

Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API).

In this step, we are importing the basic libraries required for data manipulation and Bayesian Network modeling.


- pandas will help us load and manipulate the dataset.
- pgmpy is the main library for working with probabilistic graphical models, such as Bayesian Networks.

```
!pip install pgmpy
```

```
Collecting pgmpy
  Downloading pgmpy-0.1.26-py3-none-any.whl.metadata (9.1 kB)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.13.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.3.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from pgmpy) (2.1.4)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.1.4)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from pgmpy) (2.4.0+cu121)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (from pgmpy) (0.14.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from pgmpy) (4.66.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.4.2)
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.3.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from pgmpy) (2.1.1)
Requirement already satisfied: google-generativeai in /usr/local/lib/python3.10/dist-packages (from pgmpy) (0.7.2)
Requirement already satisfied: google-ai-generativelanguage==0.6.6 in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (0.7.2)
Requirement already satisfied: google-api-core in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (2.19.2)
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (2.11.0)
Requirement already satisfied: google-auth>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (2.27.0)
Requirement already satisfied: protobuf in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (3.20.3)
Requirement already satisfied: pydantic in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (2.9.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from google-generativeai->pgmpy) (4.12.2)
Requirement already satisfied: proto-plus<2.0.0dev, >=1.22.3 in /usr/local/lib/python3.10/dist-packages (from google-ai-generativelanguage->pgmpy) (1.24.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->pgmpy) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->pgmpy) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->pgmpy) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pgmpy) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels->pgmpy) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels->pgmpy) (24.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (3.16.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (1.13.2)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (2024.6.1)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-packages (from xgboost->pgmpy) (2.23.4)
Requirement already satisfied: googleapis-common-protos<2.0.dev0, >=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core->pgmpy) (1.63.0)
Requirement already satisfied: requests<3.0.0.dev0, >=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core->pgmpy) (2.32.0)
Requirement already satisfied: cachetools<6.0, >=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth->pgmpy) (5.5.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth->pgmpy) (0.4.0)
Requirement already satisfied: rsa<5, >=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth->pgmpy) (4.9)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy->pgmpy) (1.16.0)
Requirement already satisfied: httplib2<1.dev0, >=0.19.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client->pgmpy) (0.19.0)
Requirement already satisfied: google-auth-httplib2<1.0.0, >=0.2.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client->pgmpy) (0.2.0)
Requirement already satisfied: uritemplate<5, >=3.0.1 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client->pgmpy) (4.1.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2->pgmpy) (2.1.5)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic->pgmpy) (0.6.0)
Requirement already satisfied: pydantic-core==2.23.3 in /usr/local/lib/python3.10/dist-packages (from pydantic->pgmpy) (2.23.3)
Requirement already satisfied: mpmath<1.4, >=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->pgmpy) (1.3.0)
Requirement already satisfied: grpcio<2.0dev, >=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grpc]->pgmpy) (1.63.0)
Requirement already satisfied: grpcio-status<2.0.dev0, >=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grpc]->pgmpy) (1.63.0)
Requirement already satisfied: pyasn1<0.7.0, >=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules->pgmpy) (0.6.0)
Requirement already satisfied: charset-normalizer<4, >=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0, >=2.18.0->pgmpy) (3.3.2)
Requirement already satisfied: idna<4, >=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0, >=2.18.0->pgmpy) (3.10.1)
Requirement already satisfied: urllib3<3, >=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0, >=2.18.0->pgmpy) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0, >=2.18.0->pgmpy) (2024.7.4)
Downloading pgmpy-0.1.26-py3-none-any.whl (2.0 MB)
2.0/2.0 MB 9.1 MB/s eta 0:00:00
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.26
```

```
# Importing required libraries
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

```
# Load the dataset
data = pd.read_csv('heart.csv')
data.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

✓ Observation:

Here we see the first five rows of the dataset. Each row represents a patient, and the columns represent various factors like age, sex, chest pain type, cholesterol level, etc. The last column (target) indicates whether the patient has heart disease or not.


Before building the Bayesian Network, we must ensure that the data is suitable. This involves:

- Checking for missing values
- Discretizing continuous columns like age and cholesterol

```
# Check for missing values
print(f"Missing Values: {data.isnull().sum()}")

# Discretize age and cholesterol (as an example, more features can be discretized if needed)
data['age'] = pd.cut(data['age'], bins=3, labels=['Young', 'Middle', 'Old'])
data['chol'] = pd.cut(data['chol'], bins=3, labels=['Low', 'Normal', 'High'])

data.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	Middle	1	0	125	Low	0	1	168	0	1.0	2	2	3	0
1	Middle	1	0	140	Low	1	0	155	1	3.1	0	0	3	0
2	Old	1	0	145	Low	0	1	125	1	2.6	0	0	3	0
3	Middle	1	0	148	Low	0	1	161	0	0.0	2	1	3	0
4	Old	0	0	138	Normal	1	1	106	0	1.9	1	3	2	0

✓ Observation:


- If there were any missing values, we would handle them. In this case, we may not have any missing values.
- We have successfully discretized age and cholesterol into three categories: Young, Middle, Old for age, and Low, Normal, High for cholesterol.

In this step, we build the Bayesian Network structure using the correct column names:

- The target column is the outcome for heart disease (0 for no disease, 1 for disease).
- We create dependencies between features that influence heart disease, such as age, cholesterol, chest pain type, and heart rate.

```
# Defining the structure of the Bayesian Network using correct column names
model = BayesianNetwork([('age', 'target'),
                        ('chol', 'target'),
                        ('cp', 'target'), # cp: chest pain type
                        ('target', 'thalach')]) # thalach: maximum heart rate achieved
```

```
# Printing the model structure to verify the correct edges
model.edges()
```

```
 OutEdgeView([('age', 'target'), ('target', 'thalach'), ('chol', 'target'), ('cp', 'target')])
```

✓ Observation:

The structure of the Bayesian Network has been defined. The model has the following edges:

- Age, cholesterol, and chest pain type (cp) affect heart disease.
- Heart disease affects the maximum heart rate achieved (thalach).

We will now fit the model using the heart disease dataset to estimate the CPDs. We'll use `MaximumLikelihoodEstimator` to learn the parameters from the data.

```
# Fitting the model using Maximum Likelihood Estimation
model.fit(data, estimator=MaximumLikelihoodEstimator)
```

✓ Observation:

The Bayesian Network has been successfully trained, and the CPDs for each node have been estimated using MLE.

In this step, we will use the `VariableElimination` algorithm to infer the probability of heart disease based on given patient information. For example, we will infer the probability of a patient having heart disease when the age is "Old" and cholesterol is "High".

```
# Creating an inference object
infer = VariableElimination(model)

# Performing inference with given evidence
result = infer.query(variables=['target'], evidence={'age': 'Old', 'chol': 'High'})
print(result)
```

```
➡ +-----+-----+
  | target |   phi(target) |
  +-----+-----+
  | target(0) |      0.3615 |
  +-----+-----+
  | target(1) |      0.6385 |
  +-----+-----+
```

✓ Observation:

The output shows the probability distribution of the `target` (heart disease) variable given the evidence (age: Old, cholesterol: High). We can interpret the probability values to assess the likelihood of heart disease for this patient.

We will now test the model by adding more pieces of evidence (age, cholesterol, and chest pain type) to refine our prediction of heart disease probability.

```
# Inference with multiple pieces of evidence
result = infer.query(variables=['target'], evidence={'age': 'Middle', 'chol': 'Normal', 'cp': 2})
print(result)
```

```
➡ +-----+-----+
  | target |   phi(target) |
  +-----+-----+
  | target(0) |      0.0000 |
  +-----+-----+
  | target(1) |      1.0000 |
  +-----+-----+
```

✓ Observation:

The output shows the refined probability of having heart disease when multiple conditions are provided (age: Middle, cholesterol: Normal, chest pain type: 2). As we add more evidence, the model provides a more specific prediction.

In this notebook, we successfully constructed a Bayesian Network to model heart disease diagnosis using the UCI Heart Disease dataset. We followed the steps of loading the data, discretizing variables, building the network, learning parameters, and performing inference using the

trained model. The Bayesian Network provides a probabilistic way to diagnose heart disease based on multiple patient factors.