**Experiment No. 5**

**Aim:** Implementation of Bayesian algorithm.

**Theory:**

Naive Bayes is a statistical classification technique based on Bayes

Theorem. It is one of the simplest supervised learning algorithms. Naive

Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes

classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in

a class is independent of other features. For example, a loan applicant

is desirable or not depending on his/her income, previous loan and

transaction history, age, and location. Even if these features are

interdependent, these features are still considered independently. This

assumption simplifies computation, and that is why it is considered as

naive. This assumption is called class conditional independence.



● P(h): the probability of hypothesis h being true (regardless of the data). This is

known as the prior probability of h.

● P(D): the probability of the data (regardless of the hypothesis). This is known

as the prior probability.

● P(h|D): the probability of hypothesis h given the data D. This is known as

posterior probability.

● P(D|h): the probability of data d given that hypothesis h was true. This is

known as posterior probability.

**How does Naive Bayes classifier work?**

Let us understand the working of Naive Bayes through an example.

Given an example of weather conditions and playing sports. You need

to calculate the probability of playing sports. Now, you need to classify

whether players will play or not, based on the weather condition.

**Approach (In case of a single feature)**

Naive Bayes classifier calculates the probability of an event in the

following steps:

● Step 1: Calculate the prior probability for given class labels

● Step 2: Find Likelihood probability with each attribute for each

class

● Step 3: Put these values in Bayes Formula and calculate posterior

probability.

● Step 4: See which class has a higher probability, given the input

belongs to the higher probability class.

For simplifying prior and posterior probability calculation you can use

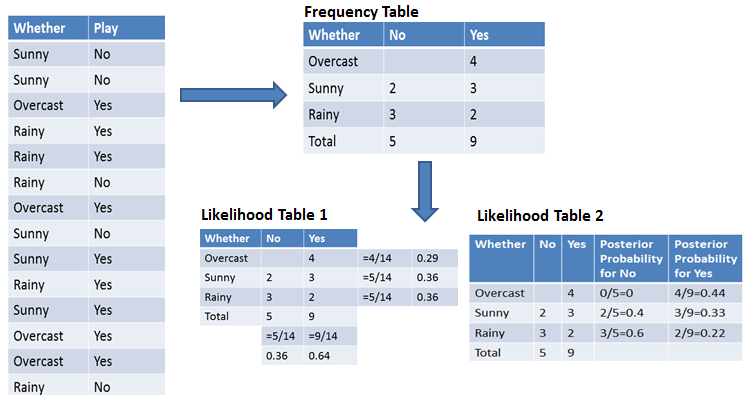
the two tables frequency and likelihood tables. Both tables will

help you to calculate the prior and posterior probability. The Frequency

table contains the occurrence of labels for all features. There are two

likelihood tables. Likelihood Table 1 is showing prior probabilities of

labels and Likelihood Table 2 is showing the posterior probability.



Now suppose you want to calculate the probability of playing when the

weather is overcast.

**Probability of playing:**

P(Yes | Overcast) = P(Overcast | Yes) P(Yes) / P (Overcast)

.....................(1)

1. Calculate Prior Probabilities:

P(Overcast) = 4/14 = 0.29

P(Yes)= 9/14 = 0.64

2. Calculate Posterior Probabilities:

P(Overcast |Yes) = 4/9 = 0.44

3. Put Prior and Posterior probabilities in equation (1)

P (Yes | Overcast) = 0.44 \* 0.64 / 0.29 = 0.98(Higher)

Similarly, you can calculate the probability of not playing

The probability of a 'Yes' class is higher. So, you can determine here if

the weather is overcast than players will play the sport.

**Code:**

import java.util.\*;

class Data {

int count = 0;

Map < String, Integer > freq = new HashMap < > ();

double probVariable = 1;

void calculateProbability(String s) {

Integer frequency = freq.get(s);

if (frequency != null && count != 0) {

probVariable \*= frequency / (double) count;

}

}

}

public class BayesAlgo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of row and column:");

int n = sc.nextInt();

int m = sc.nextInt();

sc.next();

String data[][] = new String[n][m];

for (int i = 0; i < n; i++) {

String row = sc.nextLine();

String[] values = row.split(" ");

if (values.length != m) {

System.out.println("Invalid input. Number of columns does not

match.

");

sc.close();

return;

}

for (int j = 0; j < m; j++) {

data[i][j] = values[j];

}

}

System.out.println("Enter the column name for class variable:");

String classVariable = sc.next();

Map < String, Data > distinct = new HashMap < > ();

for (int i = 0; i < data[0].length; i++) {

if (data[0][i].equals(classVariable)) {

for (int j = 1; j < data.length; j++) {

String classValue = data[j][i];

if (!distinct.containsKey(classValue)) {

distinct.put(classValue, new Data());

}

Data classData = distinct.get(classValue);

classData.count++;

for (int k = 0; k < data[0].length; k++) {

if (k == i)

continue;

int cnt = classData.freq.getOrDefault(data[j][k], 0);

classData.freq.put(data[j][k], ++cnt);

}

}

}

}

System.out.println("Enter the number of inputs:");

int col = sc.nextInt();

System.out.println("Enter the tuple in columnValue:");

for (int i = 0; i < col; i++) {

String s = sc.next();

for (Map.Entry < String, Data > it: distinct.entrySet()) {

it.getValue().calculateProbability(s);

}

}

String ans = "";

double ansProb = 0;

double total = 0;

for (Map.Entry < String, Data > it: distinct.entrySet()) {

total += (it.getValue().probVariable \* (it.getValue().count \* 1.0) / (n - 1));

if ((it.getValue().probVariable \* (it.getValue().count \* 1.0) / (n - 1)) > ansProb) {

ans = it.getKey();

ansProb = (it.getValue().probVariable \* (it.getValue().count \* 1.0) / (n - 1));

}

}

System.out.println("\nFinal class:" + ans);

System.out.println("Probability:" + (ansProb / total));

sc.close();

}

}

**Output:**

Enter the number of row and column:

6

4

Outlook Temperature Humidity PlayTennis

Sunny Hot High No

Sunny Hot High Yes

Overcast Hot High No

Rainy Mild High No

Rainy Cool Normal Yes

Enter the column name for class variable:

PlayTennis

Enter the number of inputs:

2

Enter the tuple in columnValue:

Rainy Cool

Final class: No

Probability:0.6666666666666666

**Conclusion:** Thus, in this experiment, we have implemented Bayesian algorithm.