6. PROGRAM TO DEMONSTRATE THE WORKING OF BAYESIAN NETWORK.

Contents

Introduction:

Creating a Bayesian network for five different problems requires a more complex setup. Below, I'll provide an example for each problem. Note that this example uses the `pomegranate` library, and you need to install it first using `pip install pomegranate`.

Problem 1: Student Admission

Imagine a university's student admission scenario where admission depends on two factors: High School GPA and Extracurricular Activities.

```
python

from pomegranate import *


# Define the Bayesian Network

high_school_gpa = DiscreteDistribution({'High': 0.7, 'Low': 0.3})

extracurricular = DiscreteDistribution({'High': 0.6, 'Low': 0.4})

admission = ConditionalProbabilityTable(

   [['High', 'High', 'Admitted', 0.9],

    ['High', 'High', 'Not Admitted', 0.1],

    ['High', 'Low', 'Admitted', 0.7],

    ['High', 'Low', 'Not Admitted', 0.3],

    ['Low', 'High', 'Admitted', 0.3],

    ['Low', 'High', 'Not Admitted', 0.7],

    ['Low', 'Low', 'Admitted', 0.1],

    ['Low', 'Low', 'Not Admitted', 0.9]],

   [high_school_gpa, extracurricular]

)


state_gpa = State(high_school_gpa, name="HighSchoolGPA")

state_extracurricular = State(extracurricular, name="ExtracurricularActivities")

state_admission = State(admission, name="Admission")
```

```
network = BayesianNetwork("Student Admission Network")
network.add_states(state_gpa, state_extracurricular, state_admission)
network.add_edge(state_gpa, state_admission)
network.add_edge(state_extracurricular, state_admission)
network.bake()

# Define the evidence
evidence = {'HighSchoolGPA': 'High', 'ExtracurricularActivities': 'High'}

# Perform inference
beliefs = network.predict_proba(evidence)
for state, belief in zip(network.states, beliefs):
    print(f"{state.name}: {belief.parameters[0]}")
```

Problem 2: Disease Diagnosis

Consider a medical scenario where the diagnosis of a disease depends on symptoms and medical history.

```
#python
from pomegranate import *

# Define the Bayesian Network
symptoms = DiscreteDistribution({'Present': 0.4, 'Absent': 0.6})
medical_history = DiscreteDistribution({'Positive': 0.3, 'Negative': 0.7})
diagnosis = ConditionalProbabilityTable(
    [['Present', 'Positive', 'Disease', 0.9],
     ['Present', 'Positive', 'No Disease', 0.1],
     ['Present', 'Negative', 'Disease', 0.8],
     ['Present', 'Negative', 'No Disease', 0.2],
     ['Absent', 'Positive', 'Disease', 0.2],
     ['Absent', 'Positive', 'No Disease', 0.8],
     ['Absent', 'Negative', 'Disease', 0.01],
     ['Absent', 'Negative', 'No Disease', 0.99]],
    [symptoms, medical_history]
)

state_symptoms = State(symptoms, name="Symptoms")
state_medical_history = State(medical_history, name="MedicalHistory")
state_diagnosis = State(diagnosis, name="Diagnosis")

network = BayesianNetwork("Disease Diagnosis Network")
network.add_states(state_symptoms, state_medical_history, state_diagnosis)
network.add_edge(state_symptoms, state_diagnosis)
network.add_edge(state_medical_history, state_diagnosis)
network.bake()
```

```
# Define the evidence
evidence = {'Symptoms': 'Present', 'MedicalHistory': 'Positive'}


# Perform inference
beliefs = network.predict_proba(evidence)
for state, belief in zip(network.states, beliefs):
    print(f"{state.name}: {belief.parameters[0]}")
```

You can similarly adapt and extend these examples for other problems by defining appropriate variables, distributions, and conditional probability tables based on the specifics of each problem.