



A PROJECT REPORT
on

“ROBUSNESS”

(24x7 Virtual Doctor)

Department of Computer Science & Engineering

IKG PUNJAB TECHNICAL UNIVERSITY

MAIN CAMPUS ,KAPURTHALA

Submitted by:

Kushal (1818559)

Bhuvan Sharma (1818547)

Project Mentor:

Er. Anupriya

DECLARATION

We declare that this written submission addresses our ideas in our own words, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We comprehend that any infringement of the above will be cause for disciplinary activity by the Institute and can likewise inspire punitive activity from the sources which have subsequently not been as expected referred to or from whom appropriate consent has not been taken when required.

Kushal (1818559)

Bhuvan Sharma (1818547)

ABSTRACT

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls.

An ML model has to be created wherein we could give any text input and on the basis of training data it must analyse the symptoms. A Supervised Logistic Regression machine learning algorithm can be implemented to train the model with data sets containing various diseases CSV files. The goal is to compare outputs of various models and suggest the best model that can be used for symptoms in real world inputs. Data set contains CSV file having symptoms of different diseases compiled together. The logistic regression algorithm in ML allows us to process the data efficiently. The goal here is to model the underlying structure or distribution of the data in order to learn more from the training set.

The main aim of project “**ROBUSNESS (24x7 VIRTUAL DOCTOR)**” is support online 24 x 7 to every person of society as it answers deep as well as general questions. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

ACKNOWLEDGEMENT

No undertaking is at any point total without the direction of those master who have effectively exchanged this past previously and subsequently become expert of it and therefore, our chief. Thus, we might want to take this chance to take each one of those people how have helped us in picturing this venture.

We are very grateful to our Head of the Department **Dr. Raman Kumar (Assistant Professor)** for extending his help directly and indirectly through various channels in our project work.

We would take this opportunity to thank our Project Mentor **Er. Anupriya Kaushal (Assistant Professor)** for their guidance in selecting this project and also for providing timely assistant to our query and guidance of this project.

We extend our sincerity appreciation to our Class Teacher **Dr. Pooja Sharma (Assistant Professor)** for providing us the opportunity to implement and guidance of this project.

We are truly appreciative to every one of our Teachers of **CSE DEPT. IKG PTU MAIN CAMPUS, KAPURTHALA** for their important inside and tip during the planning of the task. Their commitments have been important from multiple points of view that we think that it's hard to recognize of them person.

THANKING YOU

TABLE OF CONTENTS

S.NO.	TOPIC	PAGE
	Declaration	II
	Abstract	III
	Acknowledgement	IV
	Table of contents	V
	List of table	VII
	List of figures	VIII
	List of Symbols, Abbreviations, Nomenclature used	IX
1.	INTRODUCTION	1-2
	1.1 Introduction	1
	1.2 Purpose and scope	1
	1.3 Problem Statement	2
	1.4 Future Scope	2
2.	APPROACH USED	<u>3-4</u>
	2.1 Project Timeline	<u>3</u>
	2.2 Selection of approach	<u>4</u>
	2.3 Application of selected approach	<u>4</u>
3.	Project Design	5-8
	3.1 Block Diagram	<u>5</u>
	3.2 Data Flow Diagram	<u>5</u>
	3.3 Use Case Diagram	<u>7</u>

	3.4 Sequence Diagram	<u>8</u>
4.	Implementation	9-21
	4.1 Project Implementation Technology	9
	4.1.1 Hardware Requirement	9
	4.1.2 Software Requirement	9
	4.2 Experimental Setup	9
	4.3 Coding	10-20
	4.4 Testing	21
5.	Snapshot of Result	21-27
6.	Advantage of Model	28
7.	Conclusion	29
8.	Appendices	30
9.	List of references	31

LIST OF TABLE

Table no.	Name of Table	Page number
1	Project Timeline	3

LIST OF FIGURES

FIGURE No.	NAME OF FIGURE	PAGE NO.
1.1	Block Diagram	5
1.2	Data Flow Diagram	6
1.3	Use Case Diagram	7
1.4	Sequence Diagram	8

List of Symbols, Abbreviations, Nomenclature used

S.no.	Symbols, Abbreviations, Nomenclature used	Original word
1.	ASIC chips digital sign processors (DSPs)	Application-specific integrated circuit
2.	FPGAs	Field programmable gate arrays
3.	DSPs	Digital sign processors
4.	DFD	Data Flow Diagram
5.	SVMs	Support Vector Machine

1.INTRODUCTION

1.1 INTRODUCTION

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls. Since chat bots are basically virtual robot they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break

A creating number of clinics, nursing homes, and surprisingly private focuses, by and by use online Chat bots for human administrations on their locales. These bots associate with potential patients visiting the site, making a difference them find subject matter experts, booking their arrangements, and getting them admittance to the right treatment.

A ML model must be made wherein we could give any content info and based on preparing information it should investigate the manifestations. A Supervised Logistic Regression AI calculation can be executed to prepare the model with informational collections containing different sicknesses CSV records. The objective is to look at yields of different models and recommend all that model that can be utilized for side effects in real world inputs. Informational index contains CSV document having all sicknesses aggregated together.

The calculated relapse calculation in ML permits us to handle the information effectively. The objective here is to display the fundamental construction or dispersion of the information to gain more from the preparation set. Regardless, the usage of computerized reasoning in an industry where people's resides could be in question, actually begins doubts in people. It raises issues about whether the assignment referenced previously should be allocated to human staff. This VIRTUAL DOCTOR will assist clinics with giving medical care support online 24 x 7, it answers profound just as broad inquiries. It additionally assists with producing leads and consequently conveys the data of prompts deals. By posing the inquiries in arrangement it helps patients by managing what precisely he/she is searching for.

1.2 Purpose and Scope:

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls. Since chat bots are basically virtual robot they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break.

1.3 PROBLEM STATEMENT:

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls. Since chat bots are basically virtual robot they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break.

A ML model must be made wherein we could give any content info and based on preparing information it should investigate the manifestations. A Supervised Logistic Regression AI calculation can be executed to prepare the model with informational collections containing different sicknesses CSV records. The objective is to look at yields of different models and recommend all that model that can be utilized for side effects in real world inputs. Informational index contains CSV document having all sicknesses aggregated together.

The calculated relapse calculation in ML permits us to handle the information effectively. The objective here is to display the fundamental construction or dispersion of the information to gain more from the preparation set. Regardless, the usage of computerized reasoning in an industry where people's resides could be in question, actually begins doubts in people. It raises issues about whether the assignment referenced previously should be allocated to human staff. This VIRTUAL DOCTOR will assist clinics with giving medical care support online 24 x 7, it answers profound just as broad inquiries. It additionally assists with producing leads and consequently conveys the data of prompts deals. By posing the inquiries in arrangement it helps patients by managing what precisely he/she is searching for.

1.4 FUTURE SCOPE

VIRTUAL DOCTOR is a thing of the future which is yet to uncover its potential. Machine learning has changed the way companies were communicating with their customers. With new platforms to build various types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold.

2. Approach Used:

The principle motivation behind the plan is to construct the language hole between the client and wellbeing suppliers by giving quick answers to the Questions asked by the client. The present individuals are almost certain dependent on the web however they are not worried about their own wellbeing. They try not to go to medical clinic for little issues which may turn into a significant illness in future. Building up question answer discussions is turning into a straightforward method to answer those inquiries instead of perusing the rundown of conceivably pertinent reports from the web. A considerable lot of the current frameworks have a few limits, for example, there is no moment reaction given to the patients they need to trust that specialists will recognize for quite a while. A portion of the cycles may charge an add up to perform live visit or communication correspondence with specialists on the web. The point of this framework is to imitate an individual's conversation.

2.1 PROJECT TIMELINE:

	10/03	14/03	20/03	25/03	1/04	8/04	16/04	21/04	26/04
PROJECT INITIALISATION	-----								
PROJECT PLANNING		-----							
REQUIREMENT GATHERING		-----	-----						
UI DESIGN			-----	-----					
CODING					-----	-----	-----		
TESTING							-----	---	
IMPROVEMENTS + FINAL TESTING								-----	-----

2.2 SELECTION OF APPROACH:

The VIRTUAL DOCTOR should be written in Python, GUI links and a simple, accessible network API. The system must provide a capacity for parallel operation and system design should not introduce scalability issues with regard to the number of surface computers, tablets or displays connected at any one time. The end system should also allow for seamless recovery, without data loss, from individual device failure.

A Supervised Logistic Regression AI calculation can be executed to prepare the model with informational collections containing different sicknesses CSV records. The objective is to look at yields of different models and recommend all that model that can be utilized for side effects in real world inputs. Informational index contains CSV document having all sicknesses aggregated together.

2.2.1 Dataset Details:

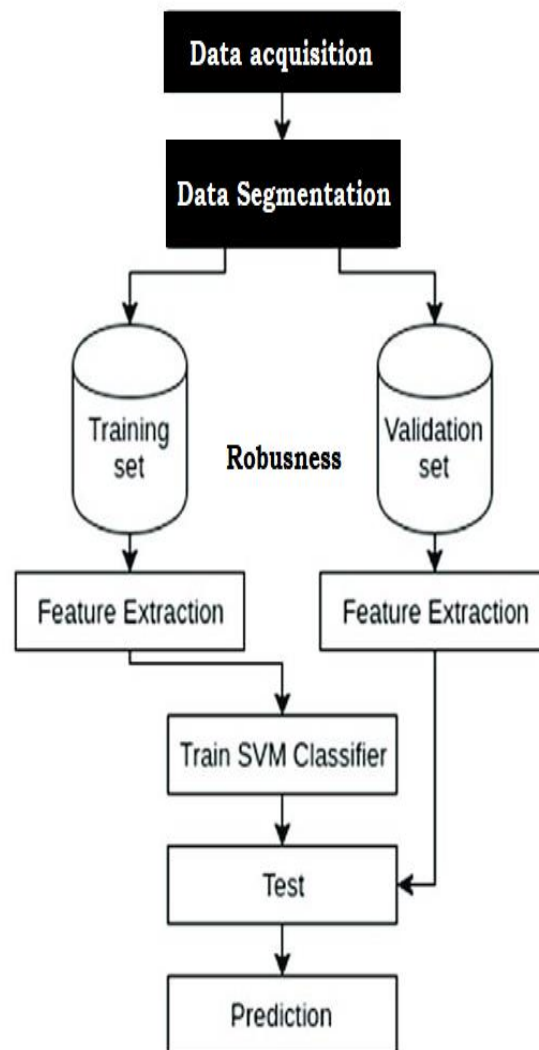
Dataset contains description of different types of diseases. There are different sets of different types of diseases. These sets consists of descriptions of a single disease with different doctors, hospitals, etc. A dataset has been created by recording sequences from over 133 number of diseases and doctors and hospitals.

2.3 Application of selected approach

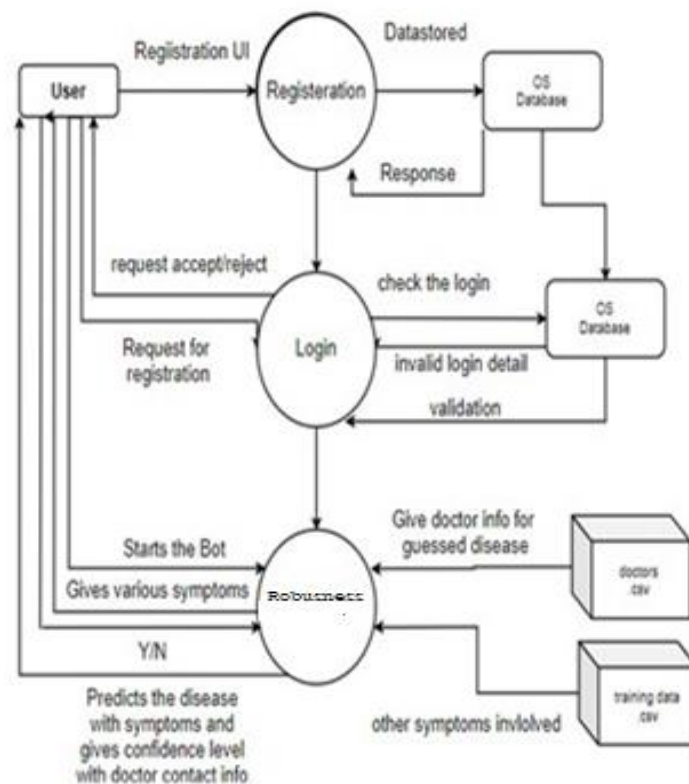
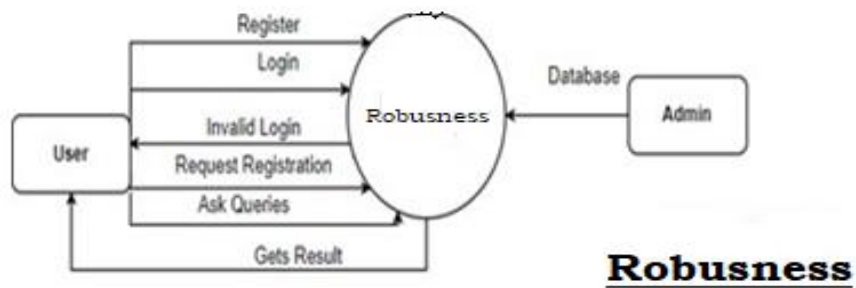
The VIRTUAL DOCTOR should be written in Python, GUI links and a simple, accessible network API. The system must provide a capacity for parallel operation and system design should not introduce scalability issues with regard to the number of surface computers, tablets or displays connected at any one time. The end system should also allow for seamless recovery, without data loss, from individual device failure. There must be a strong audit chain with all system actions logged. While interfaces are worth noting that this system is likely to conform to what is available. With that in mind, the most adaptable and portable technologies should be used for the implementation. The system has criticality in so far as it is a live system. If the system is down, then customers must not notice, or notice that the system recovers quickly (seconds). The system must be reliable enough to run, crash and glitch free more or less indefinitely, or facilitate error recovery strong enough such that glitches are never revealed to its end-users.

3. PROJECT DESIGN:

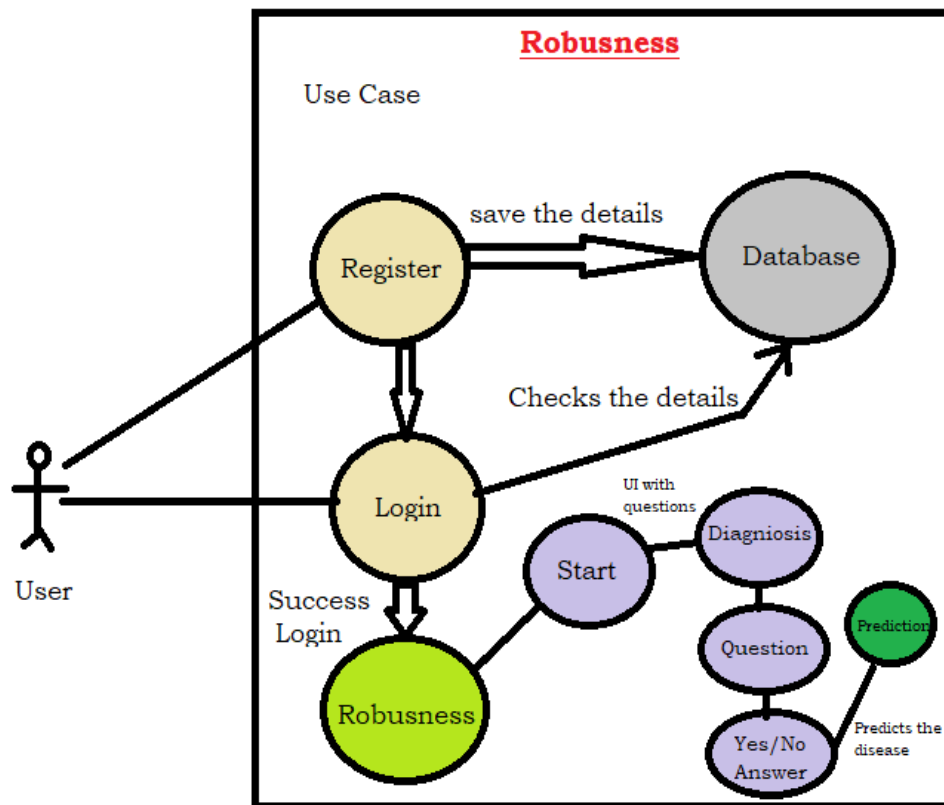
3.1 Block Diagram:



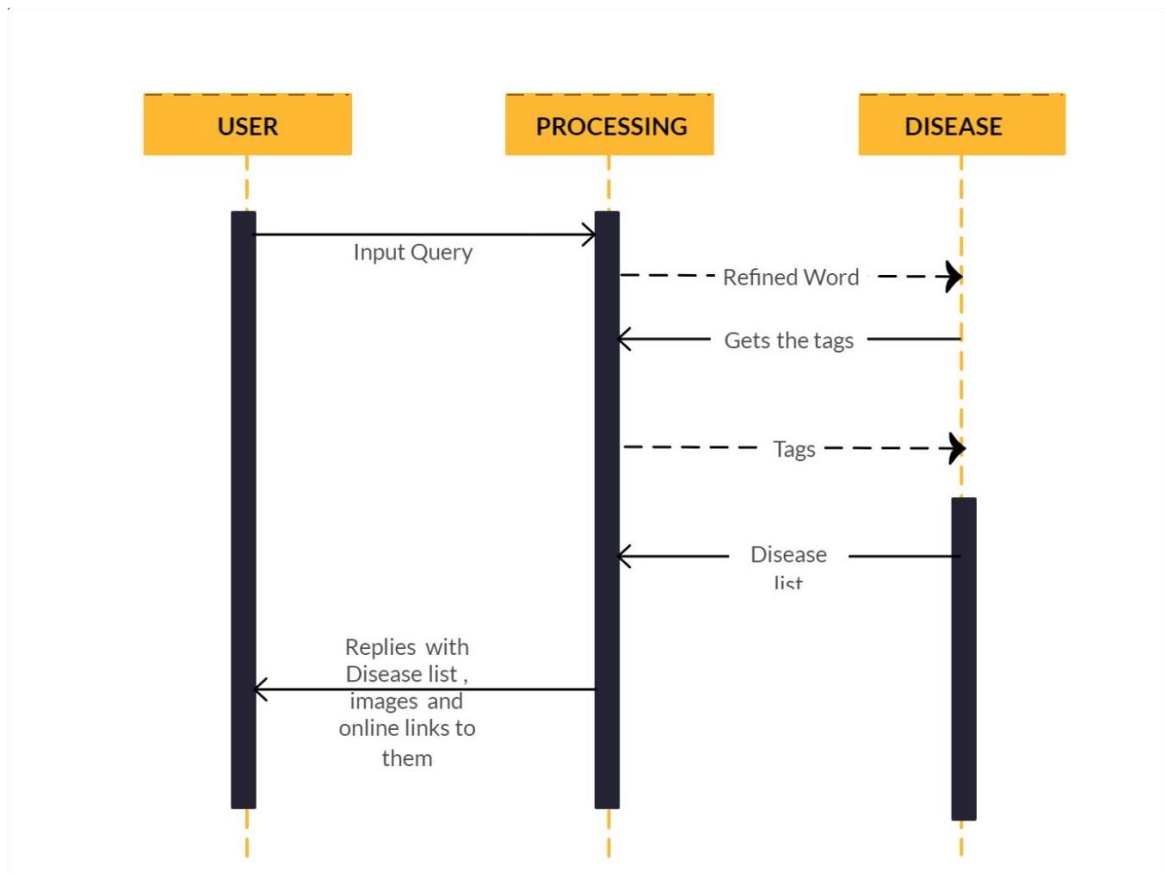
3.2 Data Flow Diagram:



3.3 USE CASE DIAGRAM



3.4 SEQUENCE DIAGRAM



4.Implementation:

4.1 Project Implementation Technology:

In machine learning, **support-vector machines (SVMs, also support-vector networks)** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high- dimensional feature spaces.

4.1.1 Hardware Requirement:

In recent years, a great variety of hardware solutions for real-time TSR has been proposed. These include conventional (general purpose) computers, custom ASIC (application-specific integrated circuit) chips, field programmable gate arrays (FPGAs), digital signal processors (DSPs) and also graphic processing units

4.1.2 Software Requirements:

In a software-based solution running on a Linux or window system with a 2.4-GHz intel i7 9th generation CPU is presented.

4.2 Experimental Setup:

The main purpose of the scheme is to build the language gap between the user and health providers by giving immediate replies to the Questions asked by the user. Today's people are more likely addicted to the internet but they are not concerned about their personal health. They avoid going to hospital for small problems which may become a major disease in future. Establishing question answer forums is becoming a simple way to answer those queries rather than browsing through the list of potentially relevant documents from the web. Many of the existing systems have some limitations such as there is no instant response given to the patients they have to wait for experts to acknowledge for a long time. Some of the processes may charge an amount to perform live chat or telephony communication with doctors online. The aim of this system is to replicate a person's discussion.

4.3 CODING

Importing the libraries

```
from tkinter import *
from tkinter import messagebox
import os
import webbrowser
import numpy as np
import pandas as pd
```

#Creating class for Managing Hyperlinks

```
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 Robusness 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 = 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])
        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()]
        strData="symptoms present: " + str(list(symptoms_present))
        QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
        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)
        strData="confidence level is: " + str(confidence_level)
        QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
        strData='The model suggests:'
        QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
        row = doctors[doctors['disease'] == present_disease[0]]
        strData='Consult ' + str(row['name'].values)
        QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
        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))
        yield strData

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

def execute_bot():
    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']
```

#Creating class QuestionDigonosis for the main page creation where user will answer the questions asked by the model to get accurate results

```
class QuestionDigonosis(Frame):
    objIter=None
    objRef=None

    def __init__(self, master=None):
        master.title("Question")
        master.state("z")
        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.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)
self.btnStart = Button(self, text="Start",width=12,bg="bisque",
command=self.btnStart_Click)
self.btnStart.grid(row=27, column=1,columnspan=20,sticky="e")

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__()

```



```

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")

```

#Creating class MainForm for the main page creation where user can go to Home page , Register or Login for contacting virtual doctor or exit from the app window

```

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("300x300")
        master.title("Account Login")
        self.createWidget()
    def createWidget(self):
        self.lblMsg=Label(self, text="ROBUSNESS", bg="turquoise2", width="300",
height="2", font=("Bradley Hand ITC", 20))
        self.lblMsg.pack()
        self.btnHome=Button(self, text="Home", bg="lightcoral", height="1",
width="10",font=("Goudy Stout", 20), command = self.lblHome_Click)
        self.btnHome.pack()
        self.btnLogin=Button(self, text="Login",bg="gold2", height="1",
width="10",font=("Goudy Stout", 20), command = self.lblLogin_Click)
        self.btnLogin.pack()
        self.btnRegister=Button(self, text="Register",bg="khaki1", height="1",
width="10",font=("Goudy Stout", 20), command = self.btnRegister_Click)

```

```

self.btnRegister.pack()

self.btnExit=Button(self, text="Exit",bg="chocolate1", height="1",
width="10",font=("Goudy Stout", 20), command = root.destroy)
self.btnExit.pack()

self.lblTeam=Label(self, text="Made by:", bg="springgreen3",font=("Bradley
Hand ITC", 20), width = "30", height = "1")
self.lblTeam.pack()
self.lblTeam1=Label(self, text="KUSHAL", bg="palegreen2",font=("Bradley
Hand ITC", 20), width = "30", height = "1")
self.lblTeam1.pack()
self.lblTeam2=Label(self, text="BHUVAN SHARMA",
bg="palegreen2",font=("Bradley Hand ITC", 20), width = "30", height = "1")
self.lblTeam2.pack()


def lblHome_Click(self):
    import webbrowser
    webbrowser.open_new_tab(r"C:\Users\91896\project
6\html\Robusness.html")


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()

```

#Creating class Login for the login page creation where user can Login for contacting virtual doctor by putting username and password details

```
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="plum1")
        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()

        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")

```

#Creating class Register for the register page creation where user can register for contacting virtual doctor by putting username and password details

```

class SignUp(Frame):
    main_Root=None
    print("SignUp Class")
    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="plum1")
        self.lblMsg.pack()
        self.username_label = Label(self, text="Username * ")
        self.username_label.pack()
        self.username = StringVar()
        self.username_entry = Entry(self, textvariable=self.username)
        self.username_entry.pack()

        self.password_label = Label(self, text="Password * ")
        self.password_label.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):
    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()

```

#Packing all classes for complete GUI

```

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

```

4.4 TESTING

Without a well-thought testing effort, the project will undoubtedly fail overall and will impact the entire operational performance of the solution. With a poorly tested solution, the support and maintenance cost will escalate exponentially, and the reliability of the solution will be poor.

Therefore, project managers need to realize that the testing effort is a necessity, not merely as an ad hoc task that is the last hurdle before deployment. The project manager should pay specific attention to developing a complete testing plan and schedule. At this stage, the project manager should have realized that this effort would have to be accommodated within the project budget, as many of the testing resources will be designing, testing, and validating the solution throughout the entire project life cycle—and this consumes work-hours and resources.

The testing effort begins at the initial project phase (i.e. preparing test plans) and continues throughout until the closure phase.

5. Snapshot of Result:

Main Page



ROBUSNESS

YOUR 24*7 VIRTUAL DOCTOR


STAY-AT-HOME


[LEARN MORE](#)



 Stay connected

 Talk about

 Stay on top

 Carry on doing



MENTAL HEALTH

During COVID-19

[LEARN MORE](#)

RELAX

Focus all of your attention on the here and now: noticing the sights, sounds, and smells around you and what you're feeling in your body. Continue to breath slowly in and out—gently bringing your mind back to your body and breath every time it drifts—until you feel more calm.

[LEARN MORE](#)



CONTACT US

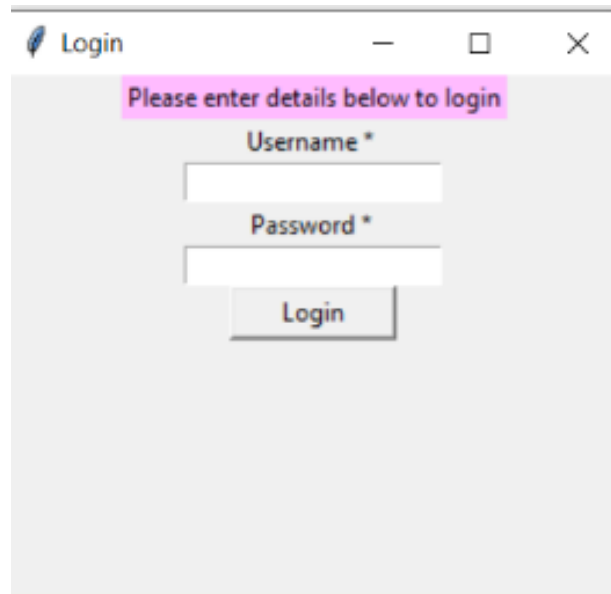
Kushal
Bhuvan Sharma



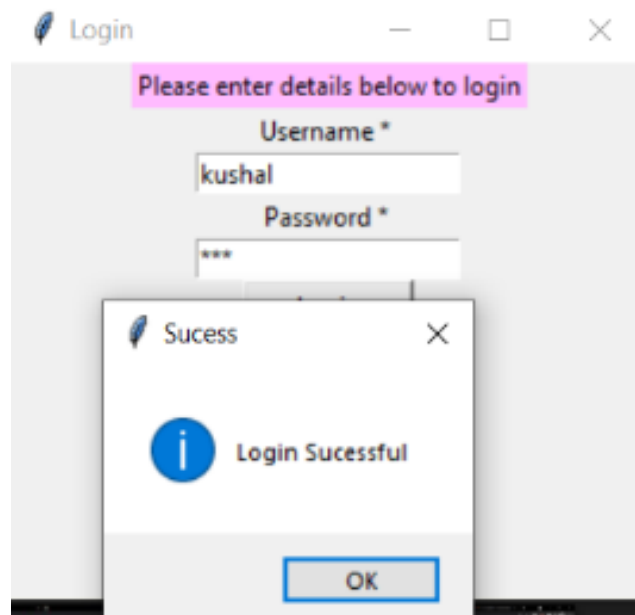
91 8968037593

91 9518471797

Login Page



A screenshot of a Python Tkinter window titled "Login". The window has a light gray background and a pink header bar with the text "Please enter details below to login". Below the header, there are two input fields: "Username *" and "Password *". The "Username *" field contains the text "kushal". Below the input fields is a "Login" button. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.



A screenshot of the same "Login" window, but with a success message displayed. The "Username *" field contains "kushal" and the "Password *" field contains "***". A small dialog box titled "Sucess" (sic) is overlaid on the main window. The dialog box has a blue information icon and the text "Login Sucessful". Below the text is an "OK" button. The dialog box has a close button (X) in the top right corner.

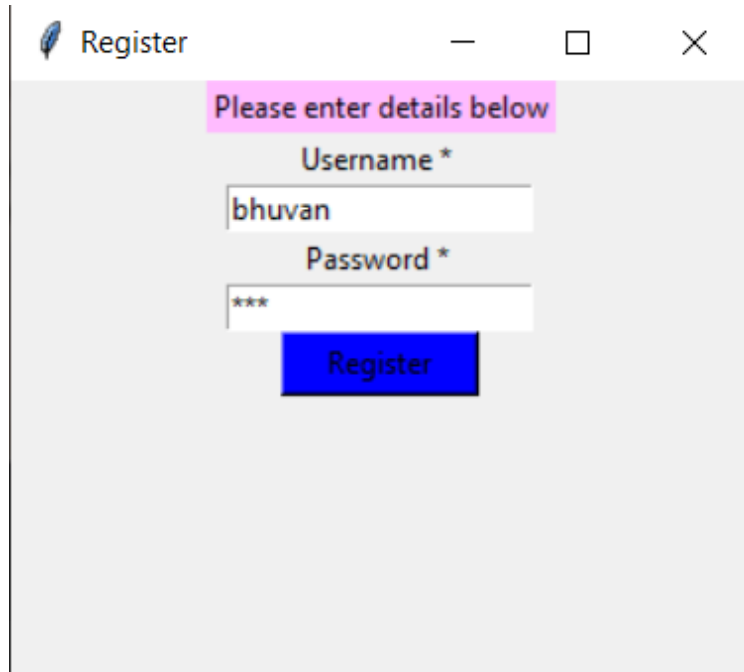
Main Page of Virtual Doctor

Question		
Digonosis		
Start		Yes
Clear		No

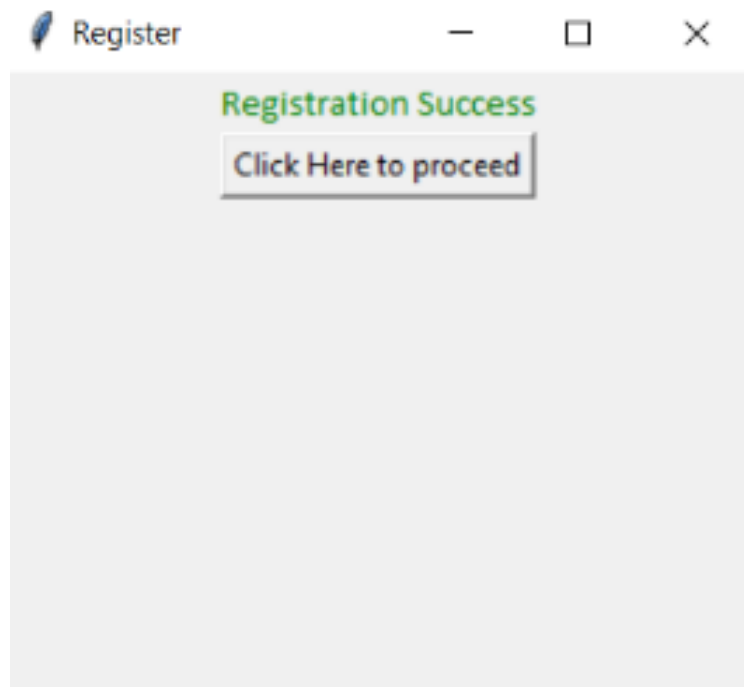
Question	weight_loss ?	
Digonosis	Please Click on Yes or No for the Above symptoms in Question	
Start		Yes
Clear		No

Question	weight_loss ?	
Digonosis	You may have :['Jaundice'] symptoms present: ['weight_loss'] symptoms given: ['itching', 'vomiting', 'fatigue', 'weight_loss', 'high_fever', 'yellowish_skin', 'dark_urine', 'abdominal_pain'] confidence level is: 0.125 The model suggests: Consult ['Dr. Sugeeta Mutreja'] Visit https://www.practo.com/delhi/doctor/sugeeta-mutreja-dietitian-nutritionist?specialization=Ayurveda&practice_id=1010860	
Start		Yes
Clear		No

Registration Page

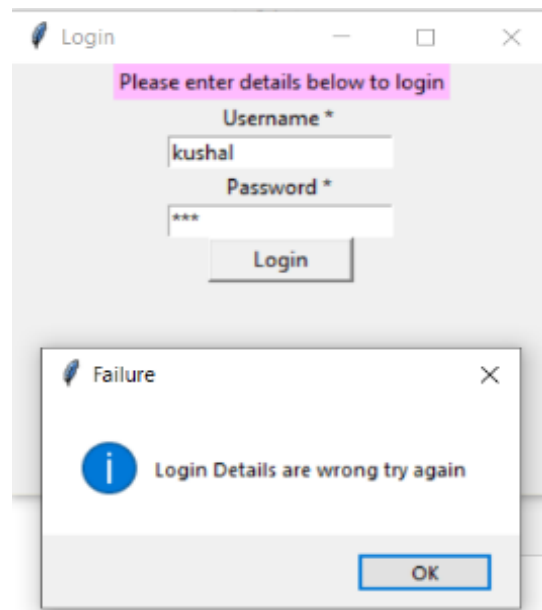


The screenshot shows a web browser window with the title "Register". The page content includes a pink instruction box that says "Please enter details below". Below this, there are two input fields: "Username *" with the text "bhuvan" and "Password *" with three asterisks "***". A blue "Register" button is positioned below the password field.



The screenshot shows the same web browser window after a successful registration. The page content now displays a green "Registration Success" message. Below the message is a button labeled "Click Here to proceed".

Error of wrong login details



6. ADVANTAGES OF MODEL

1.Omni-capable

The VIRTUAL DOCTOR converses seamlessly across multiple digital channels and retains data and context for a seamless experience. In best cases, even passing that information to a live agent if needed.

2.Free to Explore

Our app can reach, consume, and process vast amounts of data– both structured and unstructured–to surface insights from any source - to gather relevant data to solve customer issues quickly.

3.Autonomous Reasoning

The app can perform complex reasoning without human intervention. For example, a great Service VIRTUAL DOCTOR should be able to infer solutions based on relevant case histories.

4.Pre-Trained

The app is pre-trained to understand brand-specific or industry-specific knowledge and terms. Even better, it's pre-configured to resolve common customer requests of a particular industry.

5.Register/Log-in

To access this app and individual needs to register and then use the registration ID to log in to access the features.

6.User Interface

A user friendly interface which is engaging and easy to access.

7.CONCLUSION

Thus, we can conclude that this system giving the accurate result. As we are using large dataset which will ensures the better performance. Thus we build up a system which is useful for people to detect the disease by typing symptoms.

VIRTUAL DOCTOR is a thing of the future which is yet to uncover its potential. Machine learning has changed the way companies were communicating with their customers. With new platforms to build various types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold.

8.APPENDICES

APPENDIX 1: SUPPORT VECTOR MACHINE

SVM's are supervised machine learning algorithms that were introduced in 1992. They become popular because of their success in handwritten digit recognition. Experimentally it was proved that SVM's have low error rates.

SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering algorithm, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

APPENDIX 2: SUPERVISED LOGISTIC REGRESSION

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

9.LIST OF REFERENCES

- <https://en.wikipedia.org/wiki/Chatbot> <https://en.wikipedia.org/wiki/Disease>
- <https://www.youtube.com/playlist?list=PLQVvva0QuDdc2k5dwtDTyT9aCja0on8j>
- <https://www.kaggle.com/tags/healthcare> (Dataset)