Name: **GANESH SANJAY PANDHRE**

Roll Nos : **41** | Div: **D20B**

AIM: To Implement Inferencing with Bayesian Network in python

---

Experiment 1: Bayesian Network using Python

Theory

A **Bayesian Network** is a type of model used in AI to deal with **uncertainty**. It helps us figure out the chances (probabilities) of things happening based on known information.

It is made up of:

- **Variables (like Rain, Fever, Disease)**
- **Probabilities assigned to these variables**
- A **graph (called Directed Acyclic Graph – DAG)** that shows how these variables are connected or depend on each other.

What is a Directed Acyclic Graph (DAG)?

A DAG is just a diagram with arrows. It shows how one variable depends on another:

- The arrows go in one direction.
- There are **no loops** (you can't go in circles).

Example: If **Rain** → **WetGround**, it means whether the ground is wet depends on if it rained.

Basic Math Behind It

There are three main concepts:

1. **Conditional Probability**
    - This is the chance of something happening **given** that something else already happened.
    - Example: P(Rain | Clouds) → Probability of rain, given that it's cloudy.
2. **Joint Probability**
    - This is the chance of **two things happening together**.
    - Example: P(Rain and WetGround)
3. **Posterior Probability**

- This is the updated chance of something **after** looking at new evidence.
- Example: If it's cloudy and humid, we might increase our belief that it will rain.

Why Use Bayesian Networks?

- They are useful when we have **uncertain, incomplete, or noisy data**.
- They are used in **medical diagnosis**, **weather forecasting**, **robotics**, **game AI**, and more.
- They allow us to **predict**, **reason**, and **learn from data**.

Implementation Summary

In our experiment, we created a very simple Bayesian Network with just **one variable**:

Variable: `Rain`

- P(Rain = Yes) = 0.3
- P(Rain = No) = 0.7

We used the `pgmpy` library in Python, which is a toolkit for building and using Bayesian Networks.

Steps followed:

1. Installed the `pgmpy` library.
2. Created a model and added the `Rain` node.
3. Assigned the probabilities using `TabularCPD`.
4. Used `VariableElimination` to get the final output (inference).

```
!pip install pgmpy
```

Show hidden output

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

```
# STEP 3: Define the model and add the variable
model = DiscreteBayesianNetwork()
model.add_node('Rain')  # ✅ Add the node before assigning a CPD
```

```
# STEP 4: Define and add the CPD
cpd_rain = TabularCPD(variable='Rain', variable_card=2, values=[[0.7], [0.3]])  # No Rain, R
model.add_cpds(cpd_rain)
```

```
# STEP 5: Validate model
assert model.check_model()
print("Model is valid ✅")
```

```
Model is valid ✅
```

```
# STEP 6: Inference
inference = VariableElimination(model)
result = inference.query(variables=['Rain'])
print(result)
```

```
+---------+-------------+
| Rain    |   phi(Rain) |
+=========+=============+
| Rain(0) |      0.7000 |
+---------+-------------+
| Rain(1) |      0.3000 |
+---------+-------------+
```

Conclusion

In this experiment, we learned:

- The **basics of Bayesian Networks** using one simple variable.
- How **probability and logic** are used to model uncertainty.
- How to **implement** a Bayesian model using Python and `pgmpy`.

This forms the foundation to build more complex networks in the future involving multiple variables and dependencies.
