Experiment No.4

Aim: Implementation of Bayesian algorithm

Theory:

In this experiment, we utilize the Naive Bayes algorithm to predict whether a plant will be healthy('YES') or not('NOT') based on the following features:

- **Features:**
  - **Plant Family**
  - **Soil Type**
  - **Season**
  - **Activity Name**
  - **Activity Frequency**
- **Class Label:**
  - **Outcome:** Whether the plant will be Healthy ('YES') or not ('NO').

The Naive Bayes algorithm calculates the likelihood of each outcome by evaluating the conditional probabilities associated with these features, based on the training data. Given a new set of inputs (Plant Family,Soil Type, Season,Activity Name,Activity Frequency), the algorithm predicts 'YES' if the probability of being healthy if no 'NO'.

Data: The algorithm is trained on a dataset consisting of 50 rows. Below is a snapshot of a few rows from the training data:

plant_family	soil_type	season	activity_name	activity_frequency	healthy
Rosaceae	Loamy	Winter	Watering	4	Yes
Lamiaceae	Sandy	Summer	Fertilizing	1	No
Asteraceae	Clay	Spring	Watering	3	Yes
Solanaceae	Loamy	Autumn	Pruning	2	No
Poaceae	Sandy	Winter	Watering	4	Yes
Fabaceae	Loamy	Summer	Fertilizing	1	No
Lamiaceae	Sandy	Spring	Watering	4	Yes
Rosaceae	Clay	Autumn	Pruning	2	No
Solanaceae	Sandy	Winter	Watering	4	Yes
Poaceae	Loamy	Summer	Fertilizing	1	Yes
Asteraceae	Clay	Spring	Watering	3	No
Fabaceae	Sandy	Autumn	Pruning	2	Yes
Rosaceae	Loamy	Winter	Watering	4	Yes
Lamiaceae	Sandy	Summer	Fertilizing	1	No
Asteraceae	Clay	Spring	Watering	3	Yes
Solanaceae	Loamy	Autumn	Pruning	2	No

Main.java

```
package com.naive;

import java.io.IOException;
import java.util.Scanner;
import java.util.Arrays;

import static java.lang.System.*;

public class Main {
   public static void main(String[] args) {
      try {
         NaiveBayesClassifier Healthy = new NaiveBayesClassifier();
         CSVUtil.loadCSV("d:\\Users\\ishwa\\Downloads\\Healthy.csv",Healthy); // Update with
your CSV file path
         Scanner scanner = new Scanner(in);

         out.print("Enter Plant Family: ");
         String plantFamily = scanner.nextLine();
         out.print("Enter Soil Type: ");
         String soilType = scanner.nextLine();
         out.print("Enter Season: ");
         String season = scanner.nextLine();
         out.print("Enter Activity Name (e.g., Watering): ");
         String activityName = scanner.nextLine();
         out.print("Enter Activity Frequency: ");
         String activityFrequency = scanner.nextLine();

         String[] input = {plantFamily, soilType, season, activityName, activityFrequency};
         String prediction = Healthy.predict(input);

         out.println("Input: " + Arrays.toString(input));
         out.println("Prediction (Healthy): " + prediction);

         scanner.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

CSVUtil.java

```
package com.naive;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;

import java.io.FileReader;
import java.io.IOException;

public class CSVUtil {
   public static void loadCSV(String filePath, NaiveBayesClassifier classifier) throws
IOException {
      FileReader reader = new FileReader(filePath);
      CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader());

      for (CSVRecord record : csvParser) {
         String[] row = {record.get("plant_family"), record.get("soil_type"), record.get("season"),
               record.get("activity_name"), record.get("activity_frequency")};
         String outcome = record.get("healthy"); // The last column should be "healthy" (YES/NO)
         classifier.train(row, outcome);
      }

      csvParser.close();
      reader.close();
   }
}
```

NaiveBayesClassifier.java

```
package com.naive;

import java.util.HashMap;

public class NaiveBayesClassifier {
   private HashMap<String, Integer> yesCount = new HashMap<>();
   private HashMap<String, Integer> noCount = new HashMap<>();
   private int totalYes = 0;
   private int totalNo = 0;

   // Train the classifier with the input row and outcome (Healthy: Yes/No)
   public void train(String[] row, String outcome) {
      if (outcome.equalsIgnoreCase("YES")) {
         totalYes++;
         incrementCount(yesCount, row);
```

```
        } else {
            totalNo++;
            incrementCount(noCount, row);
        }
    }

    // Increment counts for "YES" or "NO" outcomes
    private void incrementCount(HashMap<String, Integer> countMap, String[] row) {
        for (String key : row) {
            countMap.put(key, countMap.getOrDefault(key, 0) + 1);
        }
    }

    // Calculate the probability of "YES" or "NO" given the input features
    public double calculateProbability(String[] input, String outcome) {
        double probability = 1.0;
        HashMap<String, Integer> countMap = outcome.equalsIgnoreCase("YES") ? yesCount :
noCount;
        int totalCount = outcome.equalsIgnoreCase("YES") ? totalYes : totalNo;
        int vocabularySize = 5; // Number of unique features in the input (Plant Family, Soil Type,
etc.)
        int laplaceConstant = 1; // Smoothing constant

        for (String key : input) {
            probability *= ((double) countMap.getOrDefault(key, 0) + laplaceConstant) /
                    (totalCount + laplaceConstant * vocabularySize);
        }
        probability *= (double) totalCount / (totalYes + totalNo);
        return probability;
    }

    // Predict if the plant is healthy (YES) or not (NO) based on input features
    public String predict(String[] input) {
        double yesProbability = calculateProbability(input, "YES");
        double noProbability = calculateProbability(input, "NO");

        System.out.println(String.format("P(Healthy = YES | input) = %.6f", yesProbability));
        System.out.println(String.format("P(Healthy = NO | input) = %.6f", noProbability));

        return yesProbability > noProbability ? "YES" : "NO";
    }
}
```

OUTPUT:

```
Enter Plant Family: Rosaceae
Enter Soil Type: Loamy
Enter Season: Winter
Enter Activity Name (e.g., Watering): Watering
Enter Activity Frequency: 4
P(Healthy = YES | input) = 0.003113
P(Healthy = NO | input) = 0.000010
Input: [Rosaceae, Loamy, Winter, Watering, 4]
Prediction (Healthy): YES
```

```
Enter Plant Family: Poaceae
Enter Soil Type: Clay
Enter Season: Rainy
Enter Activity Name (e.g., Watering): Watering
Enter Activity Frequency: 6
P(Healthy = YES | input) = 0.000014
P(Healthy = NO | input) = 0.000002
Input: [Poaceae, Clay, Rainy, Watering, 6]
Prediction (Healthy): YES
```

```
Enter Plant Family: Fabaceae
Enter Soil Type: Loamy
Enter Season: Summer
Enter Activity Name (e.g., Watering): Fertilizing
Enter Activity Frequency: 1
P(Healthy = YES | input) = 0.000084
P(Healthy = NO | input) = 0.001400
Input: [Fabaceae, Loamy, Summer, Fertilizing, 1]
Prediction (Healthy): NO
```

Conclusion:

In this experiment, the Bayesian algorithm was successfully implemented to predict outcomes based on given inputs (Plant Family,Soil Type, Season,Activity Name,Activity Frequency). The model was trained using data from a CSV file, and predictions were made by calculating the conditional probabilities for 'YES' and 'NO' outcomes. The experiment demonstrated the effectiveness of Naive Bayes in making probabilistic predictions based on historical data, showcasing its utility in decision-making scenarios.