Tomeh Lara – Group 407

Knowledge Representation and Reasoning

Project 2

Question 1

SLD Resolution stands for Selected literals, Linear pattern, over Definite clauses.

Rules:

1. In case that the patient exceeds eight hours of sleep daily, he feels tired.
2. In case that the patient is exhausted and tired, then he has a bad health.
3. In case that the patient feels nauseous and he is tired, he should have a rest from work.
4. In case that the patient has a bad health and he made previous surgeries, then he will receive a free medical review with the doctor on duty.

Questions:

1. What is the number of hours does the patient sleep every day? (number)
2. Does the patient feel exhausted? (The answer is yes or no)
3. Does the patient feel nauseous? (The answer is yes or no)
4. Has the patient had any previous surgeries? (The answer is yes or no)

The aim is defining if the patient will get free medical review with the doctor or not.

The knowledgebase written as positive horn propositional clauses, (sld.txt):

[[(sleep_more_than_8_hours), tired], [(exhausted), (tired), bad_health], [(feels_nauseous), (tired), take_rest], [(bad_health), (previous_surgeries), receive_free_medical_review]].

The solution procedure is a restricted form of resolution, where each new clause is a resolvent of the previous clause and a clause from the original set S. This version of resolution is sufficient for Horn clauses. From C5, slide 17.
Two approach can used to implement the SLD resolution, backward chaining and forward chaining.

Question 2

Vagueness

Rules written in verbose format:

1. If house location is remote or house space is small, then the price is low.
2. If house location is nearby and house space is medium, then the price is suitable.
3. If house location is close and house space is wide, then the price is high.
4. If house location is nearby and house space is wide, then the price is suitable.

Rules written in symbolic format:

1. (house_location==remote) | (house_space==small) => (price=low)
2. (house_location==nearby) & (house_space==medium) => (price=suitable)
3. (house_location==close) & (house_space==wide) => (price=high)
4. (house_location==nearby) & (house_space==wide) => (price=suitable)

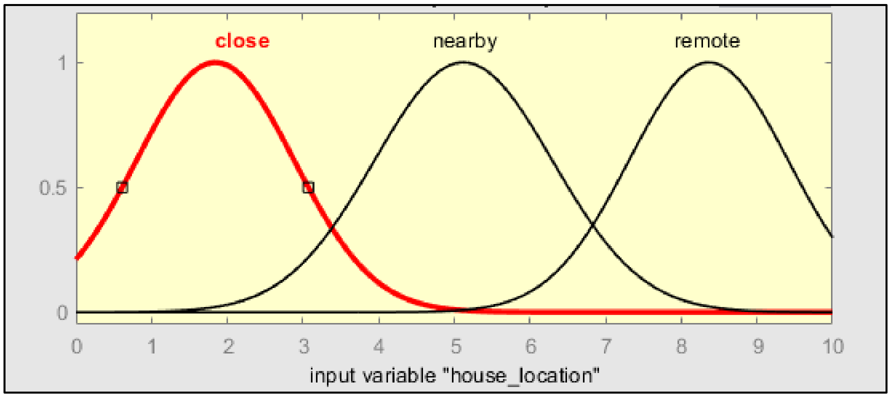
Fuzzy Logic is used to design systems that do not have clear parameters in order to solve problems in an efficient manner and make the best decisions.

The problem solved in two ways:

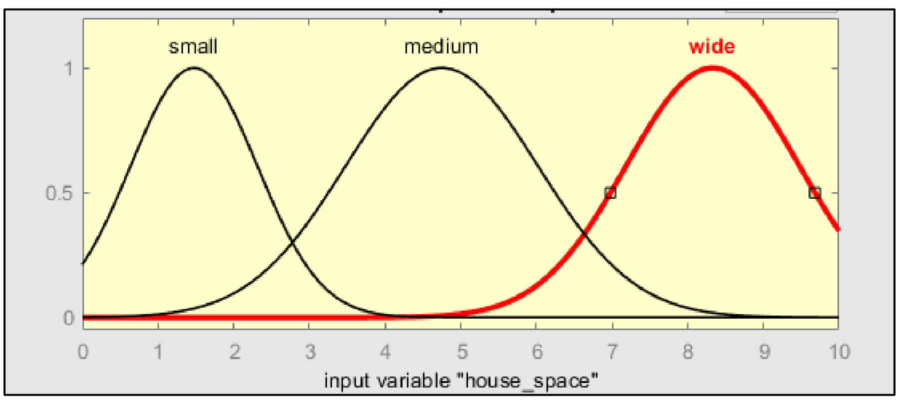
- Using Matlab.
- Using Python, solved without using any external helper libraries

System variables degree curves in MATLAB:

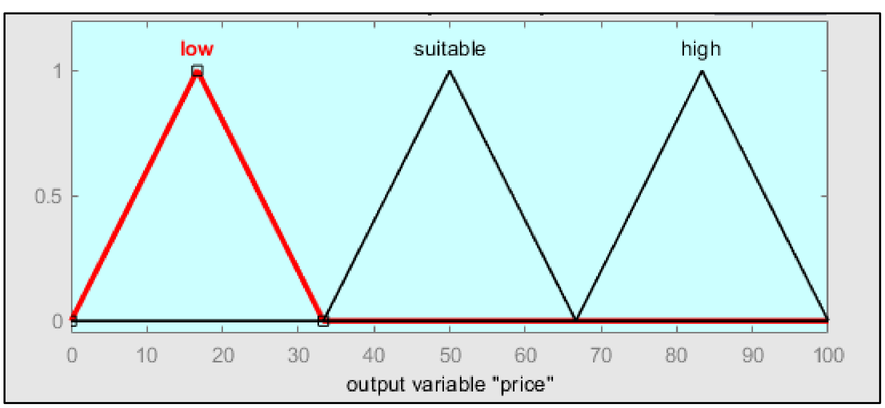
House location from 0 to 10, Since the lower the value, the closer the house is to the center.



House space from 0 to 10, Since the higher the value, the larger the house area.



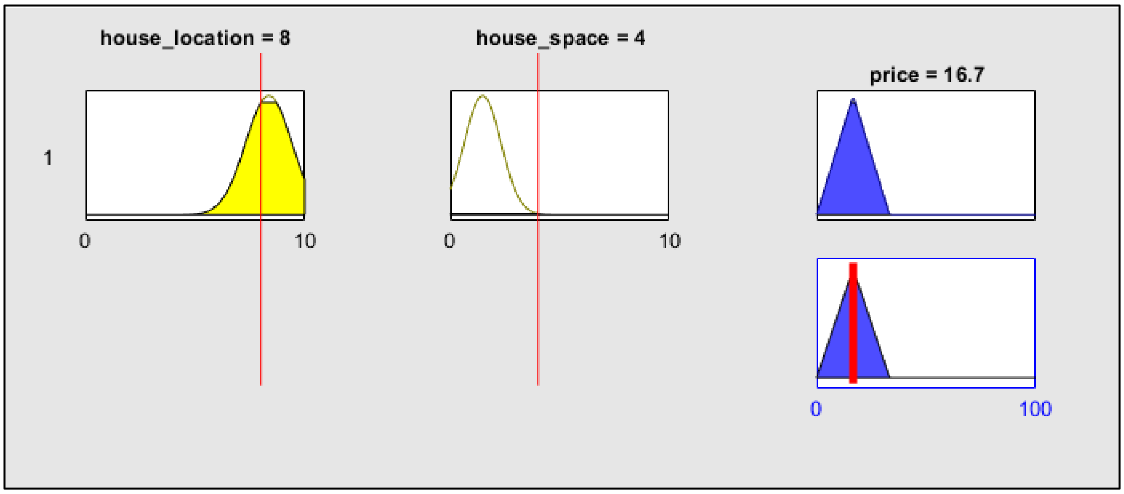
House price as output variable, from 0 to 100, since the higher the value, the more expensive the house.



Degree curve for some examples, applying some rules of the system:

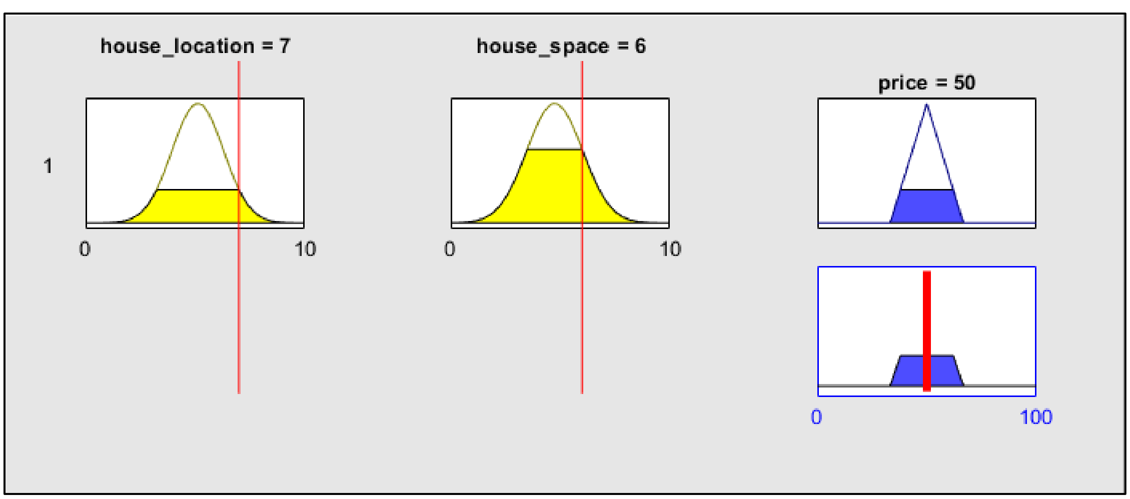
House location = 8, house space = 4, predicted price = 16.7

The house is far from the city center, so its price is low, regardless of its size, Since the price chart is completely full, the prediction is correct.



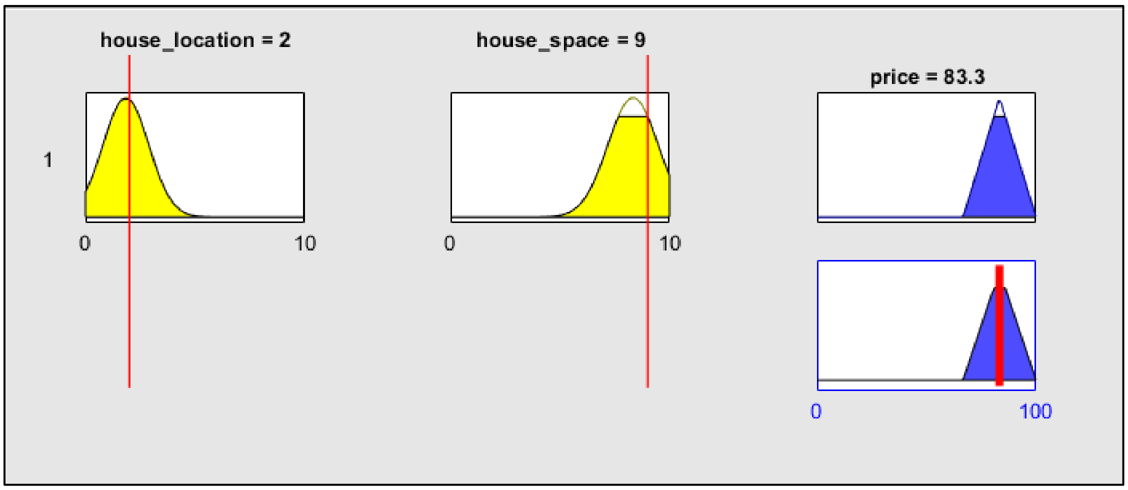
House location = 7, house space = 6, predicted price = 50

Since the house is located in the middle and its area is considered average, neither too big nor too small, there is a 50% chance that the price is somewhat appropriate.



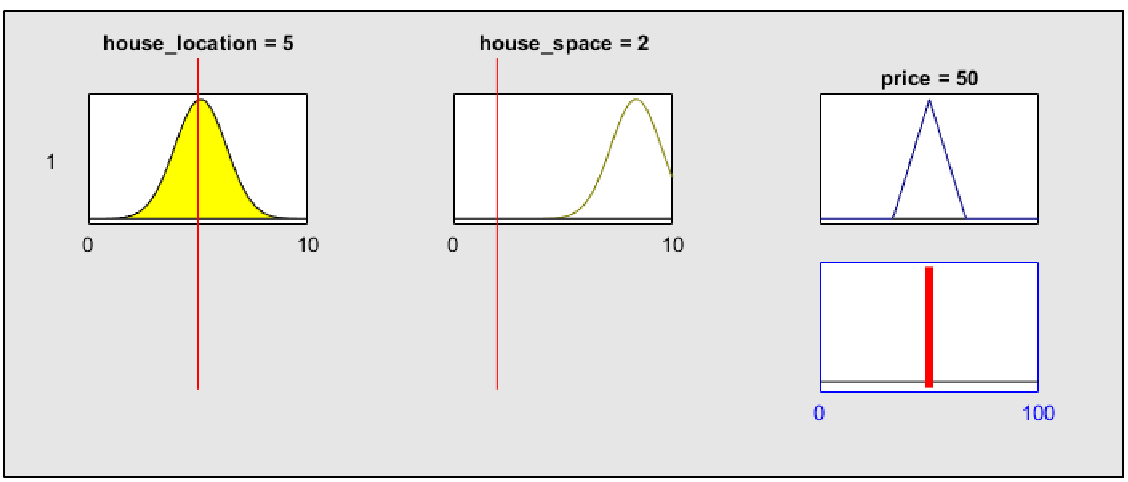
House location = 2, house space = 9, predicted price = 83.3

Since the house is close to the center and its area is large, the premise that it would be expensive is high.

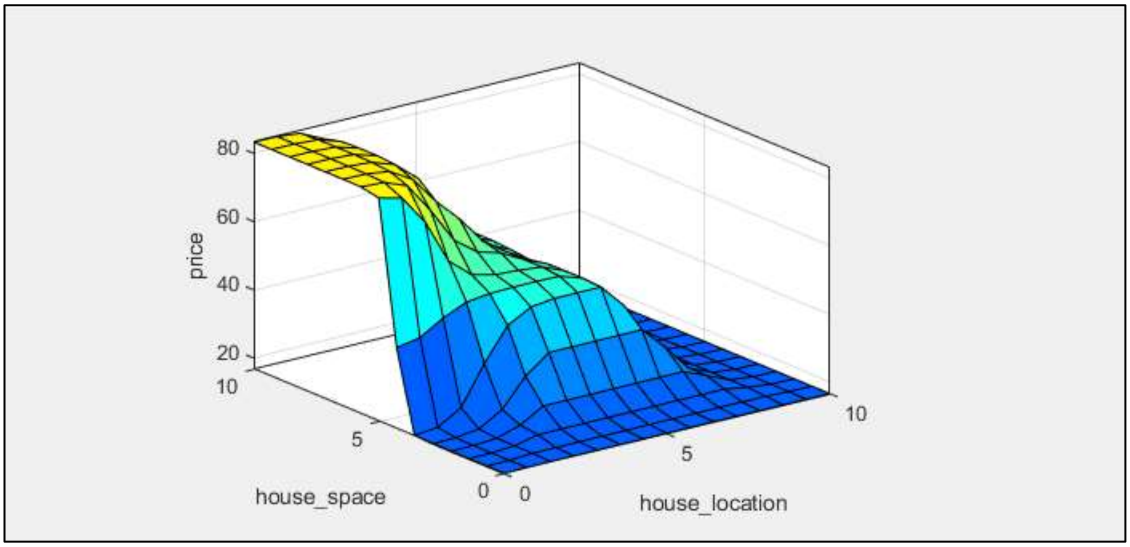


House location = 5, house space = 2, predicted price = 50

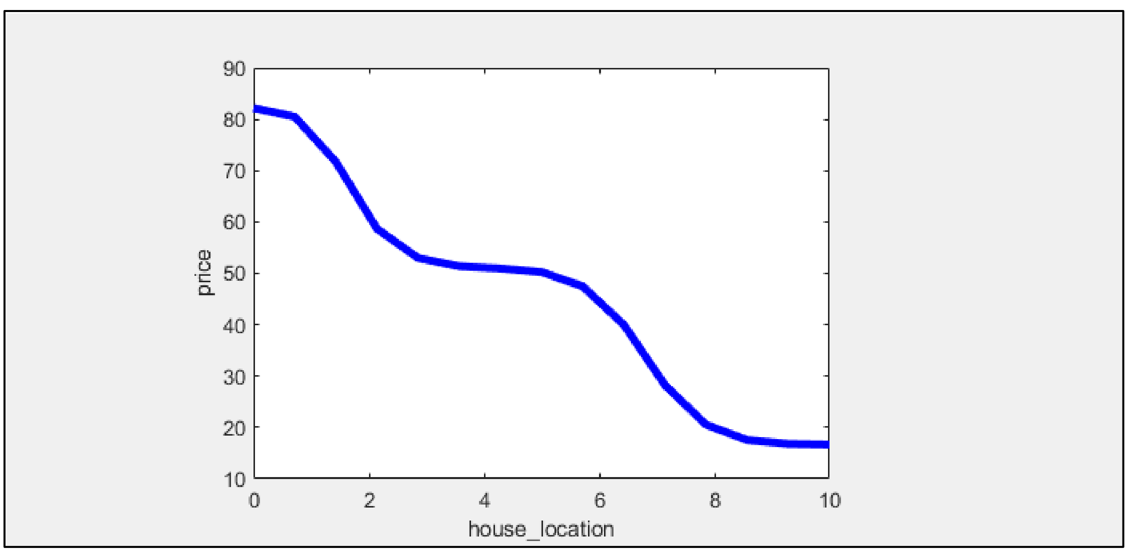
In this example, the position appears to be somewhat close, but its area is very small, so its price is not appropriate according to the last rule.



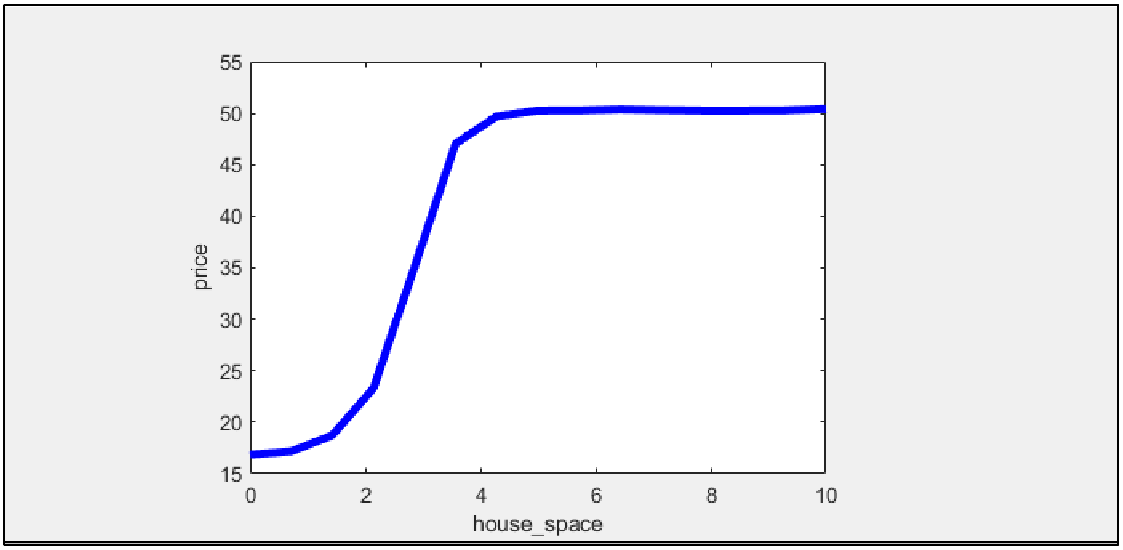
The surface viewer shows the relationship between the system variables "input and output variables", The drawing shows that the small the area of the house and the far from the city center the lower its price, and vice versa.



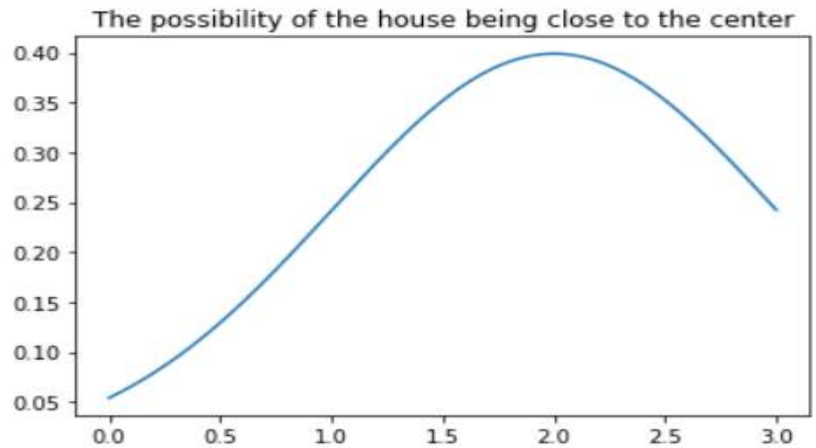
This graph represents the relationship between house location and its price.

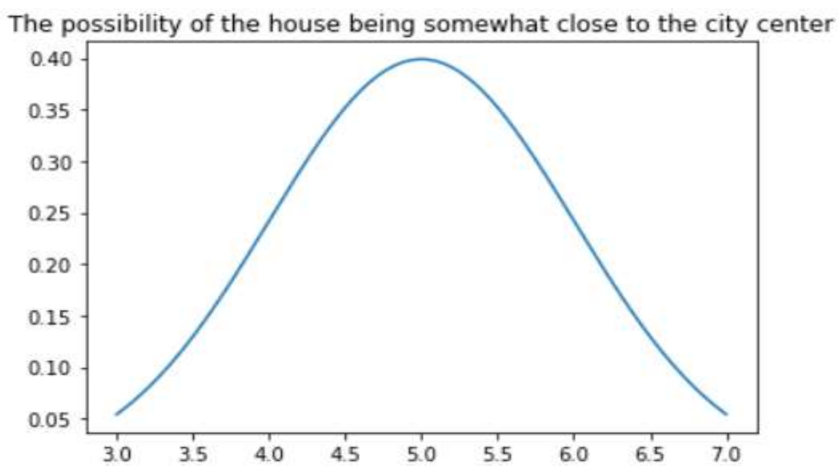


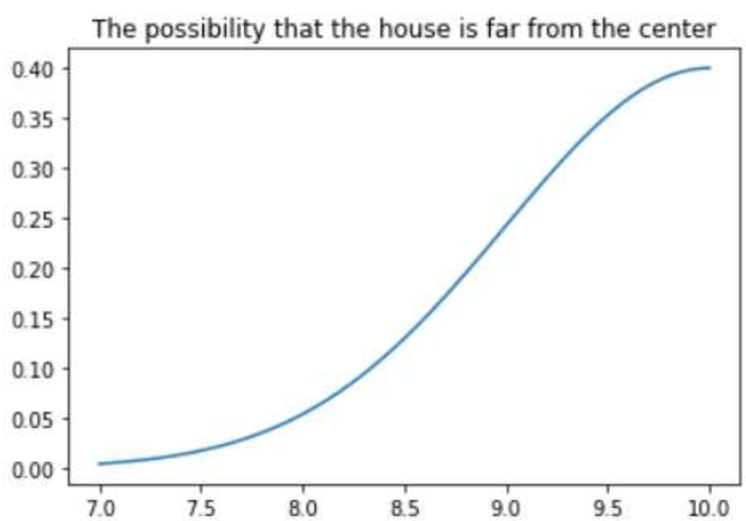
This chart displays the relationship between house space and its price.



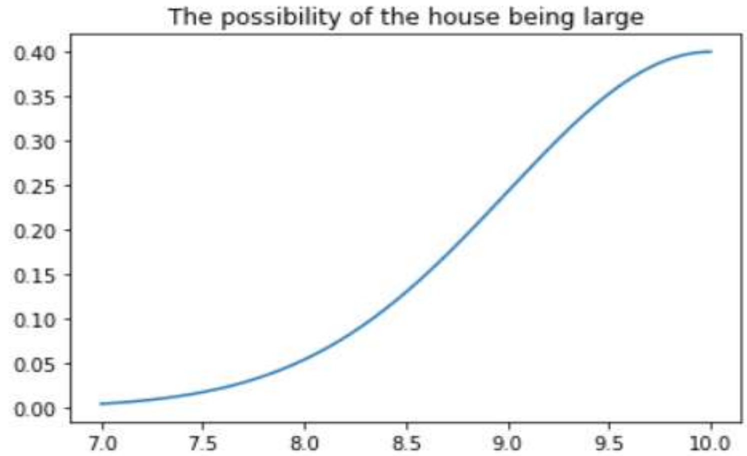
System variables degree curves in python: <u>(house location)</u>

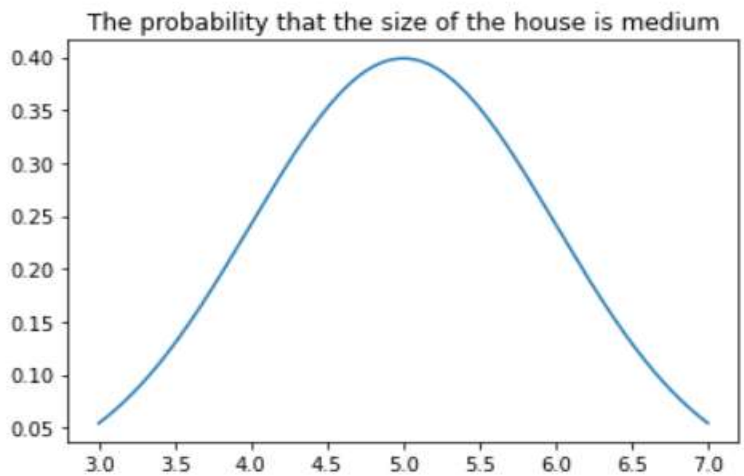


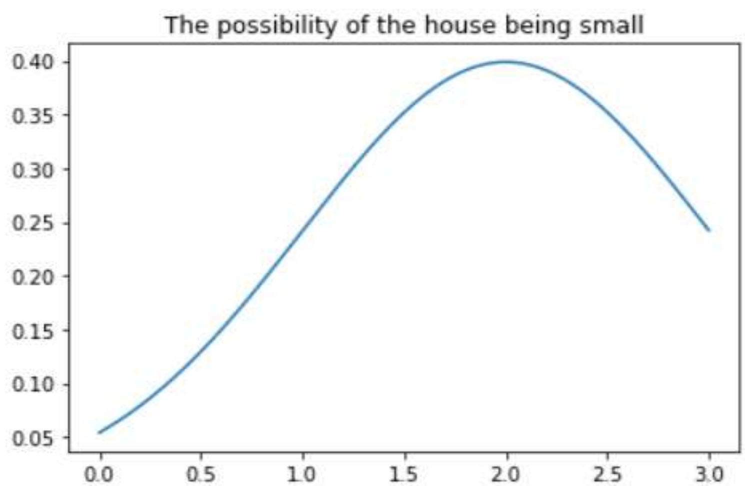




<u>(house space)</u>

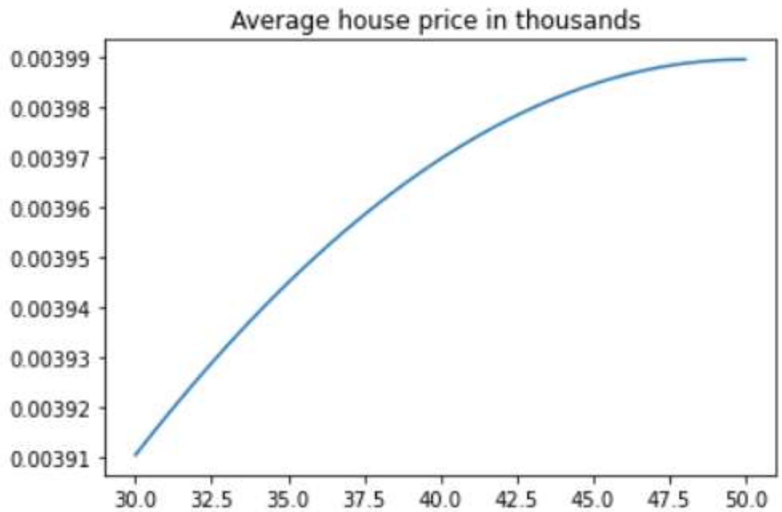






The approximate price of the house if it is cheap, the graph shown is the output of the system.

```
The house is not too expensive
<function matplotlib.pyplot.show>
```



The system can predict the value of any home based on its location and size, and displays a graph showing the approximate price. Hence the term "ambiguity, vagueness", which is the absence of a definite, clear, and certain value for a particular thing.

Code of question 1:

```
:- consult().

% Sleep

patient_sleep_to_actual(Sleep, SleepActual):-

    Sleep > 8,

    SleepActual = [sleep_more_than_8_hours].

patient_sleep_to_actual(Sleep, SleepActual):-

    Sleep =< 8,

    SleepActual = [].

% Exhausted

patient_exhausted_to_actual(Exhausted, ExhaustedActual):-

    Exhausted = yes,

    ExhaustedActual = [exhausted].

patient_exhausted_to_actual(Exhausted, ExhaustedActual):-

    Exhausted = no,

    ExhaustedActual = [].

% Nauseous

patient_nauseous_to_actual(Nauseous, NauseousActual):-

    Nauseous = yes,

    NauseousActual = [feels_nauseous].

patient_nauseous_to_actual(Nauseous, NauseousActual):-

    Nauseous = no,

    NauseousActual = [].

% Surgeries

patient_surgeries_to_actual(Surgeries, SurgeriesActual):-
```

```
    Surgeries = yes,
    SurgeriesActual = [previous_surgeries].
patient_surgeries_to_actual(Surgeries, SurgeriesActual):-
    Surgeries = no,
    SurgeriesActual = [].
% Interactive Console
main:-
    repeat,
    write('\nYou Are Welcome\n'),
    write('What is the number of hours does the patient sleep every day?\n'),
    read(Sleep), nl,
    patient_sleep_to_actual(Sleep, SleepActual),
    write('Does the patient feel exhausted?\n'),
    read(Exhausted), nl,
    patient_exhausted_to_actual(Exhausted, ExhaustedActual),
    write('Does the patient feel nauseous?\n'),
    read(Nauseous), nl,
    patient_nauseous_to_actual(Nauseous, NauseousActual),
    write('Has the patient had any previous surgeries?\n'),
    read(Surgeries), nl,
    patient_surgeries_to_actual(Surgeries, SurgeriesActual),
    write('Sleep: '), write(SleepActual), nl,
    write('Exhausted: '), write(ExhaustedActual), nl,
    write('Nauseous: '), write(NauseousActual), nl,
    write('Previous surgeries: '), write(SurgeriesActual), nl,
```

```
    sort([SleepActual, ExhaustedActual, NauseousActual, SurgeriesActual],
PatientKBWithEmptyClauses),
    delete(PatientKBWithEmptyClauses, [], PatientKB),
    append(PriorKB, PatientKB, KB),
    Questions = [Receive_free_medical_review],
nl,nl,
    write('Will the patient get a free medical review? '), nl, nl,write('Yes'),nl,
    write('\nLook at another patient?\n'),
    read(AnotherPatient),
    AnotherPatient = stop,
    !.
```

Code of question 2 (MATLAB):

```
[System]

Name='house_prices'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=4
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]

Name='house_location'
Range=[0 10]
NumMFs=3
MF1='close':'gaussmf',[1.044 1.842]
MF2='remote':'gaussmf',[1.054 8.36]
MF3='nearby':'gaussmf',[1.177 5.12]

[Input2]

Name='house_space'
Range=[0 10]
NumMFs=3
MF1='wide':'gaussmf',[1.147 8.331]
MF2='medium':'gaussmf',[1.272 4.75]
MF3='small':'gaussmf',[0.8381 1.476]

[Output1]

Name='price'
Range=[0 100]
NumMFs=3
```

MF1='low':'trimf',[0 16.67 33.33]
MF2='suitable':'trimf',[33.33 50 66.67]
MF3='high':'trimf',[66.67 83.33 100]

[Rules]

2 3, 1 (1) : 2
3 2, 2 (1) : 1
1 1, 3 (1) : 1
3 1, 2 (1) : 1

Code of question 2 (Python):

```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
from scipy.stats import norm

house_location = randint(1,10)
house_space = randint(1,10)

print(house_location)

print(house_space)

if house_location in range (7,11) or house_space in range (1,4):
  house_price = np.arange(30, 50, 0.001)
  plt.plot(house_price, norm.pdf(house_price,50,100))
  print("The house is not too expensive")
elif house_location in range (4,7) and house_space in range (4,7):
  house_price = np.arange(50, 100, 0.001)
  plt.plot(house_price, norm.pdf(house_price,100,10))
  print("The house price is suitable")
elif house_location in range (1,4) and house_space in range (7,11):
  house_price = np.arange(100, 200, 0.001)
  plt.plot(house_price, norm.pdf(house_price,150,10))
  print("The house price is expensive")
elif house_location in range (4,7) and house_space in range (7,11):
  house_price = np.arange(50, 100, 0.001)
  plt.plot(house_price, norm.pdf(house_price,100,10))
  print("The house price is suitable")
else:
  print("The house price depends on other factors")

plt.title('Average house price in thousands ' )
plt.show

house_close = np.arange(0, 3, 0.001)
plt.plot(house_close, norm.pdf(house_close,2,1))
plt.title('The possibility of the house being close to the center')
```

```
plt.show()

house_nearby = np.arange(3, 7, 0.001)
plt.plot(house_nearby, norm.pdf(house_nearby,5,1))
plt.title('The possibility of the house being somewhat close to the city center')
plt.show()

house_remote = np.arange(7, 10, 0.001)
plt.plot(house_remote, norm.pdf(house_remote,10,1))
plt.title('The possibility that the house is far from the center')
plt.show()

house_huge = np.arange(7, 10, 0.001)
plt.plot(house_huge, norm.pdf(house_huge,10,1))
plt.title('The possibility of the house being large')
plt.show()

house_medium = np.arange(3, 7, 0.001)
plt.plot(house_medium, norm.pdf(house_medium,5,1))
plt.title('The probability that the size of the house is medium')
plt.show()

house_small = np.arange(0, 3, 0.001)
plt.plot(house_small, norm.pdf(house_small,2,1))
plt.title('The possibility of the house being small')
plt.show()
```