Q1. Implement Bayes Theorum uning Python

Q2. Implement Conditional and joing Probability using Python.

Q3. Desing a Fuzzy based Application using Python.

Q4. Write an application to implement clustering algorithm

Q5. Write an application to implement support vector machine algorithm.

Q6. Design an AI application to implement intelligent agents.

Q7. Design an application to simulate semantic web.

Q1. Implement Bayes Theorum uning Python

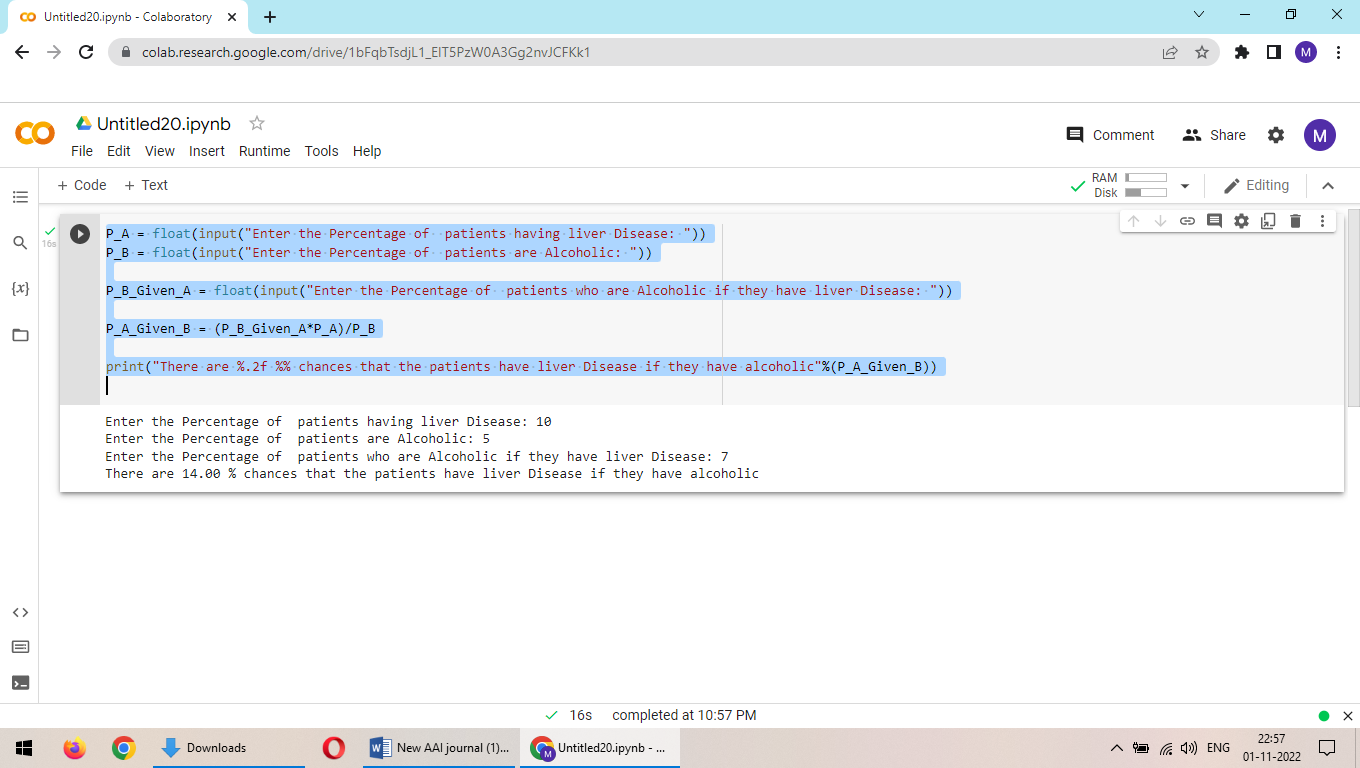
P\_A = float(input("Enter the Percentage of  patients having liver Disease: "))

P\_B = float(input("Enter the Percentage of  patients are Alcoholic: "))

P\_B\_Given\_A = float(input("Enter the Percentage of  patients who are Alcoholic if they have liver Disease: "))

P\_A\_Given\_B = (P\_B\_Given\_A\*P\_A)/P\_B

print("There are %.2f %% chances that the patients have liver Disease if they have alcoholic"%(P\_A\_Given\_B))



Q2. Implement Conditional and joing Probability using Python.

import pandas as pd

import numpy as np

Conditional:

#create pandas DataFrame with raw data

df = pd.DataFrame({'gender': np.repeat(np.array(['Male', 'Female']), 150),

                   'sport': np.repeat(np.array(['Baseball', 'Basketball', 'Football',

                                                'Soccer', 'Baseball', 'Basketball',

                                                'Football', 'Soccer']),

                                    (34, 40, 58, 18, 34, 52, 20, 44))})

#produce contingency table to summarize raw data

survey\_data = pd.crosstab(index=df['gender'], columns=df['sport'], margins=True)

#view contingency table

print(survey\_data)

#extract value in second row and first column

print(survey\_data.iloc[1, 0])

#calculate probability of being male, given that individual prefers baseball

print(survey\_data.iloc[1, 0] / survey\_data.iloc[2, 0])

#calculate probability of preferring basketball, given that individual is female

print(survey\_data.iloc[0, 1] / survey\_data.iloc[0, 4])

Joint Probability:

Card\_Colour= input('Enter the Colour of the card: ')

Card\_Number= input('Enter the Number of the card: ')

# P(A) is the Probability of drawing a card with entered color

P\_A = 26/52

# P(B) is the Probability of drawing a card with entered Number

P\_B = 4/52

print("Probability of drawing a ",Card\_Colour,' card is ',round(P\_A,2))

print("Probability of drawing a card with number ",Card\_Colour,' is ',round(P\_B,2))

P\_A\_AND\_B = round(P\_A\*P\_B,2)

print("Probability of drawing ", Card\_Colour, ' card with the number ', Card\_Number, ' from a normal deck of 52 playing card is ', P\_A\_AND\_B)

Q3. Desing a Fuzzy based Application using Python.

#pip3 install scikit-fuzzy

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')

service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')

tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

quality.automf(3)

service.automf(3)

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])

tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])

tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

quality['average'].view()

service.view()

tip.view()

rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])

rule2 = ctrl.Rule(service['average'], tip['medium'])

rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

rule1.view()

tipping\_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])

tipping = ctrl.ControlSystemSimulation(tipping\_ctrl)

# Pass inputs to the ControlSystem using Antecedent labels with Pythonic API

tipping.input['quality'] = 6.5

tipping.input['service'] = 9.8

# Crunch the numbers

tipping.compute()

print (tipping.output['tip'])

tip.view(sim=tipping)

Q4. Write an application to implement clustering algorithm

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

######

from sklearn.cluster import KMeans

data = list(zip(x, y))

inertias = []

for i in range(1,11):

    kmeans = KMeans(n\_clusters=i)

    kmeans.fit(data)

    inertias.append(kmeans.inertia\_)

plt.plot(range(1,11), inertias, marker='o')

plt.title('Elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()

#####

data = list(zip(x, y))

kmeans = KMeans(n\_clusters=2)

kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels\_)

plt.show()

Q5. Write an application to implement support vector machine algorithm.

import random

def display(room):

    print(room)

room = [

    [1, 1, 1, 1],

    [1, 1, 1, 1],

    [1, 1, 1, 1],

    [1, 1, 1, 1],

]

print("All the rooom are dirty")

display(room)

x =0

y= 0

while x < 4:

    while y < 4:

        room[x][y] = random.choice([0,1])

        y+=1

    x+=1

    y=0

print("Before cleaning the room I detect all of these random dirts")

display(room)

#starting location

x =0

y= 0

z=0 #number of rooms cleaned

#agent code

while x < 4:

    while y < 4:

        if room[x][y] == 1:

            print("Vaccum in this location now,",x, y)

            room[x][y] = 0

            print("cleaned", x, y)

            z+=1

        y+=1

    x+=1

    y=0

pro= (100-((z/16)\*100))

print("Room is clean now")

display(room)

print('performance=',pro,'%')

Q6. Design an AI application to implement intelligent agents.

Q7. Design an application to simulate semantic web.

Save this file as **myfoaf.rdf**

<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:foaf="http://xmlns.com/foaf/0.1/"

xmlns:admin="http://webns.net/mvcb/">

<foaf:Person rdf:nodeID="me">

<foaf:name>Mithilesh</foaf:name>

<foaf:knows>

<foaf:Person>

<foaf:name>Bhumika</foaf:name>

</foaf:Person>

</foaf:knows>

<foaf:knows>

<foaf:Person>

<foaf:name>Tanush</foaf:name>

</foaf:Person>

</foaf:knows>

</foaf:Person>

</rdf:RDF>

**#semanticweb.py :**

Fetching results about a specific subject using RDFLib

import rdflib

mygraph = rdflib.Graph()

mygraph.parse("myfoaf.rdf")

qres = mygraph.query(

"""SELECT DISTINCT ?fname ?lname

WHERE {

?a foaf:knows ?b .

?a foaf:name ?fname .

?b foaf:name ?lname .

}""")

for myrow in qres:

print("%s knows %s" % myrow)