Data Mining Lab Book

Name:- Sumit Sunil Koundanya
PRN:- 2019BTEIT00023
Class: Final Year-IT-Sem I (2022-2023)

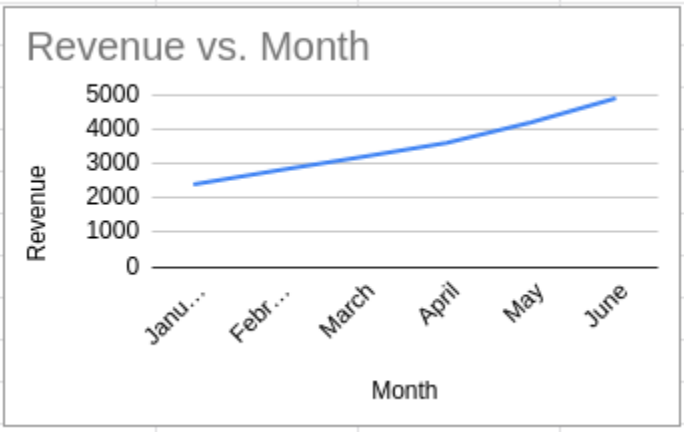
Experiment No. 1

Title :- Study and use of different types of graphs and charts (use MS-XLS).

Aim :- To study and use different types of graphs and charts (use MS-XLS).

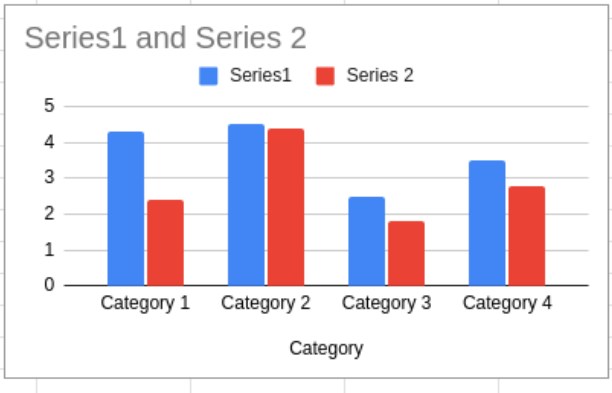
Line chart- In a line chart, category data is distributed evenly along the horizontal axis, and all value data is distributed evenly along the vertical axis. Line charts can show continuous data over time on an evenly scaled axis, so they're ideal for showing trends in data at equal intervals, like months, quarters, or fiscal years.

Month	Revenue
January	2400
February	2800
March	3200
April	3600
May	4200
June	4900



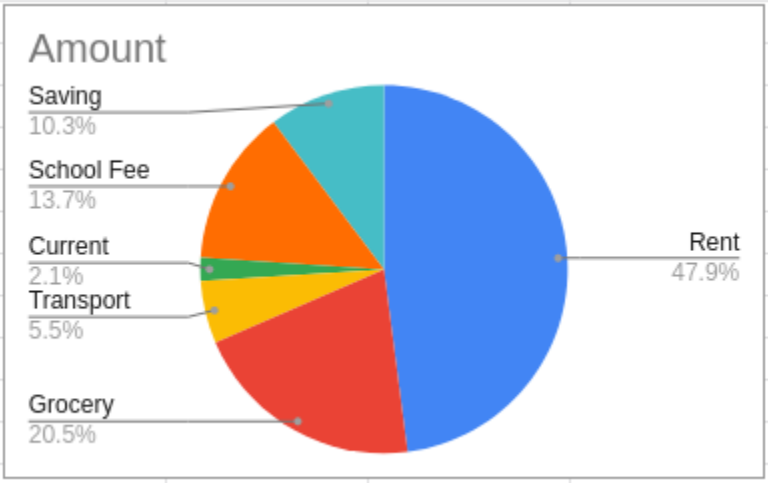
Column chart- A column chart is a data visualization where each category is represented by a rectangle, with the height of the rectangle being proportional to the values being plotted. Column charts are also known as vertical bar charts.

Category	Series1	Series 2
Category 1	4.3	2.4
Category 2	4.5	4.4
Category 3	2.5	1.8
Category 4	3.5	2.8



Pie chart :- A pie chart is a pictorial representation of data in the form of a circular chart or pie where the slices of the pie show the size of the data. A list of numerical variables along with categorical variables is needed to represent data in the form of a pie chart.A pie chart is a type of a chart that visually displays data in a circular graph. It is one of the most commonly used graphs to represent data using the attributes of circles, spheres, and angular data to represent real-world information.

Expenses	Amount
Rent	7000
Grocery	3000
Transport	800
Current	300
School Fee	2000
Saving	1500

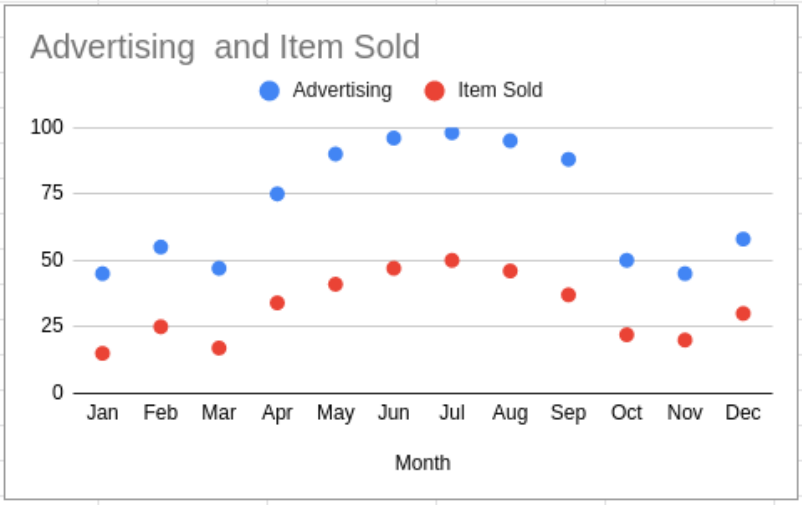


Bar chart:- A bar chart is a statistical approach to represent given data using vertical and horizontal rectangular bars. The length of each bar is proportional to the value they represent. It is basically a graphical representation of data with the help of horizontal or vertical bars with different heights. In real life, bar graphs are mainly used in the corporate sector.

Store Inventory		
Product	Quantity	Price
Tablet	320	800
Printer	450	250
Laptop	300	1200
Monitor	120	150
Computer	280	300

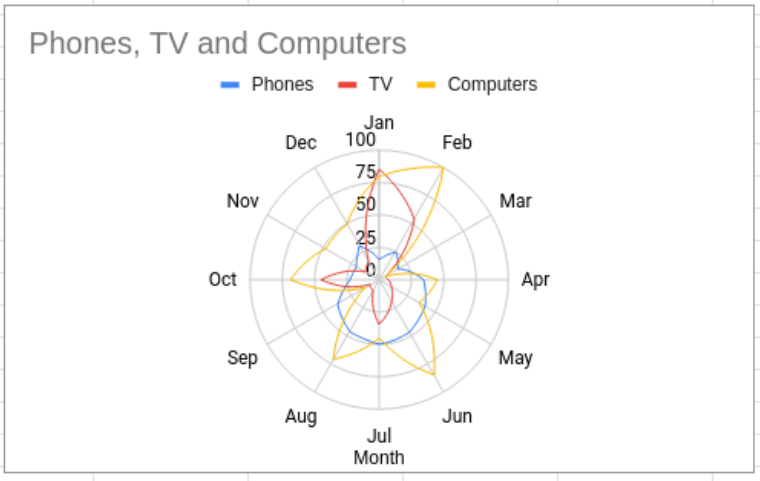
Scatter chart:- Scatter charts are based on basic line charts with the x axis changed to a linear axis. To use a scatter chart, data must be passed as objects containing X and Y properties. Scatter plots/charts are used to plot data points on a horizontal and a vertical axis in the attempt to show how much one variable is affected by another. A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables.

Month	Advertising	Item Sold
Jan	$45	15
Feb	$55	25
Mar	$47	17
Apr	$75	34
May	$90	41
Jun	$96	47
Jul	$98	50
Aug	$95	46
Sep	$88	37
Oct	$50	22
Nov	$45	20
Dec	$58	30



Radar chart:- A radar chart is a way of showing multiple data points and the variation between them. They are often useful for comparing the points of two or more different data sets. Radar Charts are used to compare two or more items or groups on various features or characteristics.

Month	Items Sold		
	Phones	TV	Computers
Jan	15	85	80
Feb	25	54	100
Mar	17	5	5
Apr	34	7	45
May	41	11	36
Jun	47	20	85
Jul	50	35	45
Aug	46	10	71
Sep	37	8	12
Oct	22	45	69
Nov	20	12	48
Dec	30	16	50

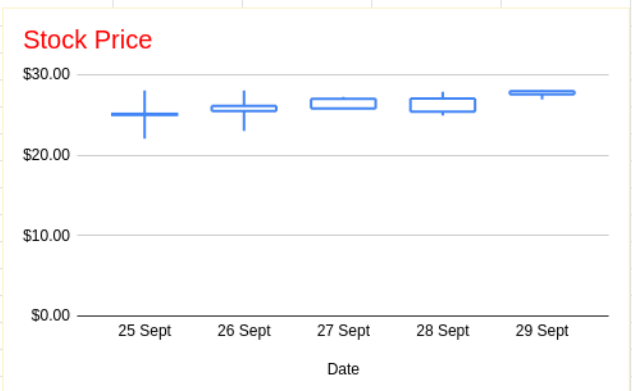


Histogram chart :- A histogram is a chart that groups numeric data into bins, displaying the bins as segmented columns. They're used to depict the distribution of a dataset: how often values fall into ranges.

Custmer Waiting time in seconds			
10.4	12	18.8	15.9
10.9	12.4	17.3	10.7
12.5	14.3	11.4	11.9
17.3	11.3	10.2	20.3

Candlestick chart:- Candlestick charts display an asset price's Open, High, Low, and Close prices over a period of time. They are sometimes referred to as the Japanese Candlestick chart. Its name comes from its appearance: The graph looks like candles with a wick sticking out from both sides of the wax.

Stock Prices (in Million USD)				
Date	Open Value	High Value	Low Value	Closing Value
25 Sept	$24.98	$28.00	$22.05	$25.12
26 Sept	$26.12	$28.00	$23.01	$25.45
27 Sept	$26.98	$27.22	$25.75	$25.80
28 Sept	$27.00	$27.85	$24.92	$25.38
29 Sept	$27.93	$28.00	$26.90	$27.55



Experiment No 1.

Title - Study and use of different types of graphs and charts

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

Theory - There are various types of charts used in data Mining.

A) Line chart

In a line chart, category data is distributed evenly along the horizontal axis, and all value data is distributed evenly along vertical axis. Line chart can show continuous data over time on an evenly scaled axis, so they are ideal for showing trends in data at equal intervals.

B) column chart

A column chart is a data visualization where each category is represented by a rectangle, with the height of the rectangle being proportional to the values being plotted. column charts are known as Vertical bar Charts

c) Pie chart

A pie chart is a pictorial representation of data in the form of circular chart or pie where the slices of the pie shows the size of the data. A pie chart is a type of chart that visually displays data in a circular graph. It is one of the most commonly used graphs to represent data using attributes of circle, spheres, and angular data to represent real-world information.

d) Scatter Chart

Scatter charts are based on basic line charts with x-axis changed to a linear axis. To use a scatter chart, data must be passed as objects containing X and Y properties. A scatter plot uses dots to represent values for two different numeric variables.

E) Candlestick chart

They displays an asset price's open, High, Low and close prices over a period of time. They are sometimes referred to as Japanese candlestick chart. Its names comes from its appearance - graph looks like candles with a wick sticking out from both sides of the wax.

Conclusion :- Different types of MS-XLS charts/graph can be used to visualize different data.

Experiment No. 2

Title:- To perform Normalization of data (Min-max and Z-score).

Data Normalization:- The data normalization (also referred to as data pre-processing) is a basic element of data mining. It means transforming the data, namely converting the source data into another format that allows processing data effectively. The main purpose of data normalization is to minimize or even exclude duplicated data. This is a very essential and important issue because it is increasingly problematic to keep data in relational databases, which store identical data in more than one place.

The use of data mining normalization has a number of advantages:
- the application of data mining algorithms becomes easier
- the data mining algorithms get more effective and efficient
- the data is converted in to the format that everyone can get their heads around
- the data can be extracted from databases faster
- it is possible to analyze the data in a specific manner

Min-Max Normalization-
The first technique we will cover is min-max normalization. It is the linear transformation of the original unstructured data. It scales the data from 0 to 1. It is calculated by the following formula:

$$v' = \frac{v - \text{minF}}{\text{maxF} - \text{minF}} (new\_max_F - new\_min_F) + new\_min_F ,$$

Where v - is the respective value of the attribute.

Z-Score Normalization-
The next technique is z-score normalization. It is also called zero-mean normalization. The essence of this technique is the data transformation by the values conversation to a common scale where an average number equals zero and a standard deviation is one. A value is normalized to under the formula:

$$v' = \frac{v - \bar{F}}{\sigma_F},$$

Where v - actual data
F(bar) - mean value of data
σ (sigma) - standard deviation of data

Experiment No 2

Aim - To perform Normalization of data (Min-Max and z-score)

Theory - Data Normalization is a basic element of data Mining. It means transforming the data, namely converting the source data into another format that allows processing data effectively. The main purpose of data Normalization is to minimize or even exclude duplicated data. This is a very essential and important issue because it is increasingly problematic to keep data in relational databases, which store identical data in more than one place.

The use of data Mining Normalization has a no. of advantages -

- Data mining algorithms get more effective and efficient
- Data is converted into the format that everyone can get their heads around
- Data can be extracted from databases faster
- It is possible to analyze the data in a specific manner.

Formula :-

A) Min-Max Normalization

$$v' = \frac{v - minF}{maxF - minF} (new\_maxF - new\_minF) + new\_minF$$

where v' - New value

v - Respected value of the attribute

$maxF$ - maximum value from given dataset

$minF$ - minimum value from given dataset

B) Z-Score Normalization

$$v' = \frac{v - \bar{F}}{\sigma_F}$$

where v - actual data value

$\bar{F}$ - mean value of data

σ_F - standard deviation of data

Algorithm -

1) Take data set and Read data from csv file
2) If we choose Min-max Normalization then find max and min value from given dataset and if it is z-score Normalization then calculate mean and standard deviation of data
3) For Min-max normalization consider new_max and new_min by considering suitable range
4) Calculate Normalized value for all data in the given dataset with using respective normalized formula

Example - a) Min-Max Normalization.

Data [Marks]

20
75
34
55
63

Here, min = 20
Max = 75
New_min = 0
New_max = 1.

- For Marks 20 :

$$v'_{20} = \frac{20 - 20}{75 - 20} \times (1 - 0) + 0$$

$$= 0$$

- For Marks 75 :

$$v'_{75} = \frac{75 - 20}{75 - 20} \times (1 - 0) + 0$$

$$= 1$$

- For Marks 34 :

$$v'_{34} = \frac{34 - 20}{75 - 20} \times (1 - 0) + 0$$

$$= 0.2545$$

- For Marks 55 :

$$v'_{55} = \frac{55 - 20}{75 - 20} \times (1 - 0) + 0$$

$$= 0.6363$$

- For Marks 63 :

$$v'_{63} = \frac{63 - 20}{75 - 20} \times (1 - 0) + 0$$

$$= 0.7818$$

Normalized Table -

Marks	Normalized Marks
20	0
75	1
34	0.25
55	0.64
63	0.78

B) Z - Score Normalization.

Marks

9

12

15

20

$$\text{Mean} = \frac{9 + 12 + 15 + 20}{4}$$

$$\mu = 14$$

Standard deviation,

$$\sigma = \sqrt{\frac{(9-14)^2 + (12-14)^2 + (15-14)^2 + (20-14)^2}{4}}$$

$$= \sqrt{\frac{25 + 4 + 1 + 36}{4}}$$

$$= \sqrt{16.5}$$

$$\sigma = 4.062$$

- For Marks 9 :

$$Z_9 = \frac{9 - 14}{4.062} = -1.231$$

- For Marks 12 :

$$Z_{12} = \frac{12 - 14}{4.062} = -0.4923$$

- z-score

I/P Marks	z-score Normalized
9	-1.066
12	-0.426
15	0.213
20	1.279

- conclusion -

Normalization is an important process or task in data preprocessing. It is used to ensure consistency in data records. In order to bring all attribute on the same scale min-max Normalization is used. Standardizing score on same scale by dividing a scores deviation by standard deviation for that z-score is used. It scales a data to particular small scale which helps to allow processing data effectively.

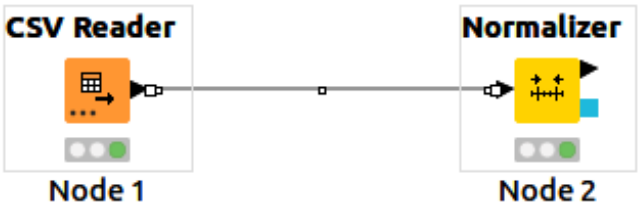
Program:-

```
#include <bits/stdc++.h>
#include <fstream>
using namespace std;
int main()
{
    // Declaring all variables
    double tmp, mini, maxi, new_mini, new_maxi;
    double sum, cnt, square_sum, mean, standard_deviation;
    // Opening file in reading mode
    ifstream in1("exp2_input_MinMax.csv");
    ifstream in2("exp2_input_MinMax.csv");
    ifstream in3("exp2_input_Zscore.csv");
    ifstream in4("exp2_input_Zscore.csv");

    int opt;
    cout << "\nEnter option: \n1.Min-Max Normalization \n2.Z-Score
Normalization\nOption: ";
    cin >> opt;

    ofstream out1("exp2_output_MinMax.csv");
    ofstream out2("exp2_output_Zscore.csv");

    switch (opt)
    {
    case 1: // Finding Min and Max

        if (!in1)
        {
            cout << "Error opening file, try again.";
            exit(0);
        }
        out1 << "Data "
             << ","
             << "Normalized Data"
             << "\n";

        in1 >> tmp;
```

```
        mini = tmp;
        maxi = tmp;
        while (in1)
        {
            if (tmp > maxi)
                maxi = tmp;
            if (tmp < mini)
                mini = tmp;
            in1 >> tmp;
        }
        // Min max Normalization
        cout << "Enter new min: ";
        cin >> new_mini;
        cout << "\nEnter new max: ";
        cin >> new_maxi;

        in2 >> tmp;
        while (in2)
        {
            double tmp2 = (((tmp -
                             mini) /
                            (maxi - mini)) *
                           (new_maxi - new_mini)) +
                          new_mini;
            out1 << tmp << "," << tmp2 << "\n";
            in2 >> tmp;
        }
        out1.close();
        break;

    case 2:
        if (!in3)
        {
            cout << "Error opening file, try again.";
            exit(0);
        }

        out2 << "Data "
             << ","
```

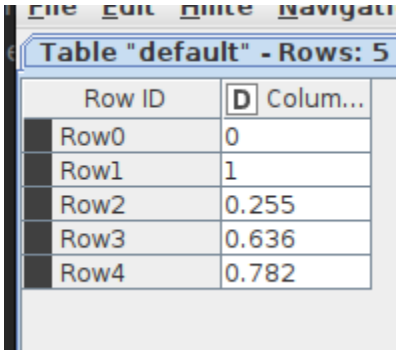
```
             << "Normalized Data"
             << "\n";

        in4 >> tmp;
        while (in4)
        {
            sum += tmp;
            cnt++;
            in4 >> tmp;
        }

        mean = sum / cnt; // calculate mean

        in4.clear();
        in4.seekg(0, ios::beg); // to set pointer again start of file

        in4 >> tmp;
        while (in4)
        {
            square_sum += (tmp - mean) * (tmp - mean);
            in4 >> tmp;
        }

        standard_deviation = sqrt(square_sum / cnt); // to find
standard deviation

        in4.clear();
        in4.seekg(0, ios::beg);
        in4 >> tmp;
        while (in4)
        {
            double tmp2 = ((tmp - mean) / standard_deviation);
            out2 << tmp << "," << tmp2 << endl;
            in4 >> tmp;
        }
        out2.close();
        break;

    default:
```

```
            cout << "Wrong Option"<<endl;
            out1.close();
            out2.close();
            break;
        }

        return 0;
}
```



Output:-

Min - Max Normalization



Z-Score Normalization

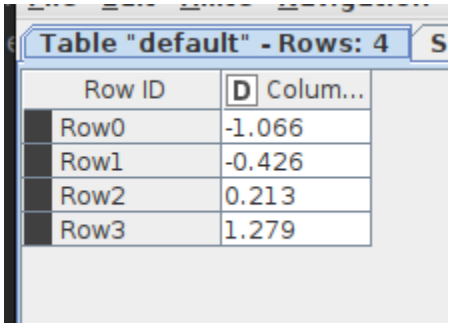
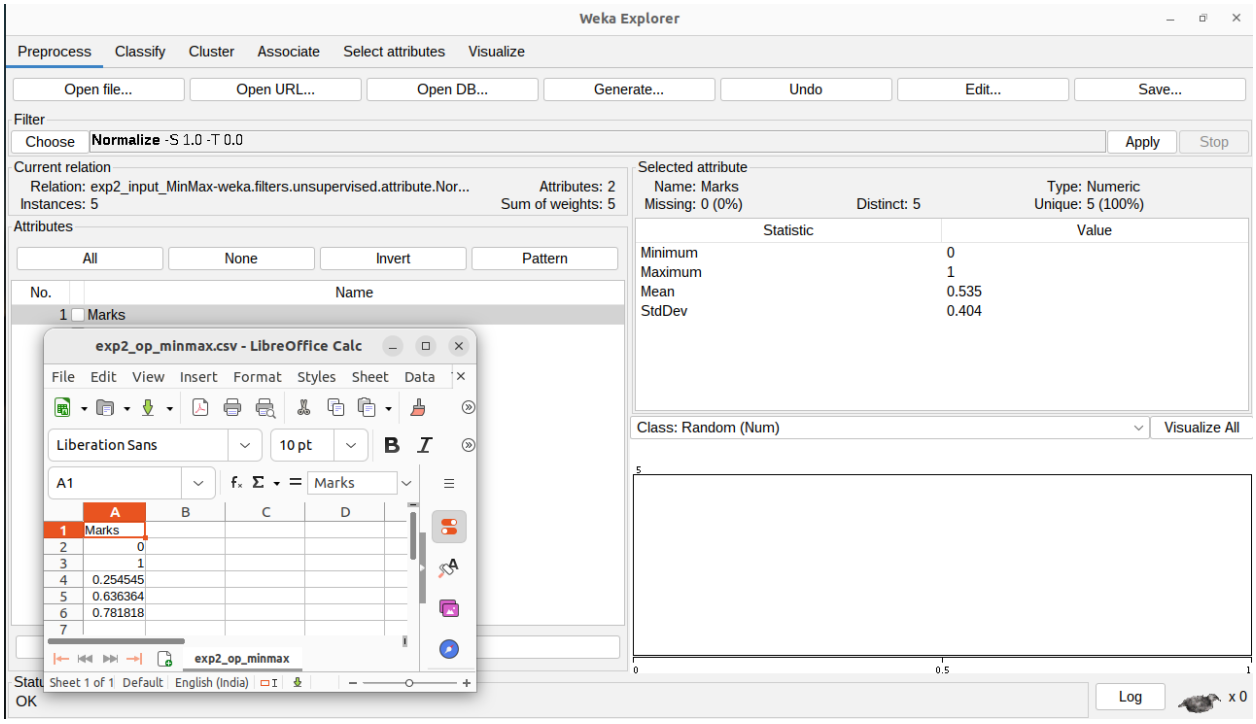


Table "default" - Rows: 4

Row ID	D Colum...
Row0	-1.066
Row1	-0.426
Row2	0.213
Row3	1.279

Weka Output -



```
Enter option:
1.Min-Max Normalization
2.Z-Score Normalization
Option: 1
Enter new min: 0

Enter new max: 1
```

```
 Enter option:
 1.Min-Max Normalization
 2.Z-Score Normalization
 Option: 2
sumit@sumit-15:~/Documents/DM Lab$ g++ minmaxnormalization.cpp
sumit@sumit-15:~/Documents/DM Lab$ ./a.out

 Enter option:
 1.Min-Max Normalization
 2.Z-Score Normalization
 Option: 3
 Wrong Option
```

Output file:- [Output file for Normalized Data]

Experiment No. 3

Title:- Find Info Gain of an attribute from given data

Theory:-

- **Entropy -** Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations. It determines how the decision tree chooses to split data.It actually affects how a Decision Tree draws its boundaries. Consider a dataset of N classes ,then entropy(E) can be calculated as -

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

where pi - probability of randomly selecting an example in class I

- **Information Gain -** Information Gain Measures the expected reduction in entropy caused by partitioning the example according to attribute. Information gain (IG) measures how much "information" a feature gives us about the class. It helps to determine the order of attributes in the nodes of a decision tree. It can help us to determine the quality of splitting. It also helps to determine how good the splitting of nodes in a decision tree is.

Information gain = entropy (parent) – [weightes average] * entropy (children)

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where, $Values(A)$ is the all possible values for attribute A, and S_v is the subset of S for which attribute A has value v.

Experiment No 3

Aim:- To Find Info Gain of an attribute from given data

Theory -

- Entropy - Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations. It determines how the decision tree choose to split data. It actually affects how a Decision Tree draws its boundaries.
- Information gain - It measures the expected reduction in the entropy caused by partitioning the example according to attribute. Information gain measures how much "information" a feature gives us about the class. It helps to determine the order of attributes in the nodes of a decision tree. It can help us to determine the quality of splitting. It also helps to determine how good the splitting of nodes in a decision tree is.

Formula -

A) Entropy

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

Where N - Total dataset of 'N' classes and

P_i - probability of randomly selecting an example

B) Information gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where values(A) is the all possible values for attribute A and Sv is subset of S for which A has value v

Algorithm -

1) Read data from .csv file
2) Defined a function to calculate entropy
   i) calculate frequency of distinct value in particular column.
   ii) calculate probability of distinct value in particular column.
   iii) calculate entropy of class and Attribute
3) Define function to calculate gain
   i) split the child column over unique values
   ii) Find entropy of Attributes
   iii) Find Info gain by comparing with attribute
4) Show the expected Result.

Example -

Level	Routine	PlayGame	class
High	InDoor	No	false
High	outDoor	NO	false
High	InDoor	yes	true
Normal	InDoor	yes	true
High	InDoor	yes	true
Normal	outDoor	No	false
Normal	outDoor	yes	true
High	InDoor	NO	false
Normal	InDoor	yes	true
Normal	InDoor	yes	true
Normal	outDoor	yes	true
High	outDoor	yes	true
Normal	InDoor	yes	true
High	outDoor	No	false

- Find Entropy of class (parent)

True (+) - 9

False (-) - 5

$$E(S) = -\left[\frac{9}{14}\log_2\left(\frac{9}{14}\right) + \frac{5}{14}\log_2\left(\frac{5}{14}\right)\right]$$

$$= -[-0.9402]$$

Entropy(S) = 0.9402

- Consider Attribute 'Routine'

Values (Routine) = InDoor, outDoor

S = [9+, 5-]

S_{InDoor} ← [6+, 2-]

$S_{outDoor}$ ← [3+, 3-]

$$Gain(S, Routine) = Entropy(S) - \sum_{v \in \{Indoors, Outdoors\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - \left(\frac{8}{14}\right) Entropy(S_{Indoors}) - \left(\frac{6}{14}\right) Entropy(S_{Outdoors})$$

$$= 0.9402 - \left(\frac{8}{14}\right)\left[-\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)\right] - \left(\frac{6}{14}\right)\left[-\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)\right]$$

$$= 0.9402 - \left(\frac{8}{14}\right) \times 0.811 - \left(\frac{6}{14}\right) \times 1.00$$

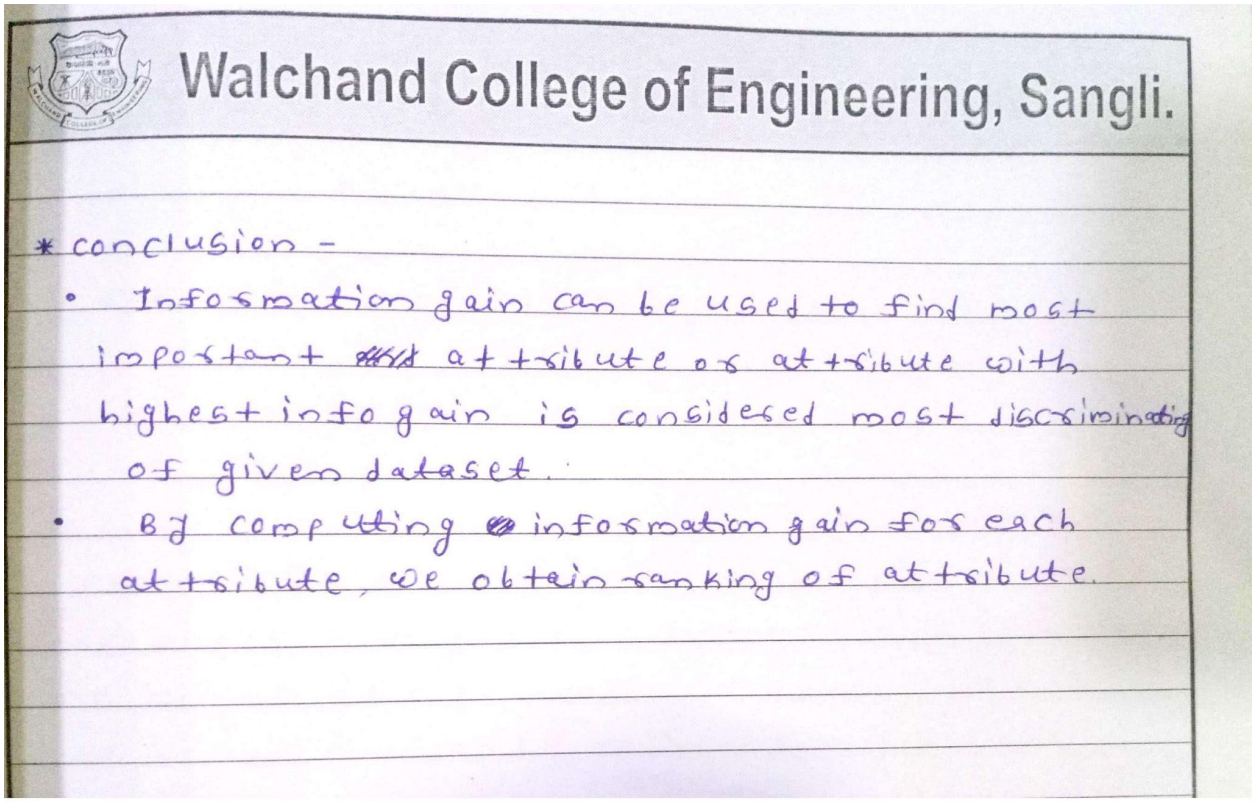
$$= 0.9402 - 0.4634 - 0.4286$$

$$= 0.0482$$

$\therefore$ Gain(S, Routine) = 0.0482

Conclusion -

- From the calculated value of info gain Routine has higher gain making it a better spliting attribute
- Info gain can be used to find best splitting criteria from a set of attribute



Input file:- [Input DataSet for Info Gain]

Program:-

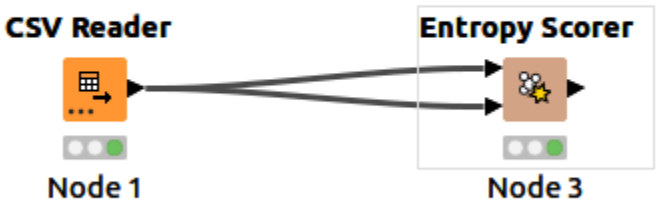
```
#include <bits/stdc++.h>
using namespace std;

int main()
{

    ifstream file("exp3_input.csv");

    string line, word;
    string day,level, Routine, playGame,value;

    map<string, int> parent;
    map<string, map<string, int>> child;
```

```
if (!file.is_open())
{
    perror("Error in opening input file : ");
    return -1;
}

int i = 0;
string childName;
while (getline(file, line))
{
    stringstream str(line);

    getline(str, day, ',');
    getline(str,level,',');
    getline(str, Routine, ',');
    getline(str, playGame, ',');
    getline(str, value, ',');


    int choice;

    if (i == 0)
    {
        i++;
        cout << "Enter Child Column Number : ";
        cin >> choice;
        continue;
    }

    switch (choice)
    {
        case 1:
            childName = day;
            break;

        case 2:
            childName = level;
            break;
```

```
            case 3:
                childName = Routine;
                break;

            case 4:
                childName = value;
                break;

            default:
                childName = Routine;
                break;
        }

        parent[playGame]++;
        child[childName][playGame]++;
    }


    double pos = parent["Yes"], neg = parent["No"];
    double total = pos + neg;
    // cout << pos << " " << neg << "\n";

    double parent_entropy = -((pos / total) * log2(pos / total) +
(neg / total) * log2(neg / total));
    cout << "Parent Entropy: " << parent_entropy << "\n";

    double child_entropy = 0;
    for (auto p : child)
    {
        string val = p.first;
        double pR = child[val]["Yes"], nR = child[val]["No"];

        // cout << val << " " << pR << " " << nR << "\n";
        double tR = pR + nR;

        child_entropy += -((pR + nR) / total) * ((pR / tR) * log2(pR
/ tR) + (nR / tR) * log2(nR / tR));
    }
```

```
    cout << "Child Entropy * Their proportion : " << child_entropy <<
"\n";

    cout << "Info gain : " << parent_entropy - child_entropy << "\n";

    return 0;
}
```

Output:-



File

Clustering statistics

Data Statistics

Statistics	Value
Number of clusters found:	2
Number of objects in clusters:	14
Number of reference clusters:	2
Total number of patterns:	14

Data Statistics

Score	Value
Entropy:	0.8922
Quality:	0.1078

Row ID	I Size	D Entropy	D Normalized Entropy	D Quality
InDoor	8	0.811	0.811	?
OutDoor	6	1	1	?
Overall	14	0.892	0.892	0.108

File

Clustering statistics

Data Statistics

Statistics	Value
Number of clusters found:	2
Number of objects in clusters:	14
Number of reference clusters:	2
Total number of patterns:	14

Data Statistics

Score	Value
Entropy:	0.7885
Quality:	0.2115

Row ID	I Size	D Entropy	D Norma...	D Quality
Normal	7	0.592	0.592	?
High	7	0.985	0.985	?
Overall	14	0.788	0.788	0.212

File

Clustering statistics

Data Statistics

Statistics	Value
Number of clusters found:	2
Number of objects in clusters:	14
Number of reference clusters:	2
Total number of patterns:	14

Data Statistics

Score	Value
Entropy:	0.0
Quality:	1

Row ID	I Size	D Entropy	D Norma...	D Quality
No	5	0	0	?
Yes	9	0	0	?
Overall	14	0	0	1

```
sumit@sumit-15:~/Documents/DM Lab$ g++ exp3.cpp
sumit@sumit-15:~/Documents/DM Lab$ ./a.out
Enter Child Column Number : 5
Parent Entropy: 0.940286
Child Entropy * Their proportion : 0.892159
Info gain : 0.048127
sumit@sumit-15:~/Documents/DM Lab$
```

Experiment No. 4

Title:- Find t and d weight of a data.

Theory:- To represent descriptive data mining results in the form of rules,two weighted measures t-weight and d-weight are introduced.

- ☐ **T-weight** of a generalized term Ta is the ratio of the tuples covered by Ta in the target class versus all the tuples in the initial target class.
- ☐ **D-weight** of a generalized term Da in the target class is the ratio of the number of tuples in the initial target class that are covered by Da versus the total number of tuples in both initial target and contrasting classes that are covered by Da.

Table shows the number of Biographical and Horror films produced by the movie industry in the year 2021 with t-weight and d-weight information associated. The t-weight for (Biographical,Bollywood) is 56% because the number of Biographical films produced by bollywood industry (150) represents only 56% of the Biographical films produced in both movie industry(270) ; whereas d-weight for (Biographical,Bollywood) is 78% because the number of biographical films produced by bollywood (150) represent 78% of the total (Biographical plus Horror) films produced in Bollywood film industry(which is 190).

Experiment No. 4

Title - Find t and d weight of a data

Aim - To find t and d weight of a data

Theory - To represent descriptive data mining results in the form of rules, two weighted measures t-weight and d-weight are introduced

- T-weight of a generalized term Ta is the ratio of tuples covered by Ta in the target class verses all tuples in the initial target class
- D-weight of a generalized term Da in the target class is the ratio of numble of tuples in the initial target class that are covered by Da versus the total numbers of tuples in both initial target and contrasting classes that are covered by Da.

Algorithm -

1) Find total count for each attribute
2) For each value, Find t-weight
   take row-wise count for value and find percentage
3) For each value, find d-weight, take column wise count and find percentage
4) Find t and d weight for total attribute

Example -

class	Type	count
Action Films	Bollywood	150
	Tollywood	120
Horror Films	Bollywood	40
	Tollywood	60

Films/ Type	Bollywood			Tollywood			Total		
	count	T	D	count	T	D	count	T	D
Action	150	56%	78%	120	44%	67%	270	100%	73%
Horror	40	40%	22%	60	60%	33%	100	100%	27%
Total	190	52%	100%	180	48%	100%	370	100%	100%

Conclusion -

- D-weight helps to find the weight of an attribute across the class
- T-weight helps to find the weight of an attribute within the class
- T-weight and D-weight, two weighted measures are used to present descriptive mining results

Program:-

```
#include <bits/stdc++.h>
#include <fstream>
using namespace std;

int main()
{
    fstream file("exp4_input.csv", ios::in);

    if (!file.is_open())
    {
        cout << "Couldn't Open file";
        return 0;
    }

    string line, word;
    string col, row, count;
    int val;

    map<string, map<string, int>> classrowcolMap;
    map<string, int> colMap;
    map<string, int> rowMap;

    int i = 0;

    while (getline(file, line))
    {
        stringstream str(line);

        if (i == 0)
        {
            i++;
            continue;
        }

        getline(str, row, ',');
        getline(str, col, ',');
        getline(str, count, ',');
```

```
        val = stoi(count);

        // cout << col << " " << row << " "<< stoi(count) << " " <<
"\n";

        classrowcolMap[row][col] = val;
        colMap[col] += val;
        rowMap[row] += val;
    }
    for(auto r:rowMap)
    {

        for(auto c:colMap)
        {
            cout<<r.first<<"-"<<c.first<<":";
            cout<<classrowcolMap[r.first][c.first]<<endl;
        }
    }
    for(auto r:rowMap)
    {
        cout<<r.first<<"->"<<r.second<<endl;
    }

    for(auto c:colMap)
    {
        cout<<c.first<<"->"<<c.second<<endl;
    }


    int colSum = 0, rowSum = 0;

    for (auto c : colMap)
    {
        colSum += c.second;
    }

    cout << "colSum : " << colSum << "\n";
```

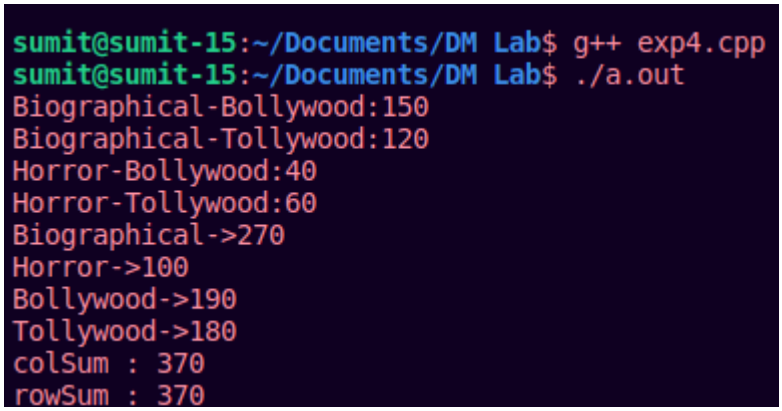
```
    for (auto r : rowMap)
    {
        rowSum += r.second;
    }
    cout << "rowSum : " << rowSum << "\n";

    ofstream fw("exp4_output.csv", ios::out);

    fw << "Column\\row , ,Bollywood ,,  ,Tollywood ,,  ,Total,,, " <<
endl;
    fw << "
,Count,t-weight,d-weight,Count,t-weight,d-weight,Count,t-weight,d-wei
ght" << endl;

    for (auto r : rowMap)
    {
        row = r.first;
        fw << row << ",";

        for (auto c : colMap)
        {
            col = c.first;

            fw << classrowcolMap[row][col] << ",";
            fw << ((float)classrowcolMap[row][col] / rowMap[row]) *
100 << "%,";
            fw << ((float)classrowcolMap[row][col] / colMap[col]) *
100 << "%,";
        }

        fw << rowMap[row] << "," << ((float)rowMap[row] /
rowMap[row]) * 100 << "%," << ((float)rowMap[row] / (colSum)) * 100
<< "%" <<endl;
    }

    fw << "Total ,";

    for (auto c : colMap)
    {
```

```
            col = c.first;

            fw << colMap[col] << ",";
            fw << ((float)colMap[col] / colSum) * 100 << "%,";
            fw << ((float)colMap[col] / colMap[col]) * 100 << "%,";
        }

        fw << colSum << "," << "100%, 100%" << endl;

        fw.close();

        return 0;
    }
```

Input file:- [Input DataSet]

Output file:- [Output with t-weight and d-weight associated with each class]

Output:-



Experiment No. 5

Title:- Find 5 no summary of a dataset

Theory:- The five-number summary is a set of descriptive statistics that provides information about a dataset. The 5 number summary is an exploratory data analysis tool that provides insight into the distribution of values for one variable. It consists of the five most important sample percentiles:

1. the sample minimum *(smallest observation)*
2. the lower quartile or *first quartile*
3. the median (the middle value)
4. the upper quartile or *third quartile*
5. the sample maximum (largest observation)

Experiment No 5

Title - Find 5 no summary of a dataset

Aim - To find 5 no summary of dataset and obtain boxplot.

Theory - The five number summary is a set of descriptive statistics that provides information about a dataset. The 5 number summary is an exploratory data analysis tool that provides insight into the distribution of values for one variable. It consists of the five most important sample percentiles:

1) The sample minimum (smallest observation)
2) The lower quartile or first quartile
3) The median
4) The upper quartile or third quartile
5) The sample maximum (largest observation)

Formula -

1) Median (X) $= \begin{cases} x\left[\frac{n+1}{2}\right] & \text{if } n \text{ is odd} \\ \frac{x\left[\frac{n}{2}\right] + x\left[\frac{n}{2}+1\right]}{2} & \text{if } n \text{ is even} \end{cases}$

where x - ordered list of values in the dataset

n - no. of values in dataset

2) Quartile for Q1 = $\frac{1}{4}(n+1)^{th}$ term

3) Quartile Q3 = $\frac{3}{4}(n+1)^{th}$ term

Algorithm -

1) Take any particular data set and put in ascending order.
2) Find minimum and maximum for your dataset
3) Find median.
4) If number of value is odd then leave that median value and consider data before and after median to find Q1 and Q3
5) If total number of values is even then consider the values that are used for finding median and before value in first dataset to find Q1 and after value in second dataset and apply same logic of median to find Q3
6) Find Q1 and Q3
7) At last plot the five numbers summary of dataset (Boxplot)

Example -

dataset - 2, 4, 5, 8, 10, 11, 1, 1, 2, 6, 6, 7

i) Arrange data in ascending order

1, 1, 2, 2, 4, 5, 6, 6, 7, 8, 10, 11

ii) Here n = 12 (even)

iii) Maximum = 11

Minimum = 1

iv) Median = Avg of 6th and 7th term

$= \frac{5 + 6}{2}$

$= 5.5$

v) For Q1

Take [1, 1, 2, 2, 4, 5]

$Q1 = \frac{2 + 2}{2} = 2$

vi) For Q3

Take - [6, 6, 7, 8, 10, 11]

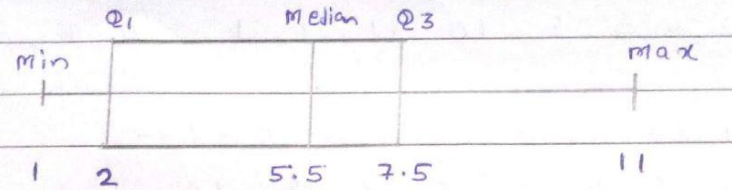
$Q3 = \frac{7 + 8}{2} = \frac{15}{2}$

$Q3 = 7.5$

- Summary -

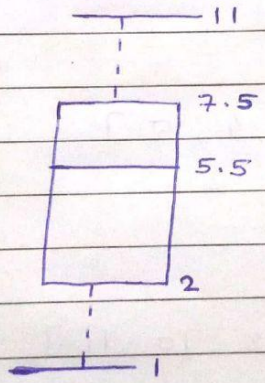
1) Minimum = 1
2) Maximum = 11
3) Median = 5.5
4) Q1 = 2
5) Q3 = 7.5

- Boxplot



- ~~Regression~~ Result -
  - Simple output has been obtained by calculating 5 Numbers summary.
  - 5 No. Summary can be used to obtain quick distribution of any given data.

Knime output



- Conclusion -

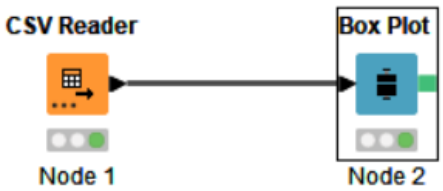
The five numbers summary is an exploratory data analysis tool that provides insights into distribution of values for one variable. It is useful in descriptive analyses or during the preliminary investigation of large data set. It gives a general sense of whether the distribution is symmetrical or skewed by comparing Q1, median and Q3

Program:-

```
#include <bits/stdc++.h>
using namespace std;

float median(vector<int> a)
{
    int size = a.size();
    if (size % 2 == 1)
        return a[size/2];
    else
        return (a[(size / 2) - 1] + a[size / 2]) / 2.0;
}

float quartile1(vector<int> v)
{
    int n = v.size();
    vector<int> first;

    for (int i = 0; i < n / 2; i++)
    {
        first.push_back(v[i]);
    }
    return median(first);
}

float quartile3(vector<int> v)
{
    int n = v.size();
    vector<int> last;
    if (n % 2 == 0)
    {
        for (int i = n / 2; i < n; i++)
        {
            last.push_back(v[i]);
        }
    }
```

```
    else
    {
        for (int i = n / 2 + 1; i < n; i++)
        {
            last.push_back(v[i]);
        }
    }

    return median(last);
}

int main()
{
    ifstream in("exp5_input.csv");
    if(!in.is_open())
    {
        cout<<"Couldn't open file";
        exit(0);
    }

    ofstream out("exp5_output.csv");

    int i=0;
    string line,mark;
    vector<int> arr;
    while(getline(in,line))
    {
        if(i==0)
        {
            i++;
            continue;
        }
        stringstream str(line);

        getline(str,mark,',');
        int x = stoi(mark);
        arr.push_back(x);

    }
```

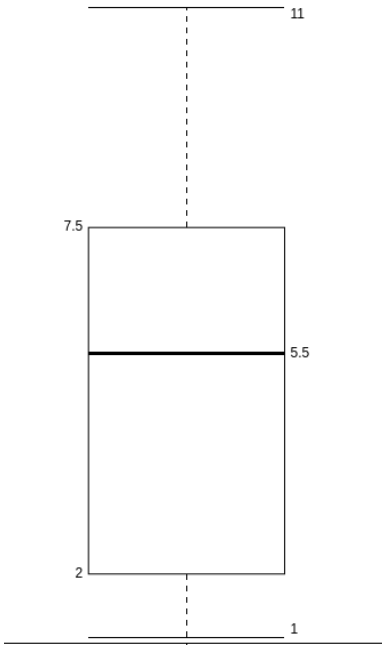
```
    int n = arr.size();
    sort(arr.begin(), arr.end());

    out << "Minimum value: "<<","<<arr[0]<<"\n";
    out << "Quartile1 value: "<<","<<quartile1(arr)<<"\n";
    out << "Median value: "<<","<<median(arr)<<"\n";
    out << "Quartile3 value: "<<","<<quartile3(arr)<<"\n";
    out << "Maximum value: "<<","<<arr[n-1]<<"\n";

    cout << "Minimum value is " << arr[0] << endl;
    cout << "Q1: " << quartile1(arr) << endl;
    cout << "Median: " << median(arr) << endl;
    cout << "Q3: " << quartile3(arr) << endl;
    cout << "Maximum value is " << arr[n - 1] << endl;

    return 0;
}
```



Input file:- Input Dataset

Output file:- Output of 5 Number Summary

Output:-




```
sumit@sumit-15:~/Documents/DM Lab$ g++ exp5.cpp
sumit@sumit-15:~/Documents/DM Lab$ ./a.out
Minimum value is 0
Q1: 3
Median: 8.5
Q3: 30
Maximum value is 100
sumit@sumit-15:~/Documents/DM Lab$
```

Experiment No. 6

Title:- Find frequent itemset from given transaction data.

Theory:- When items are grouped together they form an itemset. An itemset that occurs frequently is called a frequent itemset. Frequent itemset mining is a data mining technique to identify the items that often occur together. A set of items is called frequent if it satisfies a minimum threshold value for support and confidence. Support shows transactions with items purchased together in a single transaction.confidence shows transactions where the items are purchased one after the other.

- Frequent items are determined by Apriori Algorithm.

Key concept:-

- ☐ **Support:-** It refers to the popularity of a product in a transaction - A measure of interestingness. This tells about the usefulness and certainty of rules.

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

- ☐ **Confidence:-** Confidence shows the possibilities that the customer bought items one after another in a single transaction.

$$\text{Confidence } (A \longrightarrow B) = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

- ☐ **Support_count(X):-** Number of transactions in which X appears. If X is A union B then it is the number of transactions in which A and B both are present.

Experiment No 6

Title - Find frequent itemset from given transaction data.

Aim - To find frequent itemset from given transaction data

Theory - When items are grouped together they form an itemset. An itemset that occurs frequently is called frequent itemset Frequent itemset mining is a data mining technique to identify the items that occur together. A set of items is called frequent if it satisfies a minimum threshold value for support and confidence. support shows transactions with items purchased together in a single transaction. confidence shows transactions where the items are purchased one after the other.

· Frequent items are determined by Apriori Algorithm

Formula -

A) Support - Refers to the popularity of a product in a transaction. A measure of interestingness.

$$\text{Support}(A) = \frac{\text{No. of transaction in which A appears}}{\text{Total no. of transactions}}$$

B) Confidence - It shows the possibilities that the customer bought items one after another in a single transaction.

$$\text{confidence}(A \rightarrow B) = \frac{\text{support}(A \cup B)}{\text{support}(A)}$$

Algorithm -

1) Take min_support as input from user.
2) Read excel / CSV file
3) Calculate min_frequency for dataset.
4) For each level calculate frequency of itemset and support_count
5) If support_count of itemset is greater than minimum support, declare it as frequent itemset and print it.

Example -

Transaction	Itemsets
T1	{A, B, C}
T2	{A, C}
T3	{A, D}
T4	{B, E, F}

Minimum support = 50 %

Minimum confidence = 50 %

Minimum support count $= \frac{50}{100} \times 4 = 2$

step 1 : K = 1

C1 (candidate set)

Itemset	sup-count
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

Compare candidate set item's support count with minimum support count

L1 =

Itemset	sup-count
{A}	3
{B}	2
{C}	2

step 2 : K = 2

Generate candidate set C2 using L1

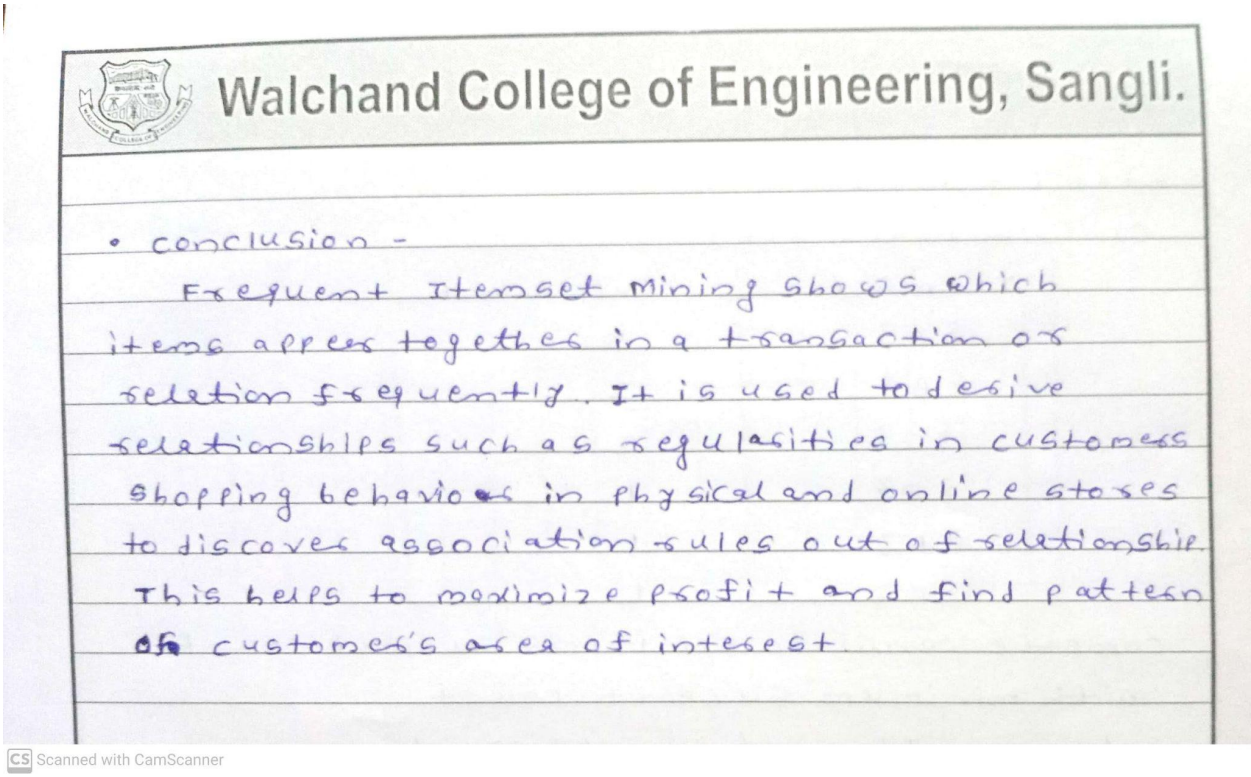
Itemset	sup-count
{A, B}	1
{A, C}	2
{B, C}	1

Compare candidate set C2 support count with minimum support. this gives us itemset L2

Itemset	sup-count
{A, C}	2

We stop here as no frequent itemset are found further ∴ Frequent Itemset = {A, C}

Page No.



Program:-

```
#include <bits/stdc++.h>
#include <map>
using namespace std;

ifstream fin;
double minfre;
vector<set<string>> datatable;
set<string> products;
map<string, int> freq;
vector<string> wordsof(string str)
{
    vector<string> tmpset;
    string tmp = "";
    int i = 0;
    while (str[i])
```

```
    {
        if (isalnum(str[i]))
            tmp += str[i];
        else
        {
            if (tmp.size() > 0)
                tmpset.push_back(tmp);
            tmp = "";
        }
        i++;
    }

    if (tmp.size() > 0)
        tmpset.push_back(tmp);

    return tmpset;
}

string combine(vector<string> &arr, int miss)
{
    string str;
    for (int i = 0; i < arr.size(); i++)
        if (i != miss)
            str += arr[i] + " ";
    str = str.substr(0, str.size() - 1);
    return str;
}

set<string> cloneit(set<string> &arr)
{
    set<string> dup;
    for (set<string>::iterator it = arr.begin(); it != arr.end();
it++)
        dup.insert(*it);
    return dup;
}

set<string> apriori_gen(set<string> &sets, int k)
{
```

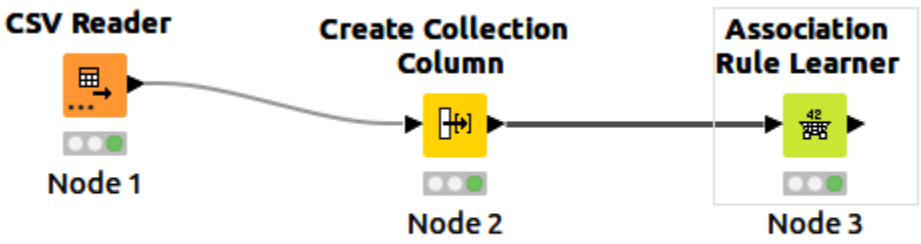
```
    set<string> set2;
    for (set<string>::iterator it1 = sets.begin(); it1 != sets.end();
it1++)
    {
        set<string>::iterator it2 = it1;
        it2++;
        for (; it2 != sets.end(); it2++)
        {
            vector<string> v1 = wordsof(*it1);
            vector<string> v2 = wordsof(*it2);

            // cout << "\nVector 1 :";
            // for(auto s : v1){
            //     cout << s << " ";
            // }
            // cout << "\n";

            // cout << "\nVector 2 :";
            // for(auto s : v2){
            //     cout << s << " ";
            // }
            // cout << "\n";

            bool alleq = true;
            for (int i = 0; i < k - 1 && alleq; i++)
                if (v1[i] != v2[i])
                    alleq = false;

            v1.push_back(v2[k - 1]);
            if (v1[v1.size() - 1] < v1[v1.size() - 2])
                swap(v1[v1.size() - 1], v1[v1.size() - 2]);

            for (int i = 0; i < v1.size() && alleq; i++)
            {
                string tmp = combine(v1, i);
                if (sets.find(tmp) == sets.end())
                    alleq = false;
            }
```

```
            if (alleq)
                set2.insert(combine(v1, -1));
        }
    }
    return set2;
}

int main()
{
    fin.open("freqitem.csv", ios::in);

    if (!fin.is_open())
    {
        perror("Error in opening file : ");
    }
    cout << "Frequency % :";
    cin >> minfre;

    string str;
    while (!fin.eof())
    {
        getline(fin, str);
        vector<string> arr = wordsof(str);
        set<string> tmpset;
        for (int i = 0; i < arr.size(); i++)
            tmpset.insert(arr[i]);
        datatable.push_back(tmpset);

        for (set<string>::iterator it = tmpset.begin(); it !=
tmpset.end(); it++)
        {
            products.insert(*it);
            freq[*it]++;
        }
    }
    fin.close();

    cout << "No of transactions: " << datatable.size() << endl;
    minfre = minfre * datatable.size() / 100;
```

```
    cout << "Min frequency:" << minfre << endl;

    queue<set<string>::iterator> q;
    for (set<string>::iterator it = products.begin(); it !=
products.end(); it++)
        if (freq[*it] < minfre)
            q.push(it);

    while (q.size() > 0)
    {
        products.erase(*q.front());
        q.pop();
    }

    int pass = 1;
    cout << "\nFrequent " << pass++ << " -item set : \n";
    for (set<string>::iterator it = products.begin(); it !=
products.end(); it++)
        cout << "{" << *it << "} " << freq[*it] << endl;

    int i = 2;
    set<string> prev = cloneit(products);

    while (i)
    {
        set<string> cur = apriori_gen(prev, i - 1);

        if (cur.size() < 1)
        {
            break;
        }

        for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
        {
            vector<string> arr = wordsof(*it);

            int tot = 0;
            for (int j = 0; j < datatable.size(); j++)
```

```
            {
                bool pres = true;
                for (int k = 0; k < arr.size() && pres; k++)
                    if (datatable[j].find(arr[k]) ==
datatable[j].end())
                        pres = false;
                if (pres)
                    tot++;
            }
            if (tot >= minfre)
                freq[*it] += tot;
            else
                q.push(it);
        }

        while (q.size() > 0)
        {
            cur.erase(*q.front());
            q.pop();
        }

        // cout << "Flag : " << flag << "\n";
        bool flag = true;

        for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
        {
            vector<string> arr = wordsof(*it);

            if (freq[*it] < minfre)
                flag = false;
        }

        if (cur.size() == 0)
            break;

        cout << "\n\nFrequent " << pass++ << " -item set : \n";
        for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
```

```
            cout << "{" << *it << "} " << freq[*it] << endl;

        prev = cloneit(cur);
        i++;
    }
    ofstream fw("ferqitem_op.csv", ios::out);

    for (auto it = prev.begin(); it != prev.end(); it++)
    {
        fw << "{" << *it << "}" << endl;
    }

    return 1;
}
```



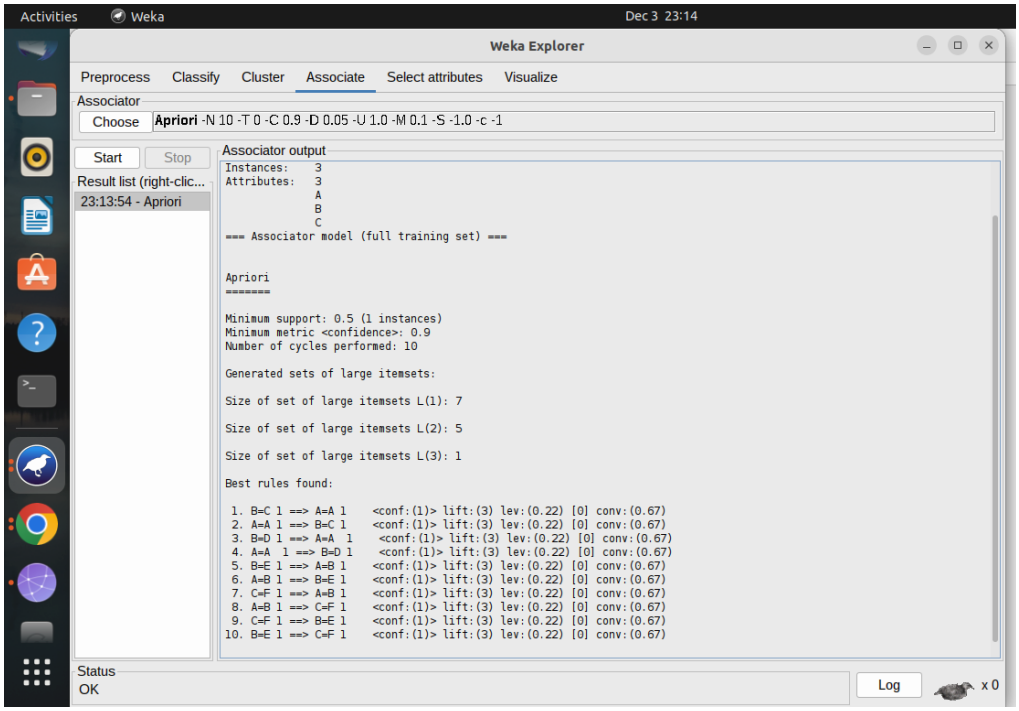
Input file:- [Dataset for Frequent Itemset]

Output file :- [Output of Frequent Itemsets]

Output:-

Table "default" - Rows: 4 | Spec - Co

Row ID	D Suppo...	[...] Items
item set 0	0.5	[B]
item set 1	0.5	[A,C]
item set 2	0.5	[A,?]
item set 3	0.75	[A]

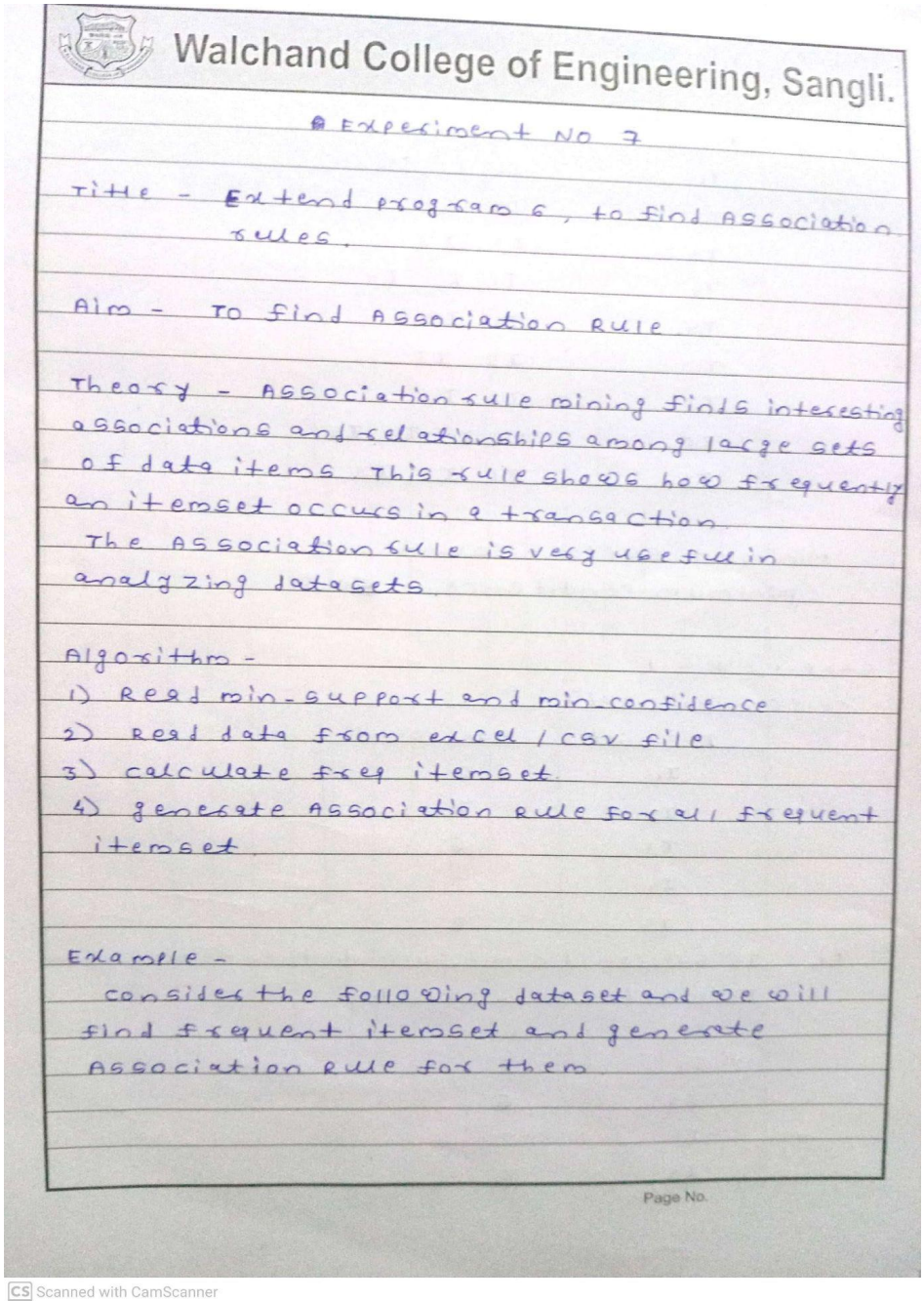



```
sumit@sumit-15:~/Documents/7th Sem/DM Lab$ g++ exp6.cpp
sumit@sumit-15:~/Documents/7th Sem/DM Lab$ ./a.out
Frequency % :50
No of transactions: 4
Min frequency:2

Frequent 1 -item set :
{A} 3
{B} 2
{C} 2


Frequent 2 -item set :
{A C} 2
```

Experiment No 7

Title:- Extend program 6, to find association rules.

Theory:- Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction. The Association rule is very useful in analyzing datasets.



Walchand College of Engineering, Sangli.

Experiment No 7

Title - Extend program 6, to find Association rules.

Aim - To find Association Rule

Theory - Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction.
The Association rule is very useful in analyzing datasets.

Algorithm -
1) Read min-support and min-confidence
2) Read data from excel / csv file
3) calculate freq itemset
4) generate Association rule for all frequent itemset

Example -
consider the following dataset and we will find frequent itemset and generate Association rule for them

Page No.

TID	Items
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3

Minimum support count = 2
Minimum confidence = 50 %

Step-1 : K = 1

C1 (Candidate set)

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

L1 : If sup_count < min_support then remove items

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

step : 2 :- K = 2

C2 :

Itemset	sup_count
I1, I2	4
I1, I3	4
I1, I4	1
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2
I3, I4	0
I3, I5	1
I4, I5	0

L2 :

Itemset	sup_count
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2

step 3 :- K = 3

C3 -

Itemset	sup_count
I1, I2, I3	2
I1, I2, I5	2

L3 :-

Itemset	sup_count
I1, I2, I3	2
I1, I2, I5	2

Step 4 -

Check all subsets of these itemset are frequent or not. Here itemset formed by joing L_3 is $\{I_1, I_2, I_3, I_5\}$ so its subset contain $\{I_1, I_3, I_5\}$ which is not frequent so no itemset in C_4

We stop here because no frequent itemset are found further.

Thus we have discovered all frequent item-sets

For Association Rule,

We need to find confidence

$$\text{confidence}(A \to B) = \frac{\text{sup\_count}(A \cup B)}{\text{sup\_count}(A)}$$

Itemset - $\{I_1, I_2, I_3\}$ from L_3

so rule can be

$[I_1 \wedge I_2] \to [I_3] \Rightarrow$ confidence $= 2/4 \times 100 = 50\%$

$[I_1 \wedge I_3] \to [I_2] \Rightarrow$ confidence $= 2/4 \times 100 = 50\%$

$[I_2 \wedge I_3] \to [I_1] \Rightarrow$ confidence $= 2/4 \times 100 = 50\%$

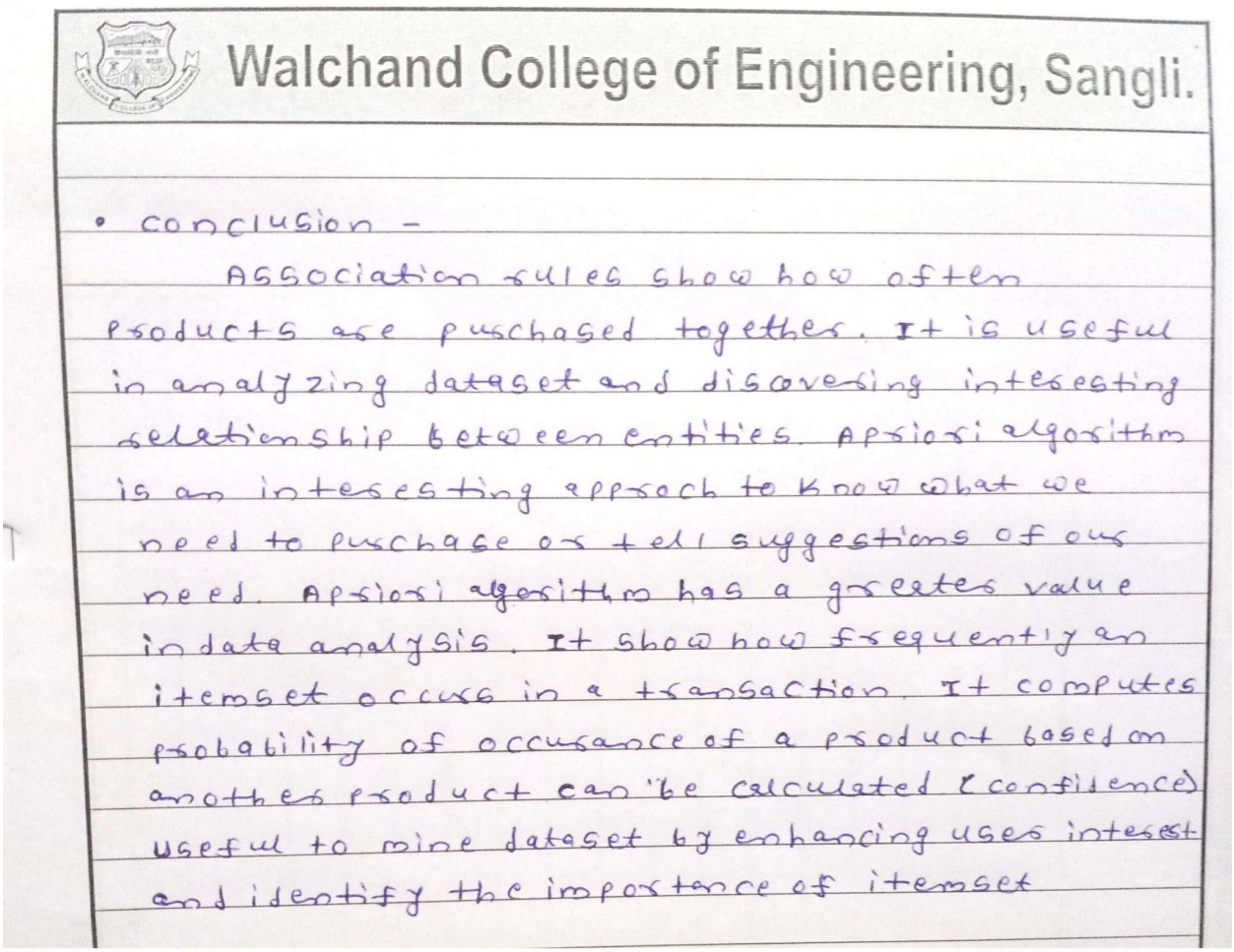
$[I_1] \to [I_2 \wedge I_3] \Rightarrow$ confidence $= 2/6 \times 100 = 33\%$

$[I_2] \to [I_1 \wedge I_3] \Rightarrow$ confidence $= 2/7 \times 100 = 28\%$

$[I_3] \to [I_1 \wedge I_2] \Rightarrow$ confidence $= 2/6 \times 100 = 33\%$

So if min confidence is 50% then first 3 rules can be considered as strong Association rules

$I_1 \wedge I_2 \to I_3$
$I_2 \wedge I_3 \to I_1$
$I_1 \wedge I_3 \to I_2$



Program:-

```
#include <bits/stdc++.h>
#include <map>
using namespace std;

ifstream fin;
double minfre;
vector<set<string>> datatable;
set<string> products;
map<string, int> freq;
```

```
double confidence;

vector<string> wordsof(string str)
{
   vector<string> tmpset;
   string tmp = "";
   int i = 0;
   while (str[i])
   {
       if (isalnum(str[i]))
           tmp += str[i];
       else
       {
           if (tmp.size() > 0)
               tmpset.push_back(tmp);
           tmp = "";
       }
       i++;
   }

   if (tmp.size() > 0)
       tmpset.push_back(tmp);

   return tmpset;
}

string combine(vector<string> &arr, int miss)
{
   string str;
   for (int i = 0; i < arr.size(); i++)
       if (i != miss)
           str += arr[i] + " ";
   str = str.substr(0, str.size() - 1);
   return str;
}

set<string> cloneit(set<string> &arr)
{
   set<string> dup;
```

```
   for (set<string>::iterator it = arr.begin(); it != arr.end();
it++)
       dup.insert(*it);
   return dup;
}

set<string> apriori_gen(set<string> &sets, int k)
{
   set<string> set2;
   for (set<string>::iterator it1 = sets.begin(); it1 != sets.end();
it1++)
   {
       set<string>::iterator it2 = it1;
       it2++;
       for (; it2 != sets.end(); it2++)
       {
           vector<string> v1 = wordsof(*it1);
           vector<string> v2 = wordsof(*it2);

           bool alleq = true;
           for (int i = 0; i < k - 1 && alleq; i++)
               if (v1[i] != v2[i])
                   alleq = false;

           v1.push_back(v2[k - 1]);
           if (v1[v1.size() - 1] < v1[v1.size() - 2])
               swap(v1[v1.size() - 1], v1[v1.size() - 2]);

           for (int i = 0; i < v1.size() && alleq; i++)
           {
               string tmp = combine(v1, i);
               if (sets.find(tmp) == sets.end())
                   alleq = false;
           }
           if (alleq)
               set2.insert(combine(v1, -1));
       }
   }
   return set2;
```

```
}

int countOccurences(vector<string> v)
{

   int count = 0;

   for (auto s : datatable)
   {
       bool present = true;

       for (auto x : v)
       {
           if (s.find(x) == s.end())
           {
               present = false;
               break;
           }
       }

       if (present)
           count++;
   }

   return count;
}

ofstream fw1("exp7_output.csv", ios::out);

void subsets(vector<string> items, vector<string> v1, vector<string>
v2, int idx)
{

   if (idx == items.size())
   {
       if (v1.size() == 0 || v2.size() == 0)
           return;

       int count1 = countOccurences(items); // Total support
```

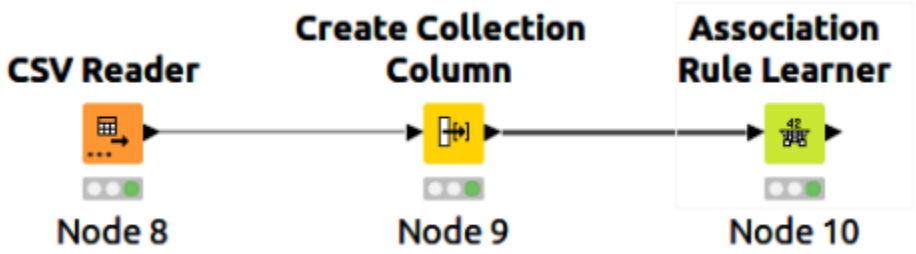
```
        int count2 = countOccurences(v1);

        double conf = (((double)count1) / count2) * 100;

        if (conf >= confidence)
        {

            fw1 << "{ ";
            for (auto s : v1)
            {
                fw1 << s << " ";
            }

            fw1 << "} ,"
                << "-> "
                << ", {";

            for (auto s : v2)
            {
                fw1 << s << " ";
            }

            fw1 << "} , " << conf << endl;
        }

        return;
    }

    v1.push_back(items[idx]);
    subsets(items, v1, v2, idx + 1);

    v1.pop_back();
    v2.push_back(items[idx]);
    subsets(items, v1, v2, idx + 1);
    v2.pop_back();
}

void generateAssociationRules(set<string> freqItems)
{
```

```
    for (auto it = freqItems.begin(); it != freqItems.end(); it++)
    {
        vector<string> items = wordsof(*it);

        subsets(items, {}, {}, 0);
    }
}

int main()
{
    fin.open("exp7_input.csv", ios::in);

    if (!fin.is_open())
    {
        perror("Error in opening file : ");
    }
    cout << "Enter Support % :";
    cin >> minfre;

    cout << "Enter Confidence % : ";
    cin >> confidence;

    string str;
    while (!fin.eof())
    {
        getline(fin, str);
        vector<string> arr = wordsof(str);
        set<string> tmpset;
        for (int i = 0; i < arr.size(); i++)
            tmpset.insert(arr[i]);
        datatable.push_back(tmpset);

        for (set<string>::iterator it = tmpset.begin(); it !=
tmpset.end(); it++)
        {
            products.insert(*it);
            freq[*it]++;
        }
    }
```

```
   fin.close();
   // cout<<datatable.size()<<endl;
   cout << "No of transactions: " << datatable.size() << endl;
   minfre = minfre * datatable.size()/100;
   cout << "Min frequency:" << minfre << endl;

   queue<set<string>::iterator> q;
   for (set<string>::iterator it = products.begin(); it !=
products.end(); it++)
       if (freq[*it] < minfre)
           q.push(it);

   while (q.size() > 0)
   {
       products.erase(*q.front());
       q.pop();
   }

   int pass = 1;
   cout << "\nFrequent " << pass++ << " -item set : \n";
   for (set<string>::iterator it = products.begin(); it !=
products.end(); it++)
       cout << "{" << *it << "} " << freq[*it] << endl;

   int i = 2;
   set<string> prev = cloneit(products);

   while (i)
   {
       set<string> cur = apriori_gen(prev, i - 1);

       if (cur.size() < 1)
       {
           break;
       }

       for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
       {
```

```
            vector<string> arr = wordsof(*it);

            int tot = 0;
            for (int j = 0; j < datatable.size(); j++)
            {
                bool pres = true;
                for (int k = 0; k < arr.size() && pres; k++)
                    if (datatable[j].find(arr[k]) ==
datatable[j].end())
                        pres = false;
                if (pres)
                    tot++;
            }
            if (tot >= minfre)
                freq[*it] += tot;
            else
                q.push(it);
        }

        while (q.size() > 0)
        {
            cur.erase(*q.front());
            q.pop();
        }

        // cout << "Flag : " << flag << "\n";
        bool flag = true;

        for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
        {
            vector<string> arr = wordsof(*it);

            if (freq[*it] < minfre)
                flag = false;
        }

        if (cur.size() == 0)
            break;
```

```
        cout << "\n\nFrequent " << pass++ << " -item set : \n";
        for (set<string>::iterator it = cur.begin(); it != cur.end();
it++)
            cout << "{" << *it << "} " << freq[*it] << endl;

        prev = cloneit(cur);
        i++;
    }

    generateAssociationRules(prev);

    return 1;
}
```



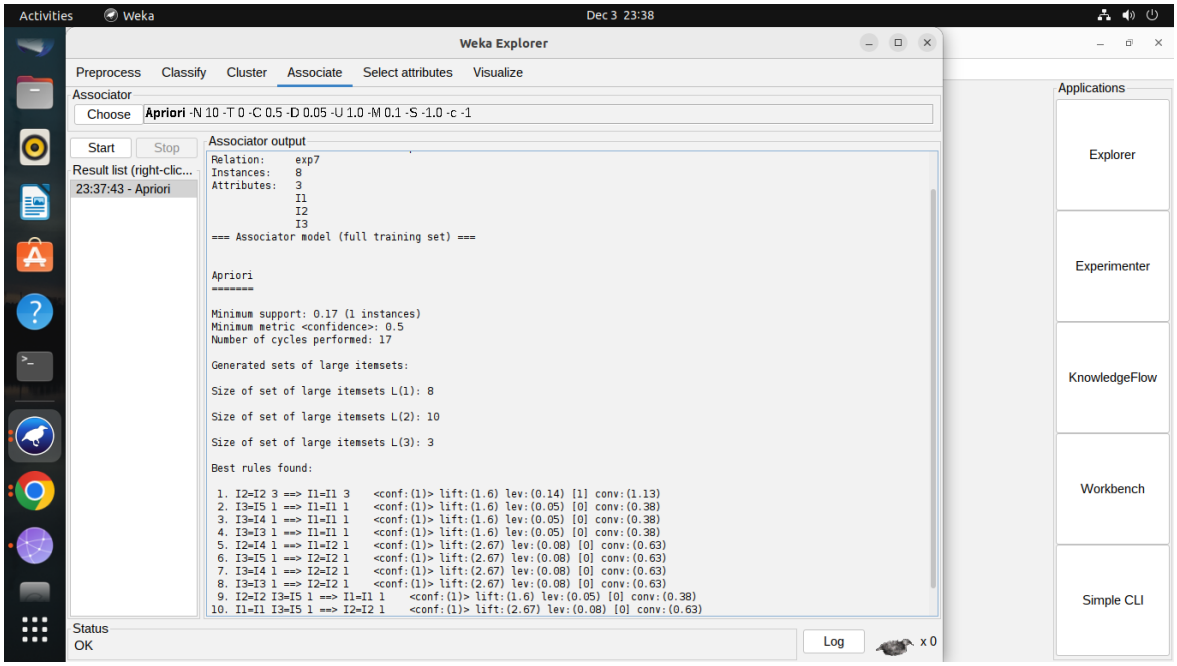
Input file :- Input for Association Rule

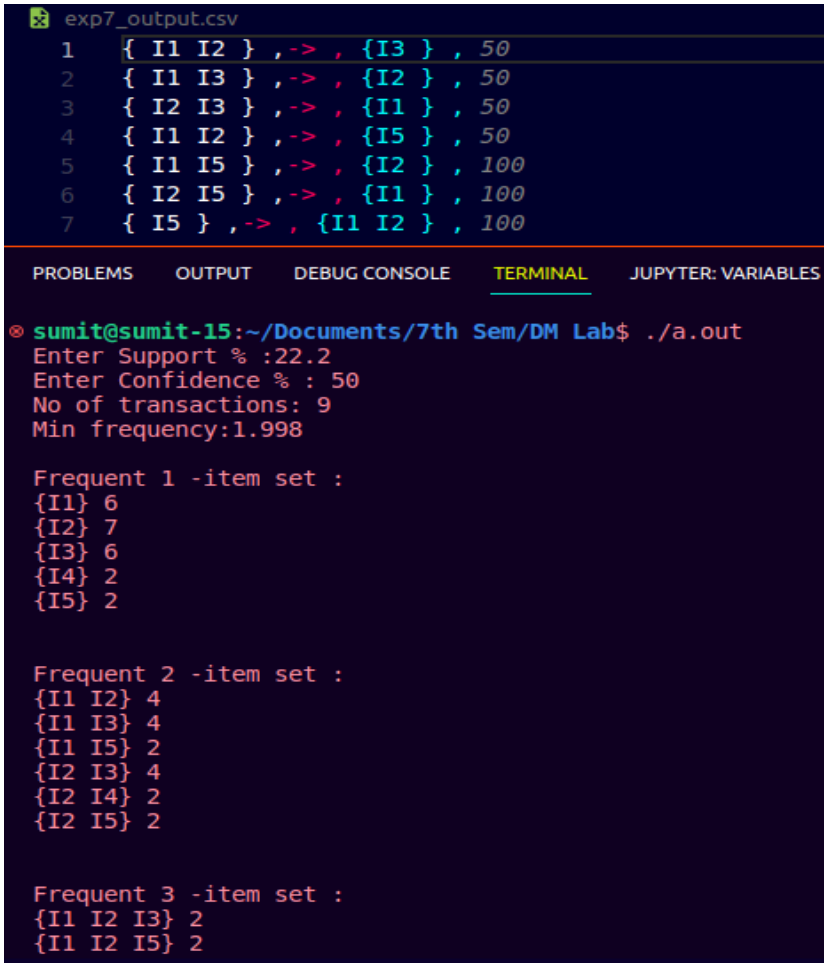
Output file:- Output - Association Rule

Output:-

Table "default" - Rows: 29 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	Support	Confidence	Lift	Consequent	implies	Items
rule0	0.222	1	1.5	I1	<---	[I2,I5]
rule1	0.222	1	1.286	I2	<---	[I1,I5]
rule2	0.222	0.5	2.25	I5	<---	[I1,I2]
rule3	0.222	0.5	0.75	I1	<---	[I2,I3]
rule4	0.222	0.5	0.643	I2	<---	[I1,I3]
rule5	0.222	0.5	0.75	I3	<---	[I1,I2]
rule6	0.222	1	1.286	I2	<---	[?,I4]
rule7	0.222	1	1.125	?	<---	[I2,I4]
rule8	0.333	0.5	0.75	I1	<---	[I2,?]
rule9	0.333	0.6	0.771	I2	<---	[I1,?]
rule10	0.333	0.75	0.844	?	<---	[I1,I2]
rule11	0.333	0.6	0.9	I1	<---	[?,I3]
rule12	0.333	0.75	0.844	?	<---	[I1,I3]
rule13	0.333	0.6	0.9	I3	<---	[I1,?]
rule14	0.333	0.6	0.771	I2	<---	[?,I3]
rule15	0.333	0.75	0.844	?	<---	[I2,I3]
rule16	0.333	0.5	0.75	I3	<---	[I2,?]
rule17	0.444	0.571	0.857	I1	<---	[I2]
rule18	0.444	0.667	0.857	I2	<---	[I1]
rule19	0.444	0.667	1	I1	<---	[I3]
rule20	0.444	0.667	1	I3	<---	[I1]
rule21	0.444	0.667	0.857	I2	<---	[I3]
rule22	0.444	0.571	0.857	I3	<---	[I2]
rule23	0.556	0.625	0.938	I1	<---	[?]
rule24	0.556	0.833	0.938	?	<---	[I1]
rule25	0.556	0.833	0.938	?	<---	[I3]
rule26	0.556	0.625	0.938	I3	<---	[?]
rule27	0.667	0.75	0.964	I2	<---	[?]
rule28	0.667	0.857	0.964	?	<---	[I2]


Activities
Weka
Dec 3 23:38
Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-clic...
23:37:43 - Apriori
Associator output
Relation: exp7
Instances: 8
Attributes: 3
I1
I2
I3
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.17 (1 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 17
Generated sets of large itemsets:
Size of set of large itemsets L(1): 8
Size of set of large itemsets L(2): 10
Size of set of large itemsets L(3): 3
Best rules found:
1. I2=I2 3 ==> I1=I1 3 <conf:(1)> lift:(1.6) lev:(0.14) [1] conv:(1.13)
2. I3=I5 1 ==> I1=I1 1 <conf:(1)> lift:(1.6) lev:(0.05) [0] conv:(0.38)
3. I3=I4 1 ==> I1=I1 1 <conf:(1)> lift:(1.6) lev:(0.05) [0] conv:(0.38)
4. I3=I3 1 ==> I1=I1 1 <conf:(1)> lift:(1.6) lev:(0.05) [0] conv:(0.38)
5. I2=I4 1 ==> I1=I2 1 <conf:(1)> lift:(2.67) lev:(0.08) [0] conv:(0.63)
6. I3=I5 1 ==> I2=I2 1 <conf:(1)> lift:(2.67) lev:(0.08) [0] conv:(0.63)
7. I3=I4 1 ==> I2=I2 1 <conf:(1)> lift:(2.67) lev:(0.08) [0] conv:(0.63)
8. I3=I3 1 ==> I2=I2 1 <conf:(1)> lift:(2.67) lev:(0.08) [0] conv:(0.63)
9. I2=I2 I3=I5 1 ==> I1=I1 1 <conf:(1)> lift:(1.6) lev:(0.05) [0] conv:(0.38)
10. I1=I1 I3=I5 1 ==> I2=I2 1 <conf:(1)> lift:(2.67) lev:(0.08) [0] conv:(0.63)
Status
OK
Log
x 0
Applications
Explorer
Experimenter
KnowledgeFlow
Workbench
Simple CLI


exp7_output.csv
1 { I1 I2 } ,-> , {I3 } , 50
2 { I1 I3 } ,-> , {I2 } , 50
3 { I2 I3 } ,-> , {I1 } , 50
4 { I1 I2 } ,-> , {I5 } , 50
5 { I1 I5 } ,-> , {I2 } , 100
6 { I2 I5 } ,-> , {I1 } , 100
7 { I5 } ,-> , {I1 I2 } , 100
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER: VARIABLES
sumit@sumit-15:~/Documents/7th Sem/DM Lab$ ./a.out
Enter Support % :22.2
Enter Confidence % : 50
No of transactions: 9
Min frequency:1.998
Frequent 1 -item set :
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2
Frequent 2 -item set :
{I1 I2} 4
{I1 I3} 4
{I1 I5} 2
{I2 I3} 4
{I2 I4} 2
{I2 I5} 2
Frequent 3 -item set :
{I1 I2 I3} 2
{I1 I2 I5} 2

Experiment No 8

Title:- Find correlation between items/entities.

Theory:- Correlation coefficients are used in statistics to measure how strong a relationship is between two variables. If the value of correlation coefficient is 0 then no relation exists between two variables. If the value is less than 0 then negative correlation and greater than 0 then positive correlation.

Experiment No 8

Title - Find correlation between items / entities

Aim - To find correlation between items / entities and find correlation coefficient

Theory - Correlation coefficients are used in statistics to measure how strong a relationship is between two variables. If the value of correlation coefficient is 0 then no relation exists between two variables. If the value is less then 0 then negative correlation and greater than 0 then positive correlation.

Formula -

$$\text{Correlation coefficient} = \frac{P(A \cup B)}{P(A) \cdot P(B)}$$

Algorithm -

1) Take input Dataset in csv / excel file
2) If needed convert data entries into binary (1/0) or boolean (Y/N) format
3) To find correlation coefficient, take input from user that in which two entities user wish to find correlation.
4) Find 'Y' count for entity 1 as well as entity 2 and count the events at which both the entities are 'yes'
5) Apply formula and find correlation coefficient.

Example -

Tid	M	T	W	Th	F	S
1	Y	Y	N	N	Y	N
2	N	Y	Y	N	N	Y
3	Y	Y	Y	N	Y	Y
4	N	N	N	Y	Y	Y

For 'y' value

Correlation ratio (1-2) = $\frac{1}{3 \times 3} = \frac{1}{9}$

correlation ratio (1-3) = $\frac{3}{3 \times 5} = \frac{1}{5}$

correlation ratio (1-4) = $\frac{1}{3 \times 3} = \frac{1}{9}$

Correlation ratio (2-3) = $\frac{3}{3 \times 5} = \frac{1}{5}$

correlation ratio (2-4) = $\frac{1}{3 \times 3} = \frac{1}{9}$

correlation ratio (3-4) = $\frac{2}{5 \times 3} = \frac{2}{15}$

Conclusion -

By using correlation, the study of closeness of relationship between different entities i.e degree to which variables are associated can be carried out. To find correlation, statistical measure, correlation coefficient used which describe the strength and direction of an association between variables.

Program:-

```
import openpyxl
import random

wb_obj = openpyxl.load_workbook("Correlation_Input.xlsx")
sheet_obj = wb_obj.active
n = sheet_obj.max_row-1


#function for finding correlation
def find_correlation(tid1, tid2):
    tid1_count = 0
    tid2_count = 0
    total_common_count = 0 #count of same "yes" count in two transaction
simultaneously
    for j in range (2,9):
        if (sheet_obj.cell(row = tid1+1, column = j).value) == "Y":
            tid1_count += 1
        if (sheet_obj.cell(row = tid2+1, column = j).value) == "Y":
            tid2_count += 1
        if ((sheet_obj.cell(row = tid1+1, column = j).value) == "Y") and ((sheet_obj.cell(row
= tid2+1, column = j).value) == "Y"):
            total_common_count += 1
    if(tid1_count == 0 or tid2_count == 0):
        return 0
    return total_common_count/(tid1_count * tid2_count)


data = []
for i in range(1,n+1):
    for j in range(i+1,n+1):
        ans = find_correlation(i,j)
        if(ans == 0):
            verdict = "No relationship between entities"
        elif(ans < 0):
            verdict = "Negative correlation"
        elif(ans > 0):
            verdict = "Positive correlation"
```

```
        else:
            verdict = "Not defined"
        print ("Correlation ratio " + str(i) + " & " + str(j) + " = " + str(ans) + "  " + verdict +
"\n")
        list  = [str(i),str(j),str(ans),verdict]
        data.append(list)


# writing answer to file

workbook = openpyxl.Workbook()
sheet_obj = workbook.active

sheet_obj.cell(row = 1 , column = 1).value = "item 1 with tid"
sheet_obj.cell(row = 1 , column = 2).value = "item 2 with tid"
sheet_obj.cell(row = 1 , column = 3).value = "Correlation coefficient"
sheet_obj.cell(row = 1 , column = 4).value = "Type of correlation"
for i in range(0,len(data)):
    for j in range(0,len(data[i])):
        sheet_obj.cell(row = i+2 , column = j+1).value = data[i][j]

workbook.save("Correlation_output.xlsx")
```

Input file:- **Input Dataset**

Output file:- **Correlation Output**

Experiment No 9

Title:- Distance and cluster

Theory:- Clustering consists of grouping certain objects that are similar to each other, it can be used to decide if two items are similar or dissimilar in their properties.
Euclidean distance is considered the traditional metric for problems with geometry. It can be simply explained as the ordinary distance between two points. It is one of the most used algorithms in cluster analysis. One of the algorithms that use this formula would be K-mean. Mathematically it computes the root of squared differences between the coordinates between two objects.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Experiment NO 9.

Title - Distance and cluster

Aim - To compute centre of cluster assuming all multidimensional points belonging to one cluster.
- To find distance of all points with obtained cluster centre using suitable distance f^n
- To display result in upper triangular or lower triangular matrix

Theory - K-means clustering is an unsupervised learning algorithm which groups the unlabelled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process as if K = 2, there will be two clusters that need to be created, if K = 3, there will be 3 clusters It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabelled dataset on its own without the need for any training The algorithm takes the dataset and divides it into K-numbers of clusters and it repeats the process until it does not find the best cluster.

The K-mean clustering algorithm mainly performs two tasks -

i) Determines the best value for K centers points or centroid by an iterative process

ii) Assign each data point to its closest K-centers. Those data points which are near to the particular K-centers, creates a clusters.

Algorithm -

1) Choose number of clusters.
2) Randomly select any K data points as clusters centers. Select clusters centers in such a way that they are farthest.
3) Calculate distance between each point and each clusters centers. Any distance function can be used here such as "Euclidian function"
4) Assign each data point to some clusters A data point is assigned to that clusters whose centre is nearest to that data point
5) Recompute centre by taking mean.
6) Keep repeating step (3) to step (5) until centers do not change or data points remain in same clusters or maximum iterations are reached.

Example -

Let's the points be -

points	co-ordinate
P1	(10, 40)
P2	(20, 10)
P3	(15, 20)
P4	(25, 30)
P5	(15, 5)

As a given in the problem statement, considering these 5 points as a part of single cluster

Imaginary centre $C'(x', y')$

$$x' = \frac{10 + 20 + 15 + 25 + 15}{5} = \frac{85}{5} = 17$$

$$y' = \frac{40 + 10 + 20 + 30 + 5}{5} = \frac{105}{5} = 21$$

$$C'(x', y') \equiv (17, 21)$$

Using Euclidian formula, finding distances

$$d = \sqrt{(x' - x)^2 + (y' - y)^2}$$

$$d(C', P1) = \sqrt{(17-10)^2 + (40-21)^2} = 20.24$$
$$d(C', P2) = \sqrt{(17-20)^2 + (21-10)^2} = 11.40$$
$$d(C', P3) = \sqrt{(17-15)^2 + (21-20)^2} = 2.23$$
$$d(C', P4) = \sqrt{(17-25)^2 + (21-30)^2} = 12.04$$
$$d(C', P5) = \sqrt{(17-15)^2 + (21-5)^2} = 16.12$$

∴ The Nearest point from imaginary centre is P3 (15, 20)

Now finding distances between points considering P3 as center and plotting triangular Matrix

Distance Matrix	P1	P2	P3	P4	P5
P1	0				
P2	31.62	0			
P3	20.61	11.18	0		
P4	18.02	20.61	14.14	0	
P5	35.35	7.07	15	26.92	0

- conclusion -

K-means clustering partitions dataset into K predefined distinct non-overlapping subgroups (clusters) where each point belongs to only one group. clustering can be done using suitable distance function and distance matrix points with closeness can be merged together to form a single cluster.

Program:-

```
#include<bits/stdc++.h>
#include<limits>
using namespace std;

float distance(float x1,float y1,int x2,int y2)
{
    // float x2 = float(x2);
    // float y2 = float(y2);
    return sqrt(((float)x2-x1)*((float)x2-x1)+((float)y2-y1)*((float)y2-y1));
}

int main()
{
    string line;
    int mid_point;
    string point,x,y;
    int i=0;
    int val1;
    int val2;
    vector<pair<int,int>>v;
    fstream in("cluster_input.csv",ios::in);
    if(!in.is_open())
    {
        cout<<"couldn't open file";
        return -1;
    }
    while(getline(in,line))
    {
        stringstream str(line);
        if(i==0)
        {
            i++;
            continue;
        }
        getline(str,point,',');
        getline(str,x,',');
        getline(str,y,',');

        val1 = stoi(x);
        val2 = stoi(y);
```

```
    v.push_back({val1,val2});
}

int n = v.size();
for(int i=0;i<v.size();i++)
{
    int first = v[i].first;
    int second = v[i].second;
    // cout<<first<<" "<<second<<endl;
}
int x_sum =0,y_sum=0;
for(int i=0;i<v.size();i++)
{
    int first = v[i].first;
    int second = v[i].second;

    x_sum += first;
    y_sum += second;

}
float mid_x = (float) x_sum/n;
float mid_y = (float) y_sum/n;
cout<<"Mid Point: "<<"("<< mid_x<<","<<mid_y<<")"<<endl;

ofstream out("cluster_output.csv");
out<<" , p1 ,p2 ,p3 ,p4,C";
out<<"\n";
for(int i=0;i<v.size();i++)
{
    if(i < v.size())
        out<<"p"<<i+1<<",";

    for(int j=0;j<=i;j++)
    {
        int f_x1 = v[i].first;
        int s_y1 = v[i].second;
        int f_x2 = v[j].first;
        int s_y2 = v[j].second;

        if(f_x1==f_x2 && s_y1 == s_y2)
        {
            out<<"0"<<",";
            break;
        }
        float dis = distance(f_x1,s_y1,f_x2,s_y2);
```

```
        out<< dis<<",";

    }
    out<<"\n";
}
out<<"C"<<",";
pair<int,int>p;
int ans=0;
float x_new;
float y_new;
float nearer=INT_MAX;
for(int i=0;i<v.size();i++)
{
    int first = v[i].first;
    int second = v[i].second;



    float d = distance(mid_x,mid_y,first,second);
    cout<<"Distance of p"<<i+1 <<" from centre: "<<d<<endl;


    if(nearer > d)
    {
        nearer = d;
        ans = i+1;
        x_new = first;
        y_new = second;
    }



    out<<d<<",";

    if(i==v.size()-1)
        out<<"0"<<",";
}


cout<<"Nearer Distance: "<<nearer<<endl;
cout<<"\n";
cout<<"Nearest point from Centre is: "<<"p"<<ans<<endl;
out<<",";
out<<"\n";
```

```
//New Centre
out<<" , p1 ,p2 ,p3 ,p4";
out<<"\n";
for(int i=0;i<v.size();i++)
{

    if(i < v.size())
        out<<"p"<<i+1<<",";

    for(int j=0;j<=i;j++)
    {
        int f_x1 = v[i].first;
        int s_y1 = v[i].second;
        int f_x2 = v[j].first;
        int s_y2 = v[j].second;

        if(f_x1==f_x2 && s_y1 == s_y2)
        {
            out<<"0"<<",";
            break;
        }
        float dis = distance(f_x1,s_y1,f_x2,s_y2);


        out<< dis<<",";

    }
    out<<"\n";
}

out<<"p"<<ans<<"(New Center)"<< ",";
for(int i=0;i<v.size();i++)
{
    int first = v[i].first;
    int second = v[i].second;



    float d = distance(x_new,y_new,first,second);
    cout<<"Distance of p"<<i+1 <<" from "<<"p"<<ans<<":"<<d<<endl;


    out<<d<<",";
```

```
        if(i==v.size()-1)
            out<<"0"<<",";
    }

    return 0;
}
```

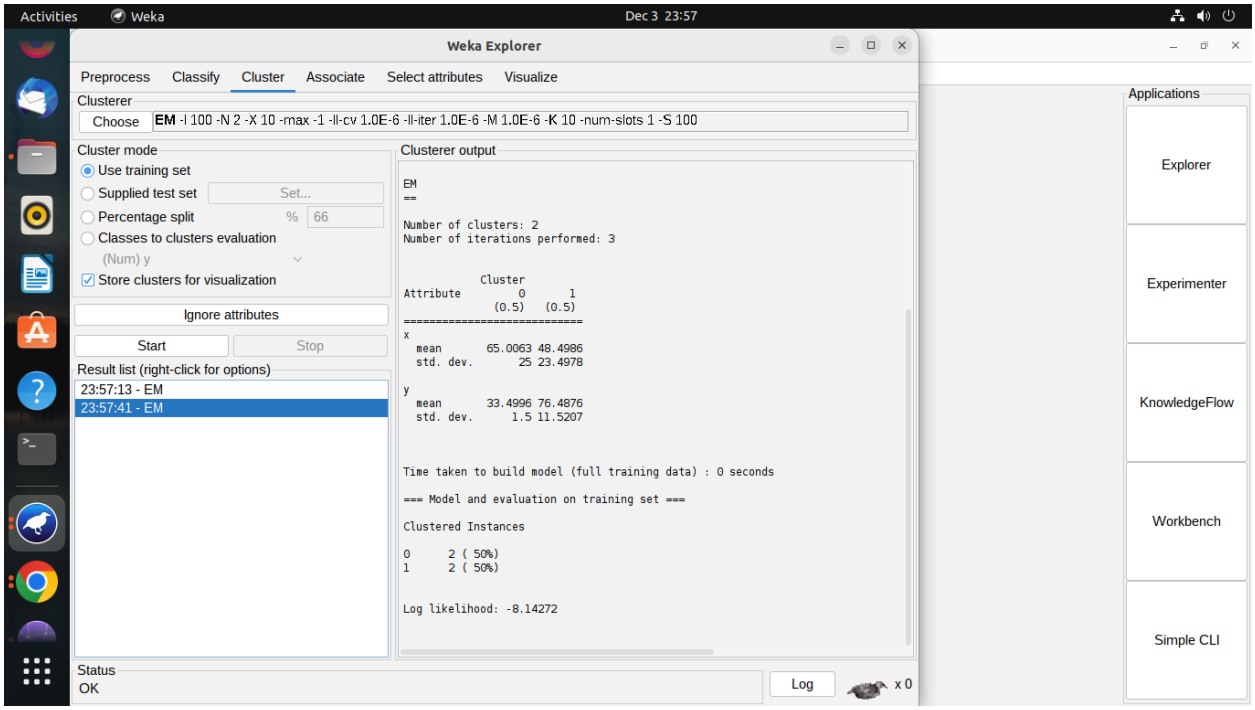
Input file:- [Input points]

Output file:- [Output in Lower triangular matrix]

Output:-

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

Row ID	Points	x	y	Cluster
Row0	p1	25	65	cluster_0
Row1	p2	72	88	cluster_1
Row2	p3	40	35	cluster_0
Row3	p4	90	32	cluster_1

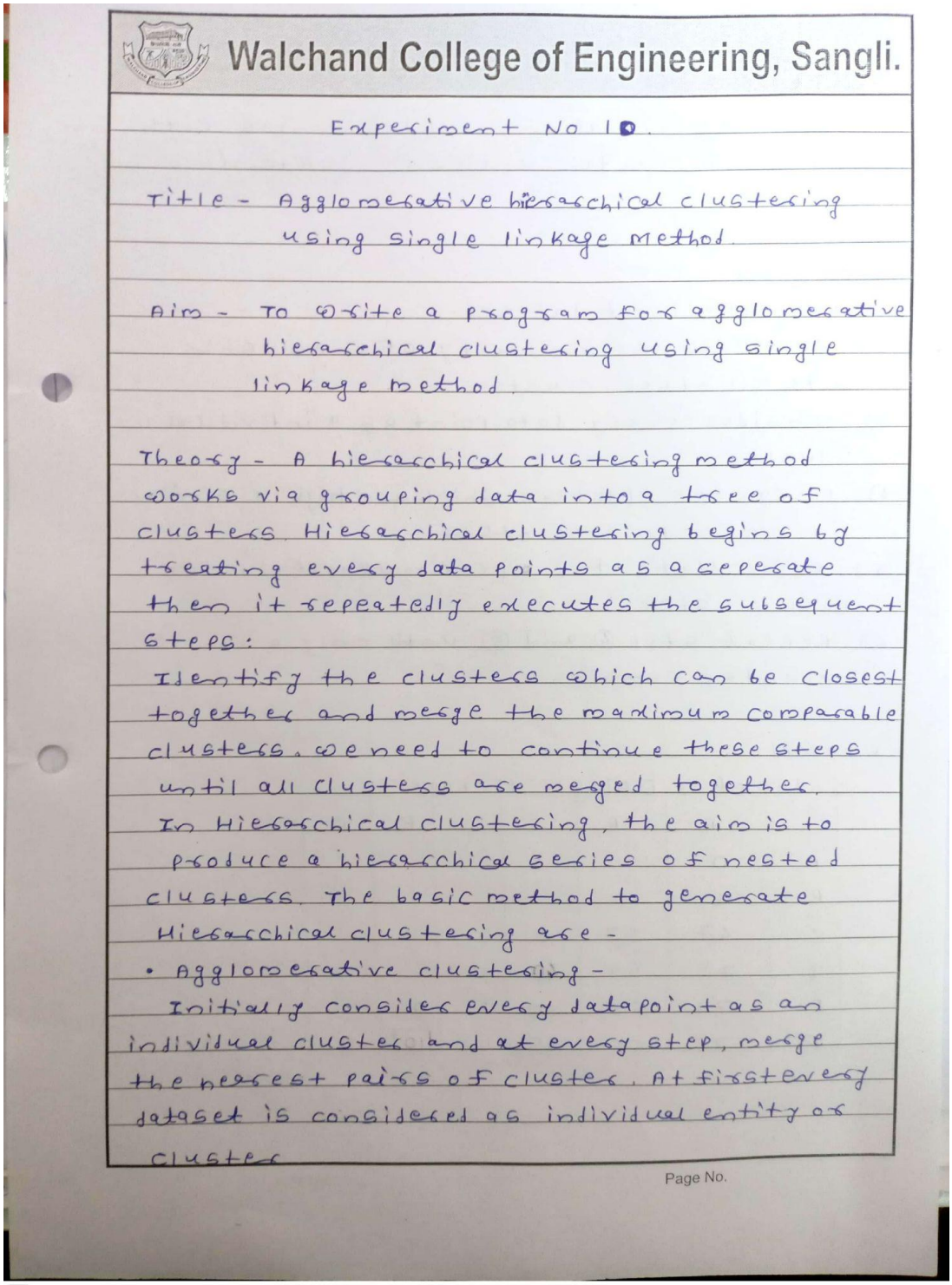



```
it@it:~/Documents/New/Sumit$ g++ clustering.cpp
it@it:~/Documents/New/Sumit$ ./a.out
Mid Point: (56.75,55)
Distance of p1 from centre: 33.2876
Distance of p2 from centre: 36.3533
Distance of p3 from centre: 26.0876
Distance of p4 from centre: 40.4297
Nearer Distance: 26.0876

Nearest point from Centre is: p3
Distance of p1 from p3:33.541
Distance of p2 from p3:61.9112
Distance of p3 from p3:0
Distance of p4 from p3:50.0899
```

	column 1	column 2	column 3	column 4	column 5	column 6
1		p1	p2	p3	p4	
2	p1	0				
3	p2	52.3259	0			
4	p3	33.541	61.9112	0		
5	p4	72.8972	58.8218	50.0899	0	
6	C	33.2876	36.3533	26.0876	40.4297	0
7		p1	p2	p3	p4	
8	p1	0				
9	p2	52.3259	0			
10	p3	33.541	61.9112	0		
11	p4	72.8972	58.8218	50.0899	0	
12	p3(New Centre)	33.541	61.9112	0	50.0899	0

Experiment No 10

Title :- Agglomerative Hierarchical clustering using single linkage method



Walchand College of Engineering, Sangli.

Experiment No 10.

Title - Agglomerative hierarchical clustering using single linkage method.

Aim - To write a program for agglomerative hierarchical clustering using single linkage method.

Theory - A hierarchical clustering method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a seperate then it repeatedly executes the subsequent steps:

Identify the clusters which can be closest together and merge the maximum comparable clusters. We need to continue these steps until all clusters are merged together.

In Hierarchical clustering, the aim is to produce a hierarchical series of nested clusters. The basic method to generate Hierarchical clustering are -

- Agglomerative clustering -

Initially consider every datapoint as an individual cluster and at every step, merge the nearest pairs of cluster. At first every dataset is considered as individual entity or cluster

Page No.

At every iteration, clusters merges with different clusters until one cluster is formed.

Algorithm -

1) Read the input dataset.
2) Calculate the similarity of one cluster with all other clusters.
3) Consider every data point as a individual cluster.
4) Merge the clusters which are highly similar or close to each other.
5) Recalculate the proximate matrix for each cluster.
6) Repeat step ④ and ⑤ until only a single cluster remains.

Example -

Dataset - [Single Linkage]

	A	B	C	D	E	F
A	0					
B	16	0				
C	47	37	0			
D	72	57	40	0		
E	77	65	30	31	0	
F	79	66	35	23	10	0

To obtain New distance Matrix

merge (EF)

	A	B	C	D	EF
A	0				
B	[16]	0			
C	47	37	0		
D	72	57	40	0	
EF	77	65	30	23	0

Merge AB in this iteration.

	AB	C	D	EF
AB	0			
C	37	0		
D	57	40	0	
EF	65	30	[23]	0

Merging D and EF in next Iteration.

	AB	C	DEF
AB	0		
C	37	0	
DEF	57	[30]	0

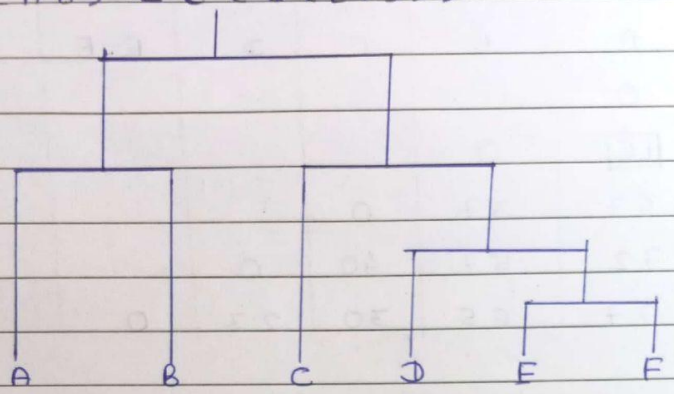
Now Merging C and DEF

	AB	CDEF
AB	0	
CDEF	37	0

Now the distance between cluster AB and CDEF is 37.

- Dendogram generated

(A B) C C (D (E F)))



Conclusion -

Hierarchical agglomerative clustering starts with treating each observation as an individual cluster i.e begins with singleton sets of each point that is each data point is its own cluster and then iteratively merges clusters until all the data points are merged into a single cluster. Dendrogram is generated to represent hierarchical relationship between object.

Program:-

```
#include <bits/stdc++.h>
using namespace std;

int op = 1;

ofstream fwtr("exp10_output.csv", ios::out);

string algomerative(string input)
{

   map<string, map<string, int>> dm;

   fstream file(input, ios::in);

   string line;
   getline(file, line);

   int pt = 0;
   stringstream st(line);

   int i = 0;
   string point;
   vector<string> points;
   while (getline(st, point, ','))
   {
       if (i == 0)
       {
           i++;
           continue;
       }
       points.push_back(point);
   }

   while (getline(file, line))
   {
```

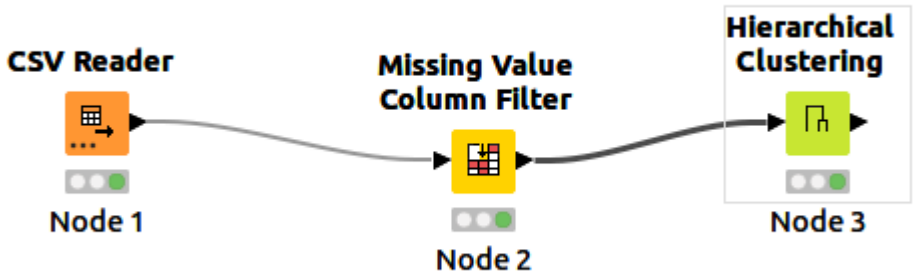
```
    stringstream str(line);

    getline(str, point, ',');

    string dist;
    int idx = 0;
    while (getline(str, dist, ','))
    {

        if (dist.length() != 0)
            dm[point][points[idx]] = stoi(dist);

        idx++;
    }
}

string pt1, pt2;
int min_dist = INT_MAX;

for (auto p : dm)
{
    for (auto pp : p.second)
    {

        string p1 = p.first, p2 = pp.first;
        int dist = pp.second;

        if (p1 != p2 && dist < min_dist)
        {
            pt1 = p1;
            pt2 = p2;
            min_dist = dist;
        }
    }
}

cout << "Clusters Choosen : " << pt1 << " " << pt2<<endl;

string up, down;
```

```
if (pt1[0] > pt2[0])
{
    up = pt2;
    down = pt1;
}
else
{
    up = pt1;
    down = pt2;
}

string newPt = down + up;

for (auto p : dm)
{
    point = p.first;
    if (point[0] > newPt[0])
    {
        dm[point][newPt] = min(dm[point][up], dm[point][down]);
    }
}

for (auto p : dm[down])
{
    point = p.first;

    int d1 = p.second;

    if (point[0] < up[0])
        d1 = min(d1, dm[up][point]);
    else
        d1 = min(d1, dm[point][up]);

    dm[newPt][point] = d1;
}

for (auto p : dm)
{
    point = p.first;
```

```
    auto mtemp = p.second;

    if (point[0] >= up[0])
    {
        int d1 = dm[point][up];

        if (down[0] > point[0])
            d1 = min(d1, dm[down][point]);
        else
            d1 = min(d1, dm[point][down]);

        dm[point][newPt] = d1;
        dm[point].erase(up);

        if (point[0] >= down[0])
            dm[point].erase(down);
    }
}

dm.erase(up);
dm.erase(down);

string output = "output" + to_string(op++) + ".csv";

ofstream fw(output, ios::out);
fw << ",";
for (auto p : dm)
{
    fw << p.first << ",";
}
fw << "\n";

for (auto p : dm)
{
    fw << p.first << ",";
    for (auto pp : p.second)
    {
        fw << pp.second << ",";
```

```
        }
        fw << "\n";
    }

    fw.close();

    fwtr << down << " & " << up << "\n";

    return output;
}

int main()
{
    string input = "exp10_input.csv";

    fstream file1(input, ios::in);

    string line;
    getline(file1, line);

    int pt = 0;
    stringstream st(line);

    int j = 0, len = 0;
    string point;
    while (getline(st, point, ','))
    {
        if (j == 0)
        {
            j++;
            continue;
        }
        len++;
    }

    for (int i = 1; i <= len - 2; i++)
    {
        string output = algomerative(input);
        input = output;
```

```
    }
    return 0;
}
```



Input file:- Input Distance Table

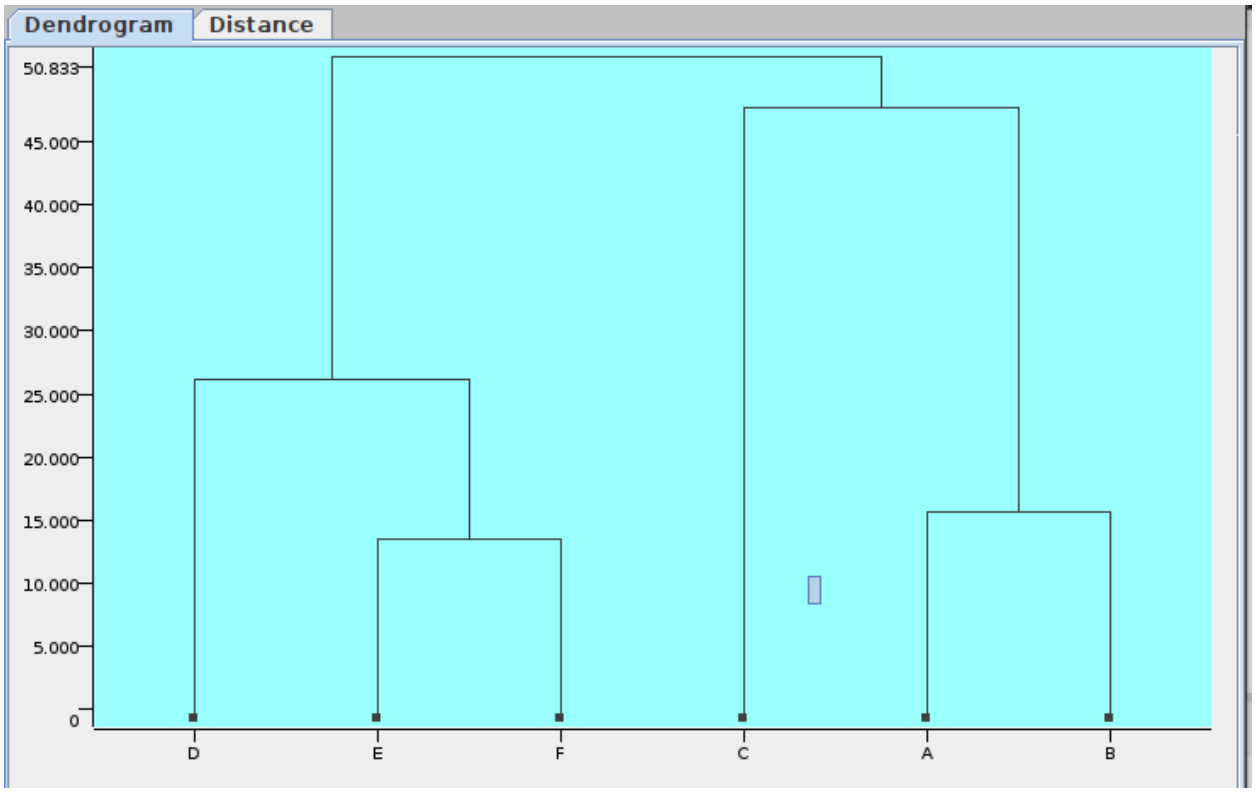
Output files:- i) First Grouping
ii) Second Grouping
iii) Third Grouping
iv) Fourth Grouping
v) Final Output

Output:-

File Edit Hilite Navigation View

Table "default" - Rows: 6 | Spec - Columns: 8 | Properties | Flow Variables

Row ID	S Colum...	I A	I B	I C	I D	I E	I F	S Cluster
Row2	C	47	37	0	?	?	?	cluster_0
Row0	A	0	?	?	?	?	?	cluster_1
Row1	B	16	0	?	?	?	?	cluster_1
Row3	D	72	57	40	0	?	?	cluster_2
Row4	E	77	65	30	31	0	?	cluster_2
Row5	F	79	66	35	23	10	0	cluster_2


Dendrogram
Distance
50.833
45.000
40.000
35.000
30.000
25.000
20.000
15.000
10.000
5.000
0
D
E
F
C
A
B

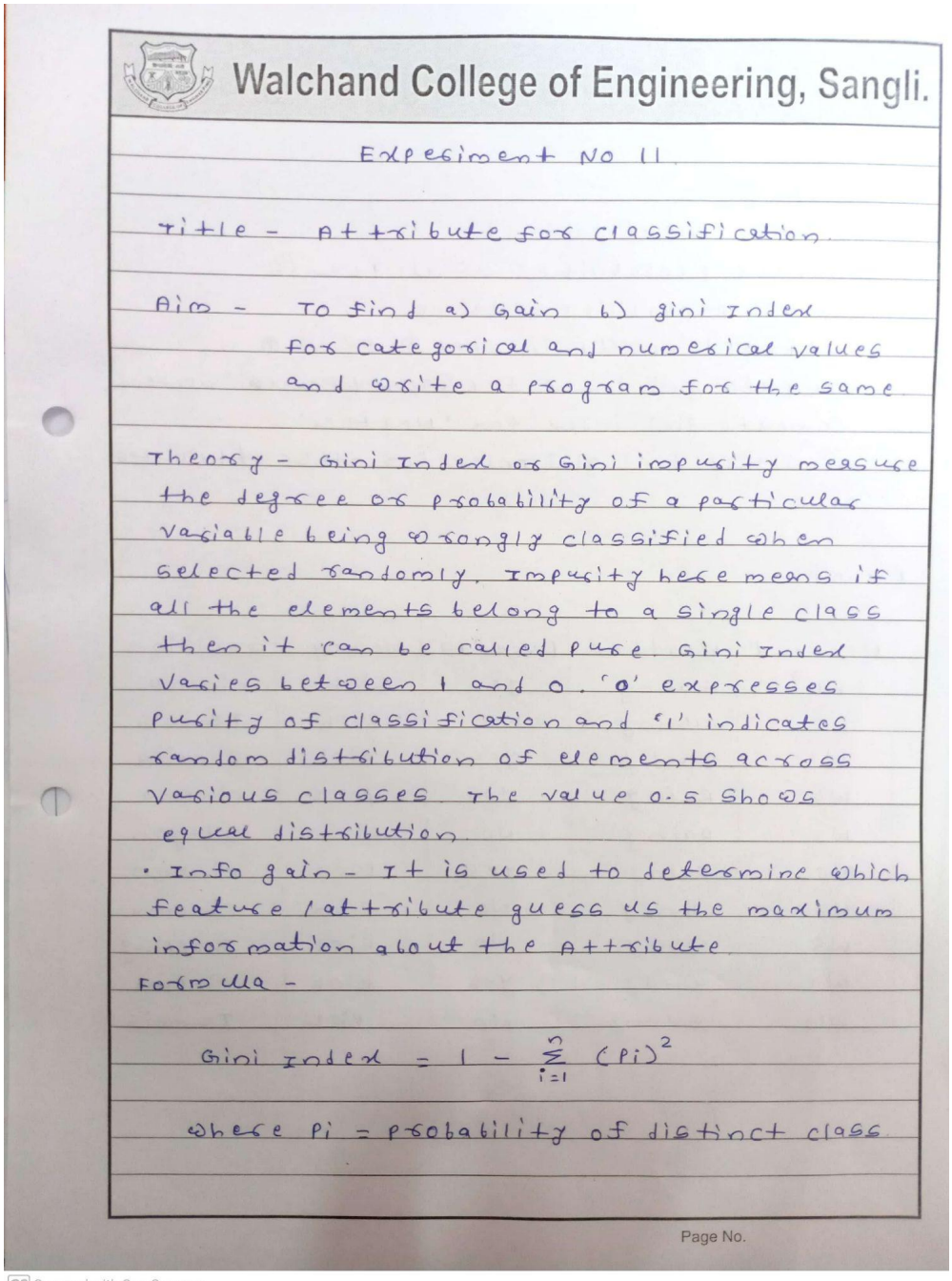
Experiment No 11

Title:- Attribute for classification ,Write a program to find

A. Gain

B. Gini index

For categorical and numerical values



Walchand College of Engineering, Sangli.

Experiment No 11

Title - Attribute for classification.

Aim - To find a) Gain b) gini index for categorical and numerical values and write a program for the same

Theory - Gini Index or Gini impurity measure the degree or probability of a particular variable being wrongly classified when selected randomly. Impurity here means if all the elements belong to a single class then it can be called pure. Gini Index varies between 1 and 0. '0' expresses purity of classification and '1' indicates random distribution of elements across various classes. The value 0.5 shows equal distribution.

- Info gain - It is used to determine which feature / attribute guess us the maximum information about the Attribute.

Formula -

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

where Pi = probability of distinct class

Page No.

Algorithm

1) Read the input dataset
2) compare probabilities of all trends
3) Add all probability squares
4) subtract result in step ④ from ①
5) compute gini index for 'Weekend'
6) compute gini index for 'Weather'
7) similarly find gini Index for other attributes also.

Example -

Weekend	Weather	parents	Money	Decision
W1	sunny	yes	Rich	cinema
W2	sunny	No	Rich	Tennis
W3	windy	yes	Rich	cinema
W4	Rainy	yes	poor	cinema
W5	Rainy	No	Rich	stayIn
W6	Rainy	yes	poor	cinema
W7	windy	No	poor	cinema
W8	windy	No	Rich	shopping
W9	windy	yes	Rich	cinema
W10	sunny	No	Rich	Tennis

• For Gain -

- Finding Class Entropy

$$\text{Entropy}(S) = -\sum_{i=\{\text{Cinema, Shopping, Tennis, Staying}\}}^{4} Pi \log_2(Pi)$$

$$= -\left(\frac{6}{10}\right)\log_2\left(\frac{6}{10}\right) - \left(\frac{2}{10}\right)\log_2\left(\frac{2}{10}\right)$$

$$-\left(\frac{1}{10}\right)\log_2\left(\frac{1}{10}\right) - \left(\frac{1}{10}\right)\log_2\left(\frac{1}{10}\right)$$

$$= 0.4422 + 0.4644 + 0.3322 + 0.3322$$

$$= 1.571$$

Now we need to find best of :

$$\text{Gain}(S, \text{weather}) = 1.571 - (0.3)\times(0.918) - (0.4) * (0.81125) - (0.3)\times(0.918)$$

$$= 0.70$$

$$\text{Gain}(S, \text{parents}) = 1.571 - (0.5)\times 0 - (0.5)\times(1.922)$$

$$= 1.571 - 0.961$$

$$= 0.61$$

$$\text{Gain}(S, \text{Money}) = 1.571 - (0.7)\times(1.842) - (0.3)\times 0$$

$$= 1.571 - 1.2894$$

$$= 0.2816$$

Attribute	Gain
Weather	0.70
parents	0.61
Money	0.2816

From above we observe that Weather has highest gain and hence it will choosen as root node in decision tree

* Calculating Gini Index for 'Decision' Attribute

Cinema → 6

Tennis → 2

Stay In → 1

Shopping → 1

$\therefore$ Gini (S) $= 1 - \left[\left(\frac{6}{10}\right)^2 + \left(\frac{2}{10}\right)^2 + \left(\frac{1}{10}\right)^2 + \left(\frac{1}{10}\right)^2\right]$

$= 1 - 0.42$

$= 0.58$

* Calculating gini Index for Money Attribute

Rich (+) → 7

Poor (-) → 3

- For Money = poor, these are 3 examples with "cinema"

Gain (POOR) $= 1 - \left[\left(\frac{3}{3}\right)^2\right] = 0$

- For Money = Rich,

Tennis → 2, cinema → 3, StayIn → 1, Shopping → 1

Gain (RICH) $= 1 - \left[\left(\frac{2}{7}\right)^2 + \left(\frac{3}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2\right]$

$= 0.694$

. Weighted Average (Money)

$= 0 \times \frac{3}{10} + 0.694 \times \left(\frac{7}{10}\right)$

$= 0.486$

* calculating gini Index for 'parents'

yes (+) → 5

NO (-) → 5

- For parents = yes, there are 5 examples all with "cinema"

Gain (yes) = $1 - \left[\left(\frac{5}{5}\right)^2\right] = 0$

- For parents = NO,

Tennis → 2

StayIn → 1

Cinema → 1

Shopping → 1

Gain (NO) = $1 - \left[\left(\frac{2}{5}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{5}\right)^2\right]$

$= 0.72$

* weighted Average (parents)

$= 0 \times \frac{5}{10} + 0.72 \times \frac{5}{10}$

$= 0.36$

* calculating gini Index for 'weather'

Sunny → 3

Windy → 4

Rainy → 3

- For weather = Sunny

cinema → 1

Tennis → 2

Gain (S) = $1 - \left[\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right] = 0.444$

- For Weather = Rainy
  Cinema → 2
  Stay In → 1

$$Gain(R) = 1 - \left[\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right] = 0.444$$

- For Weather = Windy
  Cinema → 3
  Shopping → 1

$$Gain(Windy) = 1 - \left[\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right] = 0.375$$

- Weighted Average (Weather)

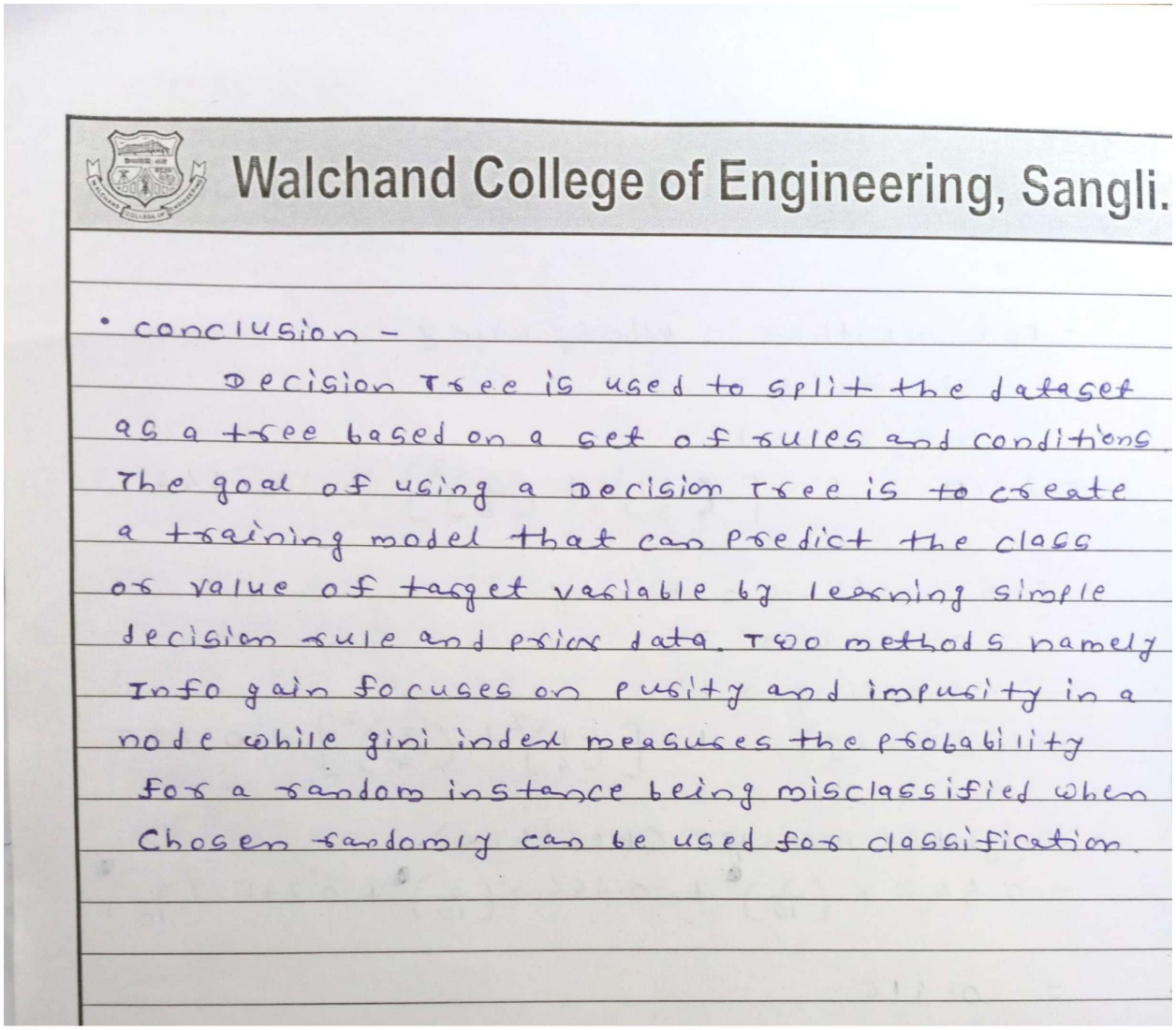
$$= 0.444 \times \left(\frac{3}{10}\right) + 0.444 \times \left(\frac{3}{10}\right) + 0.375 \times \left(\frac{4}{10}\right)$$

$$= 0.416$$

Now,

Attribute	Gini Index
Weather	0.416
parents	0.36
Money	0.486

From above, we observe that parents has lowest gini index and hence it will be chosen as root node for our decision tree



Program:-

(info gain)

```
#include<bits/stdc++.h>
using namespace std;

vector<string> sub_classes;
map<string,int> mainClass;
map<string,unordered_set<string>> dist_val;
map<string,int> dist_val_count;
```

```
map<string,map<string,int>> val_count;

double maxGain = DBL_MIN;
string root = "null";

ofstream fw("exp11_op_gain.csv",ios::out);

void calculateGain(string subClass,double mainC_gain){

   double totR = mainClass["Yes"] + mainClass["No"];

   double ent = 0;

   for(auto dv : dist_val[subClass]){
       double tR = dist_val_count[dv];
       double pR = val_count[dv]["Yes"],nR = val_count[dv]["No"];

       if(pR != 0)
           ent += - (tR/totR) * ((pR / tR) * log2(pR / tR));

       if(nR != 0)
           ent += - (tR/totR) * ((nR / tR) * log2(nR / tR));
   }

   cout << "InfoGain ( " << subClass << "|" << "playGame ) : " << ent << "\n";
   fw << "InfoGain ( " << subClass << "|" << "playGame )," << ent << "\n";

   double gain = mainC_gain - ent;

   cout << "Gain ( " << subClass << "|" << "playGame ) : " << gain << "\n\n";
   fw << "Gain ( " << subClass << "|" << "playGame )," << gain << "\n";

   if(gain > maxGain){
       maxGain = gain;
       root = subClass;
   }

}

int main(){

   fstream file("exp11_ip_gain.csv", ios::in);

   string line, word;
   string day, outlook, temp, humidity, wind, playGame;
```

```
if (!file.is_open())
{
    perror("Error in opening input file : ");
    return -1;
}

int j = 0;
string main_class = "playgame";

while (getline(file, line))
{
    stringstream str(line);

    getline(str, day, ',');
    getline(str, outlook, ',');
    getline(str, temp, ',');
    getline(str, humidity, ',');
    getline(str, wind, ',');
    getline(str, playGame, ',');

    if(j==0){
        j++;
        sub_classes.push_back(day);
        sub_classes.push_back(outlook);
        sub_classes.push_back(temp);
        sub_classes.push_back(humidity);
        sub_classes.push_back(wind);
        continue;
    }

    dist_val["day"].insert(day);
    dist_val["outlook"].insert(outlook);
    dist_val["temp"].insert(temp);
    dist_val["humidity"].insert(humidity);
    dist_val["wind"].insert(wind);

    mainClass[playGame]++;

    dist_val_count[day]++;
    dist_val_count[outlook]++;
    dist_val_count[temp]++;
    dist_val_count[humidity]++;
    dist_val_count[wind]++;
```

```
        val_count[day][playGame]++;
        val_count[outlook][playGame]++;
        val_count[temp][playGame]++;
        val_count[humidity][playGame]++;
        val_count[wind][playGame]++;
    }

    double posR = mainClass["Yes"],negR = mainClass["No"];
    double totR = posR + negR;

    double mainC_gain = -((posR / totR) * log2(posR / totR) + (negR / totR) *
log2(negR / totR));

    cout << "Main Class Gain : " << mainC_gain << "\n";

    for(int i=1;i<5;i++){
        calculateGain(sub_classes[i],mainC_gain);
    }

    cout << "Subclass : " << root << " has maximum gain . Hence it will be
selected as root for splitting.\n";
    fw << "Subclass : " << root << " has maximum gain . Hence it will be
selected as root for splitting.\n";

    return 0;
}
```

(Gini Index)

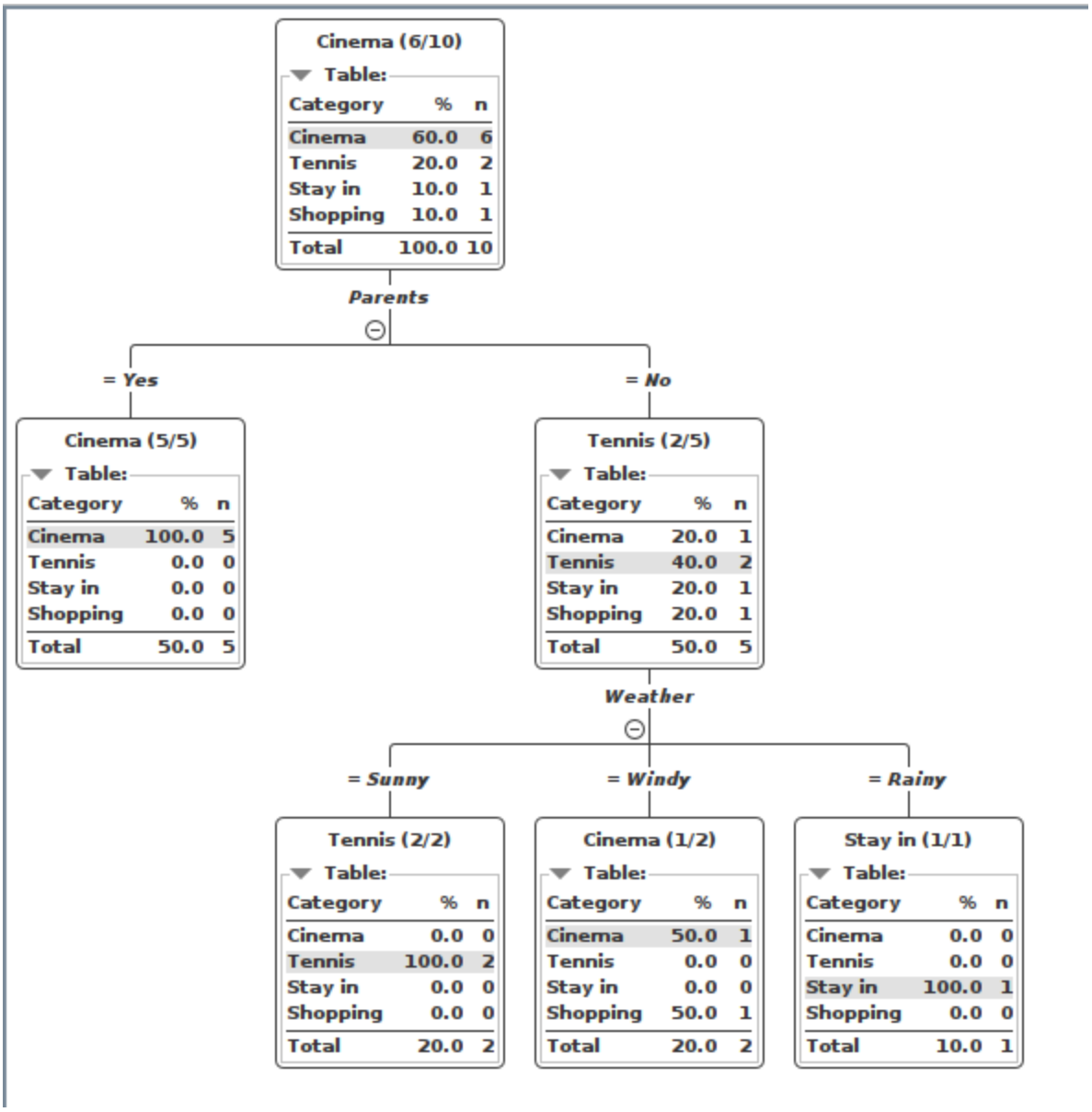
```
#include <bits/stdc++.h>
using namespace std;

double gini_of_class(double p1, double p2)
{
    int tot = p1 + p2;
    double tmp1 = (double)pow((p1 / tot), 2.0);
    double tmp2 = (double)pow((p2 / tot), 2.0);
    double ans = 1 - tmp1 - tmp2;
    return ans;
}

double gini_attribute(map<string, map<string, int>> attribute, double count)
{
```

```
    double gini = 0.0;
    for (auto i : attribute)
    {
        string val = i.first;
        double play_cnt = attribute[val]["Play"];
        double NoPlay_cnt = attribute[val]["NoPlay"];

        double tot = play_cnt + NoPlay_cnt;

        gini += (double)(tot / (double)count) * (1 - (play_cnt / tot) *
(play_cnt / tot) - (NoPlay_cnt / tot) * (NoPlay_cnt / tot));
    }
    return gini;
}


int main()
{

    ifstream file("gini_index.csv");

    string line, word;
    string outlook, temp, humidity, windy, mainclass;

    map<string, int> parent;
    map<string, map<string, int>> attribute;
    int count = 0;
    if (!file.is_open())
    {
        perror("Error in opening input file : ");
        return -1;
    }

    int i = 0;
    string attributeName, name;
    while (getline(file, line))
    {
        stringstream str(line);

        getline(str, outlook, ',');
        getline(str, temp, ',');
        getline(str, humidity, ',');
        getline(str, windy, ',');
        getline(str, mainclass, ',');
```

```
    int choice;

    if (i == 0)
    {
        i++;
        cout << "Enter Attribute Column Number : ";
        cin >> choice;
        cout << endl;
        continue;
    }

    switch (choice)
    {

    case 1:
        attributeName = outlook;
        break;

    case 2:
        attributeName = temp;
        break;

    case 3:
        attributeName = humidity;
        break;

    case 4:
        attributeName = windy;
        break;

        // case 5:
        //     attributeName = mainclass;
        //     break;

    default:
        attributeName = outlook;
        break;
    }

    parent[mainclass]++;
    attribute[attributeName][mainclass]++;
    count++;
}

int p1 = parent["Play"];
```

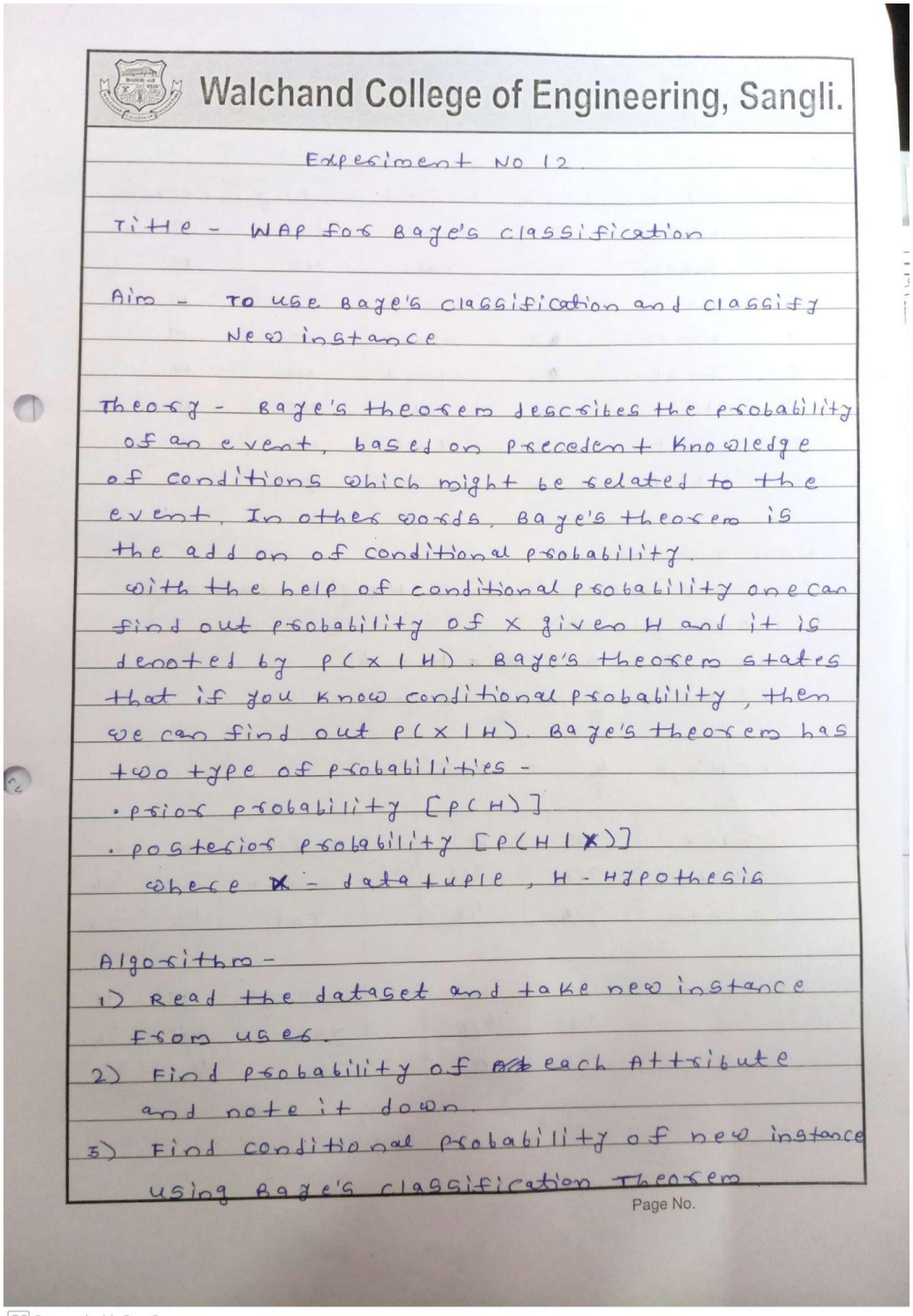
```
    int p2 = parent["NoPlay"];
    // cout << p1 << " " << p2 << endl;
    double gini_parent = gini_of_class(p1, p2);
    cout << "Gini Index "
         << "(column class):  " << gini_parent << endl;

    double gini = gini_attribute(attribute, count);
    cout << "Gini Index "
         << "(column " << i << ") : " << gini << endl;
}
```


```
sumit@sumit-15:~/Documents/7th Sem/DM Lab$ g++ gini_index.cpp
sumit@sumit-15:~/Documents/7th Sem/DM Lab$ ./a.out
Enter Attribute Column Number : 1

Gini Index (column class):  0.459184
Gini Index (column 1) : 0.342857
sumit@sumit-15:~/Documents/7th Sem/DM Lab$
```


Cinema (6/10)
Table:
Category % n
Cinema 60.0 6
Tennis 20.0 2
Stay in 10.0 1
Shopping 10.0 1
Total 100.0 10
Parents
= Yes
= No
Cinema (5/5)
Table:
Category % n
Cinema 100.0 5
Tennis 0.0 0
Stay in 0.0 0
Shopping 0.0 0
Total 50.0 5
Tennis (2/5)
Table:
Category % n
Cinema 20.0 1
Tennis 40.0 2
Stay in 20.0 1
Shopping 20.0 1
Total 50.0 5
Weather
= Sunny
= Windy
= Rainy
Tennis (2/2)
Table:
Category % n
Cinema 0.0 0
Tennis 100.0 2
Stay in 0.0 0
Shopping 0.0 0
Total 20.0 2
Cinema (1/2)
Table:
Category % n
Cinema 50.0 1
Tennis 0.0 0
Stay in 0.0 0
Shopping 50.0 1
Total 20.0 2
Stay in (1/1)
Table:
Category % n
Cinema 0.0 0
Tennis 0.0 0
Stay in 100.0 1
Shopping 0.0 0
Total 10.0 1

Experiment No 12

Title:- Bayes classification



Walchand College of Engineering, Sangli.

Experiment No 12

Title - WAP for Baye's classification

Aim - To use Baye's classification and classify New instance

Theory - Baye's theorem describes the probability of an event, based on precedent knowledge of conditions which might be related to the event. In other words, Baye's theorem is the add on of conditional probability.
With the help of conditional probability one can find out probability of x given H and it is denoted by $P(x|H)$. Baye's theorem states that if you know conditional probability, then we can find out $P(x|H)$. Baye's theorem has two type of probabilities -
- prior probability $[P(H)]$
- posterior probability $[P(H|X)]$

where X - data tuple, H - Hypothesis

Algorithm -
1) Read the dataset and take new instance from user.
2) Find probability of each Attribute and note it down.
3) Find conditional probability of new instance using Baye's classification Theorem.

Page No.

Example -

No	Color	Legs	Height	smelly	species
1	White	3	short	yes	M
2	Green	2	Tall	NO	M
3	Green	3	short	yes	M
4	White	3	short	yes	M
5	Green	2	short	NO	H
6	White	2	Tall	NO	H
7	White	2	Tall	NO	H
8	White	2	short	yes	H

New Instance -

(Color = Green, legs = 2, Height = Tall and smelly = No)

Here, M = 4, H = 4

$P(M) = \frac{4}{8} = \frac{1}{2} = 0.5$

$P(H) = \frac{4}{8} = 0.5$

Color	M	H
White	2/4	2/4
Green	2/4	1/4

Height	M	H
short	3/4	2/4
Tall	1/4	2/4

Legs	M	H
2	1/4	4/4
3	3/4	0

smelly	M	H
yes	3/4	1/4
NO	1/4	3/4

Now,

P(M | New Instance) = P(M) * P(color = Green | M)
* P(Legs = 2 | M) * P(Height = Tall | M)
* P(Smelly = No | M)

$P(M \mid new\_Instance) = 0.5 \times \frac{2}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4}$

$= 0.0039$

P(H | new_Instance) = P(H) * P(color = Green | H)
* P(Legs = 2 | H) * P(Height = Tall | H)
* P(Smelly = No | H)

$= 0.5 \times \frac{1}{4} \times \frac{4}{4} \times \frac{2}{4} \times \frac{3}{4}$

$= 0.04687$

Here,

P(H | new_instance) > P(M | new_instance)

∴ New instance belong to species 'H'

• conclusion -

Naive Bayes classification algorithm is a probabilistic classifier. It is useful for making predications and forecasting data based on historical results. By using bayesian classifier we can classify unknown (New instance) case by training over unknown data. It Helps to specify the class of New instance to which it will belongs to.

Program:-

```
#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("exp12_input.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;
}
```

File

Class counts for Species

Class:	H	M
Count:	4	4

Total count: 8

Threshold to used for zero probabilities: 1.0E-4

Class/Color	Green	White
H	1	3
M	2	2
Rate:	38%	62%

P(Height | class=?)

Class/Height	Short	Tall
H	2	2
M	3	1
Rate:	62%	38%

Gaussian distribution for Legs per class value

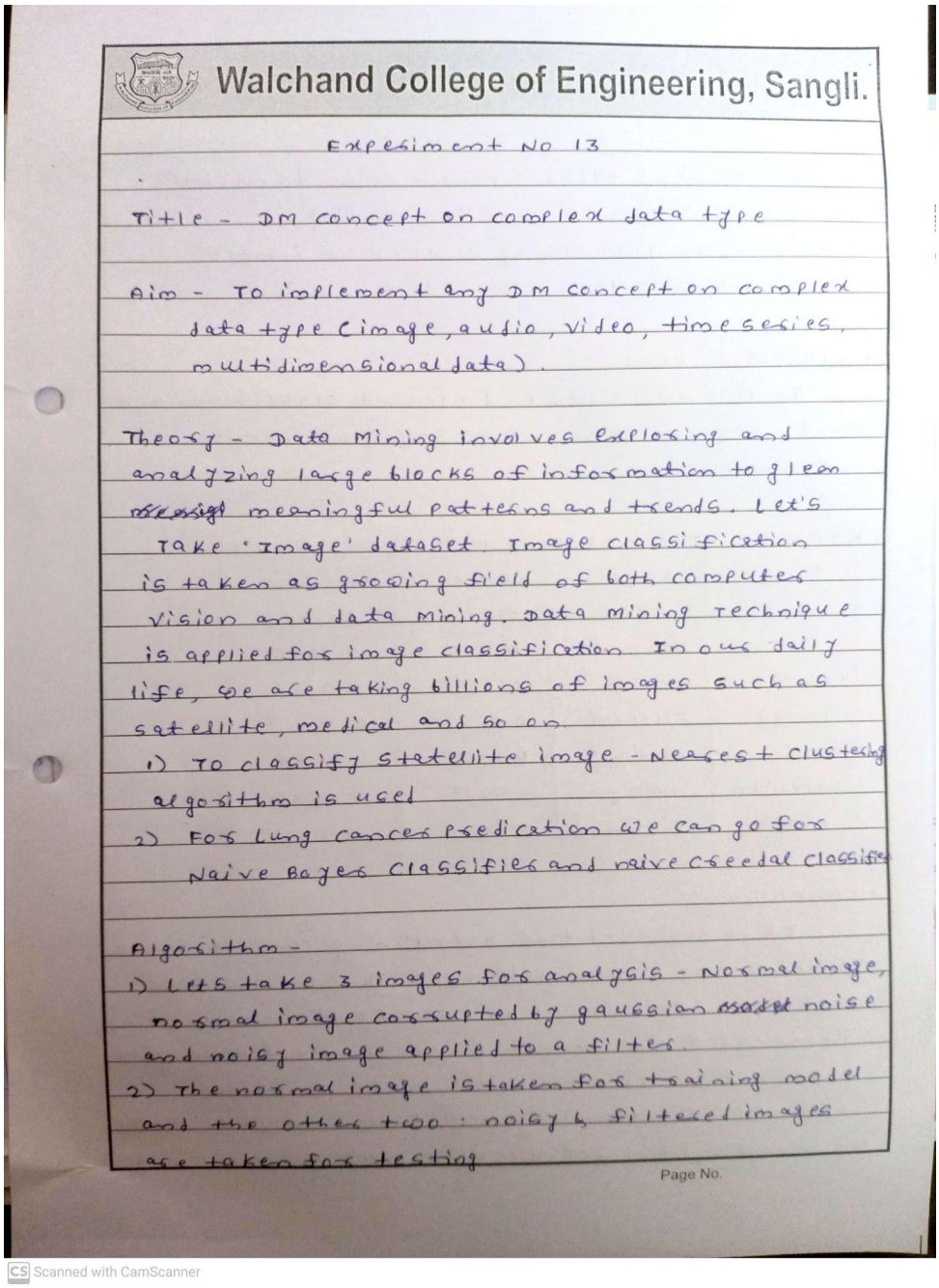
	H	M
Count:	4	4
Mean:	2	2.75
Std. Deviation:	0.0001	0.5
Rate:	50%	50%

P(Smelly | class=?)

Class/Smelly	No	Yes
H	3	1
M	1	3
Rate:	50%	50%

Experiment No 13

Title:- To implement any DM concept on complex data type (image, audio, video, time series, spatial, multidimensional data)



Walchand College of Engineering, Sangli.

Experiment No 13

Title - DM concept on complex data type

Aim - To implement any DM concept on complex data type (image, audio, video, time series, multidimensional data).

Theory - Data Mining involves exploring and analyzing large blocks of information to glean meaningful patterns and trends. Let's take 'Image' dataset. Image classification is taken as growing field of both computer vision and data mining. Data mining technique is applied for image classification. In our daily life, we are taking billions of images such as satellite, medical and so on.

1) To classify statellite image - Nearest clustering algorithm is used
2) For Lung cancer predication we can go for Naive Bayes classifier and naive creedal classifier

Algorithm -

1) Lets take 3 images for analysis - Normal image, normal image corrupted by gaussian noise and noisy image applied to a filter.
2) The normal image is taken for training model and the other two : noisy & filtered images are taken for testing

Page No.

3) Now we have added a random noise to normal image. This will be used for testing
4) Now Apply data mining concept on images like smoothing to reduce adaptive noise.

- classifiers used -

In this experiment, different classifiers are used - Decision Tree, Naive Bayes and Random Forest.

- Result - Table of Root Mean Square Error

Images	Image Type	Naive Bayes	Random Forest
cancer Image	Normal	0.216	0.0258
	Noisy	0.4351	0.2918
	Filtered	0.4282	0.2827
satellite Image	Normal	0.1205	0.0099
	Noisy	0.5229	0.3539
	Filtered	0.5073	0.3371

- Conclusion -

It is observed that Data Mining concept can be applied on any compled data type. These classifiers technique are applied to classify the region of interest from images in order to get meaningful observation. The filtered image enhances the classification accuracy. Random Forest is better for normal and filtered while Naive Bayes for Noisy image.