**Artificial Intelligence Lab**

**LAB 9 – Implementation of uncertain methods for an application**

**Aditya Kumar**

**RA1911030010100**

**O2 Section**

**Date – 1/4/22**

**Problem Statement :**

Implementation of uncertain methods for Monty Hall Problem.

The Monty Hall Problem gets its name from the TV game show, Let's Make A Deal, hosted by Monty Hall

1. The scenario is such: you are given the opportunity to select one closed door of three, behind one of which there is a prize. The other two doors hide “goats” (or some other such “non-prize”), or nothing at all. Once you have made your selection, Monty Hall will open one of the remaining doors, revealing that it does not contain the prize

2. He then asks you if you would like to switch your selection to the other unopened door, or stay with your original choice.

3. We find the probability of getting correct door by deciding to either switch or sticking with original choice.

**Algorithm :**

1. Import numpy and pomegranate modules
2. Give discrete distribution values to guest and prize
3. Create a conditional probability table
4. Create Bayesian network
5. Fetch data from the created Bayesian network

**Code:**

import numpy as np

from pomegranate import \*

guest =DiscreteDistribution({ 'A': 1/3, 'B': 1/3, 'C': 1/3 })

prize =DiscreteDistribution({ 'A': 1/3, 'B': 1/3, 'C': 1/3 })

monty = ConditionalProbabilityTable(

[['A', 'A', 'A', 0.0],

['A', 'A', 'B', 0.5],

['A', 'A', 'C', 0.5],

['A', 'B', 'A', 0.0],

['A', 'B', 'B', 0.0],

['A', 'B', 'C', 1.0],

['A', 'C', 'A', 0.0],

['A', 'C', 'B', 1.0],

['A', 'C', 'C', 0.0],

['B', 'A', 'A', 0.0],

['B', 'A', 'B', 0.0],

['B', 'A', 'C', 1.0],

['B', 'B', 'A', 0.5],

['B', 'B', 'B', 0.0],

['B', 'B', 'C', 0.5],

['B', 'C', 'A', 1.0],

['B', 'C', 'B', 0.0],

['B', 'C', 'C', 0.0],

['C', 'A', 'A', 0.0],

['C', 'A', 'B', 1.0],

['C', 'A', 'C', 0.0],

['C', 'B', 'A', 1.0],

['C', 'B', 'B', 0.0],

['C', 'B', 'C', 0.0],

['C', 'C', 'A', 0.5],

['C', 'C', 'B', 0.5],

['C', 'C', 'C', 0.0]], [guest, prize])

s1 = Node(guest, name="guest")

s2 = Node(prize, name="prize")

s3 = Node(monty, name="monty")

model = BayesianNetwork("Monty Hall")

model.add\_states(s1, s2, s3)

model.add\_edge(s1, s3)

model.add\_edge(s2, s3)

model.bake()

# Let's say the guest choose door A. We can predict all the probabilites given this

beliefs = model.predict\_proba({ 'guest' : 'A' })

for i,b in enumerate(beliefs):

print(model.states[i].name)

print(b.parameters) if hasattr(b, 'parameters') else print(b)

print("")

# Monty always know where the prize is and he always reveal a door without the prize asking to the guest if he wants to change his initial choice.

beliefs = model.predict\_proba({ 'guest' : 'A', 'monty' : 'B' })

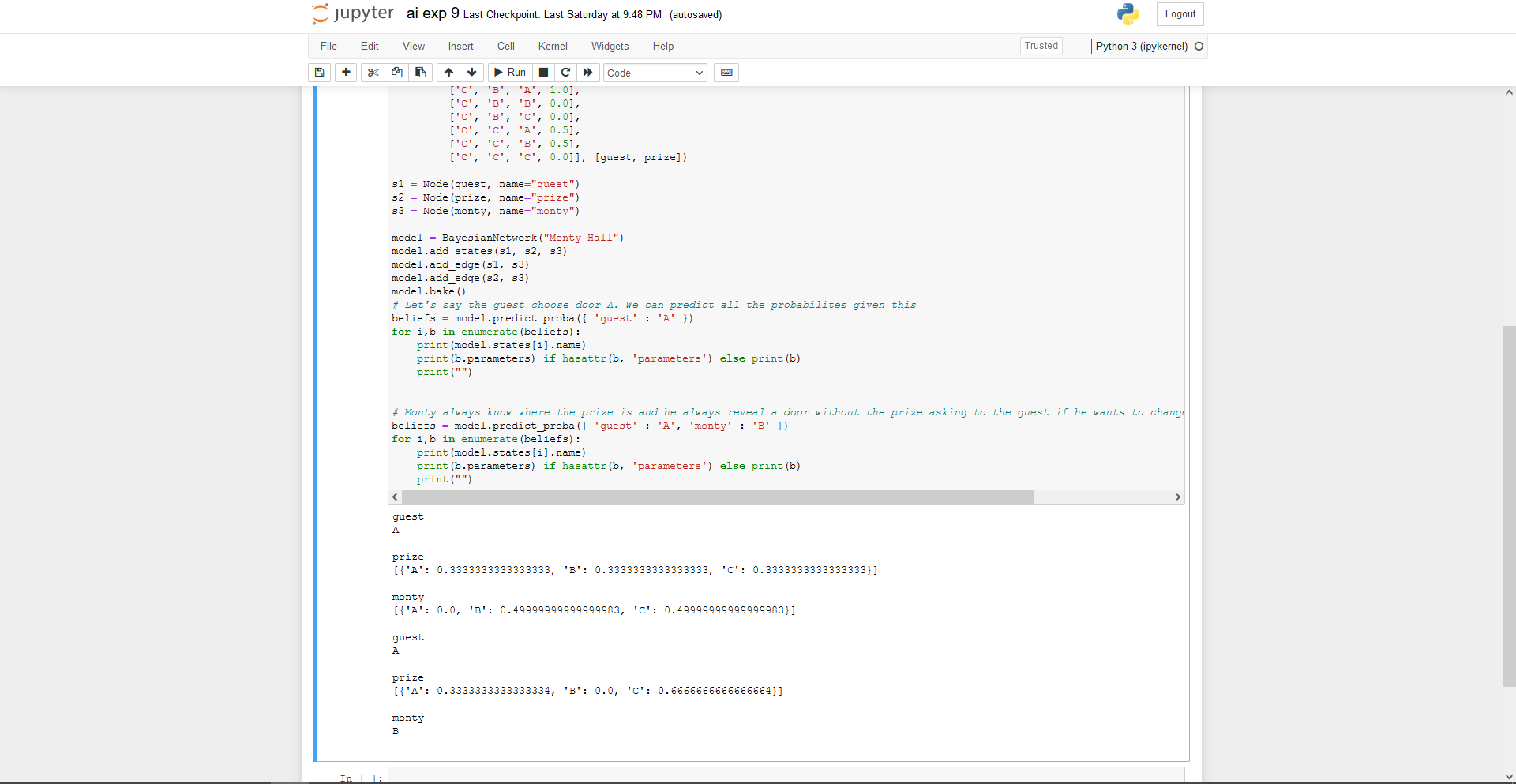
for i,b in enumerate(beliefs):

print(model.states[i].name)

print(b.parameters) if hasattr(b, 'parameters') else print(b)

print("")

OUTPUT :



RESULT :

Hence we implemented uncertain methods for an application and executed it.