Walchand College of Engineering, Sangli
Information Technology
Data Mining Lab

Experiment No: 1

Study and use of different types of graphs

Walchand College of Engineering, Sangli.

Experiment No: 1

Aim :- To study and use the different types of graphs and charts in MS-XLS

objective :- To analyze and study data using pictorial representation.

Theory :- There are various types of graphs and charts in data mining.

(1) Bar Graph :-
Bar graph/charts are among the most chart types as the name suggets a bar chart is composed of series of bars.

(2) pie chart :-
A pie chart is circular graph that shows the proportional of categories based on size of each slide represents 100% and a portion of a whole.

(3) Line Graph :-
A Line graph is graphical represent^n of data that is plotted using several lines. Line graphs display lines across the chart horizontally with value axis

Page No.

(4) Area chart:

A area chart is based on a line chart but with the area between the line ~~across~~ and axis filled in. Types of area chart are regular area chart, stacked area chart and 100% stacked area chart.

(5) Scatter plot:

A scatter plot is a type of chart that is often used in fields of statiscs and data science. It is used to find out the relationship between any two variables. It provides the data to confirm a hypothesis that two variables are related.

Algorithm: (Eicher motors shareholding dataset)

Step1: Create % of shareholdings of respective type of investors.

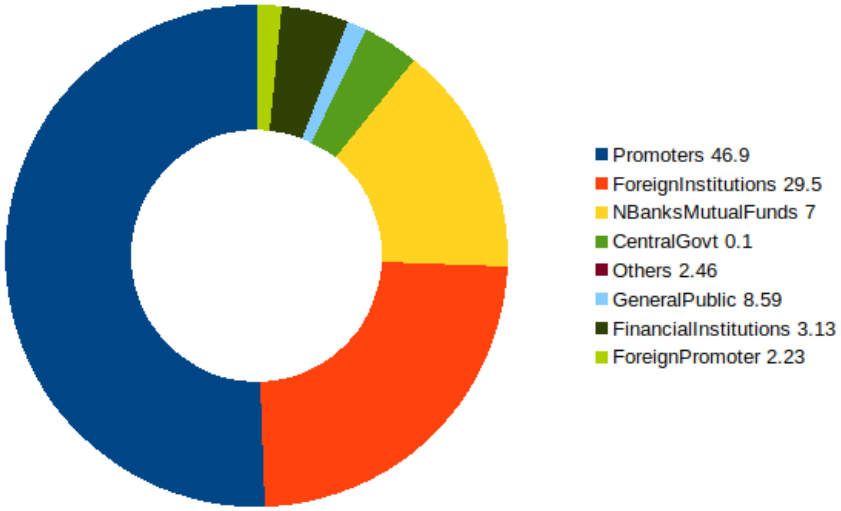
Step2: Mention no. of shares of respective ones.

Step3: using this attributes created charts like pie chart, col and line charts, Net chart.

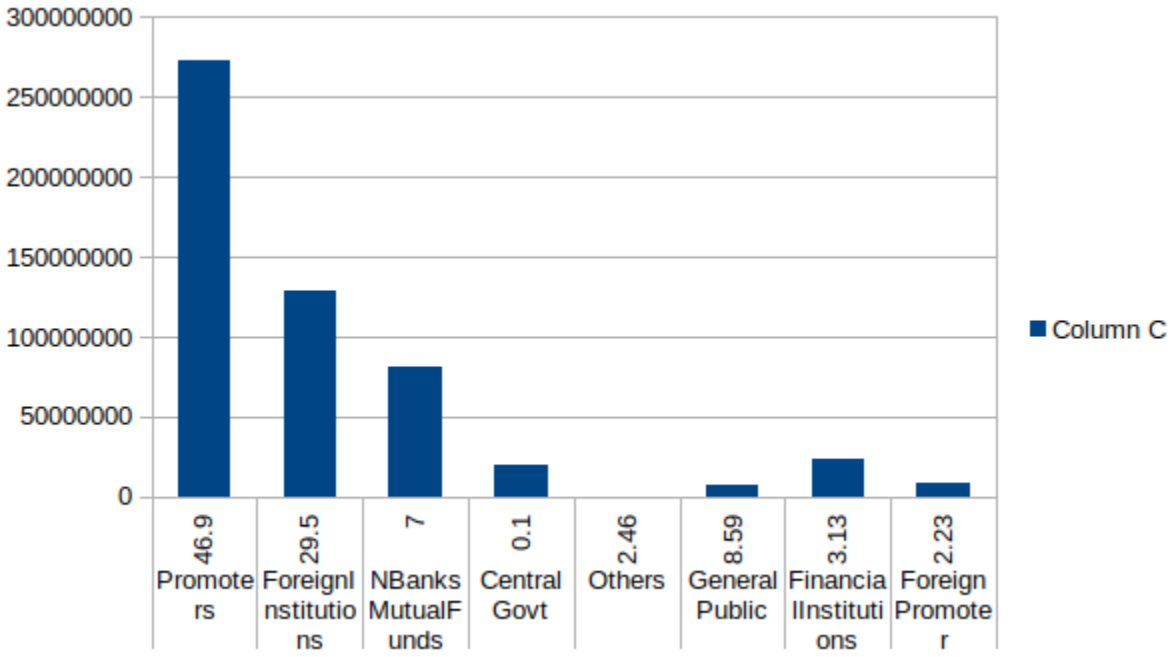
Dataset :-

Holders Name	% of shareholding	No. of shares
1. promoters	46.9	273428852
2. Foreign investors	29.5	128468207
3. NBank Mutual fund	07	80662396
4. CentralGovt	0.1	19131783
5. Others	2.46	260780
6. General public	8.59	6739571
7. Financial inst.	3.13	23499851
8. Foreign promoters	2.23	8569384

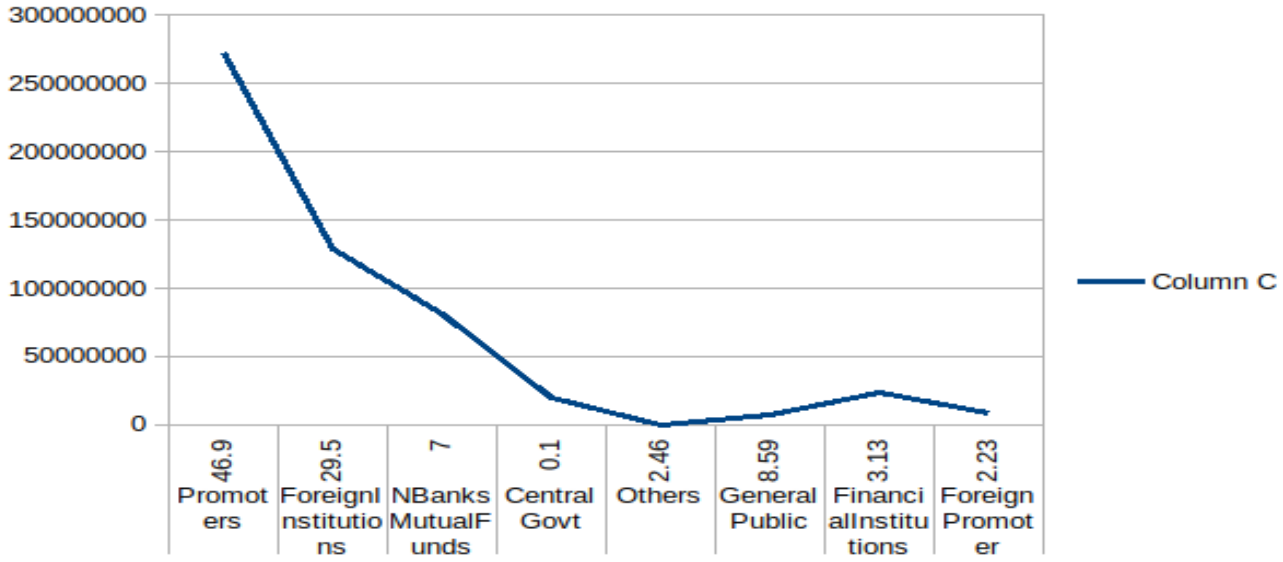
1. <u>Pie Chart:</u>



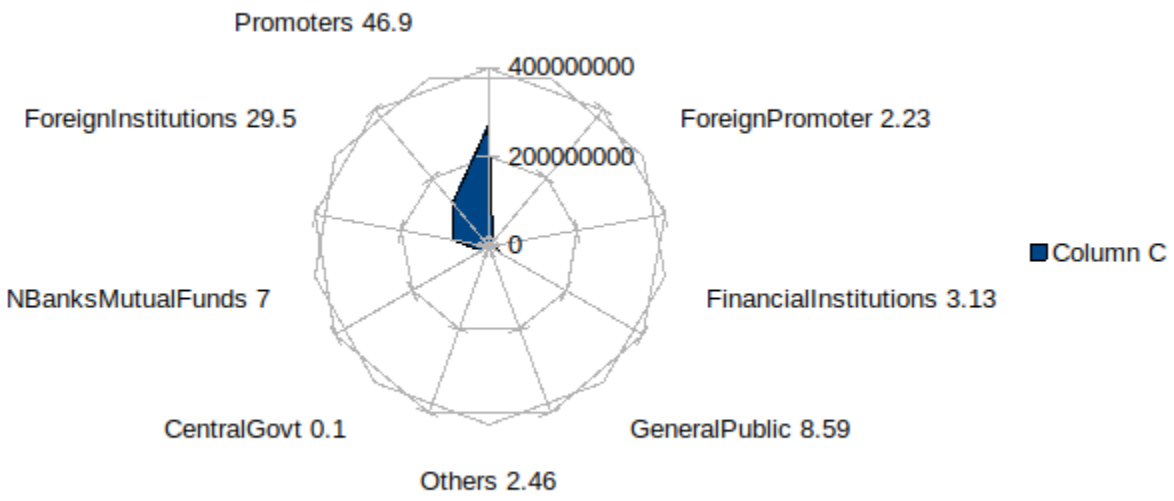
2. <u>Column and Line Chart:</u>



3. <u>Line Chart</u>



4. <u>Net Chart</u>



<u>Conclusion</u>: We studied how to create different graphs and its importance in understanding the dataset

Experiment No: 02

Perform Normalization of data (Min-max and Z-Score)

Walchand College of Engineering, Sangli.

Experiment No: 02

Aim :- To perform normalization of data using min-max and z-score transformation.

objective :-

objective :- use of min-max and z-score for normalizing large values for good data preprocessing.

Theory :-

Normalization is done generally when we are dealing with attributes on a different scale, otherwise it may lead to dilution in effectiveness of important equally attributes because of other attribute having values on large scale.

Min-Max Normalization

In this technique linear transformation is performed on original data. Minimum and maximum value is fetched and each value is replaced according to formula.

$$v' = \frac{v - min(A)}{max(A) - min(A)} \times (new_max(A) - new_min(A)) + new_min(A)$$

Example :-

marks	
8	min - the min value of given attribute is 8
10	max - the max value of given attribute is 20
15	
20	V - It is respective value of attribute $V_1 = 18$ $V_2 = 10$ $V_3 = 15$ $V_4 = 20$

new-max = 1 new-min = 0

For marks 8 :- $\text{min-max} = \frac{(8-8)}{(20-8)} \times (1-0) + 0$

$= 0$

For marks 10 :- $\text{min-max} = \frac{(10-8)}{(20-8)} \times (1-0) + 0$

$= 0.16$

For marks 15 :- $\text{min-max} = \frac{(15-8)}{(20-8)} \times (1-0) + 0$

$= 0.58$

For marks 20 :- $\text{min-max} = \frac{(20-8)}{(20-8)} \times (1-0) + 0$

$= 1$

Marks	Marks after normalizⁿ
8	0.00
10	0.16
15	0.58
20	1

Algorithm for z-score

Step1 :- Calculate the mean of attribute set
Step2 :- Calculate standard deviation of attribute se
Step3 :- Now find z-score value for each individual x value.
Step4 :- formula for z-score value for given element.

Knime for minmax

marks
0
0.167
0.583
1

Knime for z-score

marks
-0.976
-0.604
0.325
1.255

Weka for minmax

	marks	
Min	~~140~~	8
Max	~~182~~	20
Mean	~~168~~	13.25
stdDev	~~16.193~~	5.377

Weka for z-score

	marks	
Min	~~140~~	8
Max	~~182~~	20
Mean	~~168~~	13.25
stdDev	~~16.193~~	5.377

Conclusion :-
- The transformed values obtained by calculation & by Knime and weka are identical
- Successfully performed transformation of given data using min-max & z-score normalization.

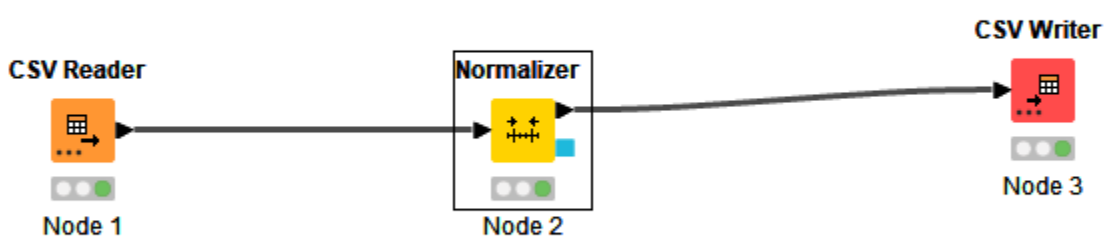
Program for Min-Max

```
// Min-Max Normalization on a Data Set
import java.util.*;
class minmax{
 public static void main(String args[]){
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter the size of the Data Set: ");
   int size = sc.nextInt();
   System.out.println("Enter the values of the Data Set: ");
   float set[] = new float[size];
   for(int i = 0; i<size; i++) set[i] = sc.nextFloat();
   System.out.println("Enter the new minimum: ");
   int new_min = sc.nextInt();
   System.out.println("Enter the new maximum: ");
   int new_max = sc.nextInt();

   normalize(new_min, new_max, set);
 }
  // Method to find minimum value from the data set
 static float minValue(float arr[]){
   float min = arr[0];
   for(int i = 1; i<arr.length; i++)
     if(arr[i]<min) min = arr[i];
   return min;
 }
  // Method to find maximum value from the data set
 static float maxValue(float arr[]){
   float max = arr[0];
   for(int i = 1; i<arr.length; i++)
     if(arr[i]>max) max = arr[i];
   return max;
 }
  // Method to find the normalized values of the data set
 static void normalize(int new_min, int new_max, float arr[]){
   System.out.println("The Data Set after Normalization: ");
   for(int i = 0 ; i<arr.length ; i++){
     float v = ((arr[i] - minValue(arr))/(maxValue(arr) - minValue(arr)))*(new_max - new_min) + new_min;
     System.out.printf("%.3f ",v);
   }}}
```

Output:

```
shubham@shubham-Vostro-3405:~/Desk
shubham@shubham-Vostro-3405:~/Desk
Enter the size of the Data Set:
4
Enter the values of the Data Set:
8
10
15
20
Enter the new minimum:
0
Enter the new maximum:
1
The Data Set after Normalization:
0.000 0.167 0.583 1.000 shubham@sh
```

Knime:



File Table - 3:1 - CSV Reader

File Edit Hilite Navigation View

Table "default" - Rows: 4 | Spec - Columns: 2 | Propertie

Row ID	S Subjects	I Marks
Row0	Java	8
Row1	Cpp	10
Row2	Dsa	15
Row3	IPR	20

Normalized table - 3:2 - Normalizer

File Edit Hilite Navigation View

Table "default" - Rows: 4 | Spec - Columns: 2 | Properties | Flow Variables

Row ID	S Subjects	D Marks
Row0	Java	0
Row1	Cpp	0.167
Row2	Dsa	0.583
Row3	IPR	1

Program for ZScore:

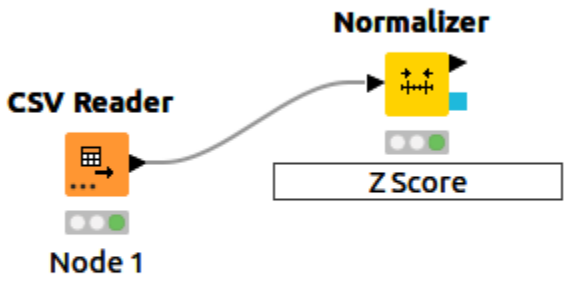
```
import java.io.*;
import java.util.*;

public class ZScore
{
	public static void main(String args[])
	{
		double mean = 0.0;
		double sum = 0.0;
		double sd = 0.0;

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter dataset size");
		int n = sc.nextInt();
		int[] myarr = new int[n];
		System.out.println("Enter values ");
		for(int i = 0 ; i < n ; i++){

			myarr[i] =  sc.nextInt();


			sum = sum + myarr[i];

		}
		// mean calculation
		mean = sum/n;





		// Standard Deviation calculation
		sum = 0.0;
		for(int i = 0 ; i < n ; i++){

			sum = sum + Math.pow(myarr[i]-mean, 2);

		}

		sd = Math.sqrt(sum/4);
```

```
        // Z-Score Calculation

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

            System.out.println("Z Score for "+ myarr[i] + " : " +
String.format("%.2f",((myarr[i]-mean)/sd)));

        }



    }
}
```

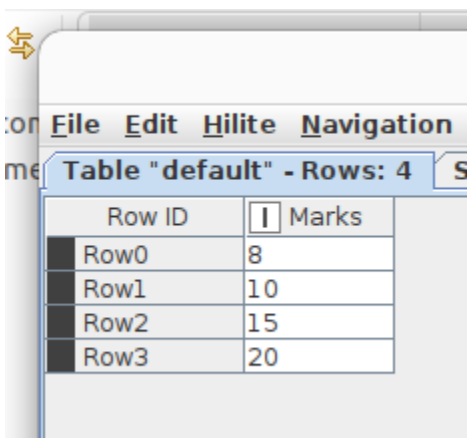
Output:

```
^Cshubham@shubham-Vostro-3405:~/Des
Enter dataset size
4
Enter values
8
10
15
20
Z Score for 8 : -1.13
Z Score for 10 : -0.70
Z Score for 15 : 0.38
Z Score for 20 : 1.45
shubham@shubham-Vostro-3405:~/Deskt
```

Knime:

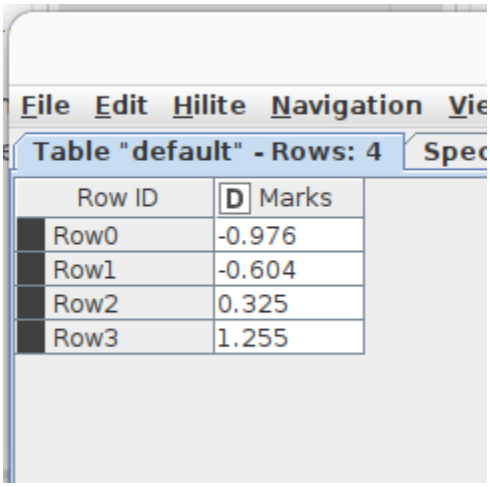


Input:



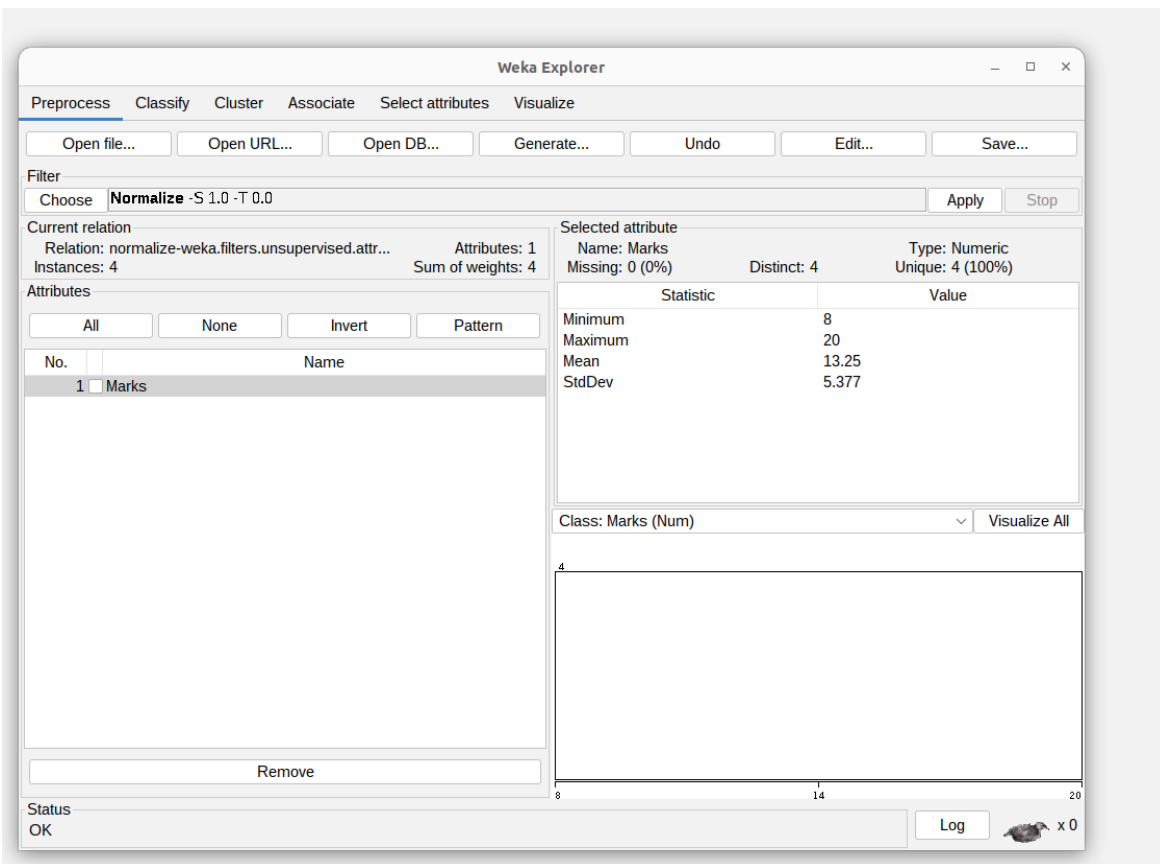
Row ID	Marks
Row0	8
Row1	10
Row2	15
Row3	20

Output:



Row ID	D Marks
Row0	-0.976
Row1	-0.604
Row2	0.325
Row3	1.255

Weka:



	Standard Marks	Standard extra
1		
2	0	0
3	0.166667	9
4	0.583333	2
5	1	2

Conclusion:

We studied what is MinMax, ZScore and importance of it and also implemented it successfully.

Experiment No: 03

Find info gain of the attribute from given dataset

Walchand College of Engineering, Sangli.

Experiment No: 3.

Aim :- To find info gain of an attribute from given data.

Theory :-

Entropy is the randomness of the data. Higher random values imply higher entropy. If the entropy of data is less, information gain will be more. The more the infor. gain the better the split.

Entropy of a dataset before and after a split is used to calculate information gain

$$\text{Entropy}(S) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Where, p_i is the probability of randomly selecting an example in class i.

Information gain :- It can be defined as measure of how much info. feature provides about a class. It helps to determine the order of attributes in the nodes of decision tree.

Gain = Eparent - Echildren

Where,

Eparent is entropy of parent node

Echildren is avg. entropy of child.

Algorithm :-

Step1 :- Read data from csv file.
Step2 : Defined function to calculate entropy
Step3 : Calculate frequency of distinct value in particular column
Step4 : Calculate probability of distinct values in particular column.
Step5 : Calculate entropy using following formula

$$E = -\sum_{i=1}^{N} P_i \log_2 P_i$$

Step6 : Similar manner, Calculate info gain and find the entropy of child column
Step7 : Return info gain.

values for apple-pie = 0.1

	10	midwest = 1	midwest = 0
0	2/0	1/2	1/2
1	8/10	6/8	2/8

values for sushi = 0.1

	10	midwest = 1	midwest = 0
0	6/10	5/6	1/6
1	4/10	2/4	2/4

Gain (sushi) = Entropy (midwest) - Entropy (midwest/sushi)

$$= 0.88 - \left[\frac{-4}{10} \left(\frac{2}{4} \log \frac{2}{4} + \frac{2}{4} \log \frac{2}{4} \right) - \frac{6}{10} \right.$$

$$= 0.88 - (0.4 + 0.39)$$

$$= 0.09$$

Info gain (apple-pie) = 0.03

Info gain (sushi) = 0.09

Info gain (potato-salad) = 0.19

Example :-
Data :-

0 = No
1 = yes.

	Age	apple-pie	potato-salad	Sushi	midwest
1	32	0	1	1	1
2	35	1	1	0	1
3	33	1	1	0	1
4	26	0	0	1	0
5	30	1	1	1	1
6	25	1	0	0	1
7	42	1	1	0	1
8	38	1	1	0	1
9	36	1	1	0	0
10	29	1	0	1	0

$$E(\text{midwest}) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

$$= -\left(\frac{7}{10}\log\frac{7}{10} + \frac{3}{10}\log\frac{3}{10}\right)$$

$$= -(-0.36 - 0.52)$$

$$= 0.88$$

$$\text{Gain}(\text{apple-pie}) = \text{Entropy}(\text{midwest}) - \text{Entropy}(\text{midwest}/\text{apple-pie})$$

$$= 0.88 - \left[\frac{8}{10}\left(\frac{6}{8}\log\frac{6}{8} + \frac{2}{8}\log\frac{2}{8}\right) - \frac{2}{10}\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)\right]$$

$$= 0.88 - (0.65 + 0.2)$$

$$= 0.03$$

Values

Knime:

RowID	Size	Entropy	Normalized Entropy	Quality
yes	8	0.811	0.811	?
no	2	1	1	?
Overall	10	0.849	0.849	0.151

Conclusion:

The calculated value of the infogain wealth attribute has higher gain making it a better splitting criteria.
Thus info gain can be used to choose the best splitting criteria among given attributes.

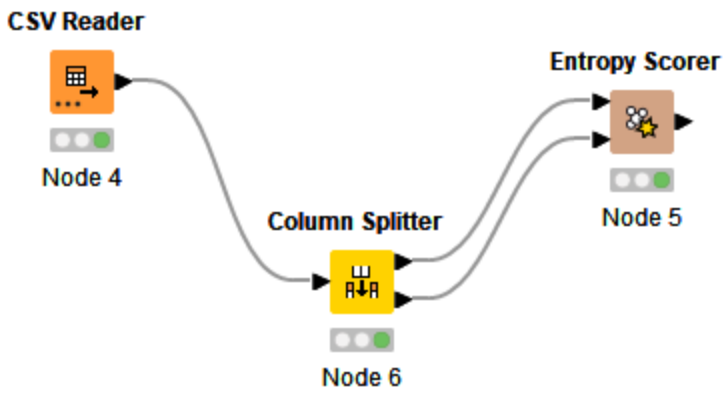
Program:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.*;
class InfoGain {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new FileReader("infogaindata.csv"));
        ArrayList<ArrayList<String>> data = new ArrayList<>();
        int n = br.readLine().trim().split(",").length;


        while (true) {
            String str[] = new String[n];
            try {
                str = br.readLine().trim().split(",");
            } catch (NullPointerException e) {
                break;
            }
            ArrayList<String> temp = new ArrayList<>();
            for (String s : str)
                temp.add(s);
            data.add(temp);
        }
        int posR = 0, negR = 0;
        for (int i = 0; i < data.size(); i++) {
            if (data.get(i).get(4).equalsIgnoreCase("Yes"))
                posR++;
            else
                negR++;
        }
        double totRecord = data.size();
        double entropy = -((1.0 * posR / totRecord) * (Math.log(1.0 * posR / totRecord)) / Math.log(2)) -
((1.0 * negR / totRecord) * (Math.log(1.0 * negR / totRecord) / Math.log(2)));
        System.out.println("Entropy of midwest: "+entropy);
        for(int i=1;i<data.get(0).size()-1;i++){
            System.out.println("Info Gain of column "+i+": "+findAllEntropy(i,entropy,data));
        }
    }

    private static double findAllEntropy(int i, double entropy, ArrayList<ArrayList<String>> data) {
        Set<String> attribute=new HashSet<>();
        for(ArrayList<String>x:data)
```

```
        attribute.add(x.get(i));
    Map<String,double[]>total=new HashMap<>();
    for(String x:attribute){
        total.put(x,new double[2]);
    }
    for(ArrayList<String> x:data){
        if(x.get(4).equalsIgnoreCase(("Yes")))
         total.get(x.get(i))[0]++;
        else
         total.get(x.get(i))[1]++;

    }
    double totalE=0.0;
    for(Map.Entry<String,double[]> x:total.entrySet()){
        double total1=x.getValue()[0]+x.getValue()[1];
        if(x.getValue()[0]==0||x.getValue()[1]==0)
            continue;
        double temp= -((1.0 * x.getValue()[0] / total1) * (Math.log(1.0 * x.getValue()[0] / total1)) /
Math.log(2)) - ((1.0 * x.getValue()[1]/ total1) * (Math.log(1.0 * x.getValue()[1] / total1) / Math.log(2)));
         totalE+=(total1/ data.size())*temp;
    }

    double finalE=entropy-totalE;
    return finalE;
  }
}
```

```
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$
Entropy of midwest: 0.8812908992306927
Info Gain of column 1: 0.03226839966338635
Info Gain of column 2: 0.19163120400671674
Info Gain of column 3: 0.09127744624168022
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$
```

<u>Knime:</u>



Input:

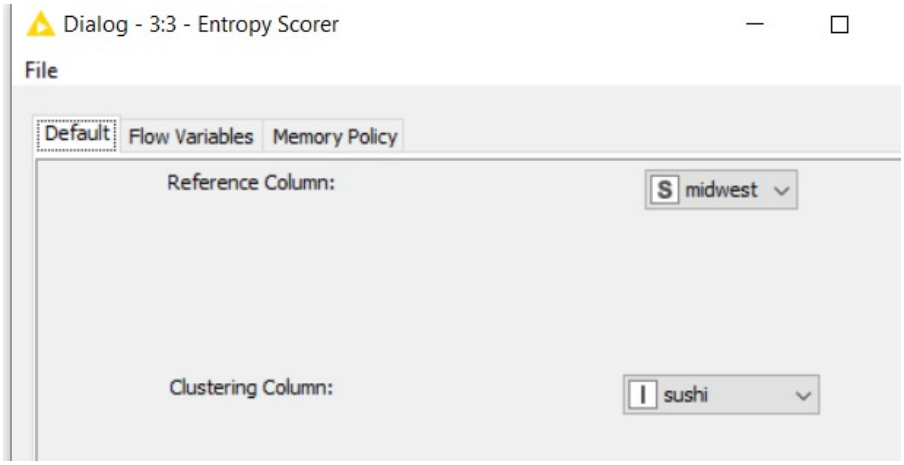
File Ta

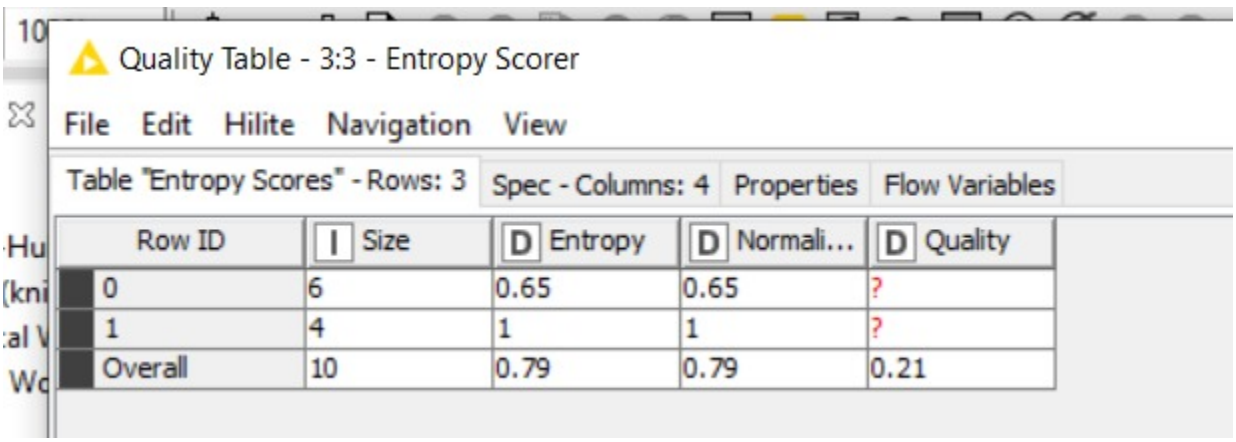
File Edit Hilite Navigation View

Table "default" - Rows: 10 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	I Nos	I Age	S apple_...	S potat...	S sushi	S midwest
Row0	1	32	no	yes	yes	yes
Row1	2	25	yes	yes	no	yes
Row2	3	33	yes	yes	no	yes
Row3	4	26	no	no	yes	no
Row4	5	30	yes	yes	yes	yes
Row5	6	25	yes	no	no	yes
Row6	7	42	yes	yes	no	yes
Row7	8	38	yes	yes	no	yes
Row8	9	36	yes	yes	no	no
Row9	10	29	yes	no	yes	no

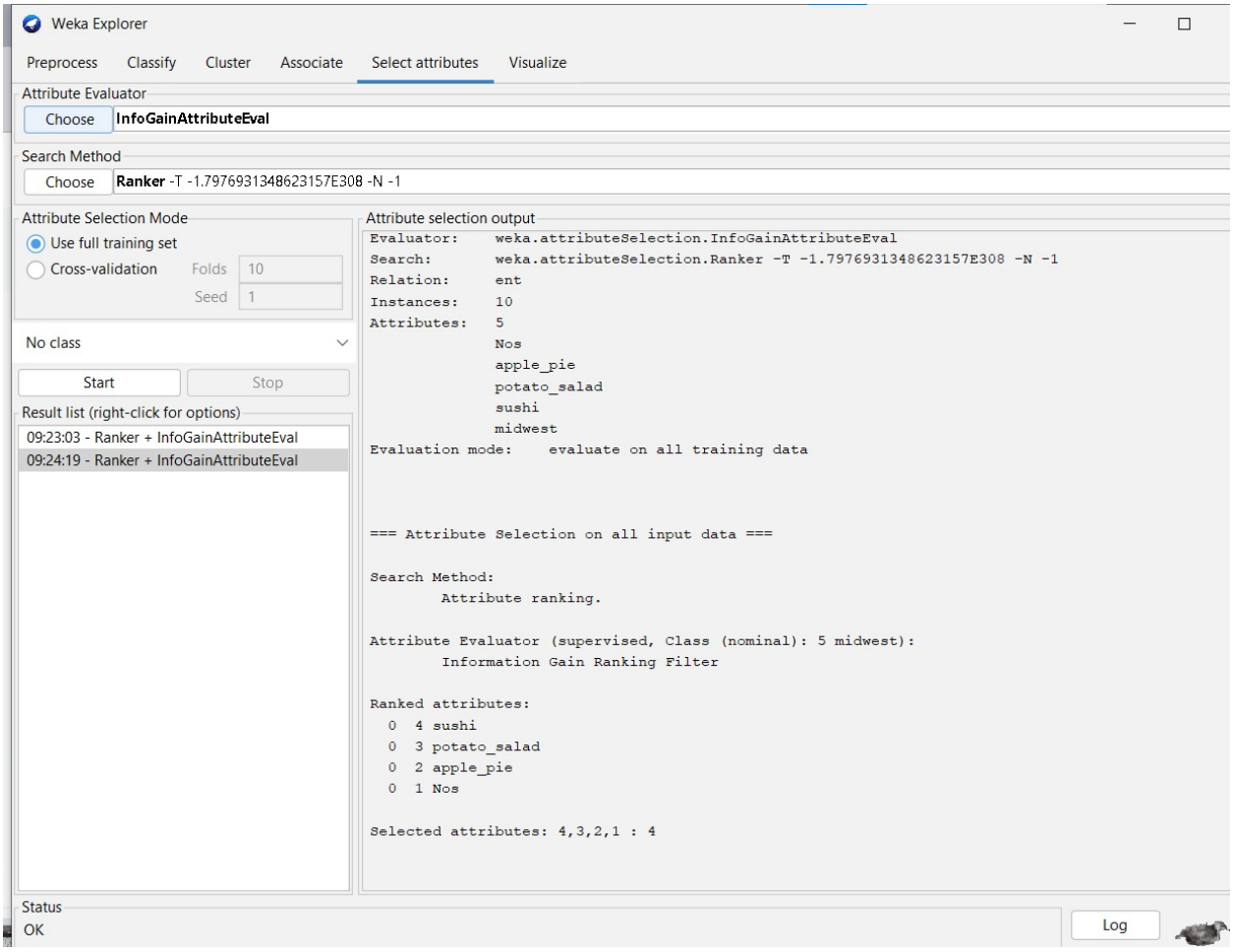
Output:





Row ID	I Size	D Entropy	D Normali...	D Quality
0	6	0.65	0.65	?
1	4	1	1	?
Overall	10	0.79	0.79	0.21

<u>Weka:</u>



<u>Conclusion</u>: we studied and implemented Infogain, entropy, information gain.

Experiment No: 04

Find t and d-weight of a data

Walchand College of Engineering, Sangli.

Experiment No: 4

Aim:- To find T and D weight

Theory:- The t weight is a quantative characteristic rule which can be logically represented as t-weight
t-weight implies weight within class.

d weight is a quantative discriminant rule which can be logically represented as d-weight
d-weight implies weight across class.

A higher d-weight of given class that concept represented by generalized tuple is primarily derived from target class.

Data:-

class	Boys	Girls	Total
FY	80	60	140
TY	73	50	123
Total	153	110	263

t-weight of FY Boys $= \frac{80}{140} \times 100 = 57.14$

d-weight of FY Boys $= \frac{80}{153} \times 100 = 52.29$

t-weight of FY Girls = $\frac{60}{140} \times 100 = 42.86$

d-weight of FY Girls = $\frac{60}{110} \times 100 = 54.59$

t-weight of TY Boys = $\frac{73}{123} \times 100 = 59.35$

d-weight of TY Boys = $\frac{73}{153} \times 100 = 47.71$

t-weight of TY Boys = $\frac{50}{123} \times 100 = 40.65$

d-weight of TY Boys = $\frac{50}{110} \times 100 = 45.45$

Algorithm:

(1) Calculate sum of attributes within the class.
(2) Calculate sum of attributes across the class.
(3) For each element to find t-weight divide its count by its sum within the class.
(4) For each element to find d-weight, divide its count by its sum across the class.

Conclusion:
- d-weight helps to find the weight of an attribute across class
- t-weight helps to find the wt. of an attribute within class.

Program:

```
import java.util.*;
import java.io.*;
public class TWeight_DWeight
{
   public static List<Integer> getDweight(List<Integer>l,int sum)
   {
      List t=new ArrayList();
      for(int i=0;i<l.size();i++)
      {
           float f=(float)l.get(i)/sum;

           t.add(String.format("%.3f",f));

      }
      return t;

   }
   public static void getTweight(List<Integer>l,List<Integer>l2)
   {
          List D1=new ArrayList();
          List D2=new ArrayList();
          int sum=0;
          for(int i=0;i<l.size();i++)
          {
               sum=(int)l.get(i)+(int)l2.get(i);
               float f=(float)((int)l.get(i))/sum;
               float f1=(float)((int)l2.get(i))/sum;
               D1.add(String.format("%.3f",f));
               D2.add(String.format("%.3f",f1));

          }
          System.out.println("TweightB:\t"+D1);
          System.out.println("TweightG:\t"+D2);

   }
   public static void main(String args[])
   {
      try{
      Scanner sc=new Scanner(new File("TDweight.csv"));
      List<Integer> Boys=new ArrayList();
      List<Integer>girls=new ArrayList();
      int i=0;
      int sumBoy=0;
```

```
        int sumgirl=0;
        while(sc.hasNext())
        {
            if(i==0)
            {

                sc.nextLine();
                i++;
                continue;

            }
            String s[]=(sc.nextLine()).split("\\,");

            Boys.add(Integer.parseInt(s[1]));

            girls.add(Integer.parseInt(s[2]));
        }

        int sum=Boys.stream().mapToInt(Integer::intValue).sum();
        System.out.println(sum);
         int sum1=girls.stream().mapToInt(Integer::intValue).sum();
        System.out.println(sum1);
        List dweightb=getDweight(Boys,sum);
        List dweightg=getDweight(girls,sum1);

        System.out.println("DweightB:\t"+dweightb);
        System.out.println("DweightG:\t"+dweightg);

        getTweight(Boys,girls);


        }catch(Exception e){}


    }


}
```

Input:

TWeight.txt

```
class Maths Science
11 80 88
12 98 90
```

Output:

```
shubham@shubham-Vostro-3405:~/Desktop/OSS $ java TDWeight
row 1 total: 168
row 2 total: 188
col 1 total: 178
col 2 total: 178

t-weight for data 80is---> 47.6190
d-weight for data 80is---> 44.9438
t-weight for data 88is---> 52.3810
d-weight for data 88is---> 49.4382
t-weight for data 98is---> 52.1277
d-weight for data 98is---> 55.0562
t-weight for data 90is---> 47.8723
shubham@shubham-Vostro-3405:~/Desktop/OSS $
```

Conclusion: we successfully studied and implemented t-weight and d-weight experiment

Experiment No: 05

Find 5 no summary of a dataset (BoxPlot)

Walchand College of Engineering, Sangli.

Experiment No: 05

Aim: To find 5 number summary of a dataset

Theory:

5 minutes summary of a dataset includes

1. minimum
2. maximum
3. Quartile1
4. Median
5. Quartile2

5 number summary is calculated to give the quick distribution of data. For box plot & whiskers length.

Minimum :- Smallest element of dataset

Quartile1 :- The Median of data between minimum & median

Median :- Depending the length of data the median is calculated.

Quartile2 :- The median of data between median & maximum

Maximum : largest element of dataset.

Algorithm

(1) Sort the given elements
(2) Store them in an array.
(3) Find minimum by finding the first element
(4) Find maximum by finding the last element
(5) If the number of elements are odd. the median is middle number in array. else if no. of elements are even the median of middle two elements is median.
(6) Divide the numbers in two groups through the median.
(7) Using step 5, find Q_1 & Q_3 that is median of respective sections.
(8) Calculate IQR
(9) Calculate higher & lower calculate & draw box-plot.

Data:-

54 4 45 69 62 16 27 77 72

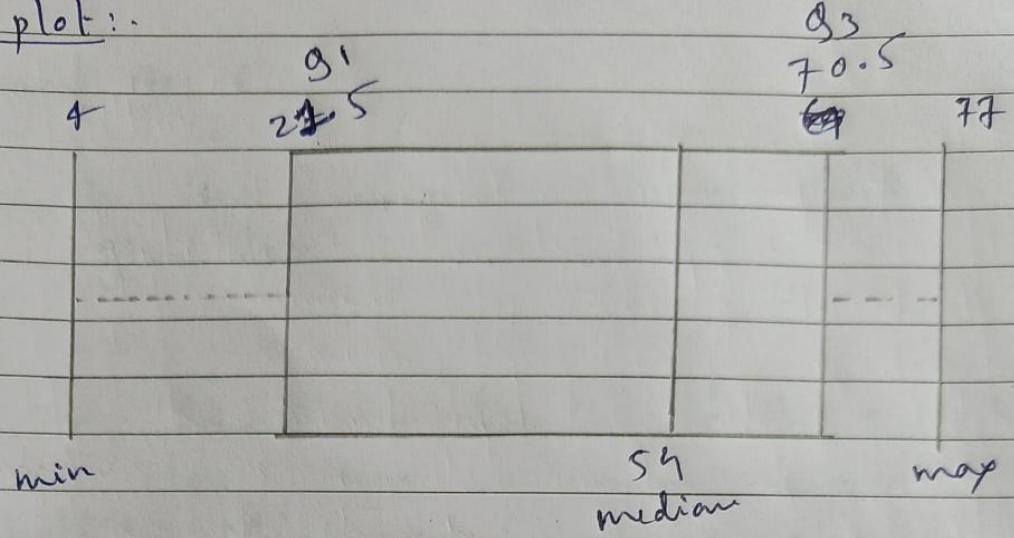
→ sort

4 16 27 45 54 62 69 72 77

① Minimum = 4
② Maximum = 77
③ Total elements = 9
∴ median = $\left(\frac{9+1}{2}\right)^{th}$ element = 54

④ $Q_1 = \frac{16+27}{2} = 21.5$ ⑤ $Q_3 = \frac{69+72}{2} = 70.5$

Box plot:-



Weka output

min = 4
max = 77
median = 54
q_1 = 21.5
q_3 = 70.5

~~Knime output~~.

Knime output.

min = 4
max = 77
median = 54
q_1 = 21.5
q_3 = 70.5

Conclusion:-

(1) ~~Similar~~ output ~~has been~~
Box plot helps to analyze data in more easier ways.

(2) Box plots help visualize the distribution of quantitative values in fields.

Input:

```
it@it-OptiPlex-3050:~/Desktop$ java BoxPlot
Sorted array is
15 18 20 22 25 29 32 34 38 41 43 46 54 76
```

Code:

```
import java.lang.*;
import java.util.*;
import java.io.*;
class BoxPlot
{

  public static double findMedian(int arr[],int start,int end)
  {
      int n=end-start+1;
      if(n%2==0)
      {
          return (arr[start+n/2-1]+arr[start+n/2])/2;
      }
      else
      {
          return arr[start+n/2];
      }

  }
  public static void main(String[] args) throws Exception
  {
   BufferedReader   br = new BufferedReader(new FileReader("FiveNoSummary.csv"));


   String [] fields = br.readLine().trim().split(","); // Splits at the space
   int i=0;
   int [] arr=new int[fields.length];
   for(String str:fields){
   int x=Integer.parseInt(str); //prints out name
   arr[i++]=x;
   }

   Arrays.sort(arr);
```

```
    int n=arr.length;
    double median=findMedian(arr,0,n-1);
    double q1=findMedian(arr,0,n/2-1);
    double q3=0;


        q3=findMedian(arr,n/2+1,n-1);


    PrintWriter out=new PrintWriter(new File("output_boxplot.csv"));
    out.printf("%s,%s","Attribute","Value");
    out.printf("\n");

    out.printf("%s,%d\n","Max",arr[n-1]);
    out.printf("%s,%d\n","Min",arr[0]);
    out.printf("%s,%.3f\n","Median",median);
    out.printf("%s,%.3f\n","Q1",q1);
    out.printf("%s,%.3f\n","Q3",q3);

    double iqr=q3-q1;
    double upper_out=q3 + (1.5)*iqr;
     double lower_out=q1 - (1.5)*iqr;
    out.printf("%s,%.3f\n","upper_out",upper_out);
    out.printf("%s,%.3f\n","lower_out",lower_out);



    out.close();
   }
}
```

Output:

output_boxplot.csv

```
Attribute,Value
Max,77
Min,4
Median,54.000
Q1,21.000
Q3,70.000
upper_out,143.500
lower_out,-52.500
```

Knime:

Input:

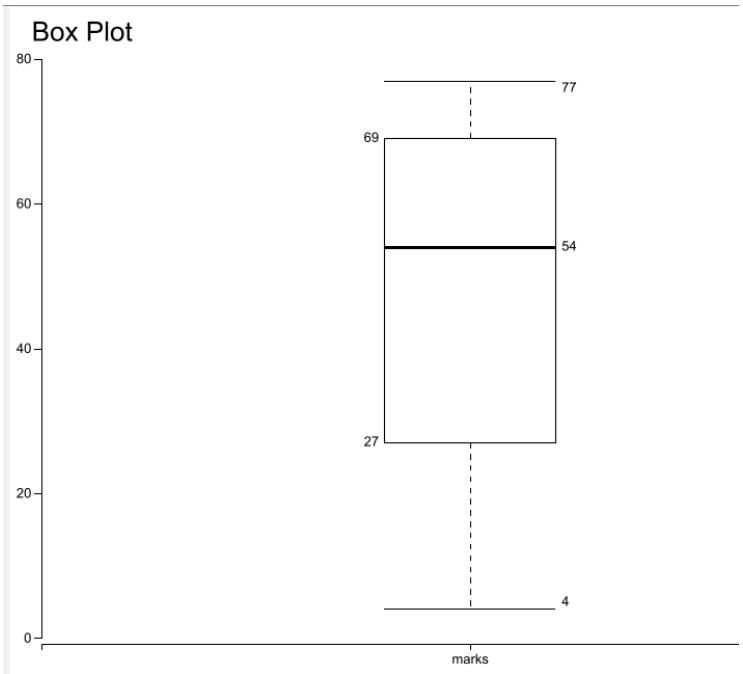
File Table - 3:1 - CSV Reader

File Edit Hilite Navigation View

Table "default" - Rows: 9 Spec - Column: 1 P

Row ID	marks
Row0	54
Row1	4
Row2	45
Row3	69
Row4	62
Row5	16
Row6	27
Row7	77
Row8	72

Output:



Experiment No: 06

Find frequent itemset from a given transaction data

Walchand College of Engineering, Sangli.

Experiment No: 6 :

Aim :- To find frequent itemset from given data.

Theory :-

An itemset is frequent if its support is no less than minimum Support threshold. Minimum Support is always supposed according to choice.

$$\text{Support}(A) = \frac{\text{Number of transactions } (A)}{\text{Total no. of transaction}}$$

$$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$

Confidence is the probability that if a person buys an item A, then he will also buy A an item B

A minimum Support ~~Con~~ Criteria can be applied to get all frequent item sets in a dataset.

Algorithm :-

(1) Calculate all the frequencies of Unique items in a dataset.
(2) find sum of Unique transactions in the dataset.
(3) Calculate the Support for each Unique item in dataset.
(4) Set minimum Support Criteria.
(5) Find the frequent items by finding frequent item sets whose Support is above the minimum Support.

Data :-

Tid	0	1	2	3
1	Soap	Detergent	Shampoo	
2	Shampoo	Soap		
3	pen	Book	Soap	milk
4	milk	Bread	pen	
5	Soap	Milk	pen	Bread
6	Shampoo	pen	Bread	
7	milk	Rice	Bread	Soap
8	detergent	Rice	milk	
9	Pen	book	milk	
10	milk			

Let's assume minimum suppart at 0.2

Itemset : Book $= \frac{2}{10} = 0.2$

Bread $= \frac{4}{10} = 0.4$

Milk $= \frac{7}{10} = 0.7$

Rice $= \frac{2}{10} = 0.2$

Book, Milk $= \frac{2}{10} = 0.2$

pen, Book $= \frac{1+1}{10} = 0.2$

Milk, Bread $= \frac{1+2}{10} = 0.3$

shampoo, Bread $= \frac{1+1}{10} = 0.2$

Detergent, milk $= \frac{1+1}{10} = 0.2$

pen, milk $= 0.4$

All the itemset with suppart above 0.2 will be Considered as frequent

Knime Result:-

RowID	Support	[...] items
Itemset0	0.3	[milk, Soap]
Itemset1	0.3	[shampoo, ?]
Itemset2	0.3	[pen, ?]
Itemset3	0.3	[bread, pen]
Itemset4	0.3	[bread, milk]
Itemset5	0.4	[bread]
Itemset6	0.4	[milk, ?]
Itemset7	0.4	[pen, milk]
Itemset8	0.5	[Soap]
Itemset9	0.5	[pen]
Itemset10	0.7	[?]
Itemset11	0.7	[milk]

Conclusion:-

(1) The itemset obtained at min Support by Calculating and by using weka are Similar

(2) Minimum Support Criteria is used to Calculate frequent item set from a list of transaction.

Program:

```
import java.io.*;
import java.util.*;
public class AprioriDataMining
{
	Vector<String> candidates=new Vector<String>();
	List<String> itemSet = new ArrayList<String>();
	List<String> finalFrequentItemSet = new ArrayList<>();
	HashMap<String,Integer> frequentItems = new HashMap<String, Integer>();
	String newLine = System.getProperty("line.separator");
	int
itemsCount,countItemOccurrence=0,displayFrequentItemSetNumber=2,displayTransactionNumber=1;

	public static void main(String args[]) throws IOException
	{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

		int noOfTransactions,minimumSupport;
		double minimumConfidence;
		String sampleFile = args[0];
		List<String> transactions = new ArrayList<String>();

		String newLine = System.getProperty("line.separator");

		System.out.println(newLine+"'APRIORI ALGORITHM'");
		System.out.print("Enter the Minimum Support = ");
		minimumSupport = Integer.parseInt(br.readLine());
		System.out.print("Enter the Minimum Confidence (in %) = ");
		minimumConfidence = Double.parseDouble(br.readLine());
		minimumConfidence = minimumConfidence/100;

		File file = new File(sampleFile);
		Scanner sc = new Scanner(file);

		while (sc.hasNextLine())
		{
			String str = sc.nextLine();
			transactions.add(str);
		}

		noOfTransactions = transactions.size();
```

```
		AprioriDataMining a = new AprioriDataMining();
		a.display(noOfTransactions, transactions, minimumSupport, minimumConfidence);
	}

	public void display(int noOfTransactions, List<String> transactions, int minimumSupport,
double minimumConfidence)
	{
		for(int i = 0; i<noOfTransactions;i++)
		{
			String str = transactions.get(i);
			String[] words = str.split(" ");
			int count = words.length;
			for(int j=0;j<count;j++)
			{
				if(i==0)
				{
					itemSet.add(words[j]);
				}
				else
				{
					if(!(itemSet.contains(words[j])))
					{
						itemSet.add(words[j]);
					}
				}
			}
		}

		itemsCount = itemSet.size();
		System.out.println(newLine+"No of Items = "+itemsCount);
		System.out.println("No of Transactions = "+noOfTransactions);
		System.out.println("Minimum Support = "+minimumSupport);
		System.out.println("Minimum Confidence = "+minimumConfidence+newLine);

		System.out.println("'Items present in the Database'");
		for(String i : itemSet)
		{
			System.out.println("------> "+i);
		}

		System.out.println(newLine+"TRANSACTION ITEMSET");
		for(String i : transactions)
		{
			System.out.println("Transaction "+displayTransactionNumber+" = "+i);
```

```
                displayTransactionNumber++;
            }

firstFrequentItemSet(noOfTransactions,transactions,minimumSupport,minimumConfidence);

        }

        public void firstFrequentItemSet(int noOfTransactions,List<String> transactions,int
minimumSupport, double minimumConfidence)
        {
            System.out.println();
            System.out.println("Frequent Itemset 1");
            for(int items=0;items<itemSet.size();items++)
            {
                countItemOccurrence=0;
                String itemStr = itemSet.get(items);
                for(int t=0;t<noOfTransactions;t++)
                {
                    String transactionStr = transactions.get(t);
                    if(transactionStr.contains(itemStr))
                    {
                        countItemOccurrence++;
                    }
                }

                if(countItemOccurrence >= minimumSupport)
                {
                    System.out.println(itemStr+" => support = "+countItemOccurrence);
                    finalFrequentItemSet.add(itemStr);
                    frequentItems.put(itemStr, countItemOccurrence);
                }
            }

            aprioriStart(noOfTransactions,transactions,minimumSupport,minimumConfidence);
        }

        public void aprioriStart(int noOfTransactions,List<String> transactions,int minimumSupport,
double minimumConfidence)
        {
            int itemsetNumber=1;

            for(int i=0;i<finalFrequentItemSet.size();i++)
            {
                String str = finalFrequentItemSet.get(i);
```

```
                candidates.add(str);
        }

        do
        {
                itemsetNumber++;
                generateCombinations(itemsetNumber);
                checkFrequentItems(noOfTransactions,transactions,minimumSupport);
        }
        while(candidates.size()>1);

        // System.out.println("Association Rules for Frequent Itemset"+newLine);

        // generateAssociationRules(noOfTransactions, transactions, minimumConfidence);
}

private void generateCombinations(int itr)
{
        Vector<String> candidatesTemp = new Vector<String>();
        String s1, s2;
        StringTokenizer strToken1, strToken2;
        if(itr==2)
        {
                for(int i=0; i<candidates.size(); i++)
                {
                        strToken1 = new StringTokenizer(candidates.get(i));
                        s1 = strToken1.nextToken();
                        for(int j=i+1; j<candidates.size(); j++)
                        {
                                strToken2 = new StringTokenizer(candidates.elementAt(j));
                                s2 = strToken2.nextToken();
                                String addString = s1+" "+s2;
                                candidatesTemp.add(addString);
                        }
                }
        }
        else
        {
                for(int i=0; i<candidates.size(); i++)
                {
                        for(int j=i+1; j<candidates.size(); j++)
                        {
                                s1 = new String();
                                s2 = new String();
```

```
                                        strToken1 = new StringTokenizer(candidates.get(i));
                                        strToken2 = new StringTokenizer(candidates.get(j));

                                        for(int s=0; s<itr-2; s++)
                                        {
                                                s1 = s1 + " " + strToken1.nextToken();
                                                s2 = s2 + " " + strToken2.nextToken();
                                        }

                                        if(s2.compareToIgnoreCase(s1)==0)
                                        {
                                                String addString = (s1 + " " + strToken1.nextToken() + " "
+ strToken2.nextToken()).trim();
                                                candidatesTemp.add(addString);
                                        }
                                }
                        }
                }
                candidates.clear();
                candidates = new Vector<String>(candidatesTemp);
                candidatesTemp.clear();
                System.out.println();
        }

        public void checkFrequentItems(int noOfTransactions,List<String> transactions, int
minimumSupport)
        {
                List<String> combList = new ArrayList<String>();
                for(int i=0;i<candidates.size();i++)
                {
                        String str = candidates.get(i);
                        combList.add(str);
                }
                System.out.println("Frequent Itemset "+displayFrequentItemSetNumber);
                for(int i=0;i<combList.size();i++)
                {
                        String str = combList.get(i);
                        String[] words = str.split(" ");
                        int count = words.length;
                        int flag = 0, itemSetOccurence=0;
                        for(int t=0;t<noOfTransactions;t++)
                        {
                                String transac = transactions.get(t);
```

```
                        for(int j=0;j<count;j++)
                        {
                                String wordStr = words[j];
                                if(transac.contains(wordStr))
                                {
                                        flag++;
                                }
                        }
                        if(flag==count)
                        {
                                itemSetOccurence++;
                        }
                        flag=0;
                }

                if(itemSetOccurence>=minimumSupport)
                {
                        System.out.println(str+" => support = "+itemSetOccurence);
                        frequentItems.put(str, itemSetOccurence);
                        finalFrequentItemSet.add(str);
                }
                itemSetOccurence=0;
        }
        displayFrequentItemSetNumber++;
    }

    public void generateAssociationRules(int noOfTransactions,List<String> transactions,double
minimumConfidence)
    {
        double confidence,confidence1;

        for(int i=0;i<finalFrequentItemSet.size();i++)
        {
                int spring2019count=0;
                String itemSetStr = finalFrequentItemSet.get(i);
                double value = frequentItems.get(itemSetStr);
                String str = "",str1="";
                String[] words = itemSetStr.split(" ");
                int wordCountInString = words.length;
                if(wordCountInString==2) /* for FrequentItemSet = 2*/
                {
                        double s = frequentItems.get(words[0]);
                        confidence = value/s;
                        spring2019count++;
```

```
                    if(confidence>=minimumConfidence)
                    {
                    System.out.println(words[0]+" -> "+words[1]+" = Confidence = "+
confidence*100 +" and Support = "+(int)value+"");

                    }
                    double s1 = frequentItems.get(words[1]);
                    confidence = value/s1;
                    spring2019count++;
                    if(confidence>=minimumConfidence)
                    {
                        System.out.println(words[1]+" -> "+words[0]+" = Confidence = "+
confidence*100+" and Support = "+(int)value+"");

                    }
                }
                else /* for FrequentItemSet > 2 */
                {
                    for(int a=0;a<wordCountInString-1;a++)
                    {
                        if(a==0)
                        {
                            str = str+words[a];
                            spring2019count++;
                        }
                        else
                        {
                            str = str+" "+words[a];
                            spring2019count++;
                        }
                        for(int j=a+1;j<wordCountInString;j++)
                        {
                            {
                                str1=str1+" "+words[j];
                                spring2019count++;
                            }
                        }
                        double s = frequentItems.get(str);
                        confidence = value/s;
                        String st = str1.trim();
                        double s1 = frequentItems.get(st);
                        confidence1 = value/s1;
                        if(confidence>=minimumConfidence)
                        {
```

```
                                        System.out.println(str+" -> "+str1+" = Confidence =
"+confidence*100+" and Support = "+(int)value+"");

                                    }
                                    if(confidence1>=minimumConfidence)
                                    {
                                        System.out.println(st+" -> "+str+" = Confidence =
"+confidence1*100+" and Support = "+(int)value+"");

                                    }
                                    str1="";
                                }
                                str="";str1="";
                            }

                            //System.out.println(spring2019count++);
                        }
                    }
}
```

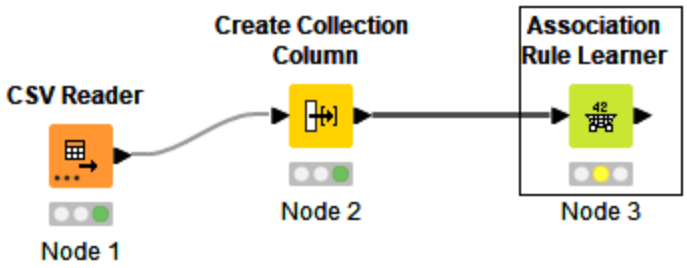
Input:

6_7.csv

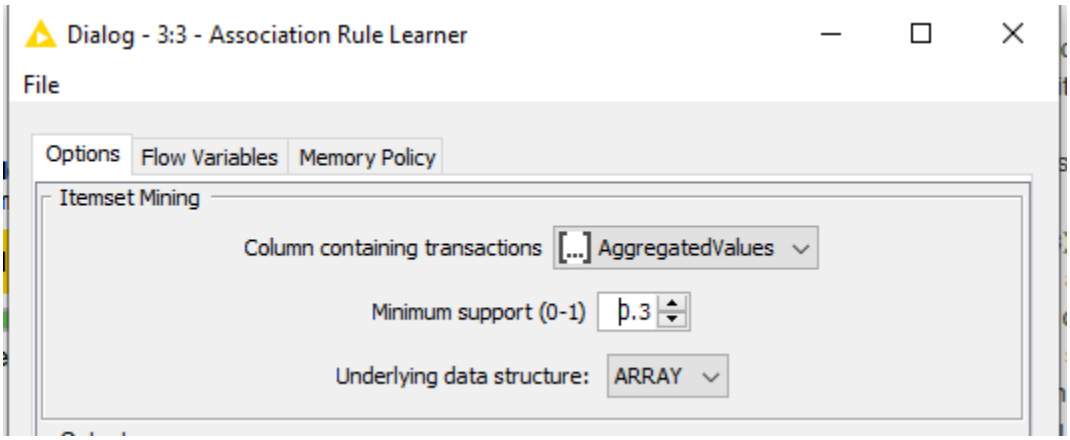
```
soap,detergent,shampoo
shampoo,soap
pen,book,soap,milk
milk,bread,pen
soap,milk,pen,bread
shampoo,pen,bread
milk,rice,bread,soap
detergent,rice,milk
pen,book,milk
milk
```

```
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$ javac AprioriDataMining.java
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$ java AprioriDataMining "6_7.csv"

 'APRIORI ALGORITHM'
 Enter the Minimum Support = 3
 Enter the Minimum Confidence (in %) = 40

 No of Items = 10
 No of Transactions = 10
 Minimum Support = 3
 Minimum Confidence = 0.4

 'Items present in the Database'
 ------> soap,detergent,shampoo
 ------> shampoo,soap
 ------> pen,book,soap,milk
 ------> milk,bread,pen
 ------> soap,milk,pen,bread
 ------> shampoo,pen,bread
 ------> milk,rice,bread,soap
 ------> detergent,rice,milk
 ------> pen,book,milk
 ------> milk

 TRANSACTION ITEMSET
 Transaction 1 = soap,detergent,shampoo
 Transaction 2 = shampoo,soap
 Transaction 3 = pen,book,soap,milk
 Transaction 4 = milk,bread,pen
 Transaction 5 = soap,milk,pen,bread
 Transaction 6 = shampoo,pen,bread
 Transaction 7 = milk,rice,bread,soap
 Transaction 8 = detergent,rice,milk
 Transaction 9 = pen,book,milk
 Transaction 10 = milk

 Frequent Itemset 1
 milk => support = 7

 Frequent Itemset 2
```

Knime:



Input:



Output:

Frequent itemsets/Association rules - 3:3 - Association Rule Learner

File Edit Hilite Navigation View

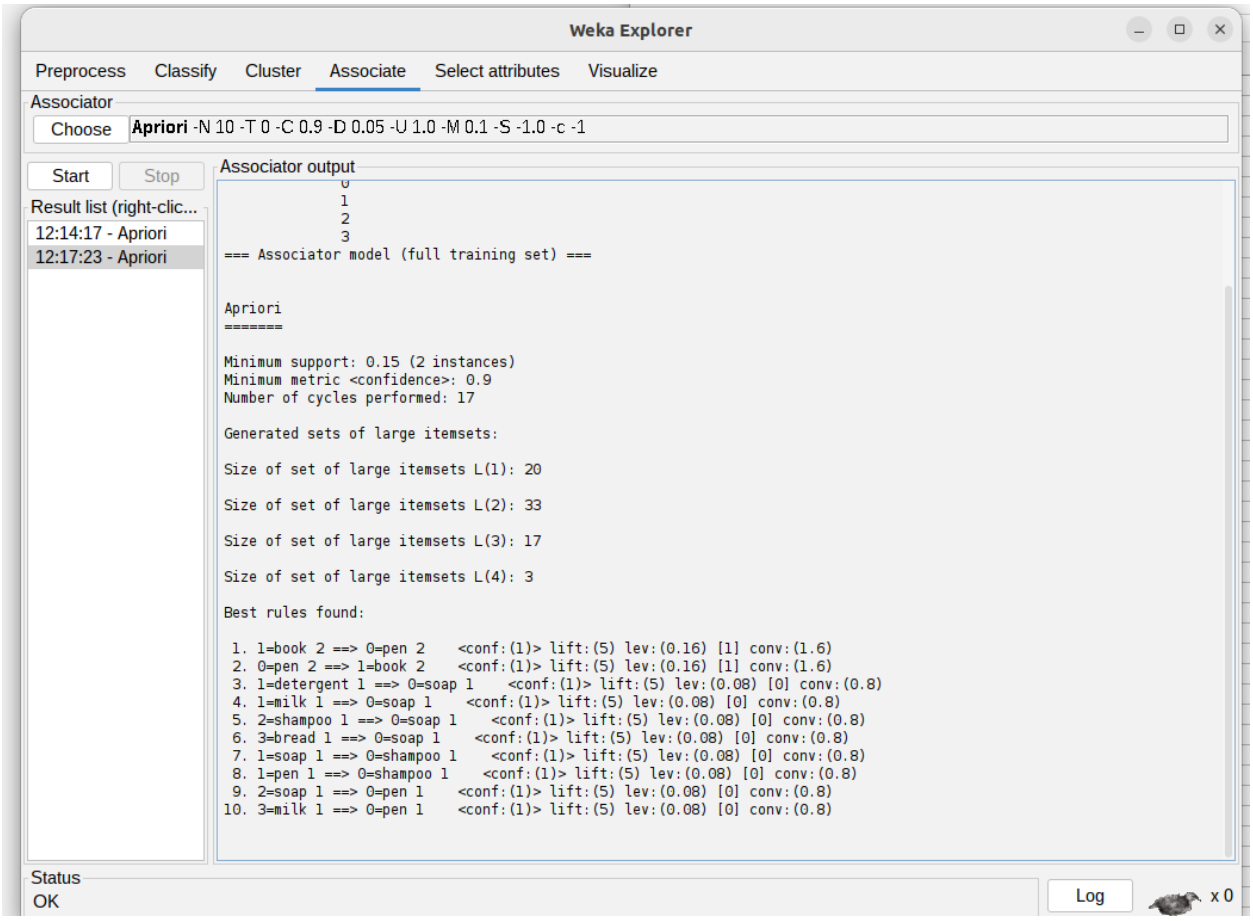
Table "default" - Rows: 12 | Spec - Columns: 2 | Properties | Flow Variables

Row ID	D Suppor...	[...] Items
item set 0	0.3	[milk,soap]
item set 1	0.3	[shampoo,?]
item set 2	0.3	[pen,?]
item set 3	0.3	[bread,pen]
item set 4	0.3	[bread,milk]
item set 5	0.4	[bread]
item set 6	0.4	[milk,?]
item set 7	0.4	[pen,milk]
item set 8	0.5	[soap]
item set 9	0.5	[pen]
item set 10	0.7	[?]
item set 11	0.7	[milk]

Weka:

Input file:

	A	B	C	D	E
1	0	1	2	3	
2	soap	detergent	shampoo		
3	shampoo	soap			
4	pen	book	soap	milk	
5	milk	bread	pen		
6	soap	milk	pen	bread	
7	shampoo	pen	bread		
8	milk	rice	bread	soap	
9	detergent	rice	milk		
10	pen	book	milk		
11	milk				
12					


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-clic...
12:14:17 - Apriori
12:17:23 - Apriori
Associator output
0
1
2
3
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17
Generated sets of large itemsets:
Size of set of large itemsets L(1): 20
Size of set of large itemsets L(2): 33
Size of set of large itemsets L(3): 17
Size of set of large itemsets L(4): 3
Best rules found:
1. 1=book 2 ==> 0=pen 2 <conf:(1)> lift:(5) lev:(0.16) [1] conv:(1.6)
2. 0=pen 2 ==> 1=book 2 <conf:(1)> lift:(5) lev:(0.16) [1] conv:(1.6)
3. 1=detergent 1 ==> 0=soap 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
4. 1=milk 1 ==> 0=soap 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
5. 2=shampoo 1 ==> 0=soap 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
6. 3=bread 1 ==> 0=soap 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
7. 1=soap 1 ==> 0=shampoo 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
8. 1=pen 1 ==> 0=shampoo 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
9. 2=soap 1 ==> 0=pen 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
10. 3=milk 1 ==> 0=pen 1 <conf:(1)> lift:(5) lev:(0.08) [0] conv:(0.8)
Status
OK
Log
x 0

Experiment No: 07

Walchand College of Engineering, Sangli.

Experiment No: 7 Extend program 6 to find Association rule.

Aim:- To find Association rules.

Theory:-

Association rule finds relations among Large sets of data items. It shows how frequently an itemset occurs in a transaction.

A typical example is market Basket analysis.

Given a set of transactions we can find rules that will predict the occurrence of an item based on occurrence of others items in transaction.

Association Rule

An Implicit Expression of the factor $X \rightarrow Y$ where X and Y are only 2 itemsets.

Algorithm

(1) Find the number of Unique items.
(2) Find the sum of frequencies of each item.
(3) Find the number of transactions.
(4) Find suppat of each item set.
(5) Select a minimum Suppat.
(6) find Confidence of item set.

Data:

Tid	0	1	2	3
1	Soap	Detergent	shampoo	
2	shampoo	Soap		
3	pen	Book	Soap	Milk
4	Milk	Bread	pen	
5	Soap	Milk	pen	Bread
6	Shampoo	pen	Bread	
7	Milk	Rice	Bread	Soap
8	Detergent	Rice	Milk	
9	pen	Book	Milk	
10	Milk			

Unique items and their support

Rice	0.2
pen	0.5
Book	0.2
Bread	0.4
milk	0.7
Shampoo	0.4
soap	0.5
Detergent	0.3

Support for 2 items:

milk, Rice	0.2
milk, Bread	0.3
milk, Book	0.2
milk, pen	0.4
milk, Detergent	0.2
milk, Soap	0.3
Rice, Bread	0.1
Rice, Book	0.1
Rice, pen	0.1
Rice, Detergent	0.2
Rice, Soap	0.1

Bread, Book	0.1
Bread, pen	0.2
Bread, Detergent	0
Bread, Soap	0.1

Calculating the Confidence using

$$\text{Confidence } (A \rightarrow B) = \frac{\text{Support } (A \cup B)}{\text{Support } (A)}$$

$$\text{Confidence } (\text{Rice} \rightarrow \text{milk}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Rice} \rightarrow \text{Detergent}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Book} \rightarrow \text{pen}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Bread} \rightarrow \text{pen}) = \frac{0.3}{0.4} = 0.75$$

$$\text{Confidence of } (\text{pen} \rightarrow \text{Bread}) = \frac{0.3}{0.5} = 0.6$$

$$\text{Confidence of } (\text{Book} \rightarrow \text{Milk}) = \frac{0.4}{0.5} = 0.8$$

Weka output.

Best Rules found:

1. Book → pen
2. pen → Book
3. Detergent → soap
4. Milk → soap
5. Shampoo → Soap
6. Bread → soap
7. Shampoo → Soap
8. Soap → shampoo
9. pen → shampoo
10. Soap → pen

Conclusion:

(1) Association Rule learning is a rule-based machine learning method for discovering interesting relations between variables in larger database

Program:

```
import java.io.*;
import java.util.*;
public class AprioriDataMining
{
	Vector<String> candidates=new Vector<String>();
	List<String> itemSet = new ArrayList<String>();
	List<String> finalFrequentItemSet = new ArrayList<>();
	HashMap<String,Integer> frequentItems = new HashMap<String, Integer>();
	String newLine = System.getProperty("line.separator");
	int
itemsCount,countItemOccurrence=0,displayFrequentItemSetNumber=2,displayTransactionNumber=1;

	public static void main(String args[]) throws IOException
	{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

		int noOfTransactions,minimumSupport;
		double minimumConfidence;
		String sampleFile = args[0];
		List<String> transactions = new ArrayList<String>();

		String newLine = System.getProperty("line.separator");

		System.out.println(newLine+"'APRIORI ALGORITHM'");
		System.out.print("Enter the Minimum Support = ");
		minimumSupport = Integer.parseInt(br.readLine());
		System.out.print("Enter the Minimum Confidence (in %) = ");
		minimumConfidence = Double.parseDouble(br.readLine());
		minimumConfidence = minimumConfidence/100;

		File file = new File(sampleFile);
		Scanner sc = new Scanner(file);

		while (sc.hasNextLine())
		{
			String str = sc.nextLine();
			transactions.add(str);
		}

		noOfTransactions = transactions.size();
```

```
		AprioriDataMining a = new AprioriDataMining();
		a.display(noOfTransactions, transactions, minimumSupport, minimumConfidence);
	}

	public void display(int noOfTransactions, List<String> transactions, int minimumSupport,
double minimumConfidence)
	{
		for(int i = 0; i<noOfTransactions;i++)
		{
			String str = transactions.get(i);
			String[] words = str.split(" ");
			int count = words.length;
			for(int j=0;j<count;j++)
			{
				if(i==0)
				{
					itemSet.add(words[j]);
				}
				else
				{
					if(!(itemSet.contains(words[j])))
					{
						itemSet.add(words[j]);
					}
				}
			}
		}

		itemsCount = itemSet.size();
		System.out.println(newLine+"No of Items = "+itemsCount);
		System.out.println("No of Transactions = "+noOfTransactions);
		System.out.println("Minimum Support = "+minimumSupport);
		System.out.println("Minimum Confidence = "+minimumConfidence+newLine);

		System.out.println("'Items present in the Database'");
		for(String i : itemSet)
		{
			System.out.println("------> "+i);
		}

		System.out.println(newLine+"TRANSACTION ITEMSET");
		for(String i : transactions)
		{
```

```
                System.out.println("Transaction "+displayTransactionNumber+" = "+i);
                displayTransactionNumber++;
            }

firstFrequentItemSet(noOfTransactions,transactions,minimumSupport,minimumConfidence);

        }

        public void firstFrequentItemSet(int noOfTransactions,List<String> transactions,int
minimumSupport, double minimumConfidence)
        {
            System.out.println();
            System.out.println("Frequent Itemset 1");
            for(int items=0;items<itemSet.size();items++)
            {
                countItemOccurrence=0;
                String itemStr = itemSet.get(items);
                for(int t=0;t<noOfTransactions;t++)
                {
                    String transactionStr = transactions.get(t);
                    if(transactionStr.contains(itemStr))
                    {
                        countItemOccurrence++;
                    }
                }

                if(countItemOccurrence >= minimumSupport)
                {
                    System.out.println(itemStr+" => support = "+countItemOccurrence);
                    finalFrequentItemSet.add(itemStr);
                    frequentItems.put(itemStr, countItemOccurrence);
                }
            }

            aprioriStart(noOfTransactions,transactions,minimumSupport,minimumConfidence);
        }

        public void aprioriStart(int noOfTransactions,List<String> transactions,int minimumSupport,
double minimumConfidence)
        {
            int itemsetNumber=1;

            for(int i=0;i<finalFrequentItemSet.size();i++)
            {
```

```
                String str = finalFrequentItemSet.get(i);
                candidates.add(str);
        }

        do
        {
                itemsetNumber++;
                generateCombinations(itemsetNumber);
                checkFrequentItems(noOfTransactions,transactions,minimumSupport);
        }
        while(candidates.size()>1);

        System.out.println("Association Rules for Frequent Itemset"+newLine);

         generateAssociationRules(noOfTransactions, transactions, minimumConfidence);
}

private void generateCombinations(int itr)
{
        Vector<String> candidatesTemp = new Vector<String>();
        String s1, s2;
        StringTokenizer strToken1, strToken2;
        if(itr==2)
        {
                for(int i=0; i<candidates.size(); i++)
                {
                        strToken1 = new StringTokenizer(candidates.get(i));
                        s1 = strToken1.nextToken();
                        for(int j=i+1; j<candidates.size(); j++)
                        {
                                strToken2 = new StringTokenizer(candidates.elementAt(j));
                                s2 = strToken2.nextToken();
                                String addString = s1+" "+s2;
                                candidatesTemp.add(addString);
                        }
                }
        }
        else
        {
                for(int i=0; i<candidates.size(); i++)
                {
                        for(int j=i+1; j<candidates.size(); j++)
                        {
                                s1 = new String();
```

```
                                        s2 = new String();

                                        strToken1 = new StringTokenizer(candidates.get(i));
                                        strToken2 = new StringTokenizer(candidates.get(j));

                                        for(int s=0; s<itr-2; s++)
                                        {
                                                s1 = s1 + " " + strToken1.nextToken();
                                                s2 = s2 + " " + strToken2.nextToken();
                                        }

                                        if(s2.compareToIgnoreCase(s1)==0)
                                        {
                                                String addString = (s1 + " " + strToken1.nextToken() + " "
+ strToken2.nextToken()).trim();
                                                candidatesTemp.add(addString);
                                        }
                                }
                        }
                }
                candidates.clear();
                candidates = new Vector<String>(candidatesTemp);
                candidatesTemp.clear();
                System.out.println();
        }

        public void checkFrequentItems(int noOfTransactions,List<String> transactions, int
minimumSupport)
        {
                List<String> combList = new ArrayList<String>();
                for(int i=0;i<candidates.size();i++)
                {
                        String str = candidates.get(i);
                        combList.add(str);
                }
                System.out.println("Frequent Itemset "+displayFrequentItemSetNumber);
                for(int i=0;i<combList.size();i++)
                {
                        String str = combList.get(i);
                        String[] words = str.split(" ");
                        int count = words.length;
                        int flag = 0, itemSetOccurence=0;
                        for(int t=0;t<noOfTransactions;t++)
                        {
```

```
                        String transac = transactions.get(t);
                        for(int j=0;j<count;j++)
                        {
                                String wordStr = words[j];
                                if(transac.contains(wordStr))
                                {
                                        flag++;
                                }
                        }
                        if(flag==count)
                        {
                                itemSetOccurence++;
                        }
                        flag=0;
                }

                if(itemSetOccurence>=minimumSupport)
                {
                        System.out.println(str+" => support = "+itemSetOccurence);
                        frequentItems.put(str, itemSetOccurence);
                        finalFrequentItemSet.add(str);
                }
                itemSetOccurence=0;
        }
        displayFrequentItemSetNumber++;
    }

    public void generateAssociationRules(int noOfTransactions,List<String> transactions,double
minimumConfidence)
    {
        double confidence,confidence1;

        for(int i=0;i<finalFrequentItemSet.size();i++)
        {
                int spring2019count=0;
                String itemSetStr = finalFrequentItemSet.get(i);
                double value = frequentItems.get(itemSetStr);
                String str = "",str1="";
                String[] words = itemSetStr.split(" ");
                int wordCountInString = words.length;
                if(wordCountInString==2) /* for FrequentItemSet = 2*/
                {
                        double s = frequentItems.get(words[0]);
                        confidence = value/s;
```

```
                    spring2019count++;
                    if(confidence>=minimumConfidence)
                    {
                    System.out.println(words[0]+" -> "+words[1]+" = Confidence = "+
confidence*100 +" and Support = "+(int)value+"");

                    }
                    double s1 = frequentItems.get(words[1]);
                    confidence = value/s1;
                    spring2019count++;
                    if(confidence>=minimumConfidence)
                    {
                        System.out.println(words[1]+" -> "+words[0]+" = Confidence = "+
confidence*100+" and Support = "+(int)value+"");

                    }
                }
                else /* for FrequentItemSet > 2 */
                {
                    for(int a=0;a<wordCountInString-1;a++)
                    {
                        if(a==0)
                        {
                            str = str+words[a];
                            spring2019count++;
                        }
                        else
                        {
                            str = str+" "+words[a];
                            spring2019count++;
                        }
                        for(int j=a+1;j<wordCountInString;j++)
                        {
                            {
                                str1=str1+" "+words[j];
                                spring2019count++;
                            }
                        }
                        double s = frequentItems.get(str);
                        confidence = value/s;
                        String st = str1.trim();
                        double s1 = frequentItems.get(st);
                        confidence1 = value/s1;
                        if(confidence>=minimumConfidence)
```

```
                                        {
                                                System.out.println(str+" -> "+str1+" = Confidence =
"+confidence*100+" and Support = "+(int)value+"");

                                        }
                                        if(confidence1>=minimumConfidence)
                                        {
                                                System.out.println(st+" -> "+str+" = Confidence =
"+confidence1*100+" and Support = "+(int)value+"");

                                        }
                                        str1="";
                                }
                                str="";str1="";
                        }

                        //System.out.println(spring2019count++);
                }
        }
}
```

Input

6_7.csv

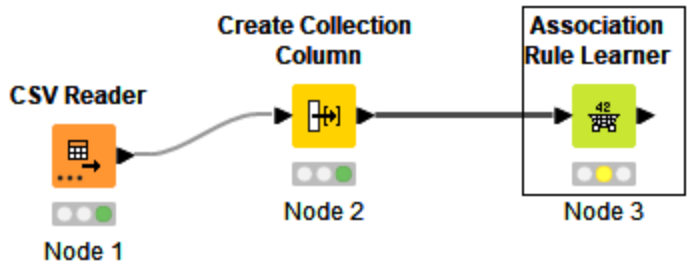
```
soap,detergent,shampoo
shampoo,soap
pen,book,soap,milk
milk,bread,pen
soap,milk,pen,bread
shampoo,pen,bread
milk,rice,bread,soap
detergent,rice,milk
pen,book,milk
milk
```

:

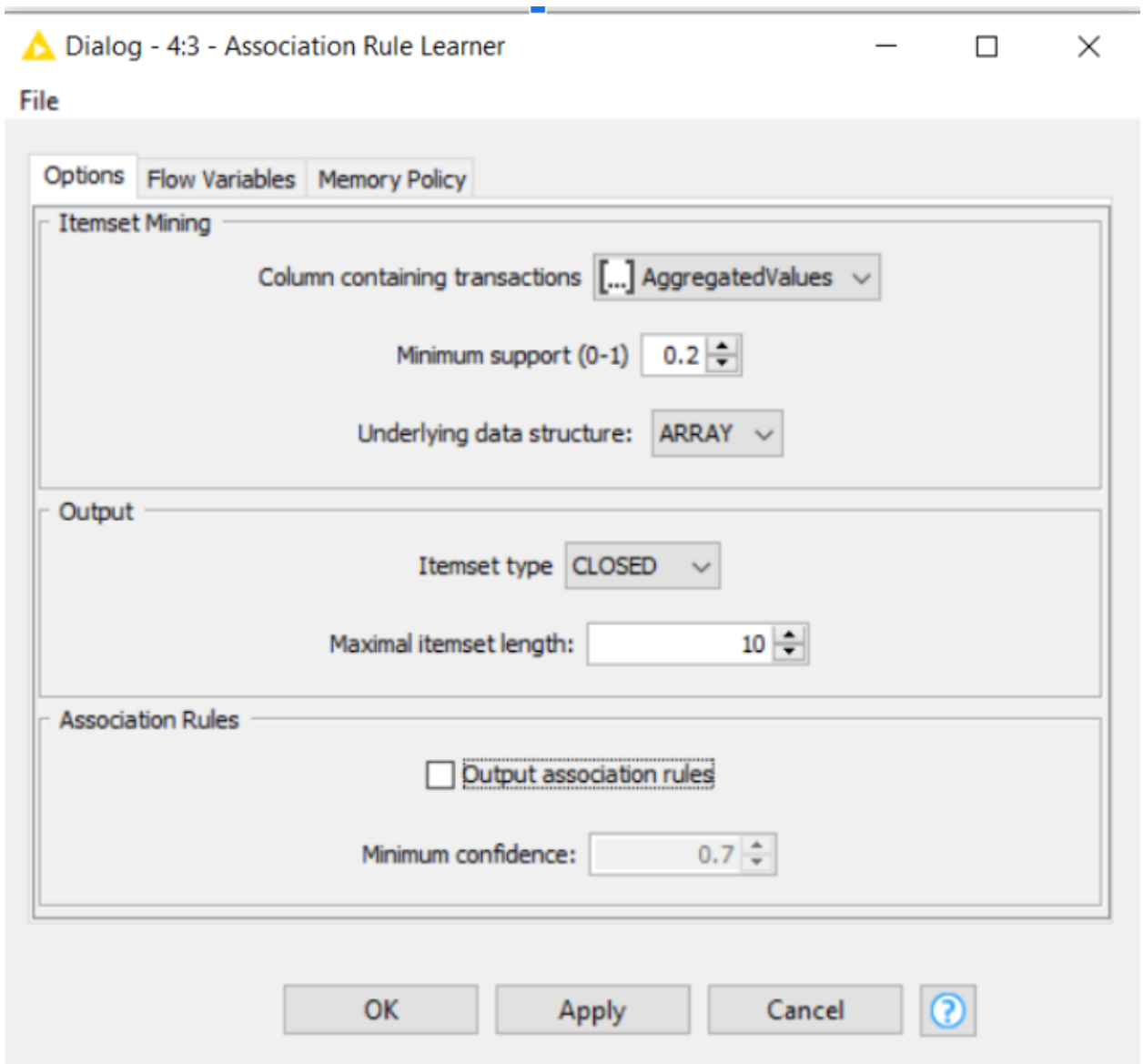
Output:

```
rohit@rnj:~/Downloads/Apriori$ javac AprioriDataMining.java
rohit@rnj:~/Downloads/Apriori$ java AprioriDataMining Apriori.txt

'APRIORI ALGORITHM'
Enter the Minimum Support = 2
Enter the Minimum Confidence (in %) = 30

No of Items = 5
No of Transactions = 7
Minimum Support = 2
Minimum Confidence = 0.3

'Items present in the Database'
------> iPhone
------> Samsung
------> Lenevo
------> Nokia
------> LG

TRANSACTION ITEMSET
Transaction 1 = iPhone Samsung Lenevo
Transaction 2 = Nokia iPhone Samsung LG Lenevo
Transaction 3 = Nokia Samsung
Transaction 4 = iPhone Samsung Nokia
Transaction 5 = Lenevo
Transaction 6 = Samsung LG
Transaction 7 = Samsung Lenevo
```

Knime:


Create Collection Column
Association Rule Learner
CSV Reader
Node 1
Node 2
Node 3

Input:


Dialog - 4:3 - Association Rule Learner
File
Options
Flow Variables
Memory Policy
Itemset Mining
Column containing transactions
AggregatedValues
Minimum support (0-1)
0.2
Underlying data structure:
ARRAY
Output
Itemset type
CLOSED
Maximal itemset length:
10
Association Rules
Output association rules
Minimum confidence:
0.7
OK
Apply
Cancel

Output:

Frequent itemsets/Association rules - 4:3 - Association Rule Learner

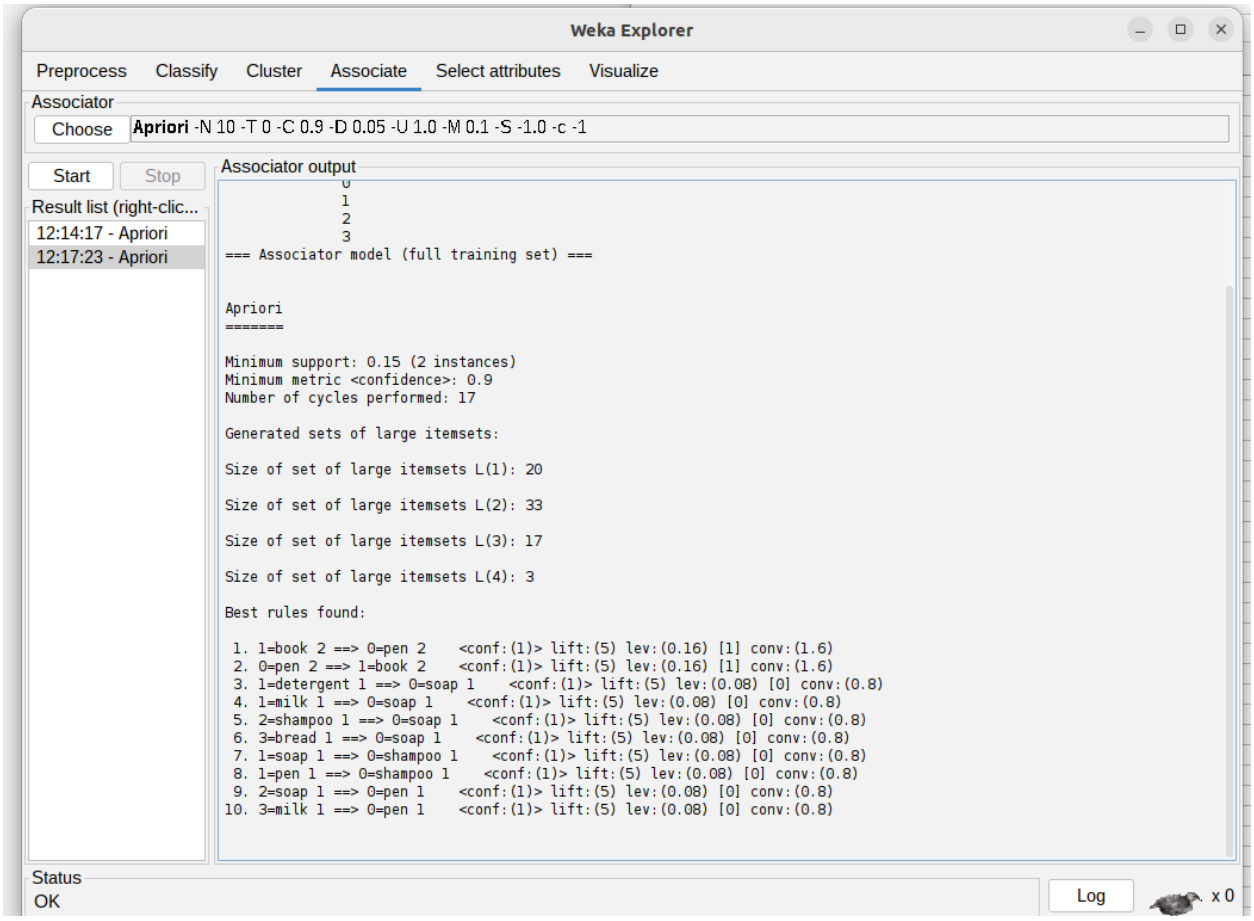
File Edit Hilite Navigation View

Table "default" - Rows: 13 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	D Support	D Confide...	D Lift	? Conseq...	S implies	[...] Items
rule0	0.2	1	1.429	?	<---	[detergent]
rule1	0.2	1	1.429	milk	<---	[rice]
rule2	0.2	1	3.333	shampoo	<---	[?,soap]
rule3	0.2	1	1.429	?	<---	[shampoo,s...
rule4	0.2	1	1.429	milk	<---	[pen,soap]
rule5	0.2	1	1.429	milk	<---	[bread,soap]
rule6	0.2	1	2	pen	<---	[bread,?]
rule7	0.2	1	2	pen	<---	[book,milk]
rule8	0.2	1	1.429	milk	<---	[book,pen]
rule9	0.3	1	1.429	?	<---	[shampoo]
rule10	0.3	0.75	1.5	pen	<---	[bread]
rule11	0.3	0.75	1.071	milk	<---	[bread]
rule12	0.4	0.8	1.143	milk	<---	[pen]

Weka:

	A	B	C	D	E
1	0	1	2	3	
2	soap	detergent	shampoo		
3	shampoo	soap			
4	pen	book	soap	milk	
5	milk	bread	pen		
6	soap	milk	pen	bread	
7	shampoo	pen	bread		
8	milk	rice	bread	soap	
9	detergent	rice	milk		
10	pen	book	milk		
11	milk				
12					



<u>Conclusion:</u> Successfully generated association rule from given dataset.

Experiment 8

Experiment No. 8: Find Co-relation between items / entities

Aim :- To find Co-relation between items / entities

Theory :-

Variables in a dataset can be dependent or independent on eachother.
It can be useful in data analysis and modify modeling to better understand the relationships between variables.
The statistical representation between 2 variables is referred to as their Correlation

Types of Correlations

1. perfect Negative Correlations
2. The variables tend to move in opp. direct
2. No Correlation.
   Variables do not have a relⁿ with each other.
3. perfect positive Correlatⁿ

There are 3 metrics that are commonly used to calculate Correlation

Algorithm.

(1) Calculate probability of each element individually.
(2) Calculate the probability of their Union
(3) Calculate Correl^n by formula

$$\frac{p(A \cup B)}{p(A) \times p(B)}$$

(4) find Covariance

$$Cov(x,y) = Sum(x - mean(x)) \times (y - mean(x)) \times \frac{1}{n-1}$$

Data:

Name	Height (cm)	Weight (kg)
Mr A	153	52
Mr B	160	55
Mr C	150	56
Mr D	152	50
Mr E	170	60
Mr F	165	62
Mr G	175	71

Mean of wt $= \frac{153 + 160 + 150 + 152 + 170 + 165 + 175}{7}$

$= 160.714$

Std. deviation of wt $= 8.940$

wt. mean $= 58.0$, std dev^n $= 6.56$

Covariance matrix $= \begin{bmatrix} 93.23 & 60.166 \\ 60.16 & 50.23 \end{bmatrix}$

Conclusion :-

(1) using covariance we conclude that heights & weight variables are not related.

Code:

```
import openpyxl
import math

wb=openpyxl.load_workbook("Experiment8.xlsx")
sheet_obj = wb.active
row = sheet_obj.max_row
col = sheet_obj.max_column

def std_dev(X,l,x_mean):
  ans=0
  for i in range(0,l):
    ans = ans + (X[i]-x_mean)*(X[i]-x_mean)
  return ans

X=[]
Y=[]
x_mean=0
y_mean=0
for i in range(2,row+1):
  val=sheet_obj.cell(row=i,column=2)
  val2=sheet_obj.cell(row=i,column=3)
  X.append(val.value)
  x_mean=x_mean+val.value
  Y.append(val2.value)
  y_mean=y_mean+val2.value

l=len(X)

x_mean=x_mean/l
y_mean=y_mean/l

cov=0
for i in range(0,l):
  cov=cov + (X[i]-x_mean)*(Y[i]-y_mean)


s = math.sqrt(std_dev(X,l,x_mean)*std_dev(Y,l,y_mean))

corelation = cov/s
print(corelation)
```

Input:

C8

	A	B	C	D
1	Year	X	Y	
2	2015	1245	100	
3	2016	1415	123	
4	2017	1312	129	
5	2018	1427	143	
6	2019	1510	150	
7	2020	1590	197	
8				
9				

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\DELL\Downloads\correlation> & C:/Users/DELL/AppData/Local/Programs/Python/Python39/py
Experiment8.py
  File "c:\Users\DELL\Downloads\correlation\Experiment8.py", line 4
    wb=openpyxl.load_workbook("C:\Users\DELL\Downloads\correlation\Experiment8.xlsx")
                                                                                    ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated
PS C:\Users\DELL\Downloads\correlation> & C:/Users/DELL/AppData/Local/Programs/Python/Python39/py
Experiment8.py
0.9157308063042234
PS C:\Users\DELL\Downloads\correlation>
```

EXP9

Walchand College of Engineering, Sangli.

Experiment No : 09 Distance & cluster.

Aim :- To form cluster by euclidean distance function.

Theory :-

Distance matrix is Symmetric matrix with values on diagonal 0 because distane for element from itself is always 0
Significance of distance matrix is to indicate magnitude of dissimilarity between two elements means higher the distance more they are likely to be dissimilar hence use fit to be clustered.
There are multiple distance functions.
The one used in this experiment is euclidian distane function.

Algorithm :-

(1) Assume one of the points to be center
(2) Calculate the distance of each point from this center.
(3) Create the distance matrix from obtained value.
(4) Assume new center.

(5) Calculate the distance of all points from this new center.

Data.

X	Y
53	34
35	87
68	34

Initial Center (x) $= \frac{53+35+68}{3} = 52$

Initial Center (y) $= \frac{34+87+34}{3} = 51$

Initial Center : (52, 51)

Distance of (53, 34) from (52, 51)

$= \sqrt{(53-52)^2 + (34-51)^2} = 17.03$

Distance of (35, 87) from (52, 51)

$= \sqrt{(35-52)^2 + (87-51)^2} = 39.812$

Distance of (68, 34) from (52, 51)

$= \sqrt{(68-52)^2 + (34-51)^2} = 23.345$

Minimum distance from center = 17.029
∴ New center = 53, 34

Distance matrix with new center

0	56	15
56	0	62
15	62	0

Weka output :-

leads to formation of 2 clusters.
1st cluster (53, 34), (68, 34)
2nd cluster (35, 87)

Conclusion :-
(1) Euclidean distance is calculated as sq. root of sum of squared differences b/w two vectors.
(2) Thus we have performed cluster by using Euclidean distance function.

Code:

```
import java.lang.*;
import java.util.*;
import java.io.*;

public class Clustering
{
	public static void main(String[] args) throws Exception
	{
		List<int[]> data=new ArrayList<>();
		data.add(new int[]{3,4});
		data.add(new int[] {13,27});
		data.add(new int[]{15,10});
		data.add(new int[] {20,30});

		int n=data.size();
		int sum_x=0;
		int sum_y=0;
		for(int i=0;i<n;i++)
		{
			sum_x+=data.get(i)[0];
			sum_y+=data.get(i)[1];
		}

		sum_x/=data.size();
		sum_y/=data.size();

		int cx=sum_x;
		int cy=sum_y;

		double[][] matrix=new double[data.size()+1][data.size()];
		for(int i=0;i<data.size();i++)
		{
			int x=data.get(i)[0];
			int y=data.get(i)[1];
			matrix[0][i]=Math.sqrt((x-cx)*(x-cx) + (y-cy)*(y-cy));
		}

		for(int i=0;i<data.size();i++)
		{
			int x=data.get(i)[0];
			int y=data.get(i)[1];
```

```
                for(int j=i;j<data.size();j++)
                {
                        int x1=data.get(j)[0];
                        int y1=data.get(j)[1];

                        matrix[i+1][j]=Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1));

                }
        }

        PrintWriter out=new PrintWriter(new File("cluster_out.csv"));
        out.printf("%s"," ");
        for(int i=0;i<data.size();i++)
        {
                out.printf("%s","P"+(i+1));
        }

        out.printf("\n");


        for(int i=0;i<=data.size();i++)
        {
                if(i==0)
                        out.printf("%s","C");
                else
                        out.printf("%s","P"+i);
                for(int j=0;j<data.size();j++)
                {
                        out.printf("%.3f",matrix[i][j]);
                }
                out.printf("\n");
        }
        out.close();
    }
}
```

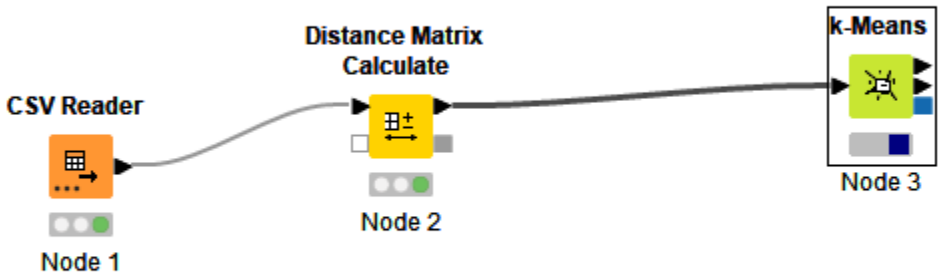
Input:

Output:

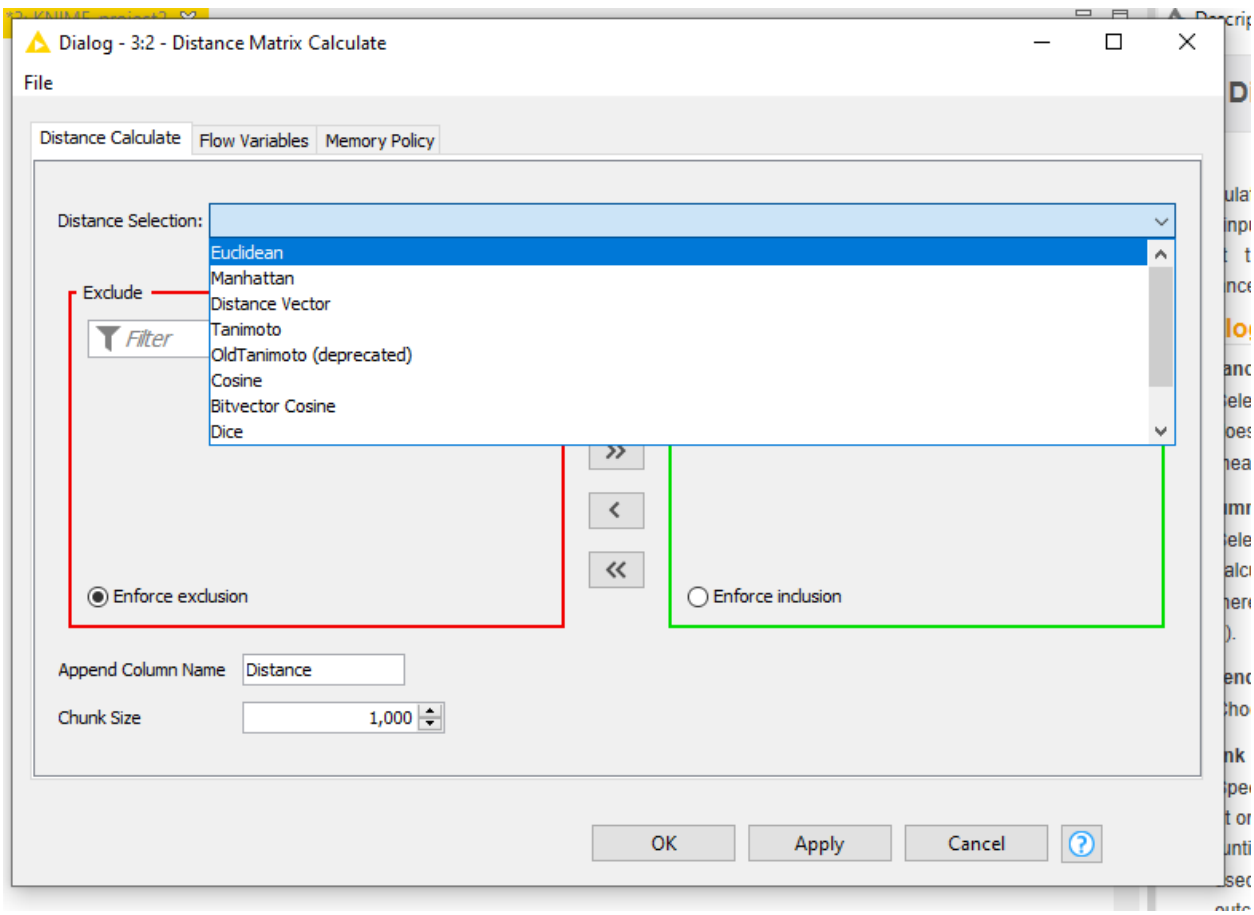
cluster_out.csv

```
,P1,P2,P3,
C,17.029,39.812,23.345,
P1,0.000,55.973,15.000,
P2,0.000,0.000,62.434,
P3,0.000,0.000,0.000,
```

Knime:


CSV Reader
Node 1
Distance Matrix Calculate
Node 2
k-Means
Node 3

Input:


Dialog - 3:2 - Distance Matrix Calculate
File
Distance Calculate
Flow Variables
Memory Policy
Distance Selection:
Euclidean
Manhattan
Distance Vector
Tanimoto
OldTanimoto (deprecated)
Cosine
Bitvector Cosine
Dice
Exclude
Filter
Enforce exclusion
Enforce inclusion
Append Column Name
Distance
Chunk Size
1,000
OK
Apply
Cancel

Output:

Labeled input - 3:3 - k-Means

File Edit Hilite Navigation View

Table "default" - Rows: 3 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	x	y	Distance	Cluster
Row0	53	34	0 []	cluster_1
Row1	35	87	1 [55.97320787662612]	cluster_0
Row2	68	34	2 [15.0, 62.433965115151864]	cluster_1

Experiment No: 10

Walchand College of Engineering, Sangli.

Experiment No: 10 : Agglomerative Hierarchial clustering.

Aim :- To write a program for agglomerative hierarchial clustering using Single Linkeage method.

Theory:

→ There are 2 main types of Hierarchial clustering
- Agglomerative
- Divise

* Agglomerative clustering:-
- Initially each object is Considered to be its own cluster. According to a particular procedure, the clusters are then merged step by step until a Single cluster remains. At the end of the cluster merging process, a cluster Containing all the elements will be formed.

* Single linkage:
- One way to measure the distance between clusters is to find the the minimum distance between points

in those clusters. That is, we can find the point in the first cluster nearest to a point in the other cluster & calculate the distance between those points

Algorithm :-

Step1 :- Compute the proximity matrix using a particular distance matrix.

Step2 :- Each data point is assigned to a cluster

Step3 :- Merge the clusters based on a metric for similarity between clusters.

Step4 :- Update the distance matrix.

Step5 :- Repeat step3 & step4 until only a single cluster remains.

Data

X	Y
4	5
2	2
3	3
6	9
8	4

Distance matrix for initial data

	A	B	C	D	E
A	0				
B	13	0			
C	5	$\boxed{2}$	0		
D	20	65	45	0	
E	17	40	26	29	0

minimum value = 2

merging B & C

	A	(B,C)	D	E
A	0			
(B,C)	$\boxed{5}$	0		
D	20	45	0	
E	17	26	29	0

minimum value = 5

merging (A, (B,C))

	(A,(B,C))	D	E
(A,(B,C))	0		
D	20	0	
E	$\boxed{17}$	29	0

minimum value = 17

merging ((A, (B, C)), E)

	((A, (B, C)), E)	D
((A, (B, C)), E)	0	
D	20	0

merging 0 we get final cluster as

(((A, (B, C)), E), D)

Knime output:

RowID	X	Y	cluster
0	4	5	cluster_0
3	6	9	cluster_1
1	2	2	cluster_2
2	3	3	cluster_2
4	8	4	cluster_2

Code:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> calmin(vector<vector<double>>arr)
{
    int tr,tc;
    vector<int>v;
    double min1=arr[1][0];
    int cnt=0;
   // cout<<arr.size()<<"hi"<<endl;

       for(int i=0;i<arr.size();i++)
       {
           for(int j=i+1;j<arr[i].size();j++)
           {
               if(min1>arr[i][j])
               {
                   min1=arr[i][j];
                   tr=j;
                   tc=i;
               }
               else if(min1==arr[i][j])
               {
                   if(cnt==0)
                   {
                       tr=j;
                       tc=i;
                   }
                   cnt++;
               }
           }
       }
       cout<<min1<<endl;
    v.push_back(tr);
    v.push_back(tc);
    return v;

}
int main()
{
    fstream fin("points.csv",ios::in);
    vector<vector<string> > content;
    vector<string> row;
```

```
string line, word;
if(fin.is_open())
{
    while(getline(fin,line))
    {
       // cout<<line<<endl;
        row.clear();
        stringstream s(line);
        while(getline(s,word,','))
        {
                row.push_back(word);
        }
        content.push_back(row);
    }
}
vector<vector<double>>arr(content.size(),vector<double>(content.size(),0.0));
// vector<vector<double>>arr{{0,0.71,5.66,3.61,4.24,3.20},
// {0.71,0,4.95,2.92,3.54,2.50},
// {5.66,4.95,0,2.24,1.41,2.50},
// {3.61,2.92,2.24,0,1.00,0.50},
// {4.24,3.54,1.41,1.00,0,1.12},
// {3.20,2.50,2.50,0.50,1.12,0}};
for(int i=0;i<arr.size();i++)
{
    for(int j=0;j<arr.size();j++)
    {
        cout<<arr[i][j]<<" ";
    }
    cout<<endl;
}
double arr1[content.size()][content.size()];
for(int i=0;i<content.size();i++)
{
    for(int j=i+1;j<content.size();j++)
    {
        double ans=0;
        for(int k=0;k<content[0].size();k++)
        {
           ans+=pow(stoi(content[i][k])-stoi(content[j][k]),2);
        }
       // cout<<ans<<" ";
        ans=sqrt(ans);
        arr[i][j]=ans;
        arr[j][i]=ans;
```

```
    }
}
vector<string>v(arr.size(),"");
for(int i=0;i<v.size();i++)
{
    v[i]=to_string(i);
}
int tr,tc;
double min1=arr[1][0];
while(arr.size()!=2)
{
    vector<int>pos=calmin(arr);
    tr=pos[0];
    tc=pos[1];
    if(tr<tc)
    {
        v[tr]="("+v[tr]+","+v[tc]+")";
        v.erase(v.begin()+tc);
    }
    else
    {
        v[tc]="("+v[tc]+","+v[tr]+")";
        v.erase(v.begin()+tr);
    }
   //cout<<tr<<tc<<endl;
   //break;
   for(int i=0;i<arr.size();i++)
   {
           if(tr<tc && tr!=i)
           {
               arr[tr][i]=min(arr[tr][i],arr[tc][i]);
               arr[i][tr]=arr[tr][i];
           }
           else if(tc<tr && tc!=i)
           {
                arr[tc][i]=min(arr[tr][i],arr[tc][i]);
               arr[i][tc]=arr[tc][i];
           }
   }
   if(tr<tc)
   {
     arr.erase(arr.begin()+tc);
      for(int i=0;i<arr.size();i++)
        {
```

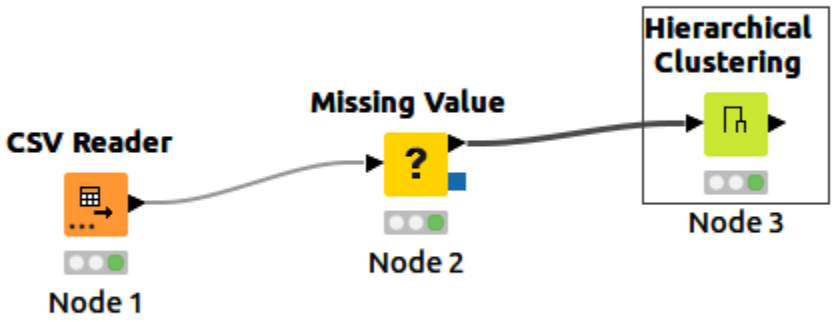
```
            arr[i].erase(arr[i].begin()+tc);
        }
    }
    else
    {
        arr.erase(arr.begin()+tr);
        for(int i=0;i<arr.size();i++)
        {
            arr[i].erase(arr[i].begin()+tr);
        }
    }
    for(int i=0;i<arr.size();i++)
    {
     for(int j=0;j<arr.size();j++)
     {
        cout<<arr[i][j]<<" ";
     }
     cout<<endl;
    }

  }
  v[0]="("+v[0]+","+v[1]+")";
  cout<<v[0];

}
```

Output:

```
PS E:\DM Assignment> cd "e:\DM Assignment\" ; if ($?) { g++ AgglomerativeCluste
 { .\AgglomerativeClustering }
0
0 3.4641 8.3666 3.74166 5.47723 4.47214 3.4641 5.74456 3 5.91608
3.4641 0 8.60233 6.48074 8.60233 7.48331 5.65685 8.06226 4.12311 8.66025
8.3666 8.60233 0 8.24621 10.7703 9.48683 7.07107 5.91608 9.11043 9.43398
3.74166 6.48074 8.24621 0 2.82843 1.41421 1.41421 3.31662 3.31662 2.23607
5.47723 8.60233 10.7703 2.82843 0 1.41421 4.24264 5.19615 5.19615 2.23607
4.47214 7.48331 9.48683 1.41421 1.41421 0 2.82843 4.12311 4.12311 1.73205
3.4641 5.65685 7.07107 1.41421 4.24264 2.82843 0 3 3 3.31662
5.74456 8.06226 5.91608 3.31662 5.19615 4.12311 3 0 6 3.74166
3 4.12311 9.11043 3.31662 5.19615 4.12311 3 6 0 5.09902
5.91608 8.66025 9.43398 2.23607 2.23607 1.73205 3.31662 3.74166 5.09902 0
1.41421
0 3.4641 8.3666 3.74166 5.47723 3.4641 5.74456 3 5.91608
3.4641 0 8.60233 6.48074 8.60233 5.65685 8.06226 4.12311 8.66025
8.3666 8.60233 0 8.24621 10.7703 7.07107 5.91608 9.11043 9.43398
3.74166 6.48074 8.24621 0 1.41421 1.41421 3.31662 3.31662 1.73205
5.47723 8.60233 10.7703 1.41421 0 4.24264 5.19615 5.19615 2.23607
3.4641 5.65685 7.07107 1.41421 4.24264 0 3 3 3.31662
5.74456 8.06226 5.91608 3.31662 5.19615 3 0 6 3.74166
3 4.12311 9.11043 3.31662 5.19615 3 6 0 5.09902
5.91608 8.66025 9.43398 1.73205 2.23607 3.31662 3.74166 5.09902 0
1.41421
0 3.4641 8.3666 3.74166 3.4641 5.74456 3 5.91608
3.4641 0 8.60233 6.48074 5.65685 8.06226 4.12311 8.66025
8.3666 8.60233 0 8.24621 7.07107 5.91608 9.11043 9.43398
3.74166 6.48074 8.24621 0 1.41421 3.31662 3.31662 1.73205
3.4641 5.65685 7.07107 1.41421 0 3 3 3.31662
5.74456 8.06226 5.91608 3.31662 3 0 6 3.74166
3 4.12311 9.11043 3.31662 3 6 0 5.09902
5.91608 8.66025 9.43398 1.73205 3.31662 3.74166 5.09902 0
```

```
1.41421
0 3.4641 8.3666 3.4641 5.74456 3 5.91608
3.4641 0 8.60233 5.65685 8.06226 4.12311 8.66025
8.3666 8.60233 0 7.07107 5.91608 9.11043 9.43398
3.4641 5.65685 7.07107 0 3 3 1.73205
5.74456 8.06226 5.91608 3 0 6 3.74166
3 4.12311 9.11043 3 6 0 5.09902
5.91608 8.66025 9.43398 1.73205 3.74166 5.09902 0
1.73205
0 3.4641 8.3666 3.4641 5.74456 3
3.4641 0 8.60233 5.65685 8.06226 4.12311
8.3666 8.60233 0 7.07107 5.91608 9.11043
3.4641 5.65685 7.07107 0 3 3
5.74456 8.06226 5.91608 3 0 6
3 4.12311 9.11043 3 6 0
3
0 3.4641 8.3666 3 5.74456
3.4641 0 8.60233 5.65685 8.06226
8.3666 8.60233 0 7.07107 5.91608
3 5.65685 7.07107 0 3
5.74456 8.06226 5.91608 3 0
3
0 3.4641 7.07107 3
3.4641 0 8.60233 8.06226
7.07107 8.60233 0 5.91608
3 8.06226 5.91608 0
3
0 3.4641 5.91608
3.4641 0 8.60233
5.91608 8.60233 0
3.4641
0 5.91608
5.91608 0
(((((0,8),((((3,(5,10)),4),6),9)),7),1),2)
```

Knime output:



File Table

File Edit Hilite Navigation View

Table "default" - Rows: 5 | Spec - Columns: 6 | Properties | Flow Variables

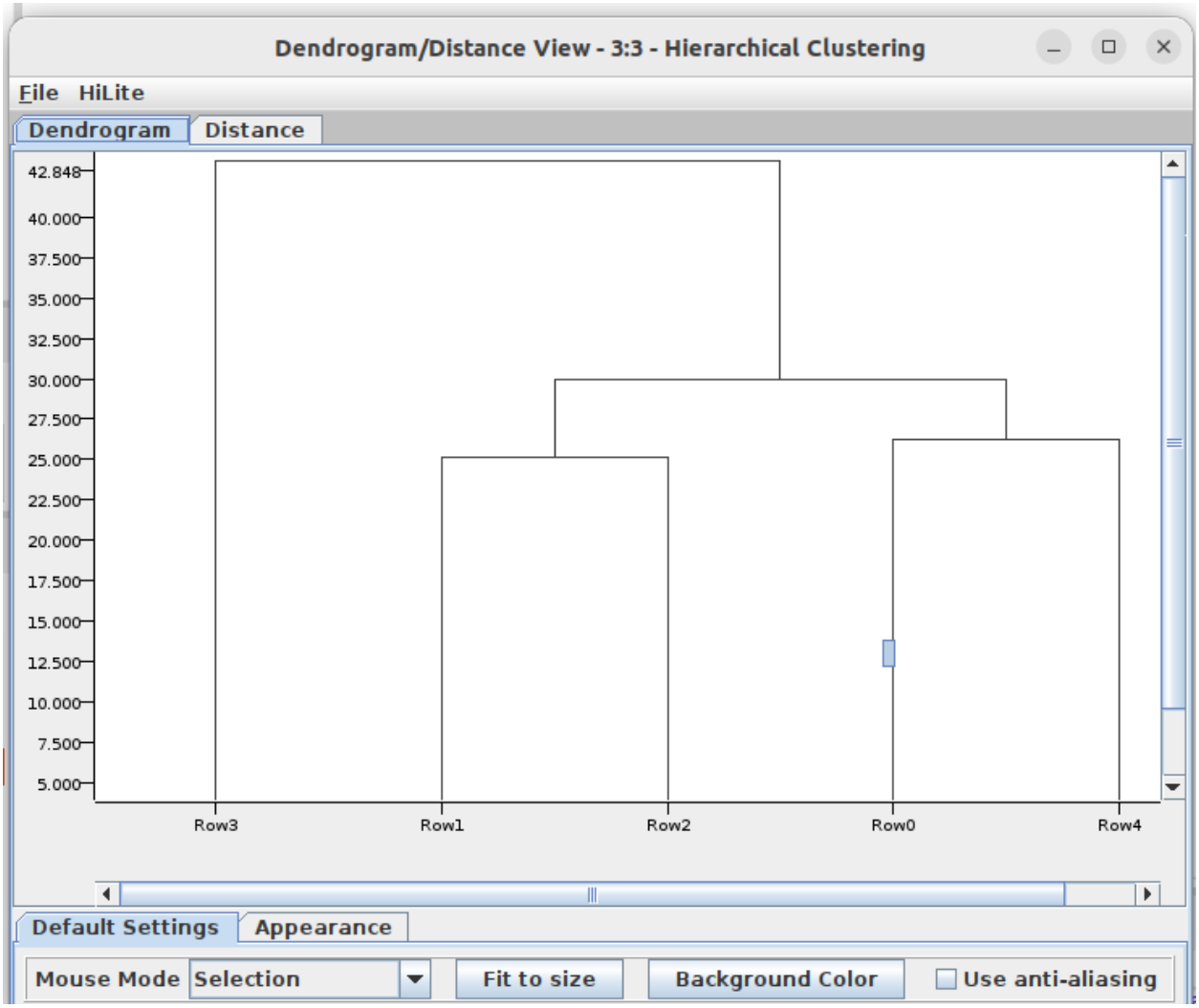
Row ID	S x	I A	I B	I C	I D	I E
Row0	A	0	?	?	?	?
Row1	B	13	0	?	?	?
Row2	C	5	2	0	?	?
Row3	D	20	65	45	0	?
Row4	E	17	40	26	29	0

Output tabl

File Edit Hilite Navigation View

Table "default" - Rows: 5 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	S x	D A	D B	D C	D D	D E
Row0	A	0	26.75	23.667	14.5	0
Row1	B	13	0	23.667	14.5	0
Row2	C	5	2	0	14.5	0
Row3	D	20	65	45	0	0
Row4	E	17	40	26	29	0



Experiment No:11

Walchand College of Engineering, Sangli.

Experiment No: 11 Gini Index

Aim :- To find gini Index of given data

Theory:

Entropy :-
- Entropy is the degree of uncertainity impurity or disorder of a random variable or a measure of impurity.

Information gain
- Information gain is used for determing the best features / attributes that render maximum info about a class
- Information gain Computes the difference between entropy before and after split and Specifies the impurity in class elements

Gain = Total Entropy - Entropy of attribute

Gini (minority)

	14	Scholarship	No Scholarship
Yes	6/14	5/6	3/6
No	8/14	4/8	4/8

Gini (minority)

$$= \frac{6}{14}\left[1-\left(\frac{5}{6}\right)^2-\left(\frac{3}{6}\right)^2\right]+\frac{8}{14}\left[1-\left(\frac{4}{8}\right)^2-\left(\frac{4}{8}\right)^2\right]$$

$$= 0.405$$

Gini (Income)

	14	Scholarship	No scholarship
Rich	7/14	2/7	5/7
poor	7/14	7/7	0

Gini (Income)

$$= \frac{7}{14}\left[1-\left(\frac{2}{7}\right)^2-\left(\frac{5}{7}\right)^2\right]+\left[\frac{7}{14}\right]\left[1-\left(\frac{7}{7}\right)^2-0\right]$$

$$= 0.204$$

Gini (marks)

	14	Scholarship	No scholarship
Average	8/14	5/8	3/8
Excellent	6/14	4/8	4/8

Gini (marks)

$$= \frac{8}{14}\left[1-\left(\frac{5}{8}\right)^2-\left(\frac{3}{8}\right)^2\right]+\frac{6}{14}\left[1-\left(\frac{4}{8}\right)^2-\left(\frac{4}{8}\right)^2\right]$$

$= 0.204$

$\therefore$ The Gini Value is minimum for income class

$\therefore$ Selected attribute = Income

Algorithm

(1) find the attribute classes.

(2) for each unique value find its corresponding class value.

(3) find the gini index by the formula

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

(4) find the minimum gini value.

* Information Gain for minority

	14	Scholarship	No scholarship
Yes	6/14	5/6	3/6
No	8/14	4/8	4/8

$$\text{InfoGain (minority)} = \text{Total Entropy} - \sum_{\text{values}} \text{Entropy} \left| \frac{S_v}{S} \right|$$

$$= \frac{-9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} + \frac{6}{14}\left[\frac{-5}{6} \log \frac{5}{6} - \frac{3}{6} \log \frac{3}{6}\right]$$

$$+ \frac{8}{14}\left[\frac{-4}{8} \log \frac{4}{8} - \frac{4}{8} \log \frac{4}{8}\right]$$

$$= 0.09$$

* Information gain for Income

	14	Scholarship	No scholarship
Rich	7/14	2/7	5/7
poor	7/14	7/7	0

$$\text{Gain (Income)} = \text{Total Entropy} - \text{Entropy (Rich)} - \text{Entropy (poor)}$$

$$= \frac{-9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} + \frac{7}{14}\left[\frac{-2}{7} \log \frac{2}{7} - \frac{5}{7} \log \frac{5}{7}\right]$$

$$+ \frac{7}{14}\left[-\log \frac{7}{7} - \log 0\right] = 0.508$$

* Information Gain of marks

	14	Scholarship	No. of scholarship
Average	8/14	5/8	3/8
Excellent	6/14	4/8	4/8

Gain (marks) = Total Entropy − Entropy (Avg) − Entropy (Excellent)

$$= -\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14} + \frac{8}{14}\left[-\frac{5}{8}\log\frac{5}{8} - \frac{3}{8}\log\frac{3}{8}\right]$$

$$+ \frac{6}{14}\left[-\frac{4}{8}\log\frac{4}{8} - \frac{4}{8}\log\frac{4}{8}\right]$$

$= 0.0013$

Gain for minority = 0.09
Gain for Income = 0.508
Gain for marks = 0.0013

∴ Selected attribute from information gain = Income.

Knime output :-

Scholarship
Yes (9/14)

	%	n
no	35.7	5
Yes	64.3	9
Total	100	14

↓

Income

Rich

	%	n
no	71.4	5
yes	28.6	2

poor

	%	n
no	0	0
yes	100	7

Selected attribute = Income

Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.*;
class GiniIndex {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new FileReader("ginni1.csv"));
        ArrayList<ArrayList<String>> data = new ArrayList<>();
        String[] dt = br.readLine().trim().split(",");


        int n=dt.length;
        while (true) {
            String str[] = new String[n];
            try {
                str = br.readLine().trim().split(",");
               // System.out.println(str[0] + " " + str[1]+ ""+ str[2]+ str[3]+str[4]);
            } catch (NullPointerException e) {
                break;
            }
            ArrayList<String> temp = new ArrayList<>();
            for (String s : str)
                temp.add(s);

            data.add(temp);
            //System.out.println(data);


        }

        int C0 = 0, C1 = 0;
        for (int i = 0; i < data.size(); i++) {
            if (data.get(i).get(4).equals("C0"))  C0++;
            else
                C1++;
        }
        System.out.println("Total Possiblity of C0 : "+C0  +  "  Total Possiblity of C1 : "+C1);
        double totRecord = data.size();
        double GineIndex = 1.0-Math.pow((C0 / totRecord),2)  - Math.pow( (C1 / totRecord),2)  ;
        System.out.println("Gine Index of C0 :"+GineIndex);
        System.out.println("==========================================================");
        for(int i=1;i<data.get(0).size()-1;i++){
```

```
            System.out.println("GineIndex Of "+ dt[i]+" :"+
String.format("%.5f",findAllGineIndex(i,GineIndex,data,dt)));

System.out.println("==========================================================");
        }
    }

    private static double findAllGineIndex(int i, double GineIndex, ArrayList<ArrayList<String>> data,
String[] dt) {
        Set<String> attribute=new HashSet<>();
        for(ArrayList<String>x:data)
            attribute.add(x.get(i));
        Map<String,double[]>total=new HashMap<>();
        for(String x:attribute){
            total.put(x,new double[2]);



        }
        for(ArrayList<String> x:data){
            //System.out.println("+++"+x);
            if(x.get(4).equals(("C0")))
                total.get(x.get(i))[0]++;
            else
                total.get(x.get(i))[1]++;

        }
        double totalE=0.0;
        System.out.println("Count Table for  :"+dt[i]);
        System.out.println(dt[i]+" C0   C1 ");
        for(Map.Entry<String,double[]> x:total.entrySet()){
            double total1=x.getValue()[0]+x.getValue()[1];
            System.out.println(x.getKey()+"  "+x.getValue()[0]+"  "+x.getValue()[1]);
            if(x.getValue()[0]==0||x.getValue()[1]==0)
                continue;
            double temp= 1-Math.pow((x.getValue()[0] / total1),2) - Math.pow((x.getValue()[1]/ total1),2);
            //System.out.println(temp);
            totalE+=(total1/ data.size())*temp;



        }

       // double finalE=GineIndex-totalE;
        return (totalE);
    }
```

}

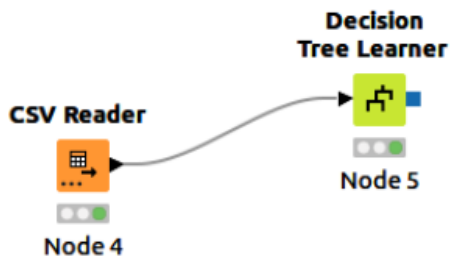
Input csv file:

	A	B	C	D	E	F
1	CustId	Gender	CarType	ShirtSize	Class	
2	1	M	Family	Small	C0	
3	2	M	Sports	Medium	C0	
4	3	M	Sports	Medium	C0	
5	4	M	Sports	Large	C0	
6	5	M	Sports	Extra large	C0	
7	6	M	Sports	Extra large	C0	
8	7	F	Sports	Small	C0	
9	8	F	Sports	Small	C0	
10	9	F	Sports	Medium	C0	
11	10	F	Luxury	Large	C1	
12	11	M	Family	Large	C1	
13	12	M	Family	Extra large	C1	
14	13	M	Family	Medium	C1	
15	14	M	Luxury	Extra large	C1	
16	15	F	Luxury	Small	C1	
17	16	F	Luxury	Small	C1	
18	17	F	Luxury	Medium	C1	
19	18	F	Luxury	Medium	C1	
20	19	F	Luxury	Medium	C1	
21	20	F	Luxury	Large	C1	
22						

Output:

```
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$ javac GiniIndex.java
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$ java GiniIndex
Total Possiblity of C0 : 9  Total Possiblity of C1 : 11
Gine Index of C0 :0.49499999999999994
=========================================================
Count Table for  :Gender
Gender C0   C1
F  3.0  7.0
M  6.0  4.0
GineIndex Of Gender :0.45000
=========================================================
Count Table for  :CarType
CarType C0   C1
Luxury  0.0  8.0
Family  1.0  3.0
Sports  8.0  0.0
GineIndex Of CarType :0.07500
=========================================================
Count Table for  :ShirtSize
ShirtSize C0   C1
Small  3.0  2.0
Medium  3.0  4.0
Large  1.0  3.0
Extra large  2.0  2.0
GineIndex Of ShirtSize :0.46643
=========================================================
shubham@shubham-Vostro-3405:~/Desktop/DMLABS/DM codes$
```

KNIME:



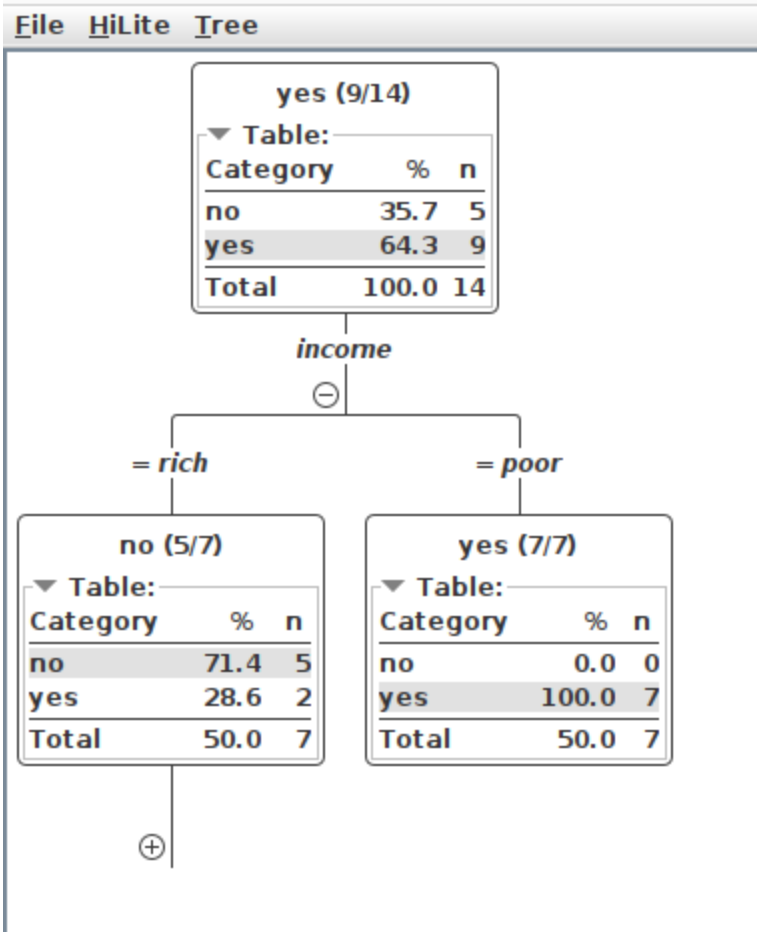
Input:

File Edit Hilite Navigation View

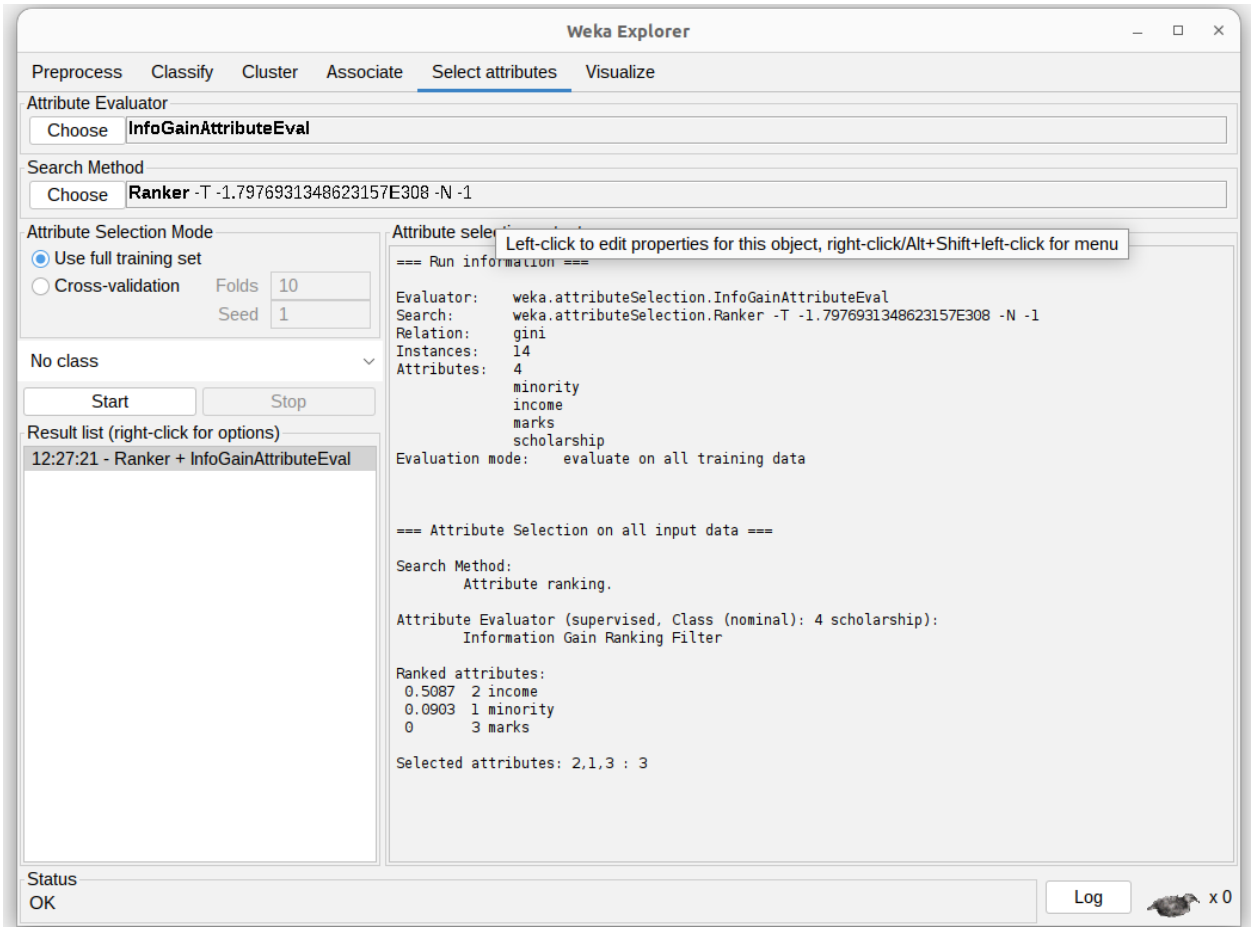
Table "default" - Rows: 14 | Spec - Columns: 4 | Properties | F

Row ID	S minority	S income	I marks	S schola...
Row0	yes	rich	53	no
Row1	no	rich	67	no
Row2	no	poor	80	yes
Row3	yes	poor	48	yes
Row4	yes	rich	72	yes
Row5	no	poor	64	yes
Row6	yes	poor	91	yes
Row7	no	poor	88	yes
Row8	no	rich	56	no
Row9	no	rich	63	no
Row10	yes	rich	93	yes
Row11	no	rich	87	no
Row12	yes	poor	58	yes
Row13	no	poor	71	yes

Output:


File HiLite Tree
yes (9/14)
Table:
Category % n
no 35.7 5
yes 64.3 9
Total 100.0 14
income
= rich
= poor
no (5/7)
Table:
Category % n
no 71.4 5
yes 28.6 2
Total 50.0 7
yes (7/7)
Table:
Category % n
no 0.0 0
yes 100.0 7
Total 50.0 7

Weka:


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Attribute Evaluator
Choose
InfoGainAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds 10
Seed 1
No class
Start
Stop
Result list (right-click for options)
12:27:21 - Ranker + InfoGainAttributeEval
Left-click to edit properties for this object, right-click/Alt+Shift+left-click for menu
=== Run information ===
Evaluator: weka.attributeSelection.InfoGainAttributeEval
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation: gini
Instances: 14
Attributes: 4
minority
income
marks
scholarship
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 4 scholarship):
Information Gain Ranking Filter
Ranked attributes:
0.5087 2 income
0.0903 1 minority
0 3 marks
Selected attributes: 2,1,3 : 3
Status
OK
Log
x 0

Weka output:-

Information Gain as Calculated on Weka

Ranked Gain	Attributes Attributes
0.50873	Income
0.09028	minority
0.00134	marles

Conclusion:-

- The attribute selected by Calculating gini index = Income
- The attribute selected by infogain on weka = Income
- Thus, the splitting attribute is selected.

Exp12

Walchand College of Engineering, Sangli.

Experiment No: 12 Bayes classification.

Aim :- Write a program for Bayes classification.

Theory :-

- Bayes classification uses Bayes algorithm to predict the Occurrance of any event
- Bayesian classifiers are statistical classifiers with the bayesian probability understandings.

It is expressed as :

$$P\left(\frac{X}{Y}\right) = \frac{P\left(\frac{Y}{X}\right) P(X)}{P(Y)}$$

$P\left(\frac{X}{Y}\right)$ is the conditional probability that describes the occurrence of event X is given that Y is true.

$P(X)$ & $P(Y)$ are marginal probabilites of observing X & Y independently of each other.

Algorithm :

(1) find the individual marginal probabilities
(2) find the conditional probabilities for each.
(3) use the formula to find the required probability.

Data :-

	outlook	play
0	Rainy	yes
1	Sunny	yes
2	Overcast	yes
3	Overcast	yes
4	Sunny	No
5	Rainy	yes
6	Sunny	yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	yes

Weather	No	Yes	
Overcast	0	5	5/14 = 0.35
Rainy	2	2	4/14 = 0.29
Sunny	2	3	5/14 = 0.35
All	4/14 = 0.29	10/14 = 0.71	

Applying Bayes theorem

$$P\left(\frac{Yes}{Sunny}\right) = \frac{P\left(\frac{Sunny}{Yes}\right) \times P(yes)}{P(Sunny)}$$

$$P\left(\frac{Sunny}{Yes}\right) = \frac{3}{10} = 0.3$$

$$P(Sunny) = 0.35$$

$$P(yes) = 0.71$$

$$P\left(\frac{Yes}{Sunny}\right) = \frac{0.3 \times 0.71}{0.35} = 0.60$$

$$P\left(\frac{No}{Sunny}\right) = \frac{P\left(\frac{Sunny}{No}\right) \times P(No)}{P(Sunny)}$$

Weka output

Attribute	class	
	yes	no
	0.69	0.31

outlook

Sunny	2	3
Overcast	0	5
Total		

Unique items and their support

Rice	0.2
pen	0.5
Book	0.2
Bread	0.4
milk	0.7
Shampoo	0.4
soap	0.5
Detergent	0.3

Support for 2 items:

milk, Rice	0.2
milk, Bread	0.3
milk, Book	0.2
milk, pen	0.4
milk, Detergent	0.2
milk, Soap	0.3
Rice, Bread	0.1
Rice, Book	0.1
Rice, pen	0.1
Rice, Detergent	0.2
Rice, Soap	0.1

Program:

```
// Bayes classifier
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>
#include <ostream>
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string line, word;
    ifstream file("infogain_cpp.csv");
    string day, outlook, three, four, five, six;
    map<string, double> parent;
    map<string, map<string, map<string, double>>> child;
    int count = 0;
    vector<string> title;
    if (file.is_open())
    {
        int i = 0;

        while (file >> line)
        {
            stringstream str(line);
            if (i == 0)
            {
                string heading;
                while (getline(str, heading, ','))
                {
                    title.push_back(heading);
                }
                i++;
                continue;
            }
            vector<string> columns;
            while (getline(str, day, ','))
            {
                columns.push_back(day);
            }

            int n = columns.size();
```

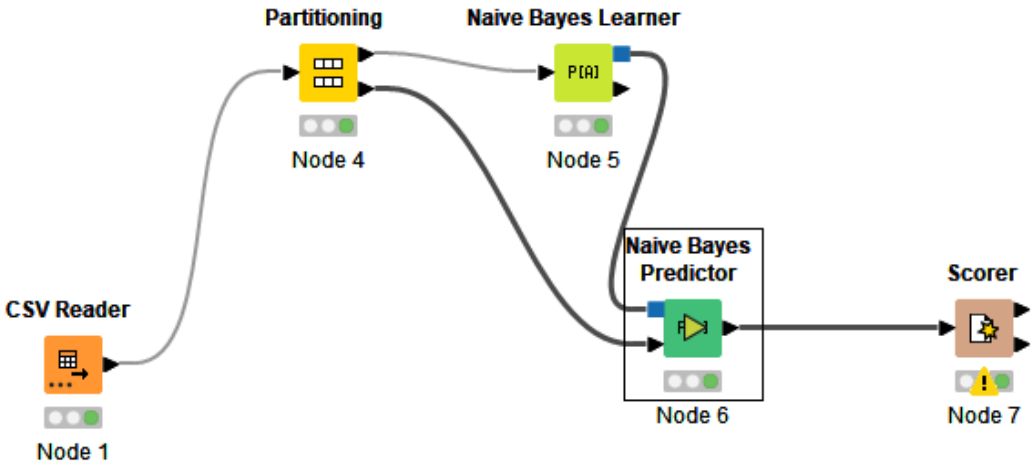
```
    parent[columns[n - 1]]++;
    for (int i = 1; i < n - 1; i++)
    {

        child[title[i]][columns[i]][columns[n - 1]]++;
    }

    count++;
}

vector<string> resultclass;
for (auto it : parent)
{
    resultclass.push_back(it.first);
}

vector<double> output(resultclass.size(), 1);
for (auto it : child)
{
    string input;
again:
    cout << "Enter " << it.first << " condition \n";
    cin >> input;

    auto curr = child[it.first].find(input);
    if (curr == child[it.first].end())
    {
        cout << "no match\n";
        goto again;
    }
    for (int i = 0; i < resultclass.size(); i++)
    {

        cout << child[it.first][input][resultclass[i]] << " / " << parent[resultclass[i]] << endl;
        double val = child[it.first][input][resultclass[i]] / parent[resultclass[i]];
        output[i] *= val;

        cout << output[i] << endl;
    }
}

for (int i = 0; i < resultclass.size(); i++)
{
```

```
            output[i] *= parent[resultclass[i]] / count;
        }
        double sum = accumulate(output.begin(), output.end(), 0.0f);
        cout << "sum " << sum << endl;
        cout << "output---------" << endl;
        for (int i = 0; i < resultclass.size(); i++)
        {
            cout << resultclass[i] << " " << output[i] << endl;
            cout << "Percentage " << (output[i] / sum) * 100 << endl;
        }
    }
    else
    {
        cout << "Could not open the file\n";
    }

    return 0;
}
```

Output:

```
Enter Humidity condition
High
4 / 5
0.8
3 / 9
0.333333
Enter Outlook condition
Rain
2 / 5
0.32
3 / 9
0.111111
Enter Temp condition
Hot
2 / 5
0.128
2 / 9
0.0246914
Enter Wind condition
Weak
2 / 5
0.0512
6 / 9
0.0164609
sum 0.0288677
output---------
No 0.0182857
Percentage 63.3431
Yes 0.010582
Percentage 36.6569
```

Knime:



Input:

File Table - 3:1 - CSV Reader

File Edit Hilite Navigation View

Table "default" - Rows: 15 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	No	Outlook	Play
Row0	0	rainy	yes
Row1	1	sunny	yes
Row2	2	overcast	yes
Row3	3	overcast	yes
Row4	4	sunny	no
Row5	5	rainy	yes
Row6	6	sunny	yes
Row7	7	overcast	yes
Row8	8	rainy	no
Row9	9	sunny	no
Row10	10	sunny	yes
Row11	11	rainy	no
Row12	12	overcast	yes
Row13	13	overcast	yes
Row14	?	?	?

Output:

The classified data - 3:6 - Naive Bayes Predictor

File Edit Hilite Navigation View

Table "default" - Rows: 5 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	I No	S Outlook	S Play	S Predic...
Row0	0	rainy	yes	yes
Row2	2	overcast	yes	yes
Row8	8	rainy	no	no
Row13	13	overcast	yes	yes
Row14	?	?	?	yes

Weka

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier

Choose NaiveBayes

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 66

More options...

(Nom) Play

Start Stop

Result list (right-click for options)

12:39:42 - bayes.NaiveBayes

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     bayes12th
Instances:    14
Attributes:   3
              No
              Outlook
              Play
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

Naive Bayes Classifier

                 Class
Attribute          yes     no    no
                (0.65) (0.24) (0.12)
=====================================
No
  mean             5.9 7.6667      9
  std. dev.     4.3463 2.8674 0.1667
  weight sum        10      3      1
  precision          1      1      1

Outlook
  rainy            3.0    3.0    1.0
  sunny            4.0    2.0    2.0
  overcast         6.0    1.0    1.0
  [total]         13.0    6.0    4.0


Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds
```

Status

OK

Log x 0

Bread, Book	0.1
Bread, pen	0.2
Bread, Detergent	0
Bread, Soap	0.1

Calculating the Confidence using

$$\text{Confidence } (A \rightarrow B) = \frac{\text{Support } (A \cup B)}{\text{Support } (A)}$$

$$\text{Confidence } (\text{Rice} \rightarrow \text{milk}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Rice} \rightarrow \text{Detergent}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Book} \rightarrow \text{pen}) = \frac{0.2}{0.2} = 1$$

$$\text{Confidence } (\text{Bread} \rightarrow \text{pen}) = \frac{0.3}{0.4} = 0.75$$

$$\text{Confidence of } (\text{pen} \rightarrow \text{Bread}) = \frac{0.3}{0.5} = 0.6$$

$$\text{Confidence of } (\text{Book} \rightarrow \text{Milk}) = \frac{0.4}{0.5} = 0.8$$

Weka output.

Best Rules found:

1. Book → pen
2. pen → Book
3. Detergent → soap
4. Milk → soap
5. Shampoo → Soap
6. Bread → soap
7. Shampoo → Soap
8. Soap → shampoo
9. pen → shampoo
10. Soap → pen

Conclusion:

(1) Association Rule learning is a rule-based machine learning method for discovering interesting relations between variables in larger database

TOP SUV

Walchand College of Engineering, Sangli.

Top suv data - october 22

No	OEM	Model	oct 22	oct 21	YoY %
1	Tata	Nexon	13767	10096	36
2	Hyundai	Creta	11880	6455	84
3	Maruti	Brezza	9941	8032	24
4	Kia	Sellos	9777	10488	-7
5	Hyundai	Venue	9585	10554	-9
6	Maruti	Vitara	8052		
7	Kia	Sonet	7614	5443	40
8	M&M	Scorpio	7438	3304	125
9	M&M	XUV300	6282	4203	49
10	M&M	XUV700	5815	3407	71

The given data has 2 missing values The missing value for oct21 for maruti Vitara can be calculated by taking average

(1) Missing value = 6887

formula for YoY = $\frac{\text{oct22} - \text{oct21}}{\text{oct21}} \times 100$

∴ using this value we can calculate

(2) YoY of missing value = 16.915

(3) The sum of sales of oct22 = 90151

The sum of sales of oct21 = 68869

oct22 has higher sales. Considering oct22 had diwali in the month there can be a

Page No.

possible Conclusion that there are increased Sales during diwali

(9) Concept Hierarchy

Cars

SUV ~~XUV~~ XUV

Tata Hyundai Maruti Kia M&M

Creta Venue Brezza Vitara Sonet Seltos XUV 300 XUV 700 Scorpio

	Model	oct22	oct21	YoY
1	Tata	13767	10096	36.36
2	Hyundai	21395	17009	25.78
3	Maruti	17993	14919	20.604
4	Kia	17391	15931	9.16
5	M&M	19535	10914	78.99

M&M had the highest YoY in all models

	model	oct22	oct21	YoY
1	SUV	70546	57955	21.725
2	XUV	19535	10914	78.99

XUV, had better YoY than SUVs

Tweight & D-weight Considering oct22, oct21 class

No	model	Tweight22	Tweight21	Dweight22	Dweight21
1	Tata	57.69	42.308	15.282	14.65
2	Hyundai	55.71	44.289	23.75	24.69
3	Maruti	54.67	45.329	19.974	21.66
4	Kia	57.190	47.80	19.305	23.132
5	M&M	64.156	35.84	21.686	15.847

M&M had highest Sale within class 22
Kia had highest sale within class 21
Hyundai had highest Sale across class 21, 22

→ : Conclusions from entire data

(1) Heavy diwali in the month can lead to increased Sales.
(2) M&M models had the highest YoY
(3) XUVs had better YoY which lead to M&M having highest YoY
(4) M&M had highest Sale within class oct22
(5) Kia had highest Sale within class oct21
(6) Hyundai had highest Sale across classes oct21 oct22
(7) People are more likely to buy XUVs during diwali than SUVs

Experiment no: 09

Distance and cluster

FINDING CENTER OF 4 coordinates

```
import java.io.*;
import java.util.*;
```

```
class Centerpt
{

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(new File("center.csv"));
        ArrayList<Integer> x = new ArrayList<>();
        ArrayList<Integer> y = new ArrayList<>();

        int i = 0;
        while(sc.hasNext()){

            String s[]=(sc.nextLine()).split("\\,");
            x.add(Integer.parseInt(s[0]));
            y.add(Integer.parseInt(s[1]));


        }

        System.out.println(x);
        System.out.println(y);

        int sumx = 0;
        int sumy = 0;
        for( i = 0 ; i < x.size();i++){

            sumx = sumx+x.get(i);
            sumy = sumy+y.get(i);

        }
        int Cx = sumx/4;
        int Cy = sumy/4;

        System.out.println("Center pts ("+Cx+","+Cy+")");

        double distance[] = new double[4];

        for( i = 0 ; i < 4 ;i++)
        {
          // distance[i]=Math.sqrt((x.get(i)-Cx * x.get(i)-Cx) + (y.get(i)-Cy * y.get(i)-Cy));
         //  System.out.println(Math.sqrt((Cx-x.get(i)) * (Cx-x.get(i))) + (Cy-y.get(i)) * (Cy)-y.get(i));

         distance[i] = (Math.sqrt(( Cx- x.get(i)) * (Cx - x.get(i)) + (Cy - y.get(i)) * (Cy - y.get(i))));
```

```
    }

    for( i = 0 ; i < 4 ; i++){

        System.out.println(String.format("%.2f",distance[i]));

    }
    System.out.println("Lower Triangular matrix ");
    //LOWER TRIANGULAR MATRIX
    double a[][] = {

            {distance[0],distance[1]},
            {distance[2],distance[3]}

    };

    int rows = 2;
    int cols = 2;


    for( i = 0; i < rows; i++){
        for(int j = 0; j < cols; j++){
          if(j > i)
            System.out.print("0 ");
          else
            System.out.print(String.format("%.2f",a[i][j]) + " ");
      }
      System.out.println();      }
  }}
```

Experiment No:11

Gini Index:

Theory:

Lab : Gini Index.

2020BTEIT00203.

PAGE NO.:
DATE / /

Gini Index :-

(1) Gini Index, also known as Gini Impurity, calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.

(2) If all elements are linked with a single class then it can be called pure.

(3) Like the properties of entropy, the Gini Index varies between values 0 and 1, where 0 expresses purity of classification, i.e. All the elements belong to a specified class or only one class exists there.

(4) And 1 indicates the random distribution of elements across various classes.

(5) The value of 0.5 of Gini Index shows an equal distribution of elements over some classes.

(6)

The Gini Index is determined by deducting the sum of squared of probabilities of each class from one, mathematically, Gini Index can be expressed as:

$$\boxed{\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2}$$

where P_i denotes the probability of an elements being classified for distinct class.

Dataset:

GiniIndex.java U | ginni1.csv U | FiveNoSummary.java

home > shubham > Desktop > K8S > ginni1.csv

```csv
CustId,Gender,CarType,ShirtSize,Class
1,M,Family,Small,C0
2,M,Sports,Medium,C0
3,M,Sports,Medium,C0
4,M,Sports,Large,C0
5,M,Sports,Extra large,C0
6,M,Sports,Extra large,C0
7,F,Sports,Small,C0
8,F,Sports,Small,C0
9,F,Sports,Medium,C0
10,F,Luxury,Large,C1
11,M,Family,Large,C1
12,M,Family,Extra large,C1
13,M,Family,Medium,C1
14,M,Luxury,Extra large,C1
15,F,Luxury,Small,C1
16,F,Luxury,Small,C1
17,F,Luxury,Medium,C1
18,F,Luxury,Medium,C1
19,F,Luxury,Medium,C1
20,F,Luxury,Large,C1
```

Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.*;
class GiniIndex {
public static void main(String[] args) throws Exception {
BufferedReader br = new BufferedReader(new FileReader("ginni1.csv"));
ArrayList<ArrayList<String>> data = new ArrayList<>();
String[] dt = br.readLine().trim().split(",");
int n=dt.length;
while (true) {
String str[] = new String[n];
try {
str = br.readLine().trim().split(",");
// System.out.println(str[0] + " " + str[1]+ ""+ str[2]+ str[3]+str[4]);
} catch (NullPointerException e) {
break;
}
ArrayList<String> temp = new ArrayList<>();
for (String s : str)
temp.add(s);
data.add(temp);
//System.out.println(data);
}
int C0 = 0, C1 = 0;
for (int i = 0; i < data.size(); i++) {
if (data.get(i).get(4).equals("C0")) C0++;
else
C1++;
}
System.out.println("Total Possiblity of C0 : "+C0 + " Total Possiblity of C1 : "+C1);
double totRecord = data.size();
double GineIndex = 1.0-Math.pow((C0 / totRecord),2) - Math.pow( (C1 / totRecord),2) ;
System.out.println("Gine Index of C0 :"+GineIndex);
System.out.println("==========================================================="
);
for(int i=1;i<data.get(0).size()-1;i++){
System.out.println("GineIndex Of "+ dt[i]+" :"+
String.format("%.5f",findAllGineIndex(i,GineIndex,data,dt)));

System.out.println("==========================================================="
);
}
}
private static double findAllGineIndex(int i, double GineIndex, ArrayList<ArrayList<String>>
```

```
data, String[] dt) {
Set<String> attribute=new HashSet<>();
for(ArrayList<String>x:data)
attribute.add(x.get(i));
Map<String,double[]>total=new HashMap<>();
for(String x:attribute){
total.put(x,new double[2]);

}
for(ArrayList<String> x:data){
//System.out.println("+++"+x);
if(x.get(4).equals(("C0")))
total.get(x.get(i))[0]++;
else
total.get(x.get(i))[1]++;
}
double totalE=0.0;
System.out.println("Count Table for :"+dt[i]);
System.out.println(dt[i]+" C0 C1 ");
for(Map.Entry<String,double[]> x:total.entrySet()){
double total1=x.getValue()[0]+x.getValue()[1];
System.out.println(x.getKey()+" "+x.getValue()[0]+" "+x.getValue()[1]);
if(x.getValue()[0]==0||x.getValue()[1]==0)
continue;
double temp= 1-Math.pow((x.getValue()[0] / total1),2) - Math.pow((x.getValue()[1]/
total1),2);
//System.out.println(temp);
totalE+=(total1/ data.size())*temp;
}
// double finalE=GineIndex-totalE;
return (totalE);
}
}
```

<u>Output:</u>

```
shubham@shubham-Vostro-3405:~/Desktop/K8S$ javac GiniIndex.java
shubham@shubham-Vostro-3405:~/Desktop/K8S$ java GiniIndex
Total Possiblity of C0 : 9  Total Possiblity of C1 : 11
Gine Index of C0 :0.49499999999999994
==========================================================
Count Table for  :Gender
Gender C0   C1
F  3.0  7.0
M  6.0  4.0
GineIndex Of Gender :0.45000
==========================================================
Count Table for  :CarType
CarType C0   C1
Luxury  0.0  8.0
Family  1.0  3.0
Sports  8.0  0.0
GineIndex Of CarType :0.07500
==========================================================
Count Table for  :ShirtSize
ShirtSize C0   C1
Small  3.0  2.0
Medium  3.0  4.0
Large  1.0  3.0
Extra large  2.0  2.0
GineIndex Of ShirtSize :0.46643
==========================================================
shubham@shubham-Vostro-3405:~/Desktop/K8S$
```

Conclusion:

1. We came to the conclusion that lesser gini index helps to identify that particular data is helpful more than any other column ie it gives more information hence can be used in decision making
2. Which data is impure and pure can be identified.