Experiment No 5

Aim: Implementation of Bayesian algorithm

Theory:

Data Mining Bayesian Classifiers

The connection between the attribute set and the class variable is non-deterministic in numerous applications. In other words, we can say the class label of a test record can't be assumed with certainty even though its attribute set is the same as some of the training examples. These circumstances may emerge due to the noisy data or the presence of certain confusing factors that influence classification, but it is not included in the analysis. For example, consider the task of predicting the occurrence of whether an individual is at risk for liver illness based on the individual's eating habits and working efficiency. Although most people who eat healthy and exercise consistently have less probability of occurrence of liver disease, they may still do so due to other factors. For example, due to the consumption of high-calorie street foods and alcohol abuse. Determining whether an individual's eating routine is healthy or the workout efficiency is sufficient is also subject to analysis, which may introduce vulnerabilities to the learning issue.

Bayesian classification uses Bayes theorem to predict the occurrence of any event. Bayesian classifiers are the statistical classifiers with the Bayesian probability understandings. The theory expresses how a level of belief, is expressed as a probability.

Bayes theorem came into existence after Thomas Bayes, who first utilized conditional probability to provide an algorithm that uses evidence to calculate limits on an unknown parameter.

Bayes's theorem is expressed mathematically by the following equation that is given below.

For proposition X and evidence Y,

- P(X), the prior, is the primary degree of belief in X
- P(X/Y), the posterior is the degree of belief having accounted for Y.
- The quotient $\frac{P(Y/X)}{P(Y)}$ represents the supports Y provides for X.

Bayes theorem can be derived from the conditional probability:

$$P(X/Y) = \frac{P(X \cap Y)}{P(Y)}, \text{ if } P(Y) \neq 0$$

$$P(Y/X) = \frac{P(Y \cap X)}{P(X)}, \text{ if } P(X) \neq 0$$

Where P (X∩Y) is the **joint probability** of both X and Y being true, because

$$P(Y \cap X) = P(X \cap Y)$$

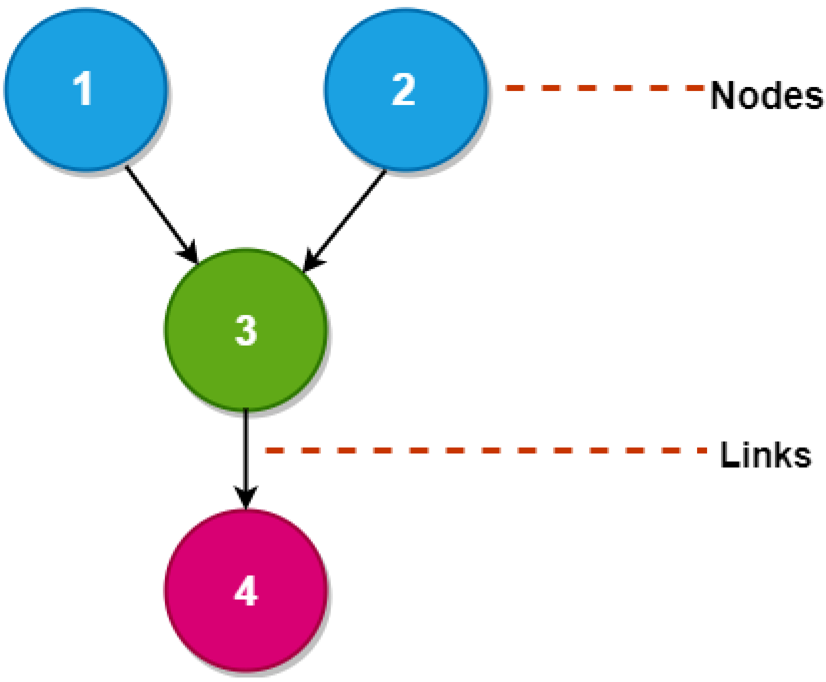
$$\text{or, } P(X \cap Y) = P(X/Y)P(Y) = P(Y/X)P(X)$$

$$\text{or, } P(X/Y) = \frac{P(Y/X)P(X)}{P(Y)}, \text{ if } P(Y) \neq 0$$

Bayesian network:

A Bayesian Network falls under the classification of Probabilistic Graphical Modelling (PGM) procedure that is utilized to compute uncertainties by utilizing the probability concept. Generally known as **Belief Networks, Bayesian Networks** are used to show uncertainties using **Directed Acyclic Graphs** (DAG)

A **Directed Acyclic Graph** is used to show a Bayesian Network, and like some other statistical graph, a DAG consists of a set of nodes and links, where the links signify the connection between the nodes.



The nodes here represent random variables, and the edges define the relationship between these variables.

A DAG models the uncertainty of an event taking place based on the Conditional Probability Distribution (CDP) of each random variable. A **Conditional Probability Table** (CPT) is used to represent the CPD of each variable in a network.

CODE:

```
import pandas as pd
import math

data = pd.read_csv("/content/inventory_bayes.csv")

price = input("Enter Price (Low, Average, High): ")
number_of_boxes = int(input("Enter Number Of Boxes: (30, 40, 10): "))
product_brand = input("Enter Product Brand (Sony, LG, Nike): ")
location = input("Enter location: (Suburbs, City, Downtown): ")

print("\n")

#prior probabilities

total_instances = len(data)
package_left_yes = len(data[data['package_left'] == 'Yes'])
package_left_no = len(data[data['package_left'] == 'No'])

p_yes = round(package_left_yes/total_instances, 4)
p_no = round(package_left_no/total_instances, 4)

print(f"Total occurances of Yes: {package_left_yes}")
print(f"Total occurances of No: {package_left_no}")

print(f"Prior probability for Yes {p_yes}")
print(f"Prior probability for No: {p_no}")

#Likelihood Probabilities

def calculate_likelihood(attribute, value, package_left):
  subset = data[data['package_left'] == package_left]
  count = len(subset[subset[attribute] == value])
  total = len(subset)
```

```
  return count / total

print('\n')
print('\n')
print("The Likelihood Probabilities are: ")

#
p_price_given_yes = round(calculate_likelihood('price', price, 'Yes'), 4)
p_price_given_no = round(calculate_likelihood('price', price, 'No'), 4)

print(f"P(price = {price} / buy='Yes') = {p_price_given_yes}")
print(f"P(price = {price} / buy='No') = {p_price_given_no}")
print('\n')


#
p_number_of_boxes_given_yes = round(calculate_likelihood('number_of_boxes',
number_of_boxes, 'Yes'), 4)
p_number_of_boxes_given_no = round(calculate_likelihood('number_of_boxes',
number_of_boxes, 'No'), 4)

print(f"P(number_of_boxes = {number_of_boxes} / buy='Yes') =
{p_number_of_boxes_given_yes}")
print(f"P(number_of_boxes = {number_of_boxes} / buy='No') =
{p_number_of_boxes_given_no}")
print('\n')

#
p_product_brand_given_yes = round(calculate_likelihood('product_brand',
product_brand, 'Yes'), 4)
p_product_brand_given_no = round(calculate_likelihood('product_brand',
product_brand, 'No'), 4)

print(f"P(product_brand = {product_brand} / buy='Yes') =
{p_product_brand_given_yes}")
print(f"P(product_brand = {product_brand} / buy='No') =
{p_product_brand_given_no}")
print('\n')

#
p_location_given_yes = round(calculate_likelihood('location', location, 'Yes'), 4)
```

```
p_location_given_no = round(calculate_likelihood('location', location, 'No'), 4)

print(f"P(location = {location} / buy='Yes') = {p_location_given_yes}")
print(f"P(location = {location} / buy='No') = {p_location_given_no}")
print('\n')

# Posterior probabilities
print("The posterior probabilities are: ")

p_yes_given_x = round(p_yes * p_price_given_yes * p_number_of_boxes_given_yes *
p_product_brand_given_yes * p_location_given_yes,4)
p_no_given_x = round(p_no * p_price_given_no * p_number_of_boxes_given_yes *
p_product_brand_given_no * p_location_given_no,4)

if p_yes_given_x > p_no_given_x:
  prediction = 'Yes'
else:
  prediction = 'No'

print(f'Probability of Purchased=Yes: {p_yes_given_x}')
print(f'Probability of Purchased=No: {p_no_given_x:}')
print(f'Prediction: {prediction}')
```

Output: