

DISEASE PREDICTION FROM SYMPTOMS

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

Adina Hanisha (38110012)
Ankita Kumari (38110035)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Adina Hanisha (38110012)** and **Ankita Kumari (38110035)** who carried out the project entitled “**Disease Prediction From Symptoms**” under my supervision from November 2021 to March 2022.

Internal Guide

Dr. B. Bharathi M.E., Ph.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Adina Hanisha (Reg No:38110012)** and **Ankita Kumari (Reg No: 38110035)** hereby declare that the Project Report entitled “**Disease Prediction From Symptoms**” done by us under the guidance of **Dr. B. Bharathi M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in 2018-2022.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E.,Ph.D., Dean**, School of Computing **Dr.S.Vigneshwari M.E., Ph.D.** and **Dr.L.Lakshmanan M.E., Ph.D.** , Heads of the Department of Computer Science and Engineering for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and a deep sense of gratitude to my Project Guide **Dr. B. Bharathi M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Health is the most important in every human's life. Weekly or monthly check up of one's health is most important for the prevention and also to stay healthy. Nowadays, the individual is not having that much time to go for health check-up. Recently, due to covid-19, no one are willing to go to hospital for health check-up due to the fear of spreading virus. In this situation, technology plays an important role. In this project, we have used Machine Learning. Machine Learning is the study of computer algorithms that improve automatically from the previous experience. It is widely used nowadays and it is the most efficient domain in health care. We will develop a GUI to get the symptoms from the user. The models used in this project are Naive Bayes and Decision Tree. The output is the disease, the accuracy of model, its definition and the treatment of the particular disease based on the symptoms given by the individual. As we all know the saying which tells that "Prevention of the disease at an early stage is much better than the cure which we take after we get affected by the disease". This project shows detailed explanation of how to find the diseases from symptoms, so that the individual can contact the respective doctor and stay healthy at an early stage.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1. OVERVIEW	2
	1.2. MACHINE LEARNING	3
	1.3 MACHINE LEARNING STRATEGIES	3
	1.3.1. SUPERVISED LEARNING	3
	1.3.2. UNSUPERVISED LEARNING	4
2	LITERATURE SURVEY	6
	2.1. RELATED WORK	6
3	METHODOLOGY	8
	3.1. EXISTING SYSTEM	8
	3.2. PROPOSED SYSTEM	8
	3.3. OBJECTIVE	8
	3.4. SOFTWARE AND HARDWARE REQUIREMENTS	9
	3.4.1. SOFTWARE REQUIREMENTS	9
	3.4.2. HARDWARE REQUIREMENTS	9
	3.4.3. LIBRARIES	9
	3.5. PROGRAMMING LANGUAGES	10
	3.5.1 PYTHON	10
	3.5.2. DOMAIN	10
	3.6. SYSTEM ARCHITECTURE	11
	3.7. ALGORITHMS USED	11

	3.7.1. NAIVE BAYES	11
	3.7.2. DECISION TREE	12
	3.8 MODULES	13
	3.8.1 DATASET COLLECTION	13
	3.8.2. TRAIN AND TEST THE MODELS	13
	3.8.3. DEPLOY THE MODELS	13
4	RESULTS AND DISCUSSION	14
	4.1. WORKING	14
5	CONCLUSION	15
	5.1. CONCLUSION	15
	REFERENCES	16
	APPENDICES	17
	A. SOURCE CODE	17
	B. SCREENSHOTS	46

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.1.	MACHINE LEARNING CLASSIFICATION	5
1.2.	MACHINE LEARNING TASK	5
3.1	SYSTEM ARCHITECTURE	11
3.2	DECISION TREE	12

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
ML	Machine Learning
NB	Naive Bayes
DT	Decision Tree
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

CHAPTER 1

INTRODUCTION

Health is the most important in every human's life. Weekly or monthly check up of one's health is most important for the prevention and also to stay healthy. Nowadays, the individual is not having that much time to go for health check-up. Recently, due to covid-19, no one are willing to go to hospital for health check-up due to the fear of spreading virus.

As we all know the saying which tells that "Prevention of the disease at an early stage is much better than the cure which we take after we get affected by the disease".

Healthcare is the most crucial parts of the human life. Nowadays, so many are not willing to go to hospital, due to work overload and negligence of their health. The doctors and nurses are putting up maximum efforts to save people's lives without even considering their own loves. There are also some villages which lack medical facilities.

We have designed a disease prediction system using ML algorithm (Naive Bayes), find the most accurate algorithm, and used it to find the disease and Tkinter for GUI. And also, we have created a chatbot using Decision Tree, which will help us in getting accurate predictions by taking into account the symptoms faced by an individual.

This project helps to get the idea about the disease of an individual based on the symptoms he/she have, and get the treatment easily by contacting the concern doctor.

A disease predictor can also be called as a virtual doctor, which can predict the disease based on symptoms. Recently due to covid-19 no one are willing to go outside. This disease predictor system can be a most useful as it identifies the disease without even contacting the individual.

1.1 OVERVIEW

A disease is a condition that affects the individual functioning of body totally. Diseases if neglected will lead to the death of an individual. Diseases can be identified by the symptoms of the body of an individual. Health is the most important in every human's life. Weekly or monthly check up of one's health is most important for the prevention and also to stay healthy.

Nowadays, the individual is not having that much time to go for health check-up. Recently, due to covid-19, no one are willing to go to hospital for health check-up due to the fear of spreading virus. As we all know the saying which tells that "Prevention of the disease at an early stage is much better than the cure which we take after we get affected by the disease".

Healthcare is the most crucial parts of the human life. Nowadays, so many are not willing to go to hospital, due to work overload and negligence of their health. The doctors and nurses are putting up maximum efforts to save people's lives without even considering their own loves. There are also some villages which lack medical facilities.

Accurate and on-time analysis of any health-related problem is important for the prevention and treatment of the illness. The traditional way of diagnosis may not be sufficient in the case of a serious ailment. In this situation, where everything has turned virtual, the doctors and nurses are putting up maximum efforts to save people's lives even if they have to danger their own.

There are also some remote villages which lack medical facilities. The dataset was processed in ML models Naive Bayes and Decision Tree. While processing the data, symptoms are given as input and the disease was received as an output. This project helps to get the idea about the disease of an individual based on the symptoms he/she have, and get the treatment easily by contacting the concern doctor.

1.2 MACHINE LEARNING

Machine learning could be a subfield of computer science (AI). The goal of machine learning typically is to know the structure information of knowledge of information and match that data into models which will be understood and used by folks. Although machine learning could be a field inside technology, it differs from ancient process approaches.

In ancient computing, algorithms are sets of expressly programmed directions employed by computers to calculate or downside solve. Machine learning algorithms instead give computers to coach on knowledge inputs and use applied math analysis so as to output values that fall inside a particular vary. thanks to this, machine learning facilitates computers in building models from sample knowledge so as to modify decision-making processes supported knowledge inputs.

1.3 Machine Learning Strategies

In machine learning, tasks square measure typically classified into broad classes. These classes square measure supported however learning is received or however feedback on the educational is given to the system developed. Two of the foremost wide adopted machine learning strategies square measure supervised learning that trains algorithms supported example input and output information that's tagged by humans, and unattended learning that provides the algorithmic program with no tagged information so as to permit it to search out structure at intervals its computer file.

1.3.1 *Supervised Learning*

In supervised learning, the pc is given example inputs that square measure labelled with their desired outputs. The aim of this technique is for the algorithmic program to be ready to “learn” by comparison its actual output with the “taught” outputs to search out errors, and modify the model consequently. Supervised learning thus uses patterns to predict label values on extra unlabelled information. For example, with supervised learning, an algorithm may be fed data with images of

sharks labelled as fish and images of oceans labelled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabelled shark images as fish and unlabelled ocean images as water.

A common use case of supervised learning is to use historical information to predict statistically probable future events. It's going to use historical stock exchange info to anticipate approaching fluctuations, or be used to filter spam emails. In supervised learning, labeled photos of dogs are often used as input file to classify unlabeled photos of dogs.

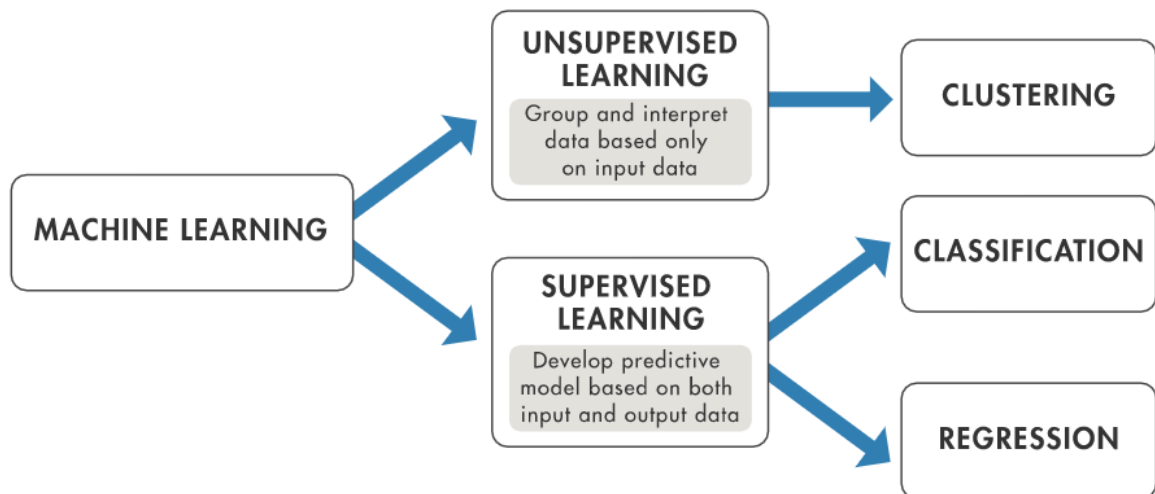
1.3.2 *Unattended Learning*

In unattended learning, information is unlabeled, that the learning rule is left to seek out commonalities among its input file. The goal of unattended learning is also as easy as discovering hidden patterns at intervals a dataset, however it should even have a goal of feature learning, that permits the procedure machine to mechanically discover the representations that square measure required to classify data.

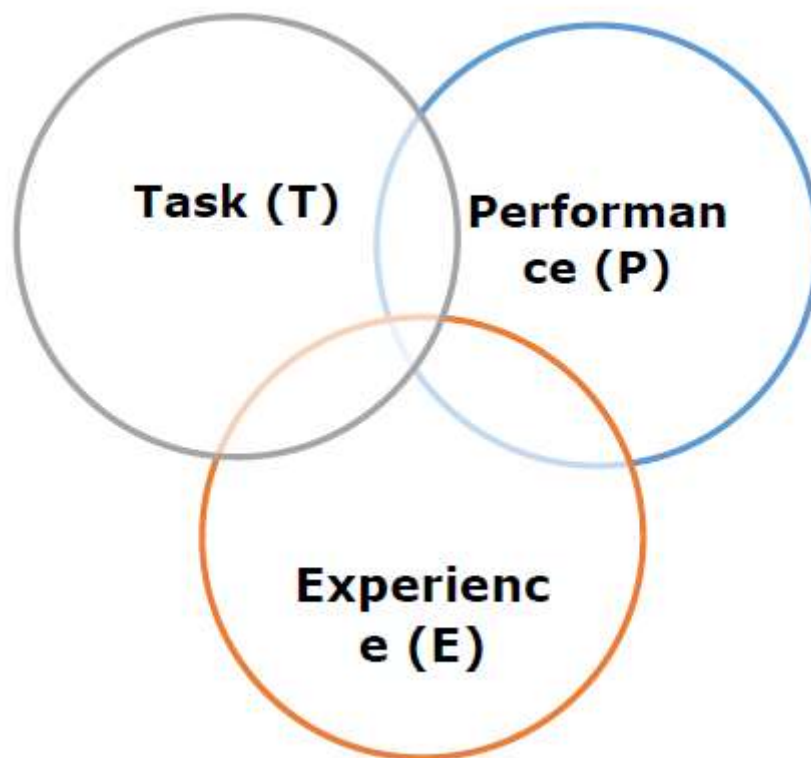
Unsupervised learning is usually used for transactional information. You will have an oversized dataset of consumers and their purchases, however as a person's you'll probably not be able to add up of what similar attributes will be drawn from client profiles and their styles of purchases.

With this information fed into Associate in Nursing unattended learning rule, it should be determined that ladies of a definite age vary UN agency obtain unscented soaps square measure probably to be pregnant, and so a promoting campaign

associated with physiological condition and baby will be merchandised.



1.1 MACHINE LEARNING CLASSIFICATION



1.2 MACHINE LEARNING TASK

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

- Tarigoppula V.S Sriram et al. in [1] collects the voice dataset from UCI Machine Learning repository and train four algorithms on that dataset. The result is the prediction of Parkinson Disease by considering the most accurate algorithm.
- Shubham Bind et al. in [2] studies about all the available researches in literature to predict the Parkinson diseases.
- K. Gomathi, D. Shanmuga Priya in [3] used different data mining techniques to predict Heart disease, Breast Cancer, Diabetes. The models used are Decision Tree and Naive Bayes Classifier. Performance of both the models was compared and the best classifier is used to predict the above diseases.
- Isha Pandya et al. in [4] used two supervised machine learning algorithms Decision Tree, accuracy 91% and Naïve Bayes classifier, accuracy 87%. Here, they used the combination of both to get the best accuracy. Naïve Bayes Classifier accuracy should be improved.
- Akash C. Jamgade, Prof. S. D. Zade in [5] paper determined the most danger diseases which occur in a person in a locality and community. But, the data collection is difficult.
- Siddhika Arunachalam in [6] six classification algorithms are used after analyzing 14 attributes in the dataset. But, we may get confused which algorithm to use.
- Ionela-Catalina ZAMFIR, Ana-Maria Mihaela IORDACHE methodologies in [7] used are Support Vector Machines, Artificial Neural Networks, K-Means Algorithm, Decision Trees, Logistic Regression and predicted diseases are breast cancer, lung cancer, heart diseases, diabetes, thyroid or kidney diseases.

- H BENJAMIN FREDRICK DAVID in [8] predicted the occurrence of heart disease using ensemble learning algorithms. But, the Research work can be made to produce an impact in the accuracy of the Decision Tree and Bayesian Classification.
- Harshit Anand et al. in [9], domains of Machine learning and Data Science are used and models are built using numpy, pandas, sklearn, and so on and the model are deployed using Django.
- Goutam Chakraborty et al. in [10], takes into account six features from 23 features in the dataset and predict the risk of chronic kidney disease. It is used to reduce the impact of Chronic Kidney disease (CKD), where creatinine test is not available for all.
- Durga Praveen et al. in [11], applied 5 models namely KNN, SVM, Random Forest, Naïve Bayes and Adaboost and found that KNN, Adaboost has the highest accuracy of all the models. So, any of these two are used for prediction and prevention of the liver disease.
- Sejin Park et al. in [12], early prediction of disease using the previous real-time stroke symptoms. It is implemented at a low cost. Random Forest algorithm is used for validating clinical significance.
- Ahan Chatterjee et al. in [13], they have used machine learning models like Decision tree, SVM, Random Forest, and so on, find the best classifier using the accuracy and use it to predict cancer disease risk in the early stage and prevent it. Simulation model is also designed to manage the patient flow in OPDs.
- Sergio Grueso et al. in [14], they have used dataset taken from ADNI database and selected 47 out of 159 studies for analysis. Deep learning combined with multimodal and multidimensional data is used to achieve the best performance.
- Upendra Kumar in [15], have used Computer-Aided Pre-Screening Tool (CAPST) which improves the accuracy of diagnosis in medicine. By this, fast and accurate prediction risk of disease is found.

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

From the above literature survey, We have inferred that all the systems existing predict only particular diseases namely lung disease, breast cancer, heart disease, diabetes by implementing various algorithms on the particular datasets.

After implementing various algorithms, the most accurate one is selected and it is used for prediction of disease. Sometimes, we may get confused of what algorithm to use. Also, all the systems find only the particular disease and not the disease based on the symptoms.

3.2 PROPOSED SYSTEM

We are proposing a system, which uses tkinter for GUI interface. It is a simple user Interface and also time efficient. Our aim with is to get the disease based on symptoms given by the user.

The domain we will use is the machine learning, in that we will be using Naïve Bayes Classifier, which will help us in getting the most accurate predictions easily and also the accuracy is given as output. To reduce time consuming, we will ask only less questions namely the name of the individual and the symptoms the individual is facing.

In this way, our system will be less time consuming and give accurate predictions. And also, we are creating a chatbot using Decision Tree, which will help us in getting accurate predictions by taking into account the symptoms faced by an individual.

3.3 OBJECTIVE OF PROJECT

Developing a project based on machine learning (ML) algorithms for prediction of any disease can help in a more accurate diagnosis than the conventional method is the main objective of the project. We have designed a disease prediction system using ML algorithm (Naive Bayes), and also chatbot using decision tree which is used it to find the disease based on the symptoms. Based on the symptoms of an individual, the ML model gives the output, i.e., the disease

that the individual might be suffering from. This project helps to get the idea about the disease of an individual based on the symptoms he/she have, and get the treatment easily by contacting the concern doctor.

3.4 SOFTWARE AND HARDWARE REQUIREMENTS

3.4.1 Software Requirements:

- ✓ Python
- ✓ Anaconda
- ✓ Jupyter Notebook

3.4.2 Hardware Requirements:

- ✓ Processor: Intel Core i5
- ✓ RAM: 8GB
- ✓ OS: Windows

3.4.3 Libraries:

- ✓ **Tkinter**- Tkinter is library of python used oftenly by everyone. It is a library which is useed to create GUI based applications easily. It contains so many widgets like radiobutton, textfiled and so on. We have used this for creating account registration screen, login or register screen, prediction interface which is a GUI based application
- ✓ **Sklearn**- Scikit Learn also known as sklearn is a open source library for python programming used for implementing machine learning algorithms. It features various classification, clustering, regression machine learning algorithms. In this it is used for importing machine learning models, get accuracy, get confusion matrix.
- ✓ **Pandas**- Library of python which can be used easily. It gives speed results and also easily understandable. It is a library which can be used without any cost. We have used it for data analysis and to read the dataset.
- ✓ **Matplotlib**- Library of python used for visualising the data using graphs, scatterplots and so on. Here, we have used it for data visualisation.

- ✓ **Numpy**- Library of python used for arrays computation. It has so many functions. We have used this module to change 2-dimensional array into contiguous flattened array by using ravel function.
- ✓ **Pandas Profiling**-This is library of python which can be used by anyone free of cost. It is used for data analysis. We have used this for getting the report of the dataset.

3.5. PROGRAMMING LANGUAGES

3.5.1 PYTHON

Python is that the best programing language fitted to Machine Learning. In step with studies and surveys, Python is that the fifth most significant language yet because the preferred language for machine learning and information science. It's owing to the subsequent strengths that Python has –

- ✓ **Easy to be told and perceive**- The syntax of Python is simpler; thence it's comparatively straightforward, even for beginners conjointly, to be told and perceive the language.
- ✓ **Multi-purpose language** – Python could be a multi-purpose programing language as a result of it supports structured programming, object-oriented programming yet as practical programming.
- ✓ **Support of open supply community** – As being open supply programing language, Python is supported by awfully giant developer community. Because of this, the bugs square measure simply mounted by the Python community. This characteristic makes Python terribly strong and adaptative.

3.5.2 DOMAIN

Machine learning could be a subfield of computer science (AI). The goal of machine learning typically is to know the structure information of knowledge of information and match that data into models which will be understood and used by folks. Although machine learning could be a field inside technology, it differs from ancient process approaches. In ancient computing, algorithms are sets of expressly programmed directions employed by computers to calculate or downside solve. Machine learning algorithms instead give computers to coach on knowledge inputs and use applied math analysis so as to output values that

fall inside a particular vary. Thanks to this, machine learning facilitates computers in building models from sample knowledge so as to modify decision-making processes supported knowledge inputs.

3.6. SYSTEM ARCHITECTURE



Fig 3.1 System Architecture

3.7 ALGORITHMS USED

3.7.1 NAIVE BAYES

It is a machine learning algorithm for classification problems which is based on Bayes theorem. The primary use of this is to do text classification. The Bayes theorem can be defined as:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

$P(C|X)$ is the probability of hypothesis C for the given data X . This is called the posterior probability.

$P(X|C)$ is the probability of data X given that the hypothesis C was true.

$P(C)$ is the probability of hypothesis C being true. This is called the prior probability of C .

$P(X)$ is the probability of the data and evidence of data and is called marginal probability.

Gaussian Naive Bayes

It follows the same procedure as the Naive Bayes. But for Naive Bayes we need a categorical dataset and for Gaussian Naive Bayes we need a dataset that has all the continuous features.

3.7.2 DECISION TREE

Decision Tree an algorithm whose input and output or known. The information is divided repeatedly using the particular parameter.

Decision tree consists of two main parts namely decision node and leaf nodes. The decision nodes specify the decision at which the parameter should be spilt.

The leaf nodes are the output bought by the decisions. A decision tree asks for either true or false to divide the data.

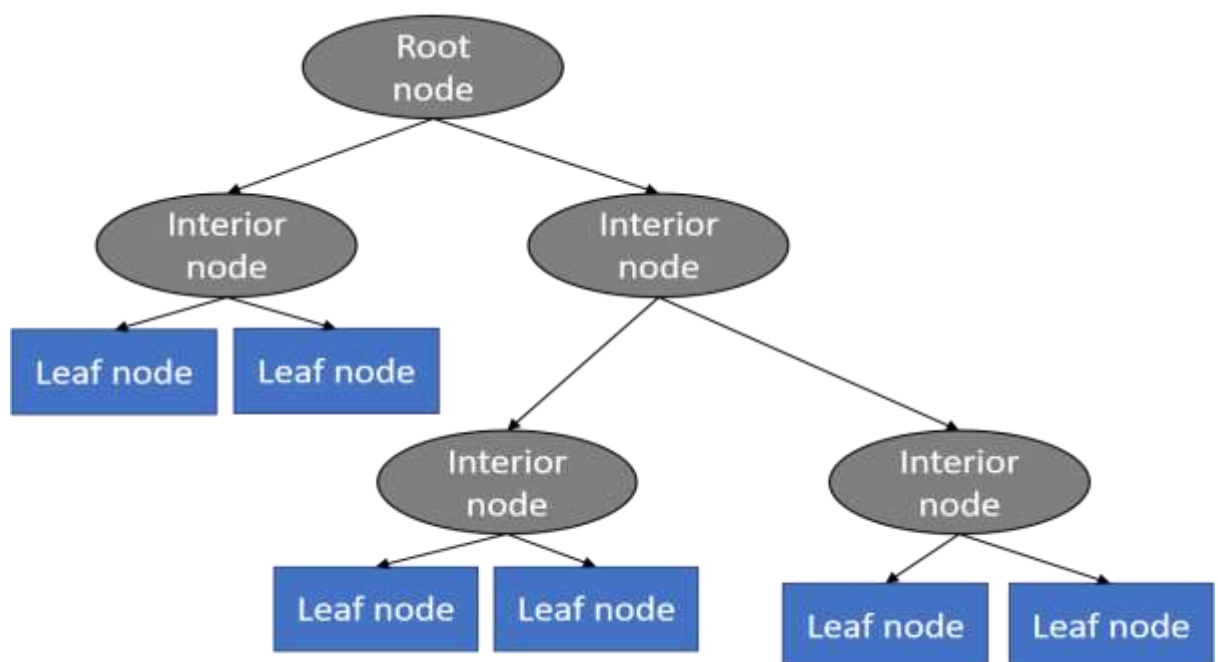


Fig 3.2. Decision Tree

3.8.MODULES

Project contains three parts:

3.8.1 DATASET COLLECTION.

3.8.2 TRAIN AND TEST THE MODEL.

3.8.3 DEPLOY THE MODELS.

- ❑ **Dataset Collection-** We had collected dataset from kaggle notebooks. The dataset contains the symptoms and the corresponding disease. It contains 4920 rows and 133 columns.
- ❑ **Train and Test the model-** We had used the Naïve Bayes Classifier as a model to train the dataset. After training, we had tested the model and found its accuracy.
- ❑ **Deploy the models-** Deployed Naïve bayes by creating interface to get the name, symptoms of an individual. By this, we will get the disease and accuracy of model as the output. We have also created a chatbot using Decision Tree which helps an individual to get the corresponding disease by checking whether he/she is being faced by the symptoms.

Following are the steps to do this project (use Jupyter Notebook):

1. Collect the dataset.
2. Import the necessary libraries.
3. Visualise the dataset.
4. Train the dataset using Naïve Bayes classifier and Decision Tree.
5. Test the model and find the accuracies of both.
6. Deploy the model- a) Naive Bayes as GUI Interface using Tkinter
b) Chatbot using Decision Tree
7. Predict the disease based on the symptoms given by the user.

CHAPTER 4

RESULTS AND DISCUSSION

Doctors and medical professionals are always required in case of an emergency. In the current situation of COVID-19, where essential resources are unavailable and people are also not willing to go outside in fear of spreading virus, our prediction system will be very helpful for finding the disease based on the symptoms in the early stage and get the correct diagnosis of a disease. This also helps in reduction of the cost and give the correct and fast result.

4.1. WORKING

In this way machine learning when implemented in healthcare can help in satisfying the individual and also take care of their particular disease easily. Naive Bayes, which is the most easy model helps to get the idea about the disease of an individual based on the symptoms he/she have, and get the treatment easily by contacting the concern doctor. We have also created chatbot using decision tree by which an individual can easily find the disease by chatting. Accuracy is defined as the ratio of sum of TP and TN to the sum of TP, TN, FP, FN.

Algorithm	Accuracy
Decision Tree	97%
Naive Bayes	100%

Table 4.1 Accuracies

CHAPTER 5

CONCLUSION

5.1. CONCLUSION

The project presented the technique of predicting the disease based on the symptoms of an individual patient. Once the disease is predicted, we could easily manage the medicine resources required for the treatment. Doctors and medical professionals are always required in case of an emergency.

In the current situation of COVID-19, where sufficient facilities and resources are unavailable, our prediction system can prove to be helpful and can be used in the diagnosis of a disease. Our prediction system will be very helpful for finding the disease based on the symptoms in the early stage and get the correct diagnosis of a disease. This also helps in reduction of the cost and give the correct and fast result.

The project presented the technique of predicting the disease based on the symptoms of an individual patient. Almost all the ML models gave good accuracy values but the most accurate one is selected and the disease given by it is considered as the disease of an individual. Once the disease is predicted, we could easily manage the medicine resources required for the treatment.

Doctors and medical professionals are always required in case of an emergency. In the current situation of COVID-19, where sufficient facilities and resources are unavailable, our prediction system can prove to be helpful and can be used in the diagnosis of a disease. This project would help in lowering the cost required in dealing with the disease and would also make the recovery process easy.

REFERENCES

1. Isha Pandya et al, "Prediction of Heart Disease Using Machine Learning Algorithms", 2018.
2. Akash C. Jamgade, Prof. S. D. Zade,"Disease Prediction using Machine Learning", International Research Journal of Engineering and Technology Volume: 06 Issue: 05 May 2019.
3. Siddhika Arunachalam," Cardiovascular Disease Prediction Model using Machine Learning Algorithms", International Journal for Research in Applied Science & Engineering Technology ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue VI June 2020.
4. Ionela-Catalina ZAMFIR, Ana-Maria Mihaela IORDACHE, "A REVIEW OF DATA MINING TECHNIQUES IN MEDICINE", JOURNAL OF INFORMATION SYSTEMS & OPERATIONS MANAGEMENT, Vol. 14.1, May 2020.
5. H BENJAMIN FREDRICK DAVID, "IMPACT OF ENSEMBLE LEARNING ALGORITHMS TOWARDS ACCURATE HEART DISEASE PREDICTION", DOI: 10.21917/ijsc.2020.0296.
6. Harshit Anand et al. "Hridaya Kalp: A Prototype for Second Generation Chronic Heart Disease Detection and Classification" ,31 July,2020.
7. Goutam Chakraborty et al. "Predicting the Risk of Chronic Kidney Disease using Machine Learning Algorithm", 11(1), 28 December, 2020 202.
8. A. Durga Praveen et al. "Intelligent Liver Disease Prediction system using Machine Learning Models" ,vol 702. Springer, Singapore. 5 Jan, 2021.
9. Sejin Park et al. "Machine-Learning-Based Elderly Stroke Monitoring System Using Electroencephalography Vital Signals", 2021, 11(4), 1761.
10. Ahan Chatterjee et al. "A Machine Learning Approach to prevent cancer", DOI: 10.4018/978-1-7998-2742-9.ch007, 2021.
11. Sergio Grueso et al. "Machine Learning methods for predicting conversion from mild cognitive impairment to Alzheimer's disease", 2021.
12. Upendra Kumar, "Applications of Machine Learning In Disease Pre-Screening", 10.4018/978-1-7998-7705- 9.ch049, 2021.

APPENDICES

A) SOURCE CODE

```
from tkinter import *  
  
import os  
  
import pandas as pd  
  
from mpl_toolkits.mplot3d import Axes3D  
  
from sklearn.preprocessing import StandardScaler  
  
import pandas_profiling  
  
import matplotlib.pyplot as plt  
  
df=pd.read_csv("Training.csv")  
  
gk=df.groupby('prognosis')  
  
pandas_profiling.ProfileReport(pd.read_csv('Training.csv'))
```

#READING THE TRAINING CSV FILE

```
df=pd.read_csv("Training.csv")  
  
print(df.shape)  
  
df.head(5)
```

#READING THE TESTING CSV FILE

```
tr=pd.read_csv("Testing.csv")  
  
print(tr.shape)  
  
tr.head(5)
```

#creation of function for prediction interface

```
def PredInterface():
```

```

import numpy as np

import pandas as pd

l1=['itching','skin_rash','nodal_skin_eruptions','continuous_sneezing','shivering','chills','joint_pain',

'stomach_pain','acidity','ulcers_on_tongue','muscle_wasting','vomiting','burning_micturition','spotting_ urination',

'fatigue','weight_gain','anxiety','cold_hands_and_feets','mood_swings','weight_loss','restlessness','lethargy',

'patches_in_throat','cough','high_fever','sunken_eyes','breathlessness','sweating','dehydration',

'indigestion','headache','yellowish_skin','dark_urine','nausea','loss_of_appetite','pain_behind_the_eyes',

'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','obesity','swollen_legs','puffy_face_and_eyes','enlarged_thyroid','brittle_nails','swollen_extremeties',

'excessive_hunger','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',

```

'passage_of_gases','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','muroid_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',

'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose',

'yellow_crust_ooze']

disease=['Fungal_infection','Allergy','GERD','Chronic_cholestasis','Drug_Reaction',

'Peptic_ulcer_disease','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_hemorrhoids(piles)',

'Heartattack','Varicoseveins','Hypothyroidism','Hyperthyroidism','Hypoglycemia','Osteoarthritis',

'Arthritis','(vertigo)_Paroxysmal_Positional_Vertigo','Acne','Urinary_tract_infection','Psoriasis',

```

'Impetigo']

l2=[]

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

    l2.append(0)

# TRAINING DATA df -----
-----

df=pd.read_csv("Training.csv")

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

'Peptic ulcer disease':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,'Osteoarthritis':34,'Arthritis':35,

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

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

print(df.head())

```

```

X= df[1]

y = df[["prognosis"]]

np.ravel(y)

print(y)

# TESTING DATA tr -----
tr=pd.read_csv("Testing.csv")

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic
cholestasis':3,'Drug Reaction':4,
'Peptic ulcer disease':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,'Osteoarthritis':34,'Arthritis':35,
'(vertigo) Paroymsal Positional Vertigo':36,'Acne':37,'Urinary tract
infection':38,'Psoriasis':39,
'Impetigo':40}},inplace=True)

X_test= tr[1]

y_test = tr[["prognosis"]]

np.ravel(y_test)

```

```
# -----
```

```
pred3=StringVar()
```

```
def NaiveBayes():
```

```
    from sklearn.naive_bayes import GaussianNB
```

```
    gnb = GaussianNB()
```

```
    gnb=gnb.fit(X,np.ravel(y))
```

```
    # calculating accuracy-----
```

```
    from sklearn.metrics import accuracy_score
```

```
    y_pred=gnb.predict(X_test)
```

```
    print(accuracy_score(y_test, y_pred))
```

```
    print(accuracy_score(y_test, y_pred,normalize=False))
```

```
    # -----
```

```
    psymptoms                                                                    =  
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get  
(  
0)  
]
```

```
    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'):
    t1.delete("1.0", END)
    t1.insert(END, disease[a])
else:
    t1.delete("1.0", END)
    t1.insert(END, "Not Found")

#function for printing accuracy score of NaiveBayes method
def score2():

    from sklearn.naive_bayes import GaussianNB

    gnb = GaussianNB()

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

    # calculating accuracy-----

    from sklearn.metrics import accuracy_score

    y_pred=gnb.predict(X_test)

    print(accuracy_score(y_test, y_pred))

    import random

    h='yes'

```

```
a=random.randint(0,23)
scr=accuracy_score(y_test, y_pred)*100-a
```

```
if (h=='yes'):
    t2.delete("1.0", END)
    t2.insert(END, scr)
else:
    t2.delete("1.0", END)
    t2.insert(END, "Not Found")
```

```
global root
root=Tk()

#from PIL import ImageTk, Image
#image=Image.open("image3.jpg")
#background_image=ImageTk.PhotoImage(image,master=root)
#background_label = Label(root, image=background_image)
#background_label.place(x=0, y=0, anchor=NW,relwidth=1, relheight=1)
```

```
#entry variables
Symptom1 = StringVar()
Symptom1.set(None)
Symptom2 = StringVar()
```

```

Symptom2.set(None)
Symptom3 = StringVar()
Symptom3.set(None)
Symptom4 = StringVar()
Symptom4.set(None)
Symptom5 = StringVar()
Symptom5.set(None)
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)

    pred3.set(" ")

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

    if qExit:

```

```

root.destroy()

exit()

# Heading

w2 = Label(root, justify=LEFT, text="DISEASE PREDICTION FROM
SYMPTOMS", fg="white", bg="black")

w2.config(font=("CALIBRI", 15))

w2.grid(row=1, column=0, columnspan=2, padx=100)

w2.grid(row=3, column=0, columnspan=2, padx=100)

w2 = Label(root, justify=LEFT, text="Fill atleast two symptoms", fg="black",
bg="white")

w2.config(font=("Tekton", 20))

w2.grid(row=5, column=0, columnspan=2, padx=100)

# labels

NameLb = Label(root, text="Name of the Patient", fg="black", bg="white")

NameLb.grid(row=6, column=0, pady=15, sticky=W)


S1Lb = Label(root, text="Symptom 1", fg="black", bg="yellow")

S1Lb.grid(row=7, column=0, pady=10, sticky=W)


S2Lb = Label(root, text="Symptom 2", fg="black", bg="yellow")

S2Lb.grid(row=8, column=0, pady=10, sticky=W)


S3Lb = Label(root, text="Symptom 3", fg="black", bg="yellow")

S3Lb.grid(row=9, column=0, pady=10, sticky=W)

```

```
S4Lb = Label(root, text="Symptom 4", fg="black", bg="yellow")
```

```
S4Lb.grid(row=10, column=0, pady=10, sticky=W)
```

```
S5Lb = Label(root, text="Symptom 5", fg="black", bg="yellow")
```

```
S5Lb.grid(row=11, column=0, pady=10, sticky=W)
```

```
ranfLb = Label(root, text="Naive Bayes", fg="black", bg="red")
```

```
ranfLb.grid(row=19, column=0, pady=10, sticky=W)
```

```
scrLb = Label(root, text="Accuracy Score", fg="black", bg="red")
```

```
scrLb.grid(row=21, column=0, pady=10, sticky=W)
```

```
# entries
```

```
list=['None']##for no selection
```

```
Options = list
```

```
OPTIONS = sorted(l1)
```

```
NameEn = Entry(root, textvariable=Name)
```

```
NameEn.grid(row=6, column=1)
```

```
S1En = OptionMenu(root, Symptom1,*Options,*OPTIONS)
```

```
S1En.grid(row=7, column=1)
```

```
S2En = OptionMenu(root, Symptom2,*Options,*OPTIONS)
```

```
S2En.grid(row=8, column=1)
```

```
S3En = OptionMenu(root, Symptom3,*Options,*OPTIONS)
```

```
S3En.grid(row=9, column=1)
```

```
S4En = OptionMenu(root, Symptom4,*Options,*OPTIONS)
```

```
S4En.grid(row=10, column=1)
```

```
S5En = OptionMenu(root, Symptom5,*Options,*OPTIONS)
```

```
S5En.grid(row=11, column=1)
```

```
lr = Button(root, text="Naive Bayes",  
command=lambda:[NaiveBayes(),score2()],bg="yellow",fg="black",width=15)
```

```
lr.grid(row=11, column=3,padx=10)
```

```
rs = Button(root,text="Reset Inputs",  
command=Reset,bg="yellow",fg="purple",width=15)
```

```
rs.grid(row=12,column=3,padx=10)
```

```
ex = Button(root,text="Exit System",  
command=Exit,bg="yellow",fg="purple",width=15)
```

```
ex.grid(row=13,column=3,padx=10)
```

```
#textfileds
```

```
t1 = Text(root, height=1, width=40,bg="orange",fg="black")
```

```
t1.grid(row=19, column=1 , padx=10)
```

```
t2 = Text(root, height=1, width=40,bg="orange",fg="black")
```

```
t2.grid(row=21, column=1, padx=10)
```

```
root.mainloop()
```

```
# DESIGNING WINDOW FOR REGISTRATION
```

```
def register():
```

```
    global register_screen
```

```
    register_screen = Toplevel(main_screen)
```

```
    register_screen.title("Register")
```

```
    register_screen.geometry("300x250")
```

```
    global username
```

```
    global password
```

```
    global username_entry
```

```
    global password_entry
```

```
    username = StringVar()
```

```
    password = StringVar()
```

```

Label(register_screen, text="Please enter details below", bg="black").pack()

Label(register_screen, text="").pack()

username_label = Label(register_screen, text="Username * ")
username_label.pack()

username_entry = Entry(register_screen, textvariable=username)
username_entry.pack()

password_label = Label(register_screen, text="Password * ")
password_label.pack()

password_entry = Entry(register_screen, textvariable=password, show='*')
password_entry.pack()

Label(register_screen, text="").pack()

Button(register_screen, text="Register", width=10, height=1, bg="white",
command = register_user).pack()

```

DESIGNING WINDOW FOR LOGIN

```

def login():

    global login_screen

    login_screen = Toplevel(main_screen)

    login_screen.title("Login")

    login_screen.geometry("300x250")

    Label(login_screen, text="Please enter details below to login").pack()

    Label(login_screen, text="").pack()


    global username_verify

```

```

global password_verify

username_verify = StringVar()
password_verify = StringVar()

global username_login_entry
global password_login_entry

Label(login_screen, text="Username * ").pack()
username_login_entry = Entry(login_screen, textvariable=username_verify)
username_login_entry.pack()
Label(login_screen, text="").pack()
Label(login_screen, text="Password * ").pack()
password_login_entry = Entry(login_screen, textvariable=password_verify,
show= '*')
password_login_entry.pack()
Label(login_screen, text="").pack()
Button(login_screen, text="Login", width=10, height=1, command =
login_verify).pack()

# IMPLEMENTING EVENT ON REGISTER BUTTON

def register_user():

    username_info = username.get()
    password_info = password.get()

```

```

file = open(username_info, "w")

file.write(username_info + "\n")

file.write(password_info)

file.close()


username_entry.delete(0, END)

password_entry.delete(0, END)


Label(register_screen, text="Registration Success", fg="green", font=("calibri",
11)).pack()


# IMPLEMENTING EVENT ON LOGIN BUTTON


def login_verify():

    username1 = username_verify.get()

    password1 = password_verify.get()

    username_login_entry.delete(0, END)

    password_login_entry.delete(0, END)


list_of_files = os.listdir()

if username1 in list_of_files:

    file1 = open(username1, "r")

    verify = file1.read().splitlines()

    if password1 in verify:

        login_sucess()

```

```

else:

    password_not_recognised()

else:

    user_not_found()

# DESIGNING POPUP FOR LOGIN SUCCESS

def login_sucess():

    global login_success_screen

    login_success_screen = Toplevel(login_screen)

    login_success_screen.title("Success")

    login_success_screen.geometry("150x100")

    Label(login_success_screen, text="Login Success").pack()

    Button(login_success_screen, text="OK", command=PredInterface).pack()

    Label(login_success_screen, text="Exit").pack()

    Button(login_success_screen, text="OK",
command=delete_login_success).pack()

# DESIGNING POPUP FOR LOGIN INVALID PASSWORD

def password_not_recognised():

    global password_not_recog_screen

    password_not_recog_screen = Toplevel(login_screen)

    password_not_recog_screen.title("Success")

    password_not_recog_screen.geometry("150x100")

```

```

Label(password_not_recog_screen, text="Invalid Password ").pack()

Button(password_not_recog_screen, text="OK",
command=delete_password_not_recognised).pack()

```

```

# DESIGNING POPUP FOR USER NOT FOUND

```

```

def user_not_found():

    global user_not_found_screen

    user_not_found_screen = Toplevel(login_screen)

    user_not_found_screen.title("Success")

    user_not_found_screen.geometry("150x100")

    Label(user_not_found_screen, text="User Not Found").pack()

    Button(user_not_found_screen, text="OK",
command=delete_user_not_found_screen).pack()

```

```

# DELETING POPUPS

```

```

def delete_login_success():

    login_success_screen.destroy()


def delete_password_not_recognised():

    password_not_recog_screen.destroy()


def delete_user_not_found_screen():

    user_not_found_screen.destroy()

```

```
# DESIGNING MAIN (FIRST) WINDOW
```

```
def main_account_screen():  
    global main_screen  
    main_screen = Tk()  
    main_screen.geometry("300x250")  
    main_screen.title("Account Login")  
    Label(text="Select Your Choice", bg="white", width="300", height="2",  
font=("Calibri", 13)).pack()  
    Label(text="").pack()  
    Button(text="Login", height="2", width="30", command = login).pack()  
    Label(text="").pack()  
    Button(text="Register", height="2", width="30", command=register).pack()  
    main_screen.mainloop()
```

```
main_account_screen()
```

```
gk
```

```
#Importing Libraries
```

```
import pandas as pd
```

```
import pyttsx3
```

```
from sklearn import preprocessing
```

```
from sklearn.tree import DecisionTreeClassifier,_tree
```

```
import numpy as np
```

```

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
import csv
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

training = pd.read_csv('Training.csv')
testing= pd.read_csv('Testing.csv')
cols= training.columns
cols= cols[:-1]
x = training[cols]
y = training['prognosis']
y1= y

reduced_data = training.groupby(training['prognosis']).max()
#mapping strings to numbers
le = preprocessing.LabelEncoder()
le.fit(y)
y = le.transform(y)

x_train,  x_test,  y_train,  y_test  =  train_test_split(x,  y,  test_size=0.33,
random_state=42)
testx=testing[cols]
testy=testing['prognosis']
testy=le.transform(testy)

```

```

clf1 = DecisionTreeClassifier()

clf = clf1.fit(x_train,y_train)

# print(clf.score(x_train,y_train))

# print ("cross result=====")

scores = cross_val_score(clf, x_test, y_test, cv=3)

# print (scores)

print (scores.mean())

```

```

importances = clf.feature_importances_

indices = np.argsort(importances)[::-1]

features = cols

```

```

def readn(nstr):

    engine = pyttsx3.init()

    engine.setProperty('voice', "english+f5")

    engine.setProperty('rate', 130)

    engine.say(nstr)

    engine.runAndWait()

    engine.stop()

```

```

severityDictionary=dict()

description_list = dict()

precautionDictionary=dict()

symptoms_dict = {}

```

```

for index, symptom in enumerate(x):

```

```

        symptoms_dict[symptom] = index
def calc_condition(exp,days):
    sum=0
    for item in exp:
        sum=sum+severityDictionary[item]
    if((sum*days)/(len(exp)+1)>13):
        print("You should take the consultation from doctor. ")
    else:
        print("It might not be that bad but you should take precautions.")

def getDescription():
    global description_list
    with open('symptom_Description.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        for row in csv_reader:
            _description={row[0]:row[1]}
            description_list.update(_description)

def getSeverityDict():
    global severityDictionary
    with open('symptom_severity.csv') as csv_file:

        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        try:

```

```

        for row in csv_reader:

            _diction={row[0]:int(row[1])}

            severityDictionary.update(_diction)

    except:

        pass

def getprecautionDict():

    global precautionDictionary

    with open('symptom_precaution.csv') as csv_file:

        csv_reader = csv.reader(csv_file, delimiter=',')

        line_count = 0

        for row in csv_reader:

            _prec={row[0]:[row[1],row[2],row[3],row[4]]}

            precautionDictionary.update(_prec)

def getInfo():

    # name=input("Name:")

    print("Your good name please \t\t\t\t\t",end="->")

    name=input("")

    print("Greetings!!!",name)

def check_pattern(dis_list,inp):

    import re

    pred_list=[]

    ptr=0

```

```

patt = "^" + inp + "$"

regexp = re.compile(inp)

for item in dis_list:

    # print(f"comparing {inp} to {item}")

    if regexp.search(item):

        pred_list.append(item)

        # return 1,item

if(len(pred_list)>0):

    return 1,pred_list

else:

    return ptr,item


def sec_predict(symptoms_exp):

    df = pd.read_csv('Training.csv')

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

    y = df['prognosis']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=20)

    rf_clf = DecisionTreeClassifier()

    rf_clf.fit(X_train, y_train)

    symptoms_dict = {}

    for index, symptom in enumerate(X):

        symptoms_dict[symptom] = index

```

```

input_vector = np.zeros(len(symptoms_dict))

for item in symptoms_exp:
    input_vector[[symptoms_dict[item]]] = 1

return rf_clf.predict([input_vector])

def print_disease(node):
    #print(node)
    node = node[0]
    #print(len(node))
    val = node.nonzero()
    # print(val)
    disease = le.inverse_transform(val[0])
    return disease

def tree_to_code(tree, feature_names):
    tree_ = tree.tree_
    # print(tree_)
    feature_name = [
        feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
        for i in tree_.feature
    ]

    chk_dis=",".join(feature_names).split(",")

```

```

symptoms_present = []

# conf_inp=int()

while True:

    print("Enter the symptom you are experiencing \t\t\t\t\t",end="->")

    disease_input = input("")

    conf,cnf_dis=check_pattern(chk_dis,disease_input)

    if conf==1:

        print("Searches related to input: ")

        for num,it in enumerate(cnf_dis):

            print(num,"",it)

            if num!=0:

                print(f"Select the one you meant (0 - {num}): ", end="")

                conf_inp = int(input(""))

            else:

                conf_inp=0

        disease_input=cnf_dis[conf_inp]

        break

    # print("Did you mean: ",cnf_dis,"?(yes/no) :",end="")

    # conf_inp = input("")

    # if(conf_inp=="yes"):

    #     break

else:

```

```

        print("Enter valid symptom.")
while True:
    try:
        num_days=int(input("Okay. From how many days ? : "))

        break
    except:
        print("Enter number of days.")

def recurse(node, depth):
    indent = " " * depth
    if tree_.feature[node] != _tree.TREE_UNDEFINED:
        name = feature_name[node]
        threshold = tree_.threshold[node]

        if name == disease_input:
            val = 1
        else:
            val = 0
        if val <= threshold:
            recurse(tree_.children_left[node], depth + 1)
        else:
            symptoms_present.append(name)
            recurse(tree_.children_right[node], depth + 1)
    else:
        present_disease = print_disease(tree_.value[node])

```

```

# print( "You may have " + present_disease )

red_cols = reduced_data.columns

symptoms_given =
red_cols[reduced_data.loc[present_disease].values[0].nonzero()]

# dis_list=list(symptoms_present)

# if len(dis_list)!=0:

#     print("symptoms present " + str(list(symptoms_present)))

# print("symptoms given " + str(list(symptoms_given)) )

print("Are you experiencing any ")


symptoms_exp=[]

for syms in list(symptoms_given):

    inp=""

    print(syms,"? : ",end="")

    while True:

        inp=input("")

        if(inp=="yes" or inp=="no"):

            break

        else:

            print("provide proper answers i.e. (yes/no) : ",end="")

    if(inp=="yes"):

        symptoms_exp.append(syms)


second_prediction=sec_predict(symptoms_exp)

# print(second_prediction)

calc_condition(symptoms_exp,num_days)

```

```

if(present_disease[0]==second_prediction[0]):
    print("You may have ", present_disease[0])

    print(description_list[present_disease[0]])

    # readn(f"You may have {present_disease[0]}")
    # readn(f"{description_list[present_disease[0]]}")

else:
    print("You may have ", present_disease[0], "or ", second_prediction[0])
    print(description_list[present_disease[0]])
    print(description_list[second_prediction[0]])

# print(description_list[present_disease[0]])
precaution_list=precautionDictionary[present_disease[0]]

print("Take following measures : ")
for i,j in enumerate(precaution_list):
    print(i+1,")",j)

# confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
# print("confidence level is " + str(confidence_level))

recurse(0, 1)

getSeverityDict()

```

getDescription()
getprecautionDict()
getInfo()
tree_to_code(clf,cols)

B) SCREENSHOTS

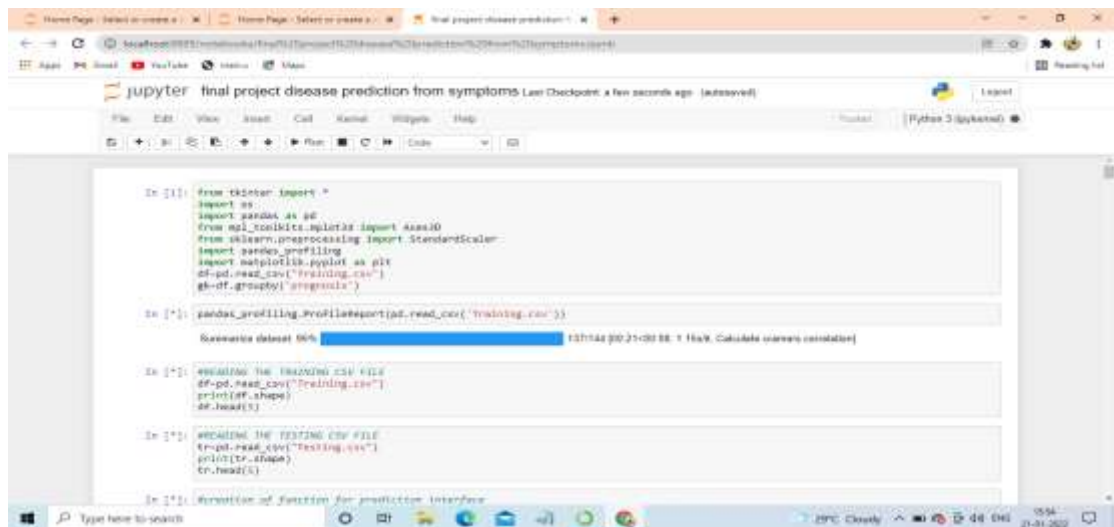


Fig B.1 Dataset



Fig B.2 Account Registration Screen

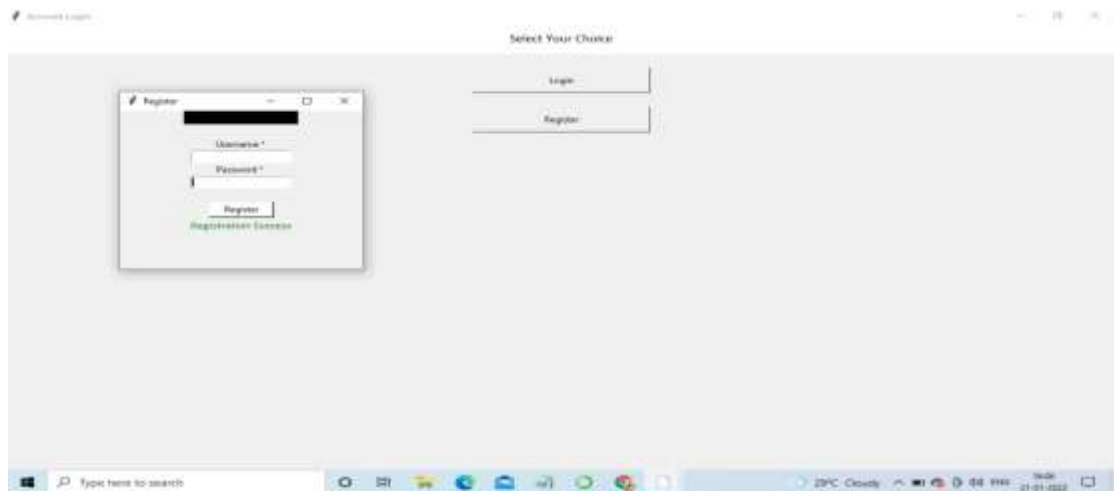


Fig B.3 Registration Success

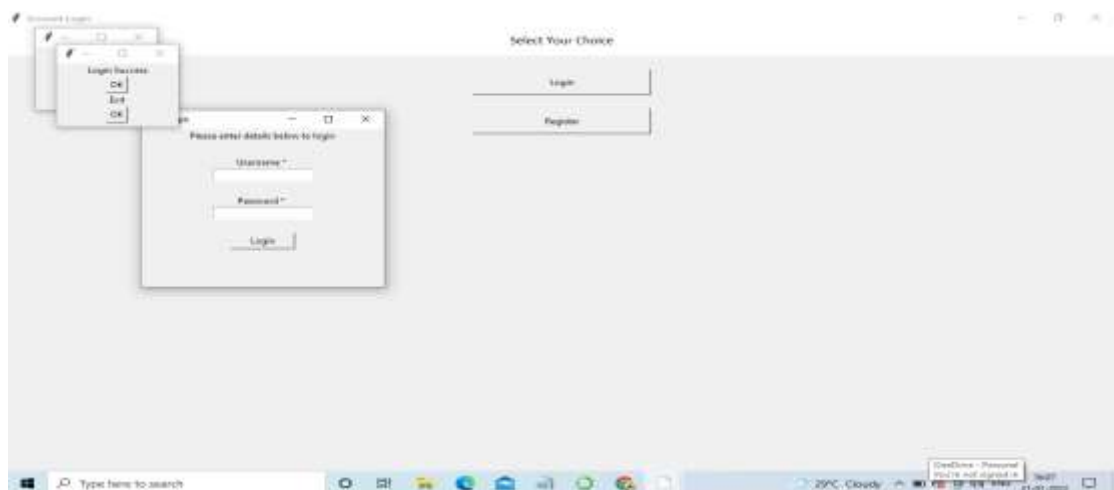


Fig B.4 Login Success

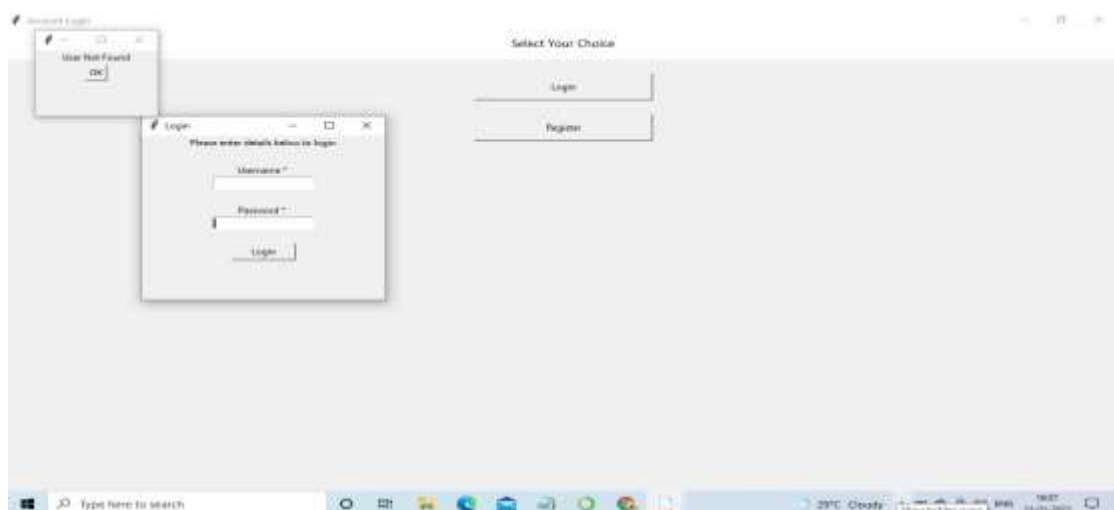


Fig B.5 Invalid Login



Fig B.6 GUI Interface



Fig B.7 Interface with inputs

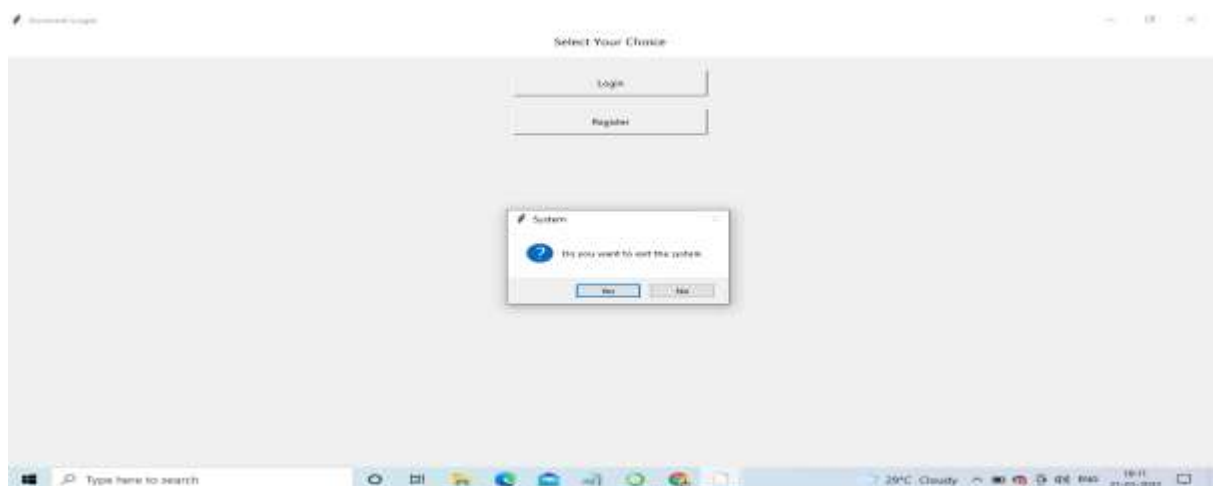
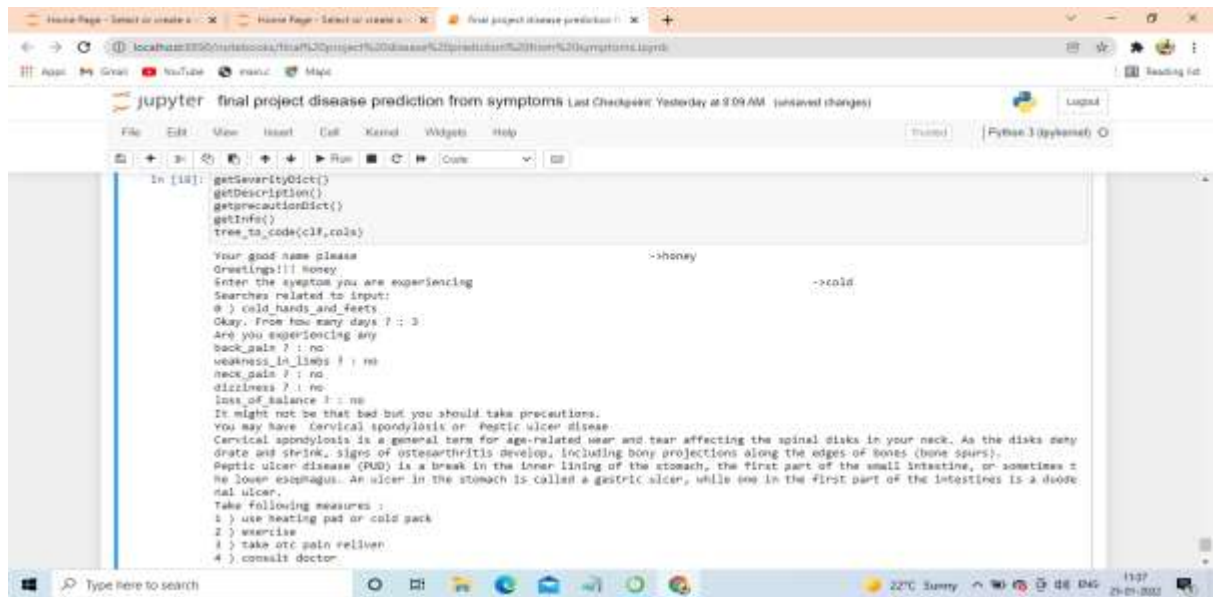


Fig B.8 Exit System



```
In [18]: getSeverityDict()
         getDescription()
         getPrecautionDict()
         getInfo()
         tree_to_code(cif,cols)

Your good name please                                ->honey
Greetings!!! Honey
Enter the symptoms you are experiencing                ->cold
Searches related to input:
0 ) cold_hands_and_feets
Okay, from how many days ? : 3
Are you experiencing any
back_pain ? : no
weakness_in_limbs ? : no
neck_pain ? : no
dizziness ? : no
loss_of_balance ? : no
It might not be that bad but you should take precautions.
You may have Cervical spondylosis or Peptic ulcer disease.
Cervical spondylosis is a general term for age-related wear and tear affecting the spinal disks in your neck. As the disks deny
drate and shrink, signs of osteoarthritis develop, including bony projections along the edges of bones (bone spurs).
Peptic ulcer disease (PUD) is a break in the inner lining of the stomach, the first part of the small intestine, or sometimes t
he lower esophagus. An ulcer in the stomach is called a gastric ulcer, while one in the first part of the intestines is a duode
nal ulcer.
Take following measures :
1 ) use heating pad or cold pack
2 ) exercise
3 ) take otc pain reliver
4 ) consult doctor
```

Fig B.9 Output of Chatbot

