

Experiment 1 -Bayesian Network using Python

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	LO1: Implement reasoning with uncertainty.

Aim: To implement the Bayesian Network using Python.

Uncertainty in AI:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but are not sure about it, so here we use probabilistic reasoning.

Probability

Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

We can find the probability of an uncertain event by using the below formula.

$$\text{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has been taken into account. It is a combination of prior probability and new information.

Bayesian Belief Network in artificial intelligence

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a **Bayes network**, **belief network**, **decision network**, or **Bayesian model**.

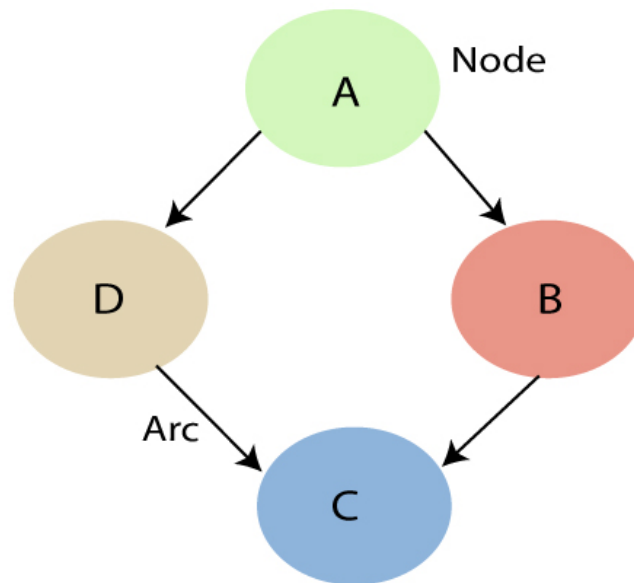
Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction**, **anomaly detection**, **diagnostics**, **automated insight**, **reasoning**, **time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.



- Each node corresponds to the random variables, and a variable can be continuous or discrete.
- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other

- In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.
- If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.
- Node C is independent of node A.

Python Library Function Used:

1. pandas -

Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures

and operations for manipulating numerical data and time series. This library is built on top of the NumPy library of Python. Pandas is fast and it has high performance & productivity for users.

2. **Seaborn** -

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3. **pgmpy**-

A Python library for working with probabilistic graphical models, including Bayesian networks.

4. **ParameterEstimator** -

A Parameter Estimator, in the context of statistics and machine learning, is a method or algorithm used to estimate the parameters of a statistical model or distribution from observed data. Parameters are the values that define the characteristics and behavior of a statistical model. Estimating these parameters is essential for fitting a model to data and making predictions or drawing inferences. Parameter estimation is a fundamental task in various fields, including statistics, machine learning, and data analysis.

5. **MaximumLikelihoodEstimator** -

Maximum Likelihood Estimation (MLE) is a widely used statistical method for estimating the parameters of a statistical model or distribution based on observed data. MLE aims to find the parameter values that maximize the likelihood function, which measures how well the model explains the observed data.


6. **VariableElimination** -

Variable Elimination is a popular algorithm used in probabilistic graphical models, particularly in the context of Bayesian networks, to perform exact probabilistic inference. It is used to calculate conditional probabilities, marginal probabilities, and other inference tasks efficiently.

Code and Observation:

Dataset :- <https://www.kaggle.com/datasets/gauravduttakiit/smoker-status-prediction>

1. Importing dataset

```
 df = pd.read_csv("/content/drive/MyDrive/AI&DS_2/Smoker_prediction/train_dataset.csv")
```

```
[ ] df
```

ight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	relaxation	...
85	97.0	0.9	0.9	1	1	118	78	...
110	110.0	0.7	0.9	1	1	119	79	...
65	86.0	0.9	0.9	1	1	110	80	...
80	94.0	0.8	0.7	1	1	158	88	...
60	81.0	1.5	0.1	1	1	109	64	...
...
60	80.0	0.4	0.6	1	1	107	60	...
55	75.0	1.5	1.2	1	1	126	72	...
105	124.0	0.6	0.5	1	1	141	85	...
55	75.0	1.5	1.5	1	1	95	69	...
60	81.1	1.0	1.0	1	1	114	66	...

2. Checking for null values

```
[ ] df.isnull()
```

	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...
38979	False	False	False	False	False	False	False
38980	False	False	False	False	False	False	False
38981	False	False	False	False	False	False	False
38982	False	False	False	False	False	False	False
38983	False	False	False	False	False	False	False

38984 rows × 23 columns

3. Getting standard values like mean, median, min, max, etc.

```
df.describe()
```



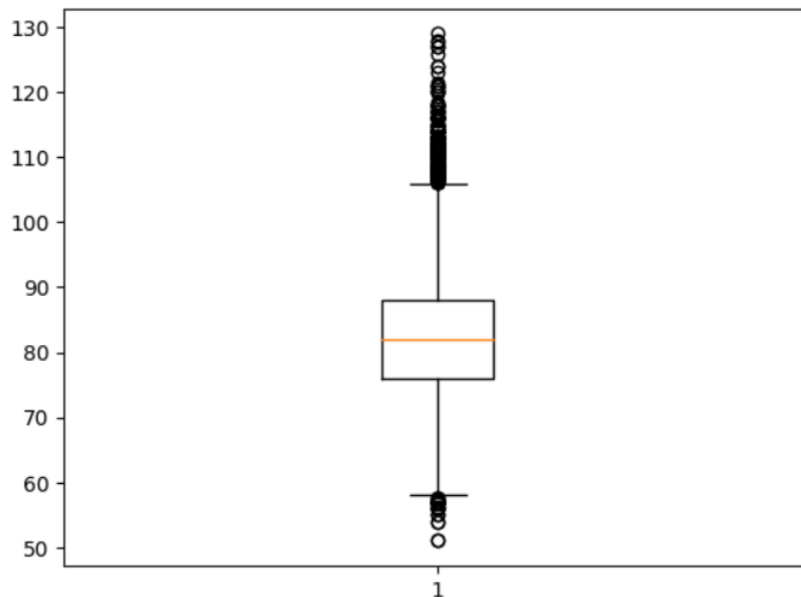
	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)
count	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000
mean	44.127591	164.689488	65.938718	82.062115	1.014955	1.008768	1.025369
std	12.063564	9.187507	12.896581	9.326798	0.498527	0.493813	0.157246
min	20.000000	130.000000	30.000000	51.000000	0.100000	0.100000	1.000000
25%	40.000000	160.000000	55.000000	76.000000	0.800000	0.800000	1.000000
50%	40.000000	165.000000	65.000000	82.000000	1.000000	1.000000	1.000000
75%	55.000000	170.000000	75.000000	88.000000	1.200000	1.200000	1.000000
max	85.000000	190.000000	135.000000	129.000000	9.900000	9.900000	2.000000

8 rows x 23 columns

4. Plotting graph for checking outliers.

```
[ ] import matplotlib.pyplot as plt
```

```
plt.boxplot(x = df['waist(cm)'])
plt.show()
```



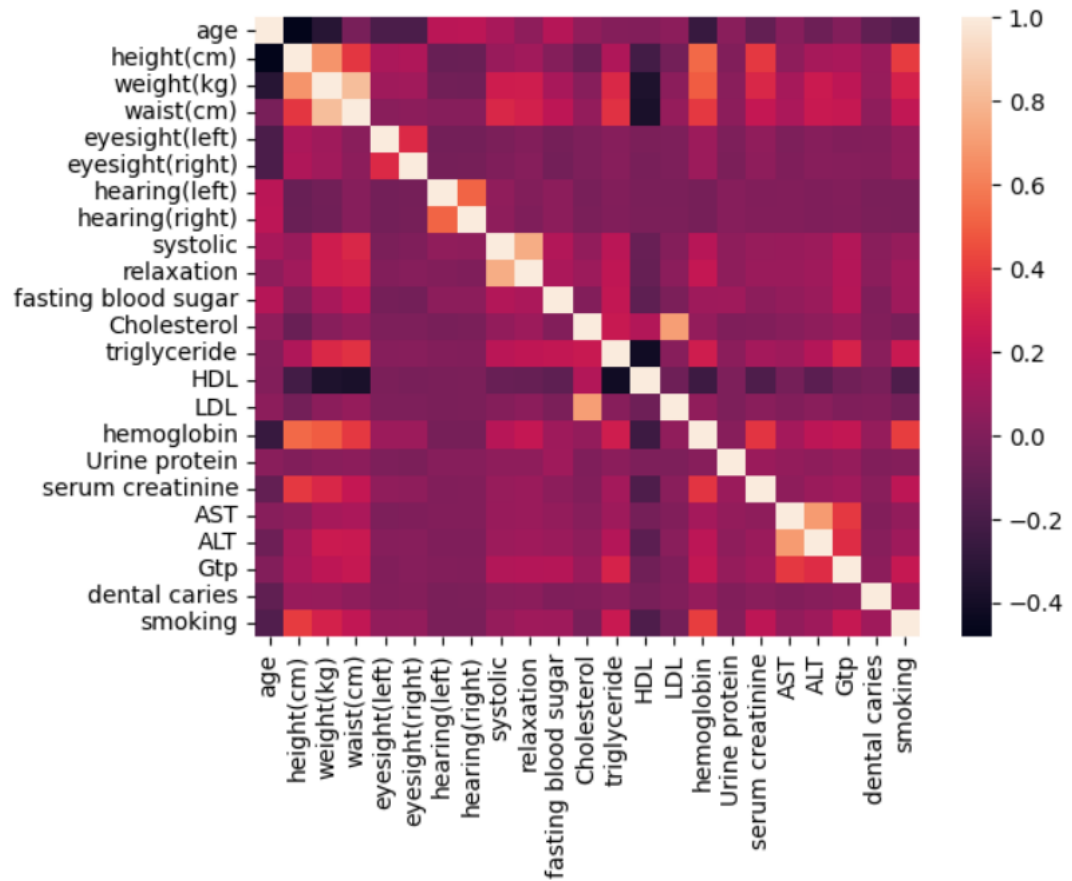
As we can see most of the values in the dataset are present as outliers hence we decided not to drop any values.

5. Plotting heatmap to visualize the correlation between features.

```
[ ] import seaborn as sns
```

```
[ ] sns.heatmap(data = df.corr())
```

<Axes: >



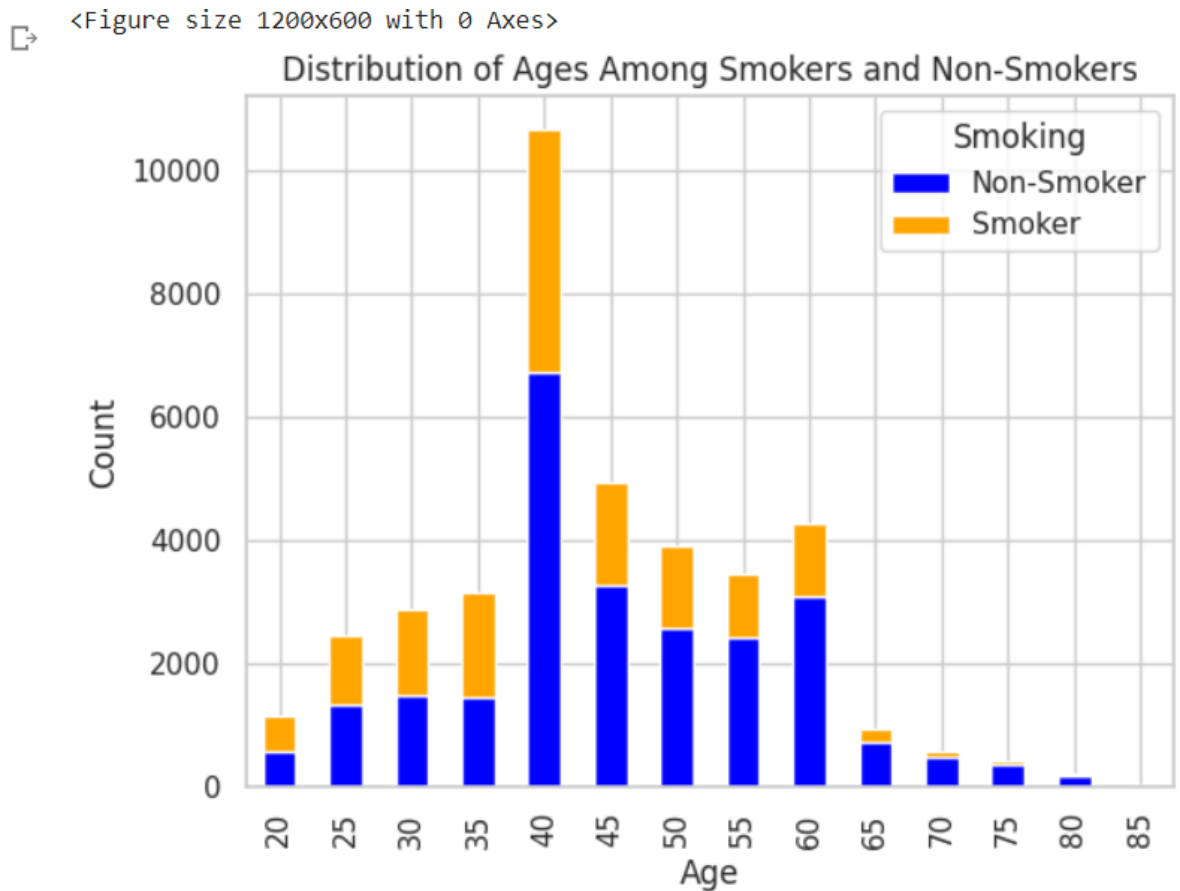

```
[ ] sns.set(style="whitegrid")

# Group the data by age and smoking status and count occurrences
age_smoking_counts = df.groupby(['age', 'smoking']).size().unstack(fill_value=0)

# Create a bar graph
plt.figure(figsize=(12, 6)) # Adjust the figure size as needed
age_smoking_counts.plot(kind='bar', stacked=True, color=['blue', 'orange'])

# Set plot title and labels
plt.title('Distribution of Ages Among Smokers and Non-Smokers')
plt.xlabel('Age')
plt.ylabel('Count')

# Display the plot
plt.legend(title='Smoking', labels=['Non-Smoker', 'Smoker'])
plt.show()
```



Install **pgmpy** library to create a bayesian network with this dataset.

```
] pip install pgmpy
```

Requirement already satisfied: pgmpy in /usr/local/lib/python3.10/dist-packages (0.1.23)

Importing necessary libraries of creating bayesian belief network

```
[ ] from pgmpy.models import BayesianNetwork
    from pgmpy.estimators import ParameterEstimator, MaximumLikelihoodEstimator
    from pgmpy.inference import VariableElimination
```

Creating a subset of the dataset which have high correlation with the target attribute

```
variables = ['smoking', 'height(cm)', 'weight(kg)', 'waist(cm)', 'triglyceride', 'serum creatinine', 'LDL']
subset = df[variables]
```

selected variables which have strong relationship with target variable (smoking)

```
[ ] model = BayesianNetwork([('triglyceride', 'smoking'), ('serum creatinine', 'smoking'), ('LDL', 'smoking')])
```

```
[ ] model.fit(df, estimator=MaximumLikelihoodEstimator)
```

```
[ ] df_infer = VariableElimination(model)
```

`.fit(df, estimator=MaximumLikelihoodEstimator)`: This is a function call on our model object, and it is used to estimate or learn the conditional probability distributions (CPDs) associated with the nodes in your Bayesian network.

The line `df_infer = VariableElimination(model)` is used to create an inference engine object based on your previously defined Bayesian network model. In this case, `VariableElimination` is being used as the inference method.

`q = df_infer.query(variables=['smoking'], evidence={'LDL': 50, 'serum creatinine': 10})`: This line of code performs a query on the Bayesian network. It is asking for the probability distribution of the variable 'smoking' given the evidence that 'LDL' is 50 and 'serum creatinine' is 10. In other words, it's trying to find the probability of 'smoking' being true or false under these conditions.

Output:

```
q=df_infer.query(variables=['smoking'],evidence={'LDL':50, 'serum creatinine': 10})
target_factor = q.values
print(target_factor)
```

```
[0.5 0.5]
```

Here the output [0.5, 0.5] indicates that,

- The first value (0.5) represents the probability that 'smoking' is true (i.e., the person is a smoker) given that 'LDL' is 50 and 'serum creatinine' is 10.
- The second value (0.5) represents the probability that 'smoking' is false (i.e., the person is not a smoker) given the same evidence.

Conclusion:

In conclusion, building a Bayesian Network model for smoker prediction offers a principled and interpretable approach to modeling uncertainty and dependencies among variables. It allows for efficient and exact probabilistic inference, transparently represents relationships between factors, and can incorporate domain knowledge effectively. These qualities make Bayesian Networks a valuable choice for predictive modeling, particularly when understanding and interpreting the results are crucial.