Lab Assignment: Bayesian Network Inference

October 25, 2025

1 Introduction and Objectives

In this lab, you will apply your knowledge of Bayesian Networks to model a real-world scenario: **campus placements**. You will construct a network from scratch based on expert-defined rules, perform inference to answer complex probabilistic queries, and analyze the model's reasoning.

Objectives:

- To model a system with causal dependencies using `pgmpy`.
- To define and implement Conditional Probability Distributions (CPDs) from a problem description.
- To perform complex inference (`query`)

2 The Scenario: Campus Placement Network

We will model the probability of a student receiving a job offer (`J`).

2.1 Variables and States

The model consists of 6 variables:

- **CGPA (C)**: `['Low', 'Medium', 'High']` (card=3)
- **Technical Skills (T)**: `['Weak', 'Strong']` (card=2)
- **Soft Skills (S)**: `['Weak', 'Strong']` (card=2)
- **Written Test (W)**: `['Fail', 'Pass']` (card=2)
- **Interview (I)**: `['Fail', 'Pass']` (card=2)
- **Job Offer (J)**: `['No', 'Yes']` (card=2)

2.2 Network Structure (Dependencies)

The relationships are defined as follows:

- The **Written Test** is influenced by both **CGPA** and **Technical Skills**.
- The **Interview** is influenced by both **Technical Skills** and **Soft Skills**.
- The **Job Offer** is influenced by both the **Written Test** and the **Interview**.
- `CGPA`, `Technical_Skills`, and `Soft_Skills` are independent root nodes.

---

3 Part 1: Model Construction

Your first task is to build the model based on the structure and probabilities provided.

3.1 Task 1.1: Visualize the Network

Using `networkx`, draw the directed graph (DAG) for this Bayesian Network. Ensure nodes are clearly labeled and arrows show the correct dependencies.

3.2 Task 1.2: Define Structure & CPDs

Define the model structure using `DiscreteBayesianNetwork`. Then, define and add all 6 `TabularCPDs` using the probabilities below. Use `state_names` for clarity.

Root Node Probabilities:

- **P(C)**: `[[0.2], [0.5], [0.3]]` (for Low, Medium, High)
- **P(T)**: `[[0.4], [0.6]]` (for Weak, Strong)
- **P(S)**: `[[0.5], [0.5]]` (for Weak, Strong)

Conditional Probabilities: **A Note on Column Ordering:** `pgmpy` orders the `values` columns by iterating through the `evidence` list. For `evidence=['C', 'T']` with `evidence_card=[3, 2]`, the 6 columns will be:

- (C=Low, T=Weak), (C=Medium, T=Weak), (C=High, T=Weak)
- (C=Low, T=Strong), (C=Medium, T=Strong), (C=High, T=Strong)

Follow this logic for all conditional CPDs.

1. CPD for Written Test: P(W — C, T)

- `evidence=['C', 'T']`
- `evidence_card=[3, 2]`
- `values` (Rows: W=Fail, W=Pass):
  `[[0.8, 0.6, 0.3, 0.4, 0.2, 0.1],` *# W=Fail*
  `[0.2, 0.4, 0.7, 0.6, 0.8, 0.9]]` *# W=Pass*

```
# P(W | C, T)
written_cpd = TabularCPD(
    variable='W', variable_card=2,
    values=[[0.8, 0.6, 0.3, 0.4, 0.2, 0.1],
            [0.2, 0.4, 0.7, 0.6, 0.8, 0.9]],
    evidence=['C', 'T'], evidence_card=[3, 2],
    state_names={'W': ['Fail', 'Pass'],
                 'C': ['Low', 'Medium', 'High'],
                 'T': ['Weak', 'Strong']})
```

2. CPD for Interview: P(I — T, S)

- `evidence=['T', 'S']`
- `evidence_card=[2, 2]`
- `values` (Rows: I=Fail, I=Pass):
  `[[0.7, 0.4, 0.3, 0.1],` *# I=Fail*
  `[0.3, 0.6, 0.7, 0.9]]` *# I=Pass*

```
# P(I | T, S)
interview_cpd = TabularCPD(
    variable='I', variable_card=2,
    values=[[0.7, 0.4, 0.3, 0.1],
            [0.3, 0.6, 0.7, 0.9]],
    evidence=['T', 'S'], evidence_card=[2, 2],
    state_names={'I': ['Fail', 'Pass'],
                 'T': ['Weak', 'Strong'],
                 'S': ['Weak', 'Strong']})
```

3. CPD for Job Offer: P(J — W, I)

- `evidence=['W', 'I']`
- `evidence_card=[2, 2]`
- `values` (Rows: J=No, J=Yes):
  `[[0.99, 0.8, 0.6, 0.1],` *# J=No*
  `[0.01, 0.2, 0.4, 0.9]]` *# J=Yes*

```
# P(J | W, I)
offer_cpd = TabularCPD(
    variable='J', variable_card=2,
```

```
    values=[[0.99, 0.8, 0.6, 0.1],
            [0.01, 0.2, 0.4, 0.9]],
    evidence=['W', 'I'], evidence_card=[2, 2],
    state_names={'J': ['No', 'Yes'],
                 'W': ['Fail', 'Pass'],
                 'I': ['Fail', 'Pass']})
```

4 Part 2: Inference & Analysis

4.1 Task 2.1: Setup Inference

Create a `VariableElimination` object for your completed model.

4.2 Task 2.2: Probabilistic Queries

Perform the following queries and report their results.

1. **Baseline:** What is the overall probability of a student receiving a Job Offer? `P(J)`
2. **Predictive Inference:** A student has `High` CGPA and `Strong` Technical Skills. What is their probability of receiving a Job Offer?
   `P(J | C='High', T='Strong')`
3. **Diagnostic Inference:** A student received a Job Offer. What is the probability they had `Strong` Technical Skills?
   `P(T | J='Yes')`
4. **"Explaining Away" (Analysis):**
   - First, calculate: `P(W='Pass' | J='Yes')`
     (The probability a student passed the **Written Test**, given they got the job).
   - Next, calculate: `P(W='Pass' | J='Yes', I='Pass')`
     (The same probability, but now we also know they passed the **Interview**).
   - **Analyze:** How did the probability `P(W='Pass')` change when you added the new evidence about the Interview? Why do you think this happened? (Hint: Think about the v-structure $W \rightarrow J \leftarrow I$).

5 Parameter Learning

1. Generate 20,000 data samples from your `student_model`.
2. Create a new, empty `DiscreteBayesianNetwork` with only the structure.
3. Use `model.fit(data, ...)` to learn the parameters from your generated data.
4. Compare the learned CPD for `P(J | W, I)` with the CPD you defined in Part 1. Are they similar?

6 Submission

1. Your complete Jupyter Notebook or Python script (`.ipynb` or `.py`).