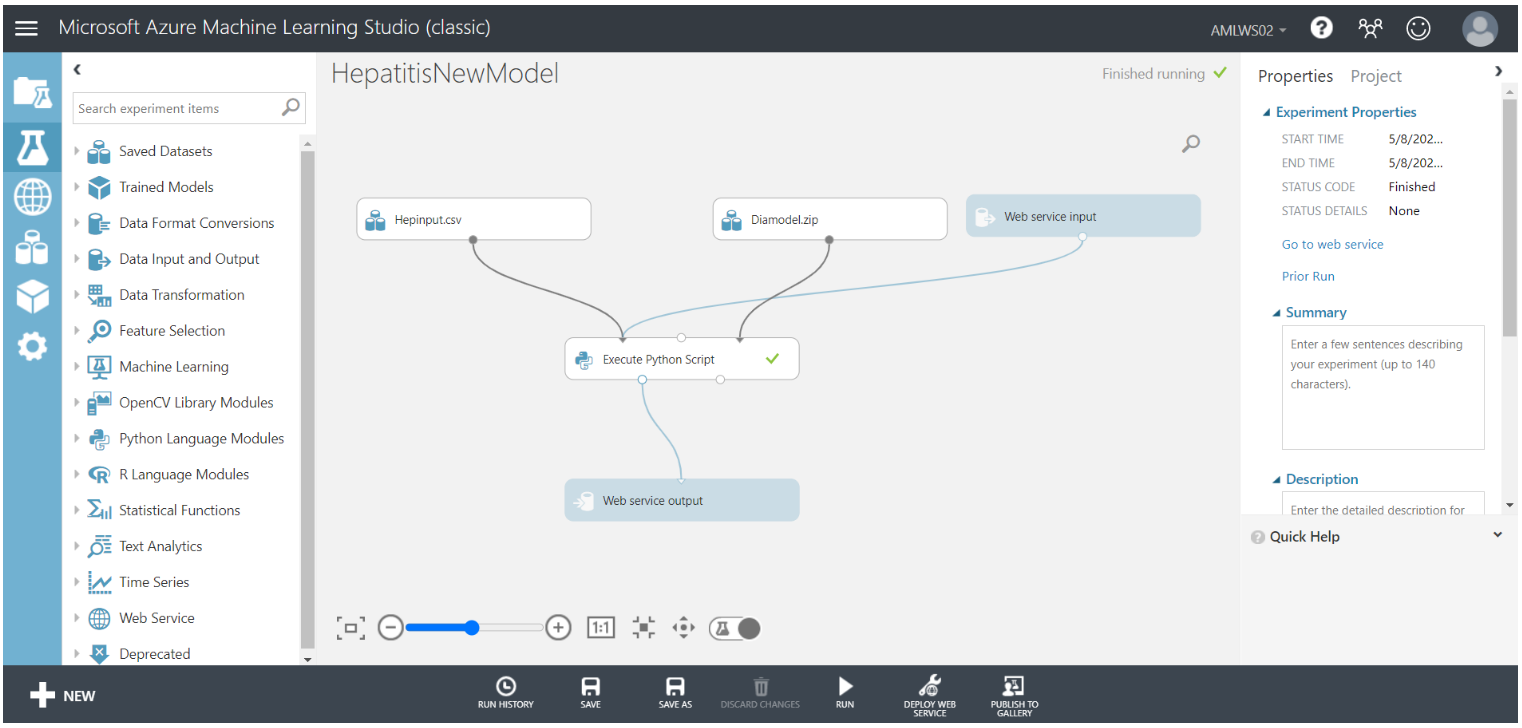
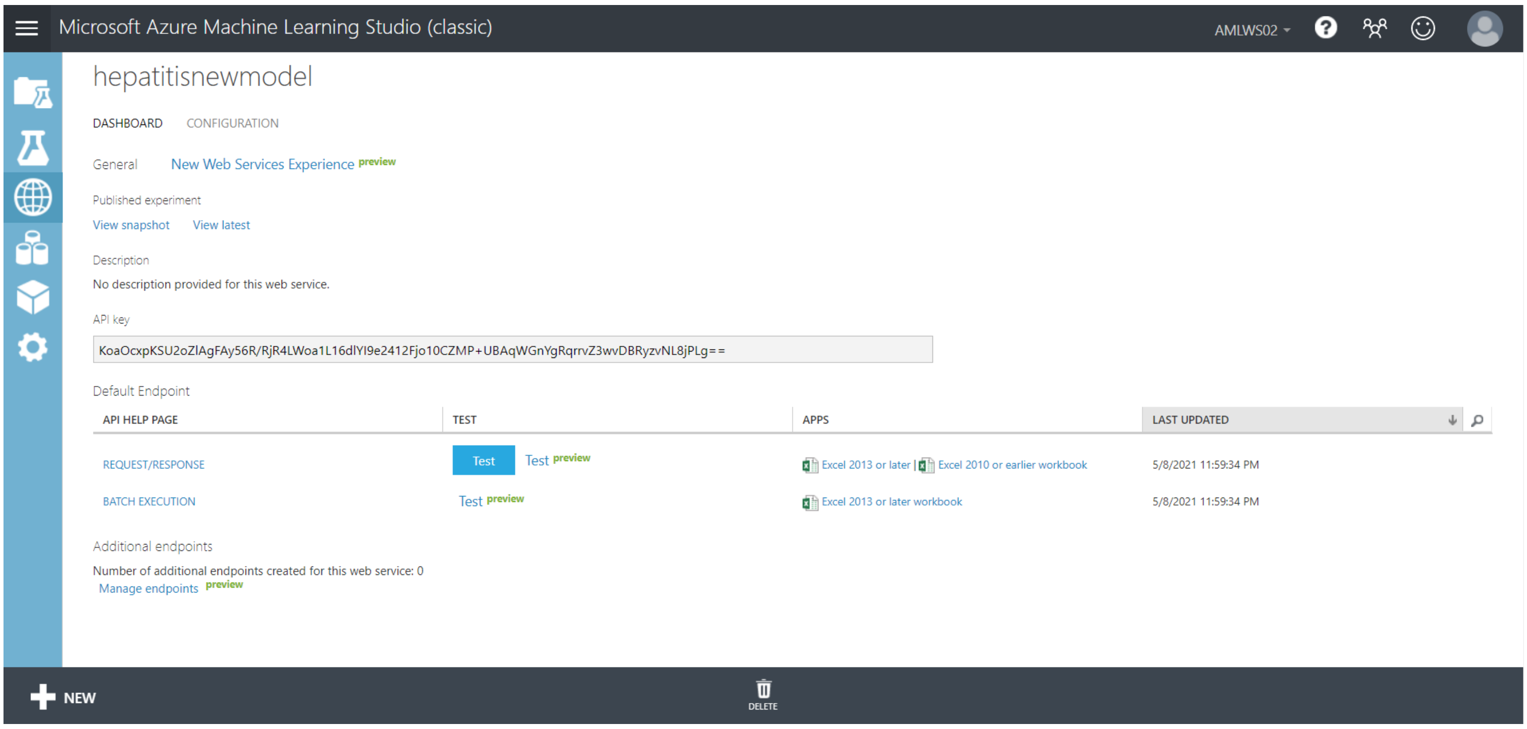
# RECOMMENDATION SYSTEM

## (1). **Model used for Recommendation System (Hepatitis Model):**



This is the model that we have used for our Recommendation System.

## (2). **Web-Services used for the Recommendation System:**



hepatitisnewmodel

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment

View snapshot View latest

Description

No description provided for this web service.

API key

KoaOcxpKSU2oZlAgFAy56R/RjR4LWoa1L16dlYI9e2412Fjo10CZMP+UBAqWGnYgRqrrvZ3wvDBRyzvNL8jPLg==

Default Endpoint

| API HELP PAGE | TEST | APPS | LAST UPDATED |
|---|---|---|---|
| REQUEST/RESPONSE | Test Test preview | Excel 2013 or later \| Excel 2010 or earlier workbook | 5/8/2021 11:59:34 PM |
| BATCH EXECUTION | Test preview | Excel 2013 or later workbook | 5/8/2021 11:59:34 PM |

Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints preview

### Request

| Method | Request URI | HTTP Version |
|---|---|---|
| POST | https://ussouthcentral.services.azureml.net/workspaces/a0597a9dd2f742f2a4159802a83137be/services/3c661031375441e59e86b18a1d086bfa/execute?api-version=2.0&details=true | HTTP/1.1 |

After deploying the '**Hepatitis**' model, this is the web-service that we have used for our recommendation system. In our recommendation system we have used the '**API Key**' and the '**Request URI**' so that our recommendation bot can fetch the data from our model while using it.

## (3). Recommendation system techniques:

A recommendation engine is a system that proposes products, administrations, data to users based on analysis of data. Notwithstanding, the recommendation can get from an assortment of variables like the historical backdrop of the user, and the conduct of comparative users.

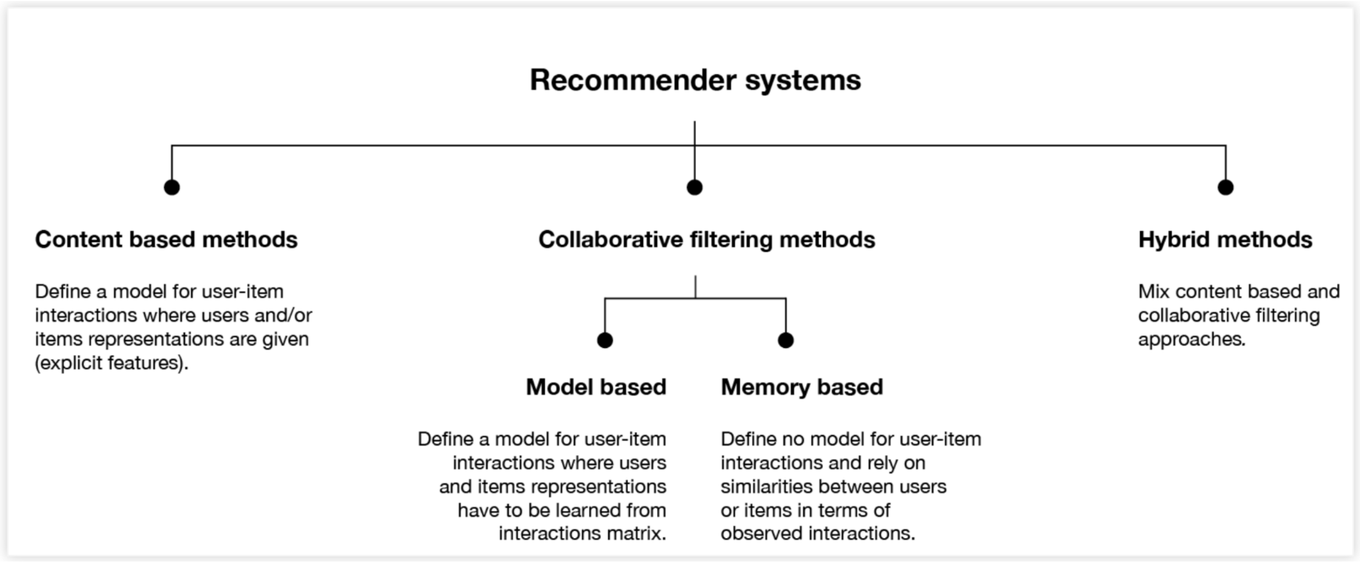

Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

Fig-1: Recommendation system techniques

## (4). Design and build a chatbot:

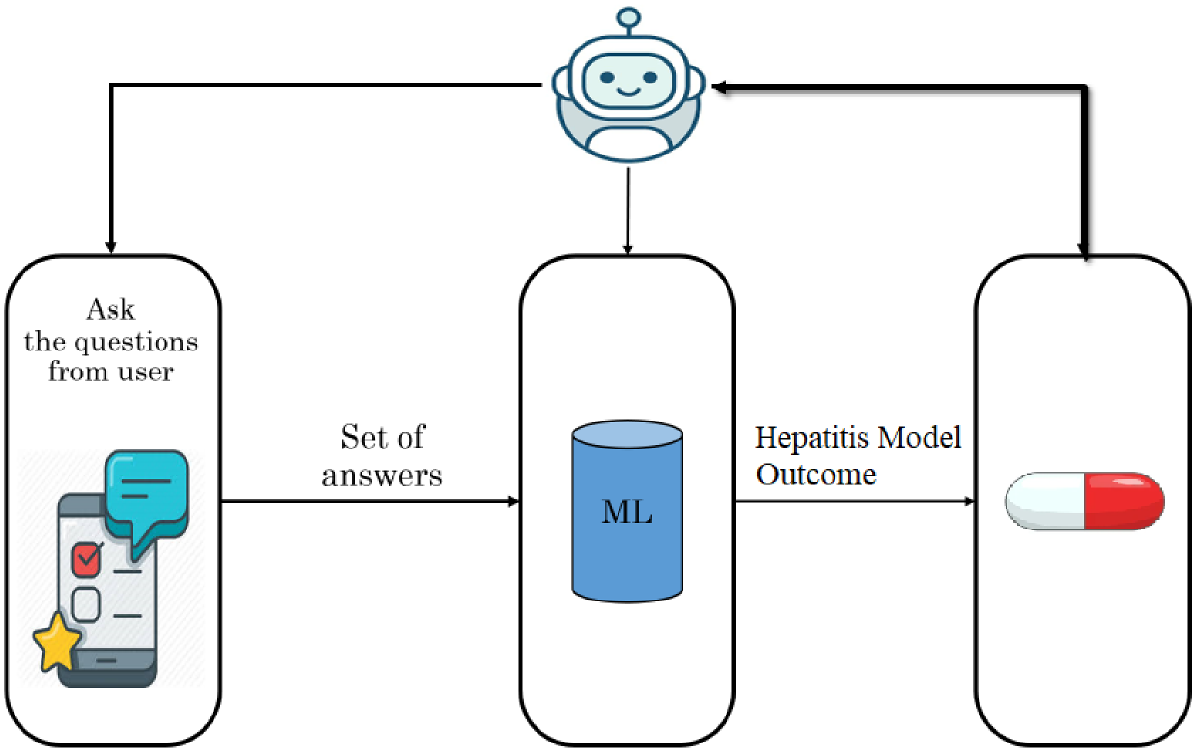


Fig-2: Outline of the Hepatitis chatbot

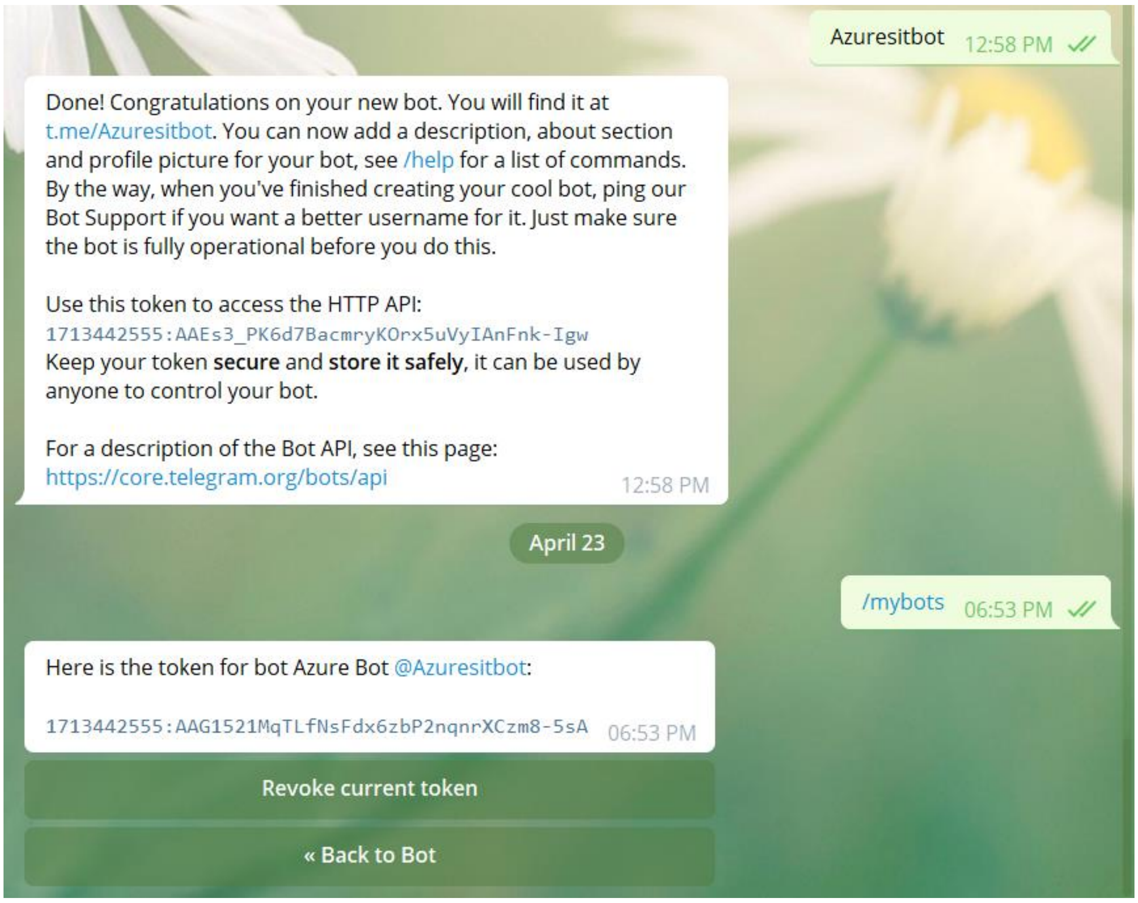
## (5). **Chatbot using Telegram:**

We're going to build a bot using the Telegram Application. First, we need to register a chatbot using 'BotFather'.

<u>To do this:</u>

1. Search @**BotFather** in telegram search box. Or use '**https://t.me/BotFather**' directly.
2. Create a new bot by telling him **/newbot** command.

After providing a name and a username, we will be given a bot name and a token:



Azuresitbot 12:58 PM

Done! Congratulations on your new bot. You will find it at t.me/Azuresitbot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
`1713442555:AAEs3_PK6d7BacmryKOrx5uVyIAnFnk-Igw`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

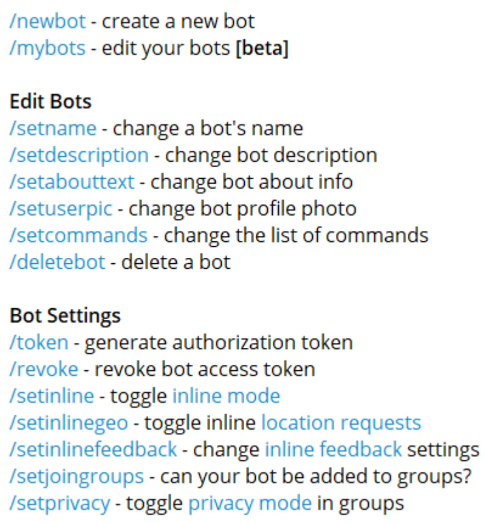
For a description of the Bot API, see this page:
https://core.telegram.org/bots/api 12:58 PM

April 23

/mybots 06:53 PM

Here is the token for bot Azure Bot @Azuresitbot:

`1713442555:AAG1521MqTLfNsFdx6zbP2nqnrXCzm8-5sA` 06:53 PM



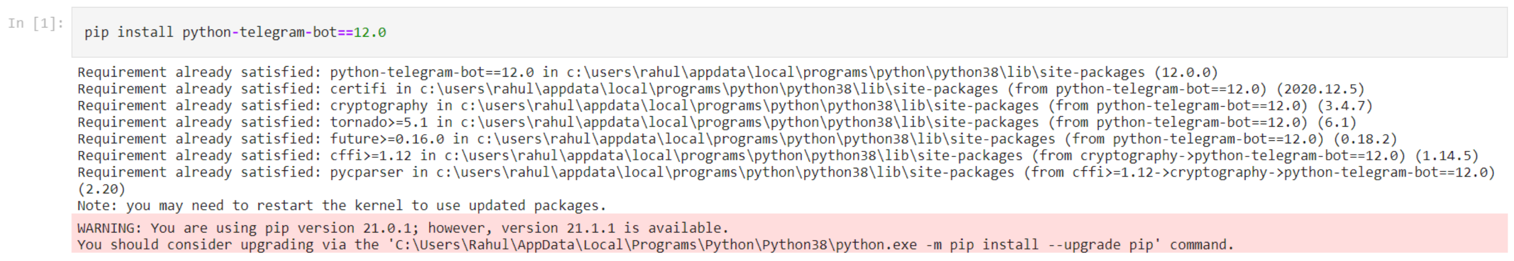
/newbot - create a new bot
/mybots - edit your bots **[beta]**

**Edit Bots**
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot

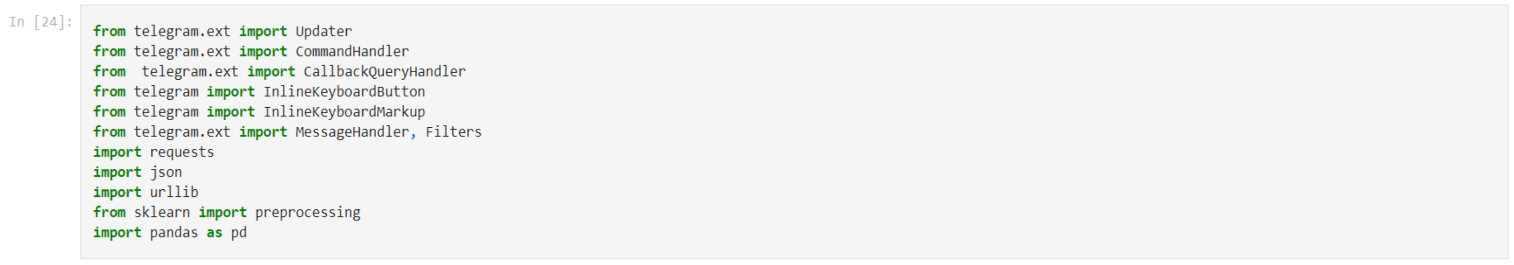
**Bot Settings**
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups

We can even edit the name, token, bot description, privacy, profile photo and many more setting by using these commands from above.

## (6). Install Libraries used for chatbot:



```
In [1]:  pip install python-telegram-bot==12.0

Requirement already satisfied: python-telegram-bot==12.0 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (12.0.0)
Requirement already satisfied: certifi in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (2020.12.5)
Requirement already satisfied: cryptography in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (3.4.7)
Requirement already satisfied: tornado>=5.1 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (6.1)
Requirement already satisfied: future>=0.16.0 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (0.18.2)
Requirement already satisfied: cffi>=1.12 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from cryptography->python-telegram-bot==12.0) (1.14.5)
Requirement already satisfied: pycparser in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from cffi>=1.12->cryptography->python-telegram-bot==12.0)
(2.20)
Note: you may need to restart the kernel to use updated packages.
WARNING: You are using pip version 21.0.1; however, version 21.1.1 is available.
You should consider upgrading via the 'C:\Users\Rahul\AppData\Local\Programs\Python\Python38\python.exe -m pip install --upgrade pip' command.
```



```
In [24]:
from telegram.ext import Updater
from telegram.ext import CommandHandler
from  telegram.ext import CallbackQueryHandler
from telegram import InlineKeyboardButton
from telegram import InlineKeyboardMarkup
from telegram.ext import MessageHandler, Filters
import requests
import json
import urllib
from sklearn import preprocessing
import pandas as pd
```

First, we'll install the python telegram bot so that we can use the telegram bot services in our python.

**(A). 'from telegram.ext import Updater':**

It plays the main role to control the conversation between the bot and user.

1. This class associates the user to the chatbot page which is made by 'Botfather' (**interface the user to the bot**).

2. It handles every one of the user's and bot's inputs by running the functions which is created for this reason as it were.

3. It stops the proposed functions when there is no need any longer to send or get message or inquiry among user and bot through the functions.

**(B). 'from telegram.ext import CommandHandler'**

**'from telegram.ext import CallbackQueryHandler'**

**'from telegram.ext import Message Handler, Filters'**

These libraries are used for sending message from user to bot.

**(C). 'from telegram import InlineKeyboardButton'**

**'from telegram import InlineKeyboardMarkup'**

These libraries are used for sending message from bot to user.

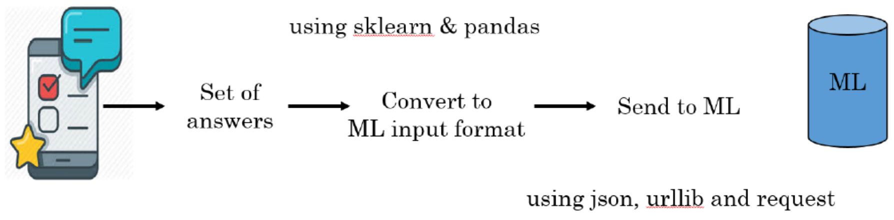
**(D). import requests**

**import json**

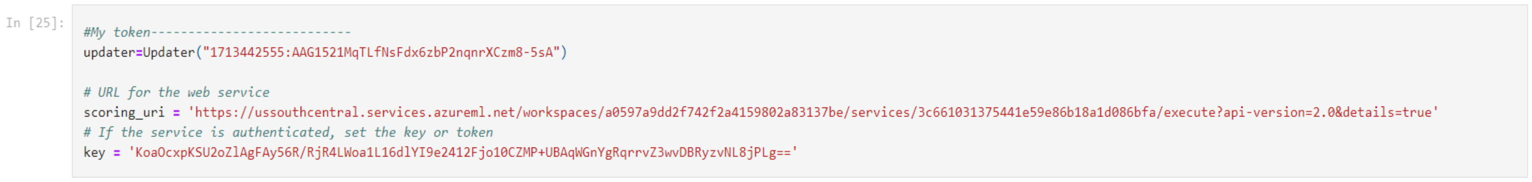
**import urllib**

**from sklearn import preprocessing**

**import pandas as pd**



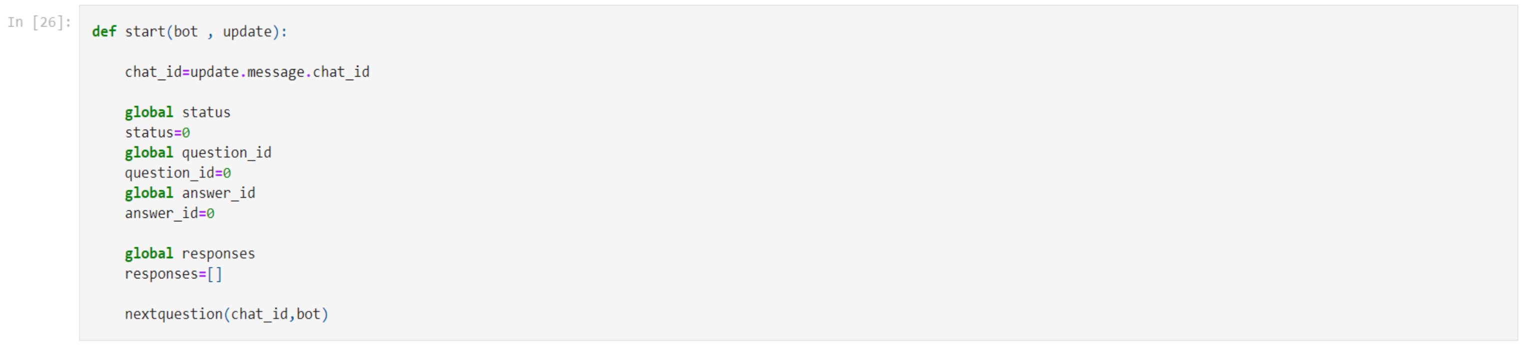
## (7). Adding Token and Request URI:



```
In [25]:
#My token--------------------------
updater=Updater("1713442555:AAG1521MqTLfNsFdx6zbP2nqnrXCzm8-5sA")

# URL for the web service
scoring_uri = 'https://ussouthcentral.services.azureml.net/workspaces/a0597a9dd2f742f2a4159802a83137be/services/3c661031375441e59e86b18a1d086bfa/execute?api-version=2.0&details=true'
# If the service is authenticated, set the key or token
key = 'KoaOcxpKSU2oZlAgFAy56R/RjR4LWoa1L16dlYI9e2412Fjo10CZMP+UBAqWGnYgRqrrvZ3wvDBRyzvNL8jPLg=='
```

We'll add the token from Botfather that we have received after making the bot in the 'updater' section which will act as an interface between the user and the bot. We'll also add the 'Key' & 'Request URI (or scoring uri)' from the Hepatitis model that we have deployed on Azure.

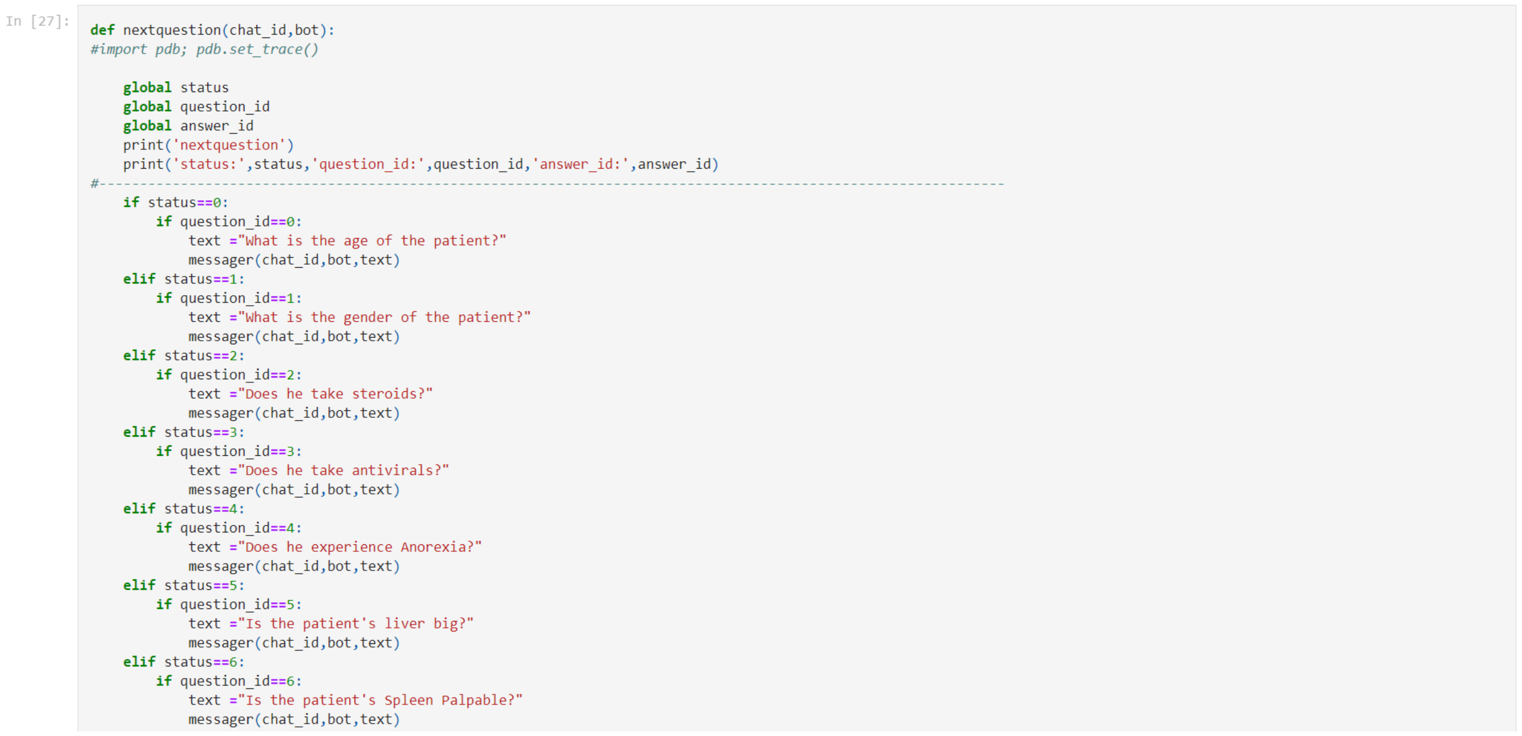
## (8). Start function of the Bot:



```
In [26]:
def start(bot , update):

    chat_id=update.message.chat_id

    global status
    status=0
    global question_id
    question_id=0
    global answer_id
    answer_id=0

    global responses
    responses=[]

    nextquestion(chat_id,bot)
```

Here the bot and update will be sent to the start function. We'll start the bot in Telegram using '/start'. Update stores the information of updated chat, like chat.id. In this function, entering '/start', should make the bot Jump to the next question (nextquestion function will do this part).

## (9). Next Question function of the Bot:



```
In [27]:
def nextquestion(chat_id,bot):
#import pdb; pdb.set_trace()

    global status
    global question_id
    global answer_id
    print('nextquestion')
    print('status:',status,'question_id:',question_id,'answer_id:',answer_id)
#------------------------------------------------------------------------------------------------------------------------
    if status==0:
        if question_id==0:
            text ="What is the age of the patient?"
            messager(chat_id,bot,text)
    elif status==1:
        if question_id==1:
            text ="What is the gender of the patient?"
            messager(chat_id,bot,text)
    elif status==2:
        if question_id==2:
            text ="Does he take steroids?"
            messager(chat_id,bot,text)
    elif status==3:
        if question_id==3:
            text ="Does he take antivirals?"
            messager(chat_id,bot,text)
    elif status==4:
        if question_id==4:
            text ="Does he experience Anorexia?"
            messager(chat_id,bot,text)
    elif status==5:
        if question_id==5:
            text ="Is the patient's liver big?"
            messager(chat_id,bot,text)
    elif status==6:
        if question_id==6:
            text ="Is the patient's Spleen Palpable?"
            messager(chat_id,bot,text)
    elif status==7:
        if question_id==7:
            text ="Is the patient feeling Malaise?"
            messager(chat_id,bot,text)
    elif status==8:
        if question_id==8:
            text ="Is the patient having Ascites?"
            messager(chat_id,bot,text)
    elif status==9:
        if question_id==9:
            text ="Is the patient having Varices?"
            messager(chat_id,bot,text)
    elif status==10:
        if question_id==10:
            text ="Is the patient afraid of spiders?"
            messager(chat_id,bot,text)
    elif status==11:
        if question_id==11:
            text ="Is the patient having jaundice?"
            messager(chat_id,bot,text)
    elif status==12:
        if question_id==12:
            text ="Is the patient having dark urine?"
            messager(chat_id,bot,text)
    elif status==13:
        if question_id==13:
            text ="Is the patient having low appetite"
            messager(chat_id,bot,text)
    elif status==14:
        if question_id==14:
            text ="Is the patient having itchy skin?"
            messager(chat_id,bot,text)
    elif status==15:
        if question_id==15:
            text ="Is the patient having pale stools?"
            messager(chat_id,bot,text)
```

In this function we'll create the questions that will be asked by the bot (as question_id) & answers that'll be provided by the user according to the deployed model dataset (as answer_id). The status will update after every question which will automatically lead to the next question.

## (10). **Messager Function:**

In [28]:
```
def messager(chat_id,bot,text1):


    # the below function is explained in this link : https://python-telegram-bot.readthedocs.io/en/stable/telegram.bot.html#telegram.Bot.send_chat_action
    bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
    bot.send_message(chat_id = chat_id ,text = text1)
    global question_id
    question_id=question_id+1
```

This function is used to send questions to the bot.

**SendChatAction**:  Utilize this strategy when you need to tell the user that something is going on the bot's side. The status is set for 5 seconds or less (when a message shows up from your bot, Telegram clients clear its typing status). Telegram possibly suggests utilizing this technique when a response from the bot will set aside an observable measure of effort to show up.

## (11). **ContQuestions Function:**

In [29]:
```
def contQuestions(bot,update):
    print("contQuestions")

    chat_id=update.message.chat_id
    text=update.message.text

    global status
    global answer_id
    global question_id

    if answer_id == 0:
        global age
        age=text
        responses.append(age)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)

    elif answer_id == 1:
        global Gender
        Gender=text
        responses.append(Gender)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 2:
        global Steroids
        Steroids=text
        responses.append(Steroids)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 3:
        global Antivirals
        Antivirals=text
        responses.append(Antivirals)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 4:
        global Anorexia
        Anorexia=text
        responses.append(Anorexia)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 5:
        global BigL
        BigL=text
        responses.append(BigL)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 6:
        global SpleenP
        SpleenP=text
        responses.append(SpleenP)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
```

```
    elif answer_id == 7:
        global Malaise
        Malaise=text
        responses.append(Malaise)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 8:
        global Ascites
        Ascites=text
        responses.append(Ascites)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 9:
        global Varices
        Varices=text
        responses.append(Varices)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 10:
        global Spiders
        Spiders=text
        responses.append(Spiders)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
```

```
    elif answer_id == 11:
        global Jaundice
        Jaundice=text
        responses.append(Jaundice)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 12:
        global Durine
        Durine=text
        responses.append(Durine)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 13:
        global Lowapp
        Lowapp=text
        responses.append(Lowapp)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 14:
        global itchyskin
        itchyskin=text
        responses.append(itchyskin)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 15:
        global Palestools
        Palestools=text
        responses.append(Palestools)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        #updater.dispatcher.remove_handler(consQuestion_handler)
        end(chat_id,bot)
        updater.dispatcher.remove_handler(consQuestion_handler)
```

This function gets the text (update.message.text ) from the user and stores the messages entered by the user. If the user is using text answers then only, we can use this function. We'll make global variables to every answer_id and pass it to program.

## (12). End Function:

```
In [30]:
def end(chat_id,bot):
    print('end')
    bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
    bot.send_message(chat_id = chat_id ,text = "Please wait...")

    array1=responses
    global db
    X=db
    array2=["","","","","","","","","","","","","","","",""]
    array2[0]=int(array1[0])
    array2[1]=transformation(X,1,array1[1])
    array2[2]=transformation(X,2,array1[2])
    array2[3]=transformation(X,3,array1[3])
    array2[4]=transformation(X,4,array1[4])
    array2[5]=transformation(X,5,array1[5])
    array2[6]=transformation(X,6,array1[6])
    array2[7]=transformation(X,7,array1[7])
    array2[8]=transformation(X,8,array1[8])
    array2[9]=transformation(X,9,array1[9])
    array2[10]=transformation(X,10,array1[10])
    array2[11]=transformation(X,11,array1[11])
    array2[12]=transformation(X,12,array1[12])
    array2[13]=transformation(X,13,array1[13])
    array2[14]=transformation(X,14,array1[14])
    array2[15]=transformation(X,15,array1[15])
    print(array2)
```

```
    data = {

        "Inputs": {

            "input1":
                {
                    "ColumnNames": ["Age", "Gender", "Steroids", "Antivirals", "Anorexia", "Big Liver",
                    "Palpable Spleen", "Malaise", "Ascites", "Varices", "Spiders", "Jaundice",
                    "Dark Urine", "Low Appetite","Itchy Skin","Pale Stools"],
                    "Values": [array2,array2,]
                }, },
        "GlobalParameters": {
        }
    }
    print(data)
    body = str.encode(json.dumps(data))
    # body=json.dumps(data)
    # print(data['Inputs']['input1'])

     # print(body)

    headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer ' + key)}

     # req = urllib.Request(url, body, headers)
    req = urllib.request.Request(scoring_uri, body, headers)
    try:

# If you are using Python 3+, replace urllib2 with urllib.request in the above code:
        response = urllib.request.urlopen(req)
        result = response.read()

        encoding = response.info().get_content_charset('utf-8')
        JSON_object = json.loads(result.decode(encoding))
        print(JSON_object['Results']['output1']['value']['Values'][0][0])
        if JSON_object['Results']['output1']['value']['Values'][0][0] == '1':
            bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
            bot.send_message(chat_id = chat_id ,text = "The patient is having Hepatitis")
            bot.send_message(chat_id = chat_id ,text = "The patient may not live long until he changes his lifestyle")


        elif JSON_object['Results']['output1']['value']['Values'][0][0] == '0':
            bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
            bot.send_message(chat_id = chat_id ,text = "The patient is not having Hepatitis")
            bot.send_message(chat_id = chat_id ,text = "The Patient is free to live the hospital")

    except urllib.error.HTTPError as error:
        print("The request failed with status code: " + str(error.code))

         # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
        print(error.info())

        print(json.loads(error.read()))
```

In the ‘end function’ we’ll store all the user’s response into an array. The ‘end function’ will send all the user’s response to the deployed ML model (Hepatitis Model).

To send the information we’ll use JSON for information exchange. Following methods are available in the JSON module:

| Method | Description |
|---|---|
| dumps() | encoding to JSON objects |
| dump() | encoded string writing on file |
| loads() | Decode the JSON string |
| load() | Decode while JSON file read |

**From Python data to JSON:**

Converting Python data to JSON is called an ‘**Encoding operation**’. Encoding is done with the help of JSON library method called as ‘**dumps()**’.

dumps() method converts dictionary object of python into JSON string data format and it can be found in the ‘**Request Response API Documentation for the used Model**’ in the sample code section (in Python).

We’ll send the data to ML model on Azure using the ‘**Scoring URI & Key**’ of deployed ML model.

After that we’ll convert the JSON information to python for our required output.

```
encoding = response.info().get_content_charset('utf-8')
JSON_object = json.loads(result.decode(encoding))
print(JSON_object['Results']['output1']['value']['Values'][0][0])
```

## **The Recommendation will be:**

After answering all the questions according to our dataset we’ll be arrived at this recommendation by our designed bot:

If class label=1, it means that the "The patient is having Hepatitis".

If class label=0, it means that the "The patient is not having Hepatitis".

(13). **Main:**

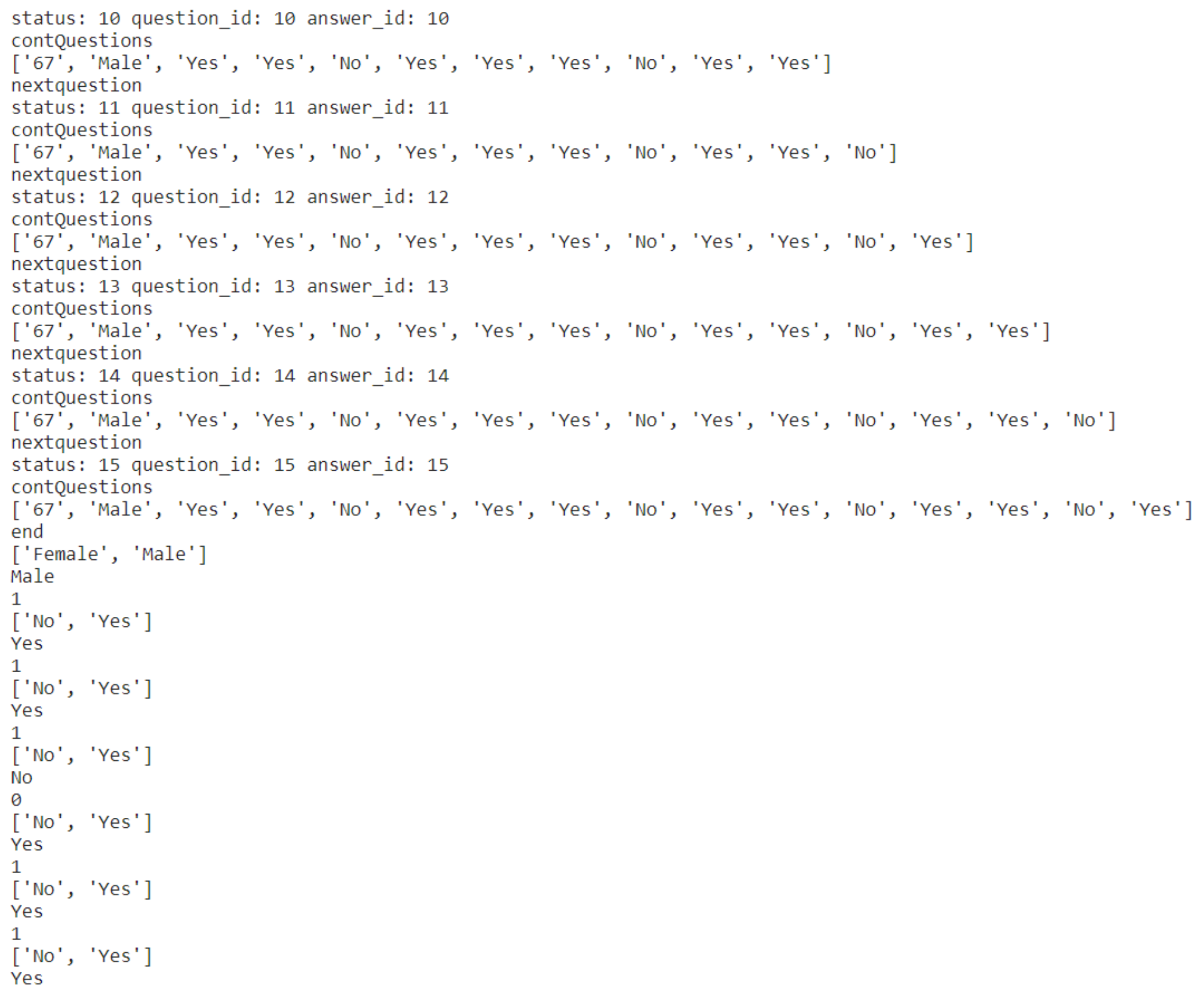
```
In [ ]:
url = r"D:\Deakin University (Applied AI (Professional)\SIT788 (Engineering AI Solutions)\Tasks and Assignments\Task- 6.1P\Hepatitis\New\hepatitisdata.csv"
dataset = pd.read_csv(url)
df = dataset.infer_objects()
    #print(df.dtypes[0])
for ij in range (0,16):
    if df.dtypes[ij]!='int64':
        dataset.iloc[:,ij] = dataset.iloc[:,ij].map(lambda x: x.strip())
            #print(dataset.iloc[:,ij])
array = dataset.values
X = array[:,0:16]

global db
db=X

# listen to get from user

command_start= CommandHandler('start',start)
consQuestion_handler = MessageHandler(Filters.text, contQuestions)

#updater
updater.dispatcher.add_handler(command_start,)
updater.dispatcher.add_handler(consQuestion_handler)

print('listening.....')

updater.start_polling()
updater.idle()
```

Here, first we'll add the dataset which will be used for prediction (hepatitisdata.csv). In this class, which utilizes the telegram.ext.Dispatcher, gives a frontend to the telegram Bot to the developer, so they can zero in on coding the bot. Its motivation is to get the updates from Telegram and to convey them to said dispatcher. It additionally runs in a separate thread, so the user can connect with the bot, for instance on the command line. The dispatcher upholds handlers for various types of data updates from Telegram, fundamental text commands and surprisingly arbitrary types.

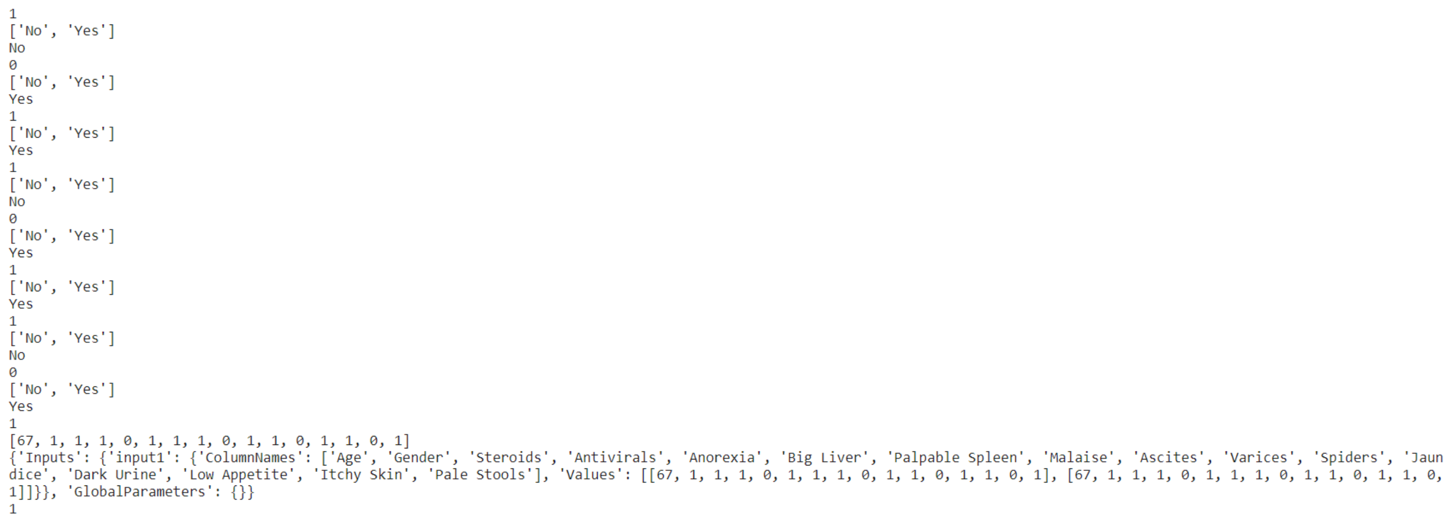
Idle: Blocks until one of the signals are received and stops the updater.

start_polling: Starts polling updates from Telegram.

```
listening.....
nextquestion
status: 0 question_id: 0 answer_id: 0
contQuestions
['67']
nextquestion
status: 1 question_id: 1 answer_id: 1
contQuestions
['67', 'Male']
nextquestion
status: 2 question_id: 2 answer_id: 2
contQuestions
['67', 'Male', 'Yes']
nextquestion
status: 3 question_id: 3 answer_id: 3
contQuestions
['67', 'Male', 'Yes', 'Yes']
nextquestion
status: 4 question_id: 4 answer_id: 4
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No']
nextquestion
status: 5 question_id: 5 answer_id: 5
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
status: 6 question_id: 6 answer_id: 6
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 7 question_id: 7 answer_id: 7
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes']
nextquestion
status: 8 question_id: 8 answer_id: 8
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No']
nextquestion
status: 9 question_id: 9 answer_id: 9
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
```



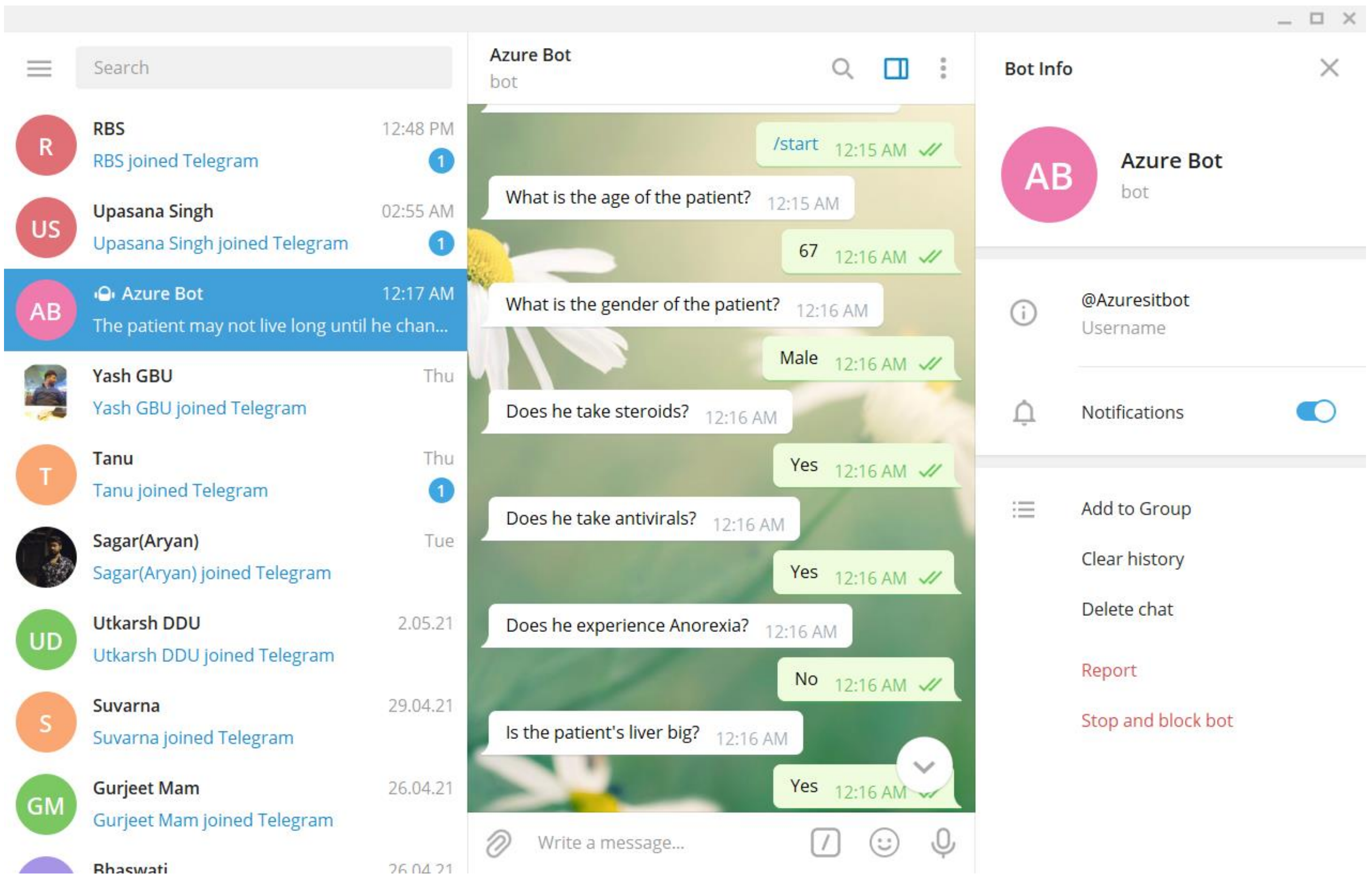
```
status: 10 question_id: 10 answer_id: 10
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 11 question_id: 11 answer_id: 11
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No']
nextquestion
status: 12 question_id: 12 answer_id: 12
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
status: 13 question_id: 13 answer_id: 13
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 14 question_id: 14 answer_id: 14
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No']
nextquestion
status: 15 question_id: 15 answer_id: 15
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
end
['Female', 'Male']
Male
1
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
```



```
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
[67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
{'Inputs': {'input1': {'ColumnNames': ['Age', 'Gender', 'Steroids', 'Antivirals', 'Anorexia', 'Big Liver', 'Palpable Spleen', 'Malaise', 'Ascites', 'Varices', 'Spiders', 'Jaun
dice', 'Dark Urine', 'Low Appetite', 'Itchy Skin', 'Pale Stools'], 'Values': [[67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1], [67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
1]]}}, 'GlobalParameters': {}}
1
```

This is the output that gets generated on the Jupyter notebook after answering all the question in the Telegram Bot. As we can see that the code runs successfully and we finally implemented our Telegram Bot in python.

(14). **Output in Telegram Bot:**



Search

RBS 12:48 PM
RBS joined Telegram 1

Upasana Singh 02:55 AM
Upasana Singh joined Telegram 1

Azure Bot 12:17 AM
The patient may not live long until he chan...

Yash GBU Thu
Yash GBU joined Telegram

Tanu Thu
Tanu joined Telegram 1

Sagar(Aryan) Tue
Sagar(Aryan) joined Telegram

Utkarsh DDU 2.05.21
Utkarsh DDU joined Telegram

Suvarna 29.04.21
Suvarna joined Telegram

Gurjeet Mam 26.04.21
Gurjeet Mam joined Telegram

Bhaswati 26.04.21

Azure Bot
bot

/start 12:15 AM
What is the age of the patient? 12:15 AM
67 12:16 AM
What is the gender of the patient? 12:16 AM
Male 12:16 AM
Does he take steroids? 12:16 AM
Yes 12:16 AM
Does he take antivirals? 12:16 AM
Yes 12:16 AM
Does he experience Anorexia? 12:16 AM
No 12:16 AM
Is the patient's liver big? 12:16 AM
Yes 12:16 AM

Write a message...

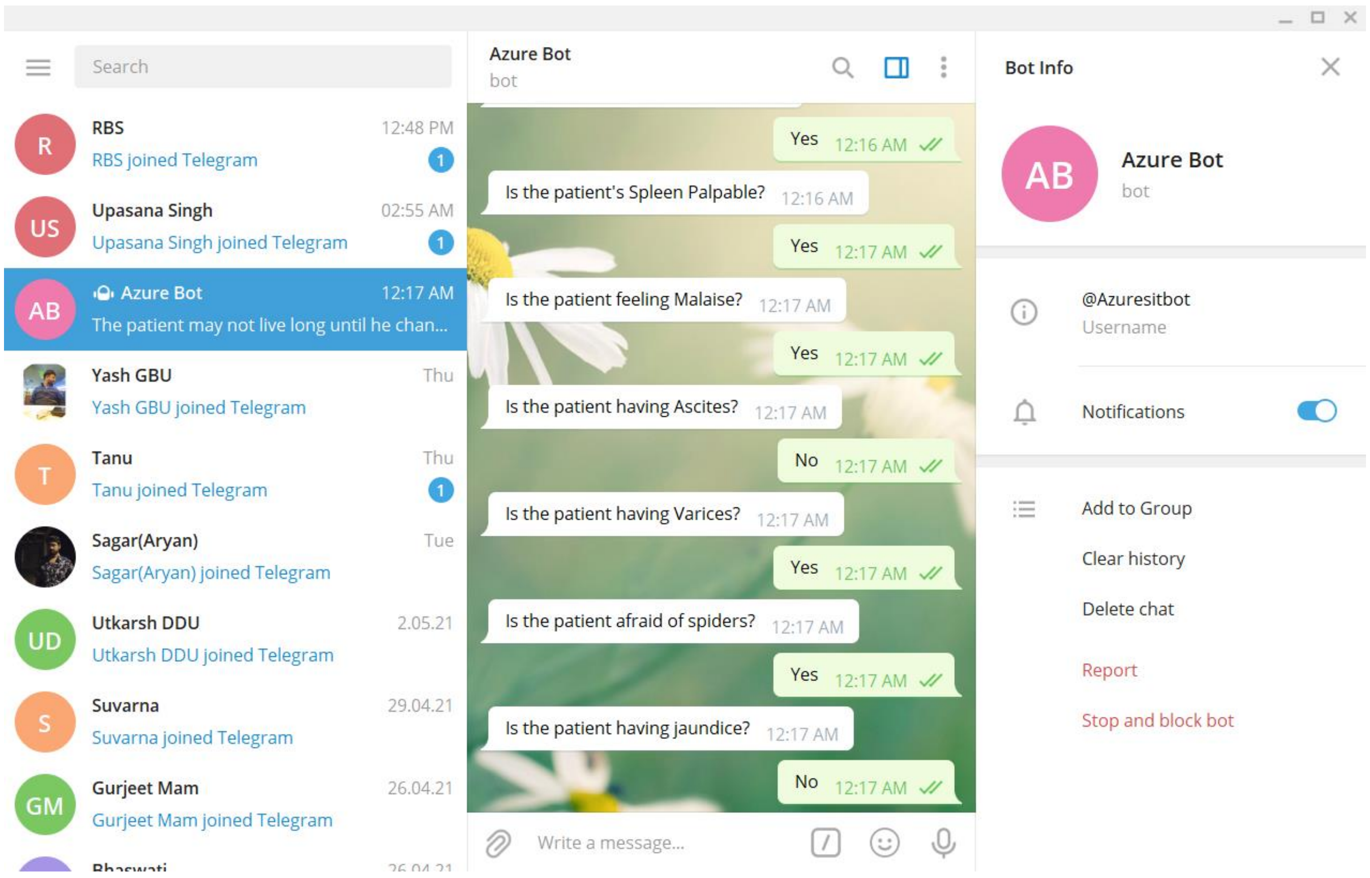
Bot Info

Azure Bot
bot

@Azuresitbot
Username

Notifications

Add to Group
Clear history
Delete chat
Report
Stop and block bot



Search

RBS 12:48 PM
RBS joined Telegram 1

Upasana Singh 02:55 AM
Upasana Singh joined Telegram 1

Azure Bot 12:17 AM
The patient may not live long until he chan...

Yash GBU Thu
Yash GBU joined Telegram

Tanu Thu
Tanu joined Telegram 1

Sagar(Aryan) Tue
Sagar(Aryan) joined Telegram

Utkarsh DDU 2.05.21
Utkarsh DDU joined Telegram

Suvarna 29.04.21
Suvarna joined Telegram

Gurjeet Mam 26.04.21
Gurjeet Mam joined Telegram

Bhaswati 26.04.21

Azure Bot
bot

Yes 12:16 AM
Is the patient's Spleen Palpable? 12:16 AM
Yes 12:17 AM
Is the patient feeling Malaise? 12:17 AM
Yes 12:17 AM
Is the patient having Ascites? 12:17 AM
No 12:17 AM
Is the patient having Varices? 12:17 AM
Yes 12:17 AM
Is the patient afraid of spiders? 12:17 AM
Yes 12:17 AM
Is the patient having jaundice? 12:17 AM
No 12:17 AM

Write a message...

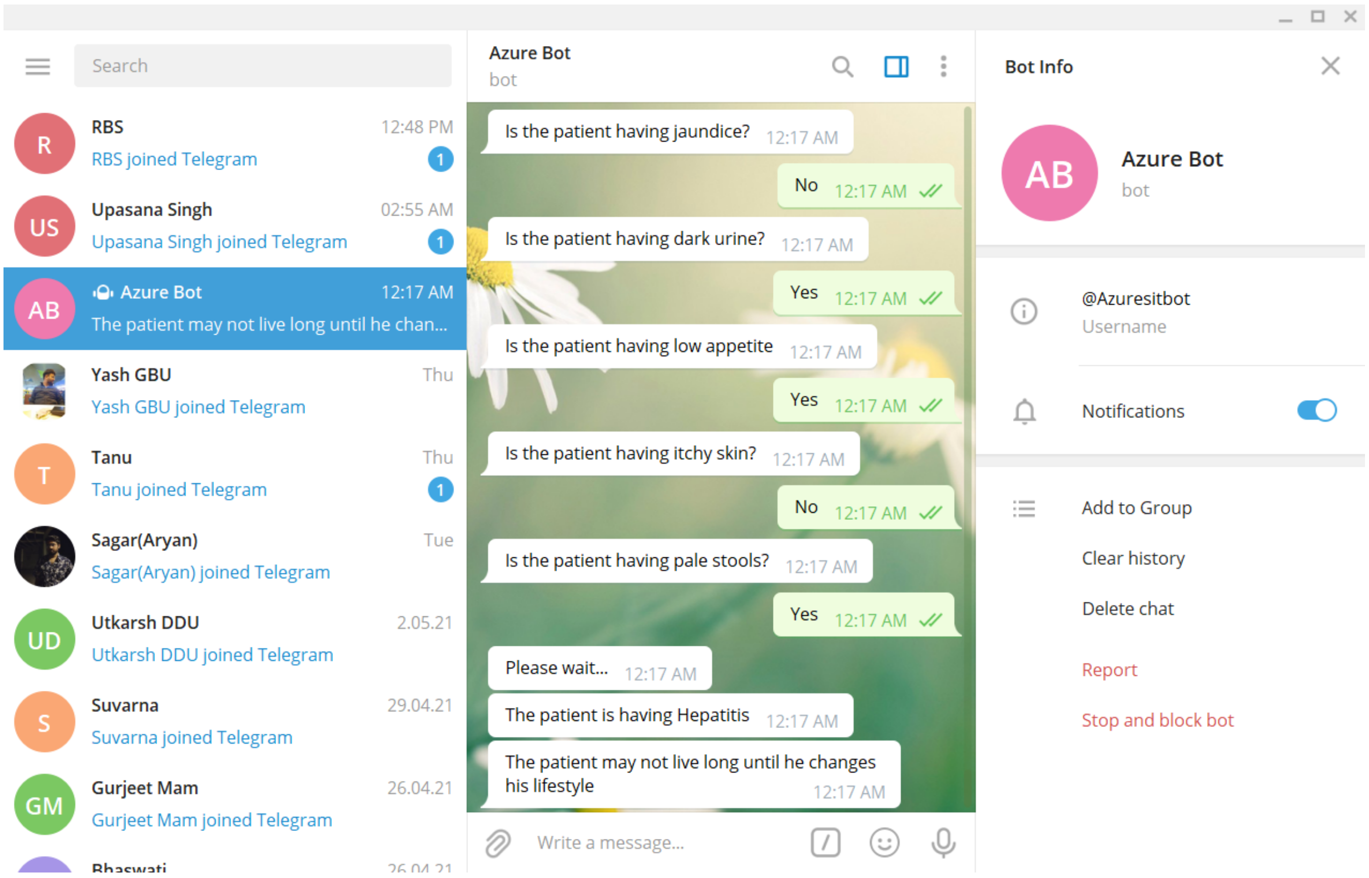
Bot Info

Azure Bot
bot

@Azuresitbot
Username

Notifications

Add to Group
Clear history
Delete chat
Report
Stop and block bot


Search

RBS 12:48 PM
RBS joined Telegram 1

Upasana Singh 02:55 AM
Upasana Singh joined Telegram 1

Azure Bot 12:17 AM
The patient may not live long until he chan...

Yash GBU Thu
Yash GBU joined Telegram

Tanu Thu
Tanu joined Telegram 1

Sagar(Aryan) Tue
Sagar(Aryan) joined Telegram

Utkarsh DDU 2.05.21
Utkarsh DDU joined Telegram

Suvarna 29.04.21
Suvarna joined Telegram

Gurjeet Mam 26.04.21
Gurjeet Mam joined Telegram

Bhaswati 26.04.21

Azure Bot
bot

Is the patient having jaundice? 12:17 AM
No 12:17 AM
Is the patient having dark urine? 12:17 AM
Yes 12:17 AM
Is the patient having low appetite 12:17 AM
Yes 12:17 AM
Is the patient having itchy skin? 12:17 AM
No 12:17 AM
Is the patient having pale stools? 12:17 AM
Yes 12:17 AM
Please wait... 12:17 AM
The patient is having Hepatitis 12:17 AM
The patient may not live long until he changes his lifestyle 12:17 AM

Write a message...

Bot Info
Azure Bot
bot
@Azuresitbot
Username
Notifications
Add to Group
Clear history
Delete chat
Report
Stop and block bot

This is the output on the Telegram App. We can see that our bot runs successfully by receiving and comparing the inputs on our deployed model and at the end we get the final output as "The patient is having Hepatitis. The patient may not live long enough until he changes his lifestyle".

## REFERENCES

1. https://d2l.deakin.edu.au/d2l/le/content/1029605/viewContent/5442865/View

2. https://d2l.deakin.edu.au/d2l/le/content/1029605/viewContent/5442866/View

3. https://d2l.deakin.edu.au/d2l/le/content/1029605/viewContent/5442868/View

4. https://d2l.deakin.edu.au/d2l/le/content/1029605/viewContent/5442880/View

# CODE

In [1]:
```
pip install python-telegram-bot==12.0
```

```
Requirement already satisfied: python-telegram-bot==12.0 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (12.0.0)
Requirement already satisfied: certifi in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (2020.12.5)
Requirement already satisfied: cryptography in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (3.4.7)
Requirement already satisfied: tornado>=5.1 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (6.1)
Requirement already satisfied: future>=0.16.0 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from python-telegram-bot==12.0) (0.18.2)
Requirement already satisfied: cffi>=1.12 in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from cryptography->python-telegram-bot==12.0) (1.14.5)
Requirement already satisfied: pycparser in c:\users\rahul\appdata\local\programs\python\python38\lib\site-packages (from cffi>=1.12->cryptography->python-telegram-bot==12.0)
(2.20)
Note: you may need to restart the kernel to use updated packages.
WARNING: You are using pip version 21.0.1; however, version 21.1.1 is available.
You should consider upgrading via the 'C:\Users\Rahul\AppData\Local\Programs\Python\Python38\python.exe -m pip install --upgrade pip' command.
```

In [24]:
```
from telegram.ext import Updater
from telegram.ext import CommandHandler
from  telegram.ext import CallbackQueryHandler
from telegram import InlineKeyboardButton
from telegram import InlineKeyboardMarkup
from telegram.ext import MessageHandler, Filters
import requests
import json
import urllib
from sklearn import preprocessing
import pandas as pd
```

In [25]:
```
#My token--------------------------
updater=Updater("1713442555:AAG1521MqTLfNsFdx6zbP2nqnrXCzm8-5sA")

# URL for the web service
scoring_uri = 'https://ussouthcentral.services.azureml.net/workspaces/a0597a9dd2f742f2a4159802a83137be/services/3c661031375441e59e86b18a1d086bfa/execute?api-version=2.0&details=true'
# If the service is authenticated, set the key or token
key = 'KoaOcxpKSU2oZlAgFAy56R/RjR4LWoa1L16dlYI9e2412Fjo10CZMP+UBAqWGnYgRqrrvZ3wvDBRyzvNL8jPLg=='
```

In [26]:
```
def start(bot , update):

    chat_id=update.message.chat_id

    global status
    status=0
    global question_id
    question_id=0
    global answer_id
    answer_id=0

    global responses
    responses=[]

    nextquestion(chat_id,bot)
```

In [27]:
```
def nextquestion(chat_id,bot):
#import pdb; pdb.set_trace()

    global status
    global question_id
    global answer_id
    print('nextquestion')
    print('status:',status,'question_id:',question_id,'answer_id:',answer_id)
#---------------------------------------------------------------------------------------------------------------
    if status==0:
        if question_id==0:
            text ="What is the age of the patient?"
            messager(chat_id,bot,text)
    elif status==1:
        if question_id==1:
            text ="What is the gender of the patient?"
            messager(chat_id,bot,text)
    elif status==2:
        if question_id==2:
            text ="Does he take steroids?"
            messager(chat_id,bot,text)
    elif status==3:
        if question_id==3:
            text ="Does he take antivirals?"
            messager(chat_id,bot,text)
    elif status==4:
        if question_id==4:
            text ="Does he experience Anorexia?"
            messager(chat_id,bot,text)
    elif status==5:
        if question_id==5:
            text ="Is the patient's liver big?"
            messager(chat_id,bot,text)
    elif status==6:
        if question_id==6:
            text ="Is the patient's Spleen Palpable?"
            messager(chat_id,bot,text)
    elif status==7:
        if question_id==7:
            text ="Is the patient feeling Malaise?"
            messager(chat_id,bot,text)
    elif status==8:
        if question_id==8:
            text ="Is the patient having Ascites?"
            messager(chat_id,bot,text)
    elif status==9:
        if question_id==9:
            text ="Is the patient having Varices?"
            messager(chat_id,bot,text)
    elif status==10:
        if question_id==10:
            text ="Is the patient afraid of spiders?"
            messager(chat_id,bot,text)
    elif status==11:
        if question_id==11:
            text ="Is the patient having jaundice?"
            messager(chat_id,bot,text)
    elif status==12:
        if question_id==12:
            text ="Is the patient having dark urine?"
            messager(chat_id,bot,text)
    elif status==13:
        if question_id==13:
            text ="Is the patient having low appetite"
            messager(chat_id,bot,text)
    elif status==14:
        if question_id==14:
            text ="Is the patient having itchy skin?"
            messager(chat_id,bot,text)
    elif status==15:
        if question_id==15:
            text ="Is the patient having pale stools?"
            messager(chat_id,bot,text)
```

In [28]:
```
def messager(chat_id,bot,text1):


    # the below function is explained in this link : https://python-telegram-bot.readthedocs.io/en/stable/telegram.bot.html#telegram.Bot.send_chat_action
    bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
    bot.send_message(chat_id = chat_id ,text = text1)
    global question_id
    question_id=question_id+1
```

In [29]:
```
def contQuestions(bot,update):
    print("contQuestions")

    chat_id=update.message.chat_id
    text=update.message.text

    global status
    global answer_id
    global question_id

    if answer_id == 0:
        global age
        age=text
        responses.append(age)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)

    elif answer_id == 1:
        global Gender
        Gender=text
        responses.append(Gender)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 2:
        global Steroids
        Steroids=text
        responses.append(Steroids)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 3:
        global Antivirals
        Antivirals=text
        responses.append(Antivirals)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 4:
        global Anorexia
        Anorexia=text
        responses.append(Anorexia)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 5:
        global BigL
        BigL=text
        responses.append(BigL)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 6:
        global SpleenP
        SpleenP=text
        responses.append(SpleenP)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
```

```
    elif answer_id == 7:
        global Malaise
        Malaise=text
        responses.append(Malaise)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 8:
        global Ascites
        Ascites=text
        responses.append(Ascites)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 9:
        global Varices
        Varices=text
        responses.append(Varices)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 10:
        global Spiders
        Spiders=text
        responses.append(Spiders)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
```

```
    elif answer_id == 11:
        global Jaundice
        Jaundice=text
        responses.append(Jaundice)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 12:
        global Durine
        Durine=text
        responses.append(Durine)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 13:
        global Lowapp
        Lowapp=text
        responses.append(Lowapp)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler
    elif answer_id == 14:
        global itchyskin
        itchyskin=text
        responses.append(itchyskin)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        nextquestion(chat_id,bot)
        #updater.dispatcher.remove_handler(consQuestion_handler)
    elif answer_id == 15:
        global Palestools
        Palestools=text
        responses.append(Palestools)
        print(responses)
        answer_id=answer_id+1
        status=status+1
        #updater.dispatcher.remove_handler(consQuestion_handler)
        end(chat_id,bot)
        updater.dispatcher.remove_handler(consQuestion_handler)
```

In [30]:

```
def end(chat_id,bot):
    print('end')
    bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
    bot.send_message(chat_id = chat_id ,text = "Please wait...")

    array1=responses
    global db
    X=db
    array2=["","","","","","","","","","","","","","","",""]
    array2[0]=int(array1[0])
    array2[1]=transformation(X,1,array1[1])
    array2[2]=transformation(X,2,array1[2])
    array2[3]=transformation(X,3,array1[3])
    array2[4]=transformation(X,4,array1[4])
    array2[5]=transformation(X,5,array1[5])
    array2[6]=transformation(X,6,array1[6])
    array2[7]=transformation(X,7,array1[7])
    array2[8]=transformation(X,8,array1[8])
    array2[9]=transformation(X,9,array1[9])
    array2[10]=transformation(X,10,array1[10])
    array2[11]=transformation(X,11,array1[11])
    array2[12]=transformation(X,12,array1[12])
    array2[13]=transformation(X,13,array1[13])
    array2[14]=transformation(X,14,array1[14])
    array2[15]=transformation(X,15,array1[15])
    print(array2)
```

```
    data = {

        "Inputs": {

            "input1":
                {
                    "ColumnNames": ["Age", "Gender", "Steroids", "Antivirals", "Anorexia", "Big Liver",
                    "Palpable Spleen", "Malaise", "Ascites", "Varices", "Spiders", "Jaundice",
                    "Dark Urine", "Low Appetite","Itchy Skin","Pale Stools"],
                    "Values": [array2,array2,]
                }, },
        "GlobalParameters": {
        }
    }
    print(data)
    body = str.encode(json.dumps(data))
    # body=json.dumps(data)
    # print(data['Inputs']['input1'])

     # print(body)

    headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer ' + key)}

     # req = urllib.Request(url, body, headers)
    req = urllib.request.Request(scoring_uri, body, headers)
    try:

# If you are using Python 3+, replace urllib2 with urllib.request in the above code:
        response = urllib.request.urlopen(req)
        result = response.read()

        encoding = response.info().get_content_charset('utf-8')
        JSON_object = json.loads(result.decode(encoding))
        print(JSON_object['Results']['output1']['value']['Values'][0][0])
        if JSON_object['Results']['output1']['value']['Values'][0][0] == '1':
            bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
            bot.send_message(chat_id = chat_id ,text = "The patient is having Hepatitis")
            bot.send_message(chat_id = chat_id ,text = "The patient may not live long until he changes his lifestyle")


        elif JSON_object['Results']['output1']['value']['Values'][0][0] == '0':
            bot.sendChatAction(chat_id, action = 'typing' , timeout=None)
            bot.send_message(chat_id = chat_id ,text = "The patient is not having Hepatitis")
            bot.send_message(chat_id = chat_id ,text = "The Patient is free to live the hospital")

    except urllib.error.HTTPError as error:
        print("The request failed with status code: " + str(error.code))

         # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
        print(error.info())

        print(json.loads(error.read()))
```

In [ ]:

```
url = r"D:\Deakin University (Applied AI (Professional)\SIT788 (Engineering AI Solutions)\Tasks and Assignments\Task- 6.1P\Hepatitis\New\hepatitisdata.csv"
dataset = pd.read_csv(url)
df = dataset.infer_objects()
    #print(df.dtypes[0])
for ij in range (0,16):
    if df.dtypes[ij]!='int64':
        dataset.iloc[:,ij] = dataset.iloc[:,ij].map(lambda x: x.strip())
            #print(dataset.iloc[:,ij])
array = dataset.values
X = array[:,0:16]

global db
db=X

# listen to get from user

command_start= CommandHandler('start',start)
consQuestion_handler = MessageHandler(Filters.text, contQuestions)

#updater
updater.dispatcher.add_handler(command_start,)
updater.dispatcher.add_handler(consQuestion_handler)

print('listening.....')

updater.start_polling()
updater.idle()
```

```
listening.....
nextquestion
status: 0 question_id: 0 answer_id: 0
contQuestions
['67']
nextquestion
status: 1 question_id: 1 answer_id: 1
contQuestions
['67', 'Male']
nextquestion
status: 2 question_id: 2 answer_id: 2
contQuestions
['67', 'Male', 'Yes']
nextquestion
status: 3 question_id: 3 answer_id: 3
contQuestions
['67', 'Male', 'Yes', 'Yes']
nextquestion
status: 4 question_id: 4 answer_id: 4
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No']
nextquestion
status: 5 question_id: 5 answer_id: 5
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
status: 6 question_id: 6 answer_id: 6
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 7 question_id: 7 answer_id: 7
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes']
nextquestion
status: 8 question_id: 8 answer_id: 8
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No']
nextquestion
status: 9 question_id: 9 answer_id: 9
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
```

```
status: 10 question_id: 10 answer_id: 10
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 11 question_id: 11 answer_id: 11
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No']
nextquestion
status: 12 question_id: 12 answer_id: 12
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
nextquestion
status: 13 question_id: 13 answer_id: 13
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes']
nextquestion
status: 14 question_id: 14 answer_id: 14
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No']
nextquestion
status: 15 question_id: 15 answer_id: 15
contQuestions
['67', 'Male', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
end
['Female', 'Male']
Male
1
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
```

```
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
['No', 'Yes']
Yes
1
['No', 'Yes']
No
0
['No', 'Yes']
Yes
1
[67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
{'Inputs': {'input1': {'ColumnNames': ['Age', 'Gender', 'Steroids', 'Antivirals', 'Anorexia', 'Big Liver', 'Palpable Spleen', 'Malaise', 'Ascites', 'Varices', 'Spiders', 'Jaun
dice', 'Dark Urine', 'Low Appetite', 'Itchy Skin', 'Pale Stools'], 'Values': [[67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1], [67, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
1]]}}, 'GlobalParameters': {}}
1
```