Experiment- 9 **A N V SREEVISHNU**

Date: 28.04.2021 **RA1811003010333**

Implementation of uncertain methods for an application

Aim: To verify the implementation of uncertain methods for an application.

Algorithm:

1. All the necessary libraries are imported.
2. For guest and prize the event probability is initialized
3. The conditional probability table is then created.
4. Bayesian probability function is used.
5. The probabilities are then calculate accordingly.

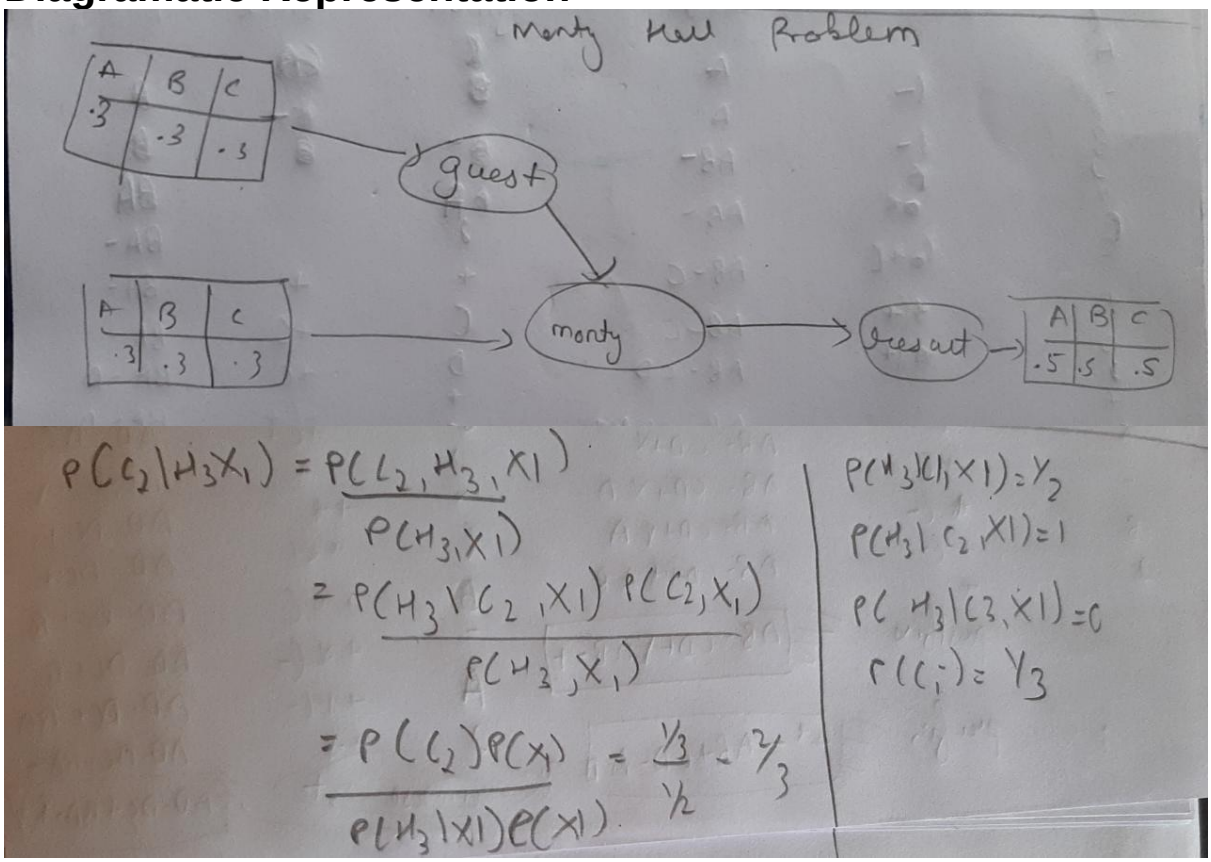
Code:

```
import matplotlib.pyplot as plt
import seaborn; seaborn.set_style('whitegrid')
import numpy
from pomegranate import *
numpy.random.seed(0)
numpy.set_printoptions(suppress=True)
# The guests initial door selection is completely random
guest = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})
# The door the prize is behind is also completely random
prize = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})
# Monty is dependent on both the guest and the prize.
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])
# State objects hold both the distribution, and a high level name.
s1 = State(guest, name="guest")
s2 = State(prize, name="prize")
s3 = State(monty, name="monty")
# Create the Bayesian network object with a useful name
model = BayesianNetwork("Monty Hall Problem")
# Add the three states to the network
model.add_states(s1, s2, s3)
# Add edges which represent conditional dependencies, where the second node is
conditionally dependent on the first node (Monty is dependent on both guest and prize)
model.add_edge(s1, s3)
model.add_edge(s2, s3)
model.bake()
model.probability([['A', 'B', 'C']])
model.probability([['A', 'B', 'C']])
print(model.predict_proba({}))
print(model.predict_proba([[None, None, None]]))
print(model.predict_proba([['A', None, None]]))
print(model.predict_proba([{'guest': 'A', 'monty': 'B'}]))
```

Diagramatic Representation



Output:

```
[{
    "class" : "Distribution",
    "dtype" : "str",
    "name" : "DiscreteDistribution",
    "parameters" : [
      {
        "A" : 0.33333333333333337,
        "B" : 0.33333333333333337,
        "C" : 0.33333333333333337
      }
    ],
    "frozen" : false
}
{
    "class" : "Distribution",
    "dtype" : "str",
    "name" : "DiscreteDistribution",
    "parameters" : [
      {
        "A" : 0.33333333333333337,
        "B" : 0.33333333333333337,
        "C" : 0.33333333333333337
      }
    ],
    "frozen" : false
}
{
```

```
    "class" : "Distribution",
    "dtype" : "str",
    "name" : "DiscreteDistribution",
    "parameters" : [
      {
        "C" : 0.3333333333333333,
        "A" : 0.3333333333333333,
        "B" : 0.3333333333333333
      }
    ],
    "frozen" : false
 }                          ]
[array([{
        "class" : "Distribution",
        "dtype" : "str",
        "name" : "DiscreteDistribution",
        "parameters" : [
          {
            "A" : 0.33333333333333337,
            "B" : 0.33333333333333337,
            "C" : 0.33333333333333337
          }
        ],
        "frozen" : false
      }                          ,
      {
        "class" : "Distribution",
```

```
      "dtype" : "str",
      "name" : "DiscreteDistribution",
      "parameters" : [
        {
          "A" : 0.33333333333333337,
          "B" : 0.33333333333333337,
          "C" : 0.33333333333333337
        }
      ],
      "frozen" : false
    }                              ,
    {
      "class" : "Distribution",
      "dtype" : "str",
      "name" : "DiscreteDistribution",
      "parameters" : [
        {
          "C" : 0.3333333333333333,
          "A" : 0.3333333333333333,
          "B" : 0.3333333333333333
        }
      ],
      "frozen" : false
    }                              ], dtype=object)]
[array(['A', {
          "class" : "Distribution",
          "dtype" : "str",
```

```
            "name" : "DiscreteDistribution",
            "parameters" : [
              {
                "A" : 0.3333333333333333,
                "B" : 0.3333333333333333,
                "C" : 0.3333333333333333
              }
            ],
            "frozen" : false
          }                              ,
        {
          "class" : "Distribution",
          "dtype" : "str",
          "name" : "DiscreteDistribution",
          "parameters" : [
            {
              "C" : 0.49999999999999983,
              "A" : 0.0,
              "B" : 0.49999999999999983
            }
          ],
          "frozen" : false
        }                              ], dtype=object)]
[array(['A', {
            "class" : "Distribution",
            "dtype" : "str",
            "name" : "DiscreteDistribution",
```

```
    "parameters" : [
      {
        "A" : 0.3333333333333334,
        "B" : 0.0,
        "C" : 0.6666666666666664
      }
    ],
    "frozen" : false
}                          , 'B'], dtype=object)]
```


Result: Verification of the implementation of uncertain methods for an application is done successfully.