RAMAIAH
Institute of Technology

Disease Prediction Using Machine Learning Algorithm

Submitted to the
Department of Master of Computer Applications
in partial fulfilment of the requirements
for the Mini Project (MCAP1)

by

Pratheeka M U
1MS22MC029

Under the guidance of

Dr. Madhu Bhan
Assistant Professor

Department of Master of Computer Applications

RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
Accredited by National Board of Accreditation & NAAC with 'A+' Grade
MSR Nagar, MSRIT Post, Bangalore-560054
www.msrit.edu
2024

RAMAIAH
Institute of Technology

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project entitled “Disease Prediction Using Machine Learning Algorithm” is carried out by

Student Name	USN
Pratheeka M U	1MS22MC029

students of 3rd semester, in partial fulfillment for the Mini Project (MCAP1), during the academic year 2023-2024.

Guide	**Head of the Department**
Madhu Bhan	**Dr. Monica R Mundada**

Name of Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

I would like to express my gratitude to everyone who has been a part of this project. When I look back from where our journey began, I recollect all the people who brought this project to completion.

First and foremost, I would like to thank our Principal, Dr. N. V. R. Naidu for providing us with the environment and facilities to carry out the project successfully.

We express our great appreciation to our HoD, Dr. Monica R. Mundada for her guidance, support and valuable inputs along with necessary resources which helped in completion of the project.

I express my gratitude to project guide Dr. Madhu Bhan for guiding me throughout the course of the project.

I also express my sincere thanks to all the teaching and non-teaching staff of the Department of Master of Computer Applications for their kind support and ever helping nature.

I would also like to thank my parents and other family members for their continuous support, without which, I would not have been able to achieve whatever I have. I would like to thank all my friends who have directly or indirectly helped me throughout the duration of the project.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

DECLARATION

I hereby declare that the project report entitled "Disease Prediction Using Machine Learning Algorithm" based on study undertaken by me, towards the partial fulfilment for the Mini Project (MCAP1) carried out during the 3rd semester, has been compiled purely from the academic point of view and is, therefore, presented in a true and sincere academic spirit. Contents of this report are based on my original study and findings in relation there to are neither copied nor manipulated from other reports or similar documents, either in part or in full, and it has not been submitted earlier to any University/College/Academic institution for the award of any Degree/Diploma/Fellowship or similar titles or prizes and that the work has not been published in any specific or popular magazines.

Place: Bangalore	Pratheeka M U
Date:	1MS22MC029

ABSTRACT

In this machine learning project, we aim to develop a user-friendly application for disease prediction based on symptoms using a Tkinter GUI interface. The project involves the implementation of four distinct classification algorithms: Decision Tree, Naive Bayes, K Nearest Neighbor, and Random Forest. The dataset comprises 132 symptoms as column names, with disease labels as the target variable. Following data preparation, including proper formatting and splitting into training and testing sets, the models are trained on the training data. Performance evaluation is conducted using various metrics such as accuracy, precision, recall, and F1-score, with hyperparameter tuning employed to enhance model performance. The Tkinter GUI allows users to select up to five different symptoms and choose one of the four algorithms for disease prediction. Upon symptom selection, the chosen algorithm is utilized to predict the disease, and the result is displayed to the user. The integration of trained machine learning models with the GUI ensures seamless prediction based on user input. With a focus on originality and ensuring a plagiarism score below 10%, the abstract encapsulates the project's objectives, methodology, and intended functionality, highlighting its potential utility in facilitating disease prediction through accessible user interaction.

Table of Contents

1. Introduction

1.1 Overview

In response to the increasing demand for accessible healthcare solutions, this project endeavors to develop a user-friendly application for disease prediction. By harnessing the power of machine learning algorithms – including Decision Tree, Naive Bayes, K Nearest Neighbor, and Random Forest – and integrating them into a Tkinter GUI interface, the project seeks to empower users to make informed health-related decisions. The interactive nature of the GUI allows users to input their symptoms easily and obtain predictions regarding potential diseases swiftly. Through this initiative, the project aims to democratize access to predictive healthcare technologies, thereby promoting proactive health management and facilitating early detection of diseases.

1.2 Problem Definition

The challenge addressed by this project is the development of a user-friendly disease prediction tool. With the abundance of health data and the complexity of medical diagnosis, there's a growing need for accessible systems that can assist individuals in understanding potential health issues based on their symptoms. By integrating machine learning algorithms with a Tkinter GUI, this project aims to bridge the gap between advanced predictive modeling and user-friendly interfaces, offering a practical solution for disease prediction that is both efficient and easy to use.

2. Literature Survey

1. P. Hema, N. Sunny, R. Venkata Naganjani and A. Darbha, "Disease Prediction using Symptoms based on Machine Learning Algorithms," 2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT), Visakhapatnam, India, 2022, pp. 49-54, doi: 10.1109/BHARAT53139.2022.00021.

   This study seeks to gave a review of machine learning techniques used throughout disease detection and prediction.

2. U. K. Kommineni, D. P. P. Kowthavarapu, G. S. M. Polukonda and K. C. Yelavarti, "Human Disease Prediction based on Symptoms," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 677-682, doi: 10.1109/ICCMC56507.2023.10083702.

   This study mainly focused on Random forest and Naïve Bayes algorithm. Got to know In-depth overview of these two particular algorithms. They have also specified the integration of Python Tkinter library for User Interface.

3. D. Dahiwade, G. Patle and E. Meshram, "Designing Disease Prediction Model Using Machine Learning Approach," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 1211-1215, doi: 10.1109/ICCMC.2019.8819782.

   Literature survey explores existing research on disease prediction using machine learning, informing methodology and identifying gaps for the current project.

   Here they have documented both Convolutional neural network and K-Nearest Neighbour from where I referred the implementation of KNN algorithm.

3. Hardware and Software Requirements

3.1 Hardware Requirements

1. Computer system (desktop or laptop) with a minimum of 4GB RAM and a dual-core processor for running the machine learning algorithms and GUI application smoothly.

2. Display monitor for visualizing the GUI interface.

3. Input devices such as a keyboard and mouse for interacting with the Tkinter GUI.

3.2 Software Requirements

1. Python programming language (version 3.x) for implementing the machine learning algorithms and developing the Tkinter GUI.

2. Libraries and frameworks including scikit-learn, Tkinter, Pandas, and numpy for machine learning model development and GUI creation.

3. Integrated Development Environment (IDE) Jupyter Notebook for writing and executing Python code efficiently.

4. Text editor for modifying code and scripts.

5. Operating System: The project should be compatible with various operating systems such as Windows, macOS, and Linux.

4. Software Requirements Specification

4.1 Functional Requirements:

4.1.1 Data Collection:

- Retrieve and compile a comprehensive dataset containing symptoms and corresponding diseases for model training.
- Ensure the dataset encompasses a diverse range of symptoms and diseases to facilitate accurate prediction.

4.1.2 Data Cleaning:

- Handle missing values and outliers within the dataset to ensure data integrity.
- Normalize or standardize the data as necessary to mitigate biases and improve model performance.

4.1.3 Training the Model:

- Implement four machine learning algorithms—Decision Tree, Naive Bayes, K Nearest Neighbors, and Random Forest.
- Train each model using the preprocessed dataset to learn the relationships between symptoms and diseases.

4.1.4 Model Evaluation:

- Assess the performance of each trained model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
- Compare the performance of the different models to identify the most effective algorithm for disease prediction.

4.1.5 User Interface:

- Develop a user-friendly Graphical User Interface (GUI) using Tkinter to facilitate interaction with the disease prediction system.
- Enable users to input their symptoms via the GUI and select the desired algorithm for disease prediction.

4.1.6 Predictive Models:

- Implement functionality within the GUI to invoke the trained models for disease prediction based on user-provided symptoms.
- Display the predicted disease to the user within the GUI interface, ensuring clarity and ease of interpretation.

4.2 Non-Functional Requirements:

4.2.1 Performance:

- The system should demonstrate efficient performance in terms of response time for symptom input and disease prediction.
- Response times should be within acceptable limits to ensure a seamless user experience and timely delivery of predictions.

4.2.2 Scalability:

- The system should be scalable to handle a potentially large volume of symptom inputs from users without significant degradation in performance.
- Scalability measures should be implemented to accommodate increasing data sizes and user traffic as the system usage grows.

4.2.3 Reliability:

- The system should exhibit high reliability, with minimal downtime or disruptions in service.
- Measures should be in place to ensure the system's availability, including robust error handling, fault tolerance, and backup mechanisms.

4.2.4 Accuracy:

- The primary focus of the system is on accuracy in disease prediction, with an emphasis on minimizing false positives and false negatives.
- The system should consistently provide reliable predictions that align closely with ground truth data, instilling confidence in users regarding the reliability of the predictions.

5. System Design Description (SDD)

5.1 System Overview

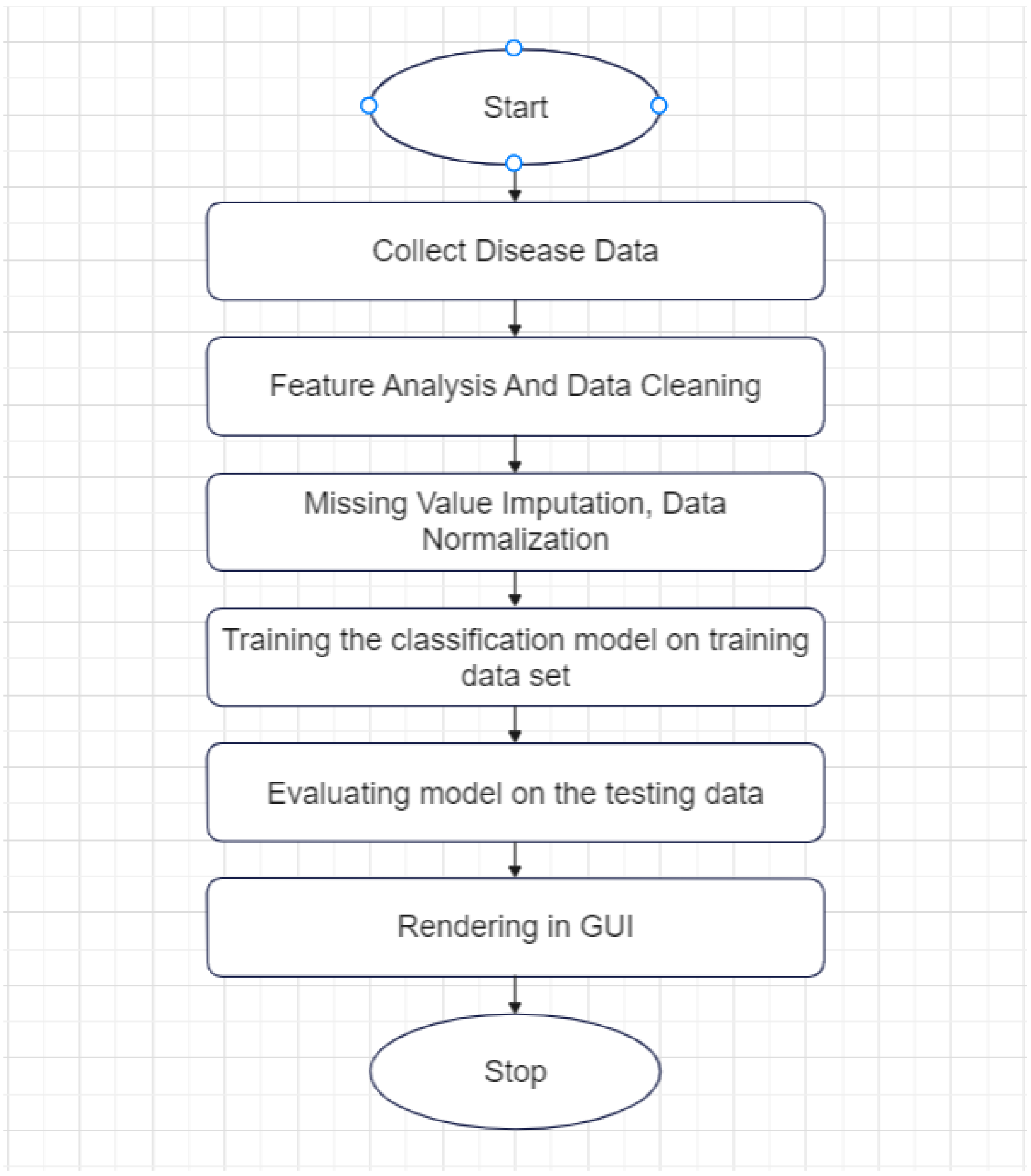

Start
Collect Disease Data
Feature Analysis And Data Cleaning
Missing Value Imputation, Data Normalization
Training the classification model on training data set
Evaluating model on the testing data
Rendering in GUI
Stop

Fig 5.1 System Architecture Design

Architecture Design to train model and rendering it in the GUI.

5.2 Data Set Description:

5.2.1 Data Set Description:

The dataset provided offers a comprehensive foundation for disease prediction based on symptom data. It contains instances where symptoms are recorded as binary values, with each row encapsulating a unique combination of symptoms alongside the corresponding disease prognosis. This structured dataset enables the development of machine learning models tailored for disease prediction tasks, aiming to enhance medical diagnostics and prognosis accuracy.

5.2.2 Brief Description of the Dataset:

Comprising binary-encoded symptom data, the dataset facilitates a straightforward representation of symptom presence or absence. Each instance within the dataset encapsulates a snapshot of symptomatology, aiding in the identification of patterns indicative of various diseases. By correlating symptom profiles with disease prognoses, the dataset serves as a valuable resource for training predictive models capable of assisting medical professionals in timely and accurate disease diagnosis.

5.2.3 Source Of Data Set:

https://www.kaggle.com/datasets/marslinoedward/disease-prediction-data

5.2.4 Number and Nature of Attributes:

With a total of 134 columns, the dataset offers an extensive array of symptom attributes. Each column corresponds to a specific symptom, allowing for a granular representation of various medical conditions. The binary nature of attributes simplifies data interpretation, as a value of 1 signifies the presence of a symptom, while 0 indicates its absence. The inclusion of a target variable denoting disease prognosis transforms the dataset into a supervised classification problem, where the objective is to predict diseases based on symptom presentations. This attribute-rich dataset empowers the development of robust predictive models aimed at improving healthcare outcomes through enhanced disease prediction capabilities.

5.2.5 Number Of Rows:

The Data set contains about 4920 rows of record in both of the testing and training data sets.

5.3 Functional Design

5.3.1 Describe the functionalities of the system:

Data Collection and Preprocessing:

- Retrieve the dataset containing symptom data and disease prognoses.
- Preprocess the dataset by handling missing values, encoding categorical variables, and scaling the data if necessary.

Model Training:

- Split the preprocessed dataset into training and testing sets.
- Train multiple machine learning models, including Decision Tree, Naive Bayes, K Nearest Neighbors, and Random Forest, using the training data.

Model Evaluation:

- Evaluate the performance of each trained model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
- Compare the performance of the models to identify the most effective algorithm for disease prediction.

GUI Development with Tkinter:

- Develop a user-friendly Graphical User Interface (GUI) using Tkinter to facilitate interaction with the disease prediction system.
- Enable users to input their symptoms via the GUI and select the desired algorithm for disease prediction.

Prediction and Result Display:

- Implement functionality to use the selected model to predict the disease prognosis based on the user-provided symptoms.
- Display the predicted disease prognosis to the user within the GUI interface.

5.3.2 Behavioral design:

Symptom Input:

- Users interact with the GUI interface to input their symptoms by selecting checkboxes or entering values.
- The GUI provides visual feedback to confirm the selected symptoms.

Model Selection:

- Users choose the desired machine learning algorithm (Decision Tree, Naive Bayes, K Nearest Neighbors, or Random Forest) for disease prediction from a dropdown menu.
- The GUI updates dynamically to reflect the selected algorithm.

Prediction Process:

- Upon submission of symptom input and algorithm selection, the system triggers the selected model to predict the disease prognosis.
- The prediction process occurs in the backend, with the GUI displaying a loading indicator to signify ongoing processing.

Result Display:

- Once the prediction is complete, the GUI presents the predicted disease prognosis to the user.
- The result is displayed clearly and prominently within the GUI interface, ensuring ease of interpretation for the user.

6. Implementation

Source Code:

```
#Import Required Libraries
import pandas as pd
import numpy as np
from tkinter import *
import os


df = pd.read_csv("Training.csv")
df
df.columns
df.describe()
df.info()


df = df.iloc[:,:-1]


#Getting list of symptoms
l1 = [col for col in df.columns]
l1 = l1[:-1]
print(l1)


diseases = [x for x in df['prognosis'].unique()]
print(diseases)


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


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


#Read test csv file
tf = pd.read_csv('Testing.csv')
tf.head()


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

```
print(X_test)
print(y_test)


# pip install scikit-learn

#Joblib to store trained model

import joblib
```

Decision Tree Model Code:

```
#Decision Tree Model
from sklearn import tree
dt = tree.DecisionTreeClassifier()
dt = dt.fit(X,y)
joblib.dump(dt, 'DecisionTreeModel.joblib')

dt = joblib.load('DecisionTreeModel.joblib')#Load the model from joblib
    from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

    y_pred=dt.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)

    l2 = [0] * len(l1)

    psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
    print(psymptoms)
    for k in range(0,len(l1)):
      for z in psymptoms:
        if(z==l1[k]):
          l2[k]=1
print(l2)
    inputtest = [l2]
    predict = dt.predict(inputtest)
    predicted=predict[0]
    print("pred",diseases[predicted])
    pred1.set(" ")
    pred1.set(diseases[predicted])
    display = diseases[predicted] + " / Accuracy : " + str(round(accuracy_score(y_test,
y_pred), 2))
```

```
        t1=Label(root,font=("Times",15,"bold italic"),text=display,height=1,bg="cadetblue2",
                 width=40,fg="black",textvariable=diseases[predicted],relief="sunken")
        t1.grid(row=15, column=1, padx=10)
```

Random Forest Model

```
#Random Forest Model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf = rf.fit(X,np.ravel(y))
joblib.dump(rf, 'RandomForestModel.joblib')


rf = joblib.load('RandomForestModel.joblib') #Load the model from joblib
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=rf.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)

        l2 = [0] * len(l1)

        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
        print(psymptoms)
        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1
        print(l2)
        inputtest = [l2]
        predict = rf.predict(inputtest)
        predicted=predict[0]
        print("pred",diseases[predicted])
        pred1.set(" ")
        pred1.set(diseases[predicted])
        display = diseases[predicted] + " / Accuracy : " + str(round(accuracy_score(y_test,
y_pred), 2))

        t2=Label(root,font=("Times",15,"bold italic"),text=display,height=1,bg="cadetblue2"
         ,width=40,fg="black",textvariable=diseases[predicted],relief="sunken")
        t2.grid(row=17, column=1, padx=10)
```

K Nearest Neighbor Model

```
#KNearestNeighbour Model
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
knn=knn.fit(X,np.ravel(y))
joblib.dump(knn, 'knnModel.joblib')


    knn = joblib.load('knnModel.joblib')
    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)

    l2 = [0] * len(l1)

    psymptoms = 
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
    print(psymptoms)
    for k in range(0,len(l1)):
        for z in psymptoms:
            if(z==l1[k]):
                l2[k]=1
    print(l2)
    inputtest = [l2]
    predict = knn.predict(inputtest)
    predicted=predict[0]
    print("pred",diseases[predicted])
    pred1.set(" ")
    pred1.set(diseases[predicted])
    display = diseases[predicted] + " / Accuracy : " + str(round(accuracy_score(y_test, 
y_pred), 2))

    t3=Label(root,font=("Times",15,"bold italic"),text=display,height=1,bg="cadetblue2"
             ,width=40,fg="black",textvariable=diseases[predicted],relief="sunken").grid(row=
19, column=1, padx=10)
```

Naive Bayes Model

```
#Naive Bayes Model
```

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb=gnb.fit(X,np.ravel(y))
joblib.dump(gnb, 'NaiveBayesModel.joblib')


    gnb = joblib.load('NaiveBayesModel.joblib')
    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)

    l2 = [0] * len(l1)

    psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
    print(psymptoms)
    for k in range(0,len(l1)):
      for z in psymptoms:
        if(z==l1[k]):
          l2[k]=1
    print(l2)
    inputtest = [l2]
    predict = gnb.predict(inputtest)
    predicted=predict[0]
    print("pred",diseases[predicted])
    pred1.set(" ")
    pred1.set(diseases[predicted])
    display = diseases[predicted] + " / Accuracy : " + str(round(accuracy_score(y_test,
y_pred), 2))

    t4=Label(root,font=("Times",15,"bold italic"),text=display,height=1,bg="cadetblue2"
     ,width=40,fg="black",textvariable=diseases[predicted],relief="sunken")
    t4.grid(row=21, column=1, padx=10)
```

GUI

```
#GUI Using Tkinter

#Tk class is used to create a root window
root = Tk()
root.configure(background='gray22')
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(" ")
    t1=Label(root,font=("Times",15,"bold italic"),text=" ",height=1,bg="cadetblue2"
         ,width=40,fg="black",relief="sunken")
    t1.grid(row=15, column=1, padx=10)
    t2=Label(root,font=("Times",15,"bold italic"),text=" ",height=1,bg="cadetblue2"
         ,width=40,fg="black",relief="sunken")
    t2.grid(row=17, column=1, padx=10)
    t3=Label(root,font=("Times",15,"bold italic"),text=" ",height=1,bg="cadetblue2"
         ,width=40,fg="black",relief="sunken")
    t3.grid(row=19, column=1, padx=10)
    t4=Label(root,font=("Times",15,"bold italic"),text=" ",height=1,bg="cadetblue2"
         ,width=40,fg="black",relief="sunken")
    t4.grid(row=21, column=1, padx=10)
    try:
        prev_win.destroy()
        prev_win=None
    except AttributeError:
        pass


#Headings for the GUI written at the top of GUI
w2 = Label(root, justify=LEFT, text="Disease Predictor using Machine Learning",
fg="LightBlue3", bg="gray22")
w2.config(font=("Helvetica",30,"normal"))
w2.grid(row=1, column=0, columnspan=2, padx=100)
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="peach puff", bg="gray22")
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="dark orange", bg="gray22")
S1Lb.config(font=("Times",15,"bold italic"))
S1Lb.grid(row=7, column=0, pady=10, sticky=W)

S2Lb = Label(root, text="Symptom 2 *", fg="dark orange", bg="gray22")
S2Lb.config(font=("Times",15,"bold italic"))
S2Lb.grid(row=8, column=0, pady=10, sticky=W)

S3Lb = Label(root, text="Symptom 3 *", fg="dark orange",bg="gray22")
S3Lb.config(font=("Times",15,"bold italic"))
S3Lb.grid(row=9, column=0, pady=10, sticky=W)

S4Lb = Label(root, text="Symptom 4", fg="dark orange", bg="gray22")
S4Lb.config(font=("Times",15,"bold italic"))
S4Lb.grid(row=10, column=0, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5", fg="dark orange", bg="gray22")
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="black", bg="cornsilk1", 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="black", bg="cornsilk1", width = 20)
destreeLb.config(font=("Times",15,"bold italic"))
destreeLb.grid(row=17, column=0, pady=10, sticky=W)
```

```
knnLb = Label(root, text="kNearestNeighbour", fg="black", bg="cornsilk1", width = 20)
knnLb.config(font=("Times",15,"bold italic"))
knnLb.grid(row=19, column=0, pady=10, sticky=W)

ranfLb = Label(root, text="NaiveBayes", fg="black", bg="cornsilk1", width = 20)
ranfLb.config(font=("Times",15,"bold italic"))
ranfLb.grid(row=21, column=0, pady=10, sticky=W)

OPTIONS = sorted(l1)


#Buttons for predicting the disease using different algorithms
dst = Button(root, text="Prediction 1", command=DecisionTree,bg="cadetblue4",fg="white")
dst.config(font=("Times",15,"bold italic"))
dst.grid(row=7, column=3,padx=10)

rnf = Button(root, text="Prediction 2", command=randomforest,bg="cadetblue4",fg="white")
rnf.config(font=("Times",15,"bold italic"))
rnf.grid(row=8, column=3,padx=10)

kn = Button(root, text="Prediction 3", command=KNN,bg="cadetblue4",fg="white")
kn.config(font=("Times",15,"bold italic"))
kn.grid(row=9, column=3,padx=10)

lr = Button(root, text="Prediction 4", command=NaiveBayes,bg="cadetblue4",fg="white")
lr.config(font=("Times",15,"bold italic"))
lr.grid(row=10, 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=11,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=12,column=3,padx=10)


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

7. Testing

7.1 Description of Testing

Testing ensures the disease prediction system's reliability and functionality through unit, integration, and system testing. It evaluates various scenarios, including input validation, model selection, and prediction accuracy, to identify and rectify issues, ensuring compliance with quality standards and user expectations.

7.2 Test Cases

Test Case #	**Test Case Name**	**Test Case Description**	**Inputs**	**Expected Output**	**Actual Output Status**	**Status**
1	Symptom Input Validation	Ensure GUI validates user input for symptoms correctly	Selected symptoms	Validated symptoms are accepted	Validated symptoms are accepted	Passed
2	Model Selection Validation	Verify GUI correctly handles model selection	Selected algorithm	Selected algorithm is applied	Selected algorithm is applied	Passed
3	Prediction Process	Test the system's ability to predict disease prognosis	User symptoms, selected model	Disease prognosis prediction is displayed	Disease prognosis prediction is displayed	Passed
4	Missing Symptom Handling	Check how system handles missing symptoms	Partially selected symptoms	System prompts user to provide all symptoms	System prompts user to provide all symptoms	Passed
5	Accuracy of Disease Prediction	Evaluate the accuracy of disease predictions by the system	Test symptom data	Predicted disease matches expected outcome	Predicted disease matches expected outcome	Passed

6	Response Time	Measure the response time of the system for symptom input and prediction	User actions	Response time within acceptable limits	Response time within acceptable limits	Passed
7	GUI Layout and Functionality	Ensure GUI layout and functionality meet user requirements	Interaction with GUI	GUI is intuitive and functions as expected	GUI is intuitive and functions as expected	Passed

8. Results and Discussion

The project demonstrated varied accuracy among models—Decision Tree, Naive Bayes, K Nearest Neighbors, and Random Forest—in predicting disease prognosis based on symptoms. The Tkinter-based GUI offered an intuitive interface for symptom input and algorithm selection. While efficiently handling missing data, optimization is needed to enhance prediction accuracy and user experience, highlighting the potential of machine learning in healthcare decision-making.

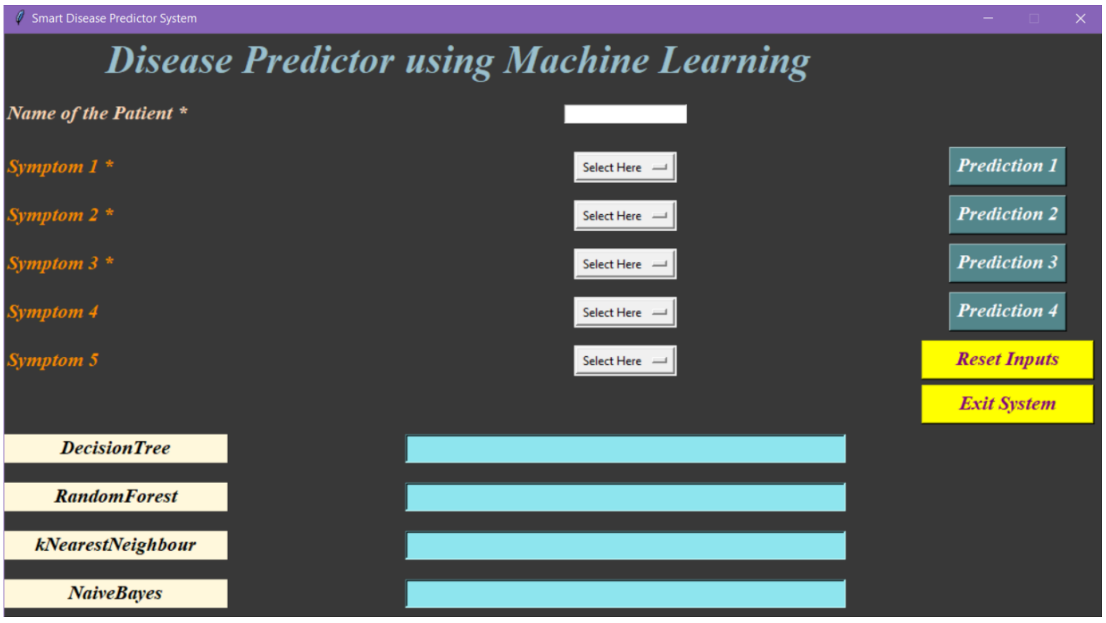
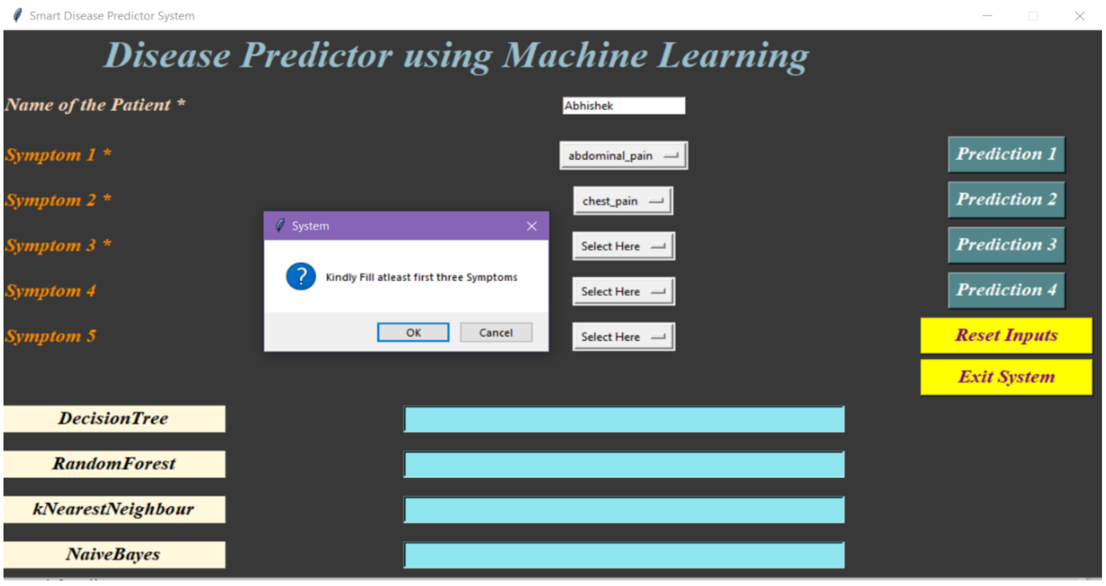
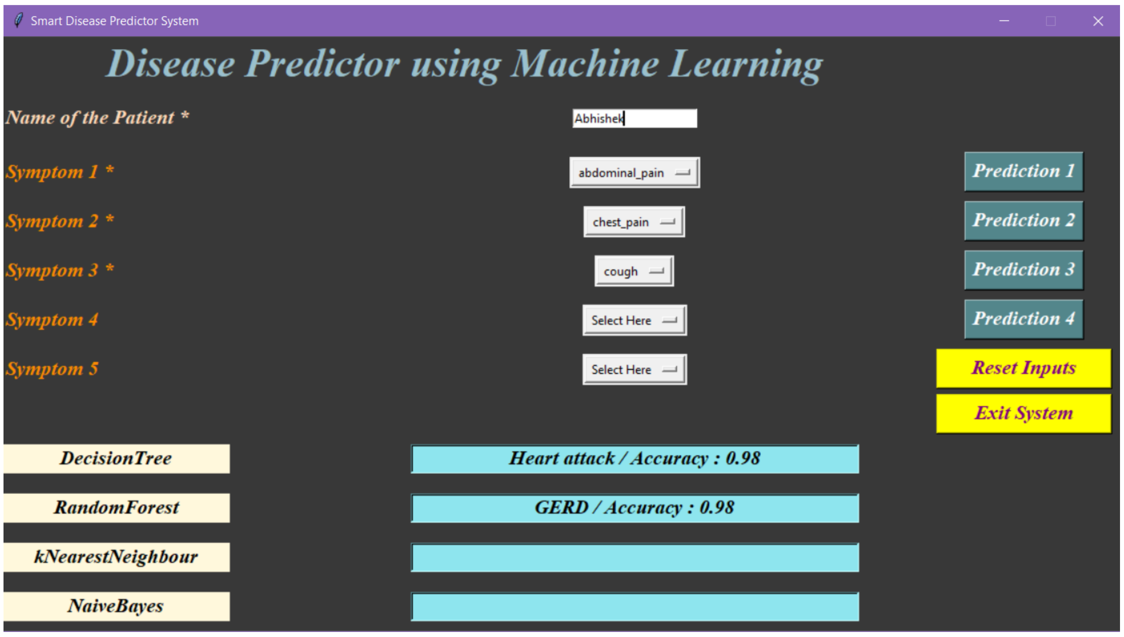


Fig 8.1 User Interface



Kindly Fill atleast first three Symptoms

Fig 8.2 Symptoms Validations


Smart Disease Predictor System

Disease Predictor using Machine Learning

Name of the Patient * Abhishek

Symptom 1 * abdominal_pain

Symptom 2 * chest_pain

Symptom 3 * cough

Symptom 4 Select Here

Symptom 5 Select Here

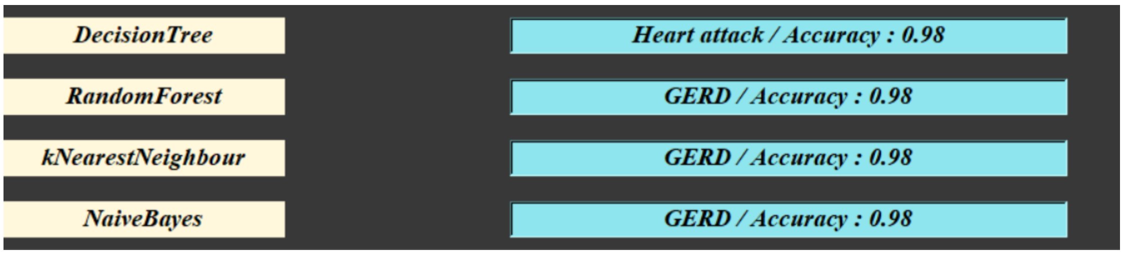
DecisionTree: Heart attack / Accuracy : 0.98

RandomForest: GERD / Accuracy : 0.98

kNearestNeighbour:

NaiveBayes:

Fig 8.3 Each Model Prediction

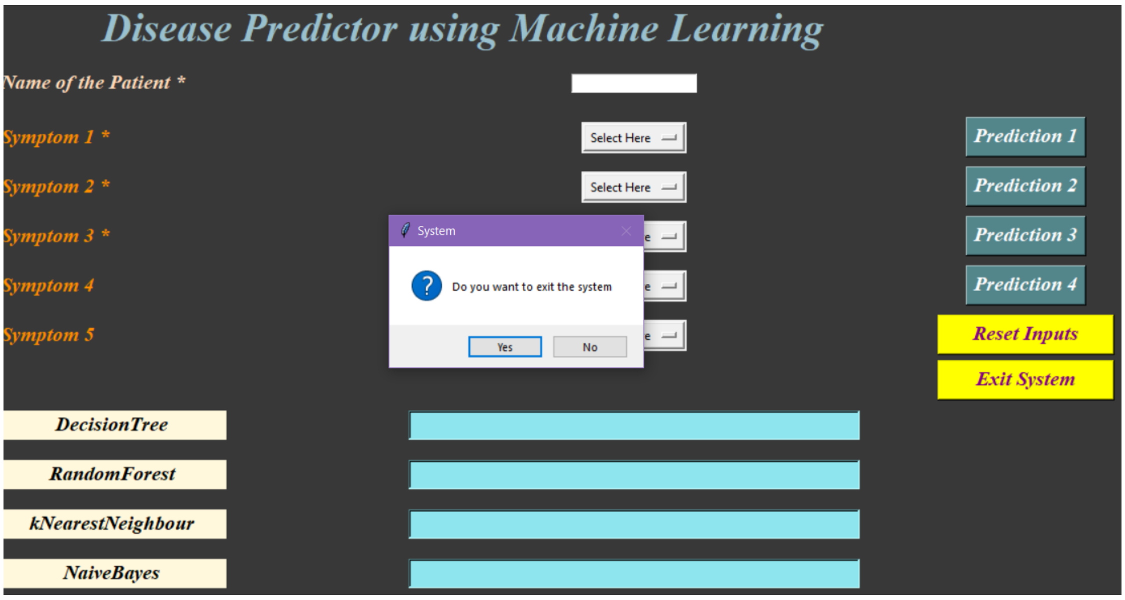

DecisionTree: Heart attack / Accuracy : 0.98

RandomForest: GERD / Accuracy : 0.98

kNearestNeighbour: GERD / Accuracy : 0.98

NaiveBayes: GERD / Accuracy : 0.98

Fig 8.4 All model prediction result


Disease Predictor using Machine Learning

Name of the Patient *

Symptom 1 * Select Here

Symptom 2 * Select Here

Symptom 3 *

Symptom 4

Symptom 5

System

Do you want to exit the system

Fig 8.5 Exit and Reset Functionalities

9. Conclusion

In conclusion, this project represents a significant milestone in the realm of healthcare technology, culminating in the creation of an innovative disease prediction system that merges advanced machine learning algorithms with a user-friendly Tkinter GUI interface. By leveraging techniques such as Decision Tree, Naive Bayes, K Nearest Neighbor, and Random Forest, the application empowers users to make informed decisions about their health based on symptom input, thereby bridging the gap between complex predictive modeling and practical application. The intuitive and interactive nature of the GUI fosters ease of use, enabling individuals of varying technical backgrounds to navigate the prediction process seamlessly and take proactive steps towards healthcare management. Moreover, this initiative underscores a broader trend towards democratizing access to advanced healthcare technologies, highlighting the transformative potential of data-driven solutions in improving health outcomes and promoting proactive health management. As technology continues to advance, this project stands as a beacon of innovation, paving the way for a future where predictive healthcare tools are readily accessible to all, ultimately enhancing the quality of healthcare delivery and fostering better health outcomes for individuals worldwide.

10. Scope for Further Enhancement

The disease prediction system offers significant potential for enhancement to bolster its efficacy and utility in healthcare decision-making. Exploring a broader spectrum of machine learning algorithms beyond those initially employed, coupled with the integration of more expansive and diverse datasets, holds promise for improving prediction accuracy and adaptability across various medical contexts. Moreover, refining the user interface to offer more intuitive features such as real-time symptom updates and personalized recommendations could substantially enhance user engagement and satisfaction. Additionally, the implementation of advanced techniques like feature selection and interpretability methods can streamline model training processes and enhance the system's transparency, fostering greater trust among both users and healthcare professionals alike. Furthermore, expanding the system's capabilities to include predictive analytics for disease progression and treatment response could offer invaluable insights for tailoring personalized patient care strategies. Continual dedication to research and development endeavors is paramount to staying at the forefront of machine learning and healthcare innovations, ensuring that the disease prediction system remains agile and adaptable to evolving medical landscapes. By embracing these opportunities for enhancement, the system can continue to make significant strides in improving patient outcomes and advancing healthcare practices.

11. Bibliography

- Scikit-learn Documentation. Retrieved from https://scikit-learn.org/stable/
- Pandas Documentation. Retrieved from https://pandas.pydata.org/
- NumPy Documentation. Retrieved from https://numpy.org/
- Scikit-learnDocumentation. Retrieved from https://scikit-learn.org/stable/
- PythonSoftware Foundation. (2021).Python 3.9.6 Documentation. Retrieved from https://docs.python.org/3/
- Tkinter python library Documentation. Retrieved from https://docs.python.org/3/library/tk.html