Name : Naikwadi Yash Shivdas

AI&DS II Experiment 01

AIM: To Implement Inferencing with Bayesian Network in python.

THEORY:

A Bayesian Network (also known as a Belief Network or Bayes Net) is a probabilistic graphical model that represents a set of variables and their conditional dependencies using a Directed Acyclic Graph (DAG). It is a powerful tool used to model uncertainty in decision-making and prediction tasks.

Components of a Bayesian Network

1. *Nodes (Variables): Each node represents a random variable that can be discrete or continuous. Example: IQ level, Exam difficulty, Student's Marks, Admission.*
2. *Edges (Dependencies): Directed edges represent conditional dependencies between variables. For instance, Marks may depend on IQ and Exam level.*
3. *Conditional Probability Tables (CPTs): Every node has a CPT that quantifies the effect of the parent nodes on it.*

What is Inferencing?

Inferencing means computing the posterior probability of one or more variables based on observed data (evidence). For example:
"Given that a student has high IQ and the exam was easy, what is the probability of getting admitted?"

This is mathematically computed using Bayes' Theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

- *P(A|B) is the probability of A given B (posterior).*
- *P(B|A) is the probability of B given A (likelihood).*
- *P(A) and P(B) are prior probabilities.*

Bayesian Networks simplify such computations using the chain rule for joint probability:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{Parents}(X_i))$$

Example Structure

Let's consider a student's performance prediction model:

- *IQ and Exam affect Marks.*
- *Marks affect Admission.*

The DAG (Directed Acyclic Graph) for this model:

```
IQ     Exam
 \    /
  \  /
  Marks
    |
 Admission
```

Example CPTs (Conditional Probabilities):

- *P(IQ):*
  - *High: 0.3, Low: 0.7*
- *P(Exam):*
  - *Easy: 0.4, Difficult: 0.6*
- *P(Marks | IQ, Exam):*
  - *High marks more likely with high IQ and easy exam*
- *P(Admission | Marks):*
  - *High probability of admission with high marks*

Inference Using Python (pgmpy Library)

To implement inference:

1. *Define the Bayesian Network structure.*
2. *Add the Conditional Probability Distributions (CPDs).*
3. *Use* ***Variable Elimination*** *for exact inference.*
4. *Query unknown variables based on known evidence.*

Recommended Diagrams for Document

1. Chain Rule for Joint Probability

Represent how the joint distribution is broken down using parent-child relationships:

$P(A,B,C,D)=P(A)\cdot P(B|A)\cdot P(C|A)\cdot P(D|B,C)$

2. Example CPT Table

Like this (insert table in your Word/Doc):

IQ	***Exam***	***Marks High***	***Marks Low***
High	*Easy*	*0.9*	*0.1*
High	*Hard*	*0.5*	*0.5*
Low	*Easy*	*0.6*	*0.4*
Low	*Hard*	*0.2*	*0.8*

2. Applications of Bayesian Networks

- ***Medical Diagnosis*** *(e.g. disease prediction)*
- ***Spam Filtering*** *(email classification)*
- ***Risk Assessment*** *(finance, insurance)*
- ***Fault Detection*** *(complex systems)*
- ***Student Performance Prediction*** *(like our example)*

CODE:

Install pgmpy

```
!pip install pgmpy networkx matplotlib
```

Import Required Libraries

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

Define Bayesian Network & CPDs

```
# Define network structure (DAG)
model      =      DiscreteBayesianNetwork([('IQ','Marks'),      ('Exam','Marks'),
('Marks','Admission')])
```

```
# Define the CPDs (Conditional Probability Tables)
cpd_iq = TabularCPD(variable='IQ', variable_card=2,
                    values=[[0.7], [0.3]],
                    state_names={'IQ': ['low', 'high']})
```

```
cpd_exam = TabularCPD(variable='Exam', variable_card=2,
                      values=[[0.6], [0.4]],
                      state_names={'Exam': ['difficult', 'easy']})
```

```
cpd_marks = TabularCPD(variable='Marks', variable_card=2,
                       values=[
                           [0.9, 0.4, 0.5, 0.1],  # low
                           [0.1, 0.6, 0.5, 0.9]   # high
                       ],
                       evidence=['IQ','Exam'],
                       evidence_card=[2, 2],
                       state_names={'Marks': ['low', 'high'],
                                    'IQ': ['low','high'],
                                    'Exam': ['difficult','easy']})
```

```
cpd_adm = TabularCPD(variable='Admission', variable_card=2,
                     values=[
                         [0.8, 0.3],  # no
                         [0.2, 0.7]   # yes
                     ],
                     evidence=['Marks'],
                     evidence_card=[2],
                     state_names={'Admission': ['no','yes'],
                                  'Marks': ['low','high']})
```

```
# Add CPDs to the model
model.add_cpds(cpd_iq, cpd_exam, cpd_marks, cpd_adm)
```

```
# Validate the model
model.check_model()
```

```
True
```

Perform Inference

```
inference = VariableElimination(model)

# Example 1: Probability of Admission given IQ = high and Exam = easy
result1  =  inference.query(variables=['Admission'],  evidence={'IQ':  'high',
'Exam': 'easy'})
print("P(Admission | IQ=high, Exam=easy):\n", result1)
```

```
P(Admission | IQ=high, Exam=easy):
 +----------------+------------------+
| Admission      |   phi(Admission) |
+================+==================+
| Admission(no)  |           0.3500 |
+----------------+------------------+
| Admission(yes) |           0.6500 |
+----------------+------------------+
```

```
# Example 2: Probability of Marks without evidence
result2 = inference.query(variables=['Marks'])
print("P(Marks):\n", result2)
```

```
P(Marks):
 +-------------+--------------+
| Marks       |   phi(Marks) |
+=============+==============+
| Marks(low)  |       0.5920 |
+-------------+--------------+
| Marks(high) |       0.4080 |
+-------------+--------------+
```

```
# Example 3: Probability of Admission given Marks = high
result3  =  inference.query(variables=['Admission'],  evidence={'Marks':
'high'})
print("P(Admission | Marks=high):\n", result3)
```

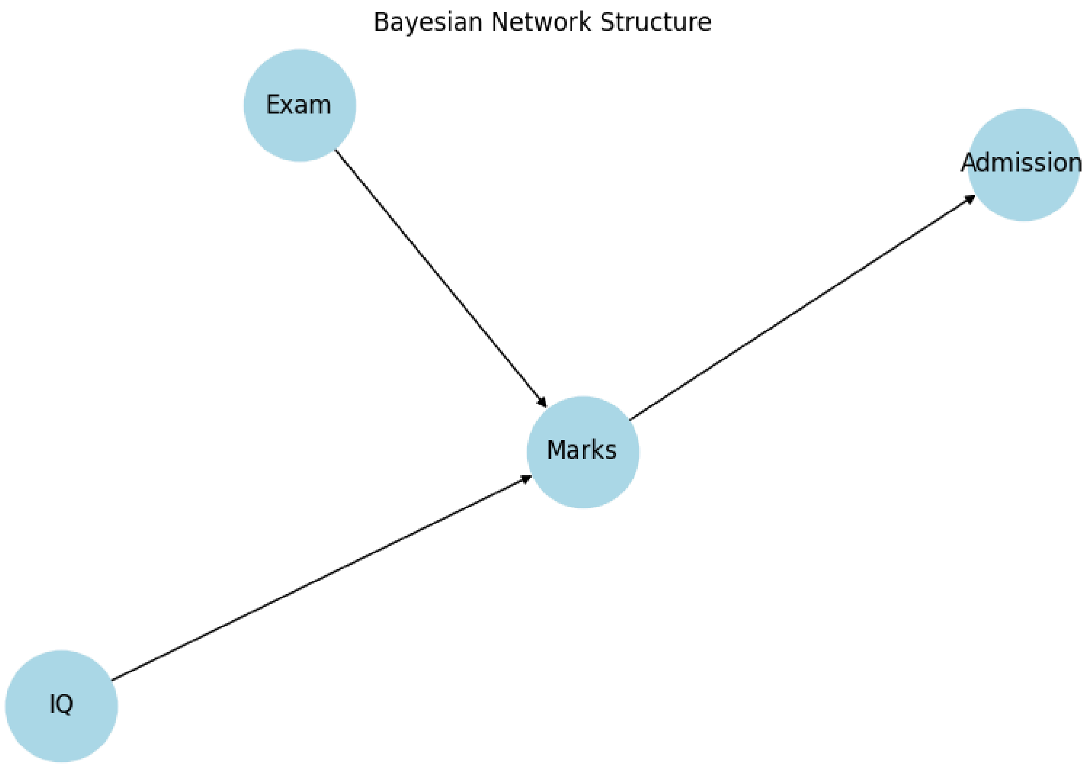
```
P(Admission | Marks=high):
 +----------------+------------------+
| Admission      |   phi(Admission) |
+=================+==================+
| Admission(no)   |           0.3000 |
+-----------------+------------------+
| Admission(yes)  |           0.7000 |
+-----------------+------------------+
```

Visualize Network

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

# Create a NetworkX graph from the model structure
G = nx.DiGraph()
G.add_edges_from(model.edges())

# Draw the graph
plt.figure(figsize=(8, 5))
pos = nx.spring_layout(G)  # or use nx.circular_layout(G)
nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=3000,
font_size=12, arrows=True)
plt.title("Bayesian Network Structure")
plt.show()
```



CONCLUSION:

Bayesian Networks offer a graphical and mathematical way to model real-world uncertainties and relationships between multiple variables. By using directed graphs and probability theory, we can predict outcomes, diagnose causes, and make informed decisions based on evidence.

In Python, libraries like pgmpy make it easy to define networks, set probabilities, and perform exact inference. This allows us to explore how changes in one variable affect others in a system modeled by uncertainty. 39_colab_01