Experiment-1

Name: Sharvari Kishor More
Class: D20B
Roll no: 35

Aim: To implement inference with a Bayesian Network in Python.

Theory:
Bayesian Network (BN)
A Bayesian Network is a probabilistic graphical model that represents a set of variables and their conditional dependencies using a directed acyclic graph (DAG). It is widely used for reasoning under uncertainty.
Each node in the network represents a random variable, and the edges represent conditional dependencies. Probabilities are assigned in the form of Conditional Probability Distributions (CPDs).
Student Performance Prediction using Bayesian Network

Variables Used

Variable	States
StudyHours	Low (0), High (1)
Attendance	Low (0), High (1)
ExamPerformance	Bad (0), Good (1)
FinalGrade	Fail (0), Pass (1)

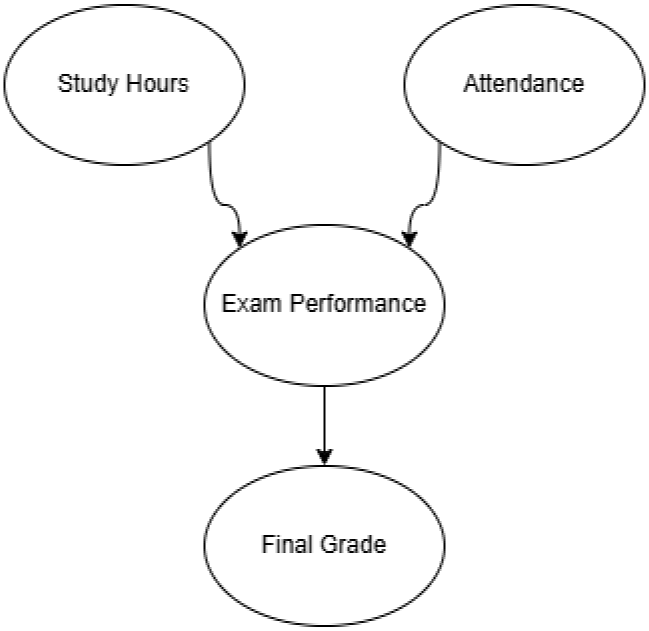
CPDs (Conditional Probability Tables)

- P(StudyHours): Prior probability of students studying.
- P(Attendance): Prior probability of attending class.
- P(ExamPerformance | StudyHours, Attendance): Conditional probability based on both factors.
- P(FinalGrade | ExamPerformance): Grade depends on performance.

Implementation Steps

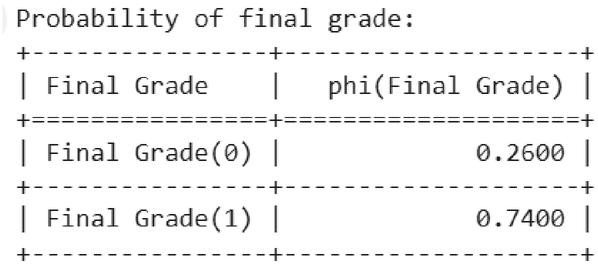
1. Import required libraries (pgmpy)
2. Define the network structure using BayesianNetwork()
3. Define CPDs using TabularCPD() for each variable
4. Add CPDs to the model and validate it
5. Use VariableElimination() for performing inference
6. Query the model for probabilities given some evidence

Diagram:



Inference Example
We query the model:

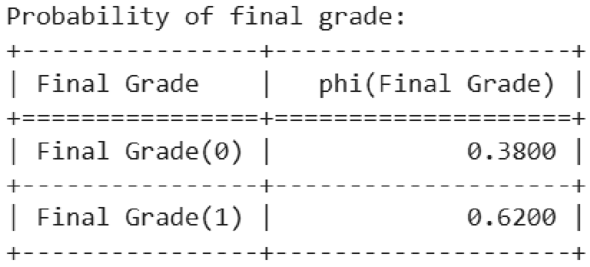
1. What is the probability of passing the final exam if the student had High Study Hours and High Attendance?



```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.2600 |
+----------------+--------------------+
| Final Grade(1) |             0.7400 |
+----------------+--------------------+
```

This indicates that the student has 0.74 chances of passing the examination.

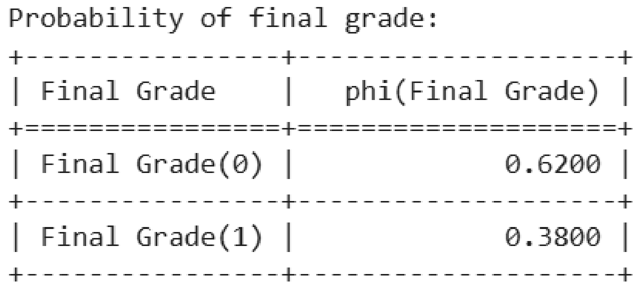
2. What is the probability of passing the final exam if the student had High Study Hours and Low Attendance?



```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.3800 |
+----------------+--------------------+
| Final Grade(1) |             0.6200 |
+----------------+--------------------+
```

This indicates that the student has 0.62 chances of passing the examination.

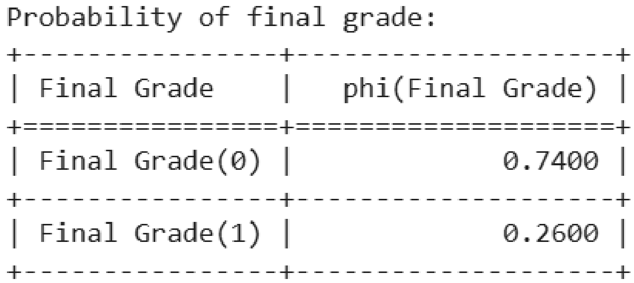
3. What is the probability of passing the final exam if the student had Low Study Hours and High Attendance?



```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.6200 |
+----------------+--------------------+
| Final Grade(1) |             0.3800 |
+----------------+--------------------+
```

This indicates that the student has 0.38 chances of passing the examination.

4. What is the probability of passing the final exam if the student had Low Study Hours and Low Attendance?



```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.7400 |
+----------------+--------------------+
| Final Grade(1) |             0.2600 |
+----------------+--------------------+
```

This indicates that the student has 0.26 chances of passing the examination.

Below is the implementation in Colab.

Experiment 1

Name: Sharvari More

Class: D20B

Roll no.: 35

Bayesian Network

Aim: To implement inferenceing with bayesian network in python

Importing libraries

```
!pip install pgmpy

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

```
Collecting pgmpy
  Downloading pgmpy-1.0.0-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from pgmpy) (3.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from pgmpy) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from pgmpy) (1.16.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from pgmpy) (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from pgmpy) (2.2.2)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (from pgmpy) (2.6.0+cu124)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (from pgmpy) (0.14.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from pgmpy) (4.67.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from pgmpy) (1.5.1)
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.11/dist-packages (from pgmpy) (3.4.0)
Collecting pyro-ppl (from pgmpy)
  Downloading pyro_ppl-1.9.1-py3-none-any.whl.metadata (7.8 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->pgmpy) (2025.2)
Collecting pyro-api>=0.1.1 (from pyro-ppl->pgmpy)
  Downloading pyro_api-0.1.2-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy) (2025.3.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->pgmpy)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->pgmpy)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->pgmpy)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->pgmpy)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->pgmpy)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->pgmpy)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->pgmpy)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->pgmpy)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->pgmpy)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch->p
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->pgmp
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->pgmpy)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch->pgmpy) (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->torc
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->pgm
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels->pgmpy) (1.0.
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels->pgmpy) (2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->panda
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch->pgmpy)
Downloading pgmpy-1.0.0-py3-none-any.whl (2.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 27.8 MB/s eta 0:00:00
Downloading pyro_ppl-1.9.1-py3-none-any.whl (755 kB)
```

Defining structure of Bayesian Network

```
model = DiscreteBayesianNetwork([
    ('Study Hours','Exam Performance'),
    ('Attendance','Exam Performance'),
    ('Exam Performance','Final Grade')
])
```

Defining CPDs (Conditional Probability Tables)

```
cpd_study = TabularCPD(variable='Study Hours', variable_card=2, values=[[0.7],[0.3]])
```

```
cpd_attendance = TabularCPD(variable='Attendance', variable_card=2, values=[[0.6],[0.4]])
```

Calculating Exam Performance

```
cpd_exam=TabularCPD(
    variable='Exam Performance',variable_card=2,
    values=[
        [0.9,0.7,0.3,0.1],
        [0.1,0.3,0.7,0.9]
    ],
    evidence=['Study Hours','Attendance'],
    evidence_card=[2,2]
)
```

Final Grade: Fail=0, Pass=1

```
cpd_grade = TabularCPD(
    variable='Final Grade',
    variable_card=2,
    values=[
        [0.8,0.2],
        [0.2,0.8]
    ],
    evidence=['Exam Performance'],
    evidence_card=[2]
)
```

Add CPDs and check

```
model.add_cpds(cpd_study,cpd_attendance,cpd_exam,cpd_grade)
assert model.check_model(),"Model is not valid"
```

Inference

```
inference = VariableElimination(model)
```

Query

```
result = inference.query(variables=['Final Grade'], evidence={'Study Hours': 1, 'Attendance': 1})
print("Probability of final grade:")
print(result)
```

```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.2600 |
+----------------+--------------------+
| Final Grade(1) |             0.7400 |
+----------------+--------------------+
```

```
result = inference.query(variables=['Final Grade'], evidence={'Study Hours': 1, 'Attendance': 0})
print("Probability of final grade:")
print(result)
```

```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.3800 |
+----------------+--------------------+
| Final Grade(1) |             0.6200 |
+----------------+--------------------+
```

```
result = inference.query(variables=['Final Grade'], evidence={'Study Hours': 0, 'Attendance': 1})
print("Probability of final grade:")
print(result)
```

```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.6200 |
+----------------+--------------------+
| Final Grade(1) |             0.3800 |
+----------------+--------------------+
```

```
result = inference.query(variables=['Final Grade'], evidence={'Study Hours': 0, 'Attendance': 0})
print("Probability of final grade:")
print(result)
```

```
Probability of final grade:
+----------------+--------------------+
| Final Grade    |   phi(Final Grade) |
+================+====================+
| Final Grade(0) |             0.7400 |
+----------------+--------------------+
| Final Grade(1) |             0.2600 |
+----------------+--------------------+
```

Conclusion: This experiment demonstrates how Bayesian Networks can be used to model uncertain real-world situations like student performance. By defining the relationships and probabilities between input factors, we can use probabilistic inference to make reliable predictions

```
Start coding or generate with AI.
```