MINI PROJECT REPORT

On

AI Chatbot For HealthCare

Submitted by

Shivam Kumar
Roll No: 171500320
Shikha Bansal
Roll No: 171500314

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA

2019

Acknowledgment

We would like to express our sincere gratitude to our mentor **Mr. Vivek Kumar** for guiding us. We deeply respect the mentor for his vast knowledge, numerous suggestions, and strong passion to complete this project. Valuable discussions with him not only made our work smooth but also encouraged us to think more professionally in the field of Machine Learning.

We also thank all our teaching and non-teaching staff for their support and well wishes.

Finally, We would like to express our deepest gratitude to our parents and friends for their encouragement and support.

ABSTRACT

Normally Users are not aware of all the treatment or symptoms regarding the particular disease. For small problems, the user has to go personally to the hospital for a check-up which is more time-consuming. Also handling the telephonic calls for the complaints is quite hectic. Such a problem can be solved by using medical ChatBot by giving proper guidance regarding healthy living. Today's people are more likely to be addicted to the internet but they are not concerned about their personal health.big disease can start from small problems such as headaches which feels normal but it may beginning of big diseases such as brain tumor .most of the disease can be identified by common symptoms so the disease can be predicted if the patient's body is analyzed periodically.

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

DECLARATION

We hereby declare that the project work entitled "**AI Chatbot For HealthCare**" submitted to the GLA University Mathura**,** is a record of an original work done by our team under the guidance of **Mr.Vivek Kumar, Assistant Professor of GLA University, Mathura**

Name of Candidates: **1. Shivam Kumar**

2. Shikha Bansal

Course: B.Tech (CSE)

Year: III

Semester: 5

TABLE OF CONTENT

CHAPTER 1 Introduction to HealthCare Chatbot

The purpose of this project to provide the admin has to collect the patient's medical history of records and filter it appropriately by applying data preprocessing techniques. Admin's functionalities are to collecting the appropriate medical records of the patients, handle missing values, handling categorical values, creating sparse matrix representation, Feeding data to the autonomous pipeline for predictions, selecting and training an appropriate machine learning algorithm.

The visitor can perform the basic task of the visitor is to access the Chatbot from the front end and reply to its queries with a binary response (Yes/No). The visitor will be shown a confidence interval related to a certain prognosis which needs to be further investigated and experimented with for better results. The first step is to start their procedure, then one by one all the symptoms come in clients' screens. They will have to reply with yes or no answer.

Once a problem is found then they will have to click yes, then the patient can see their problem on screen. The Best Part is that it will provide the doctor's information like the Doctor's name and his/her website link. So that one can easily find their doctor with don't face any type of problem, and start their treatment. This will prepare with the help of Chatbot so that one can even check their problem at any time. You have to just reply with the clicking of button Yes or No.

CHAPTER 2 Software Requirement Analysis

2.1 - Requirement Analysis

Requirement Analysis is a software engineering task that bridges the gap between system-level software allocation and software design. It provides the system engineer to specify software function and performance indicate software's interface with the other system elements and establish constraints that software must meet.

The basic aim of this stage is to obtain a clear picture of the needs and requirements of the end-user and also the organization. The analysis involves interaction between the clients and the analysis. Usually, analysts research a problem from any questions asked and reading existing documents. The analysts have to uncover the real needs of the user even if they don't know them clearly. During the analysis, it is essential that a complete and consistent set of specifications emerge for the system. Here it is essential to resolve the contradictions that could emerge from information got from various parties. This is essential to ensure that the final specifications are consistent.

It may be divided into 5 areas of effort.

- Problem recognition
- Evaluation and synthesis
- Modeling
- Specification
- Review

Each Requirement analysis method has a unique point of view. However, all analysis methods are related by a set of operational principles. They are:

- The information domain of the problem must be represented and understood.
- The functions that the software is to perform must be defined.
- The behavior of the software as a consequence of external events must be defined.
- The models that depict information function and behavior must be partitioned in a hierarchical or layered fashion.
- The analysis process must move from essential information to implementation detail.

2.2- Software Requirements Specification

Software Requirements Specification plays an important role in creating quality software solutions. The specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation.

Requirements may be specified in a variety of ways. However, there are some guidelines worth following: -

- Representation format and content should be relevant to the problem.
- Information contained within the specification should be nested.
- Diagrams and other notational forms should be restricted in number and consistent in use.
- Representations should be revisable.

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to the software as a part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, and indication of performance requirements and design constraints, appropriate validation. Criteria and other data pertinent to requirements.

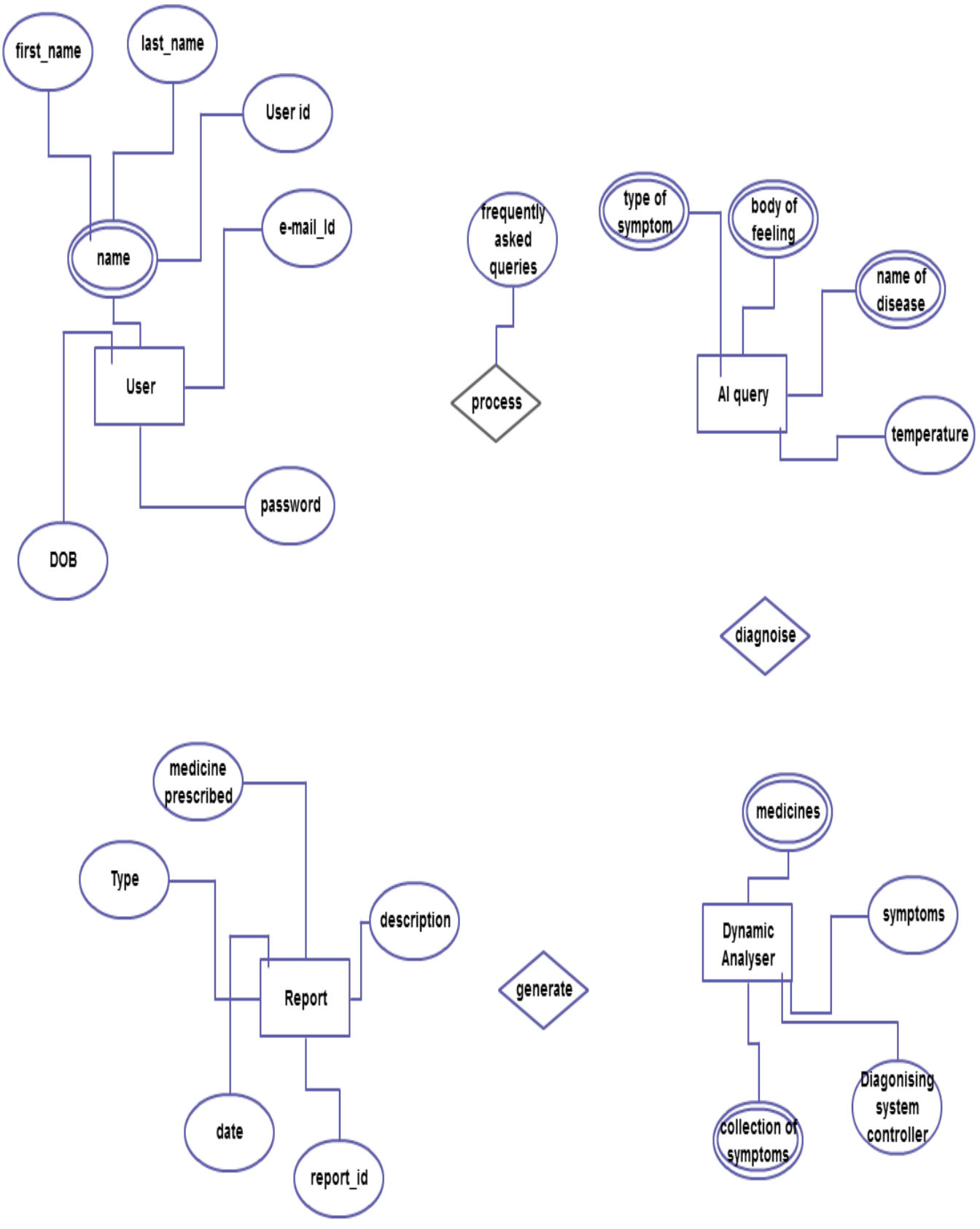
CHAPTER 3 System Design

System design is the most creative phase of system development. The term describes a final system and the process by which it is developed. The question in system design is: How the problem is to be solved?

A systematic method has to achieve beneficial results in the end. It involves starting with a vague idea and developing it into a series of steps. The series of steps for successful system design are:

The first step is to study the problem completely because we should know the goal. We should see what kind of output we require and what king of input we give so that we can get the desired result.

We should see what kind of program should be developed to reach the final goal. Then we write individual programs, which later on joining solve the specified problem. Then we test these programs and make necessary corrections to achieve the target of the programs.


first_name
last_name
User id
e-mail_Id
name
User
password
DOB
frequently asked queries
process
type of symptom
body of feeling
name of disease
AI query
temperature
diagnoise
medicine prescribed
Type
description
Report
date
report_id
generate
medicines
Dynamic Analyser
symptoms
collection of symptoms
Diagonising system controller
[online diagramming & design] creately.com

CHAPTER 4 Testing Phase

As testing is the last phase before the final software is delivered, it has the enormous responsibility of detecting any type of error that may in the software. A software typically undergoes changes even after it has been delivered. And to validate that a change has not affected some old functionality of software regression testing is performed

Levels of Testing

The basic levels of testing are unit testing, integration testing and system, and acceptance testing. These different levels of testing attempt to detect different types of faults.

Code/Unit Testing

Code testing and implementation is a critical process that can even consume more than sixty percent of the development time.

Testing

The system development life cycle involves the phases of testing and debugging after the requirement analysis, designing and coding. The project in question was tested, debugged and implemented successfully.

Two strategies of software testing adopted for the new system are as follows

Code testing

Code testing was carried out to see the correctness of the logic involved and the correctness of the modules. Tests were conducted based on the sample. All the modules are checked separately for assuming correctness and accuracy in all the calculations.

Specification Testing

It examines the specification stating what program should do and how it performs under various conditions. This testing strategy is a better strategy since it focuses on the way software is expected to work.

Unit Testing

During the phase of unit testing, different constituent modules were tested against the specifications produced during the design for the modules. Unit testing is essentially for the verification of the code produced during the coding of the phase, and the goal is to test the internal logic of the modules. The modules once tested were then considered for integration and use by others.

Test Planning

Testing needs to be planned, to be cost and time effective. Planning is setting out standards for tests. Test plans set out the context in which individual engineers can place their own work. Typical test plan contains:

Overview of the testing process

- Requirements traceability (to ensure that all requirements are tested)
- List of items to be tested
- Schedule
- Recording procedures so that test results can be audited
- Hardware and software requirements
- Constraints

CHAPTER 5 Implementation

Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the changeover, an evaluation, of change over methods. A part of planning a major task of preparing the implementation is the education of users. The more complex system is implemented, the more involved will be the system analysis and design effort required just for implementation. An implementation coordinating committee based on policies of the individual organization has been appointed. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions may regarding the equipment that has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or the inability to handle certain types of transaction while using the new system.

The major elements of the implementation plan are test plan, training plan, equipment installation plan, and a conversion plan.

Interface

Console Based

In this, we have to write Yes or No only.

If our Symptoms are not matched then we have to write no on our screen.

When our Symptoms will be matched then we just have to write yes.

```
.... execute_bot()
Please reply with yes/Yes or no/No for the following symptoms
slurred_speech ?

no
pain_behind_the_eyes ?

no
receiving_blood_transfusion ?

no
red_spots_over_body ?

no
unsteadiness ?
```

Symptoms Window

When we write Yes on our console screen, then our matched problem will be found on screen. And it will also tell the Symptoms which may a patient have.

```
increased_appetite ?

yes
['You may have Diabetes ']

symptoms present  ['increased_appetite']

symptoms given ['fatigue', 'weight_loss', 'restlessness', 'lethargy',
'irregular_sugar_level', 'blurred_and_distorted_vision', 'obesity',
'excessive_hunger', 'increased_appetite', 'polyuria']
```

CHAPTER 6 Tkinter Features and Functions

Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps − Import the *Tkinter* module.
Create the GUI application main window.
Add one or more of the above-mentioned widgets to the GUI application.
Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

Widgets are something like elements in the **HTML**. You will find different types of **widgets** to the different types of elements in the **Tkinter**.
Let's see the brief introduction to all of these widgets in the **Tkinter**.
Button: - **Button** widget is used to place the buttons in the **Tkinter**.
Canvas: - **Canvas** is used to draw shapes in your **GUI**.
Check button: - **Check button** is used to create the check buttons in your application. You can select more than one option at a time.
Entry: - **Entry** widget is used to create input fields in the **GUI**.
Frame: - **Frame** is used as containers in the **Tkinter**.
Label: - **Label** is used to create single line widgets like **text**, **images**, etc.
Menu: - **Menu** is used to create menus in the **GUI**.

Login Menu

A login is a set of credentials used to authenticate a user. Most often, these consist of a username and password. However, a login may include other information, such as a PIN number, passcode, or passphrase. Some logins require a biometric identifier, such as a fingerprint or retina scan.

Logins are used by websites, computer applications, and mobile apps. They are a security measure designed to prevent unauthorized access to confidential data. When a login fails (i.e., the username and password combination does not match a user account), the user is disallowed access. Many systems block users from even trying to log in after multiple failed login attempts.

Sign Up Menu

The Sign up module has been developed using the Tkinter GUI framework written in Python. It facilitates the user to save his/her data into the Oracle database. The signup form will be explicitly used to insert the records of doctors who will be using the disease prediction system. The doctor's data has been scraped from Internet for research purposes. The signup module opens the prediction window for a legitimate user and displays a message box in case of failed authentication.

Symptoms Window

The Symptoms window is created or called at run time when the user is inserting the symptoms into the model. When the model is satisfied with an appropriate number of inputs. It then generates a response in the form of the predicted disease, symptoms given, confidence interval and the recommendation for the doctor to visit next. The symptoms window also provides a link to book an appointment with the concerned doctor which can copy and pasted into the browser by the user for further operations.

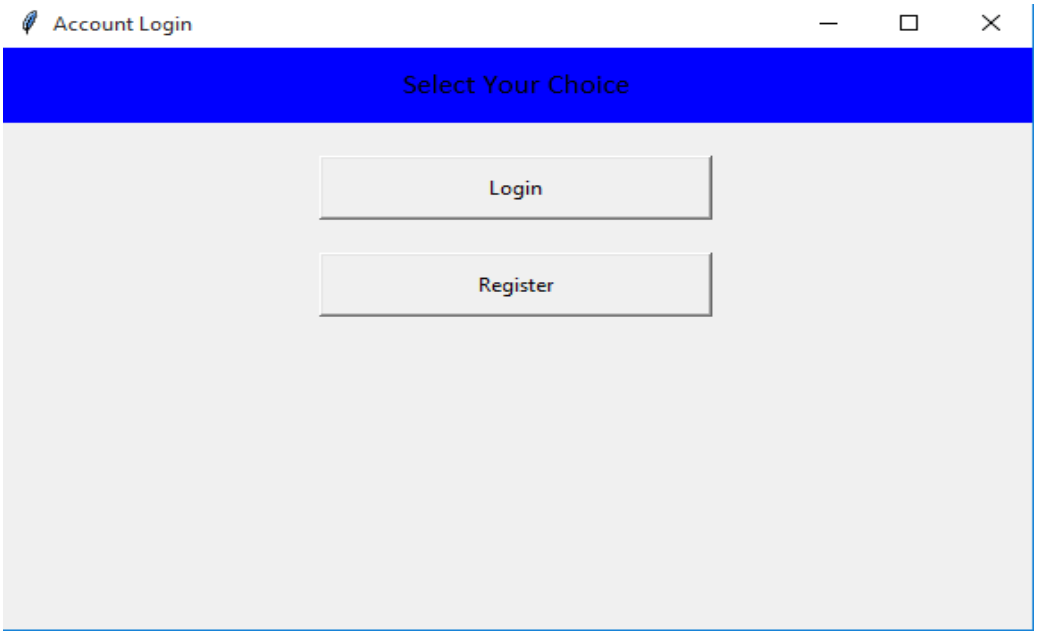


Fig 1

Choice Window

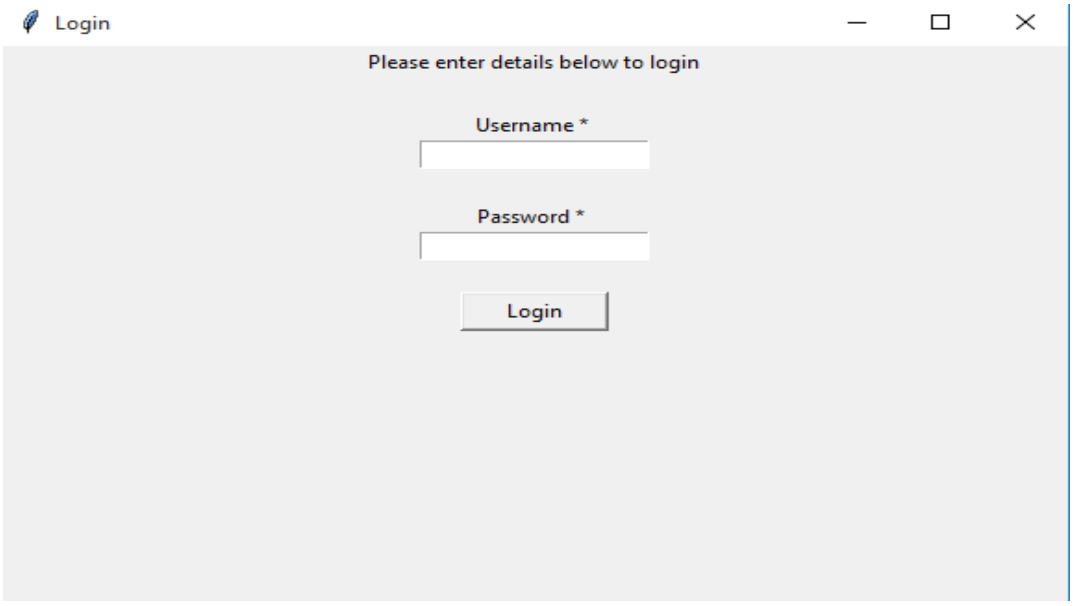


Fig 2

Login Window

Fig 3

Sign Up Window

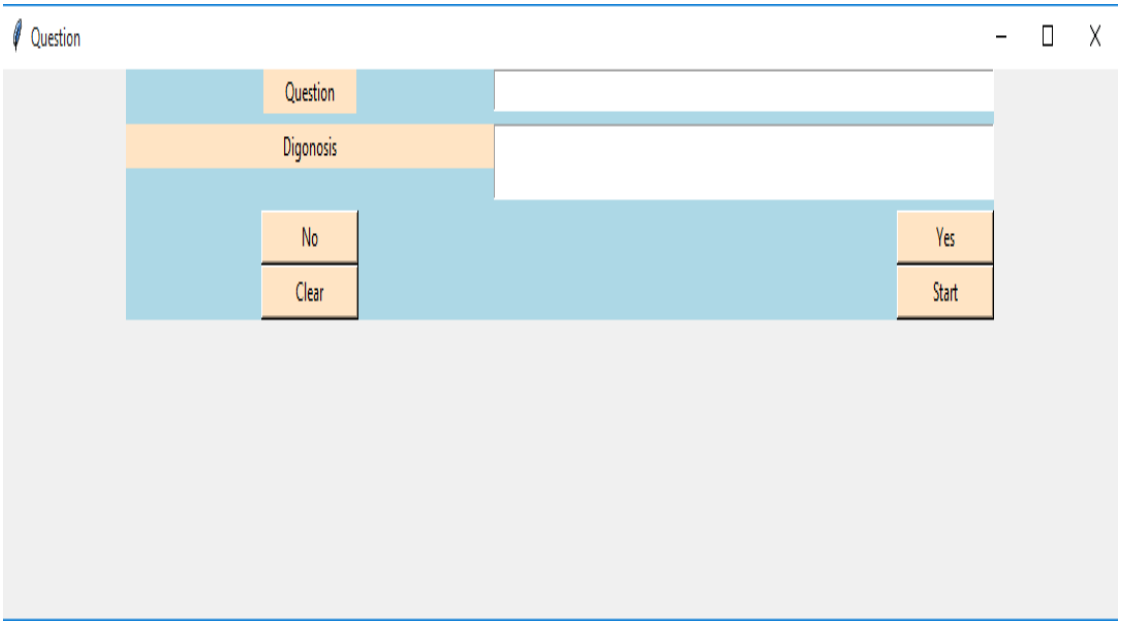


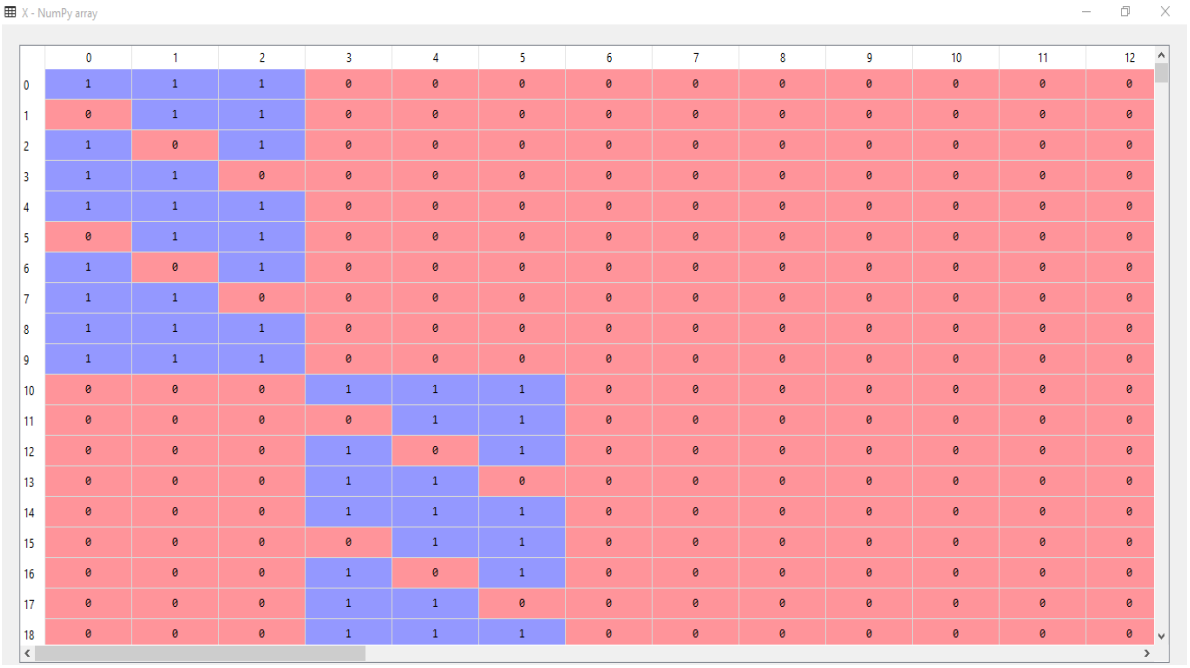
Fig 4

Main Window

CHAPTER 7 Datasets

X_ DATASET

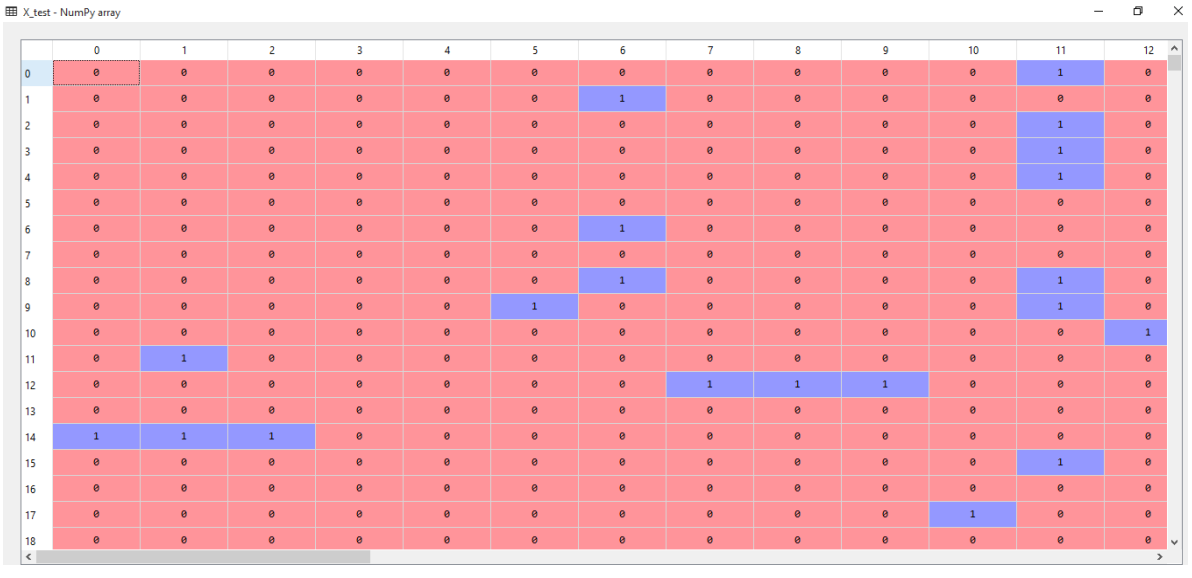
It contains number of input given by users.



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	0	0
5	0	1	1	0	0	0	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0	0	0	0	0	0	0
7	1	1	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	0	0	0	0	0	0	0	0	0	0
9	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	1	1	1	0	0	0	0	0	0	0
11	0	0	0	0	1	1	0	0	0	0	0	0	0
12	0	0	0	1	0	1	0	0	0	0	0	0	0
13	0	0	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	1	1	1	0	0	0	0	0	0	0
15	0	0	0	0	1	1	0	0	0	0	0	0	0
16	0	0	0	1	0	1	0	0	0	0	0	0	0
17	0	0	0	1	1	0	0	0	0	0	0	0	0
18	0	0	0	1	1	1	0	0	0	0	0	0	0

X_TEST DATASET

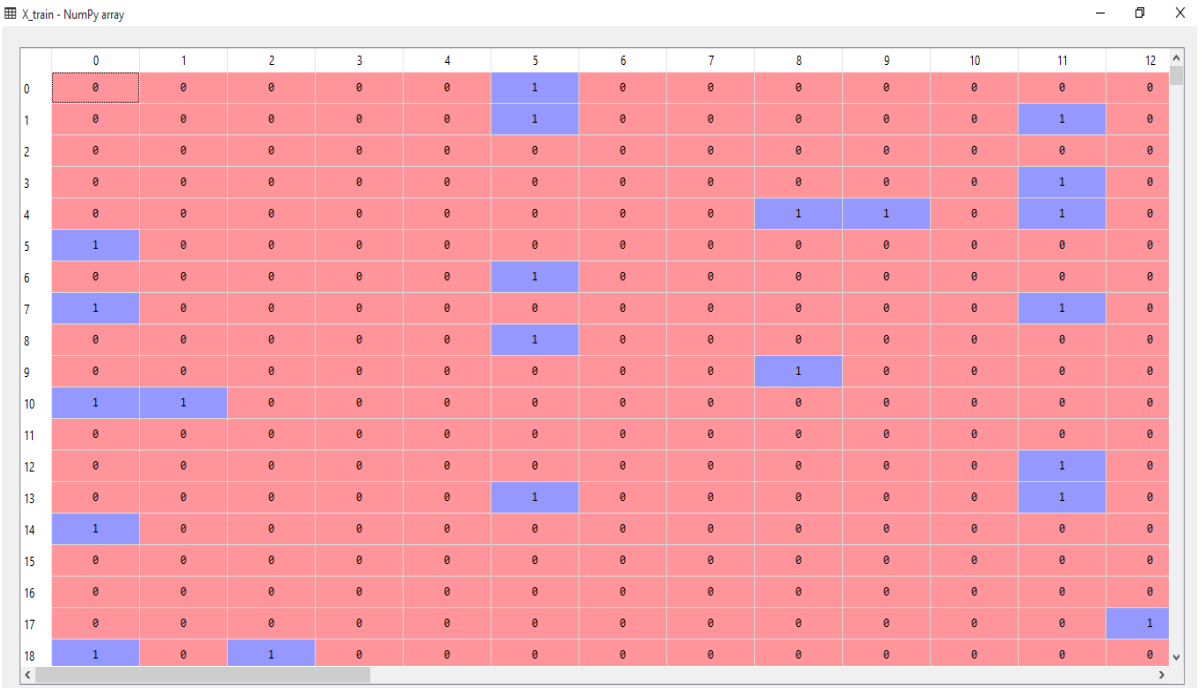
It divides the x just to check if our output will be correct or not.



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	1	0
9	0	0	0	0	0	1	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	1	1	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0

X_TRAIN DATASET

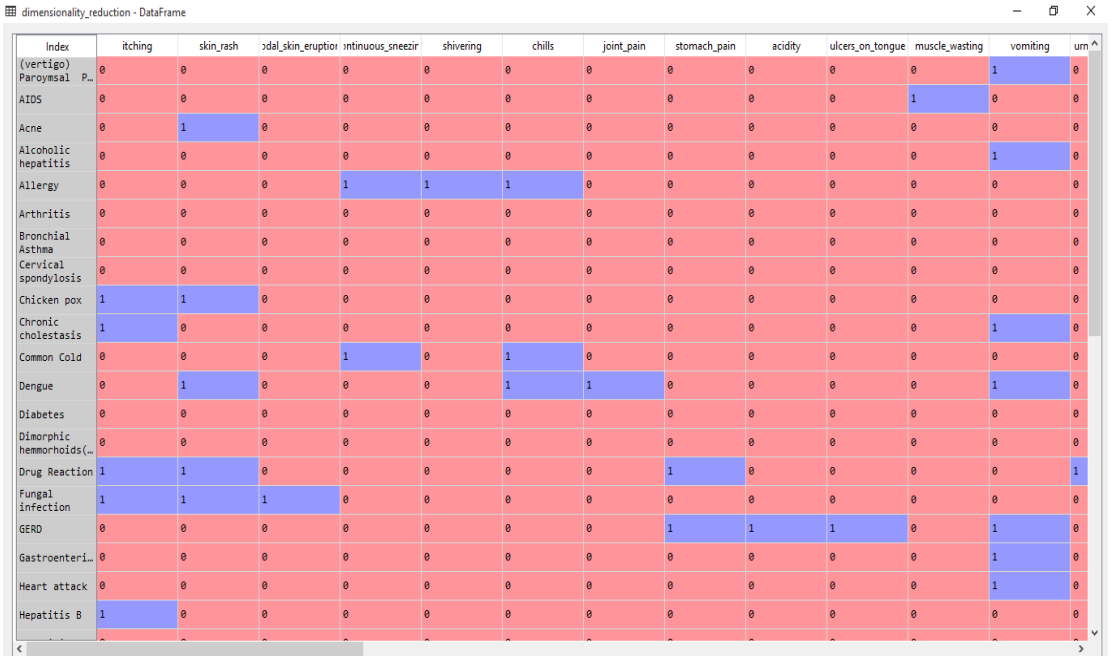
It also divide x dataset into another part called as training set.



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	1	1	0	1	0
5	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0
10	1	1	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0
13	0	0	0	0	0	1	0	0	0	0	0	1	0
14	1	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	1
18	1	0	1	0	0	0	0	0	0	0	0	0	0

DIMENSIONALIY_REDUCTION

To avoid repeat value and contain unique values.



Index	itching	skin_rash	odal_skin_eruptio	ntinuous_sneezir	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	urn
(vertigo) Paroymsal P…	0	0	0	0	0	0	0	0	0	0	0	1	0
AIDS	0	0	0	0	0	0	0	0	0	0	1	0	0
Acne	0	1	0	0	0	0	0	0	0	0	0	0	0
Alcoholic hepatitis	0	0	0	0	0	0	0	0	0	0	0	1	0
Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0
Arthritis	0	0	0	0	0	0	0	0	0	0	0	0	0
Bronchial Asthma	0	0	0	0	0	0	0	0	0	0	0	0	0
Cervical spondylosis	0	0	0	0	0	0	0	0	0	0	0	0	0
Chicken pox	1	1	0	0	0	0	0	0	0	0	0	0	0
Chronic cholestasis	1	0	0	0	0	0	0	0	0	0	0	1	0
Common Cold	0	0	0	1	0	1	0	0	0	0	0	0	0
Dengue	0	1	0	0	0	1	1	0	0	0	0	1	0
Diabetes	0	0	0	0	0	0	0	0	0	0	0	0	0
Dimorphic hemmorhoids(…	0	0	0	0	0	0	0	0	0	0	0	0	0
Drug Reaction	1	1	0	0	0	0	0	1	0	0	0	0	1
Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0
GERD	0	0	0	0	0	0	0	1	1	1	0	1	0
Gastroenteri…	0	0	0	0	0	0	0	0	0	0	0	1	0
Heart attack	0	0	0	0	0	0	0	0	0	0	0	1	0
Hepatitis B	1	0	0	0	0	0	0	0	0	0	0	0	0

DISEASES

Number of Diseases.

diseases - DataFrame

Index	prognosis
0	(vertigo) Paroymsal P…
1	AIDS
2	Acne
3	Alcoholic hepatitis
4	Allergy
5	Arthritis
6	Bronchial Asthma
7	Cervical spondylosis
8	Chicken pox
9	Chronic cholestasis
10	Common Cold
11	Dengue
12	Diabetes
13	Dimorphic hemmorhoids(…
14	Drug Reaction
15	Fungal infection
16	GERD
17	Gastroenteri…
18	Heart attack
19	Hepatitis B
20	Hepatitis C

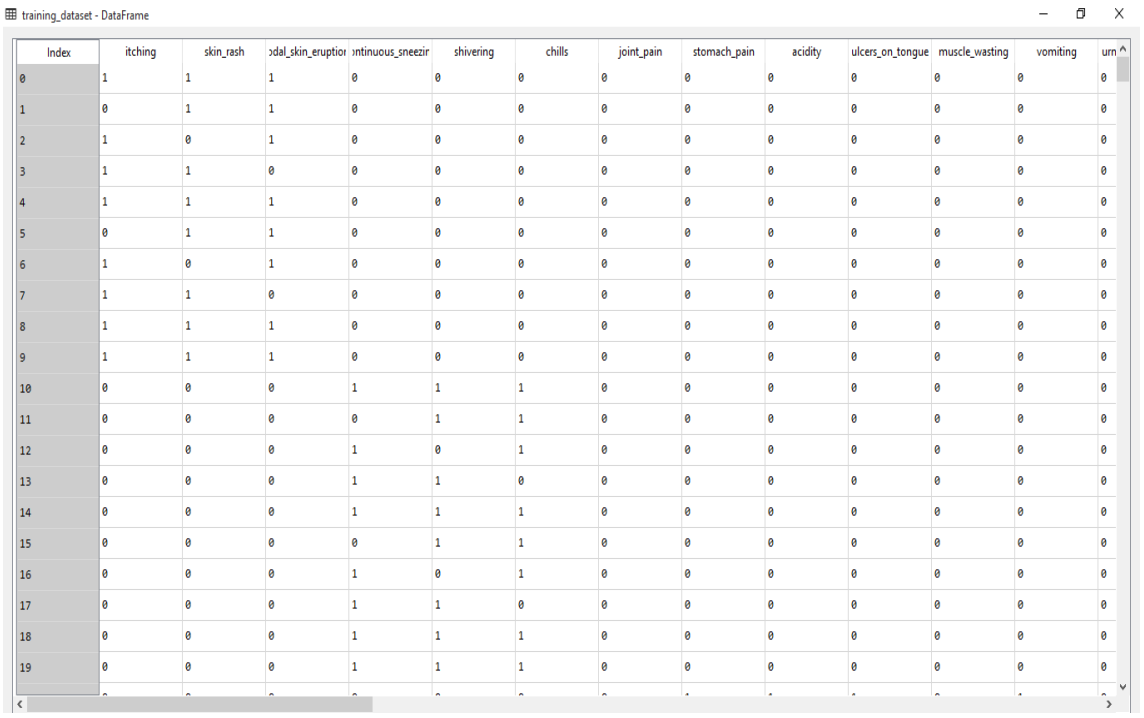
DOCTER DATASET

It will provide Doctor's name with its link where one can visit his/her site and also tell problem. It contain multiple rows and three columns.

doctors - DataFrame

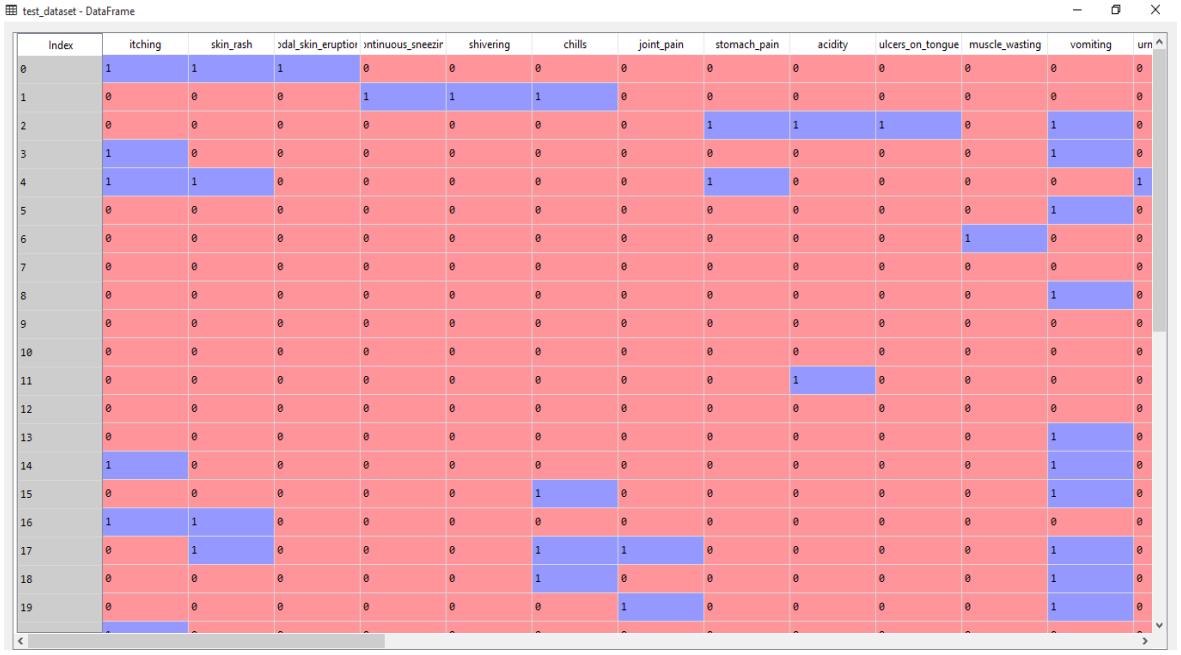
Index	name	link	disease
0	Dr. Amarpreet Singh Riar	https:// www.practo.c…	(vertigo) Paroymsal P…
1	Dr. (Maj.)Sharad…	https:// www.practo.c…	AIDS
2	Dr. Anirban Biswas	https:// www.practo.c…	Acne
3	Dr. Aman Vij	https:// www.practo.c…	Alcoholic hepatitis
4	Dr. Mansi Arya	https:// www.practo.c…	Allergy
5	Dr. Sunil Kumar Dwivedi	https:// www.practo.c…	Arthritis
6	Dr. Chhavi Bansal	https:// www.practo.c…	Bronchial Asthma
7	Dr. Sneh Khera	https:// www.practo.c…	Cervical spondylosis
8	Dr. Inderjeet Singh	https:// www.practo.c…	Chicken pox
9	Dr. Suman Mohan	https:// www.practo.c…	Chronic cholestasis
10	Dr. Manish Munjal	https:// www.practo.c…	Common Cold
11	Dr. Ajay Jain	https:// www.practo.c…	Dengue
12	Dr. Anshul Gupta	https:// www.practo.c…	Diabetes
13	Dr. B B Khatri	https:// www.practo.c…	Dimorphic hemmorhoids(…
14	Dr. Rajeev Adhana	https:// www.practo.c…	Drug Reaction
15	Dr. Vidit Tripathi	https:// www.practo.c…	Fungal infection
16	Dr. Arun Wadhawan	https:// www.practo.c…	GERD
17	Dr. Neha Sood	https:// www.practo.c…	Gastroenteri…
18	Dr. Vineet Narula	https:// www.practo.c…	Heart attack
19	Dr. Yogesh Jain	https:// www.practo.c…	Hepatitis B
20	Dr. Rakesh Singh	https:// www.practo.c…	Hepatitis C

TRAINING_DATASET


training_dataset - DataFrame

Index	itching	skin_rash	ɔdal_skin_eruptio	ɔntinuous_sneezir	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	urn
0	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	0	0
5	0	1	1	0	0	0	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0	0	0	0	0	0	0
7	1	1	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	0	0	0	0	0	0	0	0	0	0
9	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	1	1	1	0	0	0	0	0	0	0
11	0	0	0	0	1	1	0	0	0	0	0	0	0
12	0	0	0	1	0	1	0	0	0	0	0	0	0
13	0	0	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	1	1	1	0	0	0	0	0	0	0
15	0	0	0	0	1	1	0	0	0	0	0	0	0
16	0	0	0	1	0	1	0	0	0	0	0	0	0
17	0	0	0	1	1	0	0	0	0	0	0	0	0
18	0	0	0	1	1	1	0	0	0	0	0	0	0
19	0	0	0	1	1	1	0	0	0	0	0	0	0

TEST DATASET


test_dataset - DataFrame

Index	itching	skin_rash	ɔdal_skin_eruptio	ɔntinuous_sneezir	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	urn
0	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1	1	0	1	0
3	1	0	0	0	0	0	0	0	0	0	0	1	0
4	1	1	0	0	0	0	0	1	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0
14	1	0	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	1	0	0	0	0	0	1	0
16	1	1	0	0	0	0	0	0	0	0	0	0	0
17	0	1	0	0	0	1	1	0	0	0	0	1	0
18	0	0	0	0	0	1	0	0	0	0	0	1	0
19	0	0	0	0	0	0	1	0	0	0	0	1	0

Y _TEST DATASET

It contains output only for test dataset.

y_test - NumPy array

	0
0	18
1	40
2	36
3	25
4	33
5	23
6	31
7	39
8	40
9	37
10	38
11	27
12	16
13	39
14	15
15	32
16	23
17	1
18	26

Y_TRAINING DATASET

y_train - NumPy array

	0
0	34
1	36
2	24
3	21
4	16
5	19
6	37
7	9
8	34
9	30
10	8
11	24
12	25
13	37
14	19
15	13
16	26
17	38
18	15

CHAPTER 8 Code Section

BASED ON TKINTER

For Login Window

```
# import modules

from tkinter import *
import os


# Designing window for registration
def destroyPackWidget(parent):
    for e in parent.pack_slaves():
        e.destroy()
def register():
    global root,register_screen

    destroyPackWidget(root)
    register_screen=root
#    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="blue").pack()
    Label(register_screen, text="").pack()
    username_lable = Label(register_screen, text="Username * ")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
    password_lable = Label(register_screen, text="Password * ")
    password_lable.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="blue", 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 btnSucess_Click():
    global root
    destroyPackWidget(root)
def register_user():
    global root,username,password
    username_info = username.get()
    password_info = password.get()
    print("abc",username_info,password_info,"xyz")
    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(root, text="Registration Success", fg="green", font=("calibri", 11)).pack()
    Button(root,text="Click Here to proceed",command=btnSucess_Click).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=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(frmmain):
    main_screen=frmmain

    main_screen.geometry("300x250")
    main_screen.title("Account Login")
    Label(main_screen,text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13)).pack()
    Label(main_screen,text="").pack()
    Button(main_screen,text="Login", height="2", width="30", command=login).pack()
    Label(main_screen,text="").pack()
    Button(main_screen,text="Register", height="2", width="30", command=register).pack()



root = Tk()
main_account_screen(root)

root.mainloop()
```

Question Diagnosis Window

```
######## A Healthcare Domain Chatbot to simulate the predictions of a General Physician ########
######## A pragmatic Approach for Diagnosis #############

# Importing the libraries
from tkinter import *
from tkinter import messagebox
import os
import webbrowser

import numpy as np
import pandas as pd


class HyperlinkManager:

    def __init__(self, text):

        self.text = text

        self.text.tag_config("hyper", foreground="blue", underline=1)

        self.text.tag_bind("hyper", "<Enter>", self._enter)
        self.text.tag_bind("hyper", "<Leave>", self._leave)
        self.text.tag_bind("hyper", "<Button-1>", self._click)

        self.reset()

    def reset(self):
        self.links = {}

    def add(self, action):
        # add an action to the manager.  returns tags to use in
        # associated text widget
        tag = "hyper-%d" % len(self.links)
        self.links[tag] = action
        return "hyper", tag

    def _enter(self, event):
        self.text.config(cursor="hand2")

    def _leave(self, event):
        self.text.config(cursor="")

    def _click(self, event):
        for tag in self.text.tag_names(CURRENT):
            if tag[:6] == "hyper-":
                self.links[tag]()
                return
```

```
# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')

# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
Y = training_dataset.iloc[:, -1].values

# Dimensionality Reduction for removing redundancies
dimensionality_reduction = training_dataset.groupby(training_dataset['prognosis']).max()

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(Y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols     = training_dataset.columns
cols     = cols[:-1]


# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree
```

```
# Method to simulate the working of a Chatbot by extracting and formulating questions
def print_disease(node):
        #print(node)
        node = node[0]
        #print(len(node))
        val  = node.nonzero()
        #print(val)
        disease = labelencoder.inverse_transform(val[0])
        return disease
def recurse(node, depth):
            global val,ans
            global tree_,feature_name,symptoms_present
            indent = "  " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                yield name + " ?"
#                ans = input()
                ans = ans.lower()
                if ans == 'yes':
                    val = 1
                else:
                    val = 0
                if  val <= threshold:
                    yield from recurse(tree_.children_left[node], depth + 1)
                else:
                    symptoms_present.append(name)
                    yield from recurse(tree_.children_right[node], depth + 1)
            else:
                strData=""
                present_disease = print_disease(tree_.value[node])
#                print( "You may have " +  present_disease )
#                print()
                strData="You may have :" +  str(present_disease)

                QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')

                red_cols = dimensionality_reduction.columns
                symptoms_given = red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
#                print("symptoms present  " + str(list(symptoms_present)))
```

```
 print()
strData="symptoms present:  " + str(list(symptoms_present))
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
 print("symptoms given "  +  str(list(symptoms_given)) )
 print()
strData="symptoms given: "  +  str(list(symptoms_given))
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
 print("confidence level is " + str(confidence_level))
 print()
strData="confidence level is: " + str(confidence_level)
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
 print('The model suggests:')
 print()
strData='The model suggests:'
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
row = doctors[doctors['disease'] == present_disease[0]]
 print('Consult ', str(row['name'].values))
 print()
strData='Consult '+ str(row['name'].values)
QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
 print('Visit ', str(row['link'].values))
#print(present_disease[0])
hyperlink = HyperlinkManager(QuestionDigonosis.objRef.txtDigonosis)
strData='Visit '+ str(row['link'].values[0])
def click1():
    webbrowser.open_new(str(row['link'].values[0]))
QuestionDigonosis.objRef.txtDigonosis.insert(INSERT, strData, hyperlink.add(click1))
#QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
yield strData


def tree_to_code(tree, feature_names):
        global tree_,feature_name,symptoms_present
        tree_ = tree.tree_
        #print(tree_)
        feature_name = [
            feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
            for i in tree_.feature
        ]
        #print("def tree({}):".format(", ".join(feature_names)))
        symptoms_present = []
#         recurse(0, 1)


def execute_bot():
#     print("Please reply with yes/Yes or no/No for the following symptoms")
    tree_to_code(classifier,cols)


# This section of code to be run after scraping the data

doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])


diseases = dimensionality_reduction.index
diseases = pd.DataFrame(diseases)

doctors = pd.DataFrame()
doctors['name'] = np.nan
doctors['link'] = np.nan
doctors['disease'] = np.nan

doctors['disease'] = diseases['prognosis']


doctors['name'] = doc_dataset['Name']
doctors['link'] = doc_dataset['Description']

record = doctors[doctors['disease'] == 'AIDS']
record['name']
record['link']
```

```
# Execute the bot and see it in Action
#execute_bot()


class QuestionDigonosis(Frame):
    objIter=None
    objRef=None
    def __init__(self,master=None):
        master.title("Question")
        # root.iconbitmap("")
        master.state("z")
#        master.minsize(700,350)
        QuestionDigonosis.objRef=self
        super().__init__(master=master)
        self["bg"]="light blue"
        self.createWidget()
        self.iterObj=None

    def createWidget(self):
        self.lblQuestion=Label(self,text="Question",width=12,bg="bisque")
        self.lblQuestion.grid(row=0,column=0,rowspan=4)

        self.lblDigonosis = Label(self, text="Digonosis",width=12,bg="bisque")
        self.lblDigonosis.grid(row=4, column=0,sticky="n",pady=5)

        # self.varQuestion=StringVar()
        self.txtQuestion = Text(self, width=100,height=4)
        self.txtQuestion.grid(row=0, column=1,rowspan=4,columnspan=20)

        self.varDiagonosis=StringVar()
        self.txtDigonosis =Text(self, width=100,height=14)
        self.txtDigonosis.grid(row=4, column=1,columnspan=20,rowspan=20,pady=5)

        self.btnNo=Button(self,text="No",width=12,bg="bisque", command=self.btnNo_Click)
        self.btnNo.grid(row=25,column=0)
        self.btnYes = Button(self, text="Yes",width=12,bg="bisque", command=self.btnYes_Click)
        self.btnYes.grid(row=25, column=1,columnspan=20,sticky="e")

        self.btnClear = Button(self, text="Clear",width=12,bg="bisque", command=self.btnClear_Click)
        self.btnClear.grid(row=27, column=0)
    def btnNo_Click(self):
        global val,ans
        global val,ans
        ans='no'
        str1=QuestionDigonosis.objIter.__next__()
        self.txtQuestion.delete(0.0,END)
        self.txtQuestion.insert(END,str1+"\n")

    def btnYes_Click(self):
        global val,ans
        ans='yes'
        self.txtDigonosis.delete(0.0,END)
        str1=QuestionDigonosis.objIter.__next__()

         self.txtDigonosis.insert(END,str1+"\n")

    def btnClear_Click(self):
        self.txtDigonosis.delete(0.0,END)
        self.txtQuestion.delete(0.0,END)
    def btnStart_Click(self):
        execute_bot()
        self.txtDigonosis.delete(0.0,END)
        self.txtQuestion.delete(0.0,END)
        self.txtDigonosis.insert(END,"Please Click on Yes or No for the Above symptoms in Question")
        QuestionDigonosis.objIter=recurse(0, 1)
        str1=QuestionDigonosis.objIter.__next__()
        self.txtQuestion.insert(END,str1+"\n")
```

```
class MainForm(Frame):
    main_Root = None
    def destroyPackWidget(self, parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        MainForm.main_Root = master
        super().__init__(master=master)
        master.geometry("300x250")
        master.title("Account Login")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13))
        self.lblMsg.pack()
        self.btnLogin=Button(self, text="Login", height="2", width="30", command=self.lblLogin_Click)
        self.btnLogin.pack()
        self.btnRegister=Button(self, text="Register", height="2", width="30", command=self.btnRegister_Click)
        self.btnRegister.pack()
    def lblLogin_Click(self):
        self.destroyPackWidget(MainForm.main_Root)
        frmLogin=Login(MainForm.main_Root)
        frmLogin.pack()
    def btnRegister_Click(self):
        self.destroyPackWidget(MainForm.main_Root)
        frmSignUp = SignUp(MainForm.main_Root)
        frmSignUp.pack()
class Login(Frame):
    main_Root=None
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        Login.main_Root=master
        super().__init__(master=master)
        master.title("Login")
        master.geometry("300x250")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="Please enter details below to login",bg="blue")
        self.lblMsg.pack()
        self.username=Label(self, text="Username * ")
        self.username.pack()
        self.username_verify = StringVar()
        self.username_login_entry = Entry(self, textvariable=self.username_verify)
        self.username_login_entry.pack()
        self.password=Label(self, text="Password * ")
        self.password.pack()
        self.password_verify = StringVar()
        self.password_login_entry = Entry(self, textvariable=self.password_verify, show='*')
        self.password_login_entry.pack()
        self.btnLogin=Button(self, text="Login", width=10, height=1, command=self.btnLogin_Click)
        self.btnLogin.pack()
    def btnLogin_Click(self):
        username1 = self.username_login_entry.get()
        password1 = self.password_login_entry.get()
         messagebox.showinfo("Failure", self.username1+":"+password1)
        list_of_files = os.listdir()
        if username1 in list_of_files:
            file1 = open(username1, "r")
            verify = file1.read().splitlines()
            if password1 in verify:
                messagebox.showinfo("Sucess","Login Sucessful")
                self.destroyPackWidget(Login.main_Root)
                frmQuestion = QuestionDigonosis(Login.main_Root)
                frmQuestion.pack()
            else:
                messagebox.showinfo("Failure", "Login Details are wrong try again")
        else:
            messagebox.showinfo("Failure", "User not found try from another user\n or sign up for new user")
.ass SignUp(Frame):
    main_Root=None
    def destroyPackWidget(self,parent):
        for e in parent.pack_slaves():
            e.destroy()
    def __init__(self, master=None):
        SignUp.main_Root=master
        master.title("Register")
        super().__init__(master=master)
        master.title("Register")
        master.geometry("300x250")
        self.createWidget()
```

```
    def createWidget(self):
        self.lblMsg=Label(self, text="Please enter details below", bg="blue")
        self.lblMsg.pack()
        self.username_lable = Label(self, text="Username * ")
        self.username_lable.pack()
        self.username = StringVar()
        self.username_entry = Entry(self, textvariable=self.username)
        self.username_entry.pack()

        self.password_lable = Label(self, text="Password * ")
        self.password_lable.pack()
        self.password = StringVar()
        self.password_entry = Entry(self, textvariable=self.password, show='*')
        self.password_entry.pack()
        self.btnRegister=Button(self, text="Register", width=10, height=1, bg="blue", command=self.register_user)
        self.btnRegister.pack()
    def register_user(self):
         print(self.username.get())
         print("Hello")


        file = open(self.username_entry.get(), "w")
        file.write(self.username_entry.get() + "\n")
        file.write(self.password_entry.get())
        file.close()
        self.destroyPackWidget(SignUp.main_Root)
        self.lblSucess=Label(root, text="Registration Success", fg="green", font=("calibri", 11))
        self.lblSucess.pack()
        self.btnSucess=Button(root, text="Click Here to proceed", command=self.btnSucess_Click)
        self.btnSucess.pack()
    def btnSucess_Click(self):

        self.destroyPackWidget(SignUp.main_Root)
        frmQuestion = QuestionDigonosis(SignUp.main_Root)

        frmQuestion.pack()

root = Tk()

frmMainForm=MainForm(root)
frmMainForm.pack()
root.mainloop()
```

Health Care Chatbot Console

```
######## A Healthcare Domain Chatbot to simulate the predictions of a General Physician ########
######## A pragmatic Approach for Diagnosis #############

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')


# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
y = training_dataset.iloc[:, -1].values

# Dimensionality Reduction for removing redundancies
dimensionality_reduction = training_dataset.groupby(training_dataset['prognosis']).max()

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols     = training_dataset.columns
cols     = cols[:-1]
```

```
# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree

# Method to simulate the working of a Chatbot by extracting and formulating questions
def execute_bot():

    print("Please reply with yes/Yes or no/No for the following symptoms")
    def print_disease(node):
        #print(node)
        node = node[0]
        #print(len(node))
        val  = node.nonzero()
        #print(val)
        disease = labelencoder.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
        ]
        #print("def tree({}):".format(", ".join(feature_names)))
        symptoms_present = []
        def recurse(node, depth):
            indent = "  " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                print(name + " ?")
                ans = input()
                ans = ans.lower()
                if ans == 'yes':
                    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 )
                print()
                red_cols = dimensionality_reduction.columns
                symptoms_given = red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
                print("symptoms present  " + str(list(symptoms_present)))
                print()
                print("symptoms given "  +  str(list(symptoms_given)) )
                print()
                confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
                print("confidence level is " + str(confidence_level))
                print()
                print('The model suggests:')
                print()
                row = doctors[doctors['disease'] == present_disease[0]]
                print('Consult ', str(row['name'].values))
                print()
                print('Visit ', str(row['link'].values))
                #print(present_disease[0])


        recurse(0, 1)

    tree_to_code(classifier,cols)
```

```
# This section of code to be run after scraping the data

doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])


diseases = dimensionality_reduction.index
diseases = pd.DataFrame(diseases)

doctors = pd.DataFrame()
doctors['name'] = np.nan
doctors['link'] = np.nan
doctors['disease'] = np.nan

doctors['disease'] = diseases['prognosis']


doctors['name'] = doc_dataset['Name']
doctors['link'] = doc_dataset['Description']

record = doctors[doctors['disease'] == 'AIDS']
record['name']
record['link']



# Execute the bot and see it in Action
execute_bot()
```

Bibliography

Books

Learn PYTHON the HARD WAY (Third Edition)

Introduction to Machine Learning

Machine Learning with Python Cookbook

Website

www.we3schools.com

www.stackoverflow.com

www.it-ebooks.com