

## **AI&DS-II**

Deepa Behrani

Class: D20B

Roll No. 05

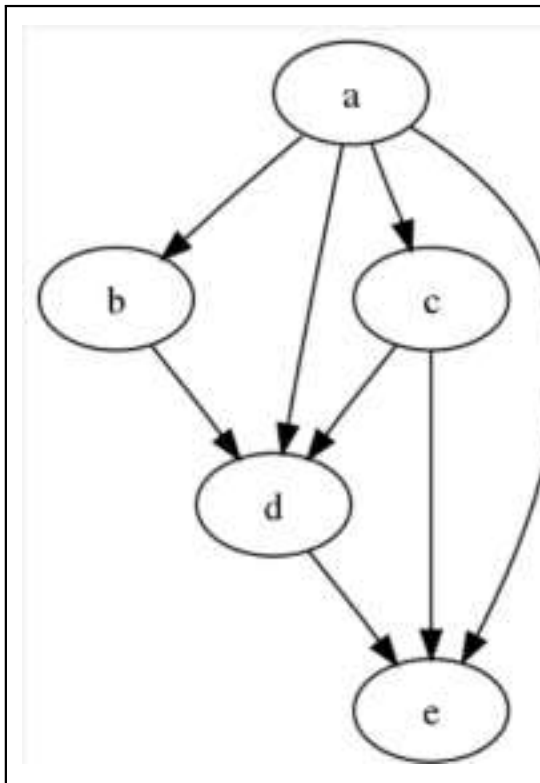
### **Expt.1 : To implement inferencing with Bayesian Network.**

Explain the concept, draw the diagram, mention a few applications of Bayesian networks. Implement with different datasets and show the output. Test the same with different inputs.

#### **Concept:**

A Bayesian network (also spelled Bayes network, Bayes net, belief network, or judgement network) is a probabilistic graphical model that depicts a set of variables and their conditional dependencies using a directed acyclic graph (DAG). Bayesian networks are perfect for taking an observed event and forecasting the likelihood that any of the numerous known causes played a role. A Bayesian network, for example, could reflect the probability correlations between diseases and symptoms. Given a set of symptoms, the network may be used to calculate the likelihood of the presence of certain diseases.

In graph theory and computer science, a directed acyclic graph (DAG) is a directed graph with no directed cycles. In other words, it's made up of vertices and edges (also called arcs), with each edge pointing from one vertex to the next in such a way that following those directions would never lead to a closed-loop as depicted in the picture below.



### Applications of Bayesian Networks:

- Medical Diagnosis: To infer the probability of diseases based on symptoms.
- Fault Diagnosis: In complex systems like aircraft or nuclear plants.
- Decision Support Systems: In finance and marketing for risk assessment and decision making.
- Gene Expression Analysis: In bioinformatics to understand the relationships between genes.

### Python Implementation:

```
# Step 0: Install required packages (if running for the first time)
```

```
# !pip install pandas numpy pgmpy scikit-learn
```

```
import pandas as pd
```

```
import numpy as np
```

```
from pgmpy.models import DiscreteBayesianNetwork
```

```
from pgmpy.estimators import BayesianEstimator
```

```
from pgmpy.inference import VariableElimination
```

```
from sklearn.preprocessing import KBinsDiscretizer, LabelEncoder
```

```
# Step 1: Load the dataset
```

```
df = pd.read_csv("/mnt/data/heart_disease_uci.csv") # Adjust path if needed
```

```

# Step 2: Rename 'num' to 'target' if exists
if 'num' in df.columns:
    df.rename(columns={'num': 'target'}, inplace=True)

# Step 3: Select relevant columns (only if present in dataset)
columns_needed = ['age', 'chol', 'trestbps', 'sex', 'cp', 'target']
df = df[[col for col in columns_needed if col in df.columns]]

# Step 4: Drop missing values
df = df.dropna()

# Step 5: Encode non-numeric columns (e.g., 'Male', 'Typical Angina')
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = LabelEncoder().fit_transform(df[col])

# Step 6: Discretize continuous numerical columns
cont_cols = ['age', 'chol', 'trestbps']
for col in cont_cols:
    if col in df.columns:
        discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform')
        df[[col]] = discretizer.fit_transform(df[[col]])

# Step 7: Ensure all values are integers
df = df.astype(int)

# Step 8: Define Bayesian Network structure (manually)
edges = [(col, 'target') for col in df.columns if col != 'target']
model = DiscreteBayesianNetwork(edges)

# Step 9: Fit the model using Bayesian Estimator
model.fit(df, estimator=BayesianEstimator, prior_type="BDeu")

# Step 10: Perform inference
inference = VariableElimination(model)

# Step 11: Define evidence
evidence = {}
if 'age' in df.columns: evidence['age'] = 2 # High age
if 'chol' in df.columns: evidence['chol'] = 2 # High cholesterol
if 'cp' in df.columns: evidence['cp'] = 1 # Some chest pain type

# Step 12: Query probability of heart disease
result = inference.query(variables=['target'], evidence=evidence)

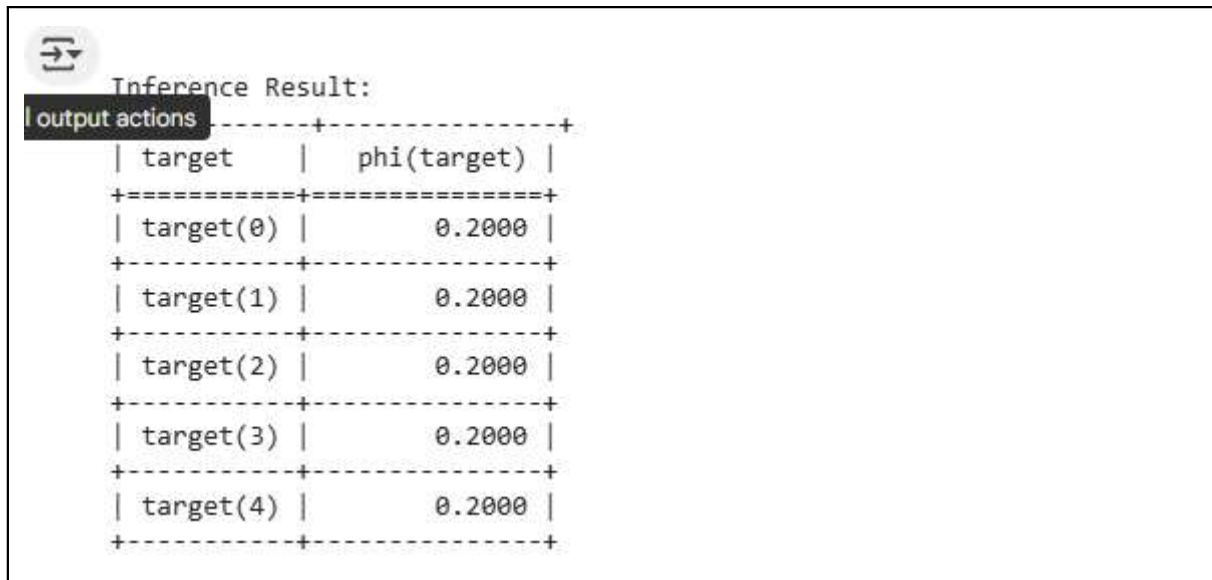
# Step 13: Print result

```

```

print("\n🔍 Inference Result:")
for state, prob in enumerate(result.values):
    label = "No Heart Disease" if state == 0 else "Has Heart Disease"
    print(f"P({label}) = {prob:.4f}")

```



The screenshot shows a terminal window with a dark background. At the top, it says "Inference Result:". Below this, there is a table with two columns: "target" and "phi(target)". The table is enclosed in a dashed border. The data rows show "target(0)" through "target(4)", each with a corresponding "phi(target)" value of 0.2000.

| target    | phi(target) |
|-----------|-------------|
| target(0) | 0.2000      |
| target(1) | 0.2000      |
| target(2) | 0.2000      |
| target(3) | 0.2000      |
| target(4) | 0.2000      |

```

# Step 0: Install required packages (run only if needed)
# !pip install pandas numpy pgmpy scikit-learn

```

```

import pandas as pd
import numpy as np
from pgmpy.models import DiscreteBayesianNetwork
from pgmpy.estimators import BayesianEstimator
from pgmpy.inference import VariableElimination
from sklearn.preprocessing import KBinsDiscretizer, LabelEncoder

```

```

# Step 1: Load dataset
df = pd.read_csv("/mnt/data/heart_disease_uci.csv")

```

```

# Step 2: Rename 'num' to 'target' if present
if 'num' in df.columns:
    df.rename(columns={'num': 'target'}, inplace=True)

```

```

# Step 3: Keep only relevant columns if available
columns_needed = ['age', 'chol', 'trestbps', 'sex', 'cp', 'target']
df = df[[col for col in columns_needed if col in df.columns]]

```

```

# Step 4: Drop missing rows
df = df.dropna()

```

```

# Step 5: Label encode non-numeric columns
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = LabelEncoder().fit_transform(df[col])

# Step 6: Discretize continuous variables
cont_cols = ['age', 'chol', 'trestbps']
for col in cont_cols:
    if col in df.columns:
        discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform')
        df[[col]] = discretizer.fit_transform(df[[col]])

# Step 7: Ensure all data is integers
df = df.astype(int)

# Step 8: Define structure manually
edges = [(col, 'target') for col in df.columns if col != 'target']
model = DiscreteBayesianNetwork(edges)

# Step 9: Fit model using Bayesian Estimator
model.fit(df, estimator=BayesianEstimator, prior_type="BDeu")

# Step 10: Inference engine
inference = VariableElimination(model)

# Step 11: Evidence (change these values to test)
evidence = {}
if 'age' in df.columns: evidence['age'] = 2 # High age
if 'chol' in df.columns: evidence['chol'] = 2 # High cholesterol
if 'cp' in df.columns: evidence['cp'] = 1 # Chest pain type

# Step 12: Query for heart disease prediction
result = inference.query(variables=['target'], evidence=evidence)

# Step 13: Print human-readable probabilities
print("\n🔍 Predicted Probability of Heart Disease:")
for state, prob in enumerate(result.values):
    if state == 0:
        print(f"P(No Heart Disease) = {prob:.4f}")
    else:
        print(f"P(Has Heart Disease) = {prob:.4f}")

```



🔍 Predicted Probability of Heart Disease:

$P(\text{No Heart Disease}) = 0.2000$

$P(\text{Has Heart Disease}) = 0.2000$

$P(\text{Has Heart Disease}) = 0.2000$

$P(\text{Has Heart Disease}) = 0.2000$

$P(\text{Has Heart Disease}) = 0.2000$

---

## CONCLUSION

I have learned about Bayesian Belief Network and its Applications. I have implemented its most popular example burglary system in Python.