Vivekanand Education Society's
Institute of Technology
(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)
Department of Information Technology

AIDS - 2 Lab
Experiment - 01

Aim: To Implement inferencing with Bayesian Network in python.

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D20B
Subject	AIDS - 2
Grade:	

EXPERIMENT - 01

AIM: To Implement inferencing with Bayesian Network in python.

DATASET:
https://towardsdatascience.com/a-beginners-guide-to-kaggle-s-titanic-problem-3193cb56f6ca

THEORY:

A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable. The Bayesian network consists of two major parts: a directed acyclic graph and a set of conditional probability distributions The directed acyclic graph is a set of random variables represented by nodes. The conditional probability distribution of a node (random variable) is defined for every possible outcome of the preceding causal node(s).

Inference is defined as the process of deriving logical conclusions based on premises known or assumed to be true. One strength of Bayesian networks is the ability for inference, which in this case involves the probabilities of unobserved variables in the system. When observed variables are known to be in one state, probabilities of other variables will have different values than the generic case. Let us take a simple example system, a television. The probability of a television being on while people are home is much higher than the probability of that television being on when no one is home.

Bayes factor:-

$$\boxed{BF = \frac{p(\text{model1} \mid \text{data})}{p(\text{model2} \mid \text{data})} = \frac{\frac{p(\text{data}|\text{model1})p(\text{model1})}{p(\text{data})}}{\frac{p(\text{data}|\text{model 2})p(\text{model 2})}{p(\text{data})}} = \frac{p(\text{data} \mid \text{model1})}{p(\text{data} \mid \text{model 2})}}$$

The advantages of Bayesian Networks are as follows:

- Bayesian Networks visually represent all the relationships between the variables in the system with connecting arcs.
- It is easy to recognize the dependence and independence between various nodes.
- Bayesian networks can handle situations where the data set is incomplete since the model accounts for dependencies between all variables.
- Bayesian networks can map scenarios where it is not feasible/practical to measure all variables due to system constraints (costs, not enough sensors, etc.)
- Help to model noisy systems.
- Can be used for any system model - from all known parameters to no known parameters.

The limitations of Bayesian Networks are as follows:

- All branches must be calculated in order to calculate the probability of any one branch.
- The quality of the results of the network depends on the quality of the prior beliefs or model. A variable is only a part of a Bayesian network if you believe that the system depends on it.
- Calculation of the network is NP-hard (nondeterministic polynomial-time hard), so it is very difficult and possibly costly.
- Calculations and probabilities using Bayes rule and marginalization can become complex and are often characterized by subtle wording, and care must be taken to calculate them properly.

IMPLEMENTATION:

1. Import Libraries
2. Reading the datasets
3. Dropping unused columns
4. HotEncoding the datasets (Train and Test)
5. Creating the Bayesian Network

```
dfnum_train
```

	Survived	Pclass	Sex	SibSp	Parch	Embarked
0	0	3	1	2	1	3
1	1	1	0	2	1	1
2	1	3	0	1	1	3
3	1	1	0	2	1	3
4	0	3	1	1	1	3
...	...	...	...	...	...	...
886	0	2	1	1	1	3
887	1	1	0	1	1	3
888	0	3	0	2	3	3
889	1	1	1	1	1	1
890	0	3	1	1	1	2

891 rows × 6 columns

6. Preparation for function for accuracy

```
def get_acc(model, df, col):
        # Get accuracy score by the model for the validation dataset df with target col
    pred = bn.predict(model, df, variables=[col])
```

```
        print(pred)
        acc = accuracy_score(df[col], pred[col])
        print('Accuracy -', acc)
        return acc
```

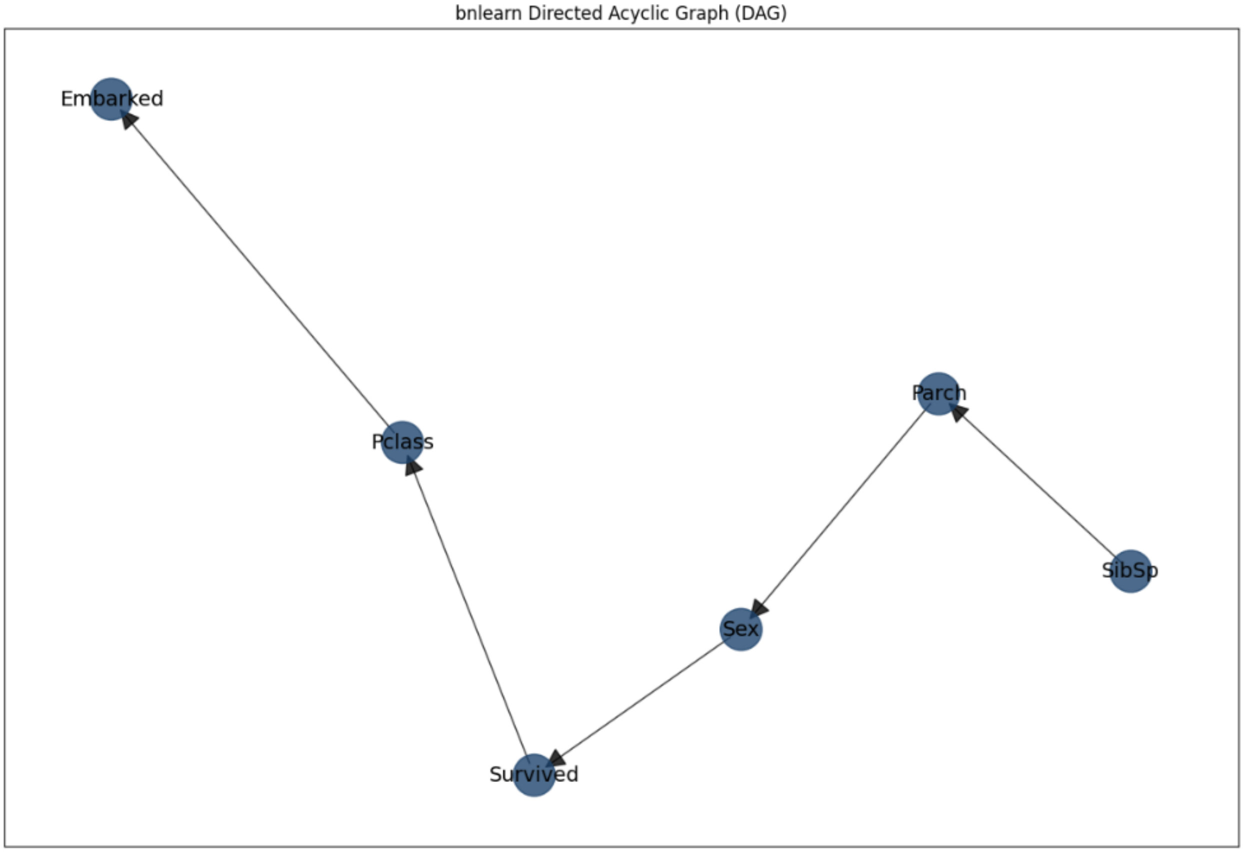
7. Creating the model based on decided parameters
    - Model 1: Graph and Accuracy

```
%%time
# Structure learning
DAG = bn.structure_learning.fit(dfnum, methodtype='hc', root_node='Survived', bw_list_method='nodes', verbose=3)

# Plot
G = bn.plot(DAG)

# Parameter learning
model = bn.parameter_learning.fit(DAG, dfnum, verbose=3);
```

bnlearn Directed Acyclic Graph (DAG)




```
# Get score of the model1
acc1 = get_acc(model, valid, 'Survived')
```

```
[bnlearn]> Remaining columns for inference: 5
100%|██████████| 59/59 [00:00<00:00, 147.90it/s]
     Survived         p
0           0  0.725084
1           0  0.725084
2           0  0.725084
3           1  0.662098
4           0  0.507407
..        ...       ...
174         0  0.507407
175         0  0.725084
176         1  0.662098
177         0  0.725084
178         0  0.725084

[179 rows x 2 columns]
Accuracy - 0.8156424581005587
```

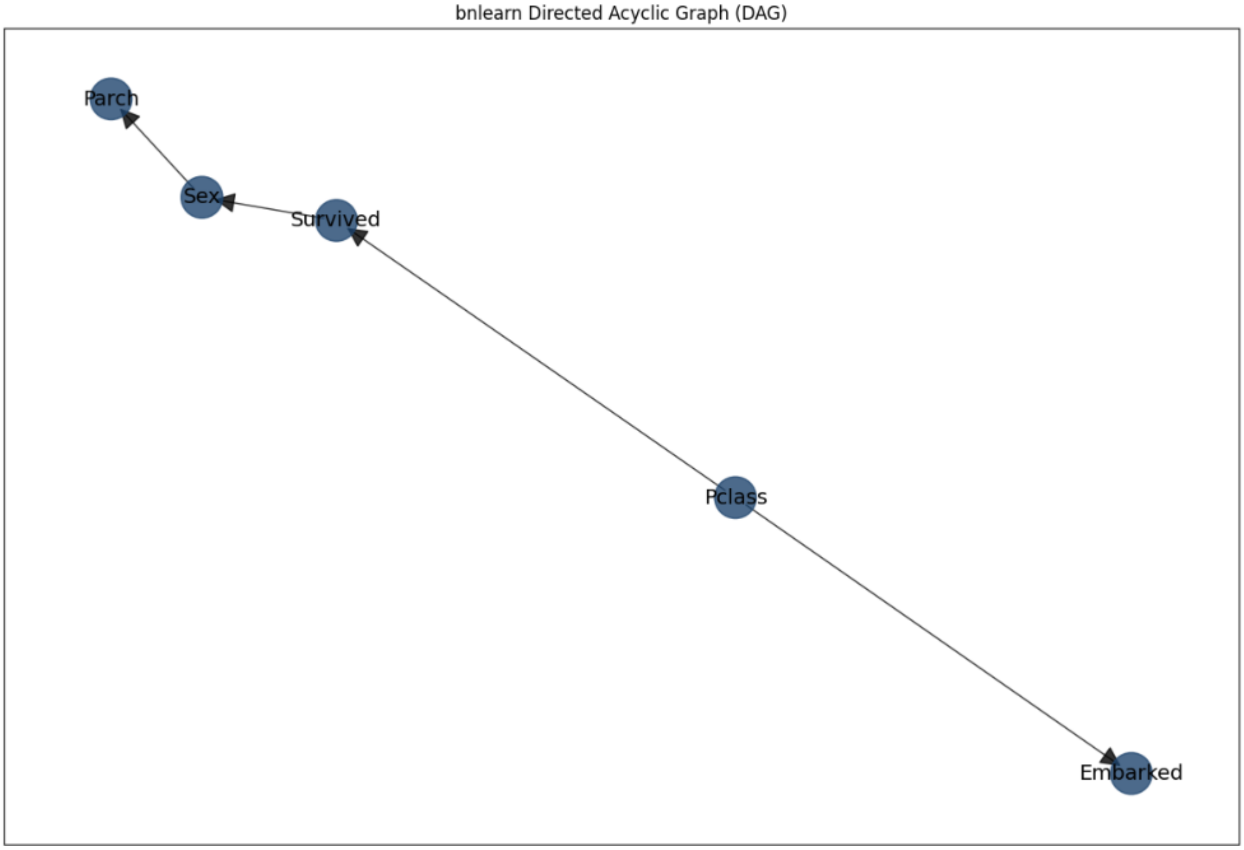
- Model 2: Graph and Accuracy

```
%%time
# Structure learning
DAG2 = bn.structure_learning.fit(dfnum, methodtype='hc', black_list=['SibSp'], root_node='Survived', bw_list_method='nodes', verbose=4)

# Plot
G2 = bn.plot(DAG2)

# Parameter learning
model2 = bn.parameter_learning.fit(DAG2, dfnum, verbose=4);
```

bnlearn Directed Acyclic Graph (DAG)



```
# Score of the model2
acc2 = get_acc(model, valid.drop(columns=['SibSp']), 'Survived')
```

```
[bnlearn]> Remaining columns for inference: 4
100%|██████████| 34/34 [00:00<00:00, 189.35it/s]     Survived         p
0           0  0.725084
1           0  0.725084
2           0  0.725084
3           1  0.662098
4           0  0.507407
..        ...       ...
174         0  0.507407
175         0  0.725084
176         1  0.662098
177         0  0.725084
178         0  0.725084

[179 rows x 2 columns]
Accuracy - 0.8156424581005587
```

8. Inference in Bayesian Network

Inference 1:-

- Target - Survived
- Dependency - Sex and Pclass

```
# Make inference
query = bn.inference.fit(model, variables=['Survived'], evidence={'Sex':True, 'Pclass':True})
print(query)
print(query.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+------------+----------+
|    |   Survived |        p |
+====+============+==========+
|  0 |          0 | 0.566487 |
+----+------------+----------+
|  1 |          1 | 0.433513 |
+----+------------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.5665 |
+-------------+-----------------+
| Survived(1) |          0.4335 |
+-------------+-----------------+
   Survived         p
0         0  0.566487
1         1  0.433513
```

Inference 2:-

- Target - survived
- Dependency - sex = 0 (female)

```
q1 = bn.inference.fit(model, variables=['Survived'], evidence={'Sex':0})
print(q1)
print(q1.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+------------+----------+
|    |   Survived |        p |
+====+============+==========+
|  0 |          0 | 0.419009 |
+----+------------+----------+
|  1 |          1 | 0.580991 |
+----+------------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.4190 |
+-------------+-----------------+
| Survived(1) |          0.5810 |
+-------------+-----------------+
   Survived         p
0         0  0.419009
1         1  0.580991
```

Inference 3:-

- Target - survived
- Dependency - sex =1 (male)

```
q2 = bn.inference.fit(model, variables=['Survived'], evidence={'Sex':1})
print(q2)
print(q2.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+------------+----------+
|    |   Survived |        p |
+====+============+==========+
|  0 |          0 | 0.643264 |
+----+------------+----------+
|  1 |          1 | 0.356736 |
+----+------------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.6433 |
+-------------+-----------------+
| Survived(1) |          0.3567 |
+-------------+-----------------+
   Survived         p
0         0  0.643264
1         1  0.356736
```

Inference 4:-

- Target - survived
- Dependency - SbiSp(No of siblings or spouse) = 1 and Parch(No of parents and children) = 2

```
q3 = bn.inference.fit(model, variables=['Survived'], evidence={'SibSp':1,'Parch':2})
print(q3)
print(q3.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+------------+----------+
|    |   Survived |        p |
+====+============+==========+
|  0 |          0 | 0.533356 |
+----+------------+----------+
|  1 |          1 | 0.466644 |
+----+------------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.5334 |
+-------------+-----------------+
| Survived(1) |          0.4666 |
+-------------+-----------------+
   Survived         p
0         0  0.533356
1         1  0.466644
```

Inference 5:-

- Target - survived
- Dependency - Sex=1(male) and Parch(No of parents and children) = 2

```
q4 = bn.inference.fit(model, variables=['Survived'], evidence={'Sex':1, 'Parch': 2})
print(q4)
print(q4.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+-----------+----------+
|    |  Survived |        p |
+====+===========+==========+
|  0 |         0 | 0.648705 |
+----+-----------+----------+
|  1 |         1 | 0.351295 |
+----+-----------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.6487 |
+-------------+-----------------+
| Survived(1) |          0.3513 |
+-------------+-----------------+
   Survived         p
0         0  0.648705
1         1  0.351295
```

Inference 6:-

- Target - survived
- Dependency - Pclass = true and Embarked = 2

```
q5 = bn.inference.fit(model, variables=['Survived'], evidence={'Pclass':True,'Embarked':2})
print(q5)
print(q5.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+-----------+----------+
|    |  Survived |        p |
+====+===========+==========+
|  0 |         0 | 0.470009 |
+----+-----------+----------+
|  1 |         1 | 0.529991 |
+----+-----------+----------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.4700 |
+-------------+-----------------+
| Survived(1) |          0.5300 |
+-------------+-----------------+
   Survived         p
0         0  0.470009
1         1  0.529991
```

Inference 7:-

- Target - survived
- Dependency - parch = 2, sibsp=0 and Embarked = True

```
q6 = bn.inference.fit(model, variables=['Survived'], evidence={'Embarked': True, 'Parch': 2, 'SibSp': 0})
print(q6)
print(q6.df)
```

Output:

```
[bnlearn] >Variable Elimination.
[bnlearn] >Warning: variable(s) [None] does not exists in DAG.
[bnlearn] >Data is stored in [query.df]
+----+------------+---------+
|    |   Survived |       p |
+====+============+=========+
|  0 |          0 | 0.52032 |
+----+------------+---------+
|  1 |          1 | 0.47968 |
+----+------------+---------+
+-------------+-----------------+
| Survived    |   phi(Survived) |
+=============+=================+
| Survived(0) |          0.5203 |
+-------------+-----------------+
| Survived(1) |          0.4797 |
+-------------+-----------------+
   Survived        p
0         0  0.52032
1         1  0.47968
```

CONCLUSION: Bayesian networks allow you to perform probabilistic reasoning and draw conclusions about uncertain events by combining prior knowledge (encoded in the network structure and CPDs) with observed evidence. The conclusions drawn from the inference process can aid decision-making and provide insights into complex systems. We have studied the Bayesian Network and implemented it in Python with the help of Titanic dataset. We have found out the probabilities of various inferences drawn from the datasets.