LP1 Assignment AIR A2

Expert System

Date - 29th October, 2020.

Assignment Number - AIR A3

Title

Expert System

Problem Definition

Implement an Expert System for Medical Diagnosis of diseases based on adequate symptoms.

Learning Objectives

- To learn and implement an expert system

Learning Outcomes

I will be able to learn and implement Expert System for Medical Diagnosis

Software Packages and Hardware Apparatus Used

- Operating System : 64-bit Ubuntu 18.04
- Programming Language : Python 3
- Jupyter Notebook Environment : Google Colaboratory
- Python3 Library : experta

Programmers' Perspective

S = {s; e; X; Y; Fme; Ff; DD; NDD}

s = start state

- s = Set of Symptoms for a set of diseases each

e = end state

- e = Final diagnosis

X = {X1}

- X1 = {Di | 0<=i<13}
- Di is the set of symptoms for ith Disease

Y = {Y1}

- Y1 = {Final Diagnosis}

Fme = {function to perform Fact based classification}

Ff = {f1,f2,f3}

where

- f1 = function to find input symptoms
- f2 = function to find states
- f3 = function to display diagnosis

DD = Set of Symptoms for a set of diseases each

NDD = No non deterministic data

Concepts related Theory

Expert Systems:

- Diagnostic expert-based systems are computer systems that seek to emulate the diagnostic decision-making ability of human experts.
- Medical expert systems generally include two components:
    a. Knowledge Base (KB) - It encapsulates the evidence-based medical knowledge that is curated by experts
    b. Rule-based inference engine - It is devised by the expert, which operates on the knowledge base to generate a differential diagnosis.
- Diagnostic knowledge bases generally consist of diseases, findings (i.e. symptoms, signs, history, or lab results), and their relationships.
- In many cases, they explicitly lay out the relationships between a set of findings and the things that cause them (diseases).

- For example, a KB might include influenza and show its relationships with fever, coughing, and congestion.

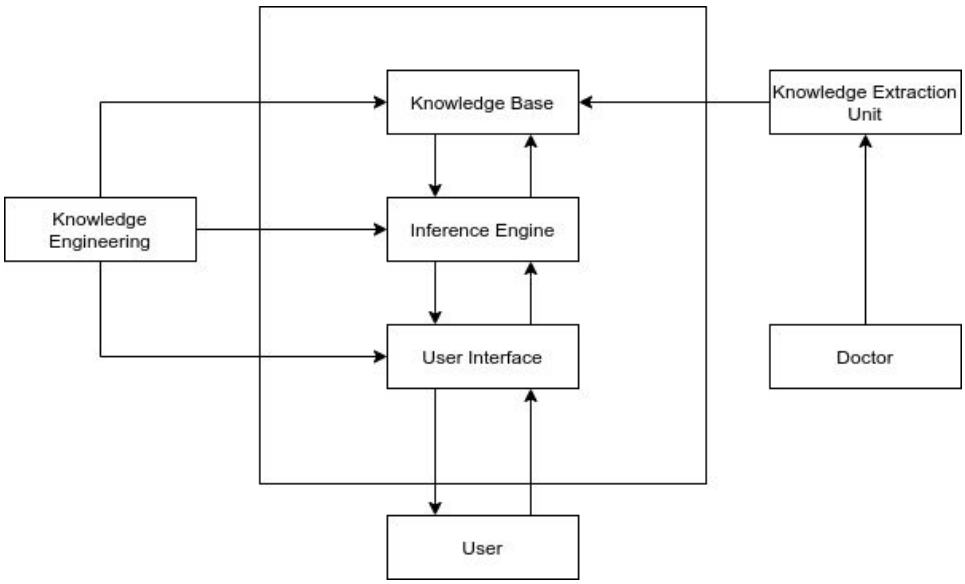
Inference Engine:

The inference engine is based on forward and backward chaining, examining the knowledge base (disease symptoms) for information that matches the user's query (kind of disease).

Knowledge Base Design:

The knowledge domain was got from facts of a collection of data about the types of symptoms and diseases to be isolated and identified, the identification methods, the expected results.

Data elicited for the isolation, identification of symptoms and possible recommendations on susceptibility patterns makes the knowledge base which was modeled into frames at the different levels of the decision trees and using the ”IF—THEN” production rules , quick deductions are made.



Perception

Representing Diseases and Symptoms

Label	Disease		Label	Symptom
x1	Jaundice		s1	Headache
x2	Alzheimers		s2	Back pain
x3	Arthritis		s3	Chest pain
x4	Tuberculosis		s4	Cough
x5	Asthma		s5	Fainting
x6	Sinusitis		s6	Sore throat
x7	Epilepsy		s7	Fatigue
x8	Heart Disease		s8	Restlessness
x9	Diabetes		s9	Low body temp
x10	Glaucoma		s10	Fever
x11	Hyperthyroidism		s11	Sunken eyes
x12	Heat Stroke		s12	Nausea
x13	Hypothermia		s13	Blurred vision

Truth Table

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
x1	0	0	0	0	0	0	1	0	0	1	0	1	0
x2	0	0	0	0	0	0	0	1	0	0	0	0	0
x3	0	1	0	0	0	0	1	0	0	0	0	0	0
x4	0	0	1	1	0	0	0	0	0	1	0	0	0
x5	0	0	1	1	0	0	0	1	0	0	0	0	0
x6	1	0	0	1	0	1	0	0	0	1	0	0	0
x7	0	0	0	0	0	0	1	0	0	0	0	0	0
x8	0	0	1	0	0	0	0	0	0	0	0	1	0
x9	0	0	0	0	0	0	1	0	0	0	0	1	1
x10	1	0	0	0	0	0	0	0	0	0	0	1	1
x11	0	0	0	0	0	0	1	0	0	0	0	1	0
x12	1	0	0	0	0	0	0	0	0	1	0	1	0
x13	0	0	0	0	1	0	0	0	1	0	0	0	0

Cognition

Example of Rule in Knowledge Base :

Disease (Patient, Jaundice):-

Symptom (Patient, Fatigue),

Symptom (Patient, Fever),

Symptom (Patient, Nausea)

s1 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ x7 ∧ ~x8 ∧ ~x9 ∧ x10 ∧ ~x11 ∧ x12 ∧ ~x13

s2 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s3 → ~x1 ∧ x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s4 → ~x1 ∧ ~x2 ∧ x3 ∧ x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ ~x8 ∧ ~x9 ∧ x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s5 → ~x1 ∧ ~x2 ∧ x3 ∧ x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s6 → ~x1 ∧ ~x2 ∧ ~x3 ∧ x4 ∧ ~x5 ∧ x6 ∧ ~x7 ∧ ~x8 ∧ ~x9 ∧ x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s7 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ ~x12 ∧ ~x13

s8 → ~x1 ∧ ~x2 ∧ x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ x12 ∧ ~x13

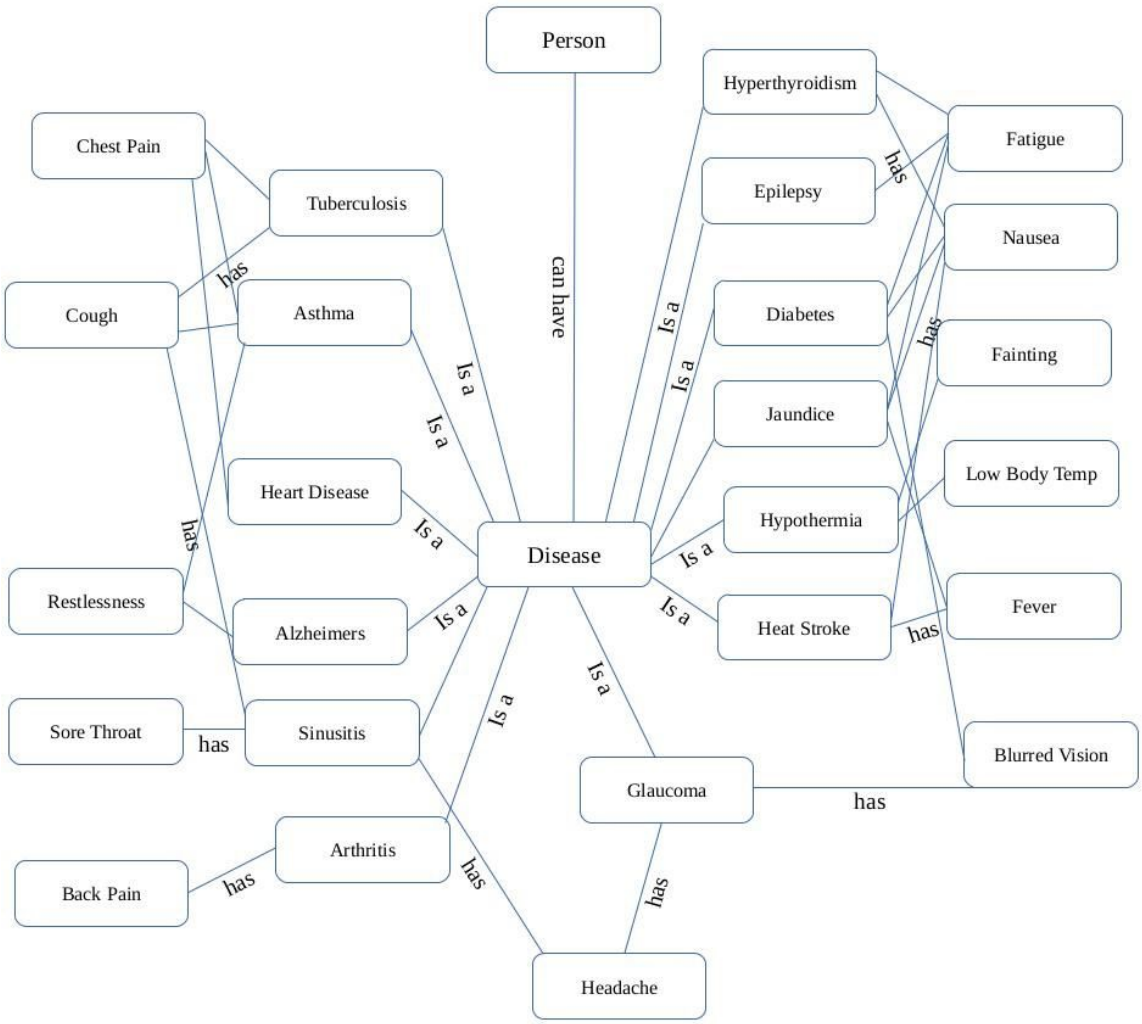
s9 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ x12 ∧ x13

s10 → x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ x12 ∧ x13

s11 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ x7 ∧ ~x8 ∧ ~x9 ∧ ~x10 ∧ ~x11 ∧ x12 ∧ ~x13

s12 → x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ ~x5 ∧ ~x6 ∧ ~x7 ∧ ~x8 ∧ ~x9 ∧ x10 ∧ ~x11 ∧ x12 ∧ ~x13

s13 → ~x1 ∧ ~x2 ∧ ~x3 ∧ ~x4 ∧ x5 ∧ ~x6 ∧ ~x7 ∧ ~x8 ∧ x9 ∧ ~x10 ∧ ~x11 ∧ ~x12 ∧ ~x13



Action

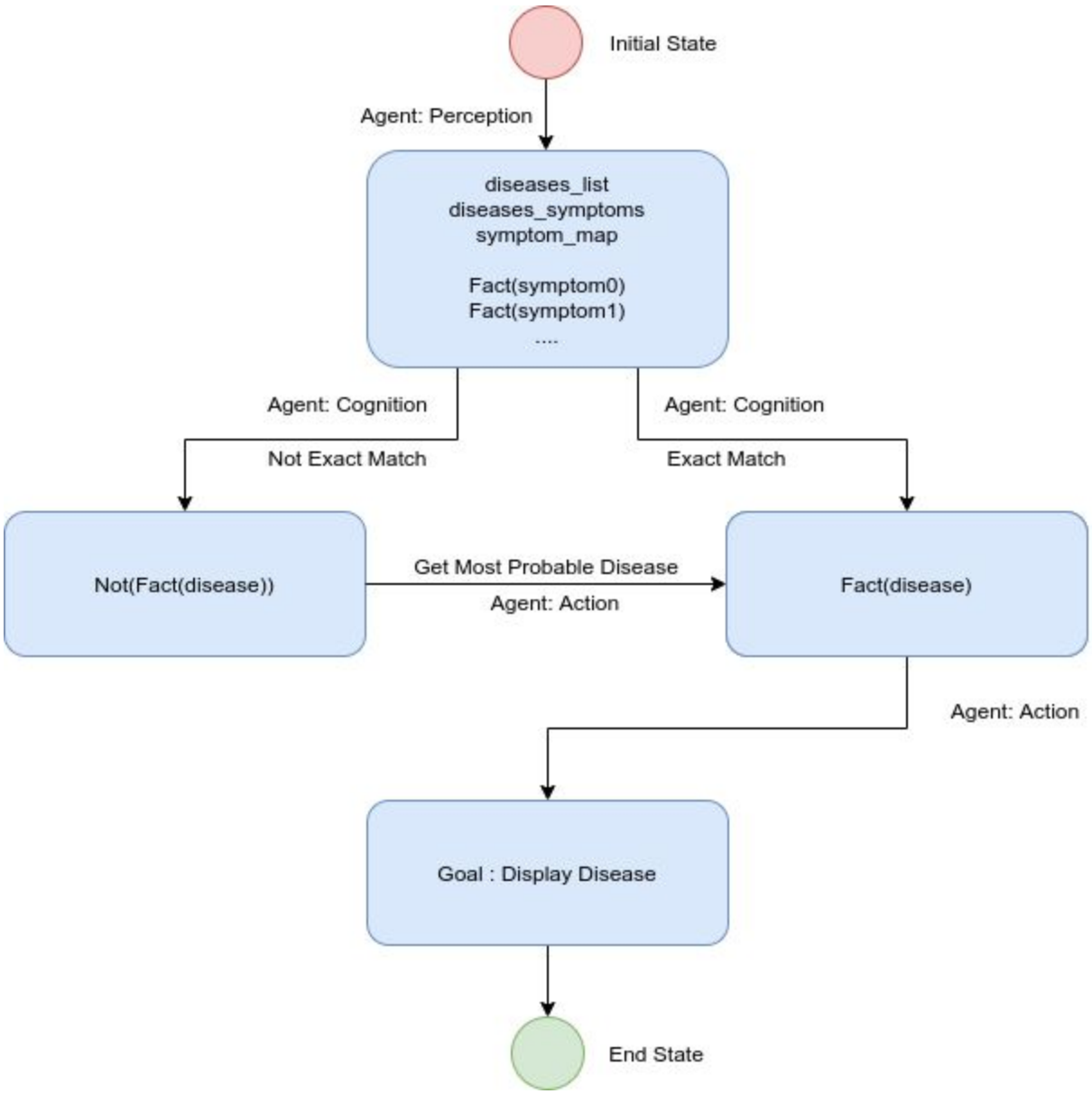
Fire Rule to Display Disease iff a Fact of Disease has been asserted in Cognition

Otherwise Fire Rule to display the most probable disease.

Goal State

Display Name of the disease or Display no disease along with most probable disease

State Diagram



Source Code

```
#Import statement/s
```

```
from experta import *

#Constant String Values
yes = "y"
no = "n"
find_disease = "find_disease"

#Global Variables
diseases_list = []
diseases_symptoms = []
symptom_map = {}

#Set up the global variables
def perception_preprocess():

 #global variables
 global diseases_list,diseases_symptoms,symptom_map
  #fetch the list of diseases
 diseases = open("/content/drive/My
Drive/medical_expert_system/diseases.txt")
 diseases_t = diseases.read()
 diseases_list = diseases_t.split("\n")
 diseases.close()
  #fetch the symptom for each disease
 for disease in diseases_list:

   #Open the file containing symptom for a disease
   disease_s_file = open("/content/drive/My
Drive/medical_expert_system/Disease symptoms/" + disease + ".txt")
   disease_s_data = disease_s_file.read()

   #Fetch the symptom as a list
   s_list = disease_s_data.split("\n")
   s_list = [yes if x=="yes" else no for x in s_list]
   diseases_symptoms.append(s_list)

   #Map the list to a disease
```

```
    symptom_map[str(s_list)] = disease

    #Close the file
    disease_s_file.close()
 #Expert System class
class MedicalExpertSystem_PCAG(KnowledgeEngine):

 #Prerequisite for KnowledgeEngine to run - find_disease action
 @DefFacts()
 def _initial_action(self):
    print("\nPYTHON MEDICAL EXPERT SYSTEM\nAnswer the following questions
in '{yes}' or '{no}'.\nDo you feel any of the following
symptoms:\n".format(yes=yes,no=no))
    yield Fact(action="find_disease")

 '''PERCEPTION'''
 #Declaring Symptom
 @Rule(Fact(action=find_disease), NOT(Fact(headache=W())),salience = 1)
 def perception_0(self):
    self.declare(Fact(headache=input("headache: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(back_pain=W())),salience = 1)
 def perception_1(self):
    self.declare(Fact(back_pain=input("back pain: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(chest_pain=W())),salience = 1)
 def perception_2(self):
    self.declare(Fact(chest_pain=input("chest pain: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(cough=W())),salience = 1)
 def perception_3(self):
    self.declare(Fact(cough=input("cough: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(fainting=W())),salience = 1)
 def perception_4(self):
    self.declare(Fact(fainting=input("fainting: ")))
```

```
 @Rule(Fact(action=find_disease), NOT(Fact(fatigue=W())),salience = 1)
 def perception_5(self):
   self.declare(Fact(fatigue=input("fatigue: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(sunken_eyes=W())),salience = 1)
 def perception_6(self):
   self.declare(Fact(sunken_eyes=input("sunken eyes: ")))
  @Rule(Fact(action=find_disease), NOT(Fact(low_body_temp=W())),salience =
1)
 def perception_7(self):
   self.declare(Fact(low_body_temp=input("low body temperature: ")))
  @Rule(Fact(action=find_disease), NOT(Fact(restlessness=W())),salience =
1)
 def perception_8(self):
   self.declare(Fact(restlessness=input("restlessness: ")))
  @Rule(Fact(action=find_disease), NOT(Fact(sore_throat=W())),salience =
1)
 def perception_9(self):
   self.declare(Fact(sore_throat=input("sore throat: ")))
  @Rule(Fact(action=find_disease), NOT(Fact(fever=W())),salience = 1)
 def perception_10(self):
   self.declare(Fact(fever=input("fever: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(nausea=W())),salience = 1)
 def perception_11(self):
   self.declare(Fact(nausea=input("nausea: ")))

 @Rule(Fact(action=find_disease), NOT(Fact(blurred_vision=W())),salience =
1)
 def perception_12(self):
   self.declare(Fact(blurred_vision=input("blurred vision: ")))

 '''COGNITION'''
 #Finding an exact match of symptoms for the disease

@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact(
```

```
fatigue=yes),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=yes),
Fact(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=no))
 def cognition_0(self):
   self.declare(Fact(disease="Jaundice"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact(
fatigue=no),Fact(restlessness=yes),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_1(self):
   self.declare(Fact(disease="Alzheimers"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=yes),Fact
(chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact
(fatigue=yes),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),
Fact(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_2(self):
   self.declare(Fact(disease="Arthritis"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=yes),Fact(cough=yes),Fact(fainting=no),Fact(sore_throat=no),Fac
t(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=yes)
,Fact(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_3(self):
   self.declare(Fact(disease="Tuberculosis"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=yes),Fact(cough=yes),Fact(fainting=no),Fact(sore_throat=no),Fac
t(fatigue=no),Fact(restlessness=yes),Fact(low_body_temp=no),Fact(fever=no)
,Fact(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_4(self):
   self.declare(Fact(disease="Asthma"))
```

```
@Rule(Fact(action=find_disease),Fact(headache=yes),Fact(back_pain=no),Fact
(chest_pain=no),Fact(cough=yes),Fact(fainting=no),Fact(sore_throat=yes),Fa
ct(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=yes
),Fact(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_5(self):
   self.declare(Fact(disease="Sinusitis"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact(
fatigue=yes),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_6(self):
   self.declare(Fact(disease="Epilepsy"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=yes),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact
(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=no))
 def cognition_7(self):
   self.declare(Fact(disease="Heart Disease"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact(
fatigue=yes),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=yes))
 def cognition_8(self):
   self.declare(Fact(disease="Diabetes"))


@Rule(Fact(action=find_disease),Fact(headache=yes),Fact(back_pain=no),Fact
(chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact
(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=yes))
```

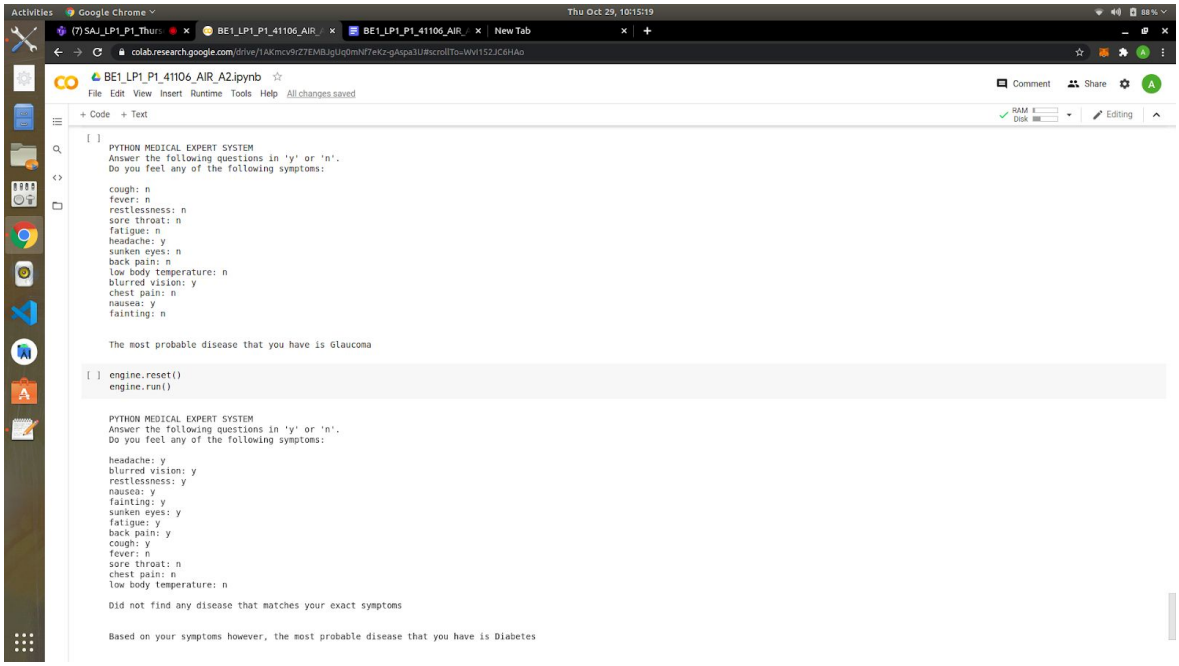
```
 def cognition_9(self):
   self.declare(Fact(disease="Glaucoma"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact(
fatigue=yes),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=no),F
act(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=no))
 def cognition_10(self):
   self.declare(Fact(disease="Hyperthyroidism"))


@Rule(Fact(action=find_disease),Fact(headache=yes),Fact(back_pain=no),Fact
(chest_pain=no),Fact(cough=no),Fact(fainting=no),Fact(sore_throat=no),Fact
(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=no),Fact(fever=yes),
Fact(sunken_eyes=no),Fact(nausea=yes),Fact(blurred_vision=no))
 def cognition_11(self):
   self.declare(Fact(disease="Heat Stroke"))


@Rule(Fact(action=find_disease),Fact(headache=no),Fact(back_pain=no),Fact(
chest_pain=no),Fact(cough=no),Fact(fainting=yes),Fact(sore_throat=no),Fact
(fatigue=no),Fact(restlessness=no),Fact(low_body_temp=yes),Fact(fever=no),
Fact(sunken_eyes=no),Fact(nausea=no),Fact(blurred_vision=no))
 def cognition_12(self):
   self.declare(Fact(disease="Hypothermia"))

 '''ACTION'''
 #Rule to be fired if disease is found
 @Rule(
   Fact(action=find_disease),
   Fact(disease=MATCH.disease),

   salience = -998
 )
 def action_0(self, disease):
```

```
   goal = "\n\nThe most probable disease that you have is
{disease}".format(disease=disease)
   print(goal)

 '''ACTION'''
 #Rule to be fired if the disease is not found
 @Rule(
    Fact(action=find_disease),

   Fact(headache=MATCH.headache),
   Fact(back_pain=MATCH.back_pain),
   Fact(chest_pain=MATCH.chest_pain),
   Fact(cough=MATCH.cough),
   Fact(fainting=MATCH.fainting),
   Fact(sore_throat=MATCH.sore_throat),
   Fact(fatigue=MATCH.fatigue),
   Fact(low_body_temp=MATCH.low_body_temp),
   Fact(restlessness=MATCH.restlessness),
   Fact(fever=MATCH.fever),
   Fact(sunken_eyes=MATCH.sunken_eyes),
   Fact(nausea=MATCH.nausea),
   Fact(blurred_vision=MATCH.blurred_vision),

   NOT(Fact(disease=W())),

   salience = -999
 )
 def action_1(self,headache, back_pain, chest_pain, cough, fainting,
sore_throat, fatigue, restlessness,low_body_temp ,fever ,sunken_eyes
,nausea ,blurred_vision):
   '''GOAL'''

   #Find Diseases whose symptom matches the highest number
   lis = [headache, back_pain, chest_pain, cough, fainting, sore_throat,
fatigue, restlessness,low_body_temp ,fever ,sunken_eyes ,nausea
,blurred_vision]
   max_count = 0
```

```
    max_disease = ""
    for key,val in symptom_map.items():
      count = 0
      temp_list = eval(key)
      for j in range(0,len(lis)):
        if(temp_list[j] == lis[j] and lis[j] == yes):
          count = count + 1
      if count > max_count:
        max_count = count
        max_disease = val
    '''GOAL'''
    goal = "\nDid not find any disease that matches your exact
symptoms\n\nBased on your symptoms however, the most probable disease that
you have is {disease}\n".format(disease=max_disease)
    print(goal)

if __name__ == '__main__':
 preprocess()
 engine = MedicalExpertSystem_PCAG()
 engine.reset()
 engine.run()
```

Output Screenshots

1) Exact Match
2) Most Probable Disease



Conclusion

I have successfully designed and implemented an Expert System for Medical System