Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #31

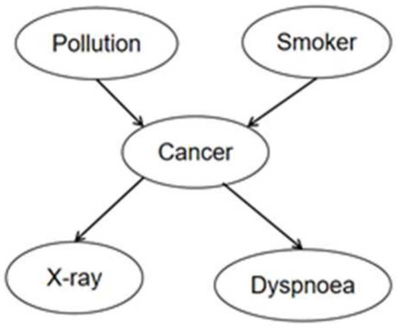
Artificial Intelligence for Data Science

Week-9:
Introduction to Probabilistic Reasoning [Part-III]
Exploring Probabilistic Reasoning using pgmpy

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Define the network structure

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

# Define the network structure.
cancer_model = DiscreteBayesianNetwork(
    [
        ("Pollution", "Cancer"),
        ("Smoker", "Cancer"),
        ("Cancer", "Xray"),
        ("Cancer", "Dyspnoea"),
    ]
)
```

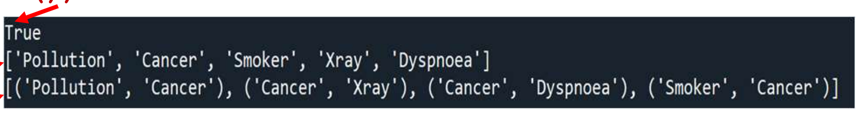


Define the Conditional Probability Distributions

```
cpd_poll = TabularCPD(variable="Pollution",variable_card=2,values=[[0.99], [0.01]])
cpd_smoke = TabularCPD(variable="Smoker", variable_card=2, values=[[0.3], [0.7]])

cpd_cancer = TabularCPD(
    variable="Cancer",
    variable_card=2,
    values=[[0.97, 0.05, 0.001, 0.02], [0.03, 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],
)
```

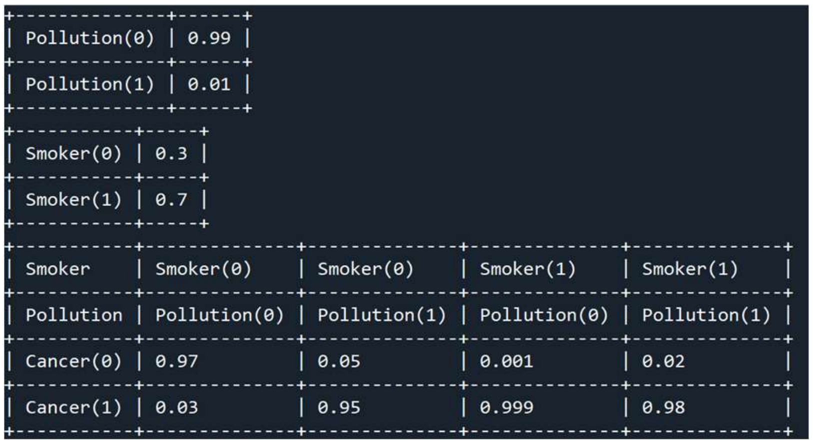
Defining Conditional Probability Distributions (CPDs)

```
cpd_dysp = TabularCPD(
    variable="Dyspnoea",
    variable_card=2,
    values=[[0.65, 0.3], [0.35, 0.7]],
    evidence=["Cancer"],
    evidence_card=[2],
)
# Add the CPDs to the model.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Check if the model is correctly defined.
print(cancer_model.check_model())

#View nodes, edges, cpds
print(cancer_model.nodes)
print(cancer_model.edges)
for cpd in cancer_model.get_cpds():
    print(cpd)
```

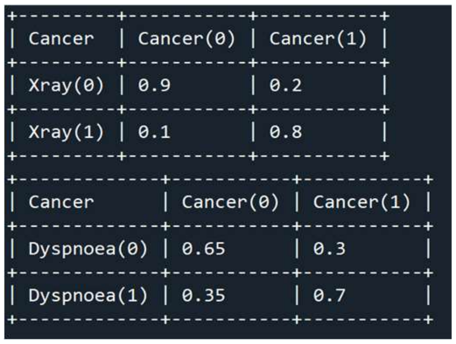


```
True
['Pollution', 'Cancer', 'Smoker', 'Xray', 'Dyspnoea']
[('Pollution', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea'), ('Smoker', 'Cancer')]
```

Printing CPDs



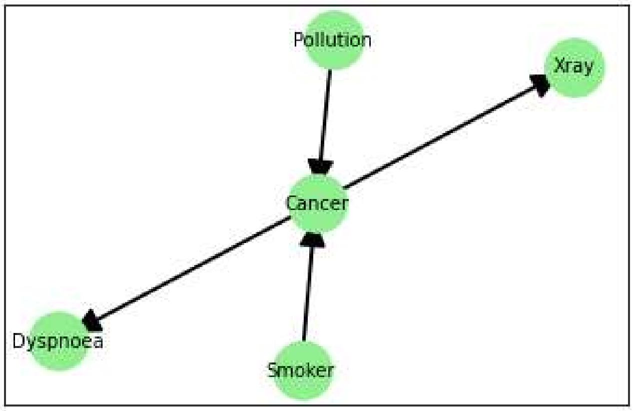
```
+--------------+------+
| Pollution(0) | 0.99 |
+--------------+------+
| Pollution(1) | 0.01 |
+--------------+------+
+-----------+-----+
| Smoker(0) | 0.3 |
+-----------+-----+
| Smoker(1) | 0.7 |
+-----------+-----+
+-----------+--------------+--------------+--------------+--------------+
| Smoker    | Smoker(0)    | Smoker(0)    | Smoker(1)    | Smoker(1)    |
+-----------+--------------+--------------+--------------+--------------+
| Pollution | Pollution(0) | Pollution(1) | Pollution(0) | Pollution(1) |
+-----------+--------------+--------------+--------------+--------------+
| Cancer(0) | 0.97         | 0.05         | 0.001        | 0.02         |
+-----------+--------------+--------------+--------------+--------------+
| Cancer(1) | 0.03         | 0.95         | 0.999        | 0.98         |
+-----------+--------------+--------------+--------------+--------------+
```



```
+---------+-----------+-----------+
| Cancer  | Cancer(0) | Cancer(1) |
+---------+-----------+-----------+
| Xray(0) | 0.9       | 0.2       |
+---------+-----------+-----------+
| Xray(1) | 0.1       | 0.8       |
+---------+-----------+-----------+
+-------------+-----------+-----------+
| Cancer      | Cancer(0) | Cancer(1) |
+-------------+-----------+-----------+
| Dyspnoea(0) | 0.65      | 0.3       |
+-------------+-----------+-----------+
| Dyspnoea(1) | 0.35      | 0.7       |
+-------------+-----------+-----------+
```

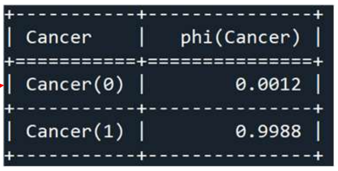
Visualizing Bayesian Network (BN)

```
import networkx as nx
import matplotlib.pyplot as plt




G=nx.DiGraph()
G.add_nodes_from(cancer_model.nodes)
G.add_edges_from(cancer_model.edges)
pos=nx.spring_layout(G,seed=888)

nx.draw_networkx_nodes(G, pos,node_size=1000,node_color='lightgreen')
nx.draw_networkx_labels(G, pos,font_size=10)
nx.draw_networkx_edges(G, pos, width=2,arrowsize=30)
plt.show()
```

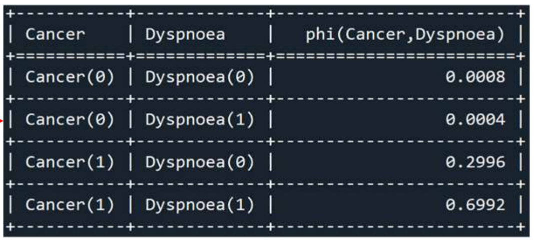


Inference Generation in Discrete BN

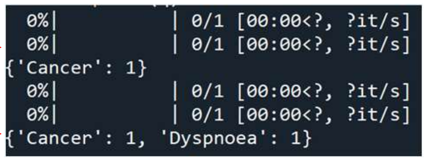
```
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(cancer_model)

# Computing the probability of Cancer given Smoker=1.
q = cancer_infer.query(variables=["Cancer"],
evidence={"Smoker": 1})
print(q)
# Computing the joint probability of Cancer and Dyspnoea
given Smoker=1
q = cancer_infer.query(variables=["Cancer", "Dyspnoea"],
evidence={"Smoker": 1})
print(q)
# Computing the MAP of Cancer given Smoker=1.
q = cancer_infer.map_query(variables=["Cancer"],
evidence={"Smoker": 1})
print(q)
# Computing the MAP of Cancer and Dyspnoea given Smoker=1
q = cancer_infer.map_query(variables=["Cancer", "Dyspnoea"],
evidence={"Smoker": 1})
print(q)
```



```
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0012 |
+-----------+---------------+
| Cancer(1) |        0.9988 |
+-----------+---------------+
```



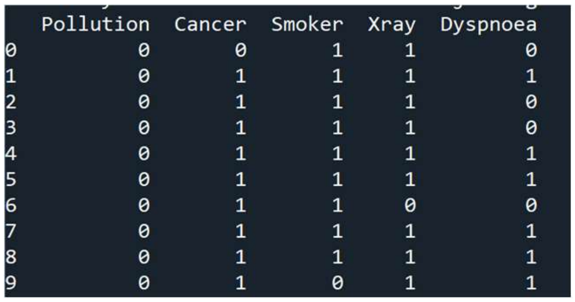
```
+-----------+-------------+------------------------+
| Cancer    | Dyspnoea    |   phi(Cancer,Dyspnoea) |
+===========+=============+========================+
| Cancer(0) | Dyspnoea(0) |                 0.0008 |
+-----------+-------------+------------------------+
| Cancer(0) | Dyspnoea(1) |                 0.0004 |
+-----------+-------------+------------------------+
| Cancer(1) | Dyspnoea(0) |                 0.2996 |
+-----------+-------------+------------------------+
| Cancer(1) | Dyspnoea(1) |                 0.6992 |
+-----------+-------------+------------------------+
```



```
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
{'Cancer': 1}
  0%|          | 0/1 [00:00<?, ?it/s]
  0%|          | 0/1 [00:00<?, ?it/s]
{'Cancer': 1, 'Dyspnoea': 1}
```

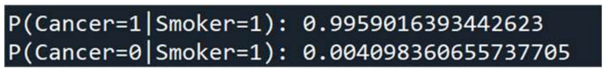
Inference Generation in Discrete BN: MCMC

```
from pgmpy.sampling import GibbsSampling

# Initialize GibbsSampling with the model
gibbs_sampler = GibbsSampling(cancer_model)

samples = gibbs_sampler.sample(size=1000)
print(samples.head(10))
```

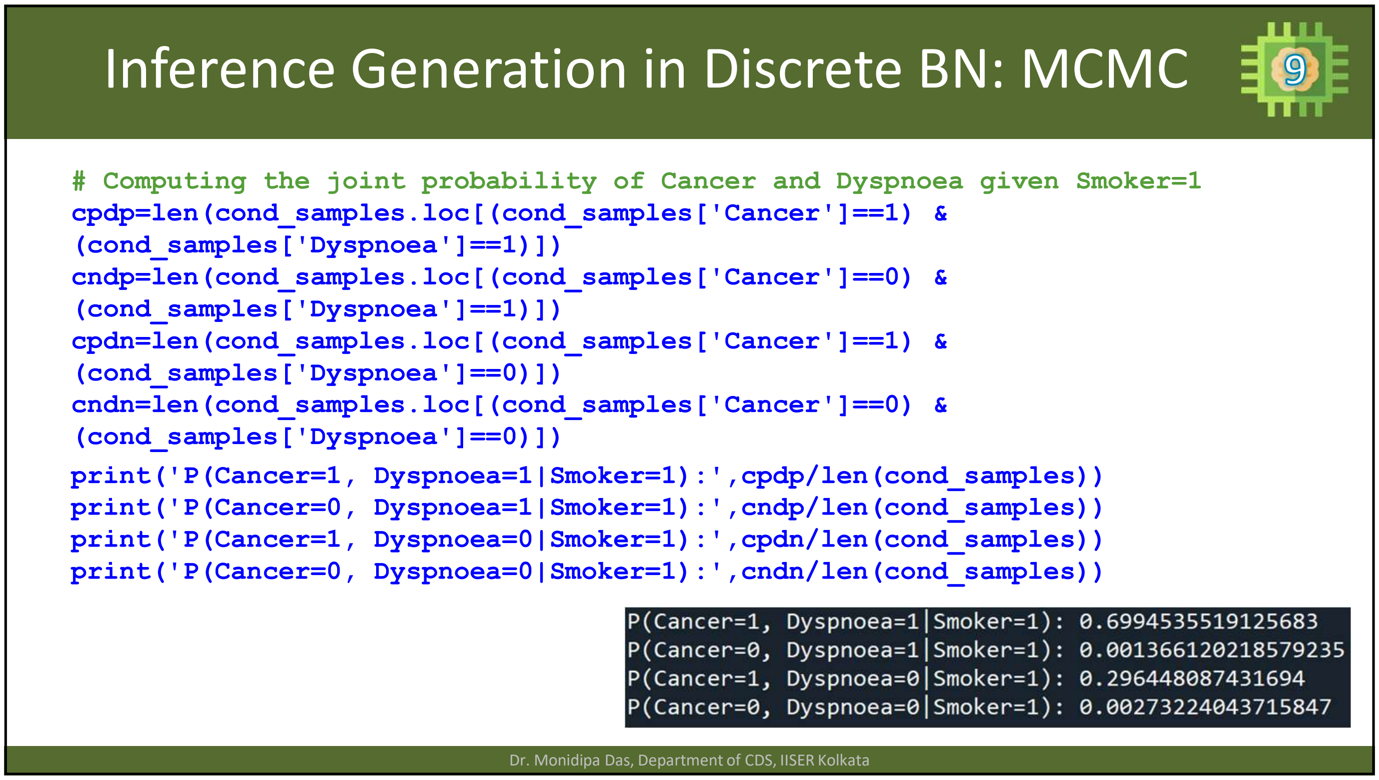


```
   Pollution  Cancer  Smoker  Xray  Dyspnoea
0          0       0       1     1         0
1          0       1       1     1         1
2          0       1       1     1         0
3          0       1       1     1         0
4          0       1       1     1         1
5          0       1       1     1         1
6          0       1       1     0         0
7          0       1       1     1         1
8          0       1       1     1         1
9          0       1       0     1         1
```

```
# Computing the probability of Cancer given Smoker=1.
cond_samples=samples.loc[samples['Smoker']==1]
cp=len(cond_samples.loc[cond_samples['Cancer']==1])
cn=len(cond_samples.loc[cond_samples['Cancer']==0])
print('P(Cancer=1|Smoker=1):',cp/len(cond_samples))
print('P(Cancer=0|Smoker=1):',cn/len(cond_samples))
```



```
P(Cancer=1|Smoker=1): 0.9959016393442623
P(Cancer=0|Smoker=1): 0.004098360655737705
```

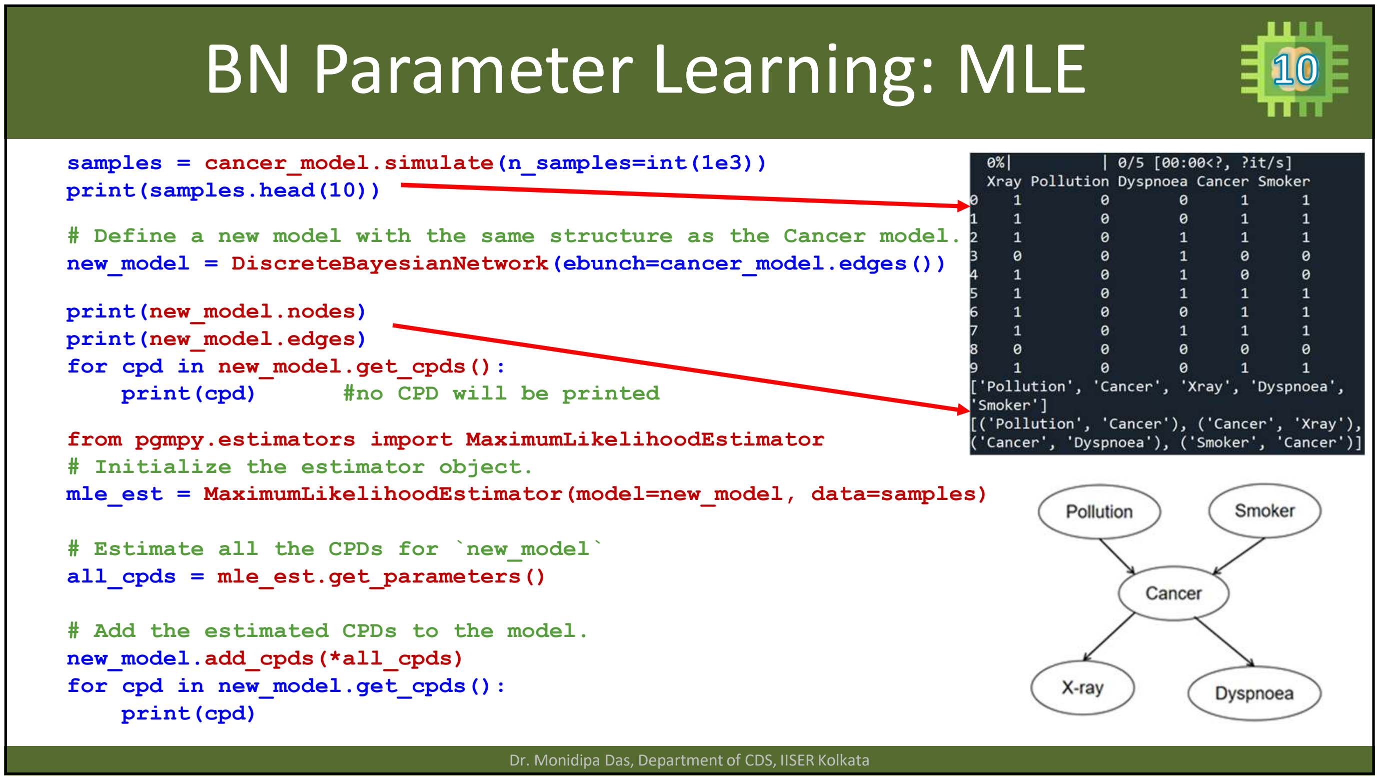


Inference Generation in Discrete BN: MCMC

9

```
# Computing the joint probability of Cancer and Dyspnoea given Smoker=1
cpdp=len(cond_samples.loc[(cond_samples['Cancer']==1) &
(cond_samples['Dyspnoea']==1)])
cndp=len(cond_samples.loc[(cond_samples['Cancer']==0) &
(cond_samples['Dyspnoea']==1)])
cpdn=len(cond_samples.loc[(cond_samples['Cancer']==1) &
(cond_samples['Dyspnoea']==0)])
cndn=len(cond_samples.loc[(cond_samples['Cancer']==0) &
(cond_samples['Dyspnoea']==0)])
print('P(Cancer=1, Dyspnoea=1|Smoker=1):',cpdp/len(cond_samples))
print('P(Cancer=0, Dyspnoea=1|Smoker=1):',cndp/len(cond_samples))
print('P(Cancer=1, Dyspnoea=0|Smoker=1):',cpdn/len(cond_samples))
print('P(Cancer=0, Dyspnoea=0|Smoker=1):',cndn/len(cond_samples))
```

```
P(Cancer=1, Dyspnoea=1|Smoker=1): 0.6994535519125683
P(Cancer=0, Dyspnoea=1|Smoker=1): 0.001366120218579235
P(Cancer=1, Dyspnoea=0|Smoker=1): 0.296448087431694
P(Cancer=0, Dyspnoea=0|Smoker=1): 0.00273224043715847
```

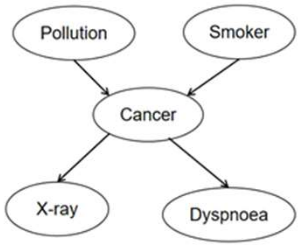


BN Parameter Learning: MLE

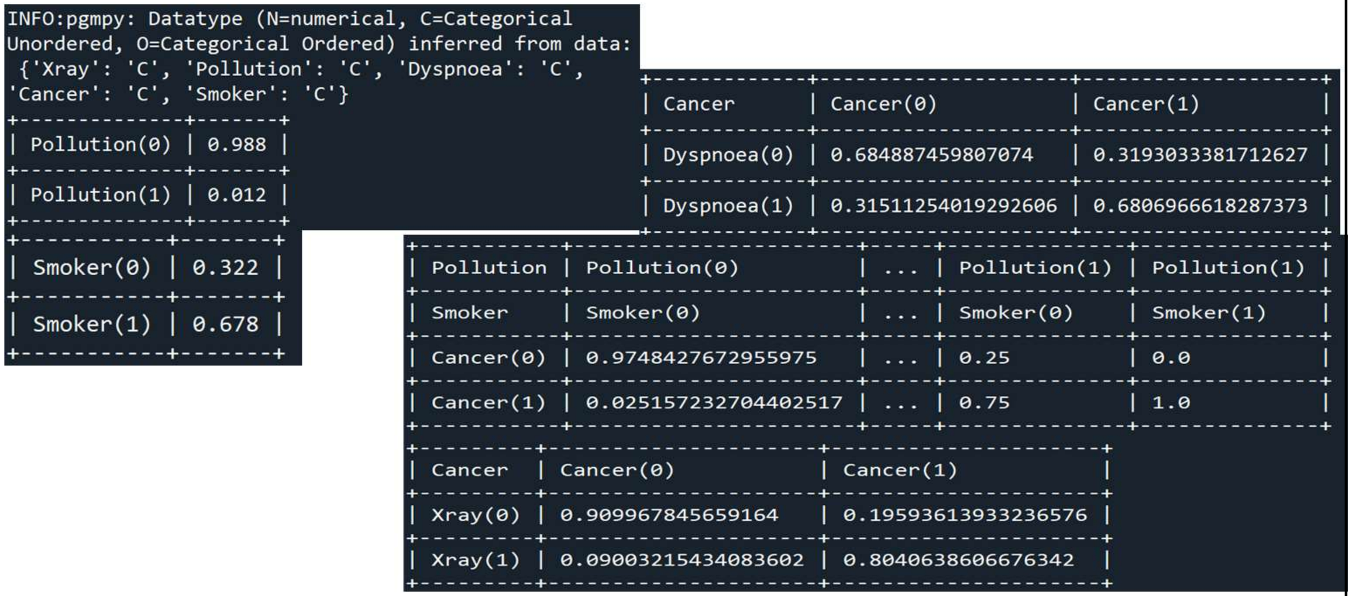
10

```
samples = cancer_model.simulate(n_samples=int(1e3))
print(samples.head(10))

# Define a new model with the same structure as the Cancer model.
new_model = DiscreteBayesianNetwork(ebunch=cancer_model.edges())

print(new_model.nodes)
print(new_model.edges)
for cpd in new_model.get_cpds():
    print(cpd)      #no CPD will be printed

from pgmpy.estimators import MaximumLikelihoodEstimator
# Initialize the estimator object.
mle_est = MaximumLikelihoodEstimator(model=new_model, data=samples)

# Estimate all the CPDs for `new_model`
all_cpds = mle_est.get_parameters()

# Add the estimated CPDs to the model.
new_model.add_cpds(*all_cpds)
for cpd in new_model.get_cpds():
    print(cpd)
```

```
  0%|          | 0/5 [00:00<?, ?it/s]
  Xray Pollution Dyspnoea Cancer Smoker
0    1         0        0      1      1
1    1         0        0      1      1
2    1         0        1      1      1
3    0         0        1      0      0
4    1         0        1      0      0
5    1         0        1      1      1
6    1         0        0      1      1
7    1         0        1      1      1
8    0         0        0      0      0
9    1         0        0      1      1
['Pollution', 'Cancer', 'Xray', 'Dyspnoea',
'Smoker']
[('Pollution', 'Cancer'), ('Cancer', 'Xray'),
('Cancer', 'Dyspnoea'), ('Smoker', 'Cancer')]
```



BN Parameter Learning: MLE



```
INFO:pgmpy: Datatype (N=numerical, C=Categorical
Unordered, O=Categorical Ordered) inferred from data:
 {'Xray': 'C', 'Pollution': 'C', 'Dyspnoea': 'C',
'Cancer': 'C', 'Smoker': 'C'}
+--------------+-------+
| Pollution(0) | 0.988 |
+--------------+-------+
| Pollution(1) | 0.012 |
+--------------+-------+
+-----------+-------+
| Smoker(0) | 0.322 |
+-----------+-------+
| Smoker(1) | 0.678 |
+-----------+-------+
```

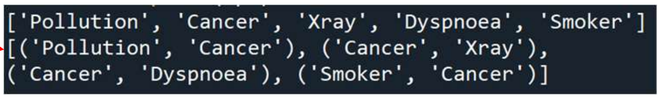
```
+-------------+---------------------+---------------------+
| Cancer      | Cancer(0)           | Cancer(1)           |
+-------------+---------------------+---------------------+
| Dyspnoea(0) | 0.684887459807074   | 0.3193033381712627  |
+-------------+---------------------+---------------------+
| Dyspnoea(1) | 0.31511254019292606 | 0.6806966618287373  |
+-------------+---------------------+---------------------+
```

```
+-----------+----------------------+-----+--------------+--------------+
| Pollution | Pollution(0)         | ... | Pollution(1) | Pollution(1) |
+-----------+----------------------+-----+--------------+--------------+
| Smoker    | Smoker(0)            | ... | Smoker(0)    | Smoker(1)    |
+-----------+----------------------+-----+--------------+--------------+
| Cancer(0) | 0.9748427672955975   | ... | 0.25         | 0.0          |
+-----------+----------------------+-----+--------------+--------------+
| Cancer(1) | 0.025157232704402517 | ... | 0.75         | 1.0          |
+-----------+----------------------+-----+--------------+--------------+
```

```
+---------+---------------------+---------------------+
| Cancer  | Cancer(0)           | Cancer(1)           |
+---------+---------------------+---------------------+
| Xray(0) | 0.909967845659164   | 0.19593613933236576 |
+---------+---------------------+---------------------+
| Xray(1) | 0.09003215434083602 | 0.8040638606676342  |
+---------+---------------------+---------------------+
```

BN Parameter Learning: Bayesian Estimator

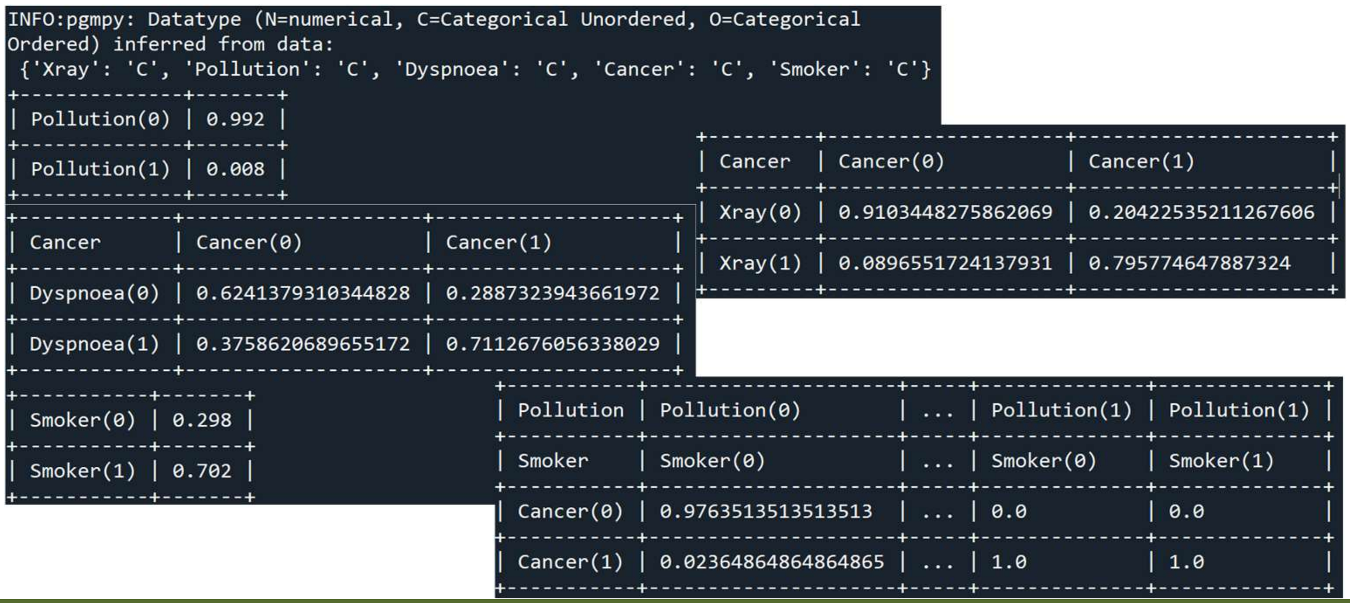
```
# Define a new model with the same structure as the Cancer model.
new_model = DiscreteBayesianNetwork(ebunch=cancer_model.edges())

print(new_model.nodes)
print(new_model.edges)
for cpd in new_model.get_cpds():
    print(cpd) #no CPD will be printed

# Initialize the Bayesian Estimator
from pgmpy.estimators import BayesianEstimator
be_est = BayesianEstimator(model=new_model, data=samples)

# Estimate all the CPDs for `new_model`
all_cpds = mle_est.get_parameters()

# Add the estimated CPDs to the model.
new_model.add_cpds(*all_cpds)
for cpd in new_model.get_cpds():
    print(cpd)
```



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

BN Parameter Learning: Bayesian Estimator

13



```
INFO:pgmpy: Datatype (N=numerical, C=Categorical Unordered, O=Categorical
Ordered) inferred from data:
 {'Xray': 'C', 'Pollution': 'C', 'Dyspnoea': 'C', 'Cancer': 'C', 'Smoker': 'C'}
+--------------+-------+
| Pollution(0) | 0.992 |
+--------------+-------+
| Pollution(1) | 0.008 |
+--------------+-------+
```

```
+---------+--------------------+---------------------+
| Cancer  | Cancer(0)          | Cancer(1)           |
+---------+--------------------+---------------------+
| Xray(0) | 0.9103448275862069 | 0.20422535211267606 |
+---------+--------------------+---------------------+
| Xray(1) | 0.0896551724137931 | 0.795774647887324   |
+---------+--------------------+---------------------+
```

```
+-------------+--------------------+--------------------+
| Cancer      | Cancer(0)          | Cancer(1)          |
+-------------+--------------------+--------------------+
| Dyspnoea(0) | 0.6241379310344828 | 0.2887323943661972 |
+-------------+--------------------+--------------------+
| Dyspnoea(1) | 0.3758620689655172 | 0.7112676056338029 |
+-------------+--------------------+--------------------+
```

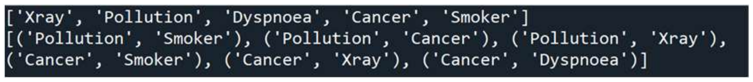
```
+-----------+-------+
| Smoker(0) | 0.298 |
+-----------+-------+
| Smoker(1) | 0.702 |
+-----------+-------+
```

```
+-----------+----------------------+-----+--------------+--------------+
| Pollution | Pollution(0)         | ... | Pollution(1) | Pollution(1) |
+-----------+----------------------+-----+--------------+--------------+
| Smoker    | Smoker(0)            | ... | Smoker(0)    | Smoker(1)    |
+-----------+----------------------+-----+--------------+--------------+
| Cancer(0) | 0.9763513513513513   | ... | 0.0          | 0.0          |
+-----------+----------------------+-----+--------------+--------------+
| Cancer(1) | 0.02364864864864865  | ... | 1.0          | 1.0          |
+-----------+----------------------+-----+--------------+--------------+
```

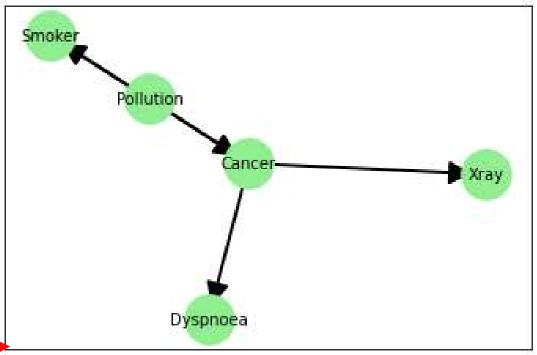
BN Structure Learning using Hill Climbing

14

```
# Learning the discrete variable Cancer model back
from pgmpy.estimators import HillClimbSearch
samples = cancer_model.simulate(n_samples=int(1e3))

est = HillClimbSearch(data=samples)
estimated_model =
DiscreteBayesianNetwork(est.estimate(scoring_method="k2",
max_indegree=4, max_iter=int(1e4)))

print(estimated_model.nodes)
print(estimated_model.edges)

G=nx.DiGraph()
G.add_nodes_from(estimated_model.nodes)
G.add_edges_from(estimated_model.edges)
pos=nx.spring_layout(G,seed=888)
nx.draw_networkx_nodes(G,
pos,node_size=1000,node_color='lightgreen')
nx.draw_networkx_labels(G, pos,font_size=10)
nx.draw_networkx_edges(G, pos, width=2,arrowsize=30)
plt.show()
```

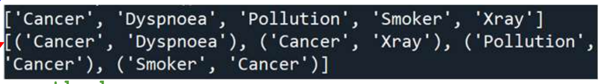


```
['Xray', 'Pollution', 'Dyspnoea', 'Cancer', 'Smoker']
[('Pollution', 'Smoker'), ('Pollution', 'Cancer'), ('Pollution', 'Xray'),
('Cancer', 'Smoker'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea')]
```

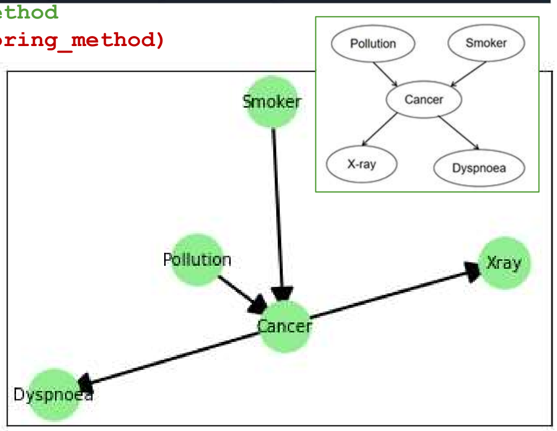


BN Structure Learning using Exhaustive Search

```
from pgmpy.estimators import ExhaustiveSearch
from pgmpy.estimators import BDeu
samples = cancer_model.simulate(n_samples=int(1e3))
# Initialize the BDeuScore scoring method
scoring_method = BDeu(data=samples)
# Initialize ExhaustiveSearch with data and scoring method
est = ExhaustiveSearch(data=samples,scoring_method=scoring_method)
# Estimate the best structure
best_model = DiscreteBayesianNetwork(est.estimate())
print(best_model.nodes)
print(best_model.edges)
G=nx.DiGraph()
G.add_nodes_from(best_model.nodes)
G.add_edges_from(best_model.edges)
pos=nx.spring_layout(G,seed=888)
nx.draw_networkx_nodes(G,
pos,node_size=1000,node_color='lightgreen')
nx.draw_networkx_labels(G, pos,font_size=10)
nx.draw_networkx_edges(G, pos, width=2,arrowsize=30)
plt.show()
```



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

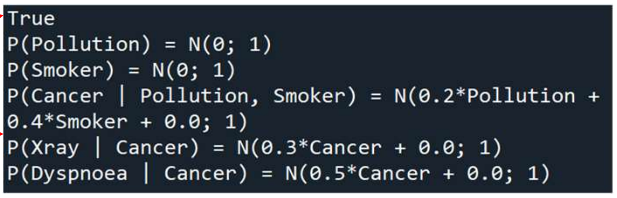


Linear Gaussian Bayesian Network

```
from pgmpy.models import LinearGaussianBayesianNetwork
from pgmpy.factors.continuous import LinearGaussianCPD

# Define the network structure.
cancer_modelc = LinearGaussianBayesianNetwork(
    [
        ("Pollution", "Cancer"),
        ("Smoker", "Cancer"),
        ("Cancer", "Xray"),
        ("Cancer", "Dyspnoea"),
    ]
)

# Define the CPDs.
cpd_poll = LinearGaussianCPD(variable="Pollution", beta=[0], std=1, evidence=[])
cpd_smoke = LinearGaussianCPD(variable="Smoker", beta=[0], std=1, evidence=[])
cpd_cancer = LinearGaussianCPD(
    variable="Cancer",
    beta=[0, 0.2, 0.4],
    std=1,
    evidence=["Pollution", "Smoker"],
)
```

Linear Gaussian Bayesian Network

```
cpd_xray = LinearGaussianCPD(
    variable="Xray",
    beta=[0, 0.3],
    std=1,
    evidence=["Cancer"],
)
cpd_dysp = LinearGaussianCPD(
    variable="Dyspnoea",
    beta=[0, 0.5],
    std=1,
    evidence=["Cancer"],
)

# Add the CPDs to the model.
cancer_modelc.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Check if the model is correctly defined.
print(cancer_modelc.check_model())

for cpd in cancer_modelc.cpds:
    print(cpd)
```



```
True
P(Pollution) = N(0; 1)
P(Smoker) = N(0; 1)
P(Cancer | Pollution, Smoker) = N(0.2*Pollution +
0.4*Smoker + 0.0; 1)
P(Xray | Cancer) = N(0.3*Cancer + 0.0; 1)
P(Dyspnoea | Cancer) = N(0.5*Cancer + 0.0; 1)
```

Questions?