!pip install pgmpy

from pgmpy.models import BayesianModel

from pgmpy.factors.discrete import TabularCPD

from pgmpy.inference import VariableElimination

# Define the structure of the Bayesian Network

model = BayesianModel([('A', 'B'), ('C', 'B'), ('B', 'D')])

# Define the Conditional Probability Distributions (CPDs)

cpd\_a = TabularCPD(variable='A', variable\_card=2, values=[[0.6], [0.4]])

cpd\_c = TabularCPD(variable='C', variable\_card=2, values=[[0.7], [0.3]])

cpd\_b = TabularCPD(variable='B', variable\_card=2,

values=[[0.8, 0.9, 0.3, 0.4], [0.2, 0.1, 0.7, 0.6]],

evidence=['A', 'C'], evidence\_card=[2, 2])

cpd\_d = TabularCPD(variable='D', variable\_card=2,

values=[[0.9, 0.2], [0.1, 0.8]],

evidence=['B'], evidence\_card=[2])

# Add CPDs to the model

model.add\_cpds(cpd\_a, cpd\_c, cpd\_b, cpd\_d)

# Check if the model is valid

model.check\_model()

# Perform inference using Variable Elimination

inference = VariableElimination(model)

posterior\_d = inference.query(variables=['D'], evidence={'A': 1, 'C': 0})

print(posterior\_d)

Algorithm:

Certainly, here's the algorithm for a Bayesian Belief Network in a shorter form:

1. \*\*Define Variables and Dependencies:\*\* Identify variables and their relationships.

2. \*\*Construct Graph:\*\* Create a directed acyclic graph (DAG) of variables and dependencies.

3. \*\*Assign Probabilities:\*\* Define conditional probabilities for variables based on dependencies.

4. \*\*Perform Inference:\*\* Use the network to calculate probabilities or make predictions.

5. \*\*Learn (Optional):\*\* If needed, learn the network structure and parameters from data.

6. \*\*Update with Evidence:\*\* Incorporate new evidence to update probabilities and make decisions.

7. \*\*Evaluate Performance:\*\* Assess the accuracy and reliability of the network's predictions.

8. \*\*Refine (Optional):\*\* Iterate to improve the model based on feedback and new information.