HEALTH CARE CHAT BOT

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

BUCHINGARI VASU [RA2011003010412]

BALA CHANDRA K[RA2011003010435]

SHAIK HUSSAIN [RA2011003010439]

Under the guidance of

DR.M ARUNA

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini project report titled **"HEALTH CARE CHAT BOT"** is the bona fide work of **BUCHINGARI VASU (RA2011003010412), BALA CHANDRA KARNA(RA2011003010435), SHAIK HUSSAIN AHAMED(RA2011003010439)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

DR. ARUNA M
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

ABSTRACT

An AI-powered conversational agent that can interact with patients and offer them individualised healthcare services, advice, and information is the goal of the healthcare chatbot project. By giving users access to healthcare services through a practical and user- friendly interface, the chatbot hopes to improve the patient experience with healthcare.The healthcare chatbot uses machine learning and natural language processing to comprehend patient questions and deliver precise, pertinent information.It can help people make appointments, give medical advice, and respond to their questions about healthcare.Byallowingpatientstoreceivehealthcareservicesremotely, easingtheburdenonhealthcare personnel, and enhancing the general calibre of healthcare services, the healthcare chatbot project has the potential to completely transform the healthcare sector.Patients can receive seamless healthcare services without any delays because to the chatbot's integration with current healthcare systems.Overall, thehealthcarechatbotprojecthasthepotentialtoincreasepatientexperiences,lower healthcare expenditures, and improve patient outcomes.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A computer programme created for the healthcare sector's chat bot initiative simulates talks with real consumers. The chat bot intends to help patients and healthcare professionals by giving medical advice, responding to inquiries about symptoms and treatments, directing users to resources for healthcare, and helping with appointment scheduling.The need for rapid and easy access to medical information as well as the rising demand for healthcareservices have increased the significance of thehealth chat bot project.Users can communicate with the healthcare system through chat bots, which provide an easy-to-use and accessible method of communication.The health care chat bot project makes use of a number of technologies, including artificial intelligence,machine learning,and natural language processing,to comprehendand reply to userrequests.Byanalysinguserinputsand deliveringpertinentinformation,thechatbotcan provide precise and prompt responses.Overall, the health care chat bot project is a useful tool for the healthcare sector, offering a low-cost and effective way to provide consumers and healthcare providers with medical advice and support.

CHAPTER 2

LITERATURE SURVEY

In recent years, the use of chatbots in the healthcare industry has increased. A number of research have been carried out to determine how useful chatbots are in various healthcare settings.

Title of Paper 1: Improving LMS Experience with AIML Base and Retrieval Base R-based chatbot Language

Brief Description: The usage of several techniques, including N-gram, Stemming, TF-IDF, and cosine similarity, is covered in detail in the study. This paper explains how to apply these techniques to produce an optimised outcome rapidly. It details various inquiries and how a chatbot might respond to them. For convenience, they also specified the functional architecture of databases and offered test cases for algorithms.The essence of chatbot quality

Title of Paper 2:Artificial Intelligence Based Personal Assistant,

Brief Description:

Here In this work, a chatbot that acts as a user's personal healthcare assistant is built for healthcare- related purposes. A user can interact with a medical assistant via a dialogue interface. It offers features like ailment diagnosis based on symptoms reported by the user, definitions of medical terms, advice from doctors, scheduling of treatments, and tracking and monitoring of the user's health metrics. Both offline and online performance were looked at overall. To assess the effectiveness and quality of the system, they also ran a number of tests.

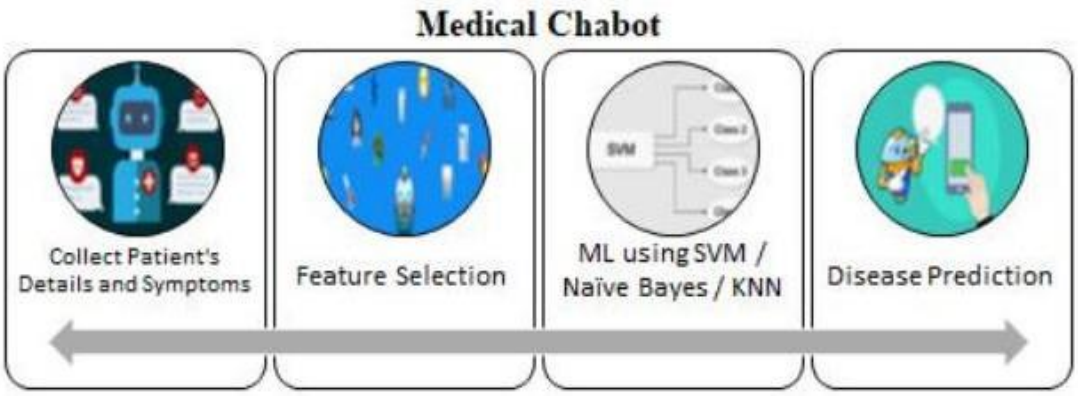
To create an eHealth environment for patients' convenience, numerous studies on this topic have been done in the recent years.

Title of Paper 3: Human-to-Machine Conversation Modelling

Brief Description: In this study, hardware and software were combined to develop a chatbot system. This chatbot is Bluetooth-enabled and can move.They controlled chatbots through voice communication.A database that is kept on the Raspberry Pi will be searched using the user's query. This chatbot only functions with particular illnesses like the common cold, typhoid, malaria, etc.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN



This is the proposed architecture for our AI-Powered Health Chatbot,and we will give further explain for every component in the next upcoming papers. First of all, the end user interacts with the chatbot through a client platform. It's important to be user friendly and give an excellent UX.

Each time, the user is having a request, it's routed to the NLP Engine using the appropriate API's, in the NLP Engine, and with NLU, the chatbot understands the request and format the data into understandable form that can be understanded by the Core Engine. Once the Core Engine receives the formatted data, it searches using Deep Learning Algorithms, the appropriate response and send it back to the NLP Engine.

CHAPTER 4

METHODOLOGY

The following steps could be used in the technique for creating a health care chatbot:

A health care chatbot's target audience must be determined before any other steps can be taken to develop it. The chatbot might be created to meet the needs of patients, physicians,nurses, or other healthcare providers. Designing the chatbot's functionalities would be aided by an understanding of the needs of the target audience.

Identify the chatbot's objectives: The next stage is to identify the objectives of the chatbot.
The objectives should match the requirements of the target audience. For instance, a chatbot created just for patients would be intended to answer general health-related concerns, arrange appointments, and send out reminders.

Gather information and create a database: A substantial amount of data that can aid the chatbot in comprehending and responding to user inquiries is necessary in order to construct one. The information might be found in patient records, clinical data, and medical literature.
By using this information to teach the chatbot, a knowledge base can be created.

Select the chatbot development platform and tools: There are a variety of chatbot development platforms and tools available. The objectives and needs of the chatbot will determine the platform and technologies to use.

Create the dialogue flow for the chatbot: This will establish how the chatbot will communicate with the user. The discussion flow has to be planned to guarantee that the user's inquiries are answered truthfully and effectively.

Create the natural language processing (NLP) system for the chatbot: The chatbot's capacity to comprehend natural language is a crucial feature. The NLP system needs to be able to handle many dialects, accents,and languages.

Test and improve the chatbot: After the chatbot is created, it should be put to the test to make sure it works as intended. Any problems or flaws with the chatbot can be found by conducting user testing. The chatbot can be enhanced and developed based on the input.

Deploy the chatbot: After it has been tested and improved, the chatbot can be made available to the intended audience. To make sure the chatbot is operating properly and answering users'
questions, regular checks should be made.

CHAPTER 5

CODING AND TESTING

README.md chat_bot.py requirements.txt symptom_Description.csv

```python
import re
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('Data/Training.csv')
testing= pd.read_csv('Data/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)
```

README.md chat_bot.py requirements.txt symptom_Description.csv

```python
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())
model=SVC()
model.fit(x_train,y_train)
print("for svm: ")
print(model.score(x_test,y_test))
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 = {}
```

README.md chat_bot.py requirements.txt symptom_Description.csv

```python
for index, symptom in enumerate(x):
      symptoms_dict[symptom] = index
1 usage
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.")


1 usage
def getDescription():
    global description_list
    with open('MasterData/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)
1 usage
def getSeverityDict():
    global severityDictionary
    with open('MasterData/symptom_severity.csv') as csv_file:

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

README.md chat_bot.py requirements.txt symptom_Description.csv

```python
        try:
            for row in csv_reader:
                _diction={row[0]:int(row[1])}
                severityDictionary.update(_diction)
        except:
            pass


1 usage
def getprecautionDict():
    global precautionDictionary
    with open('MasterData/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)


1 usage
def getInfo():
    print("-----------------------------------HealthCare ChatBot-----------------------------------")
    print("\nYour Name? \t\t\t\t",end="->")
    name=input("")
    print("Hello, ",name)

1 usage
def check_pattern(dis_list,inp):
```

README.md | chat_bot.py | requirements.txt | symptom_Description.csv

```python
def check_pattern(dis_list,inp):
    pred_list=[]
    inp=inp.replace(' ','_')
    patt = f"{inp}"
    regexp = re.compile(patt)
    pred_list=[item for item in dis_list if regexp.search(item)]
    if(len(pred_list)>0):
        return 1,pred_list
    else:
        return 0,[]
1 usage
def sec_predict(symptoms_exp):
    df = pd.read_csv('Data/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 = {symptom: index for index, symptom in enumerate(X)}
    input_vector = np.zeros(len(symptoms_dict))
    for item in symptoms_exp:
      input_vector[[symptoms_dict[item]]] = 1

    return rf_clf.predict([input_vector])


1 usage
def print_disease(node):
    node = node[0]
    val  = node.nonzero()
    disease = le.inverse_transform(val[0])
    return list(map(lambda x:x.strip(),list(disease)))

1 usage
def tree_to_code(tree, feature_names):
    tree_ = tree.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 = []

    while True:

        print("\nEnter the symptom you are experiencing  \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:
```

README.md | chat_bot.py | requirements.txt | symptom_Description.csv

```python
                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 valid input.")
    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)
```

README.md | chat_bot.py | requirements.txt | symptom_Description.csv

```python
                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]):
```

getDescription() › with open('MasterData/symptom_D... › for row in csv_reader

README.md | chat_bot.py | requirements.txt | symptom_Description.csv

```python
            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]])
            precution_list=precautionDictionary[present_disease[0]]
            print("Take following measures : ")
            for i,j in enumerate(precution_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)
print("----------------------------------------------------------------------------------------")
```

Code:

```
import re
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('Data/Training.csv')
testing= pd.read_csv('Data/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())


model=SVC()
model.fit(x_train,y_train)
print("for svm: ")
print(model.score(x_test,y_test))

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 = {}
```

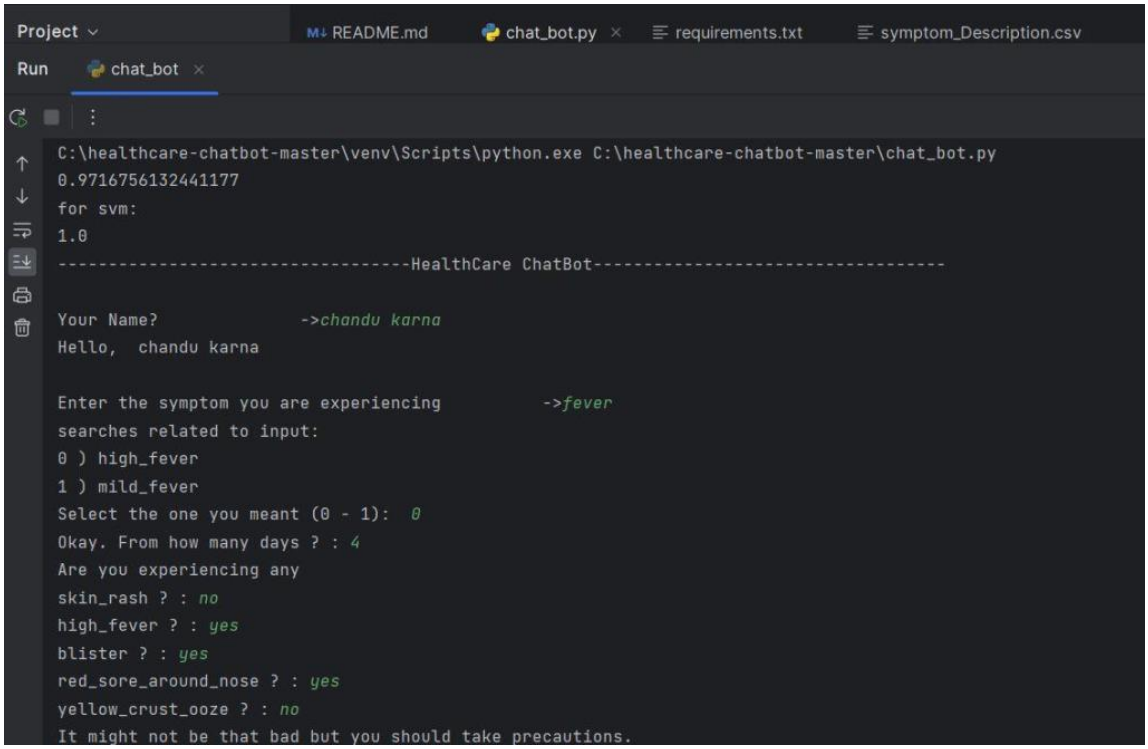
```
for index, symptom in enumerate(x):
      symptoms_dict[symptom] = index
def calc_condition(exp,days):
    sum=0
    for item in exp:
        sum=sum+severityDictionary[item]
    if((sum*days)/(len(exp)+1)>13):
        print("You should take the consultation from doctor. ")
    else:
        print("It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


def getDescription():
    global description_list
    with open('MasterData/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('MasterData/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('MasterData/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():
    print("-----------------------------------HealthCare ChatBot-----------------------------------")
    print("\nYour Name? \t\t\t\t",end="->")
    name=input("")
    print("Hello, ",name)

def check_pattern(dis_list,inp):
    pred_list=[]
    inp=inp.replace(' ','_')
    patt = f"{inp}"
    regexp = re.compile(patt)
    pred_list=[item for item in dis_list if regexp.search(item)]
    if(len(pred_list)>0):
        return 1,pred_list
    else:
        return 0,[]
def sec_predict(symptoms_exp):
    df = pd.read_csv('Data/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 = {symptom: index for index, symptom in enumerate(X)}
    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):
    node = node[0]
    val  = node.nonzero()
    disease = le.inverse_transform(val[0])
    return list(map(lambda x:x.strip(),list(disease)))

def tree_to_code(tree, feature_names):
    tree_ = tree.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 = []

    while True:

        print("\nEnter the symptom you are experiencing  \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 valid input.")
    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]])
        precution_list=precautionDictionary[present_disease[0]]
        print("Take following measures : ")
        for  i,j in enumerate(precution_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)
print("----------------------------------------------------------------------------------------")
```

CHAPTER 6

SCREENSHOTS AND RESULTS:



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion,

The suggested solution helps hospitals and medical centres by enabling patients to ask questions about their health freely via voice or text requests.

The machine gathers data from medical API diagnostic and speaks out with disease treatments. Comparatively speaking, SVM classification accuracy is higher than KNN and Naive Bayes algorithms.

In comparison to other machine learning algorithms, SVM generates 92.33% accuracy and is thus utilised to accurately predict the disease. It also saves time and space.

Future Improvement

Medical chatbots can be expanded and utilised extensively with other medical systems that allow for prediction by utilising the benefits of the SVM algorithm. It can also be used to plan doctor visits and notify patients of upcoming appointments or checkups.

it can also be extended to collect patients' feedback and this will help medical organizations to improve their processes.

REFERENCES

[1] Mohammed Javed, P. Nagabhushan, B.B. Chaudhari, “A Direct Approach for Word and Character Segmentation in Run-Length Compressed Documents with an Application to Word Spotting”, 13th International Conference on Document Analysis and Recognition (ICDAR), 2015.

[2] Naeun Lee, Kirak Kim, Taeseon Yoon, “Implementation of Robot Journalism by Programming Custombot using Tokenization and Custom Tagging”, 2017.

[3] Tao Jiang, Hongzhi Yu, Yangkyi Jam, “Tibetan Word Segmentation Systems based on Conditional Random Fields”, 2011.

[4] Jerome r. Bellagarda, “Parts-Of-Speech tagging by Latent Analogy”, IEEE Journal of Selected Topics in Signal Processing, Vol. 4, No. 6, 2010.

[5] Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, Guohong Fu, “Joint POS Tagging and Dependency Parsing with Transition-based Neural Networks”, 2018.