**Disease Prediction Using Machine Learning**

*An Application Development Report Submitted*

In partial fulfillment of the requirement for the award of the degree of

## Bachelor of Technology

**In**

**Computer Science and Engineering -Artificial Intelligence and Machine Learning**

**By**

1. **Akshitha - 21N31A6601**

**CH. Harshini - 21N31A6639**

**B. Nirmala Bai - 21N31A6622**

Under the Guidance of

**Ms. P. Swapna**

**Assistant Professor**

**Computational Intelligence Department**

**MRCET**



**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE**

**MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

(Affiliated to JNTU, Hyderabad)

**ACCREDITED by AICTE-NBA**

**Maisammaguda, Dhulapally post, Secunderabad-500014.**

2020-2024

DECLARATION

I hereby declare that the project entitled “Disease Prediction Using Machine Learning” submitted to Malla Reddy College of Engineering and Technology, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering- Artificial Intelligence and Machine Learning is a result of original research work done by me.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

A. Akshitha(21N31A6601)

CH. Harshini(21N31A6639)

B. Nirmala Bai (21N31A6622)



**CERTIFICATE**

This is to certify that this is the bonafide record of the project titled **“Disease Prediction Using Machine Learning”** submitted by **A. Akshitha (21N31A6601), CH. Harshini(21N31A6639), B. Nirmala Bai(21N31A6622),** of B.Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2023-2024. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

**Ms. P. Swapna Dr. D. Sujatha**

Assistant Professor Professor

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

**EXTERNAL EXAMINER**

# Acknowledgement

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide **Ms**. **P. Swapna (Assistant. Professor),** for her structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our application development a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the application development.

By

A. Akshitha(21N31A6601)

CH. Harshini(21N31A6639)

B. Nirmala Bai (21N31A6622)

**ABSTRACT**

With an increase in biomedical and healthcare data, accurate analysis of medical data benefits early disease detection and patient care. For small problems, the users have to go personally to the hospital for check-up which is more time consuming. Such a problem can be solved by using disease prediction applications by giving proper guidance regarding healthy living. Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user with the help of GUI. The system processes the symptoms provided by the user as input and gives the output as the probability of the disease. Over the past decade, the use of the specific disease prediction tools along with the concerning health has been increased due to a variety of diseases and less doctor-patient ratio. Thus, in this system, we are concentrating on providing immediate and accurate disease prediction to the users about the symptoms they enter. For prediction of diseases, different machine learning algorithms are used to ensure quick and accurate predictions. In one channel, the symptoms entered will be cross checked with the database.

**TABLE OF CONTENTS**

**S.No. Topic Page No.**

CHAPTER 1: INTRODUCTION 1-4

1.1 Purpose 2

1.2 Background of project 2

1.3 Scope of project 3

1.4 Project features 4

CHAPTER 2: SYSTEM REQUIREMENTS 5-6

2.1 Hardware Requirements 5

2.2 Software Requirements 5

2.3 Existing System 5

2.4 Proposed System 6

CHAPTER 3: TECHNOLOGIES AND LANGUAGES USED 7-9

CHAPTER 4: SYSTEM DESIGN 10-13

4.1 Architecture 10

4.2 UML Diagrams 11

CHAPTER 5: IMPLEMENTATION 14-29

5.1 Code 14

5.2 Output 28

CHAPTER 6: CONCLUSION AND FUTURE SCOPE 29-30

6.1 Conclusion 29

6.2 Future Scope 29

BIBLIOGRAPHY 31

**CHAPTER-1**

**INTRODUCTION**

Today healthcare industry has become a big making business. The healthcare industry uses and produces quite a large amount of data which can be used to extract information about a particular disease for a patient. This information of healthcare will further be used for effective and best possible treatment for patient’s health. This area also needs some improvement by using the informative data in healthcare sciences. But a major challenge is to extract the information from the data because the data is present in a huge amount so some data mining and machine learning techniques are used. The expected result and of this project is to predict the disease beforehand so that the risk of life can be prevented at an early stage and save life of people and the cost of treatment can be reduced to a particular extent. In India also we should adopt the non-manual system of medical treatment which is the best for improving and understanding the human health. The main motive is to use the concept of machine learning in healthcare to improvise the treatment of patients. Machine learning has already made it much more easier to identify and predict various diseases. Predictive analysis of the disease with the help of many machine learning algorithms helps us to predict the disease and helps in treating the patients in an effective manner. Disease prediction using machine learning also uses the patient history and health data by applying various concepts like data mining and machine learning techniques and also some algorithm. Many works have also applied data mining techniques to the pathological data for prediction of some particular diseases. These approaches were intended to before- hand predict the re-occurrence of certain diseases. Also, some approaches tried to do prediction while controlling the disease. The recent work of deep learning was in disparate areas of machine learning which have driven a shift to machine learning models that can learn and understand the hierarchical representations of raw data with some pre-processing. With the development of this concept called big data technology, more attention is paid to disease prediction.

* 1. **Purpose:**

**Early Detection**: ML algorithms can analyze large amounts of data to identify patterns and indicators of diseases at early stages, often before noticeable symptoms appear. This early detection can significantly improve treatment outcomes and prognosis.

**Personalized Medicine**: ML models can analyze individual patient data, including medical history, genetics, lifestyle factors, and symptoms, to provide personalized predictions and recommendations for disease prevention and management.

**Decision Support**: Healthcare providers can use ML-powered disease prediction tools to support their clinical decision-making process. By integrating predictive models into clinical workflows, healthcare professionals can receive real-time insights and recommendations to guide patient care.

**Public Health Surveillance**: ML algorithms can analyze population-level health data, such as electronic health records, environmental factors, and social determinants of health, to predict disease outbreaks, assess disease risk factors, and allocate resources effectively for disease prevention and control.

**Patient Empowerment**: GUIs provide an intuitive interface for users to interact with ML models without requiring expertise in data science or programming. By presenting predictions and insights in a user-friendly manner, patients can better understand their health risks and take proactive steps to maintain their well-being.

**Research and Development**: Disease prediction models developed using ML techniques can contribute to ongoing research efforts in understanding disease mechanisms, identifying novel biomarkers, and developing new therapeutic interventions.

**1.2 Background of project:**

The background of a project focused on disease prediction using machine learning (ML) typically emerges from the urgent need within healthcare systems to enhance early detection, treatment planning, and public health strategies. With the proliferation of digital health records, medical imaging data, and genetic information, there's a wealth of data available for analysis, offering unprecedented opportunities to develop predictive models. Such projects are often driven by the desire to address specific healthcare challenges, whether it's improving diagnostic accuracy, identifying at-risk populations, or optimizing resource allocation. Collaborative efforts between healthcare professionals, data scientists, and technologists are instrumental in leveraging ML algorithms effectively. Previous research in the field provides valuable insights and methodologies, serving as a foundation for novel approaches. Ultimately, the goal of these projects is to translate advanced ML techniques into practical solutions that can positively impact patient outcomes, healthcare delivery, and population health management.

**1.3 Scope of project:**

The scope of disease prediction using machine learning (ML) is expansive, encompassing a multifaceted approach to healthcare and medical research. ML algorithms leverage vast datasets, including patient records, genetic information, environmental factors, and lifestyle habits, to detect disease patterns and predict outcomes with increasing accuracy. This predictive capability extends across various domains, from early detection and personalized treatment plans to population-level risk assessments and public health interventions. ML facilitates proactive healthcare strategies by identifying high-risk individuals, optimizing resource allocation, and informing clinical decision-making. Furthermore, ML-driven disease prediction holds promise for advancing medical research, drug discovery, and healthcare system efficiency, ultimately contributing to improved patient outcomes, reduced healthcare costs, and better overall public health.

**1.4 Project Features:**

**Data Collection and Integration**: The project involves collecting and integrating diverse datasets relevant to the target disease(s), including electronic health records, medical imaging data, genetic information, environmental factors, and patient demographics.

**Data Preprocessing**: This feature encompasses tasks such as data cleaning, normalization, feature engineering, and handling missing values to prepare the data for analysis and model training.

**Feature Selection and Extraction**: Identifying the most relevant features or variables from the dataset that contribute significantly to predicting the target disease outcome. This step helps reduce dimensionality and focus on the most informative aspects of the data.

**Model Selection**: Selecting appropriate ML algorithms and techniques for building predictive models based on the characteristics of the dataset, such as logistic regression, decision trees, random forests, support vector machines, neural networks, or ensemble methods.

**Model Training**: This involves training the selected ML models using labeled data to learn the relationships between input features and the target disease outcome. Techniques such as cross-validation and hyperparameter tuning may be employed to optimize model performance.

**Model Evaluation**: Assessing the performance of the trained ML models using evaluation metrics such as accuracy, precision, recall, F1-score, area under the receiver operating characteristic curve (AUC-ROC), and confusion matrices on validation or test datasets.

**Predictive Analytics**: Deploying the trained ML models to make predictions on new, unseen data to identify individuals at risk of developing the target disease(s) or to predict disease outcomes, progression, or treatment responses.

**Visualization and Interpretation**: Providing visualizations and interpretability tools to understand the predictions made by the ML models, including feature importance, decision boundaries, and model explanations, to aid clinicians, researchers, and end-users in understanding and trust the predictions.

**Integration with Healthcare Systems**: Integrating the ML-based disease prediction models into existing healthcare systems or developing standalone applications with user-friendly interfaces to facilitate adoption and usage by healthcare providers, researchers, and other stakeholders.

**Continuous Monitoring and Updating**: Monitoring the performance of the deployed ML models over time and updating them as new data becomes available or as the underlying data distribution changes to ensure that the models remain accurate and effective in predicting disease outcomes.

**CHAPTER-2**

**SYSTEM REQUIREMENTS**

**2.1 Hardware Requirements:**

* Processor: i5 and above
* Ram: 8gb and above
* Hard Disk: 25 GB in local drive

**2.2 Software requirements:**

* Graphical user interface
* Python
* VScode

**2.3 Existing System:**

In the existing system the data set is typically small, for patients and diseases with specific conditions. These systems are mostly designed for the more colossal diseases such as Heart Disease, Cancer etc. The pre-selected characteristics may sometimes not satisfy the changes in the disease and its influencing factors which could lead to inaccuracy in results. As we live in a continuously evolving world, the symptoms of diseases also evolve over a course of time. Also most of the current systems make the users wait for long periods by making them answer lengthy questionnaires.

**Drawbacks of existing system:**

1.It is a very time consuming process as we need to visit the doctor and it is not trained well.

2.There are not much disease present in the dataset and it is not valid for upcoming days.

3.It does not give accurate results and accuracy.

**2.4 Proposed System:**

Here, In proposing such a system which will flaunt a simple and elegant User Interface and also be time efficient. In order to make it less time consuming we are aiming at a more specific questionnaire which will be followed by the system. Our aim with this system is to be the connecting bridge between doctors and patients. The main feature will be the machine learning, in which we will be using algorithms such as Naïve Bayes Algorithm, K-Nearest Algorithm, Decision Tree Algorithm, Random Forest Algorithm and Support Vector Machine, which will help us in getting accurate predictions and Also, will find which algorithm gives a faster and efficient result by comparatively-comparing.

**CHAPTER-3**

**TECHNOLOGIES USED**

**Technologies and Languages used to Develop**

**Algorithms:**

**Random Forest** is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

**K-Nearest Neighbour** is one of the simplest Machine Learning algorithms based on Supervised Learning technique.K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

**Naïve Bayes algorithm** is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

**Decision Tree** is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

**GUI:** A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information and represent actions that the user can take. The objects change color, size, or visibility when the user interacts with them.

The GUI was created at Xerox PARC (Palo Alto Research Center) by Alan Kay, Douglas Engelbart, and other researchers in 1981. Later, Apple introduced the Lisa computer with a GUI on January 19, 1983.

**NUMPY:** NumPy is a powerful numerical computing library for Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy

**TKINTER:** Tkinter is a Python library for creating graphical user interfaces (GUIs). It provides a set of tools to create windows, dialogs, buttons, and other GUI elements. Tkinter is based on the Tk GUI toolkit and is included with most Python installations. It's beginner-friendly and widely used for developing desktop applications with Python.

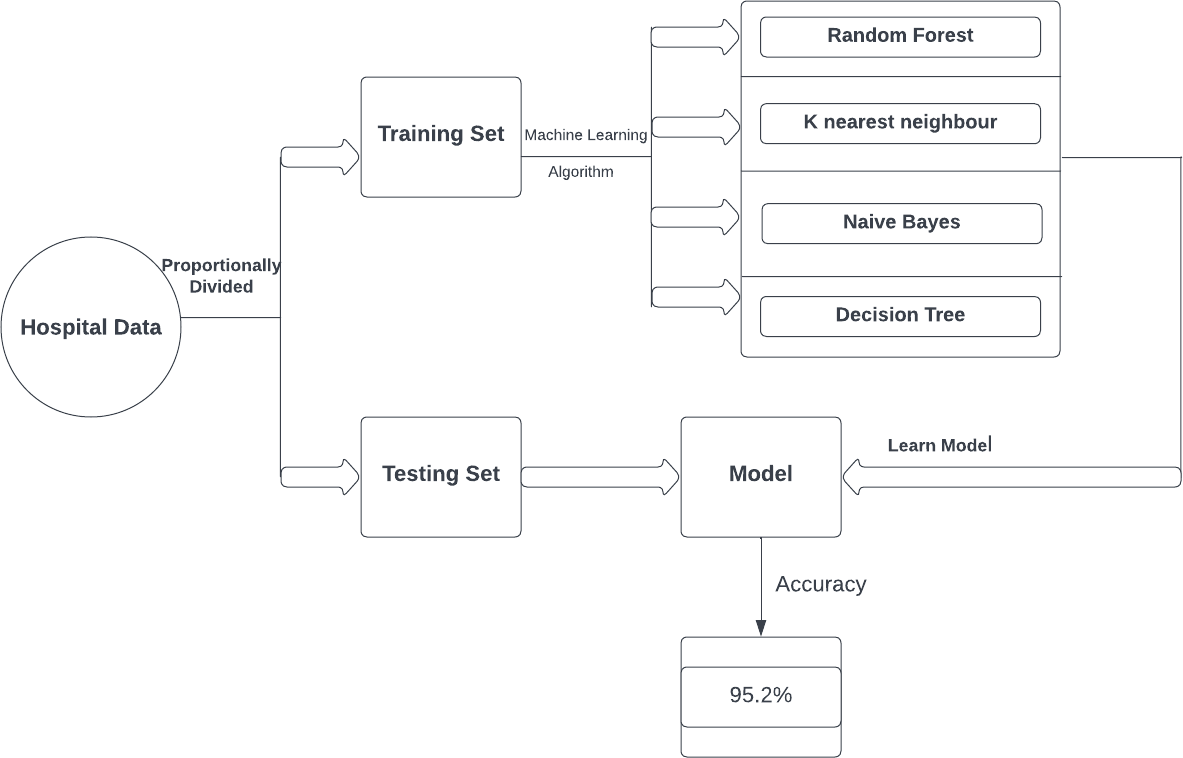
**PYTHON:** Python is a general-purpose interpreted, interactive, object-oriented, and high- level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems.

**Machine learning (ML)** ML algorithm is a process or set of processes that help the model to adapt to our dataset. An ML algorithm will specify the way the data is transformed from input to output and how the model will learn the appropriate transformation from input to the final output.

**CHAPTER-4**

**SYSTEM DESIGN**

**4.1 System Architecture**

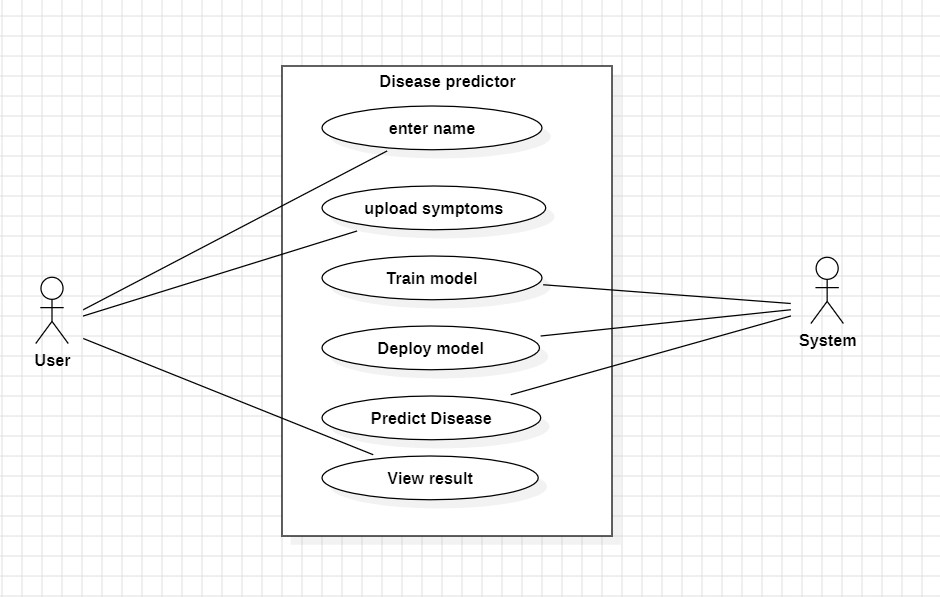


**Fig:1 Architecture**

**4.2 UML Diagrams**

**4.2.1 Use case diagram**

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

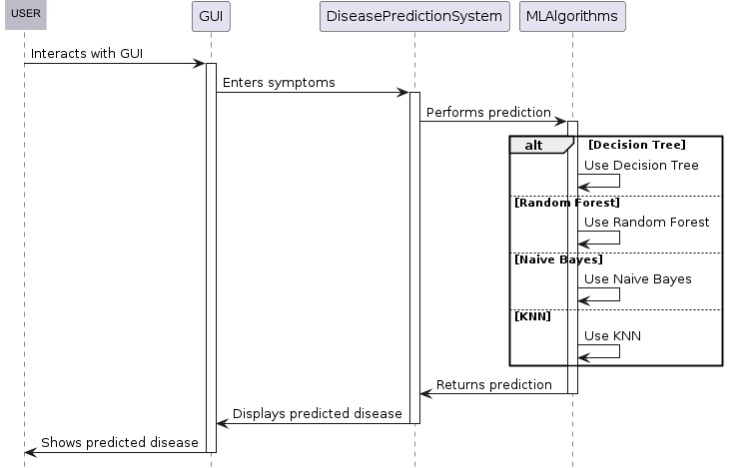


**Fig.2 Use Case diagram**

**4.2.2 Sequence diagram**

A sequence diagram is a type of interaction diagram because it describes how and in what order

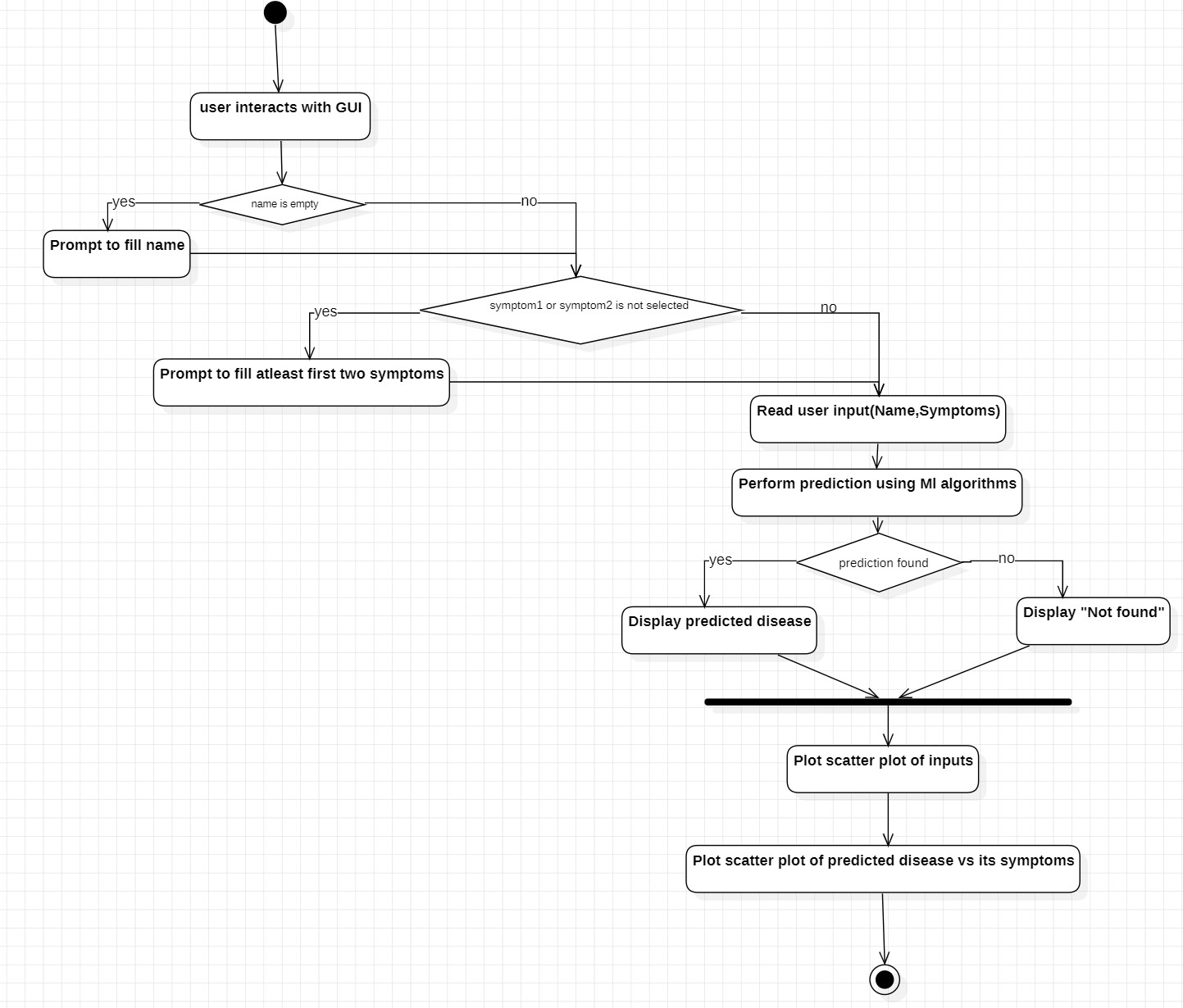
a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

****

**Fig.4 Sequence Diagram**

**4.2.3 Activity diagram**

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behavior diagram' because it describes what should happen in the modeled system.



**Fig.5 Activity Diagram**

**CHAPTER-5**

**IMPLEMENTATION**

**5.1 Code**

#Importing Libraries from matplotlib to visualize the data

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

#Importing Libraries to create GUI

from tkinter import \*

#Importing Libraries to perform calculations

import numpy as np

import pandas as pd

import os

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

l1=['back\_pain','constipation','abdominal\_pain','diarrhoea','mild\_fever','yellow\_urine',

'yellowing\_of\_eyes','acute\_liver\_failure','fluid\_overload','swelling\_of\_stomach',

'swelled\_lymph\_nodes','malaise','blurred\_and\_distorted\_vision','phlegm','throat\_irritation',

'redness\_of\_eyes','sinus\_pressure','runny\_nose','congestion','chest\_pain','weakness\_in\_limbs',

'fast\_heart\_rate','pain\_during\_bowel\_movements','pain\_in\_anal\_region','bloody\_stool',

'irritation\_in\_anus','neck\_pain','dizziness','cramps','bruising','obesity','swollen\_legs',

'swollen\_blood\_vessels','puffy\_face\_and\_eyes','enlarged\_thyroid','brittle\_nails',

'swollen\_extremeties','excessive\_hunger','extra\_marital\_contacts','drying\_and\_tingling\_lips',

'slurred\_speech','knee\_pain','hip\_joint\_pain','muscle\_weakness','stiff\_neck','swelling\_joints',

'movement\_stiffness','spinning\_movements','loss\_of\_balance','unsteadiness',

'weakness\_of\_one\_body\_side','loss\_of\_smell','bladder\_discomfort','foul\_smell\_of urine',

'continuous\_feel\_of\_urine','passage\_of\_gases','internal\_itching','toxic\_look\_(typhos)',

'depression','irritability','muscle\_pain','altered\_sensorium','red\_spots\_over\_body','belly\_pain',

'abnormal\_menstruation','dischromic \_patches','watering\_from\_eyes','increased\_appetite','polyuria','family\_history','mucoid\_sputum',

'rusty\_sputum','lack\_of\_concentration','visual\_disturbances','receiving\_blood\_transfusion',

'receiving\_unsterile\_injections','coma','stomach\_bleeding','distention\_of\_abdomen',

'history\_of\_alcohol\_consumption','fluid\_overload','blood\_in\_sputum','prominent\_veins\_on\_calf',

'palpitations','painful\_walking','pus\_filled\_pimples','blackheads','scurring','skin\_peeling',

'silver\_like\_dusting','small\_dents\_in\_nails','inflammatory\_nails','blister','red\_sore\_around\_nose',

'yellow\_crust\_ooze']

#List of Diseases is listed in list disease.

disease=['Fungal infection','Allergy','GERD','Chronic cholestasis','Drug Reaction',

'Peptic ulcer diseae','AIDS','Diabetes','Gastroenteritis','Bronchial Asthma','Hypertension',

' Migraine','Cervical spondylosis',

'Paralysis (brain hemorrhage)','Jaundice','Malaria','Chicken pox','Dengue','Typhoid','hepatitis A',

'Hepatitis B','Hepatitis C','Hepatitis D','Hepatitis E','Alcoholic hepatitis','Tuberculosis',

'Common Cold','Pneumonia','Dimorphic hemmorhoids(piles)',

'Heartattack','Varicoseveins','Hypothyroidism','Hyperthyroidism','Hypoglycemia','Osteoarthristis',

'Arthritis','(vertigo) Paroymsal Positional Vertigo','Acne','Urinary tract infection','Psoriasis',

'Impetigo']

l2=[]

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

l2.append(0)

print(l2)

#Reading the training .csv file

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

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

df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,

'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,

'Migraine':11,'Cervical spondylosis':12,

'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,

'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,

'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,

'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,

'(vertigo) Paroymsal Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,

'Impetigo':40}},inplace=True)

#printing the top 5 rows of the training dataset

df.head()

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

def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):

nunique = df1.nunique()

df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values

nRow, nCol = df1.shape

columnNames = list(df1)

nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow

plt.figure(num = None, figsize = (6 \* nGraphPerRow, 8 \* nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')

for i in range(min(nCol, nGraphShown)):

plt.subplot(nGraphRow, nGraphPerRow, i + 1)

columnDf = df.iloc[:, i]

if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):

valueCounts = columnDf.value\_counts()

valueCounts.plot.bar()

else:

columnDf.hist()

plt.ylabel('counts')

plt.xticks(rotation = 90)

plt.title(f'{columnNames[i]} (column {i})')

plt.tight\_layout(pad = 1.0, w\_pad = 1.0, h\_pad = 1.0)

plt.show()

# Scatter and density plots

def plotScatterMatrix(df1, plotSize, textSize):

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

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

df1 = df1.dropna('columns')

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

columnNames = list(df)

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

columnNames = columnNames[:10]

df1 = df1[columnNames]

ax = pd.plotting.scatter\_matrix(df1, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')

corrs = df1.corr().values

for i, j in zip(\*plt.np.triu\_indices\_from(ax, k = 1)):

ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)

plt.suptitle('Scatter and Density Plot')

plt.show()

plotPerColumnDistribution(df, 10, 5)

plotScatterMatrix(df, 20, 10)

X= df[l1]

y = df[["prognosis"]]

np.ravel(y)

print(X)

print(y)

#Reading the testing.csv file

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

#Using inbuilt function replace in pandas for replacing the values

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,

'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,

'Migraine':11,'Cervical spondylosis':12,

'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,

'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,

'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,

'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,

'(vertigo) Paroymsal Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,

'Impetigo':40}},inplace=True)

#printing the top 5 rows of the testing data

tr.head()

plotPerColumnDistribution(tr, 10, 5)

plotScatterMatrix(tr, 20, 10)

X\_test= tr[l1]

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

np.ravel(y\_test)

print(X\_test)

print(y\_test)

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

def scatterplt(disea):

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

x.drop(x[x==0].index,inplace=True)#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()

root = Tk()

pred1=StringVar()

def DecisionTree():

if len(NameEn.get()) == 0:

pred1.set(" ")

comp=messagebox.askokcancel("System","Kindly Fill the Name")

if comp:

root.mainloop()

elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):

pred1.set(" ")

sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")

if sym:

root.mainloop()

else:

print(NameEn.get())

from sklearn import tree

clf3 = tree.DecisionTreeClassifier()

clf3 = clf3.fit(X,y)

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

y\_pred=clf3.predict(X\_test)

print("Decision Tree")

print("Accuracy")

print(accuracy\_score(y\_test, y\_pred))

print(accuracy\_score(y\_test, y\_pred,normalize=False))

print("Confusion matrix")

conf\_matrix=confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

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

for k in range(0,len(l1)):

for z in psymptoms:

if(z==l1[k]):

l2[k]=1

inputtest = [l2]

predict = clf3.predict(inputtest)

predicted=predict[0]

h='no'

for a in range(0,len(disease)):

if(predicted == a):

h='yes'

break

if (h=='yes'):

pred1.set(" ")

pred1.set(disease[a])

else:

pred1.set(" ")

pred1.set("Not Found")

#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())

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")

#printing scatter plot of disease predicted vs its symptoms

scatterplt(pred2.get())

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")

#printing scatter plot of disease predicted vs its symptoms

scatterplt(pred3.get())

pred4=StringVar()

def KNN():

if len(NameEn.get()) == 0:

pred1.set(" ")

comp=messagebox.askokcancel("System","Kindly Fill the Name")

if comp:

root.mainloop()

elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):

pred1.set(" ")

sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")

if sym:

root.mainloop()

else:

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n\_neighbors=5,metric='minkowski',p=2)

knn=knn.fit(X,np.ravel(y))

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

y\_pred=knn.predict(X\_test)

print("KNN")

print("Accuracy")

print(accuracy\_score(y\_test, y\_pred))

print(accuracy\_score(y\_test, y\_pred,normalize=False))

print("Confusion matrix")

conf\_matrix=confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

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

for k in range(0,len(l1)):

for z in psymptoms:

if(z==l1[k]):

l2[k]=1

inputtest = [l2]

predict = knn.predict(inputtest)

predicted=predict[0]

h='no'

for a in range(0,len(disease)):

if(predicted == a):

h='yes'

break

if (h=='yes'):

pred4.set(" ")

pred4.set(disease[a])

else:

pred4.set(" ")

pred4.set("Not Found")

#printing scatter plot of disease predicted vs its symptoms

scatterplt(pred4.get())

#Tk class is used to create a root window

root.configure(background='Ivory')

root.title('Smart Disease Predictor System')

root.resizable(0,0)

#taking first input as symptom

Symptom1 = StringVar()

Symptom1.set("Select Here")

#taking second input as symptom

Symptom2 = StringVar()

Symptom2.set("Select Here")

#taking third input as symptom

Symptom3 = StringVar()

Symptom3.set("Select Here")

#taking fourth input as symptom

Symptom4 = StringVar()

Symptom4.set("Select Here")

#taking fifth input as symptom

Symptom5 = StringVar()

Symptom5.set("Select Here")

Name = StringVar()

#function to Reset the given inputs to initial position

prev\_win=None

def Reset():

global prev\_win

Symptom1.set("Select Here")

Symptom2.set("Select Here")

Symptom3.set("Select Here")

Symptom4.set("Select Here")

Symptom5.set("Select Here")

NameEn.delete(first=0,last=100)

pred1.set(" ")

pred2.set(" ")

pred3.set(" ")

pred4.set(" ")

try:

prev\_win.destroy()

prev\_win=None

except AttributeError:

pass

#Exit button to come out of system

from tkinter import messagebox

def Exit():

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

if qExit:

root.destroy()

exit()

#Headings for the GUI written at the top of GUI

w2 = Label(root, justify=LEFT, text="Disease Predictor using Machine Learning", fg="Red", 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: Sudhanshu,Rohan,Aditya", fg="Pink", 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

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 algorithms

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)

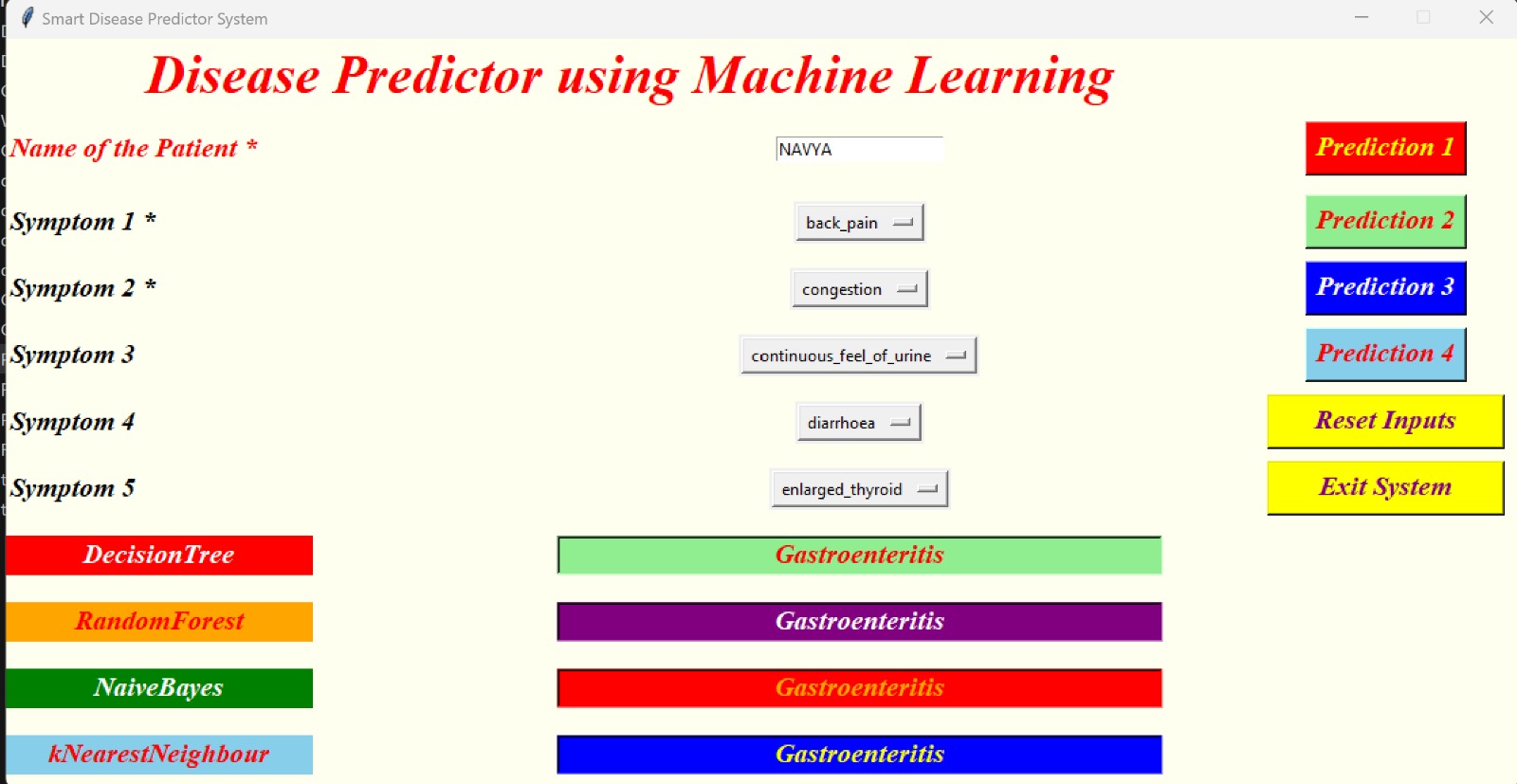
t4=Label(root,font=("Times",15,"bold italic"),text="kNearest Neighbour",height=1,bg="Blue"

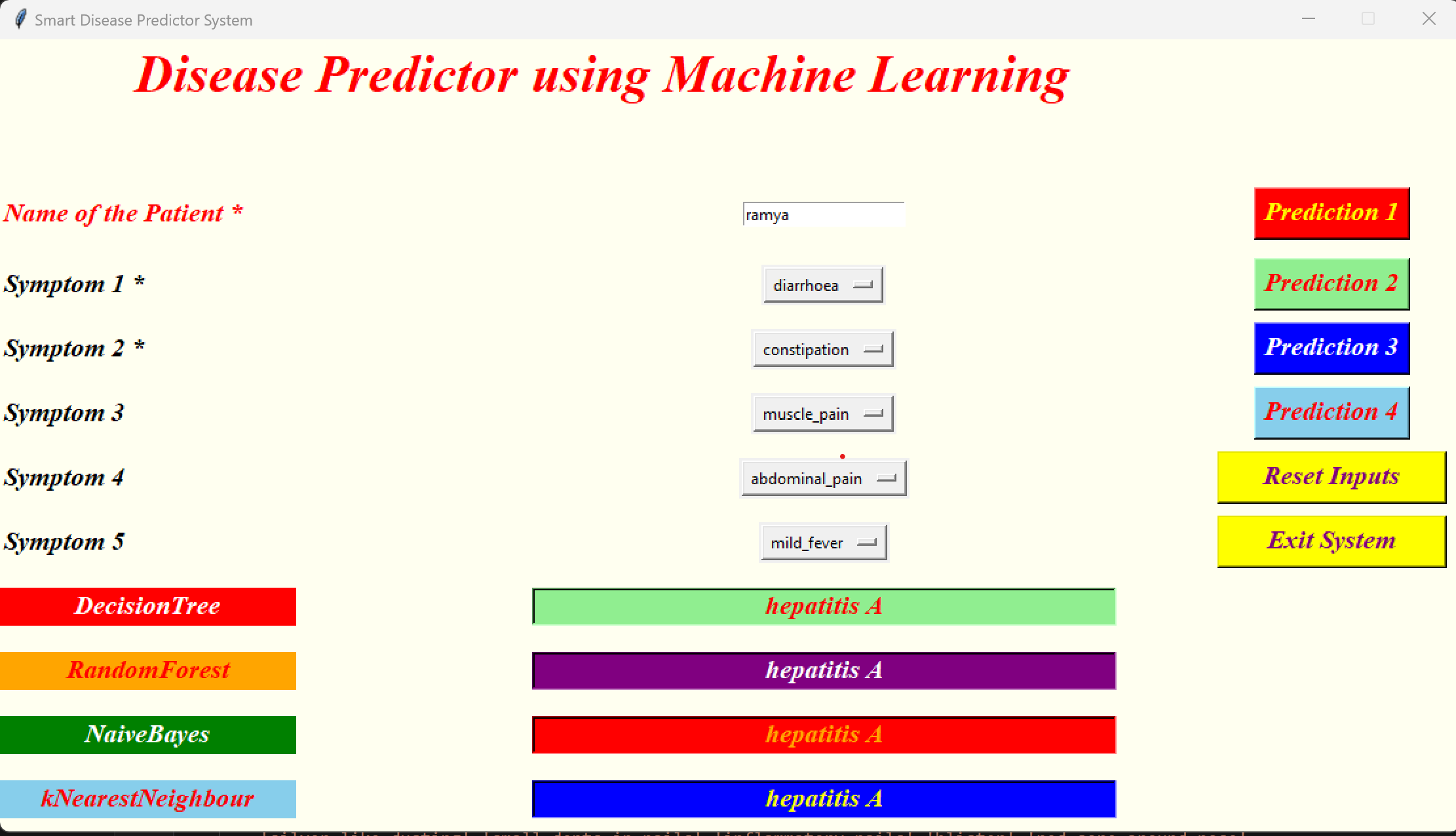
,width=40,fg="yellow",textvariable=pred4,relief="sunken").grid(row=21, column=1, padx=10)

#calling this function because the application is ready to run

root.mainloop()

**5.2 Output**





**CHAPTER-6**

**CONCLUSION & FUTURE SCOPE**

**6.1 Conclusion**

Finally, I'll say that this project's Diseases prediction system is very helpful in everyone's day-to-day life and is particularly significant for the healthcare industry, as they are the ones who regularly use these systems to predict the illnesses of the patients based on their general information and their experienced symptoms. Today, the health sector plays a significant role in helping patients recover from their illnesses, so this is a way for the sector to inform users and is also helpful to users who don't want to visit a hospital or other facility. Users can use this by entering their symptoms and any other pertinent data. By just asking the user for their symptoms and entering them into the system, the system may identify the exact and, to some extent, the accurate diseases within a matter of seconds, providing the health industry with benefits as well. The job of the doctors can be decreased and they will be able to accurately forecast the patient's sickness if the health sector embraces this idea. The goal of the disease prediction is to offer forecasts for a wide range of often recurring illnesses that, if left untreated or occasionally overlooked, can progress to fatal conditions and cause significant problems for the patients.

**6.2 Future Scope**

As nowadays we can clearly witness the increase in use of computers and technology to consider a huge amount of data, computers are being used to perform various complex tasks with commendable accuracy rates. Machine learning (ML) is a collection of multiple techniques and algorithms which permit computers to execute such complex tasks in an simplified manner. It is also used in both academics which is for students or learners and also in industry to make accurate predictions and use these diverse sources of dataset and information. Till date we can say we have grown in the fields of big data, Machine learning, and data sciences etc and have been a part of one of those industries which were able to collect such data and the staff to transform their goods and services in a desired manner. The learning methods developed for these industries and researches offer excellent potential to further improvise medical research and clinical care for the patients in the best possible manner. Machine learning uses mathematical algorithms and procedures which are used to describe the relationship between variables used in the model and the others. Our paper will explain the process of training the model and learning a suitable algorithm to predict the presence of a particular disease from the sample of the tissue based on its features. Though these algorithms work in different and unique manners depending on the way in which they are developed and used by the researchers. One way is to consider their supreme goals. The goal of our paper and statistical methods is to reach to a conclusion about the data which are collected from a wide variety of samples from our population. Though many techniques, like linear and logistic regression, are able to predict the diseases. For example, consider a case where, if we can create a model which described and understood the relationship between clinical variables and their transience then we can follow the organ transplant surgery i.e. we would need the factors and features which differentiate low mortality rate from high if we can develop such outcomes and reduce mortality rate to a desired rate in the near future also nothing can be said to be better than such situations.

**BIBILOGRAPHY**

[1] Liang H, Tsui BY, Ni H, et al. Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence. Nat Med. 2019;25(3):433- 438.

[2] Deo RC. Machine learning in medicine. Circulation. 2015;132(20):1920-1930.

[3] Rajendra Acharya U, Fujita H, Oh SL, et al. Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. Inf Sci (Ny). 2017;415-416:190-198. [4] Paniagua JA, Molina-Antonio JD, LopezMartinez F, et al. Heart disease prediction using random forests. J Med Syst. 2019;43(10):329.

[5] Poudel RP, Lamichhane S, Kumar A, et al. Predicting the risk of type 2 diabetes mellitus using data mining techniques. J Diabetes Res. 2018;2018:1686023.

[6] Al-Mallah MH, Aljizeeri A, Ahmed AM, et al. Prediction of diabetes mellitus type-II using machine learning techniques. Int J Med Inform. 2014;83(8):596-604.

[7] Tsanas A, Little MA, McSharry PE, Ramig LO. Nonlinear speech analysis algorithms mapped to a standard metric achieve clinically useful quantification of average Parkinson's disease symptom severity. J R Soc Interface. 2012;9(65):2756-2764.

[8] Arora S, Aggarwal P, Sivaswamy J. Automated diagnosis of Parkinson's disease using ensemble machine learning. IEEE Trans Inf Technol Biomed. 2017;21(1):289-299.

[9] Ahmad F, Hussain M, Khan MK, et al. Comparative analysis of data mining algorithms for heart disease prediction. J Med Syst. 2019;43(4):101.

[10] Parashar A, Gupta A, Gupta A. Machine learning techniques for diabetes prediction. Int J Emerg Technol Adv Eng. 2014;4(3):672-675.

[11] Breiman L. Random forests. Mach Learn. 2001;45(1):5-32.

[12] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer; 2009.

[13] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;12:2825-2830.

[14] McKinney W, van der Walt S, Lamoureux C, et al. Data structures for statistical computing in Python. In: Proceedings of the 9th Python in Science Conference; 2010.

[15] pickle — Python object serialization. Python documentation. https://docs.python.org/3/library/pickle.html. Accessed May 26, 2023.

[16] Huang ML, Hung CC, Hsu CY, et al. Predicting ischemic stroke using the Framingham Stroke Risk Score and a simple decision rule in Chinese patients with type 2 diabetes. Diabetes Care. 2010;33(2):427-429.

[17] Breiman L, Friedman JH, Olshen RA, Stone CJ. Classification and Regression Trees. Wadsworth and Brooks; 1984