A Project/Dissertation Review-1 Report

on

Healthcare Chat-bot for Disease Prediction Using Python

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology (CSE)

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. Ravinder Ahuja
Assistant Professor

Submitted By

Aditya Kumar	Akash Vishwakarma
18SCSE1010699	18SCSE1010657
18021011922	18021011881

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

MAY, 2022

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Healthcare chat-bot for disease prediction using python"** in partial fulfilment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of month, **JANUARY-2022 to MAY-2022**, under the supervision of **Mr. Ravinder Ahuja, Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Aditya Kumar, 18SCSE1010699

Akash Vishwakarma, 18SCSE1010657

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr. Ravinder Ahuja, Assistant

Professor)

CERTIFICATE

The Final Project Viva-Voce examination of **Aditya Kumar: 18SCSE1010699, Akash Vishwakarma : 18SCSE1010657** has been held on _________________ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

Table of contents

List of Figures

List of Tables

ABSTRACT

In the face of Covid-19, we have realized how a more technologically advanced health sector would be highly effective. We can finally see through and understand how lack of doctors and nurses can be maneuvered in hospitals, for readily available assistance to patients at any time of the day, and even so to doctors and their hectic routines. Health Recommendation systems are one of the new and growing technologies for assisting patients by understanding their history. These systems can do numerous things like help find recommended hospitals by understanding the similarities in patterns which exist in the health care sector. The healthcare industry is one of the busiest industries. There is a demand for efficiency and speed for modern patients who want easy access to information. Chat bots can revolutionize healthcare. They can boost efficiency as well as increase the accuracy of symptoms collection, identification of diseases, preventive care, post recovery care and general feedback. Chat bots have become a great tool for quick and easy automation. Now the patients don’t have to hold in line for hours before someone looks into their query because a chat bot can do it instantly. These intelligent programs can detect symptoms, manage medications and assist in various health issues. People can be guided rightly for serious illnesses and can also be assisted in scheduling appointments with professionals. Doctors are usually on a tight schedule, therefore, being available for every patient is impossible at times. This is where chat bots come into work. These chat bots provide instant information, especially when every minute is important. Patients can check for symptoms on healthcare chat bot and even measure the severity of the situation.

Chapter -1
INTRODUCTION

A health recommender system is of great importance and boom in today's era. As the name suggests, it's a recommender system designed especially for the health sector-patients-to predict diseases-to formulate and research about medicines which are described to us-to recommend hospitals, dispensaries, clinics-to recommend doctors and specialists, also make appointments. It's a very important tool today, because health is the most vital part if our existence, and which is why with the advent of technology, we might want something easier than standing in queues and feel clueless about our diagnosis or medicine. Here is when we use algorithms and codes to make our lives simpler, at the tap of our fingers. This paper focuses on a novel health recommendation system that predicts diseases based on the symptoms entered. On the basis of the severity of the disease, it gives advices. In the first stage, we process the data using Label Encoder. It will map all the strings in the dataset to numbers so that the complete dataset transforms to an array-list. The classifier will then be applied to the modified dataset. We use Random Forest Classifier on the dataset. Random samples will be collected from the dataset and the decision tree will be constructed for the same to get a prediction result. There will be a vote on each predicted result, and the result with the most votes will be counted as final prediction. We also use Support Vector Classifier as it will provide the best fit that will categorize the data. Once we get the hyperplane, we feed features into the classifier to see the final result and we map out the result via a confusion matrix.

1.1 CHATBOT

Chat bot is a software application designed to stimulate human behaviour as a conversational partner. They require continuous tuning and testing and many even fail the industry standard Turing test. They are used for various purposes like information gathering, customer services and routing requests. Some chatbot applications use sophisticated AI and a natural language processing while others just scan for general keywords and give the responses using phrases from an associated database. Most of the chatbots can be accessed through virtual assistants or website pop-ups.

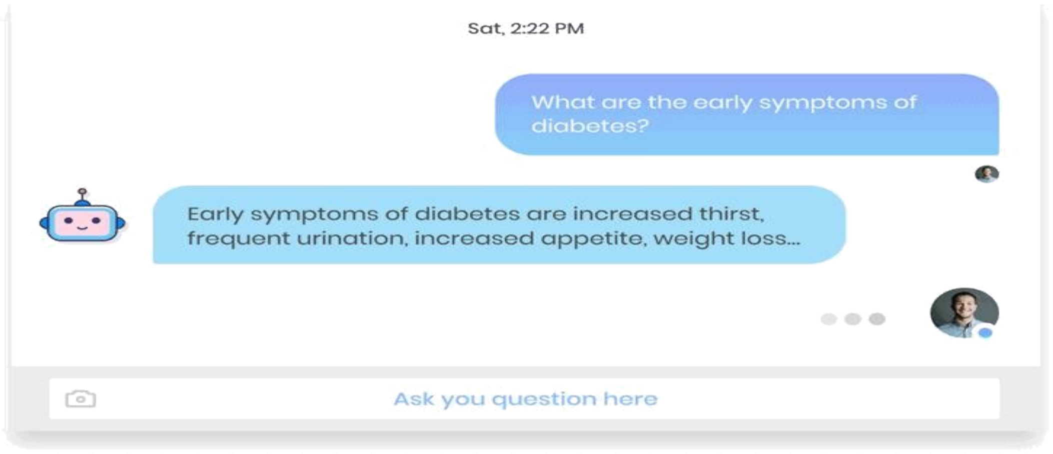
In healthcare industry, chatbots are developed using machine learning algorithms and natural language processing to stimulate a conversation with a user so that real time assistance can be provided to the patients. Healthcare payers are also starting to use these tools to simplify patient care and cutting unnecessary costs. Some chatbots which have complex self-learning algorithms can maintain nearly human like conversations. Chatbots will not only improve care delivery but will also help in cost savings and improve the patient care outcomes. These medical chatbots reduce professional's workload by reducing the unnecessary treatments and procedures, reducing hospital budgets and, also decreasing hospital re-admissions as the symptoms improve. For patients, it is also very beneficial as they have to less time in commuting to the doctor's office. They can have easy access to any doctor and less money is spent on unnecessary treatments. Chatbots reduce hospital wait times, hospital re admissions and consultation times by connecting patients with the right doctors and, also by helping them understand their conditions and treatments without having to visit a doctor. These bots ask questions about the patient's symptoms and give automated responses which provide sufficient history for the doctor which can help the doctors determine which patients to see first and which require a brief consultation. There are three kinds of chatbots used in healthcare. They can be conversational, prescriptive and informative.

1.2 INFORMATIVE CHATBOTS

provide information for users in the form of notifications, pop-ups and stories. They provide automated customer support. For example, if we search for articles about flu symptoms, a chatbot may pop up with all the information like current outbreak in your area and how it can be treated. Some health news websites also use chatbots to access detailed information about a topic. For example, while reading about addiction and withdrawal, a chatbot may pop up with: "Do you need help with alcohol addiction? You can speak with our mental health professionals."

1.3 CONVERSATIONAL CHATBOTS are built to provide responses based on user's intent but they have different levels of maturity - not all of them provide the same depth of information.

For example, level 1 maturity chatbot provides responses that are pre-built to clear questions without having the capacity to follow through with any kind of deviations. A level 2 intelligence chatbot can engage in further questions. The chatbots with higher intelligence levels understand the complete context in a better way and provide more than just pre-built answers. This is possible because these chatbots do not process sentences in isolation and rather look at the conversation as a whole. High intelligence chatbots provide more personal responses which is how the conversations resemble human interactions.

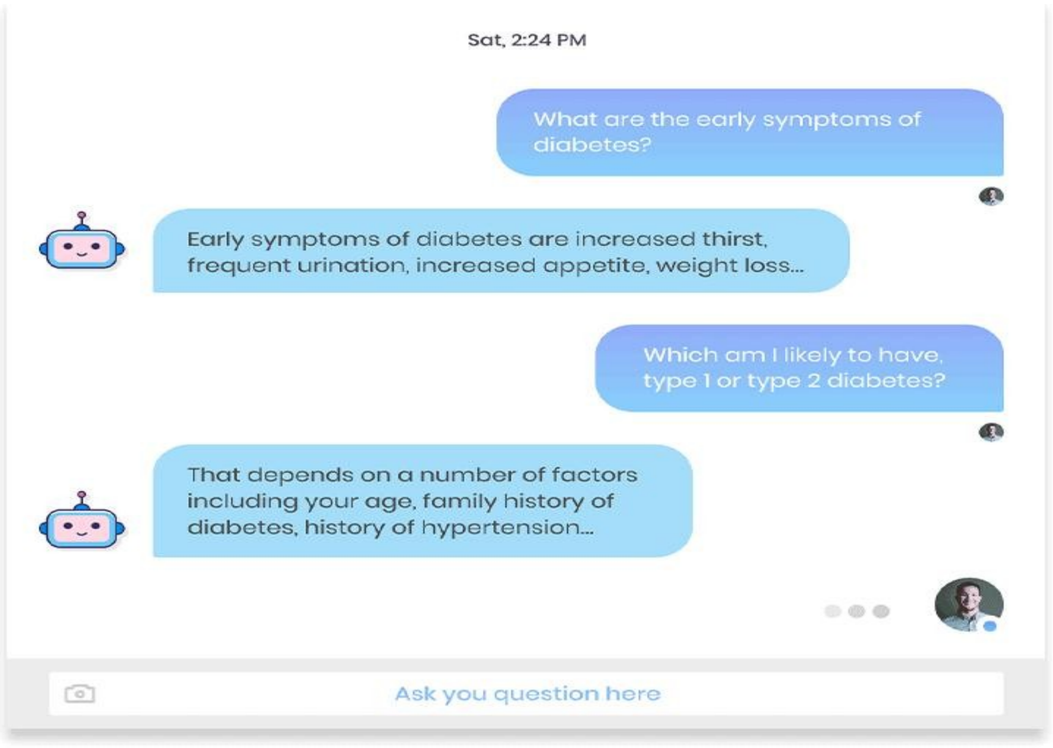


Sat, 2:22 PM

What are the early symptoms of diabetes?

Early symptoms of diabetes are increased thirst, frequent urination, increased appetite, weight loss...

Fig 1: Level-1 Conversational chat-bot



Sat, 2:24 PM

What are the early symptoms of diabetes?

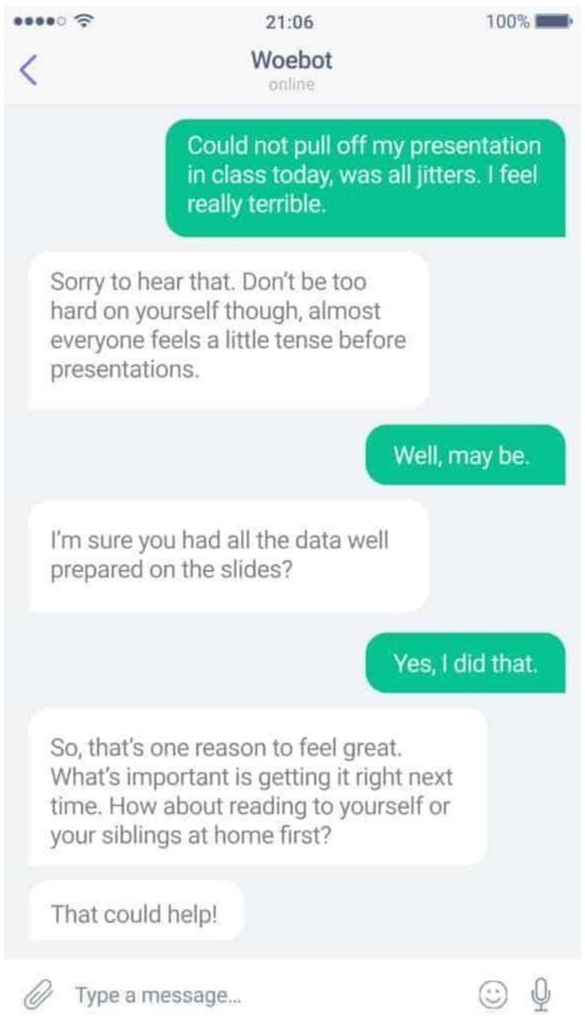
Early symptoms of diabetes are increased thirst, frequent urination, increased appetite, weight loss...

Which am I likely to have, type 1 or type 2 diabetes?

That depends on a number of factors including your age, family history of diabetes, history of hypertension...

Fig 2: Level-2 Conversational chat-bot

1.4 PRESCRIPTIVE CHATBOTS are conversational by design. They don't just provide directions but also offer therapeutic solutions. An example of a prescriptive chatbot is Woebot. It is a chatbot designed by researchers from Stanford University. It provides mental health assistance by making use of cognitive behavioural therapy techniques. People who suffer from anxiety disorders, mood disorders, depression can talk to this chatbot and it helps people treat themselves by reshaping their thought patterns and behaviour.



Could not pull off my presentation in class today, was all jitters. I feel really terrible.

Sorry to hear that. Don't be too hard on yourself though, almost everyone feels a little tense before presentations.

Well, may be.

I'm sure you had all the data well prepared on the slides?

Yes, I did that.

So, that's one reason to feel great. What's important is getting it right next time. How about reading to yourself or your siblings at home first?

That could help!

Fig 3: Perscriptive chat-bot

1.5 APPLICATIONS

ELISA was the first ever chatbot to be used in healthcare in 1966. It imitated a psychotherapist by using pattern matching and response selection. However, it had limited communicational abilities. Today, chatbots offer mental healthcare consultation, diagnosis of symptoms, nutrition facts and tracking and more.

Chatbots provide medical information. Their algorithms are trained on massive data including disease symptoms and available treatments. Public datasets continuously train these chatbots like COVIDx for diagnosis of COVID-19.

Chatbots can be integrated into the medical system to extract information about suitable doctors, clinic working days and available slots. They ask patients about their health issues, find them physicians and dentists, provide them with appointments and also provide the option of scheduling and rescheduling.

Chatbots also help in collecting patient information. They can extract information using questions like name, symptoms, current physician and even insurance details. They then store this information in the medical facility system for patient-doctor communication, symptom tracking and even record keeping.

Chatbots can also handle insurance inquiries.

They also provide mental health assistance for patients with depression, anxiety or post-traumatic stress disorder by delivering cognitive behavioral therapy. They can also help train autistic patients in improving their social and job interview skills.

Since chatbots collect patient's information like name, contact, current doctor and prescription. They can submit a request to the doctor for a final decision and contact the patient whenever the prescription refill is available which allows doctors to process these refills in batches or even automate them in certain cases.

1.6 DECISION TREE CLASSIFIER

Separation can be defined as the task of learning a specific task f clearly each attribute set x on one of the previously defined labels y.

Examples:

Assigning a news clip to one of the categories described earlier.

• Receive spam email messages based on message headers and content

• Classify cells as malignant or malignant according to the results of MRI scans

• Divide galaxies into categories

The Decision Tree is built by asking critical questions about the recording of the database we have. Each time a response is received, the next question is asked until the conclusion about the record class label. A series of possible questions and answers can be arranged in the form of a deciding tree, which is a structure with sections consisting of nodes and directed edges. The tree has three types of nodes:

• root node with no inbound and zero or outbound edges.

• Internal meditation, each with a single incoming edge and two or more outgoing edges.

• Leaf or terminal nodes, each with a single incoming edge and no outgoing edges.

In the cutting tree, each leaf node is assigned a class label. Non-terminal nodes, which include root and other internal nodes, contain attribute test conditions in different records with different characteristics.

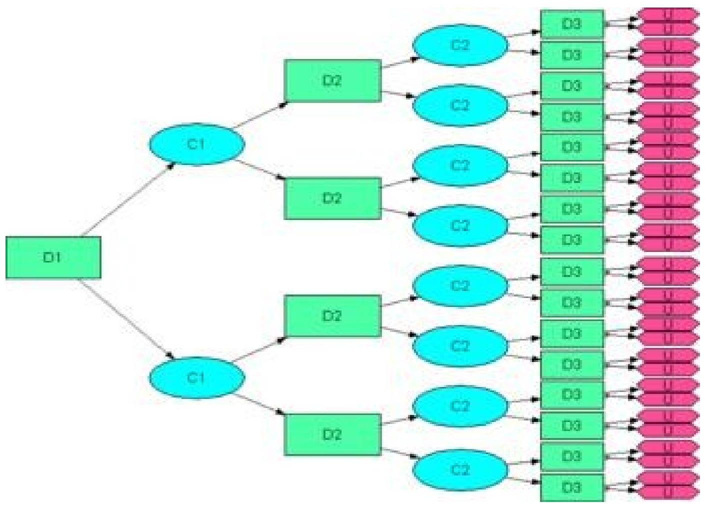


Figure 4: Decision Tree

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY TABLE

Serial number	Title	Author
1.	Chatbots as conversational healthcare services	Mladjan Jovanovic, Marcos Baez, Fabio Casati
2.	HealthAssistantBot : A Personal Health Assistant for the Italian Language	Marco Polignano, Fedelucio Narducci, Andrea Lovine, Cataldo Musto, Marco De Gemmis, Giovanni Semeraro
3.	An Efficient and Privacy-Preserving Disease Risk Prediction Scheme for E-Healthcare	XUE Yang, Rongxing Lu, Jun Shao, Xiaohu Tang, Haomiao Yang
4.	A Medical-History-Based Potential Disease Prediction Algorithm	Wenxing Hong, Ziang Xiong, Nannan Zheng, Yang Weng
5.	Designing Disease Prediction Model Using Machine Learning Approach	Dhiraj Dahiwade, Gajanan Patle, Ektaa Meshram
6.	A Proposed Model for Lifestyle Disease Prediction Using Support Vector Machine	Mrunmayi Patil, Vivian Brian Lobo, Pranav Puranik, Aditi Pavaskar, Adarsh Pai, Rupesh Mishra
7.	Efficient Heart Disease Prediction System Using Decision Tree	Purushottam, Kanak Saxena, Richa Sharma
8.	Chatbot for Healthcare System Using Artificial Intelligence	Lekha Athota, Vinod Kumar Shukla, Nitin Pandey, Ajay Rana
9.	Implementation of Interactive Healthcare Advisor Model Using Chatbot and Visualization	TAE-Ho Hwang, JuHui Lee, Se-Min Hyun, KangYoon Lee
10.	Automatized Medical Chatbot (Medibot)	Prakhar Srivastava, Nishant Singh

2.2 EXISTING METHODS

In an average Health recommender system there are two parts- patients and the end product, which could be a clinical report, or recommendation of doctors, or medicines. Patients are supposed to enter their preferences based on their health record and the data which is collected. This data is represented in a matrix which in turn gives us the value of each patient-product combination. Hence there are two engines- a patient based, and a product based.

Creating a Knowledge Repository:

It creates a repository consisting of information about patients and prepares a data base. In this data base there is background details, medical history, behaviour of the patient. A well informed data repository is very integral as it forms the backbone of any recommender system.

Sentiment Analysis: As the name suggests- it is a learning algorithm for the entire model based on the data we have accumulated.

Recommendation: by the analysis and feedback through the information we have collected, we can now give recommendations.

Patients often cannot share private or underwhelming information with their doctors in person, due to a basic human nature. Often they are not able to get the best advice possible due to geographical barriers, or that they don't get the diagnosis in time, which at time can be fatal. This work offers us a helping hand in all the given problems and more, using the concepts of Natural language Processing, RNN and CNN.

As human beings we communicate in our natural language, instead of machine generated or deciphered numerical and symbols, hence our machines need an algorithm to decipher what we are trying to say and generate reliable and efficient results. Which is where Natural Language Processing(NLP) comes into play. But NLP still falls short due to the abundance of interactions, that approaches such as the one hot vector fall short. There are dimensionality problems or the ordering of words.

2.3 RNN:

1. The recurrent neural network, as the name suggests, works like a feedback loop. It helps with the more dynamic part of the algorithm, such as timely responses or the layers which hold onto information in a sequential context.

2. In this study, they are providing the patient with a pre-diagnosis on the basis of the data they have inputted, recommend them with the domain based physician.

3. CNN here works in contextual, sentence level, textrual learning which improves the training of the model and RNN uses its pattern building through the sentences and dialogues, specificity to give a highly efficient result.
4. TFIDF clustering the further refines the result.

5. With health related recommendation models, there is always one shortcoming- prospect of training with more data and experiments, which in turn gives a better result.

6. This is used to rank the items by its relevance which is recorded in terms of a score. Such as buying a product from amazon-we can sort or filter it through its ratings, CF works the same way.

2.4 Deep Neural Network(DNN):

1. This works on the big data. It filters based on the inputs we provide. The model is made using DNN which stores the high order relation within the diseases.

2. Each Symptom and disease has a different degree of relationship, and the customer's history of diseases and symptoms need to be weighed in against the medical history, on an 'attention network'. The output is then summarized as the patient's condition. DNN is then further used to train the higher order features. Remembering, that lower-order features cannot be neglected, hence 'Factorization Machines' are used to combine the two and give a score, which predicts the disease.

3. It generates a list of plausible diseases, from which the result is chosen. Combining the lower oder and higher order relations is an added feature, which we learn from this paper because it increases the efficiency of the result by not having a tunnel vision. The noise i.e. the diseases that don't affect the result are also removed.

4. Repeating what has been said above, the ever present flaw in each model-training with more data to improve accuracy. This paper also sheds light on the fact that genetics and behavior of the patient is not an entity that can be predicted by algorithms or statistics, hence combining professional knowledge onto the model is also a direction to work on in the future.

CHAPTER 3

PROPOSED METHODOLOGY

A random forest, as its name implies, contains a large number of tropical trees that serve as a single group. Each tree in a random forest spells out a class forecast and a section with a lot of votes becomes a prediction of our model.

The maximum number of unrelated species (trees) that function as a committee will exceed any models in each category.

Low integration between models is key. Just as low-correlated investments (such as stocks and bonds) come together to form a portfolio larger than the sum of its components, unconventional models can produce more accurate forecasts than any other prediction. The reason for this positive effect is that trees protect each other from their individual mistakes (as long as they do not always deviate in the same way). While some trees may be bad, many other trees will be good, so as a group the trees are able to move in the right direction. The requirements for a random forest to do well are therefore:

1. There needs to be a certain signal in our features so that the models built using those features perform better than random guesses.

2. The predictions (therefore errors) made by each tree need to have a low correlation with each other.

3. Random Forest is a supervised learning algorithm. The "forest" is constructive, is a combination of decision-making drugs, often trained in the "integration" method. A common idea of how to pack a bag is that a combination of learning models enhances the overall effect.

4. Another major advantage of the informal forest is that it can be used for both planning and retrospective problems, which is what constitutes most of the current machine learning systems.

The random forest has the same hyper parameters as the decision tree or bag divider. Fortunately, there is no need to combine a decision tree with a bagging divider because you can easily use a random forest partition section. With the random forest, you can also tackle back-to-back tasks by using the algorithm functionality.

The random forest came more random in the model, while the trees grew. Instead of searching for the most important feature when a node is divided, it seeks the best feature among the random subset of features. This results in variations that often lead to better models.

One of the great benefits of the informal forest is its versatility. It can be used for retrofitting and separation functions, and it is also easy to view the equal value it gives you input features.
Random forest is also a very simple algorithm because the automatic hyperparameters it uses usually produce a good predictive effect. Understanding hyperparameters is very good, and also not many of them.

One of the biggest problems with machine learning is overuse, but most of the time this will not happen due to random forest planning. If there are enough trees in the forest, the separator will not exceed the model too much.

The great limitation of the random forest is that a large number of trees can make the algorithm go too slow and not work in real-time prediction. Generally, these algorithms are quick to train, but it is not too late to create predictions once they are trained. More accurate predictions require more trees, leading to a slower model. For many real-world applications, the random forest algorithm is fast enough but there can certainly be situations where working time is important and alternatives can be preferred.

CHAPTER 4

4.1 ARCHITECTURE DIAGRAM

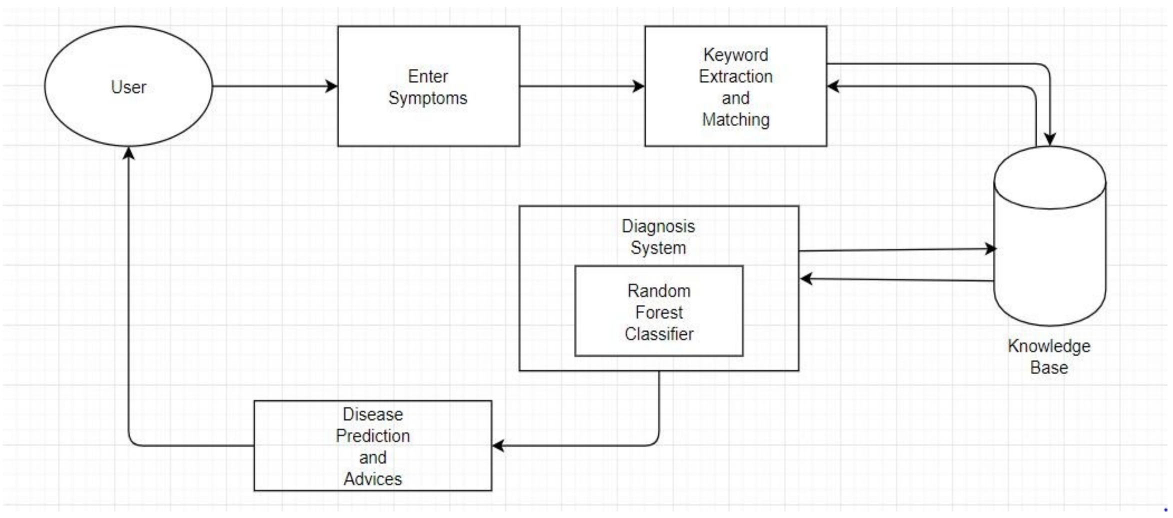


Figure 5: Architecture Diagram

The architecture diagram suggests that the chatbot will ask the users to enter the symptoms one by one. As the user keeps entering symptoms, the algorithm keeps extracting the keywords from the inputs and as it matches the symptoms from the database, individual decision trees are made for each symptom. As all the decision trees are made for each symptom, Random Forest Classifier is applied on the decision trees and the decision tree with the most number of votes is counted as the final prediction. Based on symptoms and their severity, the disease is predicted and information is provided about the disease. Additionally, the chatbot also gives advices like what kind of home measures the patient can take to cure the disease and also if the patient needs to go and refer the doctor if the symptoms are too severe.

4.2 DIFFERENCE BETWEEN DECISION TREES AND RANDOM FORESTS

While a random forest is a collection of decision trees, there are some differences.
If you enter a training database with features and labels on the decision tree, it will create certain rules, which will be used to make predictions.

For example, predicting whether a person will click on an online ad, you may collect ads that someone clicked in the past and other factors that explain their decision. If you place features and labels on the decision tree, it will generate specific rules that help predict whether the ad will be suppressed or not. By comparison, the random forest algorithm randomly selects for visibility and features to build multiple trees for decision making and then results in between.

Another difference is the trees of the "deep" decisions may suffer too much. Most of the time, the random forest prevents this by creating random subsets of features and building smaller trees using those subsets. After that, it covers the subtrees. It is important to note that this does not always work and makes computer calculations slow, depending on how many trees are formed by the random forest.

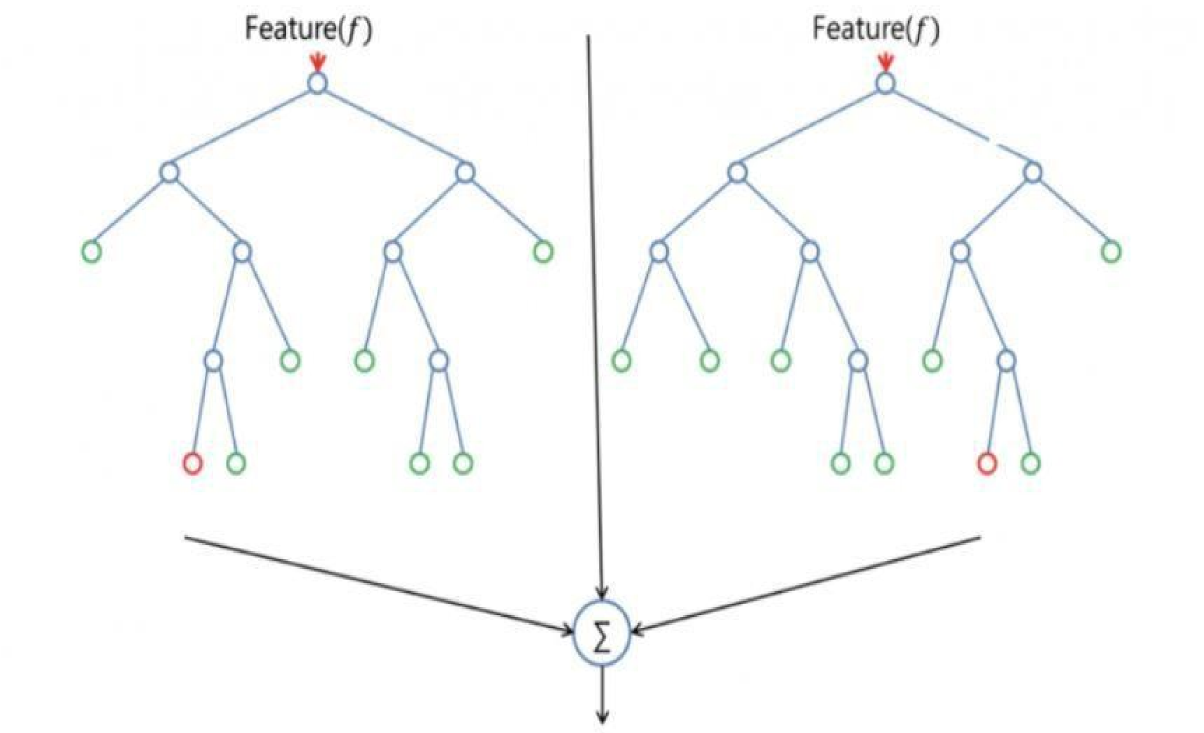


Figure 6: Random Forest Classifier

4.3 IMPORTANT HYPE RPARAMETERS

Random forest hyperparameters may be used to increase the predictability of the model or to speed up the model. Let’s take a look at hyperparameters for sklearns built with random forest operation.

1. Increase the ability to predict

First, there is the n_estimators hyper parameter, which is just the number of trees that the algorithm constructs before taking the high vote or taking the prediction rating. In general, the higher number of trees increases efficiency and makes predictions more stable, but also slows down the calculation.

Another important hyper parameter is max_feature, which is a large number of features in the random forest that looks for node partitions. Sklearn offers many options, all described in the documentation.

The last important hyper parameter is min_sample_leaf. This determines the minimum number of leaves needed to separate an internal node.

2. Increase the speed of the model

The n-jobs hyper parameter tells the engine how many processors are allowed to use. If it has one value, it can only use one processor. A value of "-1" means no limit.

The hyper_state hyper parameter makes model output repeated. The model will always produce the same results when it has the exact value of the random state and when given the same hyperparameters and the same training data.

Finally, there is oob score (also called oob sampling), which is a random way to verify the forest. In this sample, approximately one-third of the data was not used for model training and could be used to evaluate its performance. It is very similar to the leave-one-out-cross-validation method, but there is probably no additional computer load associated with it.

CHAPTER 5

MODULE DESCRIPTION

5.1 The model consists of the following layers:

1. Data Processing: We use LabelEncoder. Mark all strings in the database to numbers so that the complete database can be converted into a list of identical members. The separator will be used on this modified database.

• In machine learning, we are often faced with data sets that contain multiple labels in one or more columns. These labels can be in the form of words or numbers. To make the data understandable or human-readable, training data is often written in words.

• LabelEncoder texts converts labels into numbers to convert them into machine-readable form. Machine learning algorithms can then determine the best way to use those labels. It is an important pre-processing step for systematic databases in supervised learning.

Limitation of label Encoding

Label Encoding converts the data in a machine-readable form, but provides a unique number (starting from 0) for each data class. This can lead to the creation of a very important issue in training data sets. A high-value label can be considered more important than a low-value label.

Test Train Split : We use this function to evaluate the performance in our upcoming algorithm. Our complete dataset will be divided into 2 subsets. One subset will be used to fit the model (the training dataset). The second subset will not train the model. In fact, the input element of the dataset will be provided to the model and the prediction will be made and it will be compared to the expected values (test dataset).

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

In this tutorial, you will discover how to evaluate machine learning models using the train-test split.

- The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly.

  he train-test split is a technique for evaluating the performance of a machine learning algorithm.

  It can be used for classification or regression problems and can be used for any supervised learning algorithm.

  The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

  The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

  This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

  The train-test procedure is appropriate when there is a sufficiently large dataset available.

  The idea of "sufficiently large" is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.

  A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

  Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There will also not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad).

If you have insufficient data, then a suitable alternate model evaluation procedure would be the k-fold cross-validation procedure.

In addition to dataset size, another reason to use the train-test split evaluation procedure is computational efficiency.

Some models are very costly to train, and in that case, repeated evaluation used in other procedures is intractable. An example might be deep neural network models. In this case, the train-test procedure is commonly used.

Alternately, a project may have an efficient model and a vast dataset, although may require an estimate of model performance quickly. Again, the train-test split procedure is approached in this situation.

Samples from the original training dataset are split into the two subsets using random selection. This is to ensure that the train and test datasets are representative of the original dataset.

2. Applying Random Forest Classifier on the Dataset : Random samples will be collected from the dataset. The decision tree for the same will be constructed to get a prediction result. There will be a vote on each predicted result and the result with the most votes will be the final prediction.

   The Random forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees.
   The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

. SVC : We are using SVC in our project because it will provide the best fit that will categorize our data. Once we get our hyperplane, we will feed features into the classifier to see the final result.

The support vector machine is highly preferred by many as it produces remarkable accuracy with low calculation power. The Support Vector Machine, abbreviated as SVM can be used for retrofitting and split operations. However, it is widely used for planning purposes. The purpose of the support vector machine algorithm is to detect a hyperplane in a space equal to N (N - number of elements) that clearly separates the data points.

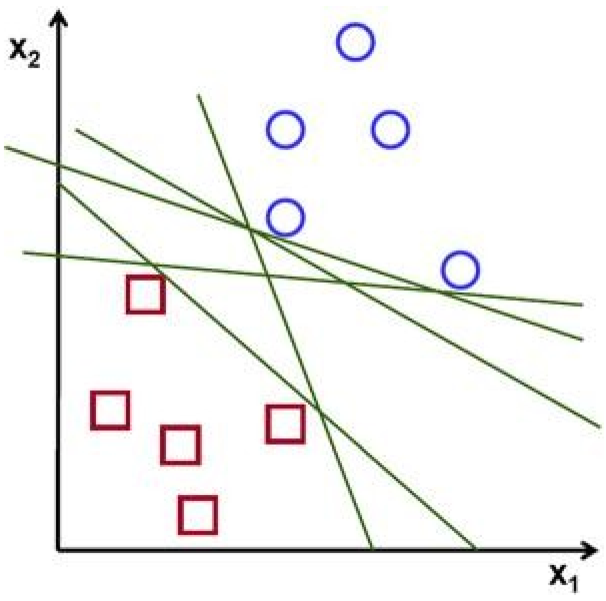


Figure 7: Non-SVC

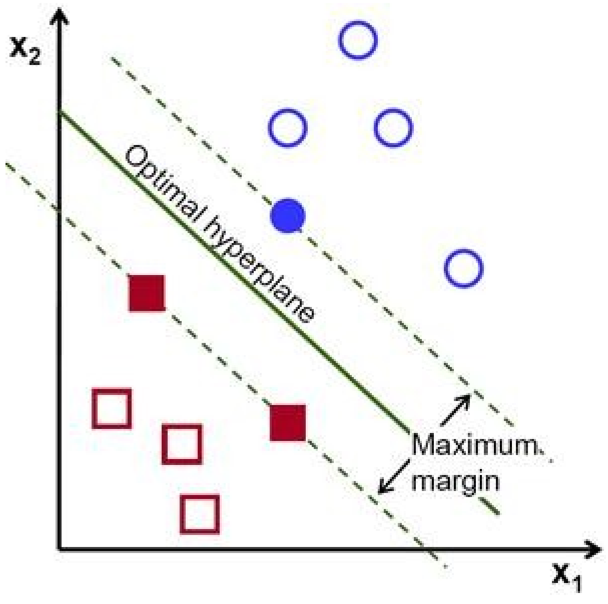


Figure 8: Optimal SVC

5.2 Hyperplanes and Support Vectors

Hyperplanes are decision-making parameters that help to separate data points. Data points falling on either side of a hyperplane can be caused by different classes. Also, the size of the hyperplane depends on the number of factors. If the number of input elements is 2, then the hyperplane is just a line. If the number of input elements is 3, then the hyperplane becomes a two-dimensional plane. It’s hard to imagine when the number of features exceeds 3. Supporting vectors are data points that are very close to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

CHAPTER 6

CODE

```
import pandas as pd

import pyttsx3

from sklearn import preprocessing

from sklearn.tree import DecisionTreeClassifier,_tree

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.svm import SVC

import csv

import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning

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

testing= pd.read_csv('Testing.csv')

cols= training.columns

cols= cols[:-1]

x = training[cols]

y = training['prognosis']

y1= y



reduced_data = training.groupby(training['prognosis']).max()
```

```
        #mapping strings to numbers

        le = preprocessing.LabelEncoder()

        le.fit(y)

        y = le.transform(y)
        x_train,    x_test,    y_train,    y_test    =    train_test_split(x,    y,
test_size=0.33, random_state=42)

        testx                                   = testing[cols]

        testy                                   = testing['prognosis']

        testy                                   = le.transform(testy)




        clf1  = DecisionTreeClassifier()

        clf = clf1.fit(x_train,y_train)

        #

print(clf.score(x_train,y_train))

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

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

        # print (scores)

        print (scores.mean())

        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('symptom_Description.csv') as csv_file:

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

line_count = 0

for row in csv_reader:

_description={row[0]:row[1]}

description_list.update(_description)

def getSeverityDict():

global severityDictionary

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


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

line_count = 0

try:

for row in csv_reader:

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

severityDictionary.update(_diction)

except:

pass




def getprecautionDict():

global precautionDictionary

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

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

line_count = 0

for row in csv_reader:

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

precautionDictionary.update(_prec)




def getInfo():

# name=input("Name:")

print("Your Name \n\t\t\t\t\t\t",end="->")

name=input("")

print("hello ",name)


def check_pattern(dis_list,inp):

import re

pred_list=[]

ptr=0

patt = "^" + inp + "$"

regexp = re.compile(inp)

for item in dis_list:


#         print(f"comparing {inp} to {item}") if regexp.search(item):

pred_list.append(item)

#         return 1,item

if(len(pred_list)>0):
```

```
return 1,pred_list

else:

return ptr,item

def sec_predict(symptoms_exp):

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

X        = df.iloc[:, :-1] y = df['prognosis']

X_train, X_test, y_train, y_test = train_test_split(X, rf_clf =
DecisionTreeClassifier() rf_clf.fit(X_train, y_train)


symptoms_dict = {}


for index, symptom in enumerate(X): symptoms_dict[symptom] =
index


input_vector = np.zeros(len(symptoms_dict))

for item in symptoms_exp:

input_vector[[symptoms_dict[item]]] = 1


return rf_clf.predict([input_vector])


def print_disease(node):

#print(node)

node = node[0]

#print(len(node))

val = node.nonzero()
# print(val)

disease = le.inverse_transform(val[0])

return disease

def tree_to_code(tree, feature_names):

tree_ = tree.tree_

#         print(tree_) feature_name = [

feature_names[i]    if    i    !=    _tree.TREE_UNDEFINED    else
"undefined!" for i in tree_.feature

]



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

symptoms_present = []




#         conf_inp=int() while True:



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

disease_input = input("")

conf,cnf_dis=check_pattern(chk_dis,disease_input)

if conf==1:

print("searches related to input: ")

for num,it in enumerate(cnf_dis):

print(num,")",it)

if num!=0:

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

conf_inp = int(input(""))
```

```
else:

conf_inp=0



disease_input=cnf_dis[conf_inp]

break

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

#          conf_inp = input("")

#          if(conf_inp=="yes"):

#          break
else:

print("Enter valid symptom.")


while True:

try:

num_days=int(input("Okay. From how many days ? : "))

break

except:

print("Enter number of days.")

def recurse(node, depth):

indent = "  " * depth

if tree_.feature[node] != _tree.TREE_UNDEFINED:

name = feature_name[node]

threshold = tree_.threshold[node]


if name == disease_input:

val = 1

else:
```

```
val = 0

if  val <= threshold:

recurse(tree_.children_left[node], depth + 1)

else:

symptoms_present.append(name)

recurse(tree_.children_right[node], depth + 1)

else:

present_disease = print_disease(tree_.value[node])

#        print( "You may have " + present_disease ) red_cols =
reduced_data.columns

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

#        dis_list=list(symptoms_present)

#        if len(dis_list)!=0:

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

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

print("Are you experiencing any ")

symptoms_exp=[]

for syms in list(symptoms_given):

inp=""

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

while True:

inp=input("")

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

break

else:

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

if(inp=="yes"):
```

```
symptoms_exp.append(syms)


second_prediction=sec_predict(symptoms_exp)

# print(second_prediction)

calc_condition(symptoms_exp,num_days)

if(present_disease[0]==second_prediction[0]):

print("You may have ", present_disease[0])


print(description_list[present_disease[0]])


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

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


else:

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

print(description_list[present_disease[0]])

print(description_list[second_prediction[0]])


# print(description_list[present_disease[0]])

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)

CHAPTER 7

OUTPUTS AND GRAPHS

Screenshots

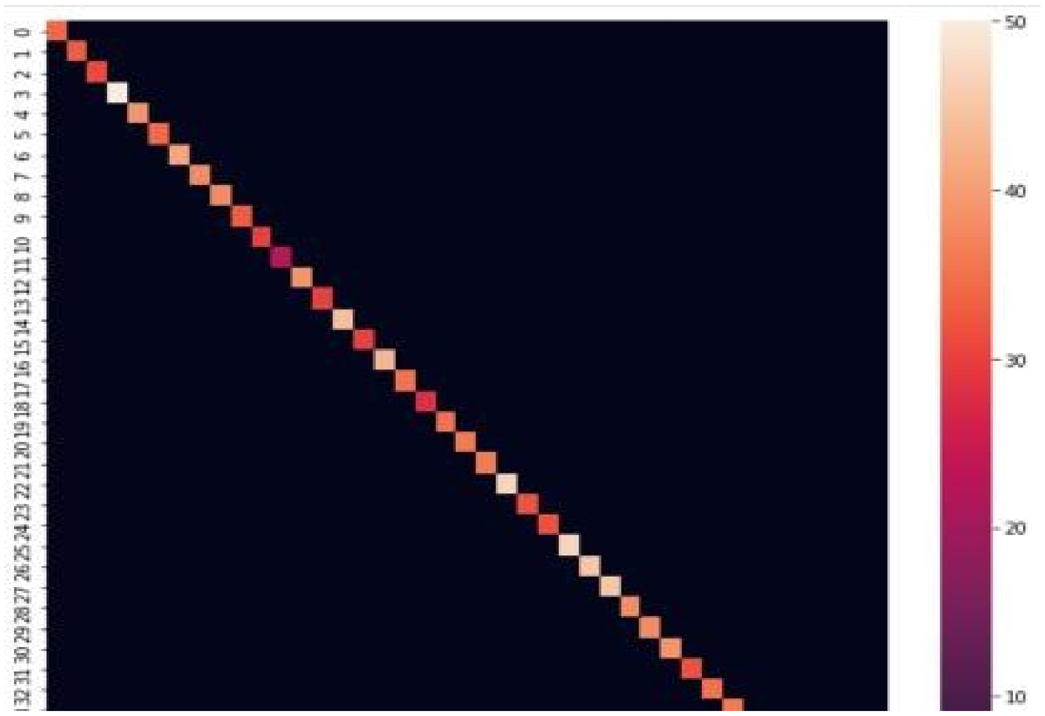
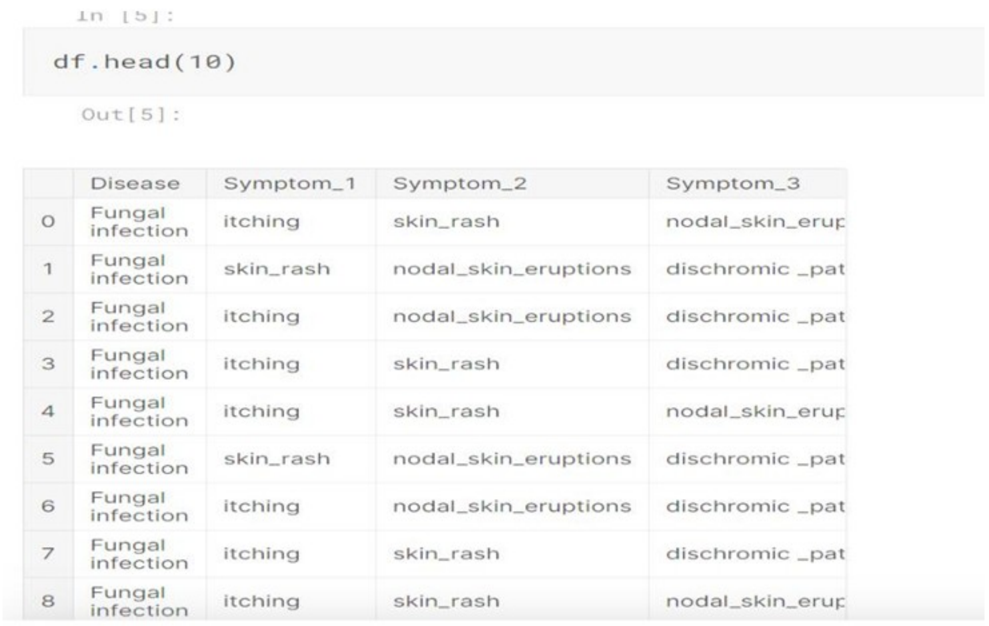


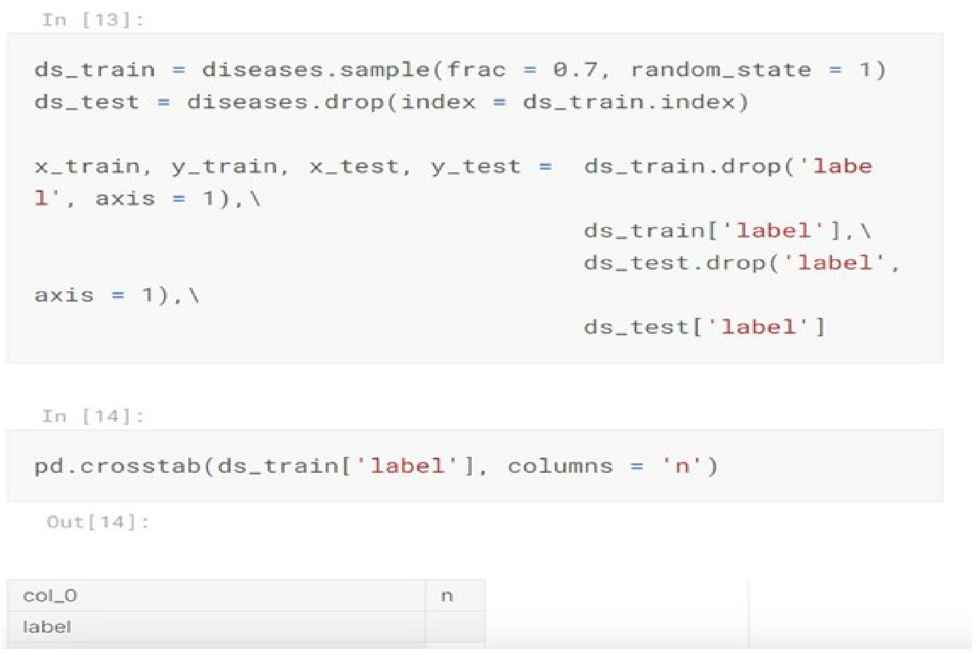
Figure 9: Confusion Matrix



```
In [5]:
df.head(10)
Out[5]:
```

	Disease	Symptom_1	Symptom_2	Symptom_3
0	Fungal infection	itching	skin_rash	nodal_skin_erup
1	Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _pat
2	Fungal infection	itching	nodal_skin_eruptions	dischromic _pat
3	Fungal infection	itching	skin_rash	dischromic _pat
4	Fungal infection	itching	skin_rash	nodal_skin_erup
5	Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _pat
6	Fungal infection	itching	nodal_skin_eruptions	dischromic _pat
7	Fungal infection	itching	skin_rash	dischromic _pat
8	Fungal infection	itching	skin_rash	nodal_skin_erup

Figure 10: Disease Name with Symptoms



```
In [13]:
```

```
ds_train = diseases.sample(frac = 0.7, random_state = 1)
ds_test = diseases.drop(index = ds_train.index)

x_train, y_train, x_test, y_test =  ds_train.drop('labe
l', axis = 1),\
                                    ds_train['label'],\
                                    ds_test.drop('label',
axis = 1),\
                                    ds_test['label']
```

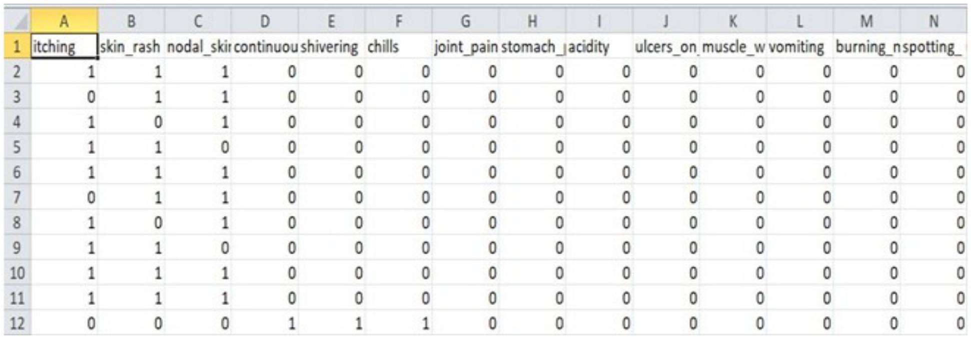
```
In [14]:
```

```
pd.crosstab(ds_train['label'], columns = 'n')
```

```
Out[14]:
```

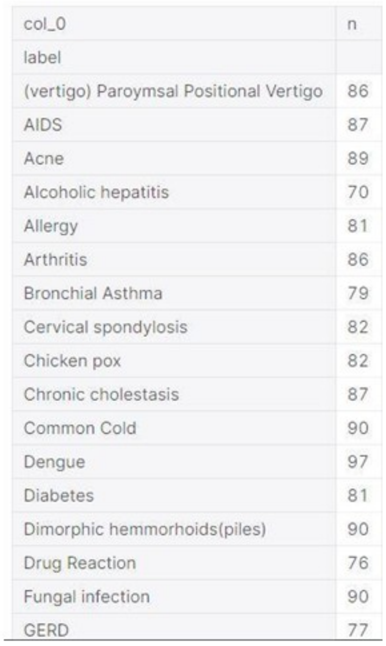
col_0	n
label	

Figure 11: Loading Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	itching	skin_rash	nodal_skir	continuou	shivering	chills	joint_pain	stomach_	acidity	ulcers_on	muscle_w	vomiting	burning_n	spotting_
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	1	1	0	0	0	0	0	0	0	0	0	0	0
8	1	0	1	0	0	0	0	0	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	0	0	0	0	0	0	0
11	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 12: Symptoms Table



col_0	n
label	
(vertigo) Paroymsal Positional Vertigo	86
AIDS	87
Acne	89
Alcoholic hepatitis	70
Allergy	81
Arthritis	86
Bronchial Asthma	79
Cervical spondylosis	82
Chicken pox	82
Chronic cholestasis	87
Common Cold	90
Dengue	97
Diabetes	81
Dimorphic hemmorhoids(piles)	90
Drug Reaction	76
Fungal infection	90
GERD	77

Figure 13: Split Train Test Data

Graphs

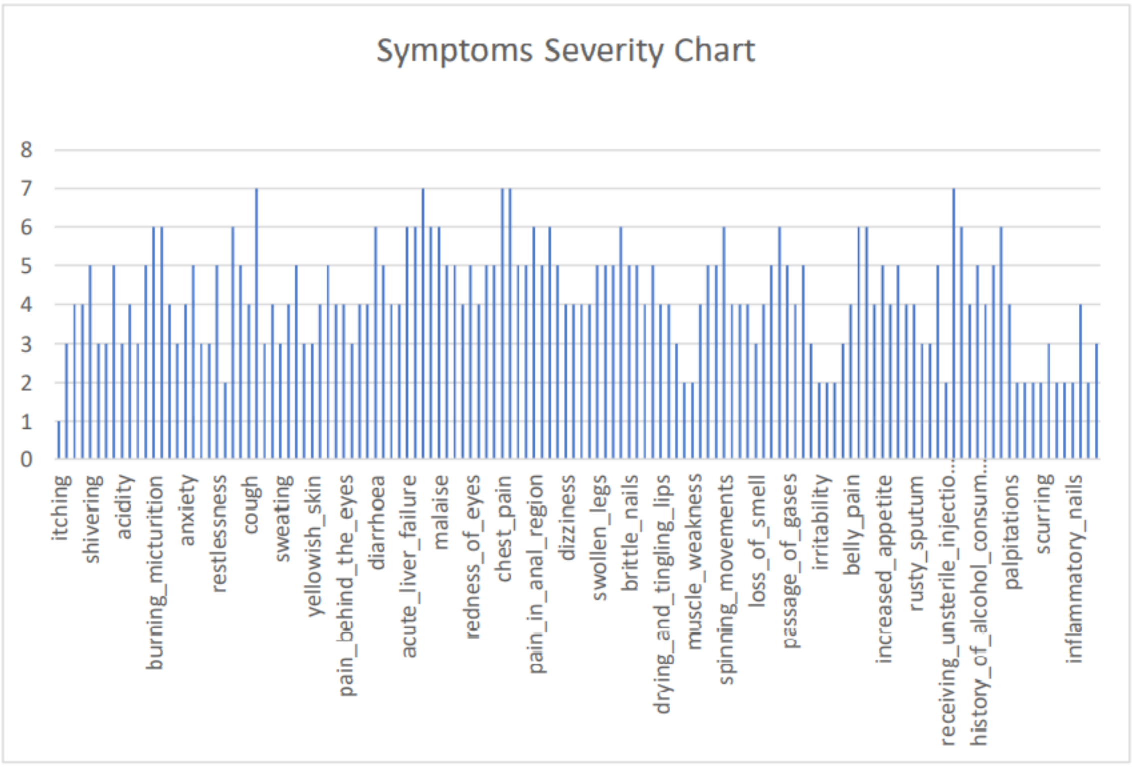


Figure 15: Symptoms Severity Chart

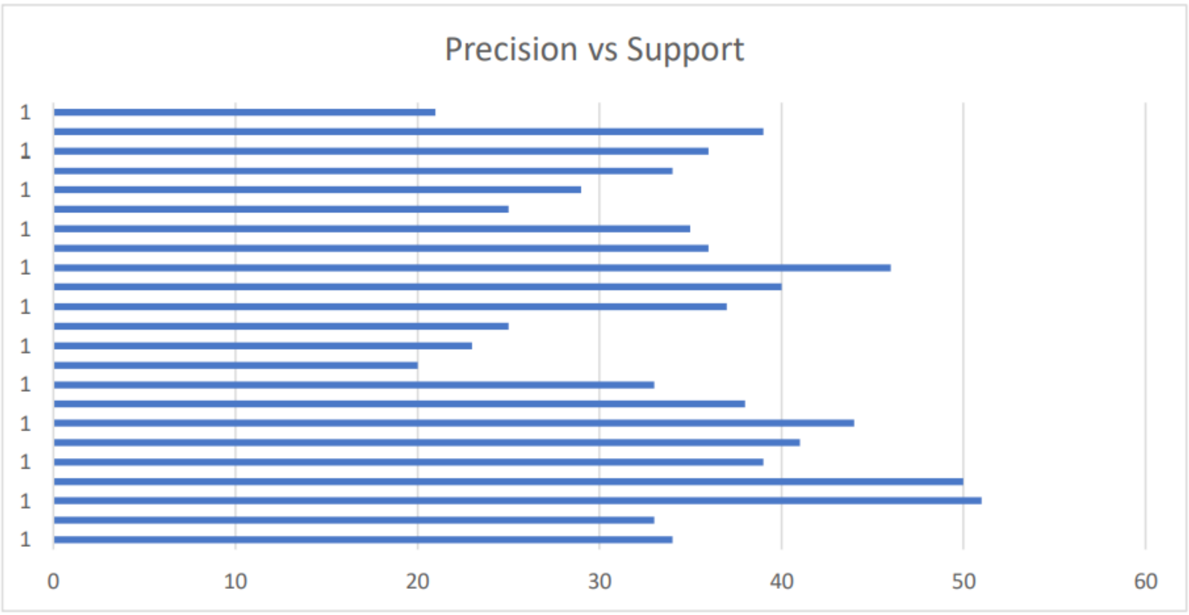
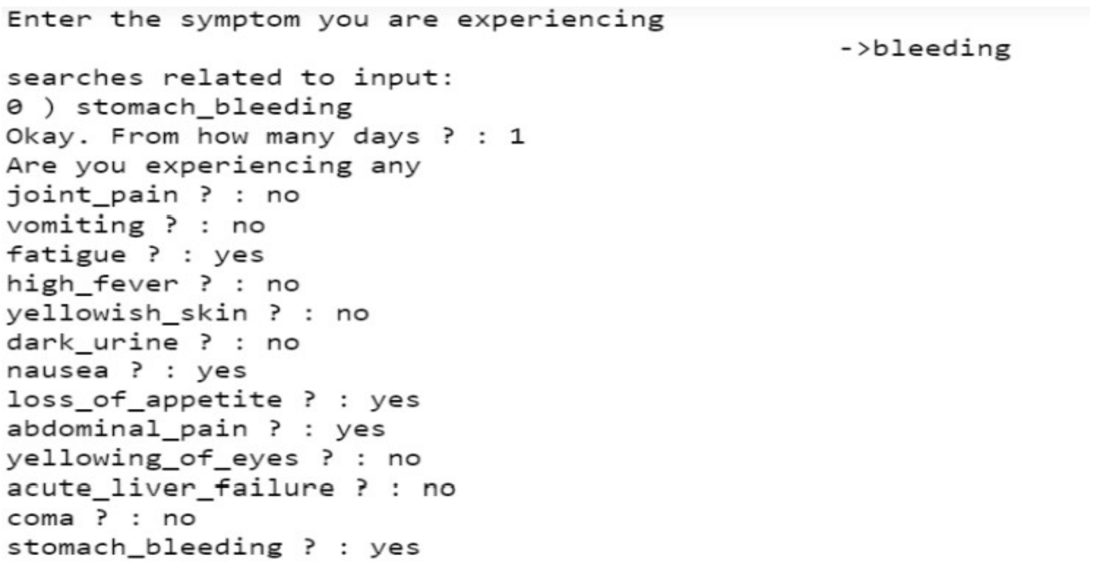
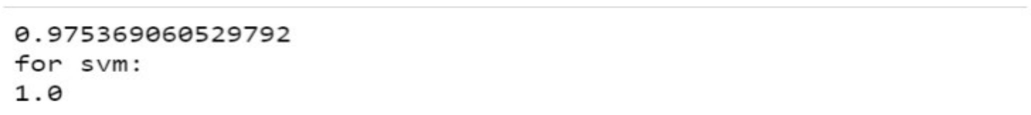


Figure 16: Precision vs Support Chart

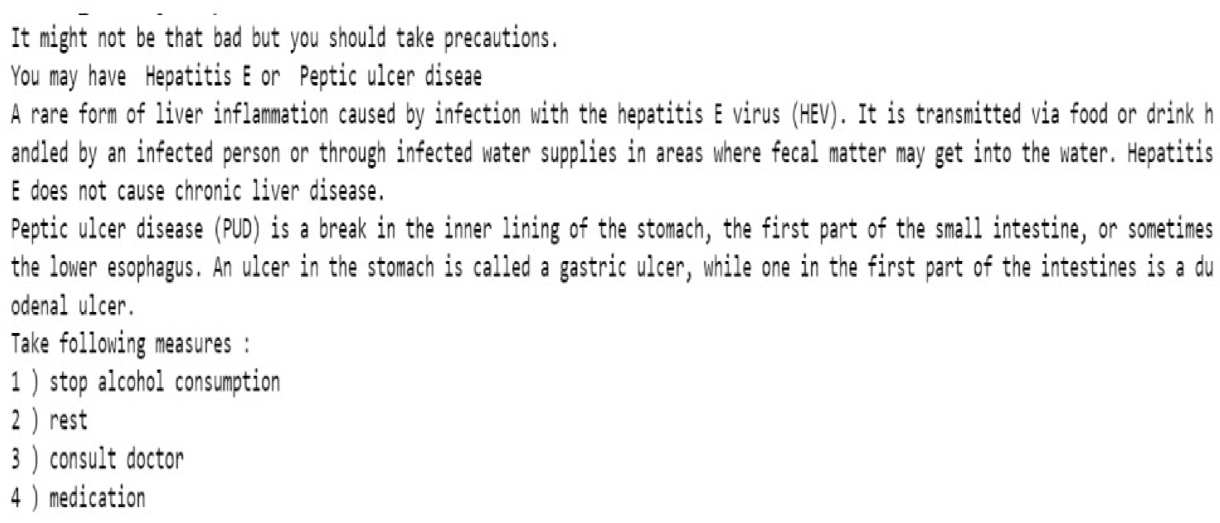
Output



```
Enter the symptom you are experiencing
                                                        ->bleeding
searches related to input:
0 ) stomach_bleeding
Okay. From how many days ? : 1
Are you experiencing any
joint_pain ? : no
vomiting ? : no
fatigue ? : yes
high_fever ? : no
yellowish_skin ? : no
dark_urine ? : no
nausea ? : yes
loss_of_appetite ? : yes
abdominal_pain ? : yes
yellowing_of_eyes ? : no
acute_liver_failure ? : no
coma ? : no
stomach_bleeding ? : yes
```



```
0.975369060529792
for svm:
1.0
```



```
It might not be that bad but you should take precautions.
You may have  Hepatitis E or  Peptic ulcer diseae
A rare form of liver inflammation caused by infection with the hepatitis E virus (HEV). It is transmitted via food or drink h
andled by an infected person or through infected water supplies in areas where fecal matter may get into the water. Hepatitis
E does not cause chronic liver disease.
Peptic ulcer disease (PUD) is a break in the inner lining of the stomach, the first part of the small intestine, or sometimes
the lower esophagus. An ulcer in the stomach is called a gastric ulcer, while one in the first part of the intestines is a du
odenal ulcer.
Take following measures :
1 ) stop alcohol consumption
2 ) rest
3 ) consult doctor
4 ) medication
```

Figure 17: Final Output

Conclusion

In this paper we have proposed a health recommendation system which is interactive through a chat bot, where patients input their symptoms and we use Random forest classifier and SVC to come to a conclusion that is subject to evaluation through its confidence level. The aim for using different algorithms is to work on the accuracy of the recommendation we are giving the users. The data in the health sector will change with the rise of any new medicine or disease coming in nature, like the Corona virus hence we make sure that we do not mislead patients into irrelevant precautionary measures as it can be detrimental to health as well.

We have achieved the accuracy of 97% using the Random Forest Classifier on our data sets.

This project being under the health care sector will always have a constant drawback in terms of variation in data, hence achieving a health care recommendation that works with absolute certainty has not been achieved yet. The aim is to focus on the output that we receive-by preprocessing the data and filtering it by weight to understand the severity of symptoms.

The severity of symptoms also comes with a warning signal of whether this requires professional assistance-and the prediction is based on the decision tree of our response to the symptom

Reference

1. Chatbots as conversational healthcare services by Mladjan Jovanovic; Marcos Baez; Fabio Casati

2. HealthAssistantBot: A Personal Health Assistant for the Italian Language by Marco Polignano; Fedelucio Narducci; Andrea Iovine; Cataldo Musto; Marco De Gemmis; Giovanni Semeraro

3. An Efficient and Privacy-Preserving Disease Risk Prediction Scheme for E-Healthcare by Xue Yang; Rongxing Lu; Jun Shao; Xiaohu Tang; Haomiao Yang

4. A Medical-History-Based Potential Disease Prediction Algorithm by Wenxing Hong; Ziang Xiong; Nannan Zheng; Yang Weng

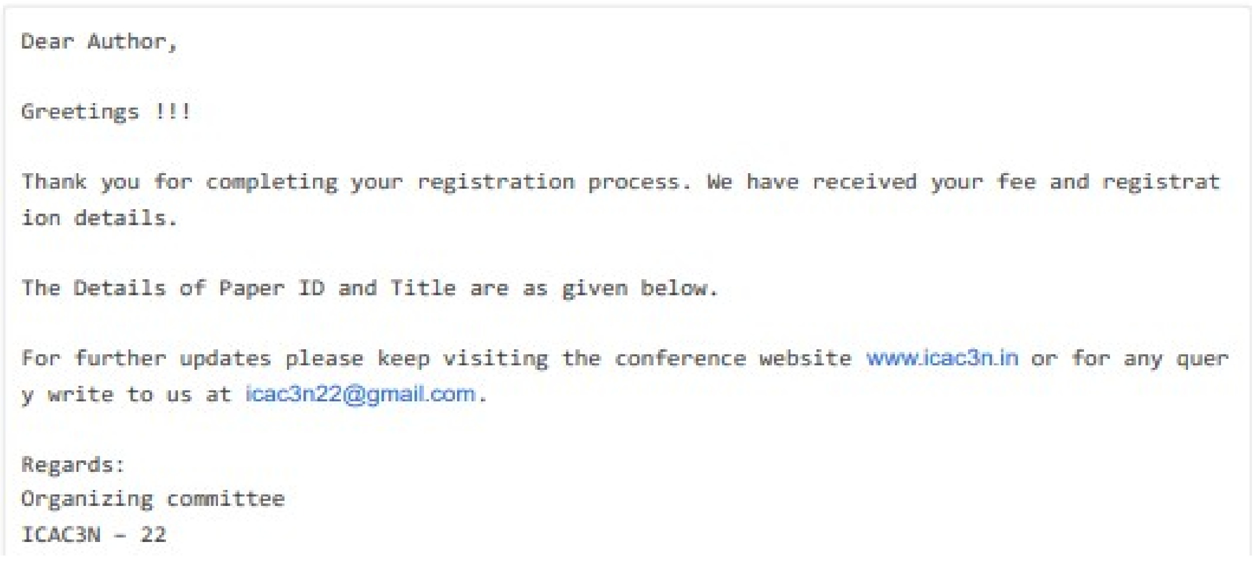
5. Designing Disease Prediction Model Using Machine Learning Approach by Dhiraj Dahiwade; Gajanan Patle; Ektaa Meshram

6. A Proposed Model for Lifestyle Disease Prediction Using Support Vector Machine by Mrunmayi Patil; Vivian Brian Lobo; Pranav Puranik; Aditi Pawaskar; Adarsh Pai; Rupesh Mishra

7. Efficient heart disease prediction system using decision tree by Purushottam;Kanak Saxena;Richa Sharma

8. Chatbot for Healthcare System Using Artificial Intelligence by Lekha Athota; Vinod Kumar Shukla; Nitin Pandey; Ajay Rana

PAYMENT CONFIRMATION



Dear Author,

Greetings !!!

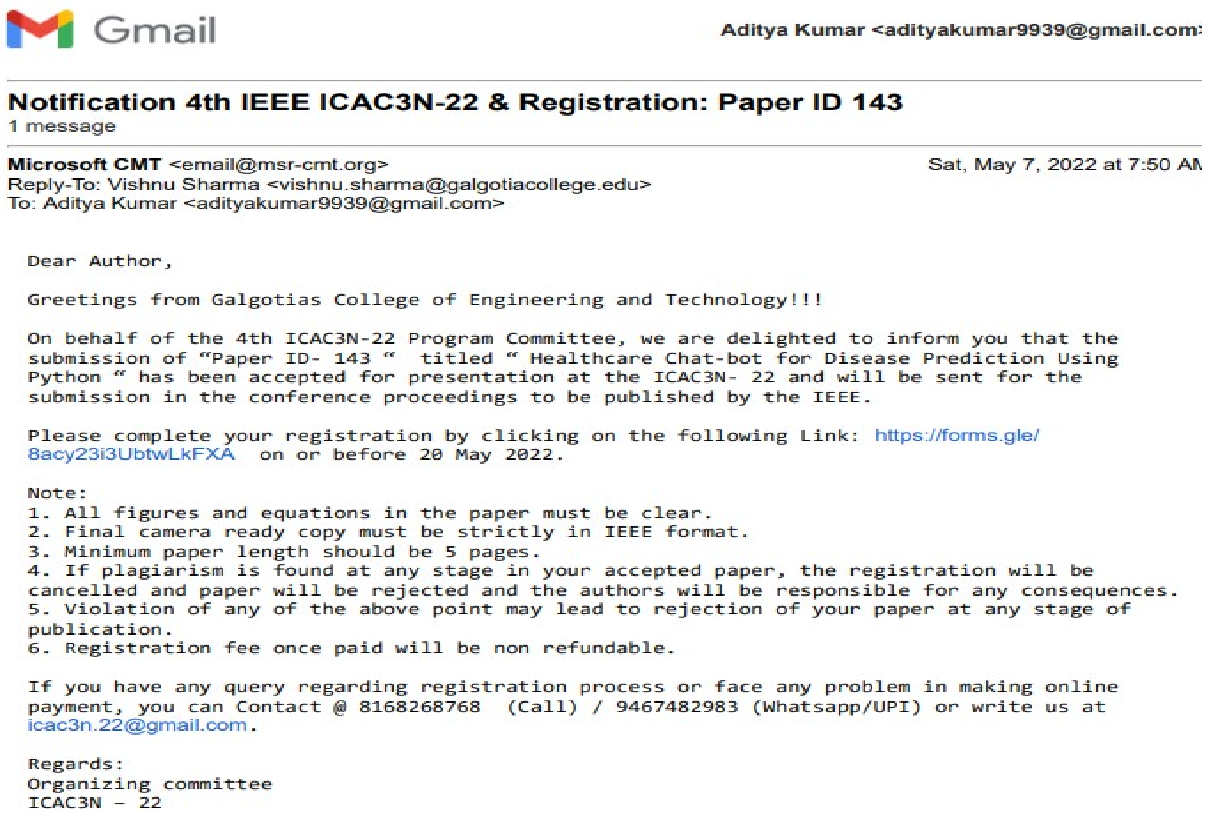
Thank you for completing your registration process. We have received your fee and registrat
ion details.

The Details of Paper ID and Title are as given below.

For further updates please keep visiting the conference website www.icac3n.in or for any quer
y write to us at icac3n22@gmail.com.

Regards:
Organizing committee
ICAC3N - 22

Acceptance Of Research Paper



Gmail

Aditya Kumar <adityakumar9939@gmail.com

Notification 4th IEEE ICAC3N-22 & Registration: Paper ID 143
1 message

Microsoft CMT <email@msr-cmt.org>
Reply-To: Vishnu Sharma <vishnu.sharma@galgotiacollege.edu>
To: Aditya Kumar <adityakumar9939@gmail.com>

Sat, May 7, 2022 at 7:50 AM

Dear Author,

Greetings from Galgotias College of Engineering and Technology!!!

On behalf of the 4th ICAC3N-22 Program Committee, we are delighted to inform you that the submission of "Paper ID- 143 " titled " Healthcare Chat-bot for Disease Prediction Using Python " has been accepted for presentation at the ICAC3N- 22 and will be sent for the submission in the conference proceedings to be published by the IEEE.

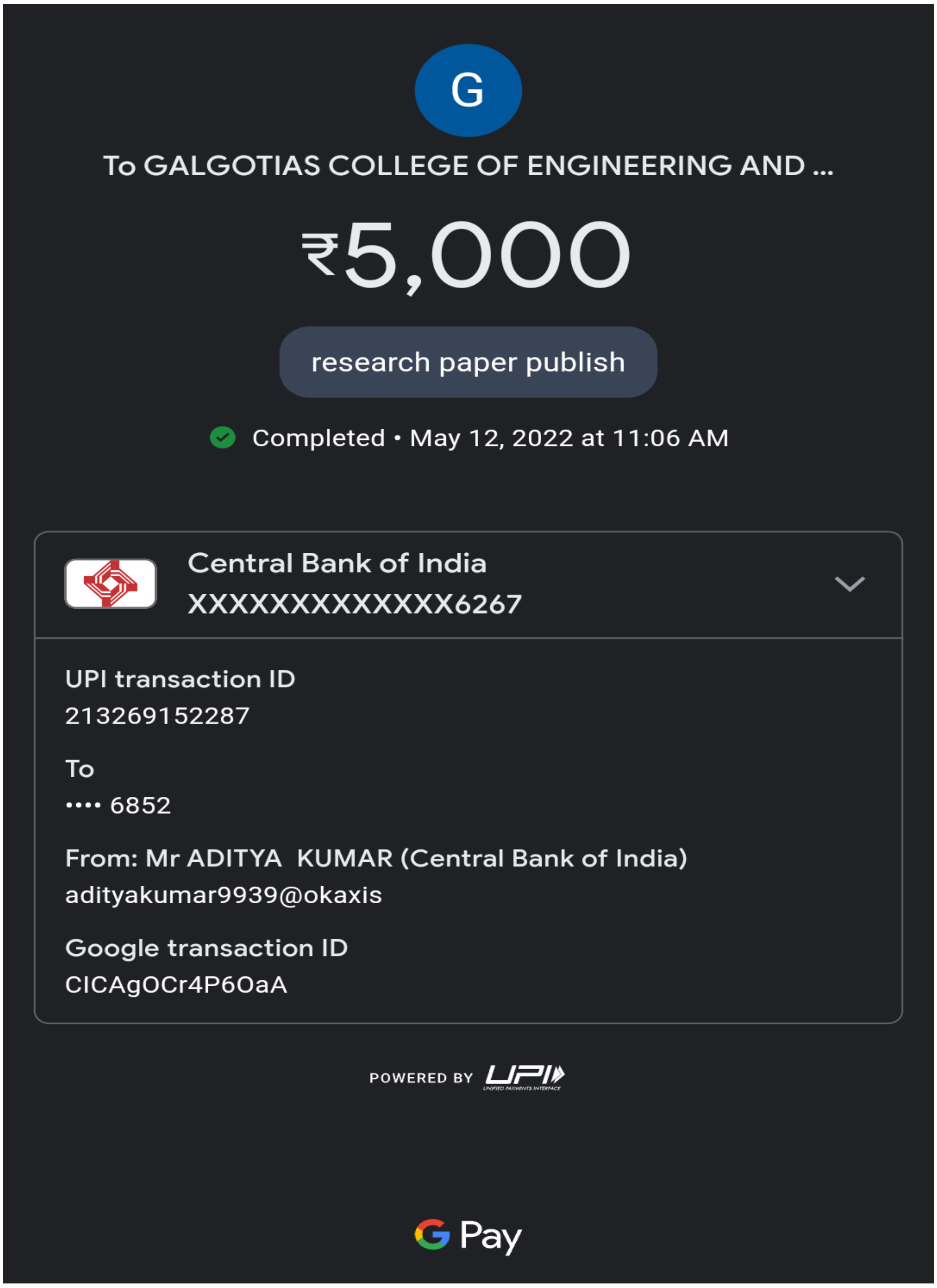
Please complete your registration by clicking on the following Link: https://forms.gle/8acy23i3UbtwLkFXA on or before 20 May 2022.

Note:
1. All figures and equations in the paper must be clear.
2. Final camera ready copy must be strictly in IEEE format.
3. Minimum paper length should be 5 pages.
4. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences.
5. Violation of any of the above point may lead to rejection of your paper at any stage of publication.
6. Registration fee once paid will be non refundable.

If you have any query regarding registration process or face any problem in making online payment, you can Contact @ 8168268768 (Call) / 9467482983 (Whatsapp/UPI) or write us at icac3n.22@gmail.com.

Regards:
Organizing committee
ICAC3N - 22

Payment Proof



G

To GALGOTIAS COLLEGE OF ENGINEERING AND ...

₹5,000

research paper publish

Completed • May 12, 2022 at 11:06 AM

Central Bank of India
XXXXXXXXXXXX6267

UPI transaction ID
213269152287

To
•••• 6852

From: Mr ADITYA KUMAR (Central Bank of India)
adityakumar9939@okaxis

Google transaction ID
CICAgOCr4P6OaA

POWERED BY UPI

G Pay