**Experiment 5**

**Aim: Implementation of Bayesian algorithm**

**Theory:**

Naïve Bayes Classifier Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in text classification that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

Naïve Bayes Classifier Algorithm

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

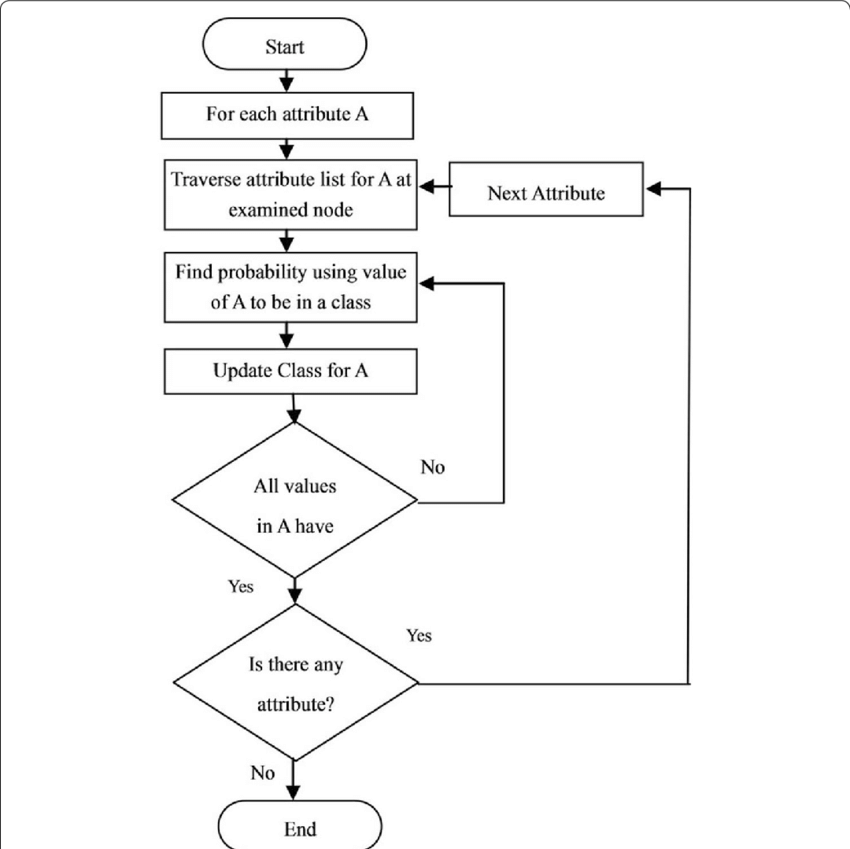
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

**Algorithm:**

* Import the CSV file of the dataset.
* For each dependent feature of the dataset, create a frequency table and
* count the occurrence of each value, concerning the positive and negative
* outcome of the class variable. For instance, count how many candidates have
* been selected who have STRONG/WEAK/AVG knowledge of DSA.
* Post the counting, we find the probability for all the values of the features.
* Then we take an arbitrary case input from the user for which prediction is to
* be made.
* We calculate the 2 probabilities using dependent features and target variable
* such that, in one case the target variable has a positive outcome and, in
* another case, it has a negative outcome.
* To make the prediction we divide each probability in Step 5 by the sum of
* both probabilities to perform normalization and convert the outcome into a
* valid probability value.
* If the value of probability is higher for the positive outcome of the target
* variable then the user input case is likely to occur, else the outcome is
* negative.

**Flowchart:**



**Fig 3.1 Flowchart of Naïve Bayes**

**Dataset used:**

| id | age | income | student | credit\_rating | Buy\_Computer |
| --- | --- | --- | --- | --- | --- |
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle\_age | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle\_age | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle\_age | medium | no | excellent | yes |
| 13 | middle\_age | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

**Code:**

import csv

def predict(test, PYES, PNO):

pinyes = AGE\_Y[test[0]] \* INCOME\_Y[test[1]] \* STUDENT\_Y[test[2]] \* CREDIT\_RATING\_Y[test[3]] \* PYES

pinno = AGE\_N[test[0]] \* INCOME\_N[test[1]] \* STUDENT\_N[test[2]] \* CREDIT\_RATING\_N[test[3]] \* PNO

sumprob = pinyes + pinno

pinyes = pinyes / sumprob

pinno = pinno / sumprob

print("Probability of Buying a Computer (Yes):", pinyes)

print("Probability of Not Buying a Computer (No):", pinno)

if pinyes > pinno:

print("Prediction: Buy a Computer")

else:

print("Prediction: Do Not Buy a Computer")

def testInput():

string = input("Enter comma-separated test tuple (age,income,student,credit\_rating):")

test = string.split(',')

return test

def probabilityComputation(dictionary):

value = sum(dictionary.values())

for key in dictionary:

dictionary[key] /= value

def preprocessing():

PYES = PNO = 0

with open('./Buy\_Computer.csv', mode='r') as csv\_file:

csv\_reader = csv.DictReader(csv\_file)

nyes = nno = 0

for row in csv\_reader:

if row['Buy\_Computer'] == 'yes':

AGE\_Y[row['age']] += 1

INCOME\_Y[row['income']] += 1

STUDENT\_Y[row['student']] += 1

CREDIT\_RATING\_Y[row['credit\_rating']] += 1

nyes += 1

else:

AGE\_N[row['age']] += 1

INCOME\_N[row['income']] += 1

STUDENT\_N[row['student']] += 1

CREDIT\_RATING\_N[row['credit\_rating']] += 1

nno += 1

PYES = nyes / (nyes + nno)

PNO = nno / (nyes + nno)

probabilityComputation(AGE\_Y)

probabilityComputation(AGE\_N)

probabilityComputation(INCOME\_Y)

probabilityComputation(INCOME\_N)

probabilityComputation(STUDENT\_Y)

probabilityComputation(STUDENT\_N)

probabilityComputation(CREDIT\_RATING\_Y)

probabilityComputation(CREDIT\_RATING\_N)

return PYES, PNO

AGE\_Y = {

'youth': 0,

'middle\_age': 0,

'senior': 0

}

AGE\_N = {

'youth': 0,

'middle\_age': 0,

'senior': 0

}

INCOME\_Y = {

'low': 0,

'medium': 0,

'high': 0

}

INCOME\_N = {

'low': 0,

'medium': 0,

'high': 0

}

STUDENT\_Y = {

'no': 0,

'yes': 0

}

STUDENT\_N = {

'no': 0,

'yes': 0

}

CREDIT\_RATING\_Y = {

'fair': 0,

'excellent': 0

}

CREDIT\_RATING\_N = {

'fair': 0,

'excellent': 0

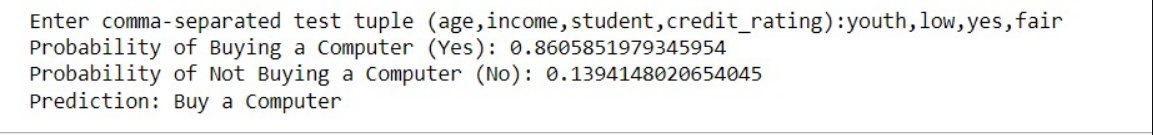
}

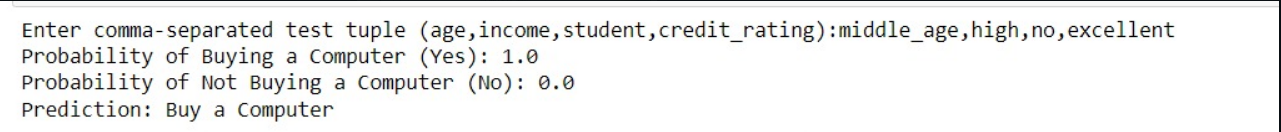
PYES, PNO = preprocessing()

test = testInput()

predict(test, PYES, PNO)

**Output:**

****

****