confusion Matrix and Accuracy. A Confusion matrix is a N x N matrix that is used to evaluate the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values to the machine learning model's anticipated values. This provides us with a comprehensive picture of how well our classification model is working and the kind of errors it is producing.
The accuracy of a machine learning model is a metric for determining which model is the best at recognising correlations and patterns between variables in a dataset based on the input, or training, data. The stronger a model's generalisation to 'unseen' data is, the better predictions and insights it can provide, and hence the more business value it can give.

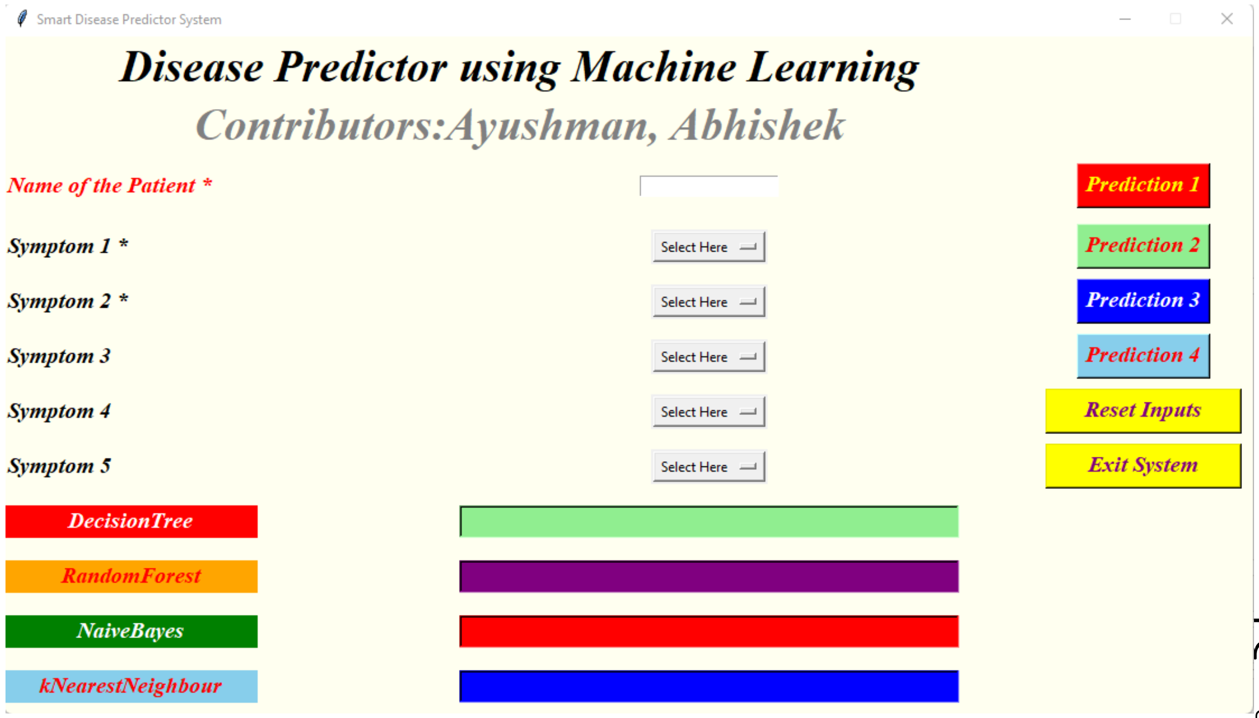


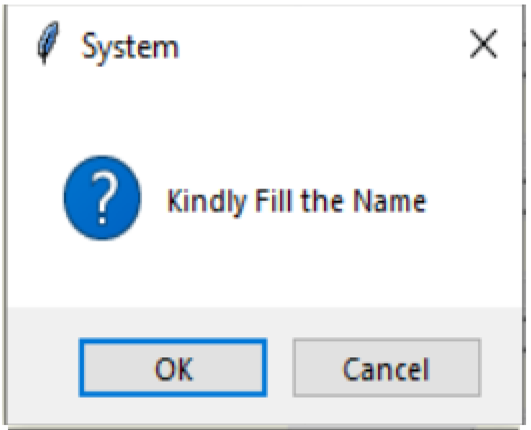




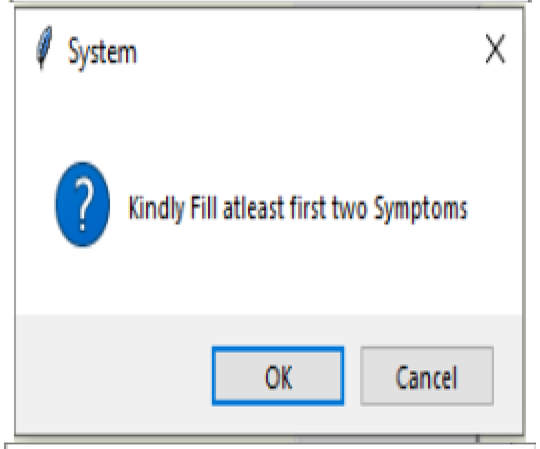
Building of GUI:

This project's GUI is a basic tkinter GUI with labels, messageboxes, buttons, text, titles, a nd option menus.
OptionMenu is used to create drop down menu.
Buttons are used to give functionalities and predict the outcome of models also two utility buttons namely exit and reset are also created.
Text is used to show output of the prediction using blank space.
Messagebox are used at three different places, one- to restrain the to enter name, two- to a sk for at least two symptoms, three- confirmation for exiting the system.

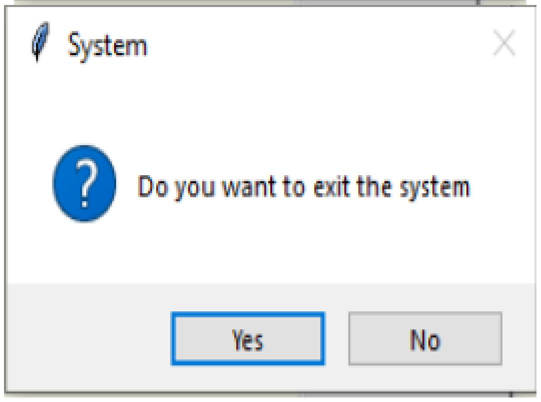



System

Kindly Fill the Name


System

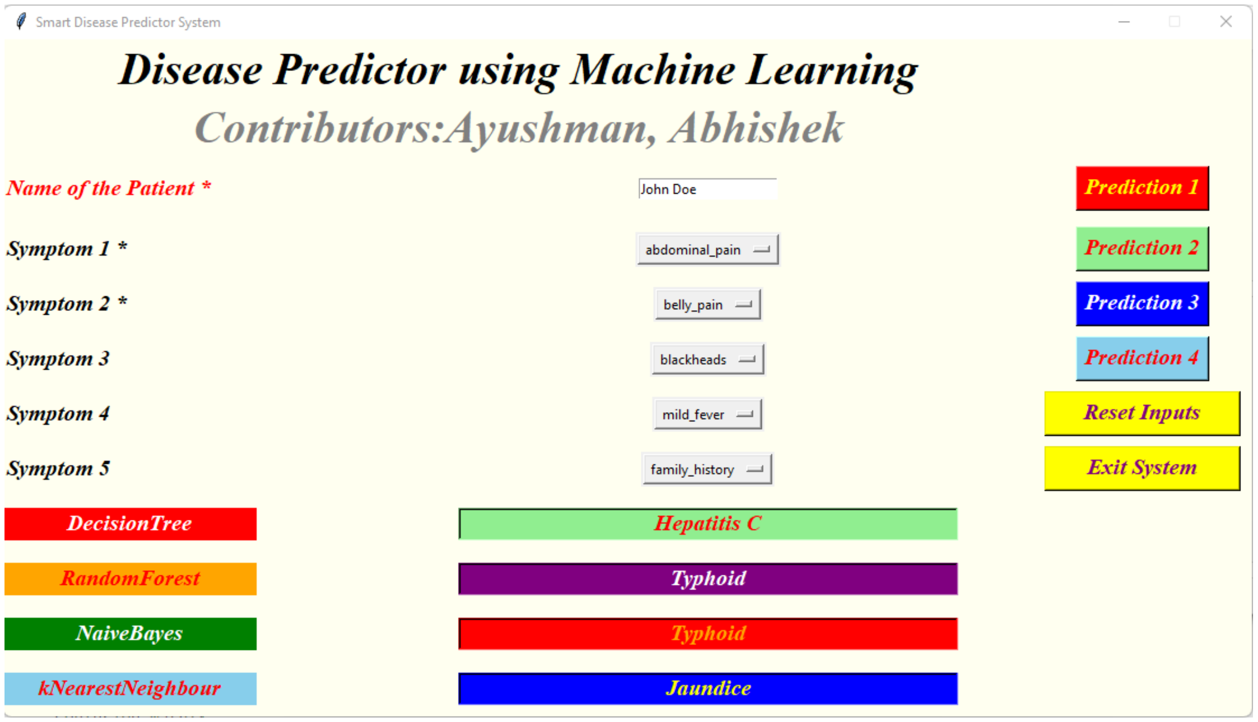
Kindly Fill atleast first two Symptoms


System

Do you want to exit the system

Result and Inference:

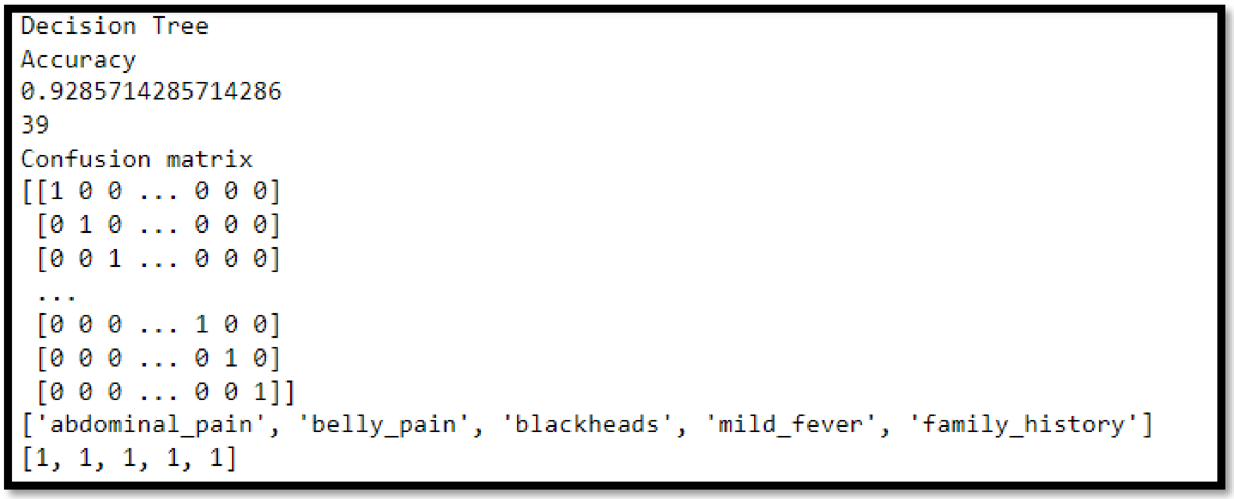
The disease prediction system was successfully built with the working GUI.


Smart Disease Predictor System
Disease Predictor using Machine Learning
Contributors:Ayushman, Abhishek
Prediction 1
Prediction 2
Prediction 3
Prediction 4
Reset Inputs
Exit System

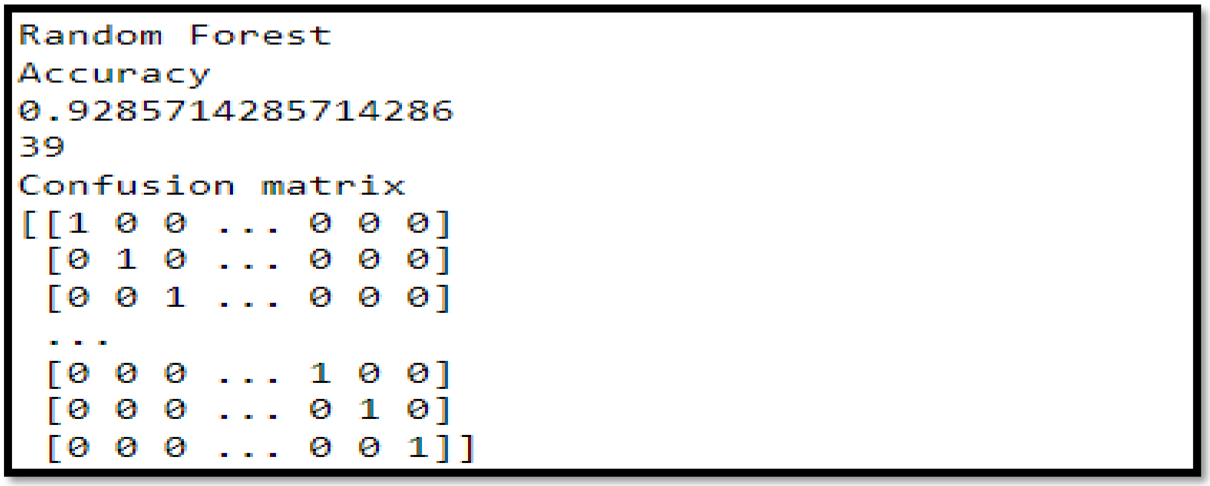
Name of the Patient * John Doe
Symptom 1 * abdominal_pain
Symptom 2 * belly_pain
Symptom 3 blackheads
Symptom 4 mild_fever
Symptom 5 family_history

DecisionTree Hepatitis C
RandomForest Typhoid
NaiveBayes Typhoid
kNearestNeighbour Jaundice

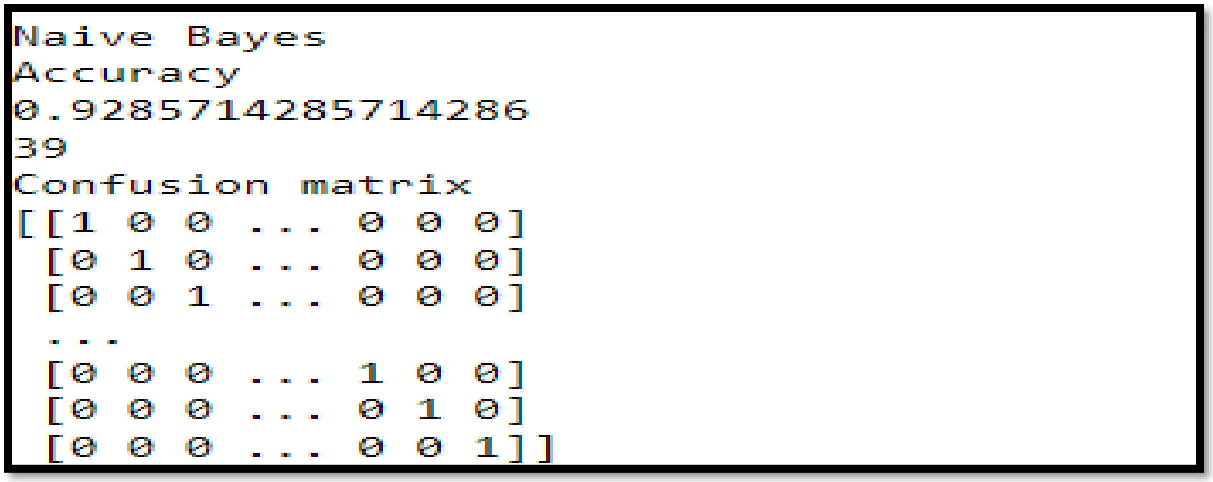
As seen above the GUI asks for the name of the patient and symptoms respectively. There can be seen four predictions with the four models used for this respective study. Metrics for the same have also been collected.



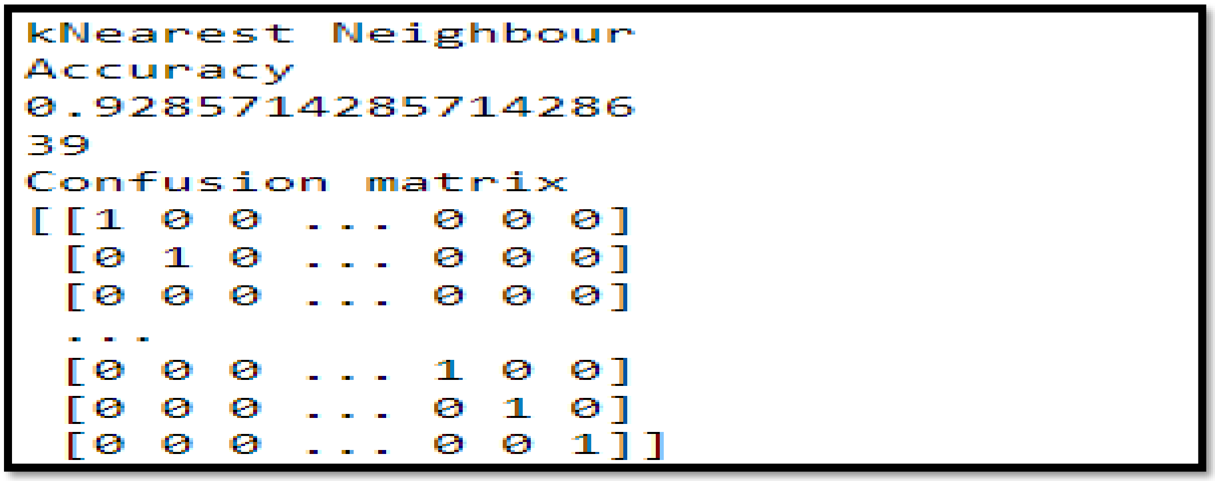
```
Decision Tree
Accuracy
0.9285714285714286
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
['abdominal_pain', 'belly_pain', 'blackheads', 'mild_fever', 'family_history']
[1, 1, 1, 1, 1]
```



```
Random Forest
Accuracy
0.9285714285714286
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]]
```

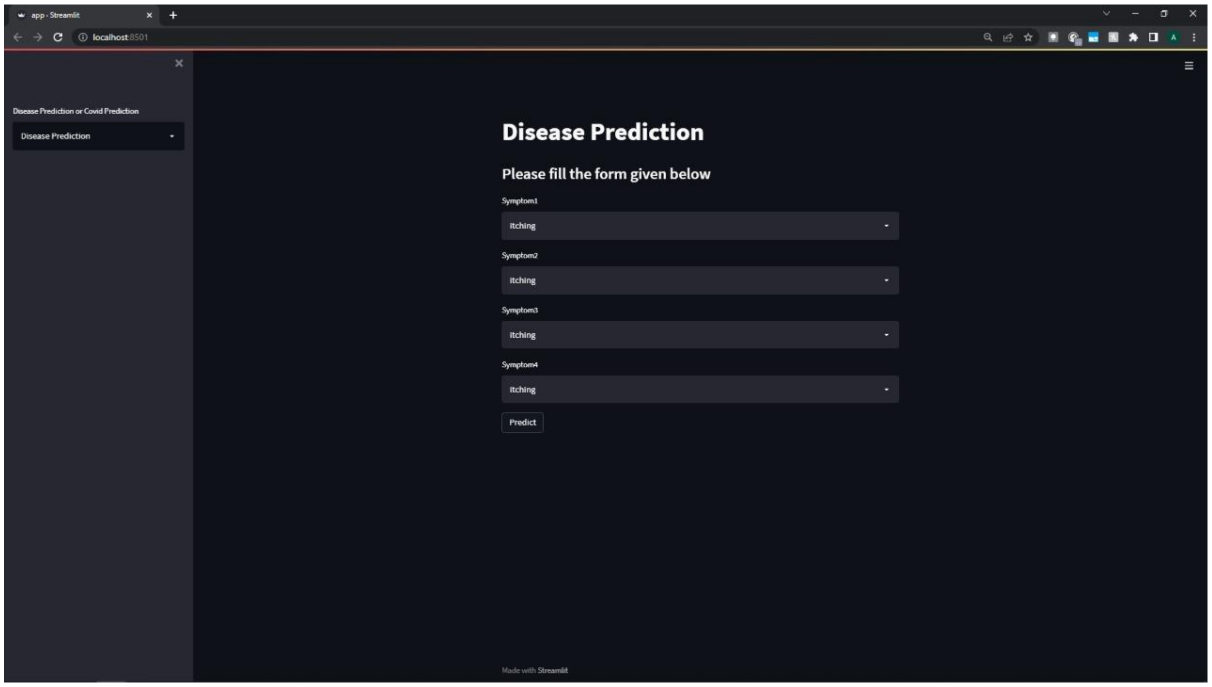


```
Naive Bayes
Accuracy
0.9285714285714286
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```



```
kNearest Neighbour
Accuracy
0.9285714285714286
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```

We can see that in all the four predictions we get an accuracy of almost 93%. So, the ques tion arises as to which model then suits the best for this type of an application. Given belo w is a brief analysis that we did of each algorithm used in this project.



Disease Prediction

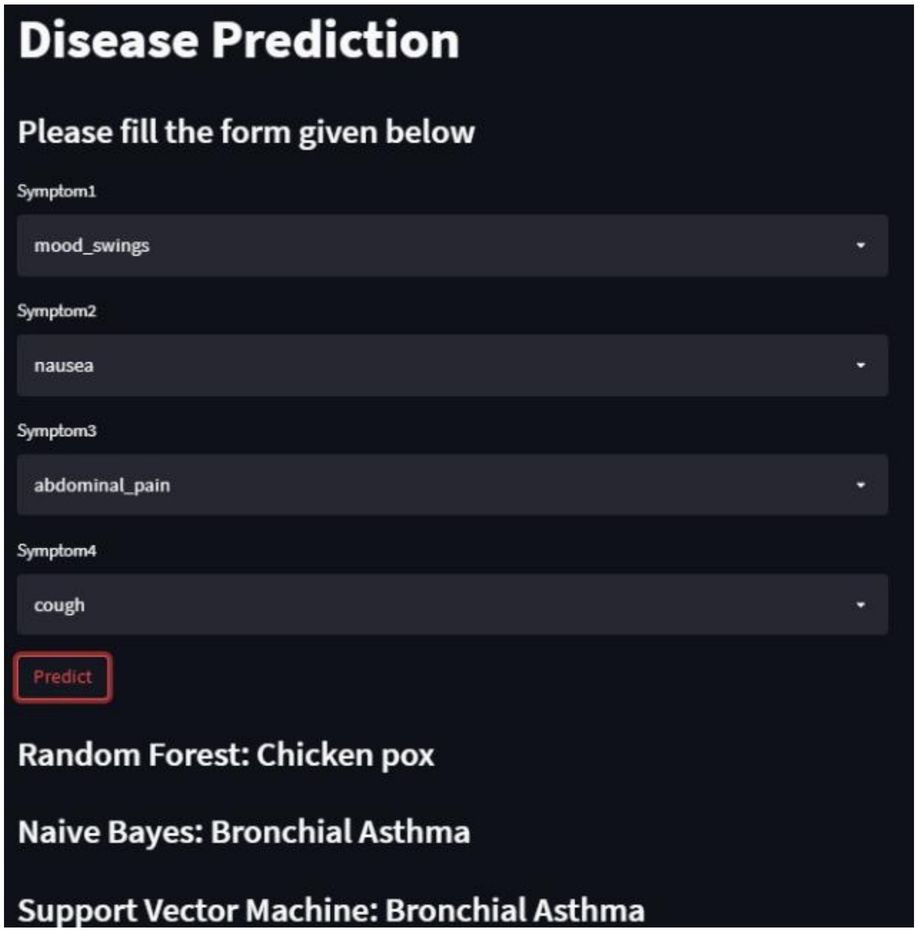
Please fill the form given below

Symptom1: itching

Symptom2: itching

Symptom3: itching

Symptom4: itching



Disease Prediction

Please fill the form given below

Symptom1: mood_swings

Symptom2: nausea

Symptom3: abdominal_pain

Symptom4: cough

Random Forest: Chicken pox

Naive Bayes: Bronchial Asthma

Support Vector Machine: Bronchial Asthma

Here we can see the simple website implementation of the same.

ANALYSIS:

- **<u>Decision Tree</u>**: Because the concept is commonly employed in our daily lives, the algorit hm is simple to comprehend, analyse, and visualise. Humans can readily comprehend the output of a Decision Tree. Decision Trees appear to be basic if-else statements that are str aightforward to comprehend. Because the Decision Tree employs a rule-based method rat her than distance computation, no feature scaling (standardisation and normalisation) is n ecessary. Missing values may be handled automatically using Decision Tree.
  When a new data point is added, the entire tree must be regenerated, and all nodes must b e computed and reconstructed. In most cases, using a Decision Tree results in data overfit ting. Overfitting causes a lot of variation in the output, which leads to a lot of inaccuracie s in the final estimation and displays a lot of inaccuracy in the findings. If the data set is e xtensive, a single tree may become complicated, resulting in overfitting.
- **<u>Random Forest</u>**: Both categorical and continuous variables function well with Random F orest. Missing values may be handled automatically using Random Forest. Unlike curve-based algorithms, non-linear parameters have no effect on the performance of a Random Forest. As a result, if the independent variables are very nonlinear, Random Forest may o utperform conventional curve-based methods. The Random Forest method has a high lev el of consistency. Even if a new data point is added to the dataset, the overall method rem ains unaffected since the new data may have an influence on one tree, but it is extremely unlikely to have an impact on all trees. Random Forest is less affected by noise than other methods.
  Random Forest generates a large number of trees (as opposed to a single tree as in a decis ion tree) and aggregates their results. In the Python sklearn package, it builds 100 trees by default. Random Forest takes a lot longer to train than decision trees since it creates a lot of trees (instead of just one) and makes decisions based on the majority of votes.
- **<u>Gaussian Naïve Bayes</u>**: A Naive Bayes classifier outperforms other models when the ass umption of independent predictors remains true. To estimate the test data, Naive Bayes re quires a modest quantity of training data. As a result, the training duration is shorter. Naiv e Bayes is similarly simple to use.
  The premise of independent predictors is the main imitation of Naive Bayes. All of the qu alities in Naive Bayes are assumed to be mutually independent. In actual life, getting a co llection of predictors that are totally independent is impossible. If a categorical variable in the test data set has a category that was not included in the training data set, the model wi ll assign a probability of 0 (zero) and will be unable to generate a prediction.
- **<u>KNN</u>**: Lazy Learner is the name given to KNN (Instance based learning). During the trai ning phase, it does not learn anything. The training data isn't used to construct any discri minative functions. In other words, it does not require any training. New data may be smo othly added without affecting the algorithm's accuracy. KNN may be implemented with o nly two parameters: the value of K and the distance function.

The cost of computing the distance between the new point and each old point is enormou s in big datasets, which lowers the algorithm's speed. The KNN algorithm does not operat e well with high-dimensional data because calculating the distance in each dimension bec omes challenging for the algorithm with a large number of dimensions. The KNN algorit hm is susceptible to dataset noise. Missing values must be manually imputed, and outliers must be removed. Before applying the KNN algorithm to any dataset, feature scaling (sta ndardisation and normalisation) is required. If we don't, KNN may make incorrect predict ions.

Conclusion

We set out to develop a system that can forecast disease based on symptoms provided to it. A system like this can cut the amount of time people spend in hospital emergency rooms and lower the workload on medical staff. We were successful in developing such a system, employing four different algorithms in the process. On average, we attained 93 percent accuracy. A system of this type can be substantially dependable given its performance.

When designing this system, we also included a method for storing the data input by the user in a database, which can be utilized in the future to help in the development of a better version of the system. Our system also offers a user-friendly interface. It also includes a variety of visual representations of the data collected and results achieved.

We were also able to see how different supervised machine learning algorithms work and analysis of the same was also done. This gave an insight as to how technologies like machine learning can help in the healthcare sector.

References:

1) Title: Web based disease prediction and recommender system
Author: Harish Rajora, N. Punn, S. K. Sonbhadra, Sonali Agarwal
Source:https://www.researchgate.net/publication/352209191_Web_based_disease_prediction_and_recommender_system
Date: Published 2021

2) Title: An Overview on Disease Prediction for Preventive Care of Health Deterioration
Author: Mohan Kumar K N, S.Sampath, Mohammed Imran
Source: https://www.ijeat.org/wp-content/uploads/papers/v8i5S/E10510585S19.pdf
Date: May 2019

3) Title: Use of Electronic Health Data for Disease Prediction
Author: Md Ekramul Hossain,Arif Khan,Mohammad Ali Moni,Shahadat Uddin
Source:https://www.researchgate.net/publication/335437382_Use_of_Electronic_Health_Data_for_Disease_Prediction_A_Comprehensive_Literature_Review
Date: August 2019

4) Title: PREDICTION OF DISEASES USING SUPERVISED LEARNING
Author: Ashish Kumar, Priya Ghansela, Purnima Soni , Chirag Goswami , Parasmani Sharma
Source:https://www.researchgate.net/publication/344296855_PREDICTION_OF_DISEASES_USING_SUPERVISED_LEARNING
Date: August 2020

5) Title: Disease Prediction by Machine Learning Over Big Data From Healthcare Communities.
Author: Min Chen, Yixue Hao , Kai Hwang , Lin Wang , Lu Wang
Source: https://ieeexplore.ieee.org/abstract/document/7912315
Date: April 2017

6) Title: Prediction of Diseases Using Machine Learning and Deep Learning
Author: Ch Aishwarya, K Suvarchala, B Aravind, G Shashank
Source: Prediction of Disease Using Machine Learning and Deep Learning | SpringerLink
Date: September 2020

7) Title: Disease prediction using Machine Learning
Author: Marouane Fethi Ferjani
Source: (PDF) Disease Prediction Using Machine Learning (researchgate.net)
Date: December 2020

8) Title: Designing Disease Prediction Model using ML approach
Author: Dhiraj Dahiwade, Prof Gajanan Patle, Prof Ektaa Meshraam
Source: IEEE Xplore Part Number: CFP19K25-ART; ISBN: 978-1-5386-7808-4
Date: June 2019

9) Title: Disease prediction from various symptoms using machine learning
Author: Rinkal Keniya, Aman Khakharia, Vruddhi Shah, Vrushabh Gada, Ruchi Manjalkar, Tirth Thaker, Mahesh Warang, Ninad Mehendale
Source: Disease Prediction From Various Symptoms Using Machine Learning by Rinkal Keniya, Aman Khakharia, Vruddhi Shah, Vrushabh Gada, Ruchi Manjalkar, Tirth Thaker, Mahesh Warang, Ninad Mehendale :: SSRN
Date: October 2020

10) Title: Development of machine learning model for diagnostic disease prediction based on lab tests
Author: Dong Jin Park, Min Woo Park, Homin Lee, Young-Jin Kim, Yeongsic Kim, Young Hoon Park
Source: Development of machine learning model for diagnostic disease prediction based on laboratory tests | Scientific Reports (nature.com)
Date: April 2021

11) Http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

12) https://issuu.com/ijraset/docs/review_of_machine_learning_techniques_for_crop_rec

13) http://theprofessionalspoint.blogspot.com/