Information Technology

Shrusti Chintawar

Class: D20B

Roll No. 09

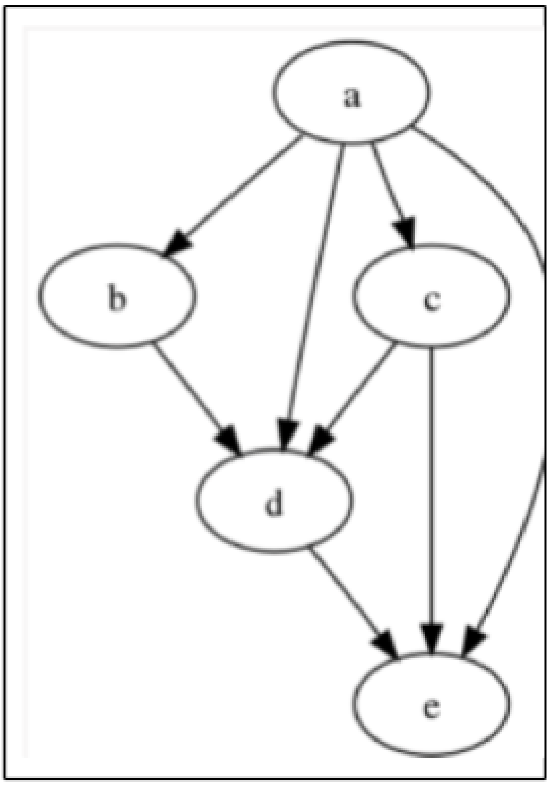
Expt.1 : To implement inferencing with Bayesian Network.

Explain the concept,draw the diagram, mention a few applications of Bayesian networks. Implement with different datasets and show the output. Test the same with different inputs.

Concept:

A Bayesian network (also spelled Bayes network, Bayes net, belief network, or judgement network) is a probabilistic graphical model that depicts a set of variables and their conditional dependencies using a directed acyclic graph (DAG). Bayesian networks are perfect for taking an observed event and forecasting the likelihood that any of the numerous known causes played a role. A Bayesian network, for example, could reflect the probability correlations between diseases and symptoms. Given a set of symptoms, the network may be used to calculate the likelihood of the presence of certain diseases.

In graph theory and computer science, a directed acyclic graph (DAG) is a directed graph with no directed cycles. In other words, it's made up of vertices and edges (also called arcs), with each edge pointing from one vertex to the next in such a way that following those directions would never lead to a closed-loop as depicted in the picture below.



Applications of Bayesian Networks:

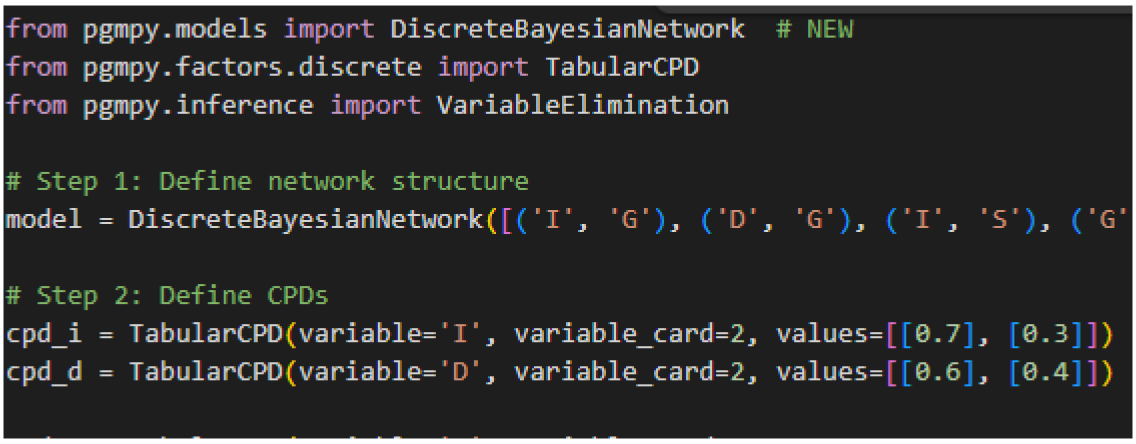
● Medical Diagnosis: To infer the probability of diseases based on symptoms.

● Fault Diagnosis: In complex systems like aircraft or nuclear plants. ● Decision Support Systems: In finance and marketing for risk assessment and decision making.

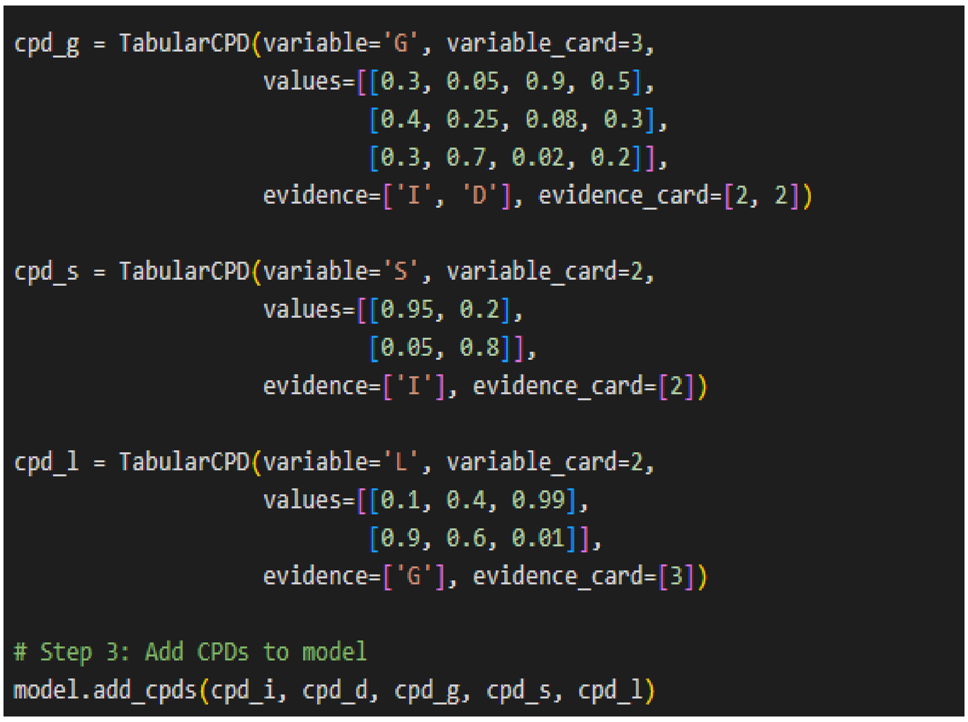
● Gene Expression Analysis: In bioinformatics to understand the relationships between genes.

Python Implementation:

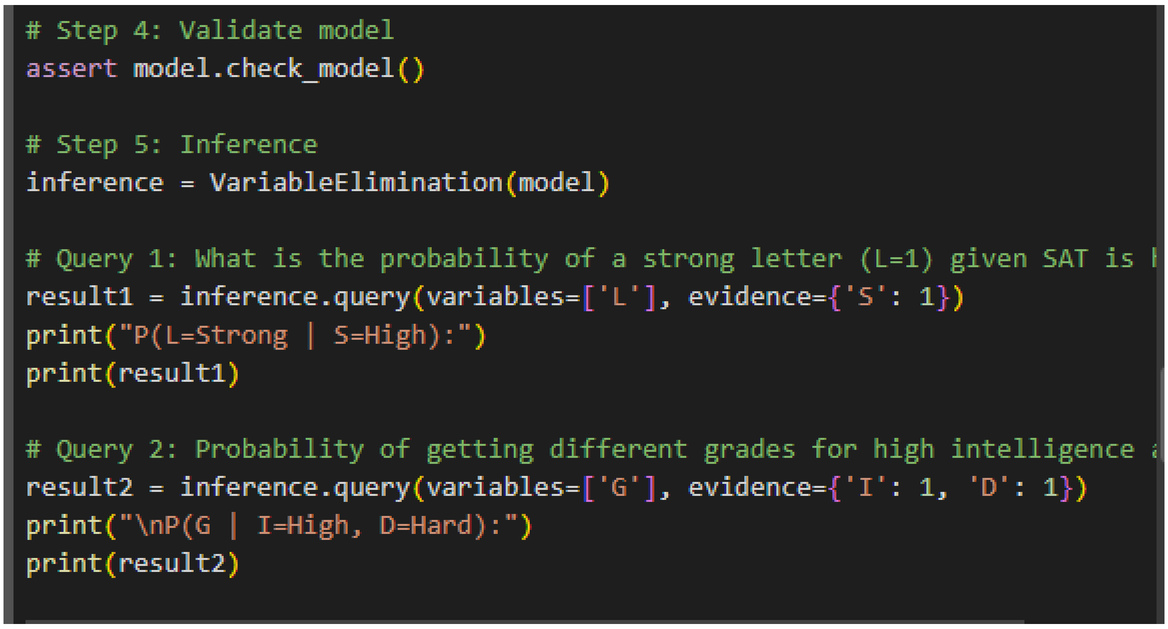
Dataset 1: Rain and Grass



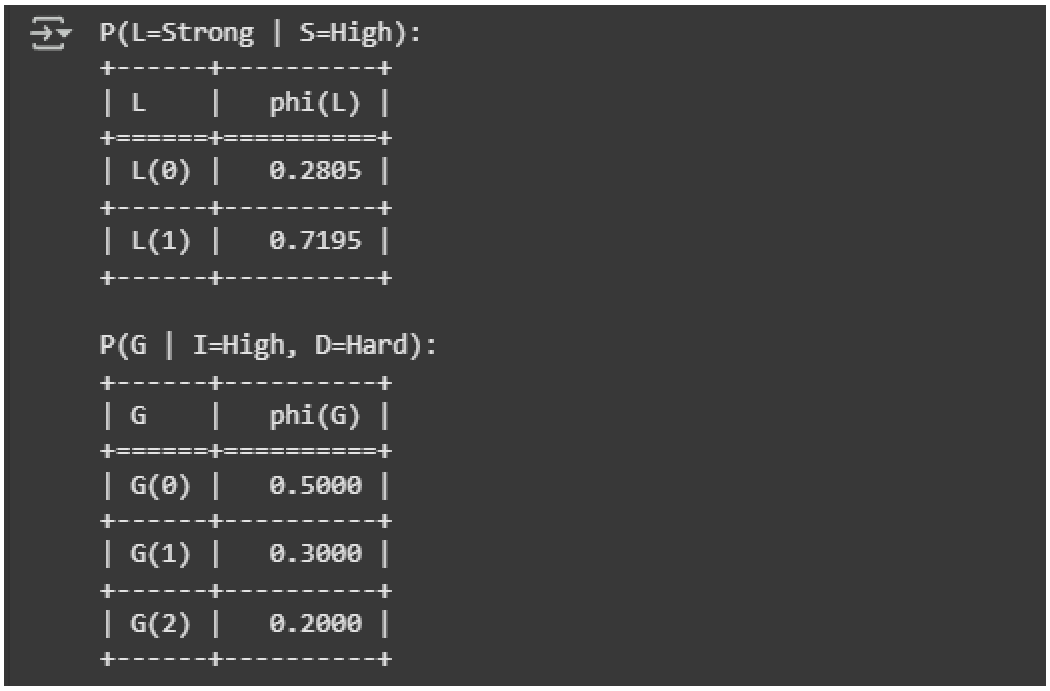
```python
from pgmpy.models import DiscreteBayesianNetwork  # NEW
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

# Step 1: Define network structure
model = DiscreteBayesianNetwork([('I', 'G'), ('D', 'G'), ('I', 'S'), ('G'

# Step 2: Define CPDs
cpd_i = TabularCPD(variable='I', variable_card=2, values=[[0.7], [0.3]])
cpd_d = TabularCPD(variable='D', variable_card=2, values=[[0.6], [0.4]])
```



```python
cpd_g = TabularCPD(variable='G', variable_card=3,
                   values=[[0.3, 0.05, 0.9, 0.5],
                           [0.4, 0.25, 0.08, 0.3],
                           [0.3, 0.7, 0.02, 0.2]],
                   evidence=['I', 'D'], evidence_card=[2, 2])

cpd_s = TabularCPD(variable='S', variable_card=2,
                   values=[[0.95, 0.2],
                           [0.05, 0.8]],
                   evidence=['I'], evidence_card=[2])

cpd_l = TabularCPD(variable='L', variable_card=2,
                   values=[[0.1, 0.4, 0.99],
                           [0.9, 0.6, 0.01]],
                   evidence=['G'], evidence_card=[3])

# Step 3: Add CPDs to model
model.add_cpds(cpd_i, cpd_d, cpd_g, cpd_s, cpd_l)
```

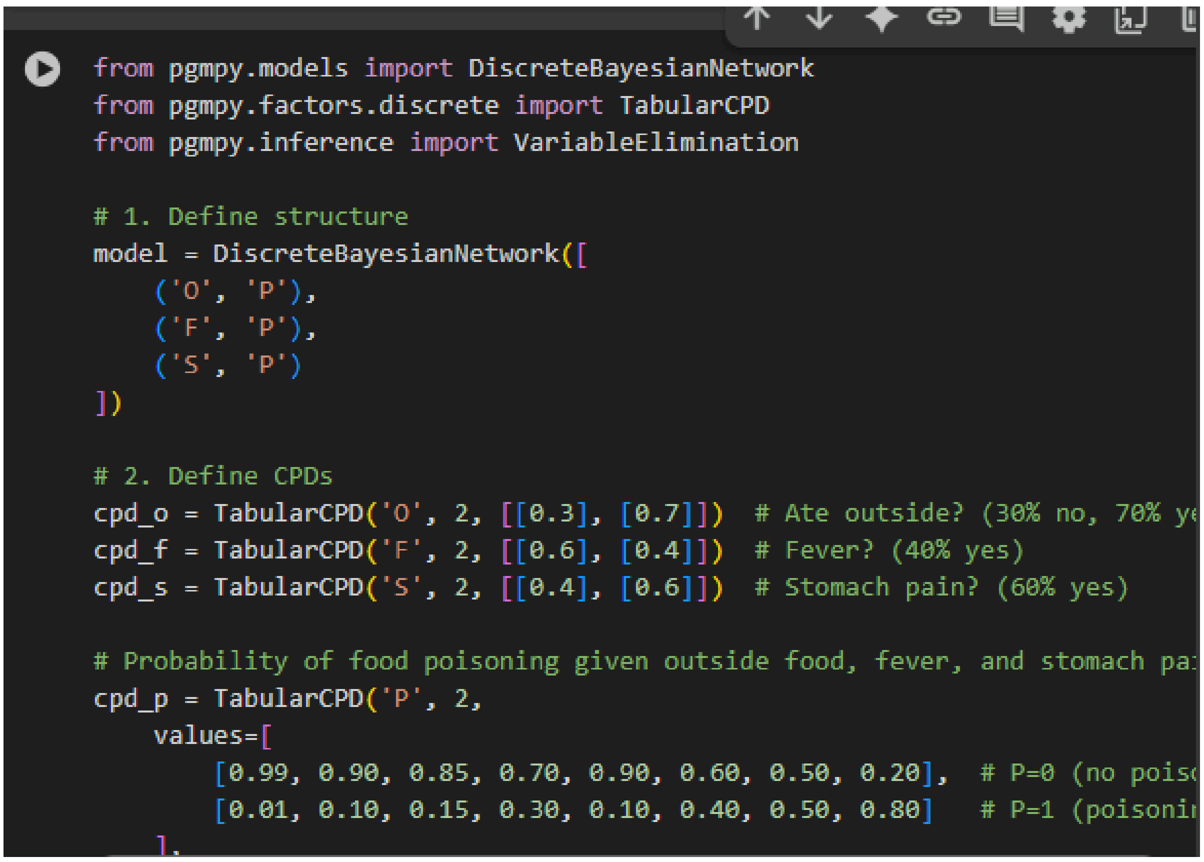


```python
# Step 4: Validate model
assert model.check_model()

# Step 5: Inference
inference = VariableElimination(model)

# Query 1: What is the probability of a strong letter (L=1) given SAT is h
result1 = inference.query(variables=['L'], evidence={'S': 1})
print("P(L=Strong | S=High):")
print(result1)

# Query 2: Probability of getting different grades for high intelligence a
result2 = inference.query(variables=['G'], evidence={'I': 1, 'D': 1})
print("\nP(G | I=High, D=Hard):")
print(result2)
```

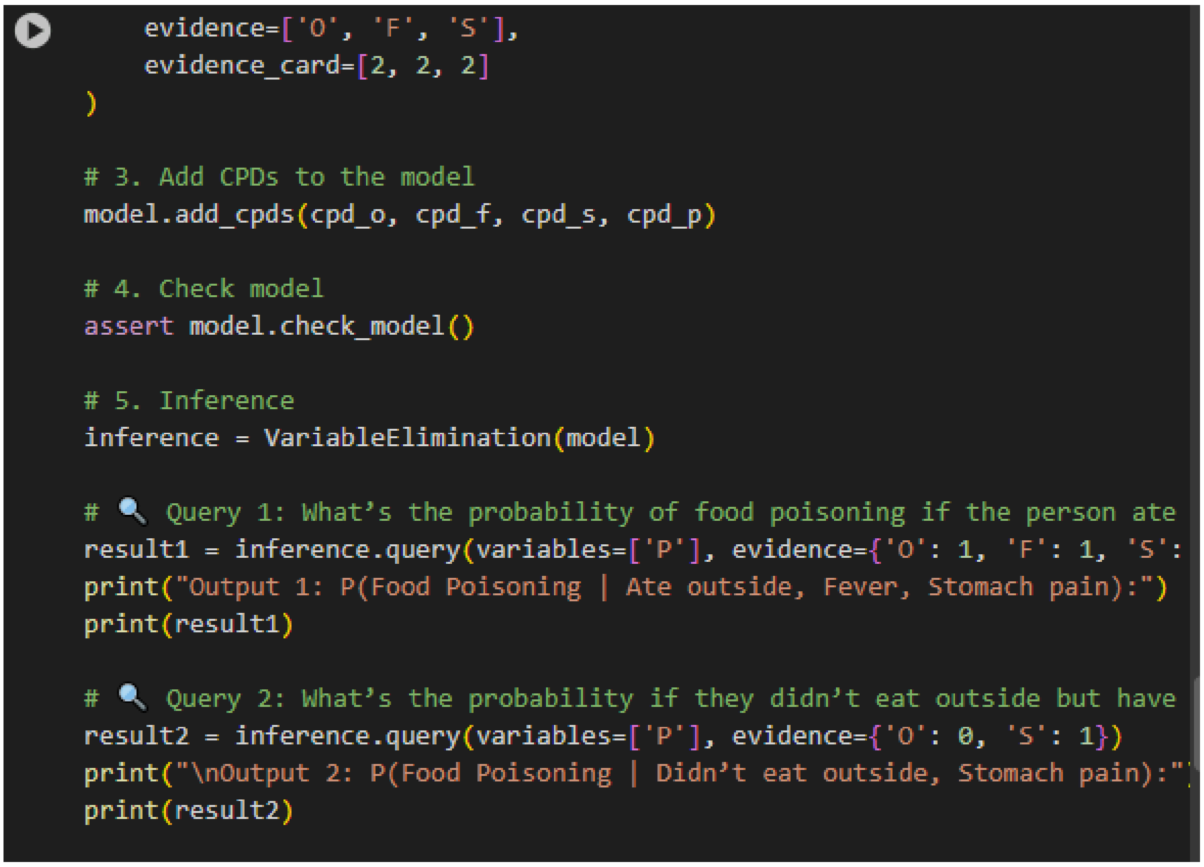


```
P(L=Strong | S=High):
+------+----------+
| L    |   phi(L) |
+======+==========+
| L(0) |   0.2805 |
+------+----------+
| L(1) |   0.7195 |
+------+----------+

P(G | I=High, D=Hard):
+------+----------+
| G    |   phi(G) |
+======+==========+
| G(0) |   0.5000 |
+------+----------+
| G(1) |   0.3000 |
+------+----------+
| G(2) |   0.2000 |
+------+----------+
```

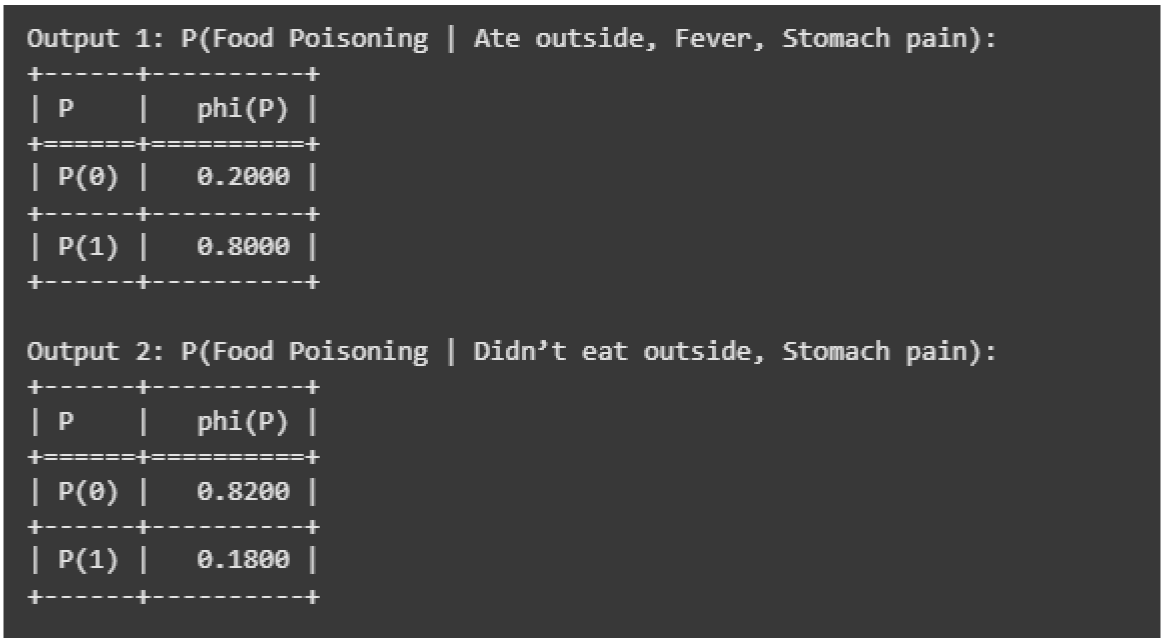
Dataset 2: Medical Diagnosis Example (Disease → Symptom)



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

# 1. Define structure
model = DiscreteBayesianNetwork([
    ('O', 'P'),
    ('F', 'P'),
    ('S', 'P')
])

# 2. Define CPDs
cpd_o = TabularCPD('O', 2, [[0.3], [0.7]])  # Ate outside? (30% no, 70% ye
cpd_f = TabularCPD('F', 2, [[0.6], [0.4]])  # Fever? (40% yes)
cpd_s = TabularCPD('S', 2, [[0.4], [0.6]])  # Stomach pain? (60% yes)

# Probability of food poisoning given outside food, fever, and stomach pa
cpd_p = TabularCPD('P', 2,
    values=[
        [0.99, 0.90, 0.85, 0.70, 0.90, 0.60, 0.50, 0.20],  # P=0 (no poiso
        [0.01, 0.10, 0.15, 0.30, 0.10, 0.40, 0.50, 0.80]   # P=1 (poisonin
    ],
```



```
    evidence=['O', 'F', 'S'],
    evidence_card=[2, 2, 2]
)

# 3. Add CPDs to the model
model.add_cpds(cpd_o, cpd_f, cpd_s, cpd_p)

# 4. Check model
assert model.check_model()

# 5. Inference
inference = VariableElimination(model)

# 🔍 Query 1: What's the probability of food poisoning if the person ate
result1 = inference.query(variables=['P'], evidence={'O': 1, 'F': 1, 'S':
print("Output 1: P(Food Poisoning | Ate outside, Fever, Stomach pain):")
print(result1)

# 🔍 Query 2: What's the probability if they didn't eat outside but have
result2 = inference.query(variables=['P'], evidence={'O': 0, 'S': 1})
print("\nOutput 2: P(Food Poisoning | Didn't eat outside, Stomach pain):"
print(result2)
```



```
Output 1: P(Food Poisoning | Ate outside, Fever, Stomach pain):
+------+----------+
| P    |   phi(P) |
+======+==========+
| P(0) |   0.2000 |
+------+----------+
| P(1) |   0.8000 |
+------+----------+

Output 2: P(Food Poisoning | Didn’t eat outside, Stomach pain):
+------+----------+
| P    |   phi(P) |
+======+==========+
| P(0) |   0.8200 |
+------+----------+
| P(1) |   0.1800 |
+------+----------+
```

Conclusion: I have learned about Bayesian Belief Network and its Applications. I have implemented its most popular example burglary system in Python.