Name: Durva Kadam Roll No: 23

St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2024-2025
Class: BE-ITA/B, Semester: VII
Subject: Data Science Lab

Experiment – 1

1. **Aim:** To implement a Bayes' Network for Alarm problem

2. **Objectives:** Students should be able to apply to apply reasoning for a problem in an uncertain domain.

3. **Prerequisite:** Python basics

4. **Requirements:** PC, Python 3.9, Windows 10/ MacOS/ Linux, IDLE IDE

5. **Pre-Experiment Exercise:**
   Theory:
   Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

   "A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."It is also called a Bayes network, belief network, decision network, or Bayesian model.

   Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.

   Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

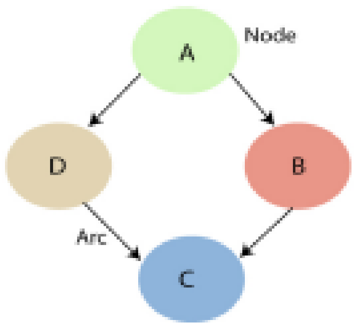
   Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

   - Directed Acyclic Graph
   - Table of conditional probabilities.

   The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.

   Bayesian Belief Network:

   A Bayesian network graph is made up of nodes and Arcs (directed links),



where:

- Each node corresponds to the random variables, and a variable can be continuous or discrete.

- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
- These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
- In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.
- If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.
- **Node C is independent of node A.**

The Bayesian network has mainly two components:

- Causal Component
- Actual numbers.

Each node in the Bayesian network has condition probability distribution P(Xi |Parent(Xi)), which determines the effect of the parent on that node.

6. Laboratory Exercise

A. Procedure

i. Use google colab for programming.
ii. Import prebuilt model for Bayes' Network.
iii.Display Network structure for the problem given as output.
iv.Display conditional probability table.
v. Add relevant comments in your programs and execute the code. Test it for various cases.

7. Post-Experiments Exercise:

A. Extended Theory:

a. Design a Bayes' network with conditional probability associated with each node for lung cancer problems.
b. Calculate number of independent variables required for a Bayes' Network?

B. Post Lab Program:

a. Write a Python program to implement Bayes' Network for Lung cancer problem.

C. Conclusion:

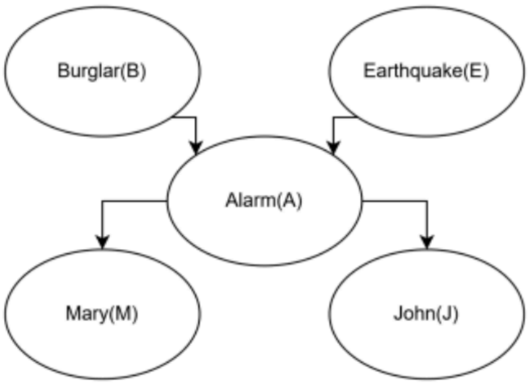
1. Write what was performed in the program (s) .
2. What is the significance of the program and what Objective is achieved?

D. References:

[1] https://colab.research.google.com/drive/1ZvtaLqPhzZtx5Vjt1UKiWdRlQttHI-M6?usp=sharing
[2]Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", Third Edition, Pearson Education

Name: Durva Kadam Roll No: 23

Consider a burglar alarm this alarm notifies in case of burglary and can occasionally response to minor earthquakes . There are 2 neighbors to a m and j , m and j promises to call a in office when they have alarm, j always calls A when he hears alarm but many times he is confused between telephone range and .. alarm , m many times misses alarm because he keeps on listening to loud music . Given the entire evidence who has or who has not called we have to estimate probability of burglary.



P(B)

Parameter	True	False
Burglar(B)	0.01	0.999

P(E)

Parameter	True	False
Earthquake(E)	0.02	0.998

P(A|B,E)

Burglar(B)	Earthquake(E)	Alarm(A)=False	Alarm(A)=True
F	F	0.999	0.001
F	T	0.71	0.29
T	F	0.06	0.94

T	T	0.05	0.95

P(J|A)

Alarm(A)	John Calls(J)=False	John Calls(J)=True
F	0.95	0.05
T	0.10	0.90

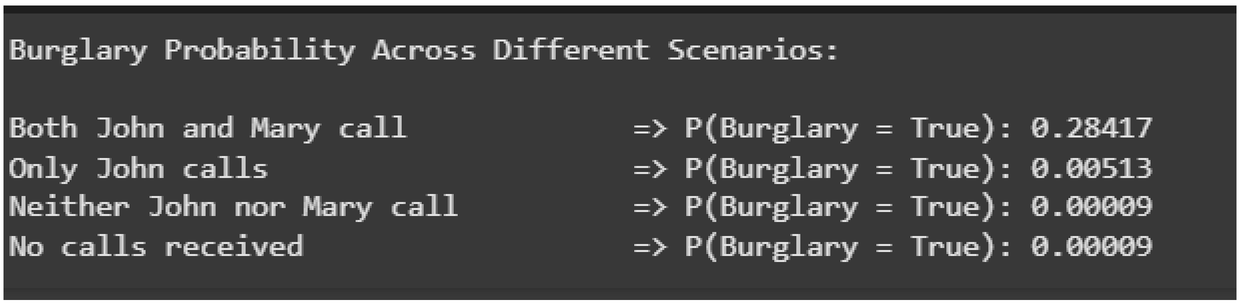
P(M|A)

Alarm(A)	Mary Calls(J)=False	Mary Calls(J)=True
F	0.99	0.01
T	0.30	0.70

CODE:

```
from pgmpy.models import DiscreteBayesianNetwork
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

# Step 1: Define the structure of the Bayesian Network
network = DiscreteBayesianNetwork([
    ('B', 'A'),  # Burglary influences Alarm
    ('E', 'A'),  # Earthquake influences Alarm
    ('A', 'J'),  # Alarm triggers John calling
    ('A', 'M')   # Alarm triggers Mary calling
])

# Step 2: Set up the CPDs for each variable
cpd_burglary = TabularCPD('B', 2, [[0.999], [0.001]])
cpd_earthquake = TabularCPD('E', 2, [[0.998], [0.002]])
cpd_alarm = TabularCPD(
    'A', 2,
    values=[
        [0.999, 0.71, 0.06, 0.05],  # P(Alarm=False | B,E)
        [0.001, 0.29, 0.94, 0.95]   # P(Alarm=True  | B,E)
    ],
    evidence=['B', 'E'],
    evidence_card=[2, 2]
)
cpd_john_calls = TabularCPD(
    'J', 2,
    values=[[0.95, 0.10], [0.05, 0.90]],  # P(JohnCalls | Alarm)
    evidence=['A'],
    evidence_card=[2]
)
cpd_mary_calls = TabularCPD(
    'M', 2,
    values=[[0.99, 0.30], [0.01, 0.70]],  # P(MaryCalls | Alarm)
    evidence=['A'],
    evidence_card=[2]
)

# Step 3: Add the CPDs to the network
network.add_cpds(cpd_burglary, cpd_earthquake, cpd_alarm, cpd_john_calls,
cpd_mary_calls)

# Step 4: Validate the network structure
assert network.check_model()

# Step 5: Prepare for inference
inference = VariableElimination(network)

# Step 6: Define various observational scenarios
evidence_scenarios = {
    "Both John and Mary call": {'J': 1, 'M': 1},
    "Only John calls": {'J': 1, 'M': 0},
    "Neither John nor Mary call": {'J': 0, 'M': 0},
    "No calls received": {'J': 0, 'M': 0}
}

# Step 7: Compute and display the probability of burglary in each scenario
print("Burglary Probability Across Different Scenarios:\n")
for scenario, observed in evidence_scenarios.items():
    query_result = inference.query(variables=['B'], evidence=observed)
    burglary_prob = query_result.values[1]  # Probability B=True
    print(f"{scenario:35s} => P(Burglary = True): {burglary_prob:.5f}")
```

Name: Durva Kadam Roll No: 23



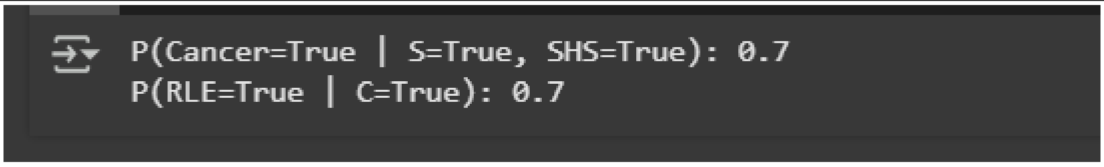
```
Burglary Probability Across Different Scenarios:

Both John and Mary call           => P(Burglary = True): 0.28417
Only John calls                   => P(Burglary = True): 0.00513
Neither John nor Mary call        => P(Burglary = True): 0.00009
No calls received                 => P(Burglary = True): 0.00009
```

7. Post Experiments Exercise:

B. Post Lab Program:

CODE:

```
from pgmpy.models import DiscreteBayesianNetwork
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

model = DiscreteBayesianNetwork([
    ('PS', 'S'),
    ('PS', 'SHS'),
    ('S', 'C'),
    ('SHS', 'C'),
    ('C', 'RLE')
])
cpd_ps = TabularCPD(variable='PS', variable_card=2, values=[[0.6], [0.4]])
cpd_s = TabularCPD(variable='S', variable_card=2, values=[[0.8, 0.3], [0.2, 0.7]],
evidence=['PS'], evidence_card=[2])
cpd_shs = TabularCPD(variable='SHS', variable_card=2, values=[[0.7, 0.2], [0.3, 0.8]],
evidence=['PS'], evidence_card=[2])
cpd_c = TabularCPD(variable='C', variable_card=2, values=[[0.99, 0.90, 0.85, 0.30],
[0.01, 0.10, 0.15, 0.70]], evidence=['S', 'SHS'], evidence_card=[2, 2])
cpd_rle = TabularCPD(variable='RLE', variable_card=2, values=[[0.9, 0.3], [0.1, 0.7]],
evidence=['C'], evidence_card=[2])

model.add_cpds(cpd_ps, cpd_s, cpd_shs, cpd_c, cpd_rle)
assert model.check_model()

infer = VariableElimination(model)
result = infer.query(variables=['C'], evidence={'S': 1, 'SHS': 1})
print("P(Cancer=True | S=True, SHS=True):", result.values[1])
result2 = infer.query(variables=['RLE'], evidence={'C': 1})
print("P(RLE=True | C=True):", result2.values[1])
```



```
P(Cancer=True | S=True, SHS=True): 0.7
P(RLE=True | C=True): 0.7
```