RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to VTU)

Department of Information Science and Engineering

June - December 2016

Submitted in partial fulfilment of the CIE for the subject

Data Mining Lab - IS711L

Shoaib Ahmed - 1MS13IS101

Contents

PART A

STEP 1: DATA SET CREATION

Data set can be created in excel file or CSV format file. To create data set in excel follow the instructions:

1. Get your data set ready in Excel. Your set can have as many rows and columns as you want, with or without a heading row
2. Insert two empty columns to the left of your data. To do this, you can right-click within the A column and click "insert" twice. Alternatively, you can click into the A column, select "Insert" from the menu bar, and click on "Columns."
3. Type = RAND() into your first empty cell. This should be in the A column, under any heading row. The function will generate a random number between 0 and 1 in that cell.
4. Copy the Rand() formula and paste it all the way down your A column. Make sure that every piece of data in your set now has a random number next to it.
5. Highlight the whole column of random numbers. Copy these values.
6. Paste the random values in the B column.Make sure you use "Paste Special" and select the "Values" option. This will tell Excel to copy the values but not the formulas.The RAND() formula recalculates a new random number each time you do anything else in the spreadsheet, but copying the values over to a new column will prevent them from changing in future.
7. Sort your random values. Select the whole B column. Click on the icon for sorting (or go to "Data" in the menu bar and hit "Sort") and select "Ascending."
    a. Make sure you select "Expand the Selection" and then "Sort" so that the other columns are rearranged along with the B column.
    b. You can now delete your A and/or B columns. You won't need them again unless you want to sort the selection again.
8. Select your random sample. You can pick however many rows or cells you want for your sample size. Just take the data you want, starting at the top of the list, to make up your sample. Because they were ordered according to random numbers, the sample will be a random selection from your data set, too.

STEP 2: PREPROCESSING

The preprocessing techniques that can be applied are as follows:

a. Handling Missing Values

1) Create Data set in excel then save as csv (comma delimited) format .

2) Click Import and drag Read CSV to workspace.

3) Go to Import Configuration Wizard -Import the csv file-Press Next, make File encoding as System -Make Column Separation as Comma “,”- Press Next and Finish.

4) Click Data Transformation -> Data Cleansing -> Replace Missing Values. Drag it to Workspace.

5) Click on Replace Missing Values-Make attribute filter type as “block_type” and default as “average”.

6) Connect the connections.

7) Play and get the Result as Data View.

b. Redundancy Techniques

Steps:-

Create Data set in excel then save as csv (comma delimited) format .

1) Click Import and drag Read CSV to workspace.

2)Go to Import Configuration Wizard

-Import the csv file

-Press Next, make File encoding as System

-Make Column Separation as Comma “,”

- Press Next and Finish.

3) Click Data Transformation -> Filtering -> Remove Duplicates. Drag it to Workspace.

4) Click on Remove Duplicates

-Select attribute filter type as “single”

-Select attribute as “a3”

5) Connect the connections.

6)Play and get the Result as Data View.

c. Binning:

Steps :-

1) Create Data set in excel then save as csv (comma delimited) format .
2) Click Import and drag Read CSV to workspace.
3) Go to Import Configuration Wizard -Import the csv file

    -Press Next, make File encoding as System

    -Make Column Separation as Comma “,”

    - Press Next and Finish.
4) Click Data Transformation -> Type Conversion -> Discretization -> Discretize by Binning. Drag it to workspace.
5) Click Discretize by Binning-Make attribute filter type as “single” -Make attribute as “a1”

    -Make number of bins as 2

    -Make min value as 0

    -Make max value as 50
6) Connect the connections.
7) Play and get the Result as Data View.

d. Normalization Techniques

Steps:-

1) Create Data set in excel then save as csv (comma delimited) format.
2) Click Import and drag Read CSV to workspace.
3) Go to Import Configuration Wizard

    -Import the csv file

    -Press Next, make File encoding as System

    -Make Column Separation as Comma “,”

    -Make tid as “label”

    - Press Next and Finish.
4) Click Data Transformation -> Value Modification -> Numerical Value Modification - > Normalize. Drag it to Workspace.
5) Click on Normalize

    -Select attribute filter type as “single”

    -Select attribute as “a1”

-Select method as “Z-Transformation”

6) Connect the connections.

7) Play and get the Result as Data View.

8) Again, Click on Normalize

-Select attribute filter type as “all”

-Select method as “Range Transformation” -Select min as 0.0

-Select max as 3.0

9) Connect the connections.

10) Play and get the Result as Data View.

STEP 3:

a) FP Growth Algorithm

Steps :-

1) Create Data set in excel then save as csv (comma delimited) format.

2) Click Import and drag Read CSV to workspace.

3)Go to Import Configuration Wizard

-Import the csv file

-Press Next, make File encoding as System

-Make Column Separation as Comma “,”

-Make tid as “label” and other attributes as “binomial”. - Press Next and Finish.

4) Click Modeling -> Association and Item Set Mining -> FP Growth. Drag it to workspace.

5) Connect the connections.

6) Click FP Growth

-Make min support as 0.2

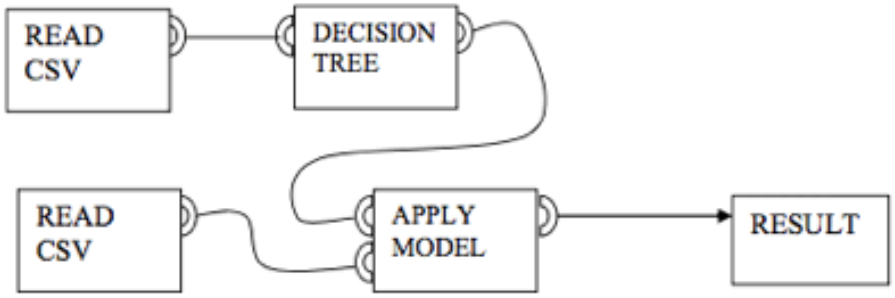
-Make max items as -1

7) Play and get the Result as Table View.

b) Decision Tree

STEPS:

1. Import the CSV file for the training set and test set.
2. Set the delimiter as comma [,].
3. Drag and drop “Decision tree”.
4. Drag and drop “Apply model”.
5. Make connections as shown in figure.
6. Run the process



SCHEMATIC DIAGRAM

c) Naïve Bayes Classification

1. Import the supervised dataset in the R tool/Rapidminer design perspective.
2. Drag and drop the missing values operator onto the design perspective and make changes in the appropriate
3. Place the duplication operator onto the design perspective and make appropriate changes in its properties.
4. Include the type conversion operator to convert the data
5. Use the Naïve Bayes operator to generate probabilities.
6. Analyze them and classify new data.

PART B

1.Write a java/c++ program to perform aggregation and discretization on a given dataset.

AggregateAndDiscretize.java

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

public class AggregateAndDiscretize {
	ReadCsv rcsv;
	static Scanner sc = new Scanner(System.in);
	int min;
	int max;
	int col;
	AggregateAndDiscretize(String fileName) throws IOException{
		rcsv = new ReadCsv(fileName, ",");
		System.out.println("Enter the column number: ");
		col = sc.nextInt() - 1;
		min = Integer.MAX_VALUE;
		max = Integer.MIN_VALUE;
		this.aggregate();
		this.discretize();
	}
	void aggregate()	{
		for(String[] line : rcsv.csvData) {
			if(Integer.parseInt(line[col]) > max)
				max = Integer.parseInt(line[col]);
			else if(Integer.parseInt(line[col]) < min)
				min = Integer.parseInt(line[col]);
		}
		System.out.println("Max : " + max);
		System.out.println("Min : " + min);
	}
	void discretize()	{
		int mean = (min + max) / 2;
		int mid1 = (min + mean) / 2;
		int mid2 = (mean + max) / 2;
		for(String[] line: rcsv.csvData) {
			for(int i=0;i<line.length; i++)	{
				if(i == col){
					int val = Integer.parseInt(line[i]);
					if(val>= min && val<mid1)
```

```
                                        System.out.print(min + "-" + mid1 + " ");
                                else if(val>=mid1 && val < mean)
                                        System.out.print(mid1 + "-" + mean + " ");
                                else if(val>=mean && val < mid2)
                                        System.out.print(mean + "-" + mid2 + " ");
                                else if(val>=mid2 && val <=max)
                                        System.out.print(mid2 + "-" + max + " ");
                        }
                        else
                                System.out.print(line[i] + " ");
                }
                System.out.println();
        }
    }
    public static void main(String[] args) throws IOException {
        System.out.println("Enter file Name: ");
        AggregateAndDiscretize s = new AggregateAndDiscretize(sc.next());
    }
}
```

ReadCSV.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class ReadCsv {
    String fileName;
    String dilimiter;
    ArrayList<String[]> csvData;
    BufferedReader br;
    ReadCsv     (String fileName,String dilimiter) throws IOException{
        this.fileName = fileName;
        this.dilimiter = dilimiter;
        csvData = new ArrayList<>();
        br = new BufferedReader(new FileReader(new File(this.fileName)));
        String line;
        while ((line = br.readLine()) !=null) {
            String[] singleLine = line.split(this.dilimiter);
            csvData.add(singleLine);
        }
    }
    void printCsv(){
        for (String[] line:csvData){
            System.out.print("len: "+ line.length + "\t");
```

```
                for(String word:line)      {
                        System.out.print(word + "\t");
                }
                System.out.println();
        }
    }
}
```

---

2. Write a java/c++ program to generate a stratified sample from a given supervised dataset.

StratifiedSample.java

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Random;
import java.util.Scanner;

public class StratifiedSample {

    HashMap<String,ArrayList<String[]>> classes;
    ReadCsv rcsv;
    StratifiedSample(String fileName) throws IOException  {
        rcsv = new ReadCsv(fileName, ","); // data format - usn,name,marks
        classes = new HashMap<>();
        classes.put("S",new ArrayList<>());
        classes.put("A",new ArrayList<>());
        classes.put("B",new ArrayList<>());
        classes.put("C",new ArrayList<>());
        classes.put("D",new ArrayList<>());
        classes.put("E",new ArrayList<>());
        classes.put("F",new ArrayList<>());
        this.stratifyData();
        this.selectRandomSample();

    }
    void stratifyData() {
        for(String[] line: rcsv.csvData){
            double val = Double.parseDouble(line[2]);
            String cl;
            if(val <= 100 && val >= 90)
                cl = "S";
            else if(val >=75 && val < 90)
                cl = "A";
            else if(val >=60 && val < 75)
                cl = "B";
```

```
                else if(val >=45 && val <60)
                        cl = "C";
                else if(val >=35 && val <45)
                        cl = "D";
                else
                        cl = "F";
                classes.get(cl).add(line);
        }
    }
    void selectRandomSample(){
        Random r = new Random();
        for(String key: classes.keySet()) {
            System.out.println("Class: " + key);
            if(classes.get(key).isEmpty())
                continue;
            String[] sample = classes.get(key).get(r.nextInt(classes.get(key).size()));
            for(String s: sample)
                System.out.print(s + " ");
            System.out.println();
        }
    }
    public static void main(String[] args) throws IOException {
        System.out.println("Enter file Name: ");
        Scanner sc = new Scanner(System.in);
        StratifiedSample s = new StratifiedSample(sc.next());
        sc.close();
    }

}
```

ReadCsv.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

public class ReadCsv {
    String fileName;
    String dilimiter;
    ArrayList<String[]> csvData;
    BufferedReader br;
    ReadCsv     (String fileName,String dilimiter) throws IOException{
        this.fileName = fileName;
        this.dilimiter = dilimiter;
        csvData = new ArrayList<>();
        br = new BufferedReader(new FileReader(new File(this.fileName)));
        String line;
```

```
		while ((line = br.readLine()) !=null) {
			String[] singleLine = line.split(this.dilimiter);
			csvData.add(singleLine);
		}
	}
	void printCsv(){
		for (String[] line:csvData){
			System.out.print("len: "+ line.length + "\t");
			for(String word:line)	{
				System.out.print(word + "\t");
			}
			System.out.println();
		}
	}
}
```

---

3. Write a java/c++ program to handle missing values Replacing by the mean for numeric attributes. Use majority voting to replace for categorical attributes.

ReplaceMissingValues.java

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

public class ReplaceMissingValues {
	static Scanner sp = new Scanner(System.in);

	void menu(ReadCsv rc) throws Exception  {
		System.out.println("Choice\n1.Avg of columns\n2.Mode of Column\n3.Print
CSV\n4.Exit");
		int op = sp.nextInt();
		switch(op)	{
			case 1:
			case 2:
				System.out.println("Enter the Column to replace");
				rc.replaceMissingValue(op, sp.nextInt());
				break;
			case 3:
				rc.printCsv();
				break;
			case 4:
				throw new Exception();
			default:
				System.out.println("Wrong choice");
		}
```

```
	}
	public static void main(String[] args) {
		System.out.println("Enter CSV file name: ");
		String name = sp.next();
		System.out.println("Enter the delimiter: ");
		String delim = sp.next();
		ReadCsv rc=null;
		try {
			rc = new ReadCsv(name,delim);
		} catch (IOException e) {
			System.out.println("File Not Found");
		}
		ReplaceMissingValues rmv = new ReplaceMissingValues();
		while(true){
			try {
				rmv.menu(rc);
			} catch (Exception e) {
				return;
			}
		}
	}
}
```

ReadCsv.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

public class ReadCsv {
	String fileName;
	String dilimiter;
	ArrayList<String[]> csvData;
	BufferedReader br;
	ReadCsv	(String fileName,String dilimiter) throws IOException{
		this.fileName = fileName;
		this.dilimiter = dilimiter;
		csvData = new ArrayList<>();
		br = new BufferedReader(new FileReader(new File(this.fileName)));
		String line;
		while ((line = br.readLine()) !=null) {
			String[] singleLine = line.split(this.dilimiter);
			csvData.add(singleLine);
		}
	}
```

```
void printCsv(){
	for (String[] line:csvData){
		System.out.print("len: "+ line.length + "\t");
		for(String word:line)	{
			System.out.print(word + "\t");
		}
		System.out.println();
	}
}
String avgOfCol(int colNumber)throws NumberFormatException{
	double total = 0.0,avg = 0.0;
	int numberOfValues = 0;
	for(String[] line:csvData){
		if(colNumber<line.length && !line[colNumber].isEmpty()){
			double colVal = Double.parseDouble(line[colNumber]);
			total += colVal;
			numberOfValues++;
		}
	}
	avg = total/numberOfValues;
	return ""+avg;
}
String modeOfCol(int colNumber)  {
	HashMap<String,Integer> wordFreq = new HashMap<>();
	for(String[] line:csvData){

		if(colNumber<line.length && !line[colNumber].isEmpty()){
			if(wordFreq.containsKey(line[colNumber])){
				int x = wordFreq.get(line[colNumber]);
				wordFreq.put(line[colNumber], ++x);
			}else{
				wordFreq.put(line[colNumber], 1);
			}
		}
	}
	String mode = "";
	if(!wordFreq.isEmpty()){
		int maxFreq = 0;
		for(String keys:wordFreq.keySet()){
			if(wordFreq.get(keys) > maxFreq)  {
				mode = keys;
				maxFreq = wordFreq.get(keys);
			}
		}
	}
	return mode;
}
void replaceMissingValue(int type, int colNumber)	{
	String replaceValue = "";
```

```
                try{
                        replaceValue =(type == 1)?this.avgOfCol(colNumber):this.modeOfCol(colNumber);
                }catch(NumberFormatException e){
                        System.out.println("Not numeric");
                }
                for(String[] line:csvData) {
                        if(colNumber < line.length && line[colNumber].isEmpty())     {
                                line[colNumber] = replaceValue;
                        }
                }
        }
}
```

---

4. Write a java/c++ program to identify strong rules from a set of rules given the confidence and support thresholds.

Generator.java

```
import java.util.ArrayList;
import java.util.Scanner;

public class Generator {
        ArrayList<Rule> rules;
        char itemset[];
        static Scanner sc = new Scanner(System.in);
        double thresholdSup, thresholdConf;
        Generator(String strItemSet, double thresholdSup, double thresholdConf)   {
                itemset = strItemSet.toCharArray();
                rules = new ArrayList<>();
                this.thresholdSup = thresholdSup;
                this.thresholdConf = thresholdConf;
                this.generateRules();
        }
        void generateRules(){
                //generating 1-subset
                for(int i=0; i<itemset.length; i++){
                        Rule temp = new Rule();
                        String lhs = "", rhs = "";
                        lhs += itemset[i];
                        for(int j=0; j<itemset.length; j++)      {
                                if(lhs.indexOf(itemset[j]) < 0){
                                        rhs += itemset[j];
                                }
                        }
                        temp.setLhs(lhs);
                        temp.setRhs(rhs);
```

```
                rules.add(temp);
        }
        //generating 2-subset
        for(int i=0; i<itemset.length; i++){
                for(int j=i+1; j<itemset.length; j++)    {
                        Rule temp = new Rule();
                        String lhs = "", rhs = "";
                        lhs += itemset[i];
                        lhs += itemset[j];
                        for(int k=0; k<itemset.length; k++){
                                if(lhs.indexOf(itemset[k]) < 0){
                                        rhs += itemset[k];
                                }
                        }
                        temp.setLhs(lhs);
                        temp.setRhs(rhs);
                        rules.add(temp);
                }
        }
        //generating 3-subset
        for(int i=0; i<itemset.length; i++){
                for(int j=i+1; j<itemset.length; j++)    {
                        for(int k=j+1; k<itemset.length; k++){
                                Rule temp = new Rule();
                                String lhs = "", rhs = "";
                                lhs += itemset[i];
                                lhs += itemset[j];
                                lhs += itemset[k];
                                for(int l=0;l<itemset.length; l++){
                                        if(lhs.indexOf(itemset[l]) < 0){
                                                rhs += itemset[l];
                                        }
                                }
                                temp.setLhs(lhs);
                                temp.setRhs(rhs);
                                rules.add(temp);
                        }

                }
        }
}
void printRules()    {
        for(Rule r: rules){
                r.printRule();
                System.out.println();
        }
}
void setSupAndConf(){
        System.out.println("\nEnter support and confidence for each rule: \n");
```

```
            for(Rule r: rules){
                  r.printRule();
                  System.out.print("   : ");
                  double sup = sc.nextDouble();
                  double conf = sc.nextDouble();
                  r.setConfSup(sup, conf);
            }
      }
      void printStrongRules(){
            for(Rule r: rules){
                  if(r.isStrongRule(thresholdSup, thresholdConf)){
                        r.printRule();
                        System.out.println();
                  }
            }
      }
      public static void main(String []args)  {
            Generator g = new Generator("abcd", 0.5, 0.7);
            System.out.println("Generated Rules: " );
            g.printRules();
            g.setSupAndConf();
            System.out.println("Generated Frequent Rules: " );
            g.printStrongRules();
      }
}
```

Rule.java

```
import java.util.ArrayList;

public class Rule {
      double support;
      double confidence;
      ArrayList<Character> lhs;
      ArrayList<Character> rhs;
      Rule() {
            lhs = new ArrayList<>();
            rhs = new ArrayList<>();
      }
      void setLhs(String strLhs) {
            for(char s: strLhs.toCharArray()) {
                  lhs.add(s);
            }
      }
      void setRhs(String strRhs) {
            for(char s: strRhs.toCharArray()) {
                  rhs.add(s);
            }
```

```
	}
	void setConfSup(double support, double confidence){
		this.support = support;
		this.confidence = confidence;
	}
	boolean isStrongRule(double thresholdSup, double thresholdConf)	{
		if(support >= thresholdSup && confidence >= thresholdConf){
			return true;
		}
		return false;
	}
	void printRule(){
		for(char s: lhs)
			System.out.print(s + " ");
		System.out.print(" -> ");
		for(char s: rhs)
			System.out.print(s + " ");
	}
}
```

5.Write a java/c++ program to implement the information gain and gini index measures to identify the best attribute to split.

BestSplit.java

```
import java.io.IOException;
import java.util.ArrayList;

public class BestSplit {
	ArrayList<Record> records;
	int totalOfCol[];
	double entropyOfCol[], giniOfCol[];
	double entropy[], gini[], gain[];
	BestSplit() throws IOException{
		ReadDataset rd = new ReadDataset("data.csv",",");
		records = rd.getRecords();
		totalOfCol = new int[7];
		entropyOfCol = new double[7];
		giniOfCol = new double[7];
		entropy = new double[3];
		gini = new double[3];
```

```
		gain = new double[3];
	}
	void calculateTotalOfCols()	{
		for(int i=0; i<records.size(); i++){
			for(int j=0; j<totalOfCol.length; j++){
				totalOfCol[j] += records.get(i).attr[j];
			}
		}
	}
	void calculateEntGini()	{
		for(int i = 0; i<records.size(); i++){
			for(int j=0; j<totalOfCol.length; j++){
				double temp = records.get(i).attr[j]/(totalOfCol[j]*1.0);
				giniOfCol[j] += Math.pow(temp, 2);
				entropyOfCol[j] += temp * Math.log(temp)/ Math.log(2);
			}
		}
		for(int i=0; i<totalOfCol.length; i++){
			giniOfCol[i] = 1 - giniOfCol[i];
			entropyOfCol[i] = -1 * entropyOfCol[i];
		}
		gini[0] = (giniOfCol[0] * totalOfCol[0] / (totalOfCol[0] + totalOfCol[1])) +
(giniOfCol[1] * totalOfCol[1]/ (totalOfCol[0] + totalOfCol[1]));
		gini[1] = (giniOfCol[2] * totalOfCol[2] / (totalOfCol[2] + totalOfCol[4])) +
(giniOfCol[3] * totalOfCol[3]/ (totalOfCol[2] + totalOfCol[3]));
		gini[2] = (giniOfCol[4] * totalOfCol[4] / (totalOfCol[4] + totalOfCol[5] +
totalOfCol[6])) + (giniOfCol[5] * totalOfCol[5]/ (totalOfCol[4] + totalOfCol[5] + totalOfCol[6]))
+ (giniOfCol[6] * totalOfCol[6]/ (totalOfCol[4] + totalOfCol[5] + totalOfCol[6]));
		entropy[0] = (entropyOfCol[0] * totalOfCol[0] / (totalOfCol[0] + totalOfCol[1])) +
(entropyOfCol[1] * totalOfCol[1]/ (totalOfCol[0] + totalOfCol[1]));
		entropy[1] = (entropyOfCol[2] * totalOfCol[2] / (totalOfCol[2] + totalOfCol[4])) +
(entropyOfCol[3] * totalOfCol[3]/ (totalOfCol[2] + totalOfCol[3]));
		entropy[2] = (entropyOfCol[4] * totalOfCol[4] / (totalOfCol[4] + totalOfCol[5] +
totalOfCol[6])) + (entropyOfCol[5] * totalOfCol[5]/ (totalOfCol[4] + totalOfCol[5] +
totalOfCol[6])) + (entropyOfCol[6] * totalOfCol[6]/ (totalOfCol[4] + totalOfCol[5] +
totalOfCol[6]));

		double
parentEntropy=-1*((0.5*(Math.log(0.5)/Math.log(2)))+(0.5*(Math.log(0.5)/Math.log(2))));
```

```
        double parentGini=1-(0.5*0.5)-(0.5*0.5);

        for(int i=0; i<gini.length; i++){
           gain[i] = parentEntropy - entropy[i];
        }

        System.out.println("Parent:");
        System.out.println("Entropy: " + parentEntropy);
        System.out.println("Gini: " + parentGini);

        System.out.println("\nNumerical Attributes: ");
        System.out.println("Entropy: " + entropy[0]);
        System.out.println("Gain: " + gain[0]);
        System.out.println("Gini: " + gini[0]);

        System.out.println("\nBinary Attributes: ");
        System.out.println("Entropy: " + entropy[1]);
        System.out.println("Gain: " + gain[1]);
        System.out.println("Gini: " + gini[1]);

        System.out.println("\nMultiway split Attributes: ");
        System.out.println("Entropy: " + entropy[2]);
        System.out.println("Gain: " + gain[2]);
        System.out.println("Gini: " + gini[2]);
    }
    public static void main(String[]args) throws IOException    {
           BestSplit bs = new BestSplit();
           bs.calculateTotalOfCols();
           bs.calculateEntGini();
    }
}
```

ReadDataset.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
```

```
public class ReadDataset {
	String fileName;
	String dilimiter;
	ArrayList<Record> records;
	BufferedReader br;
	ReadDataset  (String fileName,String dilimiter) throws IOException{
		this.fileName = fileName;
		this.dilimiter = dilimiter;
		records = new ArrayList<>();
		br = new BufferedReader(new FileReader(new File(this.fileName)));
		String line;
		while ((line = br.readLine()) !=null) {
			String[] singleLine = line.split(this.dilimiter);
			records.add(new
Record(Integer.parseInt(singleLine[0]),Integer.parseInt(singleLine[1]),Integer.parseInt(singleLine
[2]),Integer.parseInt(singleLine[3]),Integer.parseInt(singleLine[4]),Integer.parseInt(singleLine[5
]),Integer.parseInt(singleLine[6])));
		}
	}
	ArrayList<Record> getRecords(){
		return records;
	}
}
```

Record.java

```
public class Record {
	int attr[];
	Record(int attr1,int attr2,int attr3,int attr4,int attr5,int attr6,int attr7){
		attr = new int[7];
		this.attr[0] = attr1;
		this.attr[1] = attr2;
		this.attr[2] = attr3;
		this.attr[3] = attr4;
		this.attr[4] = attr5;
		this.attr[5] = attr6;
		this.attr[6] = attr7;
	}
}
```

6.Write a java/c++ program to construct a Naïve Bayesian classifier for a given dataset.

Classifier.java

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Classifier {
	static Scanner sc = new Scanner(System.in);
	ArrayList<Record> records;
	Class c[];
	Classifier() throws NumberFormatException, IOException{
		System.out.println("Enter datafile name");
		ReadDataset rd = new ReadDataset("data.csv",",");
		records = rd.getRecords();
		c = new Class[2];
		c[0] = new Class();
		c[1] = new Class();
		classifyData();
	}
	void classifyData(){
		for(Record r: records)	{
			c[r.classID].addRecord(r);
		}
		c[0].calculateMean();
		c[0].calculateVariance();
		c[1].calculateMean();
		c[1].calculateVariance();
		c[0].calculateProbablity();
		c[1].calculateProbablity();
	}
	void continousValues()	{
		System.out.println("Numerical Values: ");
		for(int i=0; i<c.length; i++){
			System.out.println("Class: " + (i));
```

```
                System.out.println("Mean: " + c[i].getMean());
                System.out.println("Variance: "+ c[i].getVariance());
                System.out.println();
            }
        }
        void catagoricalValues()   {
            System.out.println("Catogrical Values: ");
            for(int i=0; i<c.length; i++)    {
                for(String s: c[i].probability.keySet()){
                    System.out.println("Class:  "  +  i  +  "  Value:  "  +  s  +  "  =  "  +
c[i].probability.get(s));
                }
            }
        }
        void classifyNewRecord()   {
            System.out.print("Enter value: ");
            double attr1 = sc.nextDouble();
            System.out.print("Enter type: ");
            String attr2 = sc.next();
            double newProb[] = new double[2];
            for(int i=0; i<c.length; i++)    {
                double exp = Math.pow((attr1 - c[i].mean), 2)/(2 *c[i].variance);
                double     prob     =     Math.exp(-1*exp)/(Math.sqrt(2     *     3.14     *
Math.sqrt(c[i].variance)));
                newProb[i] = prob * c[i].getProbablity(attr2);
                System.out.println("Probability of class: "+ i + " = " + newProb[i]);
            }
            System.out.println("New  record  belongs  to  class  " + ((newProb[0] > newProb[1]) ?
0:1));
        }
        public static void main(String[]args)   {
            try {
                Classifier cl = new Classifier();
                cl.continousValues();
                cl.catagoricalValues();
                cl.classifyNewRecord();

            } catch (NumberFormatException | IOException e) {
                e.printStackTrace();
```

```
			}


		}
}
```

ReadDataset.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;


public class ReadDataset {
	ArrayList<Record> records;
	String filename;
	String dilimiter;
	ReadDataset(String filename ,String dilimiter) throws NumberFormatException, IOException
	{
		this.filename = filename;
		this.dilimiter = dilimiter;
		records = new ArrayList<>();
		BufferedReader br = new BufferedReader(new FileReader(new File(this.filename)));
		String line;
		while ((line = br.readLine()) !=null) {
			String[] singleLine = line.split(this.dilimiter);
			records.add(new
Record(Double.parseDouble(singleLine[0]),singleLine[1],Integer.parseInt(singleLine[2])));
		}
		br.close();
	}
	void printDataset() {
		System.out.println("Dataset:-");
		for(Record r: records)
			r.printRecord();
	}
	ArrayList<Record> getRecords()	{
		return records;
	}
}
```

Class.java

```
import java.util.ArrayList;
import java.util.HashMap;
```

```
public class Class {
	ArrayList<Record> records;
	HashMap<String, Double> probability;
	double mean;
	double variance;
	Class()	{
		records = new ArrayList<>();
		probability = new HashMap<>();
		mean = 0.0;
		variance = 0.0;
	}
	void addRecord(Record r){
		records.add(r);
	}
	void calculateMean(){
		double sum=0;
		for(int i=0; i<records.size(); i++){
			sum+=records.get(i).attr1;
		}
		mean = sum/(records.size() * 1.0);
	}
	void calculateVariance()   {
		double sum=0;
		for(int i=0; i<records.size(); i++){
			sum += Math.pow(records.get(i).attr1 - mean, 2);
		}
		variance = sum / (records.size() * (records.size() - 1) *1.0);
	}
	void calculateProbablity() {
		HashMap<String,Integer> freq = new HashMap<>();
		for(Record r: records){
			int val = 1;
			if(freq.containsKey(r.attr2))
				val += freq.get(r.attr2);
			freq.put(r.attr2, val);
		}
		for(String s: freq.keySet()){
			probability.put(s, (double) (freq.get(s)/(records.size() * 1.0)));
		}
	}
	double getProbablity(String type) {
		if(probability.containsKey(type)) {
			return probability.get(type);
		}
		return 0.0;
	}
	double getMean(){
		return mean;
	}
```

```
	double getVariance(){
		return variance;
	}
}
```

Record.java

```
public class Record {
	String attr2;
	double attr1;
	int classID;
	Record(double attr1, String attr2, int classID){
		this.attr1 = attr1;
		this.attr2 = attr2;
		this.classID = classID;
	}
	void printRecord(){
		System.out.println(this.attr1 + "\t" + this.attr2 + "\t" + this.classID);
	}
}
```

7.Write a java/c++ program to perform k-means clustering on numeric dataset.

kMeans.java

```
import java.io.IOException;
import java.util.ArrayList;

public class kMeans {
	ArrayList<Record>records;
	Centroid C[];
	kMeans() throws IOException	{
		System.out.println("Enter filename: ");
		ReadDataset rd = new ReadDataset("data.csv",",");
		records = rd.getRecords();
		C = new Centroid[3];
		initCluster();
	}
	void initCluster()  {
		C[0] = new Centroid(records.get(0).attr1,records.get(0).attr2);
		C[1] = new Centroid(records.get(3).attr1,records.get(3).attr2);
		C[2] = new Centroid(records.get(6).attr1,records.get(6).attr2);
	}
```

```
	Centroid[] generateCluster()	{		//generates new cluster and returns new centroids
		System.out.println("\nGenerating Cluster");
		int numberOfRecordsInCluster[] = new int[3];
		int newCentroidValue[][] = new int[3][2];
		for(Record r: records)	{
			r.calculateDistFromCentroid(C);
			numberOfRecordsInCluster[r.cluster]++;
			newCentroidValue[r.cluster][0] += r.attr1;
			newCentroidValue[r.cluster][1] += r.attr2;
			r.printDist();
		}
		Centroid newCentroids[] = new Centroid[3];
		for(int i=0; i<3; i++){
			newCentroids[i] = new Centroid(newCentroidValue[i][0] / (numberOfRecordsInCluster[i]*1.0) , newCentroidValue[i][1] / (numberOfRecordsInCluster[i]*1.0));
		}
		System.out.println("Counts:- ");
		for(int i=0; i<newCentroids.length; i++){
			System.out.println(i + "->" +numberOfRecordsInCluster[i]);
		}
		System.out.println("Cntroids:- ");
		for(int i=0; i<newCentroids.length; i++){
			newCentroids[i].printCentroid();
		}
		return newCentroids;
	}
	public static void main(String[] args) throws IOException	{
		kMeans km = new kMeans();
		while(true){
			Centroid newCent[] = km.generateCluster();
			if(Centroid.compareCentroid(km.C, newCent))	{
				break;
			}
			km.C=newCent;
		}

	}
}
```

ReadDataset.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
```

```
public class ReadDataset {
	String fileName;
	String dilimiter;
	ArrayList<Record> records;
	BufferedReader br;
	ReadDataset  (String fileName,String dilimiter) throws IOException{
		this.fileName = fileName;
		this.dilimiter = dilimiter;
		records = new ArrayList<>();
		br = new BufferedReader(new FileReader(new File(this.fileName)));
		String line;
		while ((line = br.readLine()) !=null) {
			String[] singleLine = line.split(this.dilimiter);
			records.add(new                    Record(Double.parseDouble(singleLine[0]),
Double.parseDouble(singleLine[1])));
		}
	}
	ArrayList<Record> getRecords(){
		return records;
	}
}
```

Record.java

```
public class Record {
	double attr1;
	double attr2;
	double distFromCentroid[];
	int cluster;
	Record(double attr1, double attr2)	{
		this.attr1 = attr1;
		this.attr2 = attr2;
		distFromCentroid = new double[3];
		cluster = -1;
	}
	void calculateDistFromCentroid(Centroid centroid[]){
		for(int i=0; i<centroid.length; i++){
			distFromCentroid[i] = Math.sqrt(Math.pow((centroid[i].attr1 - this.attr1), 2)
+ Math.pow(centroid[i].attr2 - this.attr2, 2));
		}
		double min = distFromCentroid[0];
		int ind = 0;
		for(int i=1;i<distFromCentroid.length; i++){
			if(distFromCentroid[i] < min){
				min = distFromCentroid[i];
				ind = i;
			}
		}
```

```
                cluster = ind;
        }
        void printDist(){
                System.out.printf("%.2f\t%.2f\t%.2f\t%d\n",
distFromCentroid[0],distFromCentroid[1],distFromCentroid[2] ,cluster);
        }
}
```

Centroid.java

```
public class Centroid {
        double attr1;
        double attr2;
        Centroid(double attr1, double attr2){
                this.attr1 = attr1;
                this.attr2 = attr2;
        }
        static boolean compareCentroid(Centroid[] C1, Centroid[] C2){
                for(int i=0; i<C1.length; i++){
                        if(C1[i].attr1 != C2[i].attr1 || C1[i].attr2 != C2[i].attr2)
                                return false;
                }
                return true;
        }
        void printCentroid(){
                System.out.println("(" + attr1 + "," + attr2 + ")");
        }
}
```
