VIVEK BURUGUDDA RA1911003010316

HARSHIL AGARWAL RA1911003010325

RITESH RAI RA1911003010333

PUNEET SHARMA RA1911003010331

Prakhar Vijay RA1911003010337

**18CSC305J - ARTIFICIAL INTELLIGENCE**

**Week 8: Implementation of Bayesian Belief Network**

Aim: Implementation of Bayesian Belief Network

**Code:**

#Import required packages

import math

from pomegranate import \*

# Initially the door selected by the guest is completely random

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

# The door containing the prize is also a random process

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

# The door Monty picks, depends on the choice of the guest and the prize door

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] )

d1 = State( guest, name="guest" )

d2 = State( prize, name="prize" )

d3 = State( monty, name="monty" )

#Building the Bayesian Network

network = BayesianNetwork( "Solving the Monty Hall Problem With Bayesian Networks" )

network.add\_states(d1, d2, d3)

network.add\_edge(d1, d3)

network.add\_edge(d2, d3)

network.bake()

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

print("n".join( "{}t{}".format( state.name, str(belief) ) for state, belief in zip( network.states, beliefs )))

guesttAnprizet{

"class" : "Distribution",

"dtype" : "str",

"name" : "DiscreteDistribution",

"parameters" : [

{

"A" : 0.3333333333333334,

"B" : 0.0,

"C" : 0.6666666666666664

}

],

"frozen" : false

}nmontytB

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

beliefs = map(str, beliefs)

print("n".join( "{}t{}".format( state.name, belief ) for state, belief in zip( network.states, beliefs ) ))

guesttAnprizet{

"class" : "Distribution",

"dtype" : "str",

"name" : "DiscreteDistribution",

"parameters" : [

{

"A" : 0.3333333333333333,

"B" : 0.3333333333333333,

"C" : 0.3333333333333333

}

],

"frozen" : false

}nmontyt{

"class" : "Distribution",

"dtype" : "str",

"name" : "DiscreteDistribution",

"parameters" : [

{

"A" : 0.0,

"B" : 0.49999999999999983,

"C" : 0.49999999999999983

}

],

"frozen" : false

}