Experiment 7

Write a program to construct aBayesian 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.

```
!pip install pgmpy
```

Show hidden output

+ Code + Text

```
!pip install --upgrade pgmpy
```

Show hidden output

```
!pip install --upgrade pgmpy  # Ensure pgmpy is up-to-date
from pgmpy.models import DiscreteBayesianNetwork # Import DiscreteBayesianNetwork instead of BayesianNetwork or Bayesian

cancer_model = DiscreteBayesianNetwork([('Pollution', 'Cancer'),
                              ('Smoker', 'Cancer'),
                              ('Cancer', 'Xray'),
                              ('Cancer', 'Dyspnoea')])
print(cancer_model)
```

Show hidden output

```
print(cancer_model)
```

```
DiscreteBayesianNetwork with 5 nodes and 4 edges
```

```
cancer_model.nodes()   #to print nodes
```

```
NodeView(('Pollution', 'Cancer', 'Smoker', 'Xray', 'Dyspnoea'))
```

```
cancer_model.edges() # to print edges
```

```
OutEdgeView([('Pollution', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea'), ('Smoker', 'Cancer')])
```

Conditional Probability Distribution

```
cancer_model.get_cpds()   # to show conditional probability Distribution
```

```
[]
```

Creation of Conditional Probability Table

```
from pgmpy.factors.discrete import TabularCPD
```

```
cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
                      values=[[0.9],[0.1]])
cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
                      values=[[0.3],[0.7]])
cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
                      values=[[0.03, 0.05, 0.001, 0.02],
                      [0.97, 0.95, 0.999, 0.98]],
                      evidence=['Smoker', 'Pollution'],
                      evidence_card=[2, 2])
cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                      values=[[0.9, 0.2],[0.1, 0.8]],
                      evidence=['Cancer'], evidence_card=[2])
cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                      values=[[0.65, 0.3],[0.35, 0.7]],
                      evidence=['Cancer'], evidence_card=[2])
```

Double-click (or enter) to edit

```
# associating the parameters with the model structure
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

#checking if the cpds are valid for the model
```

```
cancer_model.get_cpds()
cancer_model.check_model()
```

```
True
```

```
#If you want to stick with DiscreteBayesianNetwork and identify active trails.
#This approach identifies all nodes on the active trail, including the start and end nodes.
active_trail = cancer_model.active_trail_nodes('Pollution', observed=['Cancer'])
# Check if 'Smoker' is in the active trail
is_active = 'Smoker' in active_trail['Pollution']
print(f"Is there an active trail between Pollution and Smoker given Cancer? {is_active}")
```

```
Is there an active trail between Pollution and Smoker given Cancer? True
```

```
cancer_model.local_independencies('Xray')     #Xray and Dyspnoea are Independent
```

```
(Xray ⊥ Dyspnoea, Smoker, Pollution | Cancer)
```

```
cancer_model.local_independencies('Pollution')
```

```
(Pollution ⊥ Smoker)
```

```
cancer_model.local_independencies('Smoker')
```

```
(Smoker ⊥ Pollution)
```

```
cancer_model.local_independencies('Dyspnoea')
```

```
(Dyspnoea ⊥ Smoker, Pollution, Xray | Cancer)
```

```
cancer_model.local_independencies('Cancer')
```

```
cancer_model.local_independencies('Dyspnoea')
```

```
(Dyspnoea ⊥ Smoker, Pollution, Xray | Cancer)
```

```
cancer_model.local_independencies('Pollution')
```

```
(Pollution ⊥ Smoker)
```

```
cancer_model.get_independencies()
```

```
(Pollution ⊥ Dyspnoea | Cancer)
(Smoker ⊥ Dyspnoea | Cancer)
(Pollution ⊥ Smoker)
(Dyspnoea ⊥ Xray | Cancer)
(Smoker ⊥ Xray | Cancer)
(Pollution ⊥ Xray | Cancer)
```

```
cancer_model.get_cpds()
print(cancer_model.get_cpds('Pollution'))
```

```
+--------------+-----+
| Pollution(0) | 0.9 |
+--------------+-----+
| Pollution(1) | 0.1 |
+--------------+-----+
```

```
cancer_model.get_cpds()
print(cancer_model.get_cpds('Cancer'))
```

```
+-----------+--------------+--------------+--------------+--------------+
| Smoker    | Smoker(0)    | Smoker(0)    | Smoker(1)    | Smoker(1)    |
+-----------+--------------+--------------+--------------+--------------+
| Pollution | Pollution(0) | Pollution(1) | Pollution(0) | Pollution(1) |
+-----------+--------------+--------------+--------------+--------------+
| Cancer(0) | 0.03         | 0.05         | 0.001        | 0.02         |
+-----------+--------------+--------------+--------------+--------------+
| Cancer(1) | 0.97         | 0.95         | 0.999        | 0.98         |
+-----------+--------------+--------------+--------------+--------------+
```

```
cancer_model.get_cpds()
#conditional probabilities
```

```
[<TabularCPD representing P(Pollution:2) at 0x797c11002310>,
 <TabularCPD representing P(Smoker:2) at 0x797c102a7190>,
 <TabularCPD representing P(Cancer:2 | Smoker:2, Pollution:2) at 0x797c110197d0>,
 <TabularCPD representing P(Xray:2 | Cancer:2) at 0x797c11018110>,
 <TabularCPD representing P(Dyspnoea:2 | Cancer:2) at 0x797c1101bf90>]
```

Inferencing with Bayesian Network with Variable Elimination

```
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(cancer_model)

q=cancer_infer.query(variables=['Cancer'], evidence={'Smoker': 1})
print(q)
```

```
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0029 |
+-----------+---------------+
| Cancer(1) |        0.9971 |
+-----------+---------------+
```

```
r=cancer_infer.query(variables=['Cancer'], evidence={'Smoker': 1, 'Pollution':1})
print(r)
```

```
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0200 |
+-----------+---------------+
| Cancer(1) |        0.9800 |
+-----------+---------------+
```

```
s=cancer_infer.query(variables=['Cancer'], evidence={'Pollution':1})
print(r)
```

```
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0200 |
+-----------+---------------+
| Cancer(1) |        0.9800 |
+-----------+---------------+
```

```
r=cancer_infer.query(variables=['Smoker'], evidence={'Cancer': 1})
print(r)
```

```
+-----------+---------------+
| Smoker    |   phi(Smoker) |
+===========+===============+
| Smoker(0) |        0.2938 |
+-----------+---------------+
| Smoker(1) |        0.7062 |
+-----------+---------------+
```

Start coding or generate with AI.