EXPERIMENT 8

Academic Year	**2024-25**	**Estimated Time**	**Experiment No. 8 – 02 Hours**
Course & Semester	**T.E. (CE) – Sem. VI**	**Subject Name**	**CSL604: Artificial Intelligence**
Chapter No.	**04**	**Chapter Title**	**Reasoning Under Uncertainty**
Experiment Type	Modelling	**Software**	**Python/PROLOG**

AIM: To Create a Bayesian Network for the given Problem Statement and draw inferences from it. (You can use any Belief and Decision Networks Tool for modelling Bayesian Networks).

I. OBJECTIVES

- To review probability concepts to fully understand Bayesian Belief Networks. .

2. DEMONSTRATION OF USEFUL RESOURCES

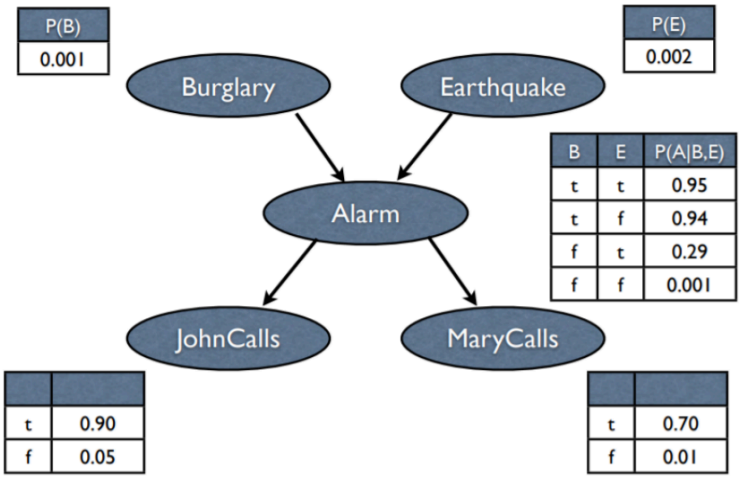
Bayesian Belief Networks and their Components:

- Bayesian Belief Networks are simple, graphical notation for conditional independence assertions.
- Bayesian network models capture both conditionally dependent and conditionally independent relationships between random variables.
- They also compactly specify the joint distributions.
- They provide a graphical model of causal relationship on which learning can be performed.

Let us consider the below mentioned example to explain Directed Acyclic Graphs and Conditional Probability Tables:

Let us consider a problem where:

- There is an **Alarm** in a house, which can be set of by events: **Burglary** and **Earthquake** with certain conditional probabilities.
- The owner of the house has gone for work to office.
- The 2 neighbours are **Mary** and **John**, who call the owner if they hear an alarm go off with certain conditional probabilities.



3. Attach the screenshot of the code.

4. Attach the screenshot of the output.

5. Conclusion

```
!pip uninstall pgmpy -y

!pip install pgmpy==0.1.18
```

```
Found existing installation: pgmpy 0.1.18
Uninstalling pgmpy-0.1.18:
  Successfully uninstalled pgmpy-0.1.18
Collecting pgmpy==0.1.18
  Using cached pgmpy-0.1.18-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (3.4.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (1.14.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (2.2.2)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (3.2.3)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (2.6.0+cu124)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (0.14.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (4.67.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from pgmpy==0.1.18) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy==0.1.18) (2.8.2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy==0.1.18) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy==0.1.18) (2025.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->pgmpy==0.1.18) (3
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels->pgmpy==0.1.18) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels->pgmpy==0.1.18) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (4.1
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (2025.3.2)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.1
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (9
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18)
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (1
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.1
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (2.21
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (12
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy==0.1.18) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch->pgmpy==0.1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->pgmpy==0.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch->pgmpy==0.1.18) (3.0.2
Using cached pgmpy-0.1.18-py3-none-any.whl (1.9 MB)
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.18
```

```
!pip uninstall numpy -y
!pip install numpy==1.26.4
```

```
Found existing installation: numpy 1.26.4
Uninstalling numpy-1.26.4:
  Successfully uninstalled numpy-1.26.4
Collecting numpy==1.26.4
  Using cached numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
Using cached numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
Installing collected packages: numpy
Successfully installed numpy-1.26.4
```

```
import pandas as pd
import numpy as np
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import HillClimbSearch, BicScore, MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination
import seaborn as sns
```

```
import numpy as np
print(np.__version__)
print(hasattr(np, "product"))  # Should be True
```

```
1.26.4
True
```

```
# Load the heart disease dataset
df = pd.read_csv("heart.csv")  # Make sure this is the Kaggle UCI dataset
```

```
# Display first few rows
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
# Check for null values
df.isnull().sum()

# If necessary, convert categorical data to string type for clarity
df['sex'] = df['sex'].map({0: 'female', 1: 'male'})
df['fbs'] = df['fbs'].map({0: 'false', 1: 'true'})
df['exang'] = df['exang'].map({0: 'no', 1: 'yes'})



hc = HillClimbSearch(df)
best_model = hc.estimate(scoring_method=BicScore(df))

model = BayesianNetwork(best_model.edges())
model.edges()
```

```
  0%                                                  11/1000000 [00:01<24:05:34, 11.53it/s]
OutEdgeView([('sex', 'thal'), ('sex', 'target'), ('sex', 'ca'), ('thal', 'target'), ('target', 'cp'), ('target', 'ca'), ('target', 'oldpeak'), ('target', 'exang'), ('target', 'restecg'), ('cp', 'exang'), ('oldpeak', 'slope')])
```

```
# Learn CPDs (Conditional Probability Distributions)
model.fit(df, estimator=MaximumLikelihoodEstimator)


print("Nodes in the model:", model.nodes())
```

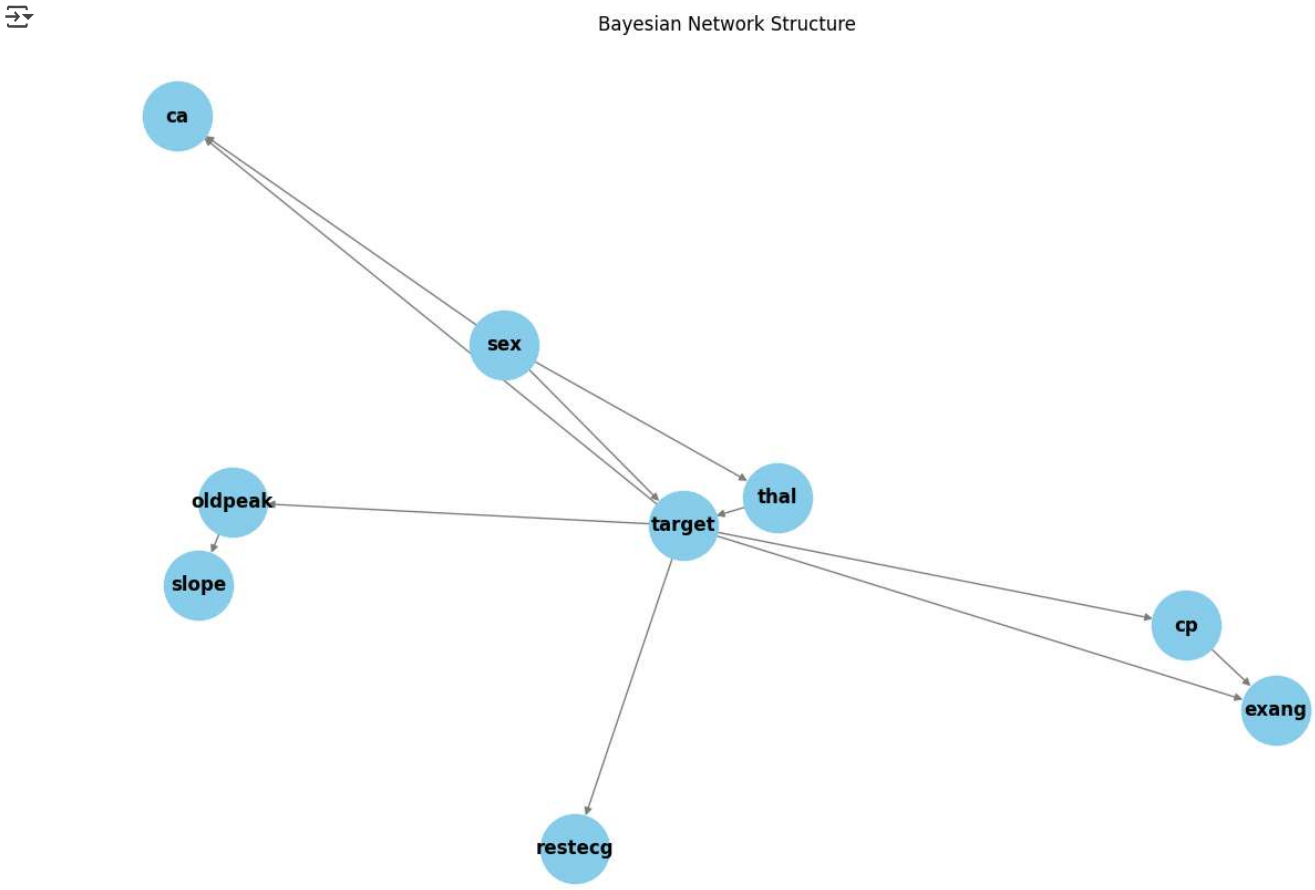
```
Nodes in the model: ['sex', 'thal', 'target', 'ca', 'cp', 'exang', 'oldpeak', 'slope', 'restecg']
```

```
infer = VariableElimination(model)

# Example Query: Probability of heart disease given age and sex
q1 = infer.query(variables=['target'], evidence={'sex': 1})  # 1 = male
print(q1)

# Another Example: Effect of chest pain (cp) on target
q2 = infer.query(variables=['target'], evidence={'cp': 3})
print(q2)
```

```
/usr/local/lib/python3.11/dist-packages/pgmpy/factors/discrete/DiscreteFactor.py:535: UserWarning: Found unknown state name. Trying
  warnings.warn(
Finding Elimination Order: : 100%                                   1/1 [00:02<00:00,  2.63s/it]
Eliminating: thal: 100%                                             1/1 [00:00<00:00, 40.50it/s]
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.5792 |
+-----------+---------------+
| target(1) |        0.4208 |
+-----------+---------------+
Finding Elimination Order: : 100%                                   2/2 [00:02<00:00,  1.27s/it]
Eliminating: sex: 100%                                              2/2 [00:00<00:00, 110.54it/s]
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.3377 |
+-----------+---------------+
| target(1) |        0.6623 |
+-----------+---------------+
```

```
import networkx as nx
import matplotlib.pyplot as plt
```

```
G = nx.DiGraph()

# Add edges from the model
G.add_edges_from(model.edges())

# Draw the network
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G, seed=42)  # Use seed for consistent layout
nx.draw(G, pos,
        with_labels=True,
        node_size=2000,
        node_color="skyblue",
        font_size=12,
        font_weight='bold',
        edge_color="gray",
        arrows=True)

plt.title("Bayesian Network Structure")
plt.show()
```




```
# Load the uploaded dataset
df = pd.read_csv("data.csv")


print("Dataset Columns:", df.columns.tolist())
print(df.head())
```

```
Dataset Columns: ['Burglary', 'Earthquake', 'Alarm', 'JohnCalls', 'MaryCalls']
   Burglary  Earthquake  Alarm  JohnCalls  MaryCalls
0      True       False   True       True      False
1     False       False  False      False      False
2      True        True   True       True       True
3     False        True   True      False       True
4     False       False   True       True       True
```

```
model = BayesianNetwork([
    ('Burglary', 'Alarm'),
    ('Earthquake', 'Alarm'),
    ('Alarm', 'JohnCalls'),
```

```
    ('Alarm', 'MaryCalls')
])


model.fit(df, estimator=MaximumLikelihoodEstimator)


print("Nodes:", model.nodes())
print("Edges:", model.edges())
```

```
Nodes: ['Burglary', 'Alarm', 'Earthquake', 'JohnCalls', 'MaryCalls']
Edges: [('Burglary', 'Alarm'), ('Alarm', 'JohnCalls'), ('Alarm', 'MaryCalls'), ('Earthquake', 'Alarm')]
```

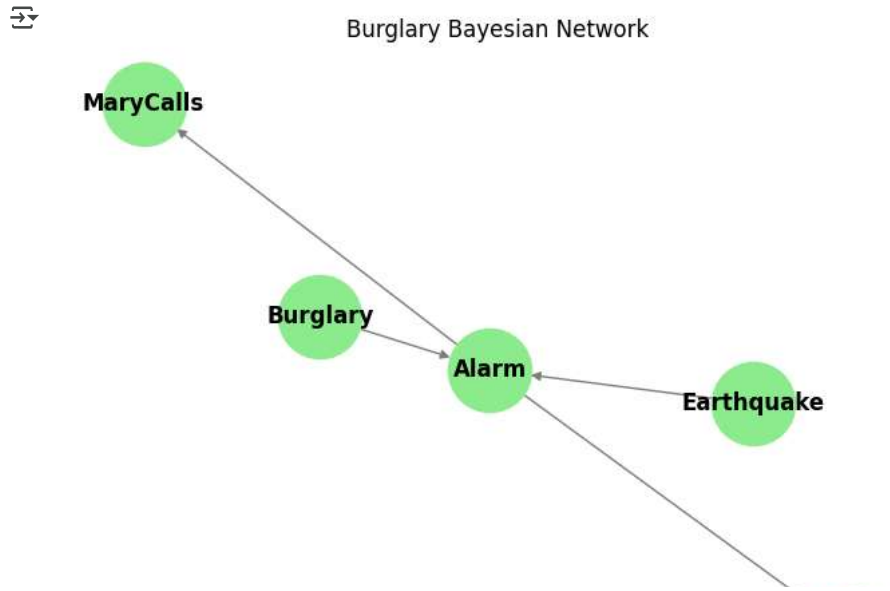
```
infer = VariableElimination(model)


q1 = infer.query(variables=['Burglary'], evidence={'JohnCalls': 1, 'MaryCalls': 1})
print("\nProbability of Burglary given both John and Mary call:")
print(q1)
```

```
Finding Elimination Order: : 100%                                   2/2 [00:46<00:00, 23.08s/it]
Eliminating: Alarm: 100%                                            2/2 [00:00<00:00, 81.10it/s]

Probability of Burglary given both John and 1Mary call:
+-----------------+-----------------+
| Burglary        |   phi(Burglary) |
+=================+=================+
| Burglary(False) |          0.5000 |
+-----------------+-----------------+
| Burglary(True)  |          0.5000 |
```

```
q2 = infer.query(variables=['Alarm'], evidence={'Earthquake': 1})
print("\nProbability of Alarm given Earthquake:")
print(q2)
```

```
Finding Elimination Order: : 100%                                   1/1 [00:56<00:00, 56.38s/it]
Eliminating: Burglary: 100%                                         1/1 [00:00<00:00, 49.90it/s]

Probability of Alarm given Earthquake:
+--------------+--------------+
| Alarm        |   phi(Alarm) |
+==============+==============+
| Alarm(False) |       0.0000 |
+--------------+--------------+
| Alarm(True)  |       1.0000 |
```

```
G = nx.DiGraph(model.edges())
pos = nx.spring_layout(G, seed=42)
nx.draw(G, pos, with_labels=True, node_size=2000, node_color="lightgreen", font_size=12, font_weight='bold', edge_color="gray")
plt.title("Burglary Bayesian Network")
plt.show()
```


Burglary Bayesian Network
MaryCalls
Burglary
Alarm
Earthquake