"Data Mining Laboratory"

Code: 5IT451

Submitted by

Mr. Abhijeet Ashok Patil

Email: abhijeet.patil@walchandsangli.ac.in

Mobile:8483911529

DEPARTMENT OF INFORMATION TECHNOLOGY

WALCHAND COLLEGE OF ENGINEERING, SANGLI

(An Autonomous Institute)

2023-2024

CERTIFICATE

This is to certify that the report entitled **"Data Mining Laboratory 5IT451***"* submitted by **MR. ABHIJEET ASHOK PATIL (2020BTEIT00077)** is a record of a student's own work carried out by him during the academic year 2023-2024, as per the curriculum/syllabus laid down for Data Mining lab at Final year B.Tech IT Sem-I. He has carried out experiments successfully.

Dr. R.R.Rathod

(Course Teacher)

Declaration

I, the undersigned, hereby declare that the BTech report entitled “Data Mining Laboratory 5IT451” submitted by me to the Data Mining Lab report at Final year BTech IT Sem-I, is my original/experimented/experience work. I further declare that, to the beat of my knowledge and belief, this report has not been previously submitted or copied by me.

I declare that this report reflects my thoughts about the subject in my own words. I have sufficiently cited and referenced the original sources, referred, or considered in this work. I have not misinterpreted, fabricated, or falsified any idea/data/fact/source in this my submission. I understand that any violation of the above will be cause for disciplinary action by the course teacher/institute.

Date: **Mr. Abhijeet Patil**

Place:

Acknowledgement

I feel immense pleasure in submitting the report entitled “Data Mining Laboratory 5IT451”. I am thankful to our guide Dr. R. R. Rathod for their valuable guidance and kind help during implementing the Data Mining Lab.

Acknowledged By,

Mr. Abhijeet Patil

Data Mining Lab Book

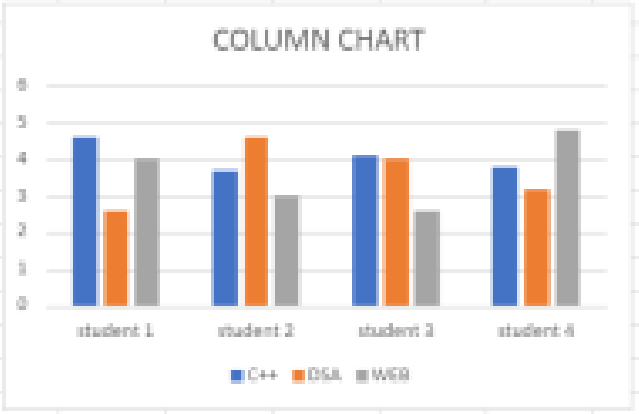
Name:- Abhijeet Ashok Patil
PRN:- 2020BTEIT00077
Class: Final Year-IT-Sem I (2023-2024)

Experiment No. 1

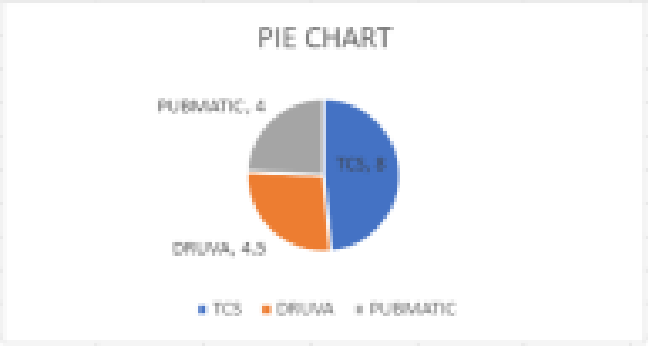
Title :- Study and use of different types of graphs and charts (use MS-XLS). **Aim :-**

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

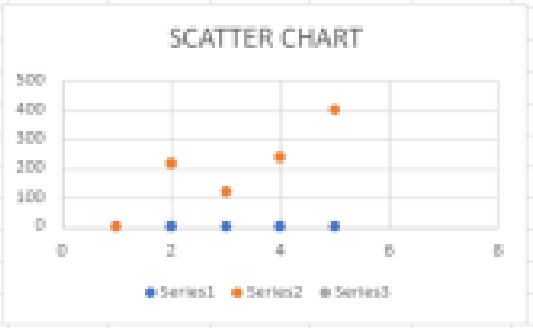
SKILL RATINGS OF STUDENTS			
	C++	DSA	WEB
student 1	4.6	2.6	4
student 2	3.7	4.6	3
student 3	4.1	4	2.6
student 4	3.8	3.2	4.8

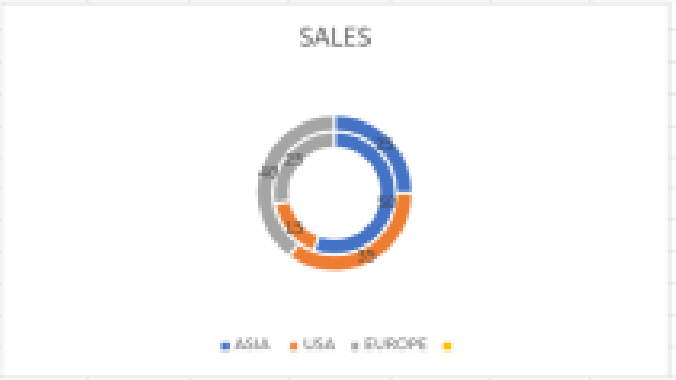


REVENUE IN BILLION OF IT COMPANIES	
TCS	8
DRUVA	4.5
PUBMATIC	4



DISTANCE OF CITIES FROM PUNE	
CITY	DISTANCE
SANGLI	320
SATARA	120
KOLHAPUR	240
MUMBAI	400





SALES BY REGION		
	2022	2023
ASIA	25	50
USA	35	15
EUROPE	40	25

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

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

Column chart: A column chart, also known as a vertical bar chart, is a graphical representation used to display data points using vertical bars. It is a popular chart type for comparing and contrasting categorical data or discrete values. Column charts are widely used in various fields to visualize and communicate data patterns, distributions, and comparisons.

Pie chart: A pie chart is a circular graphical representation used to display the distribution of data as a "pie" divided into slices, with each slice representing a proportion of the whole. Pie charts are commonly used to show how individual parts contribute to a whole and are effective for visualizing percentages or relative proportions of different categories within a dataset.

Scatter chart: A scatter chart, also known as a scatter plot, is a graphical representation used to display the relationship between two variables or sets of data. It uses a collection of individual data points, each representing a specific value for both the X-axis and the Y-axis. Scatter charts are particularly useful for identifying patterns, correlations, and outliers within data sets.

Donut Chart: A donut chart typically shows the proportions of categorical data where the size of each piece of the donut communicates the proportion of each category.

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.

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.

Experiment No. 2

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

Data Normalisation:- The data normalisation (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 normalisation is to minimise 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.

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.

Name - Abhijeet Ashok Patil
2020BTEIT00077
Data Mining Lab.

Walchand College Of Engineering, Sangli.

Title:- Perform Normalization of data.

Theory / Algo:-

Data Normalization is a technique used in data mining to transform the values of dataset into common scale.

Types -:

1] min-max → scales the value of feature to range 0 & 1. This is done by subtracting minimum value of feature from each value, then dividing by range of feature.

2] Z-score → scales values of feature to have mean of 0 & standard deviation of 1. Done by subtracting mean of feature from each value & dividing by standard deviation.

Algo for min-max:-

Step 1:- for Dataset find min & max value.

step 2:- for each datapoint apply

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \quad \text{where, } X \rightarrow \text{original data pt.}$$

X_{min} - min. value
X_{max} - max value

$X (X_{newMax} - X_{newMin}) + X_{newMin}$

Step 3: Repeat steps 1, 2.

Update

Walchand College Of Engineering, Sangli.

Algo For
Z-score Normalization →

Step 1:- For each dataset calculate mean & standard Deviation.

Step 2:- For each Datapoint Apply

$$X = \frac{X - X_{mean}}{X_{std}}$$ X_{std} → Standard deviation.

OR $\left(\frac{x-\mu}{\sigma}\right)$.

Step 3:- Repeat step 1,2.

Ex. for Min-Max

marks
8
10
15
20.

Min → 8
Max → 20

① For 8 → $\frac{(8-8)}{20-8} \times (1-0) + 0$
$= 0$

② For 10 → $\frac{(10-8)}{20-8} \times (1-0) + 0$
$= 0.16$

③ For 15 → $\frac{(15-8)}{20-8} \times (1-0) + 0$
$= 0.58$

④ for 20 → $\frac{(20-8)}{20-8} \times (1-0) + 0$
$= 1$

MinMax →
0
0.16
0.58
1

Ex. z-score

$$s = \sqrt{\frac{\Sigma (x-\bar{x})^2}{n}}$$

$$\text{mean} = \frac{8+10+15+20}{4} = 13.25$$

$$= \sqrt{\frac{(8-13.25)^2+(10-13.25)^2+(15-13.25)^2+(20-13.25)^2}{4}}$$

$$= \sqrt{\frac{(-5.25)^2+(-3.25)^2+(1.75)^2+(6.75)^2}{4}}$$

$$= \sqrt{\frac{86.74}{4}} = \sqrt{21.6} = 4.6$$

$$z\text{score} = \frac{x-u}{\sigma} = \frac{8-13.25}{4.6} = -1.14$$

$$\frac{10-13.25}{4.6} = -0.7$$

$$\frac{15-13.25}{4.6} = 0.3$$

$$\frac{20-13.25}{4.6} = 1.4$$

z score →

−1.14
−0.7
0.3
1.4

Applications:-

min-max →

1] Image Processing → pixel values are normalized using min-max scaling.

2] Recommendation system :-
Normalize user ratings to common range for better comparison.

z-score

1] Statistical Analysis → to standardize data

2] clustering Algo like k-means because it centers data around zero

conclusion:-

min-max is suitable when we have to confine our data to specific range like 0 & 1.

z-score is basically preferred in statistical & machine learning scenario.

Code:

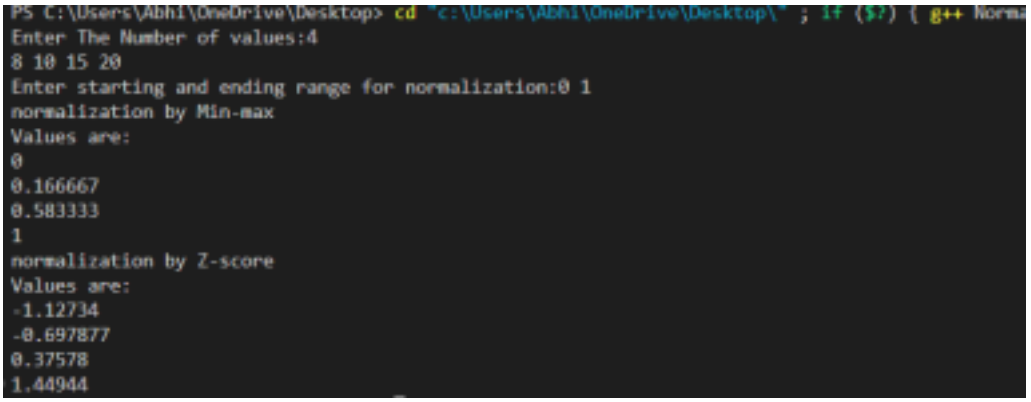
```
#include <iostream>
#include <math.h>
using namespace std;
int main()


{
 int n;
 double sum = 0;
 cout << "Enter The Number of values:";
 cin >> n;
 int a[n], r1, r2;
 for (int i = 0; i < n; i++)
 {


 cin >> a[i];
 }
 cout << "Enter starting and ending range for normalization:";
cin >> r1 >> r2;
 double max = a[0];
 int min = a[0];
 for (int i = 0; i < n; i++)


 {
 if (a[i] > max)


 {


 max = a[i];
```

```
}

else if (a[i] < min)

{

min = a[i];
}
}
float v[n];
double temp;

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

{

  temp = double(double(double(a[i] - min) * (r2 - r1)) / (max - min)) + r1;

v[i] = temp;
}

cout << "normalisation by Min-max" << endl;
cout << "Values are:" << endl;

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

```
{


cout << v[i] << endl;

}
cout << "normalization by Z-score" << endl;
double mean;


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


{


sum = sum + a[i];

}


mean = (sum / n);

double temp1 = 0, std_dev1;


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


{


temp1 = temp1 + pow(a[i] - mean, 2);  }


std_dev1 = sqrt(temp1 / n);
```

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

temp = double(a[i] - mean) / std_dev1;

v[i] = temp;
}

cout << "Values are:" << endl;

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

{

cout << v[i] << endl;
} return 0;
}
```

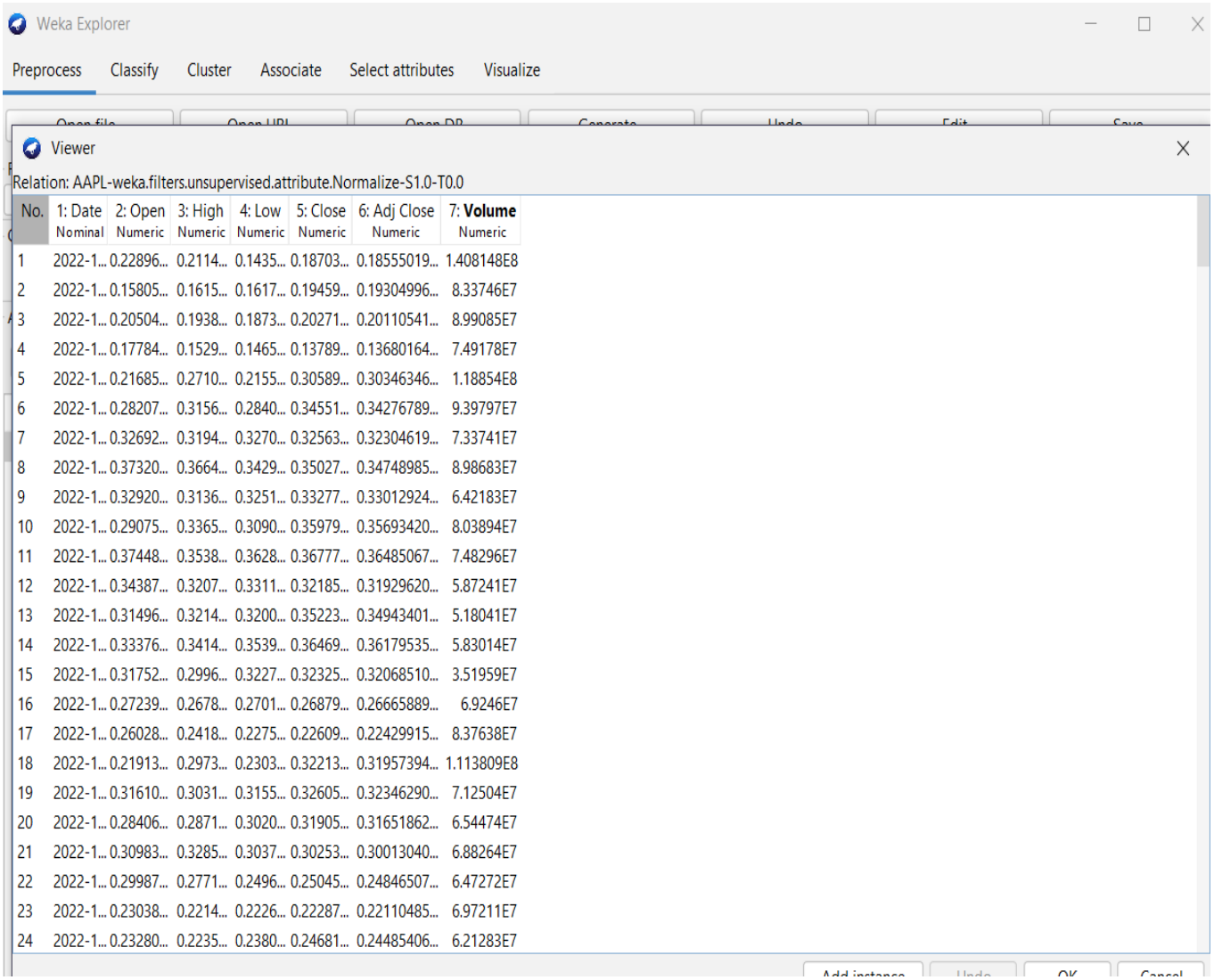
output:



WEKA INPUT:APPLE STOCK DATA 4 NOV 2023

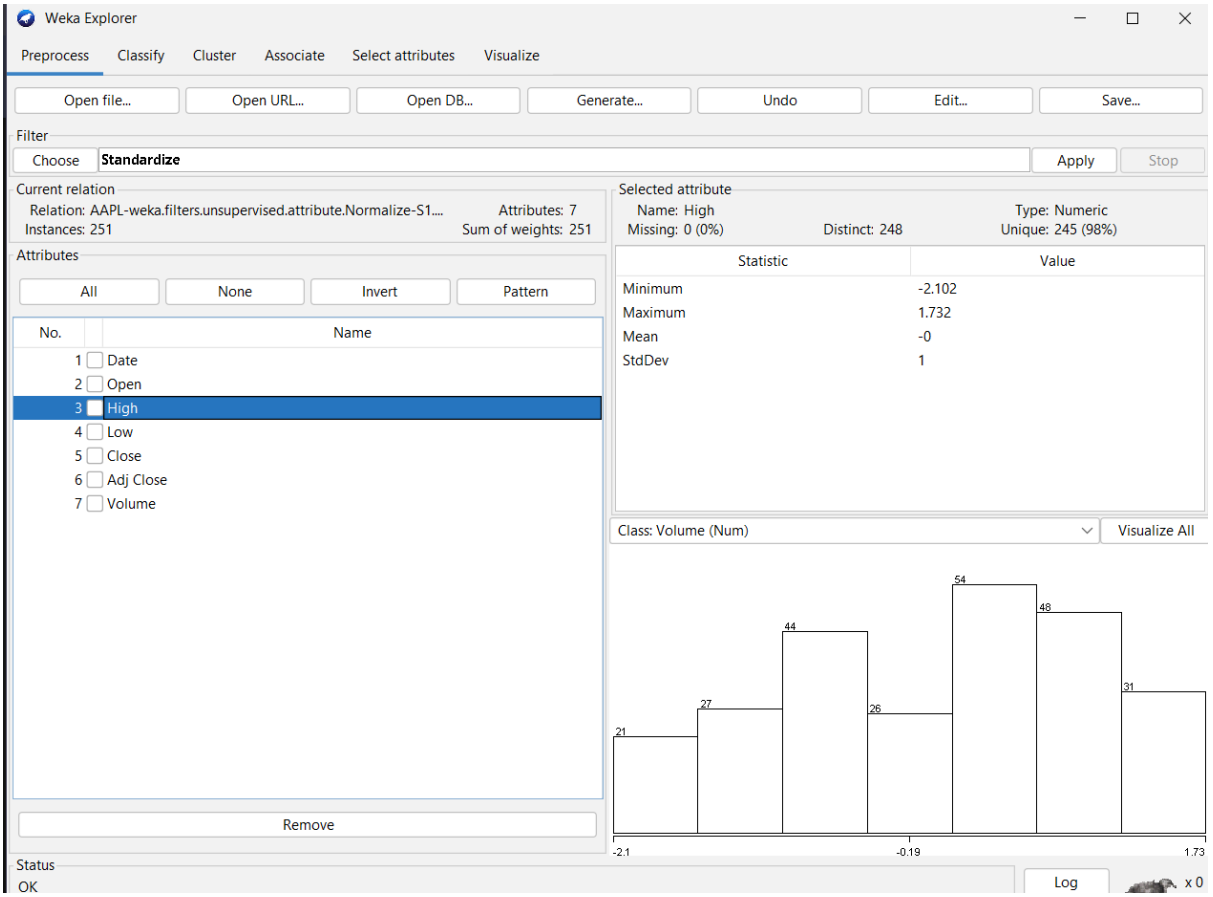
WEKA OUTPUT:

MIN-MAX



No.	1: Date Nominal	2: Open Numeric	3: High Numeric	4: Low Numeric	5: Close Numeric	6: Adj Close Numeric	7: **Volume** Numeric
1	2022-1...	0.22896...	0.2114...	0.1435...	0.18703...	0.18555019...	1.408148E8
2	2022-1...	0.15805...	0.1615...	0.1617...	0.19459...	0.19304996...	8.33746E7
3	2022-1...	0.20504...	0.1938...	0.1873...	0.20271...	0.20110541...	8.99085E7
4	2022-1...	0.17784...	0.1529...	0.1465...	0.13789...	0.13680164...	7.49178E7
5	2022-1...	0.21685...	0.2710...	0.2155...	0.30589...	0.30346346...	1.18854E8
6	2022-1...	0.28207...	0.3156...	0.2840...	0.34551...	0.34276789...	9.39797E7
7	2022-1...	0.32692...	0.3194...	0.3270...	0.32563...	0.32304619...	7.33741E7
8	2022-1...	0.37320...	0.3664...	0.3429...	0.35027...	0.34748985...	8.98683E7
9	2022-1...	0.32920...	0.3136...	0.3251...	0.33277...	0.33012924...	6.42183E7
10	2022-1...	0.29075...	0.3365...	0.3090...	0.35979...	0.35693420...	8.03894E7
11	2022-1...	0.37448...	0.3538...	0.3628...	0.36777...	0.36485067...	7.48296E7
12	2022-1...	0.34387...	0.3207...	0.3311...	0.32185...	0.31929620...	5.87241E7
13	2022-1...	0.31496...	0.3214...	0.3200...	0.35223...	0.34943401...	5.18041E7
14	2022-1...	0.33376...	0.3414...	0.3539...	0.36469...	0.36179535...	5.83014E7
15	2022-1...	0.31752...	0.2996...	0.3227...	0.32325...	0.32068510...	3.51959E7
16	2022-1...	0.27239...	0.2678...	0.2701...	0.26879...	0.26665889...	6.9246E7
17	2022-1...	0.26028...	0.2418...	0.2275...	0.22609...	0.22429915...	8.37638E7
18	2022-1...	0.21913...	0.2973...	0.2303...	0.32213...	0.31957394...	1.113809E8
19	2022-1...	0.31610...	0.3031...	0.3155...	0.32605...	0.32346290...	7.12504E7
20	2022-1...	0.28406...	0.2871...	0.3020...	0.31905...	0.31651862...	6.54474E7
21	2022-1...	0.30983...	0.3285...	0.3037...	0.30253...	0.30013040...	6.88264E7
22	2022-1...	0.29987...	0.2771...	0.2496...	0.25045...	0.24846507...	6.47272E7
23	2022-1...	0.23038...	0.2214...	0.2226...	0.22287...	0.22110485...	6.97211E7
24	2022-1...	0.23280...	0.2235...	0.2380...	0.24681...	0.24485406...	6.21283E7

Z SCORE OUTPUT:Z SCORE


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
Standardize
Apply
Stop
Current relation
Relation: AAPL-weka.filters.unsupervised.attribute.Normalize-S1....
Attributes: 7
Instances: 251
Sum of weights: 251
Selected attribute
Name: High
Type: Numeric
Missing: 0 (0%)
Distinct: 248
Unique: 245 (98%)
Attributes
All
None
Invert
Pattern
No.
Name
1 Date
2 Open
3 High
4 Low
5 Close
6 Adj Close
7 Volume
Statistic
Value
Minimum
-2.102
Maximum
1.732
Mean
-0
StdDev
1
Class: Volume (Num)
Visualize All
54
48
44
31
27
26
21
-2.1
-0.19
1.73
Remove
Status
OK
Log
x 0

Experiment No. 3

Title:- Perform Binning of data .

Theory:-

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin. This has a smoothing effect on the input data and may also reduce the chances of overfitting in the case of small datasets

There are 2 methods of dividing data into bins:

Equal Frequency Binning: bins have an equal frequency.

Equal Width Binning : bins have equal width with a range of each bin are defined as [min + w], [min + 2w] …. [min + nw] where w = (max – min) / (no of bins).

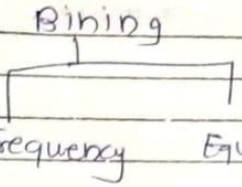
Name:- Abhijeet Ashok Patil
2020BTEIT00077
Data Mining Lab

Walchand College Of Engineering, Sangli.

TITLE:- Perform Bining of data.

Theory/Algo →

Data Bining is pre-processing method.
original data values are divided into small intervals known as bins.



1] Equal width Divides data into Equal sized intervals.

2] Equal frequency divides values into bins having equal no. of observations.

Ex. Equal width.

Algo dataset → 10, 15, 18, 20, 21, 34, 41, 46, 51, 53, 54

Step 1:- Sort dataset ascending order

10, 13, 18, 20, 31, 34, 41, 46, 51, 53, 54.

Step 2: determine $\left(\text{width} = \frac{\max - \min}{N}\right)$ N → No. of bins, W = width.

min → 10
max → 54 $W = \frac{54-10}{4} = 11$

BIN 1 → [min, (min+w-1)] → [10, 20]
BIN 2 → [min+w, (min+2w-1)] → [21, 31]
BIN 3 → [min+2w, (min+3w-1)] → [32, 42]
BIN 4 → [min+3w, max] → [43, 54]

BIN 1 :- [10,15,18,20]
BIN2 :- [31]
BIN3 :- [34,41]
BIN4 :- [46,51,53,54]

2) Equal freq :-
Dataset → 10,15,18,20,31,34,41,46,51,53,54,60

Step 1 → Sort dataset
Step 2 → Calc. frequency

freq → Total No. of data points / no. of bins

No of Datapoints → 12
Bins → 3
freq → 4

BIN 1 → 10,15,18,20
BIN 2 → 31,34,41,46
BIN3 → 51,53,54,60

Application - used to transform continuos data into discrete categories.
used to simplify data Analysis, visualization, pattern recognization.

conclusion - Data Bining Improves interpretability, enables us to recognize patterns, trends.

Code:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
// Equal Width Binning
vector<vector<double>> equalWidthBinning(const vector<double>& data, int numBins) {
 double min = *min_element(data.begin(), data.end());
 double max = *max_element(data.begin(), data.end());
 double width = (max - min) / numBins;
 vector<vector<double>> bins(numBins);

 for (const double& val : data) {
 int bin = (val - min) / width;
 if (bin == numBins) {
 bin--;
 }
 bins[bin].push_back(val);
 }

 return bins;
}

int main() {
 vector<double> data = {12.5, 7.2, 15.3, 8.7, 20.1, 10.0, 11.8, 9.4, 13.6, 6.8};
 int numBins = 3;

 vector<vector<double>> widthBins = equalWidthBinning(data, numBins);
cout << "Equal Width Bins:" << endl;
```

```
 for (int i = 0; i < numBins; i++) {
 cout << "Bin " << i + 1 << ": ";
 for (const double& val : widthBins[i]) {
 cout << val << " ";
 }
 cout << endl;
 }


 return 0;
}
```

OUTPUT:

```
Equal Width Bins:
Bin 1: 7.2 8.7 10 9.4 6.8
Bin 2: 12.5 15.3 11.8 13.6
Bin 3: 20.1
```

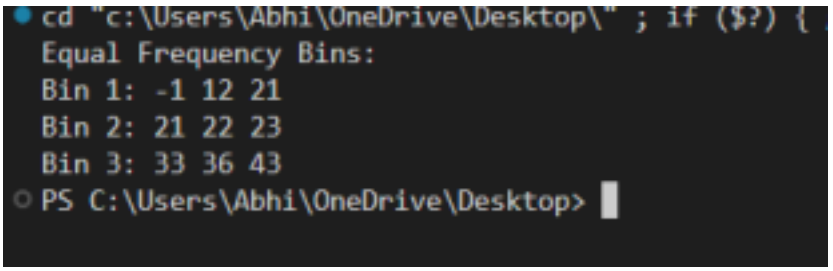
Equal frequency binning:

```
#include <iostream>
#include <vector>
#include <algorithm>


using namespace std;


// Equal Frequency Binning
vector<vector<double>> equalFrequencyBinning(const vector<double>& data,
int numBins) {
 vector<double> sortedData = data;
 sort(sortedData.begin(), sortedData.end());
```

```
 int pointsPerBin = data.size() / numBins;


 vector<vector<double>> bins(numBins);
 int dataIndex = 0;


 for (int binIndex = 0; binIndex < numBins; binIndex++) {  for (int i =
0; i < pointsPerBin && dataIndex < data.size(); i++) {
bins[binIndex].push_back(sortedData[dataIndex]);  dataIndex++;
 }
 }


 return bins;
}


int main() {
 // Input data (you can modify this as needed)
 vector<double> data = {12.5, 7.2, 15.3, 8.7, 20.1, 10.0, 11.8, 9.4, 13.6,
6.8};
 int numBins = 3;


 vector<vector<double>> frequencyBins = equalFrequencyBinning(data, numBins);
cout << "Equal Frequency Bins:" << endl;


 for (int i = 0; i < numBins; i++) {
 cout << "Bin " << i + 1 << ": ";
 for (const double& val : frequencyBins[i]) {
 cout << val << " ";
 }
```

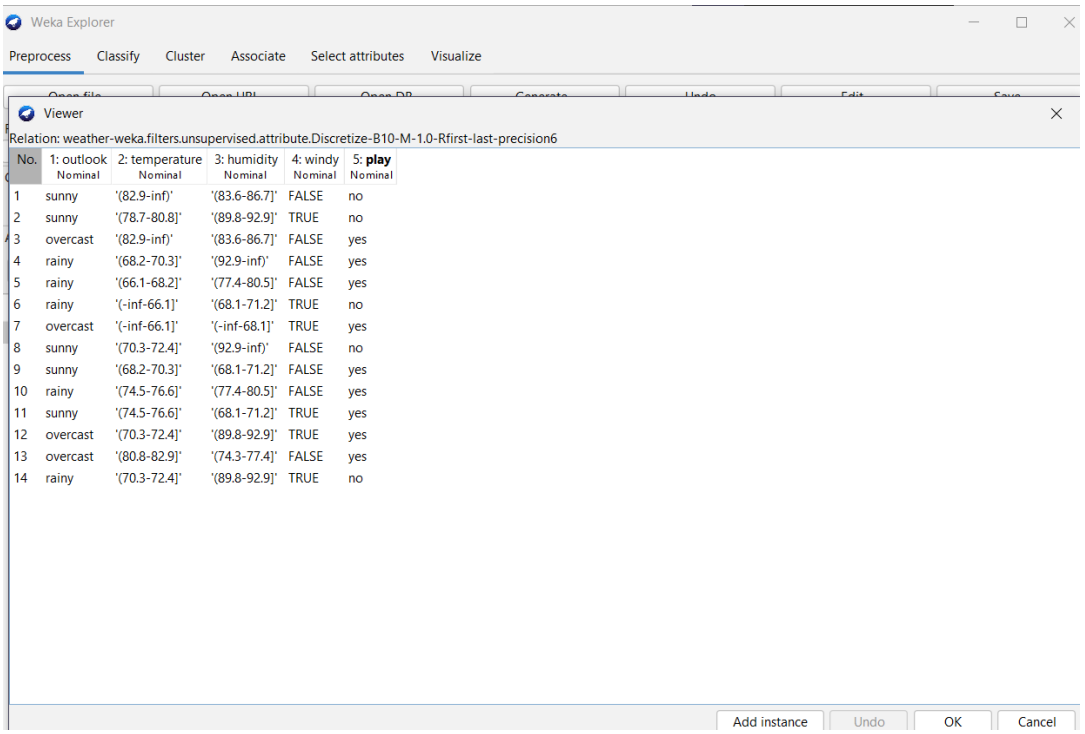
```
 cout << endl;
 }


 return 0;
}
```

OUTPUT:



Input data:BINNING DATA

OUTPUT DATA:



Experiment No. 4

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

Theory:-

Entropy - Entropy is uncertainty/ randomness in the data, the more the randomness the higher will be the entropy. Information gain uses entropy to make decisions. If the entropy is less, information will be more. Information gain is used in decision trees and random forest to decide the best split.

Information Gain - information gain is the amount of information that's gained by knowing the value of the attribute, which is the entropy of the distribution before the split minus the entropy of the distribution after it. The largest information gain is equivalent to the smallest entropy.

Walchand College Of Engineering, Sangli.

Experiment No-4

Patil Aliyeet Ashok
2020BTEIT00077

Aim- To find Gain of Attribute from given Data.

Theory -:

(1) Entropy - It measures Impurity or uncertainty in a group of observations.
It determines how decision tree choose to split data.

(2) Info-gain - It measures the expected Reduction in Entropy caused by partition example according to attribute.

Entropy → $E = -\sum_{i=1}^{N} P \log_2 P$

N → No. of Dataset
P → Probability of randomly selecting an example.

Info gain →

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

Algorithm :-

1) Read Data from .csv file
2) Define function to calculate Entropy.
3) Define function to calculate gain.
4) Show Result

Example :- Inputted csv file.

1] Entropy of class (parent)

True → 9
False → 5

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

$$= 0.9402$$

2] consider Attribute Routine.

values (Routine) = Sunny, Overcast, Rainy.

$S_{Sunny} \rightarrow [2^+, 3^-]$

$S_{Overcast} \rightarrow [4^+, 0^-]$

$S_{Rainy} \rightarrow [3^+, 2^-]$

$$Gain(S, Routine) = Entropy(S) - \frac{5}{14}Entropy(Sunny) - \frac{4}{14}Entropy(Overcast) - \frac{5}{14}Entropy(Rainy)$$

$$= 0.9402 - \frac{5}{14}\left[-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right]$$

$$- \frac{4}{14}\left[-\frac{4}{4}\log_2\left(\frac{4}{4}\right) - \frac{0}{4}\log_2\left(\frac{0}{4}\right)\right]$$

$$- \frac{5}{14}\left[-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right]$$

$$= 0.9402 - \frac{5}{14}(0.971) - \frac{4}{14} \times (1)$$

$$= 0.0482$$

Routine has higher gain making it better splitting Attribute.

code:

```
#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);
```

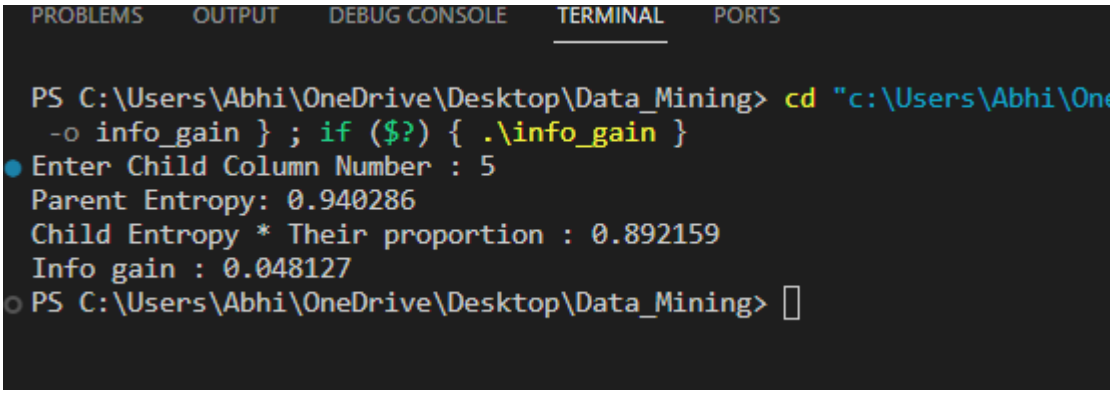
```
getline(str, day, ',');

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;
```

}

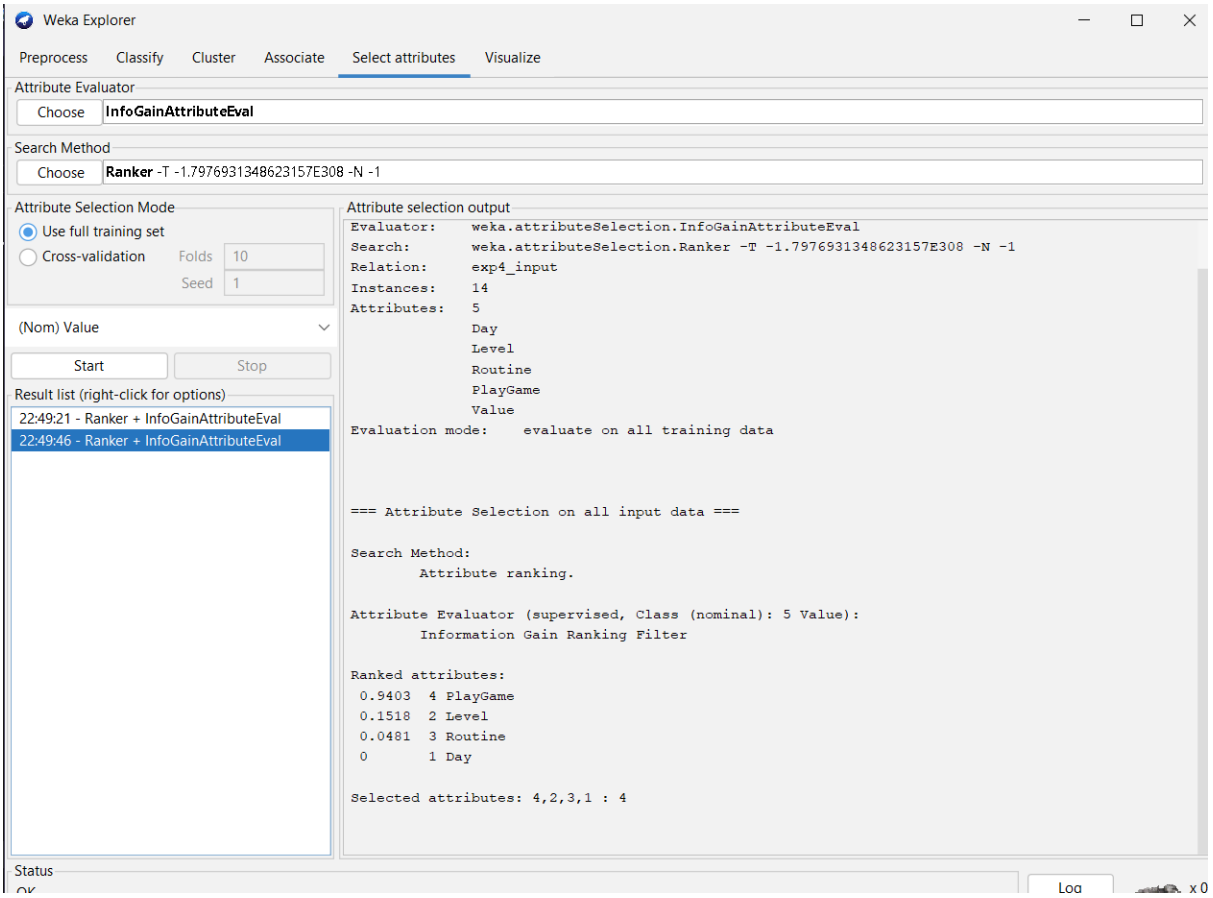
output:



WEKA :

Input data:INFO_GAIN

OUTPUT :



Experiment No. 5

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 generalised 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 generalised 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.

Experiment NO. 5

Title:- Find t & d weight of data.

Aim:- To find t & d weight

Theory:-

T-weight → Ratio of tuples covered by Ta in Target class verses all type of tuples in initial Target class.

d-weight → Ratio of tuples da in target class verses Total number of tuples in both initial & contrasting classes covered by Da.

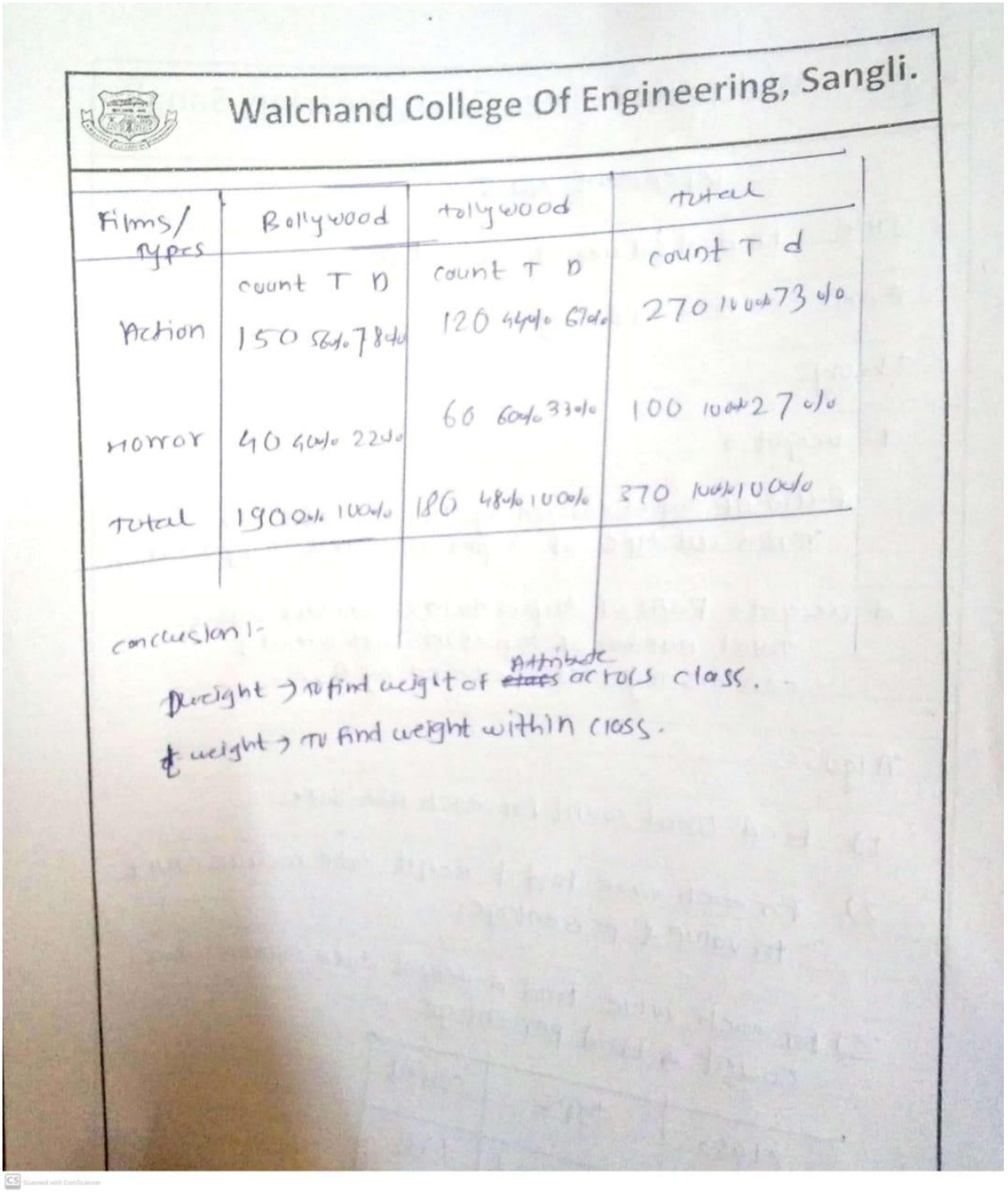
Algo:-

1) Find Total count for each Attribute.

2) For each value find t-weight, take rowwise count for value & percentage.

3) For each value find d-weight take column-wise count & Find percentage.

class	type	count
Action Films	Bollywood	150
	Tollywood	120
horror Films	Bollywood	40
	Tollywood	60



Walchand College Of Engineering, Sangli.

Films/ types	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	190 100% 100%	180 48% 100%	370 100% 100%

conclusion :-

D weight → to find weight of ~~class~~ Attribute actors class.

t weight → to find weight within class.

Input file:- Input DataSet

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

PROGRAM CODE:

```
#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;

}
```

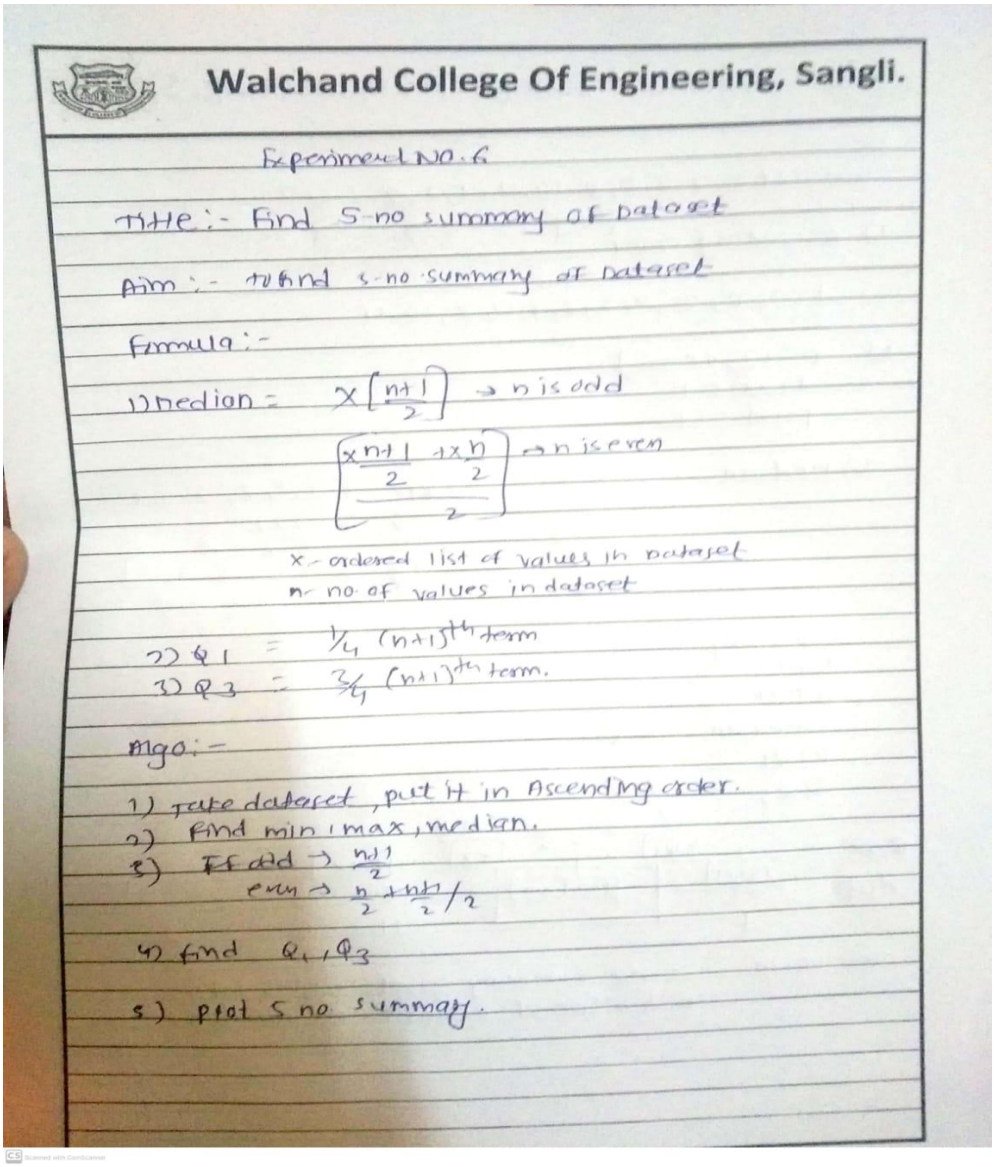
OUTPUT:

Experiment No. 6

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)



Ex.

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

1) Ascending order.

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

2) n = 12 (even)
Max = 11
min = 1

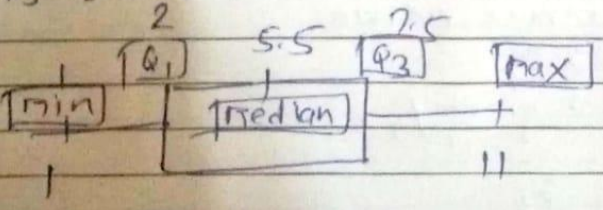
3) Median = $\frac{5^{th} + 6^{th}}{2}$
$= \frac{5+6}{2}$
$= 5.5$

4) Q_1 = [1, 1, 2, 2, 4, 5]
$Q_1 = \frac{2+2}{2} = 2$

5) Q_3 = [6, 6, 7, 8, 10, 11]
$Q_3 = \frac{7+8}{2} = 7.5$

Summary:-
1) min = 1
2) max = 11
3) median = 5.5
4) Q_1 = 2
5) Q_3 = 7.5

Box plot :-



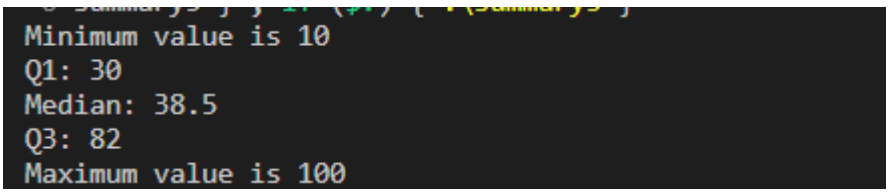
conclusion:- 5 no summary gives insights into distribution of values for one variable.

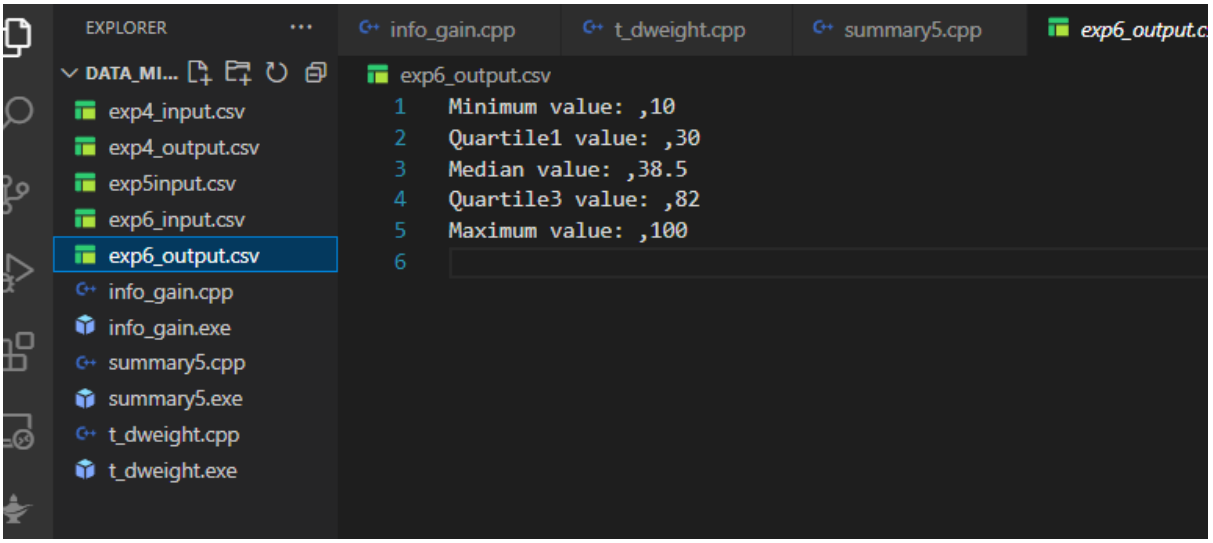
```
#include <bits/stdc++.h>
using namespace std;
float median(vector<int> a)
{
int size = a.size();
if (size % 2 == 1)
return a[size/2];
else
return (a[(size / 2) - 1] + a[size / 2]) / 2.0;

}
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;
}
```

Output:





Input file:- Input Dataset

Output file:- Output of 5 Number Summary

Experiment No. 7

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.

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

Experiment No. 7

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 frequent itemset

Formula:-

$$\text{support}(A) = \frac{\text{No. of Transactions in which A occurs}}{\text{Total No. of Transactions}}$$

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

Algo:-
1) Take min-support Input from user.
2) Read csv file
3) calculate min-frequency for dataset.
4) for each level calculate frequency of Itemset & support count.

O) If support count > min support => frequent Itemset.

Ex.

Transaction	Itemsets.
T_1	{A, B, C}
T_2	{A, C}
T_3	{A, D}
T_4	{B, E, F}

Walchand College Of Engineering, Sangli.

min support = 50%
confidence = 50%

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

Step 1] = k = 1

Itemset	support_cnt
A	3
B	2
C	2
D	1
E	1
F	1

Compare min support & eliminate other who didn't qualify condition.

L1 =

Itemset	sup_cnt
A	3
B	2
C	2

Step 2 : k = 2

Itemset	sup_count
{A, B}	1
{A, C}	2
{B, C}	1

L2 =

Itemset	sup_cnt
{A, C}	2

we stop here no. Further frequent itemset is found

∴ frequent itemset = {A, C}

conclusion :- Frequent itemset shows which item appears together in transaction or relation more frequently.

Page No.:

Program Code:

```
#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 !=
```

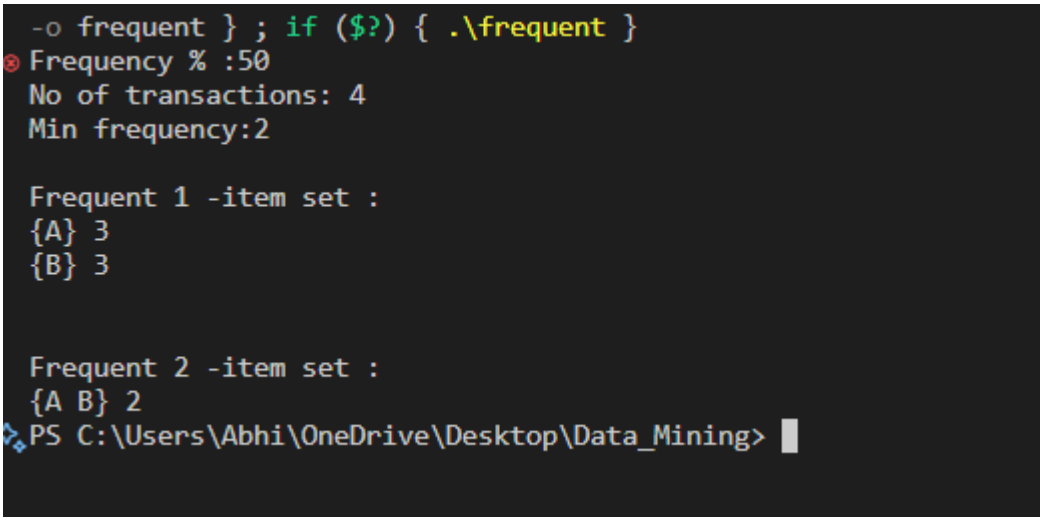
```
tmpset.end(); it++)

{
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;
}
```

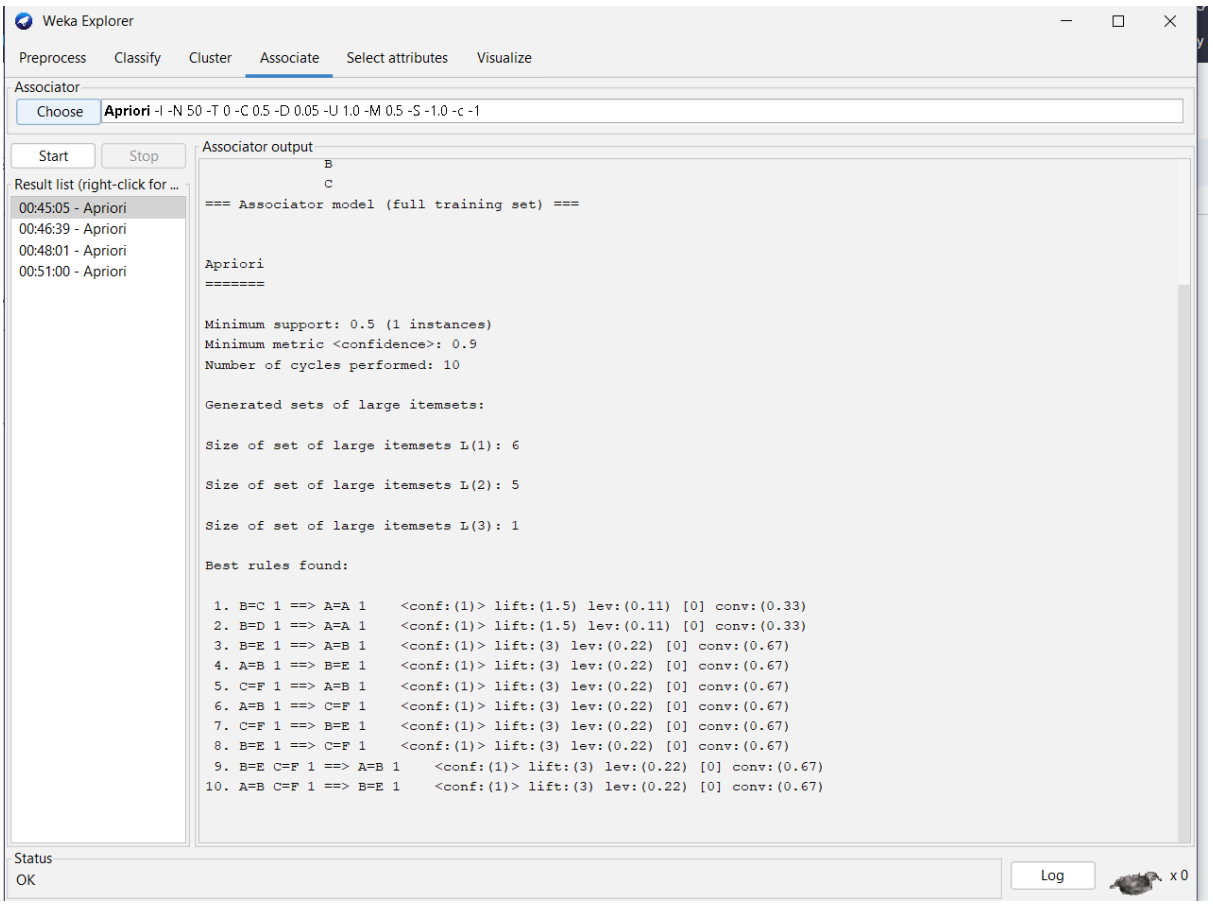
Output:



Weka

Input : FREQUENT_ITEMSET

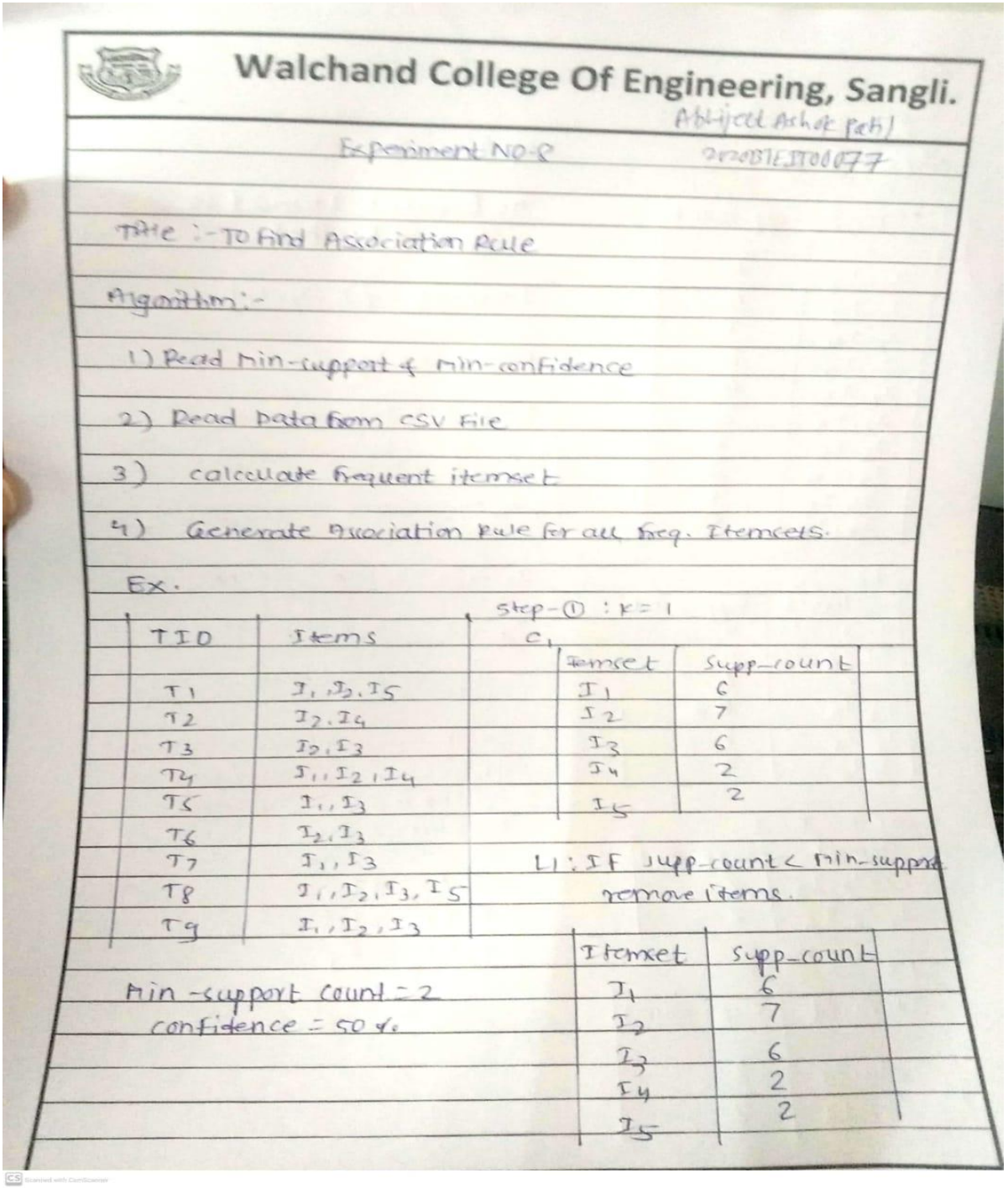
Output:


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -I -N 50 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.5 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
00:45:05 - Apriori
00:46:39 - Apriori
00:48:01 - Apriori
00:51:00 - Apriori
Associator output
B
C
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.5 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 10
Generated sets of large itemsets:
Size of set of large itemsets L(1): 6
Size of set of large itemsets L(2): 5
Size of set of large itemsets L(3): 1
Best rules found:
1. B=C 1 ==> A=A 1 <conf:(1)> lift:(1.5) lev:(0.11) [0] conv:(0.33)
2. B=D 1 ==> A=A 1 <conf:(1)> lift:(1.5) lev:(0.11) [0] conv:(0.33)
3. B=E 1 ==> A=B 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
4. A=B 1 ==> B=E 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
5. C=F 1 ==> A=B 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
6. A=B 1 ==> C=F 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
7. C=F 1 ==> B=E 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
8. B=E 1 ==> C=F 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
9. B=E C=F 1 ==> A=B 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
10. A=B C=F 1 ==> B=E 1 <conf:(1)> lift:(3) lev:(0.22) [0] conv:(0.67)
Status
OK
Log
x 0

Experiment No 8

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.

Abhijeet Ashok Pathi

Experiment No:8 2020BTEIT00077

Title :- To find Association Rule

Algorithm:-

1) Read min-support & min-confidence
2) Read data from CSV File
3) calculate frequent itemset
4) Generate Association Rule for all freq. Itemsets.

Ex.

TID	Items
T1	I_1, I_2, I_5
T2	I_2, I_4
T3	I_2, I_3
T4	I_1, I_2, I_4
T5	I_1, I_3
T6	I_2, I_3
T7	I_1, I_3
T8	I_1, I_2, I_3, I_5
T9	I_1, I_2, I_3

Min-support count = 2
confidence = 50%

Step-① : K = 1

C_1

Itemset	Supp-count
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

L1 : IF supp-count < min-support remove items.

Itemset	Supp-count
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

Step ② K=2

Itemset	supp-count
I_1, I_2	4
I_1, I_3	4
I_1, I_4	1
I_1, I_5	2
I_2, I_3	4
I_2, I_4	2
I_2, I_5	2
I_3, I_4	0
I_3, I_5	1
I_4, I_5	0

L2

Itemset	supp-count
$I_2\ I_1$ ✓	4
$I_1\ I_3$ ✓	4
$I_2\ I_2$	4
$I_2\ I_4$	2
$I_2\ I_5$	2
$I_1\ I_5$ ✓	2

Step 3 : K=3

Itemset	supp-count
I_1, I_2, I_3	2
I_1, I_2, I_5	2

L3

Itemset	supp-count
I_1, I_2, I_3	2
I_1, I_2, I_5	2

Step 4 :-

No frequent Itemset is formed further.

Itemset = $\{I_1, I_2, I_3\}$

Rules can be

$[I_1 \wedge I_2] \rightarrow [I_3]$ ⇒ confidence = $\frac{2}{4} \times 100 = 50\%$

$[I_1 \wedge I_3] \rightarrow [I_2]$ ⇒ $\frac{2}{4} \times 100 = 50\%$

$[I_2 \wedge I_3] \rightarrow [I_1]$ ⇒ $\frac{2}{4} \times 100 = 50\%$

$I_1 \rightarrow [I_2 \wedge I_3]$ ⇒ $\frac{2}{6} \times 100 = 33\%$

$I_2 \rightarrow [I_1 \wedge I_3]$ ⇒ $\frac{2}{7} \times 100 = 28\%$

$I_3 \rightarrow [I_1 \wedge I_2]$ ⇒ $\frac{2}{6} \times 100 = 33\%$

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

$I_1 \wedge I_2 \rightarrow I_3$

$I_1 \wedge I_3 \rightarrow I_2$

$I_2 \wedge I_3 \rightarrow I_1$

Program code:

```
#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++)
```

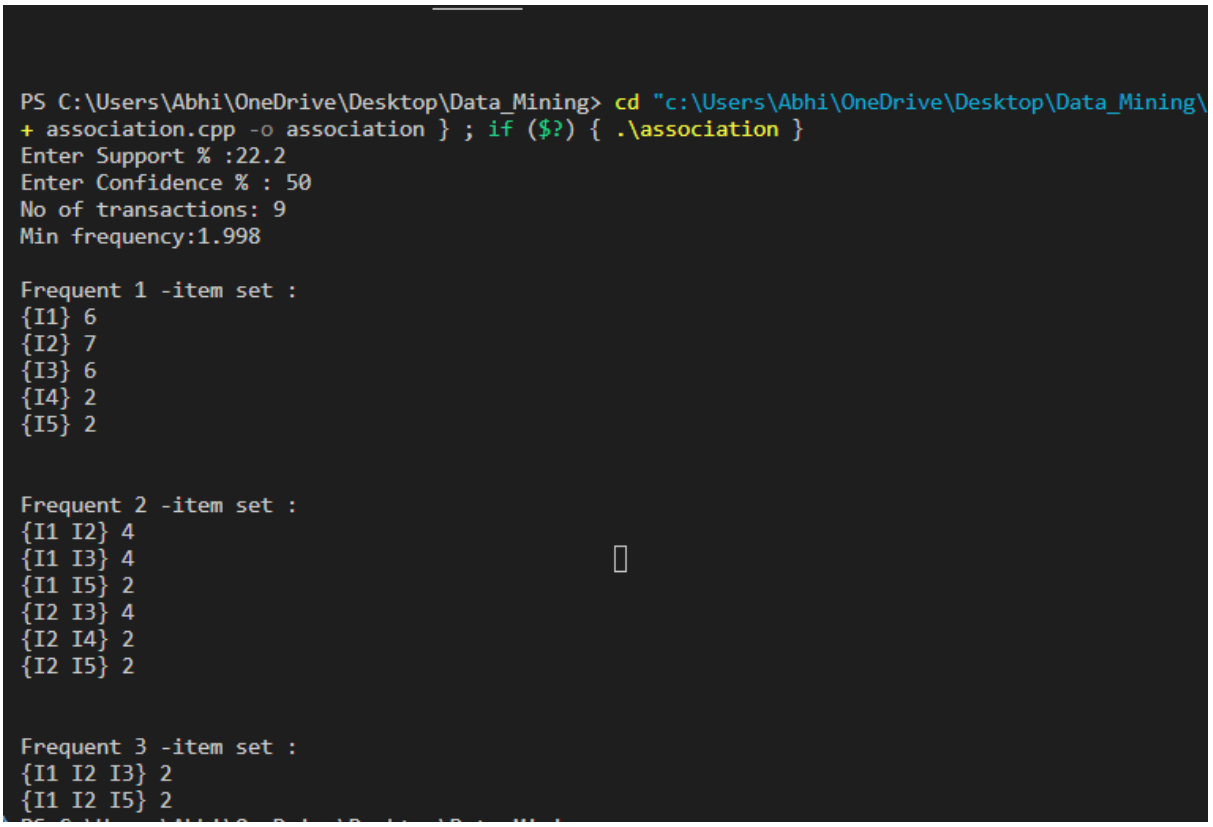
```
{
products.insert(*it);
freq[*it]++;
}
}

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++;
}
```

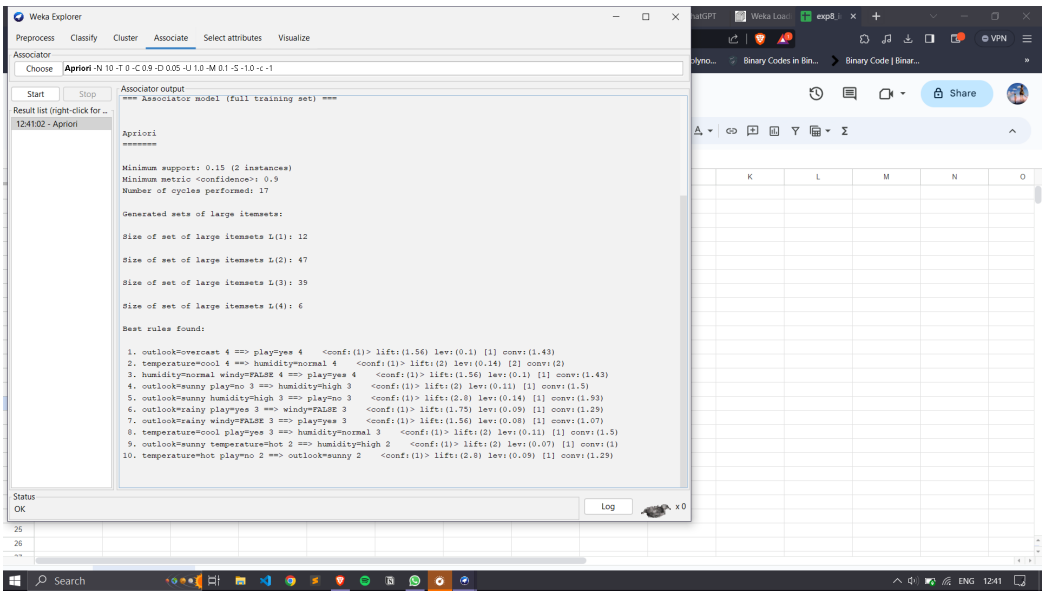
```
generateAssociationRules(prev);
return 1;
}
```

output:



INPUT DATA:Association Data

Output:


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
12:41:02 - Apriori
Associator output
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17
Generated sets of large itemsets:
Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6
Best rules found:
1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)
Status
OK
Log
x 0
Share
Search
ENG
12:41

Experiment No 9

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-9

Title- Find correlation between items/entities.

formula :-

$$\text{correlation coefficient} = \frac{P(A \cup B)}{P(A)\,P(B)}$$

Algorithm :-

1) Take input dataset
2) If needed convert data entries into Binary (1/0) or Boolean (Y/N) format.
3) Take input from user of 2 entities between which correlation has to find.
4) Find 'Y' count for entity 1 & entity 2 & count events in which both are Yes.
5) Apply formula to find correlation coefficient

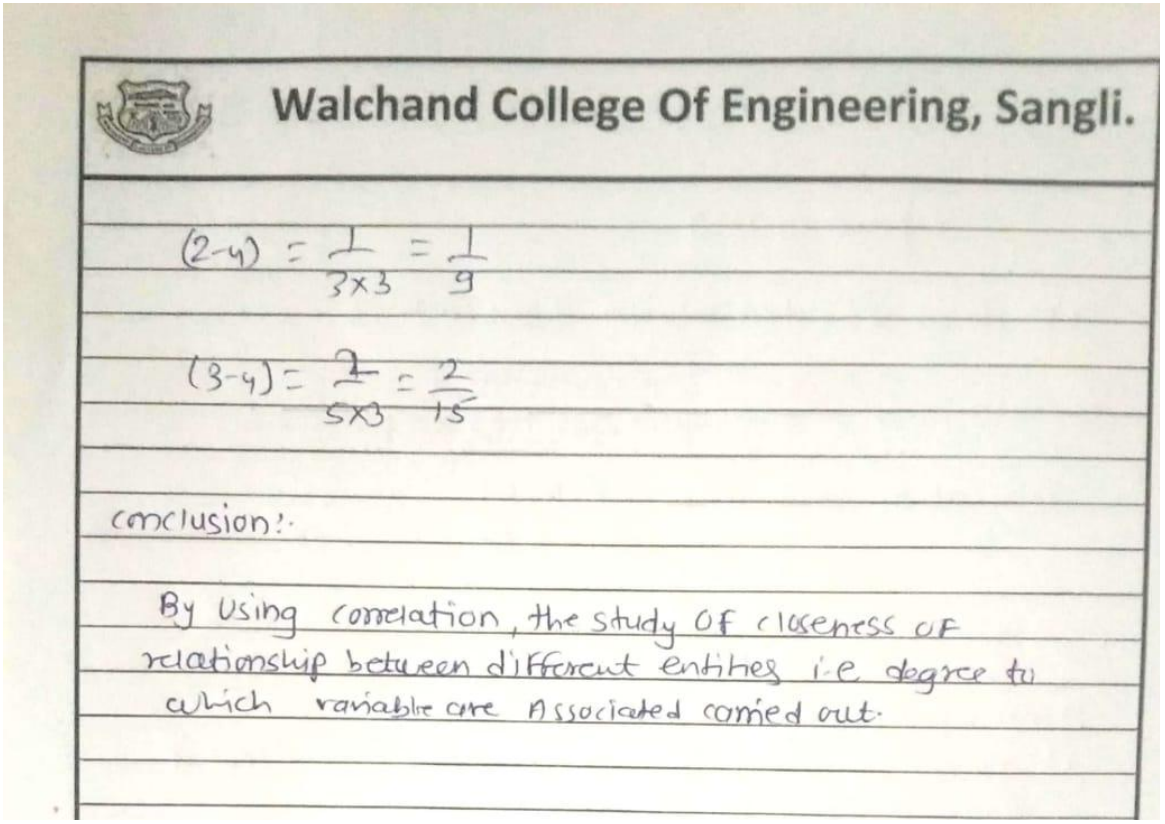
Ex.

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 $(1-2) = \frac{1}{3\times3} = \frac{1}{9}$ $(1-4) = \frac{1}{3\times3} = \frac{1}{9}$

$(1-3) = \frac{3}{3\times5} = \frac{1}{5}$ $(2-3) = \frac{3}{3\times5} = \frac{1}{5}$



Walchand College Of Engineering, Sangli.

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

$(3-4) = \frac{1}{5 \times 3} = \frac{2}{15}$

conclusion:-

By using correlation, the study of closeness of relationship between different entities i.e. degree to which variable are associated carried out.

Program code:

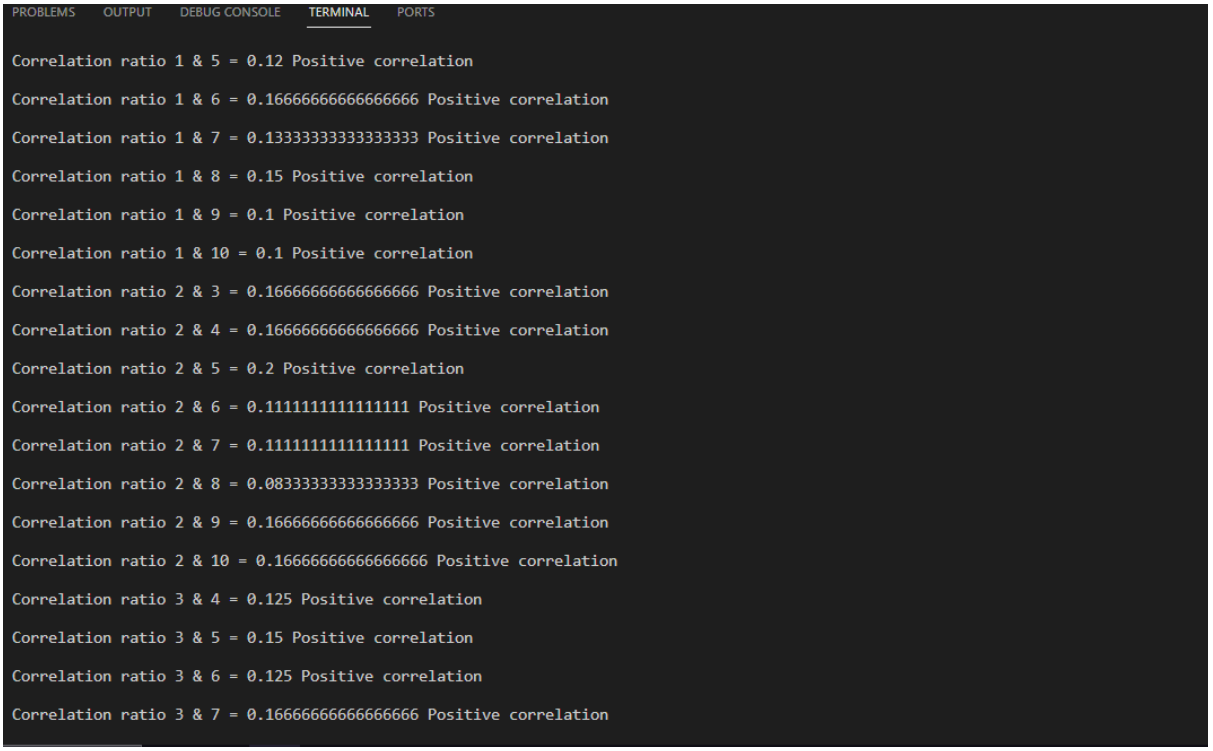
```
import openpyxl

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")
```

OUTPUT:



Input file:- **Input Dataset**

Output file:- **Correlation Output**

Experiment No .10

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.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Experiment No.10

Title:- Distance & cluster

Aim→ To compute centre of cluster assuming all multidimensional points belonging to one cluster

k-means clustering mainly performs 2 tasks:-

1) Determine Best value for k centre points or centroid by an iterative process.

2) Assign each data point to its closest k-centre.

Algorithm:-

1) choose no. of clusters

2) Randomly select any k-data points as cluster centre.

3) calculate Distance betn each point & cluster centre by using Euclidean Distance.

4) Assign each data point to some cluster.

5) Recompute centre by taking mean.

6) Repeat 2→5

Ex.

Points	co-ordinate
P_1	(10,40)
P_2	(20,10)
P_3	(15,20)
P_4	(25,30)
P_5	(15,5)

consider 5 points as a part of single cluster.

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

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

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

$c'(x', y') = (17, 21)$

using Euclidean formula, finding distance

$d(c', P_1) = \sqrt{(17-10)^2 + (21-40)^2} = 20.24$

$d(c', P_2) = \sqrt{(17-20)^2 + (21-10)^2} = 11.40$

$d(c', P_3) = \sqrt{(17-15)^2 + (21-20)^2} = 2.23$

$d(c', P_4) = \sqrt{(17-25)^2 + (21-30)^2} = 12.04$

$d(c', P_5) = \sqrt{(17-15)^2 + (21-5)^2} = 16.12$

Nearest point from Imaginary centre is $P(15, 20)$

Distance Matrix

	P_1	P_2	P_3	P_4	P_5
P_1	0				
P_2	31.62	0			
P_3	20.61	11.18	0		
P_4	18.02	20.81	14.14	0	
P_5	35.35	7.07	15	26.92	0

Program code:

```
#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"<<",";
```

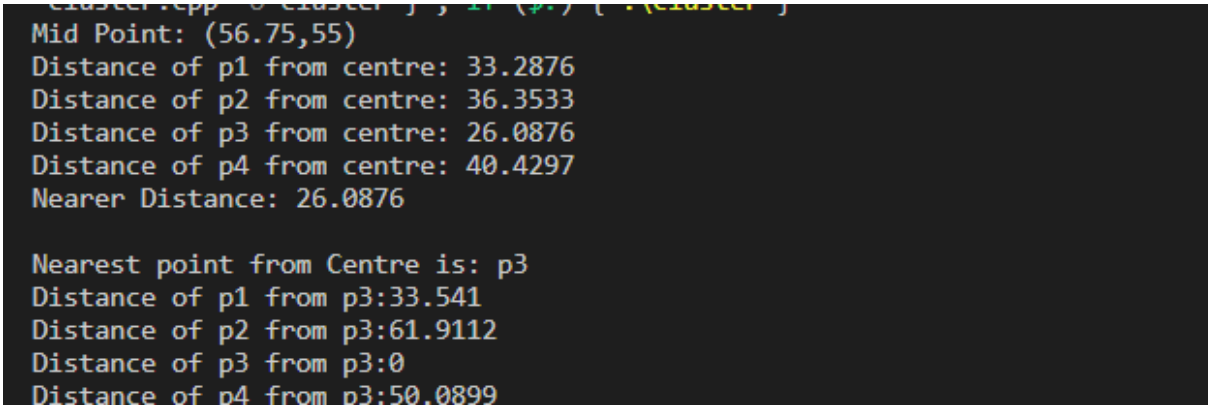
```
pair<int,int>p;

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;

}
```

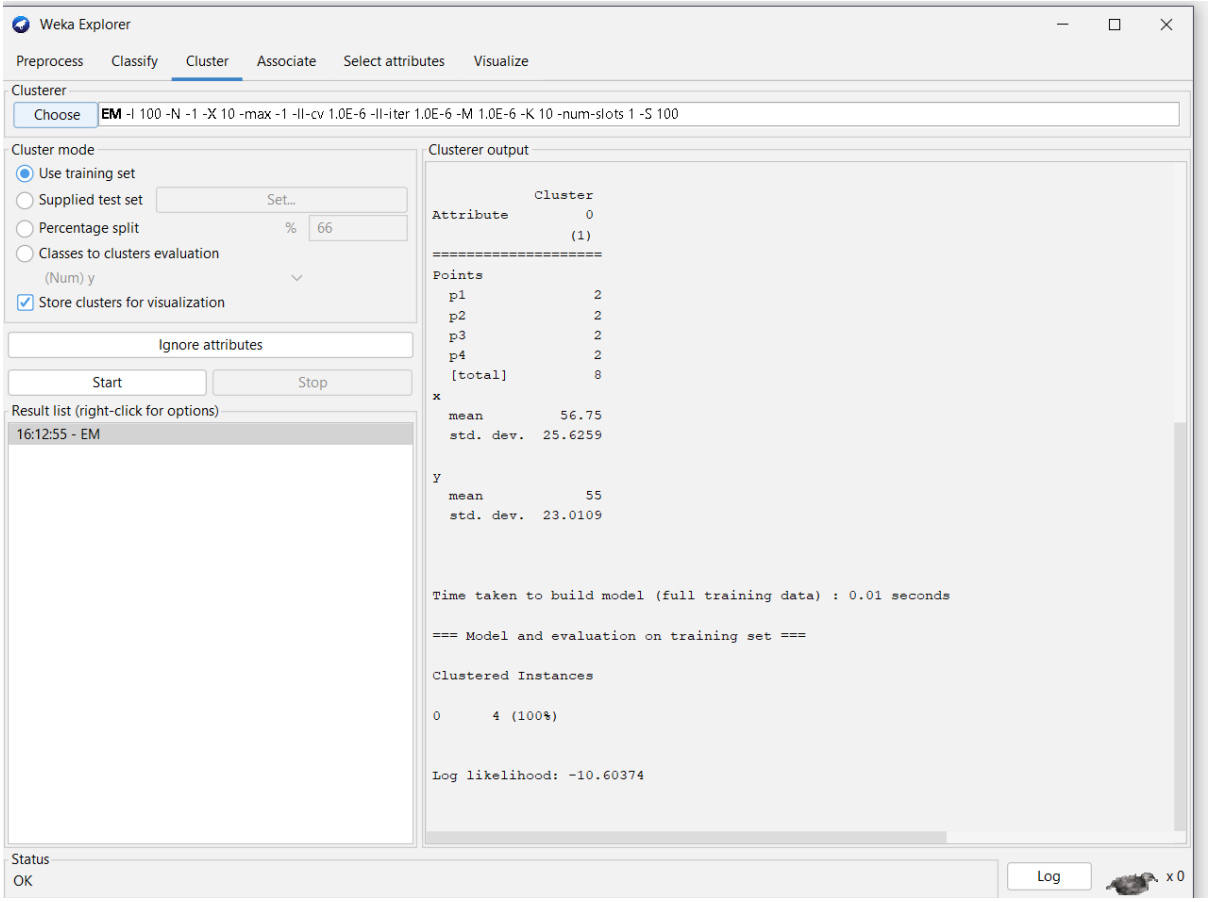
```
out<<d<<",";
if(i==v.size()-1)
out<<"0"<<",";
}

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;

}
```

output:





Input file:- Input points

Output file:- Output in Lower triangular matrix

Experiment No 11

Title :- Agglomerative Hierarchical clustering using single linkage method

Theory:

Agglomerative Hierarchical Clustering is a hierarchical clustering algorithm that starts with each data point as its own cluster and then successively merges or "agglomerates" clusters based on a defined linkage criterion. In the single linkage method (also known as the minimum linkage method), the distance between two clusters is defined as the minimum pairwise distance between the data points in the two clusters. The algorithm continues to merge clusters until all data points are in a single cluster, creating a hierarchical tree-like structure known as a dendrogram.

Walchand College Of Engineering, Sangli.

Experiment NO. 11

TITLE :- Agglomerative hierarchical clustering using single Linkage method

Algorithm :-

1) Read Input dataset
2) calculate similarity of one cluster with all other clusters.
3) consider every datapoint as a individual cluster.
4) Merge the cluster which are highly similar or close to each other.
5) Recalculate approximate matrix for each cluster.
6) Repeat 4 & 5 until single cluster is remained.

Example :-

① 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

② 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

③ Merge [AB]

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

④ Merge D & EF

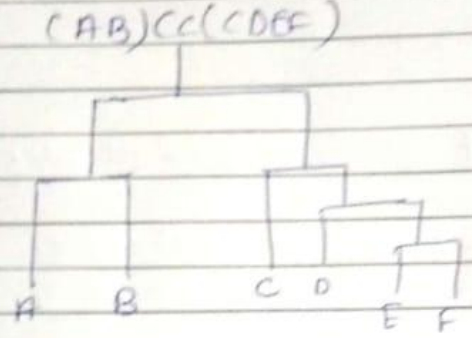
	AB	C	DEF
AB	0		
C	37	0	
DEF	57	30	0

⑤ Merge C & DEF

	AB	CDEF
AB	0	
CDEF	37	0

Now Distance between cluster AB & CDEF is 37

Dendogram



conclusion:-

Hierarchical agglomerative clustering starts with treating each observation as individual cluster i.e begins with singleton sets of each point & then iteratively merge cluster until all datapoints are merged into single cluster.

Dendogram generated is used to represent hierarchical relationship between objects.

Program Code:

```
#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;
```

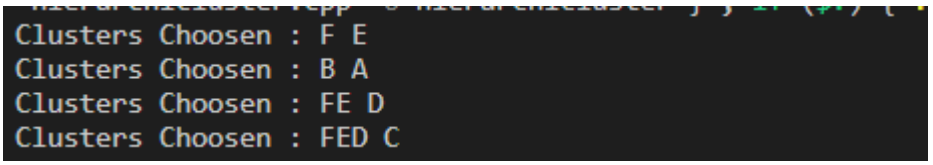
```
if (pt1[0] > pt2[0])
{
up = pt2;
down = pt1;
}
else
{
up = pt1;
down = pt2;
}
string newPt = down + up;
for (auto p : dm)
{
point = p.first;
if (point[0] > newPt[0])
{
dm[point][newPt] = min(dm[point][up], dm[point][down]);
}
}
for (auto p : dm[down])
{
point = p.first;
int d1 = p.second;
if (point[0] < 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 << ",";
}
```

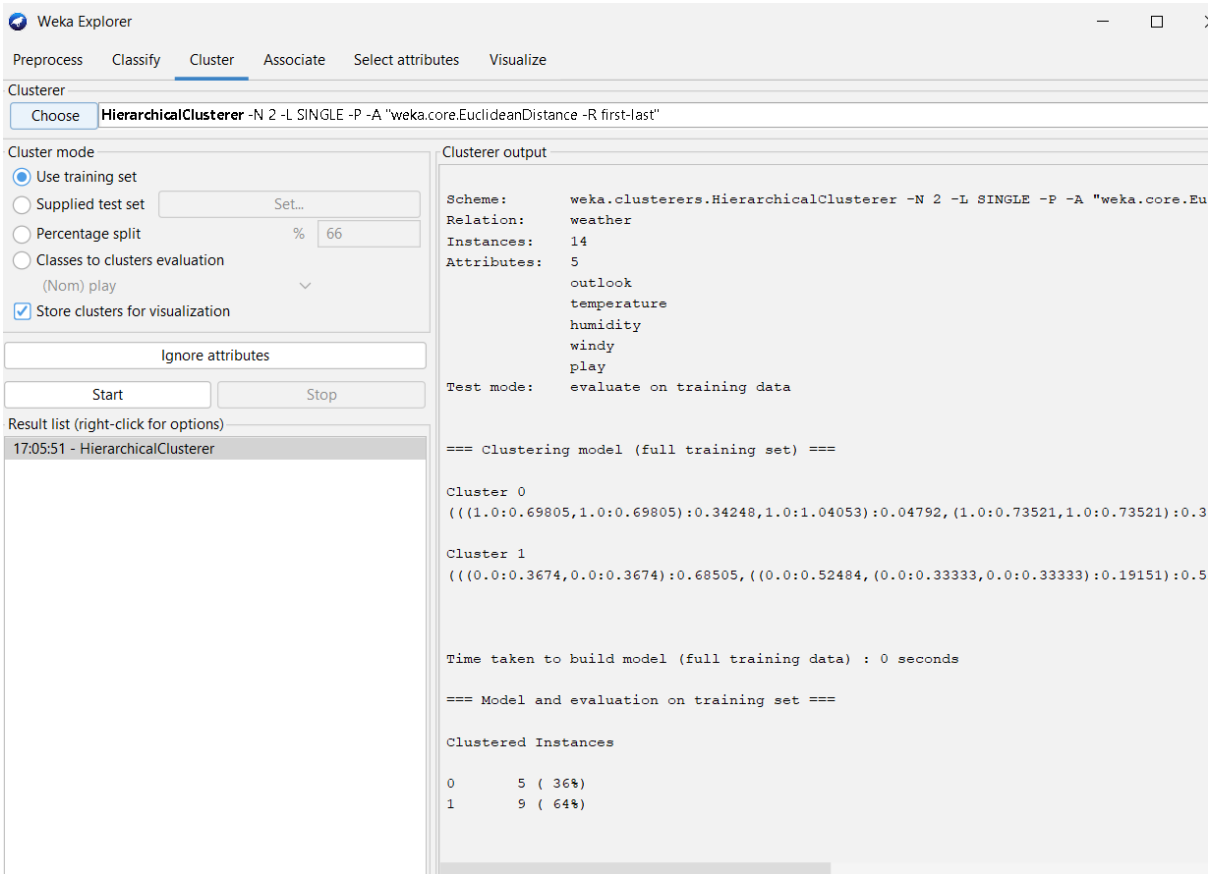
```
fw << "\n";
for (auto p : dm)
{
fw << p.first << ",";
for (auto pp : p.second)
{
fw << pp.second << ",";

}
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)
{
```

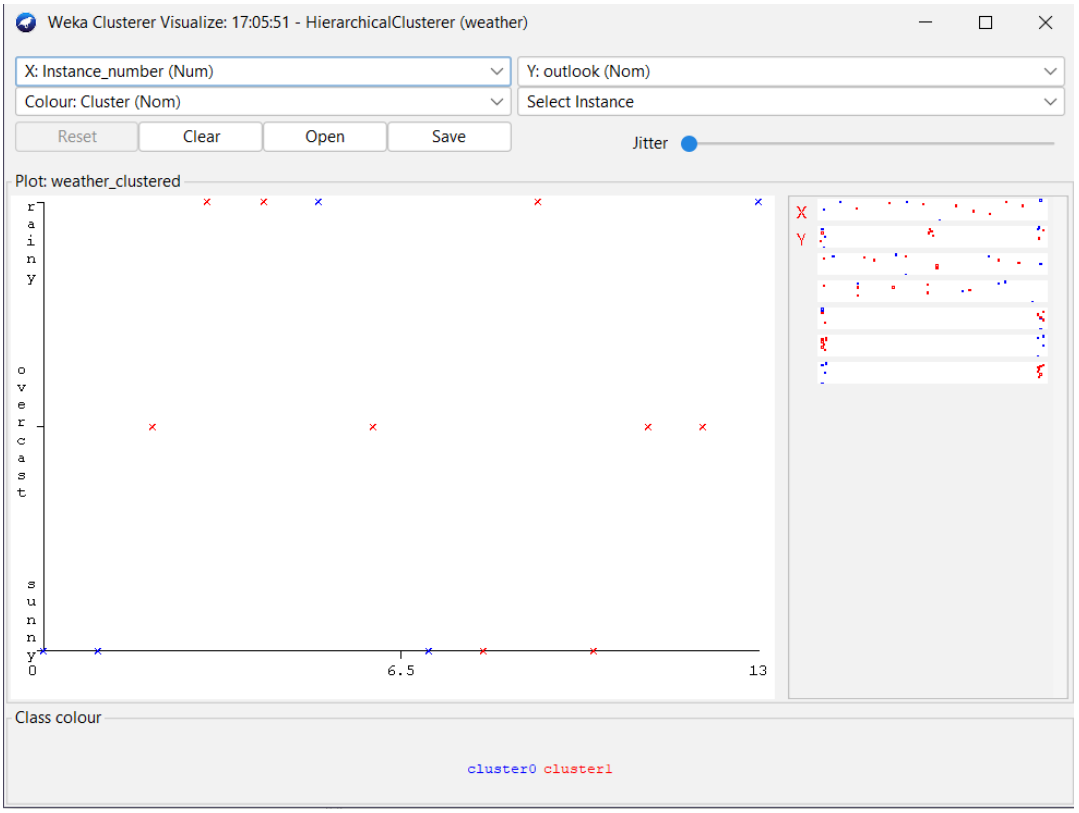
```
j++;
continue;
}
len++;
}
for (int i = 1; i <= len - 2; i++)
{
string output = algomerative(input);
input = output;

}
return 0;
}
```

Output:

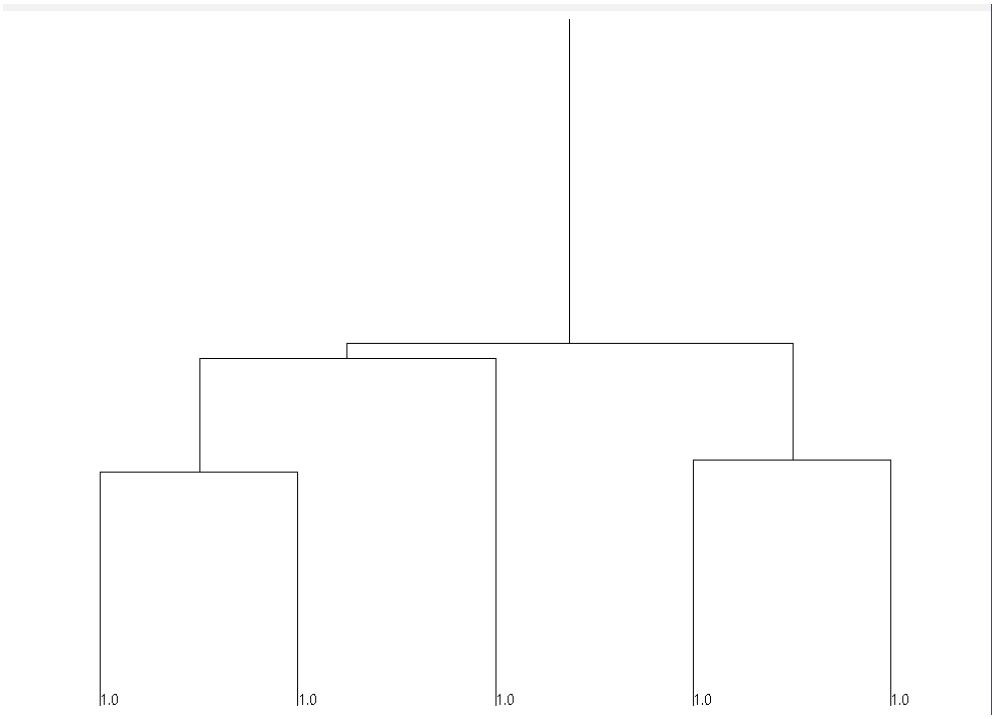


Weka output:


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) play
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
17:05:51 - HierarchicalClusterer
Clusterer output
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.Eu
Relation: weather
Instances: 14
Attributes: 5
outlook
temperature
humidity
windy
play
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
(((1.0:0.69805,1.0:0.69805):0.34248,1.0:1.04053):0.04792,(1.0:0.73521,1.0:0.73521):0.3
Cluster 1
(((0.0:0.3674,0.0:0.3674):0.68505,((0.0:0.52484,(0.0:0.33333,0.0:0.33333):0.19151):0.5
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 5 (36%)
1 9 (64%)


Weka Clusterer Visualize: 17:05:51 - HierarchicalClusterer (weather)
X: Instance_number (Num)
Y: outlook (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: weather_clustered
rainy
overcast
sunny
0
6.5
13
X
Y
Class colour
cluster0 cluster1

Dendrogram:



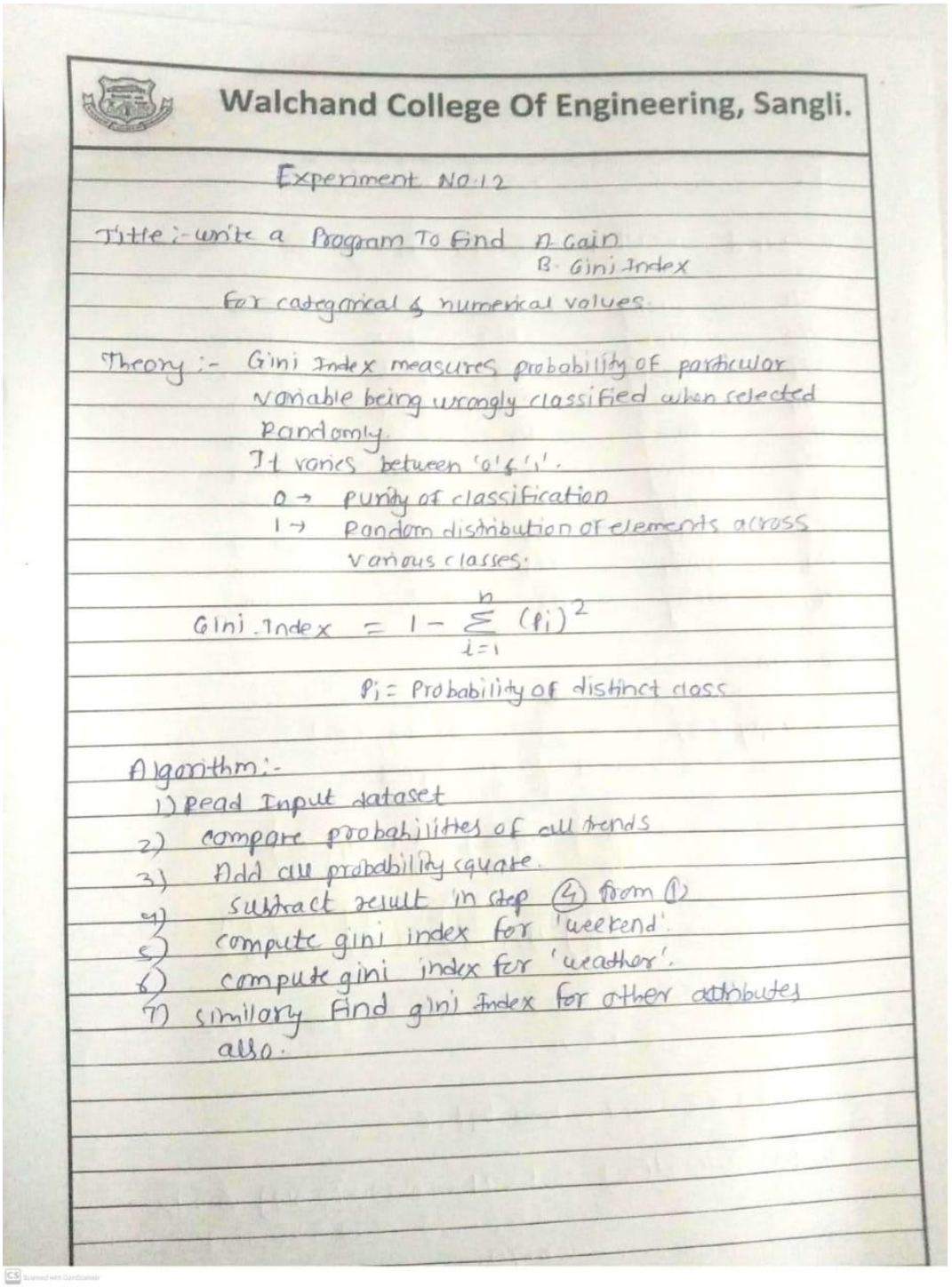
Experiment No 12

Title :write a program to find

A.Gain

B.Gini_index

For categorical and numerical values



Walchand College Of Engineering, Sangli.

Experiment No.12

Title:- write a Program To Find A Gain
B. Gini Index
For categorical & numerical values.

Theory:- Gini Index measures probability of particular variable being wrongly classified when selected Randomly.
It varies between '0' & '1'.
0 → purity of classification
1 → Random distribution of elements across various classes.

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

p_i = Probability of distinct class

Algorithm:-
1) Read Input dataset
2) compare probabilities of all trends
3) Add all probability square.
4) Subtract result in step (4) from (1)
5) compute gini index for 'weekend'.
6) compute gini index for 'weather'.
7) similary find gini Index for other attributes also.

Ex.

weekend	weather	Parents	Money	decision
W_1	Sunny	Yes	Rich	cinema
W_2	Sunny	No	Rich	Tennis
W_3	windy	Yes	Rich	cinema
W_4	Rainy	Yes	Poor	cinema
W_5	Rainy	No	Rich	Stay In
W_6	Rainy	Yes	Poor	cinema
W_7	windy	No	Poor	cinema
W_8	windy	No	Rich	Shopping
W_9	windy	Yes	Rich	cinema
W_{10}	Sunny	No	Rich	Tennis

for Gain

$$\text{Entropy}(S) = -\sum_{i=\{\text{cinema, shopping, Tennis, Stay In}}^{4} P_i \log_2(P_i)$$

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

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

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

$$= 1.571$$

Now we need to Find best of

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

$$= 0.70$$

Gain (S, Parents) = 1.571 − (0.5)×0 − (0.5)×(1.922)

= 1.571 − 0.961

= 0.61

Gain(S, Money) = 1.571 − (0.7)×(1.842) − (0.3)×0

= 1.571 − 1.2894

= 0.2816

Attribute	Gain	
Weather	0.70	→ weather has highest gain → it will be chosen as root node in decision tree.
Parents	0.61	
Money	0.2816	

calculating Gini Index for 'Decision Attribute':

Cinema → 6

Tennis → 2

Stay In → 1

Shopping → 1

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

= 1 − 0.42

= 0.58

calculating GI for 'Money'

Rich → 7

Poor → 3

For Money = Poor → 3 examples with cinema

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

For money = Rich

Tennis → 2

Cinema → 3

Stay In → 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 Avg (Money)

$$= 0 \times \frac{3}{10} + 0.694 \times \frac{7}{10}$$

$$= 0.486$$

Calculating Gini Index For 'Parents'

Yes → 5

No → 5

Parent : Yes → cinema → 5

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

Parent = NO

Tennis → 2

Stay In → 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 Avg (Parents)

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

$$= 0.36$$

calculating gini Index for 'weather'

Sunny → 3
Windy → 4
Rainy → 3

weather = sunny → cinema = 1
tennis = 2

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

$$= 0.444$$

weather = Rainy → cinema = 2
Stay In = 1

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

$$= 0.444$$

weather = windy → cinema = 3
Shopping = 1

$$\text{Gain}(W) = 1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right)$$

$$= 0.375$$

$$\text{weighted Avg (weather)} = 0.444 \times \frac{3}{10} + 0.444 \times \frac{3}{10} + 0.375 \times \frac{4}{10}$$

= 0.416

Attribute GINI Index

weather	0.416	→ Parents have lowest Gini Index
Parents	0.36	→ Root Node for Decision Tree.
Money	0.486	

(conclusion):-

Decision tree is used to split dataset as tree based on set of rules & conditions.

Info gain → focuses on purity & impurity

Gini Index → Probability for a random instance to further classify it.

Program code:

```
#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;
}
```

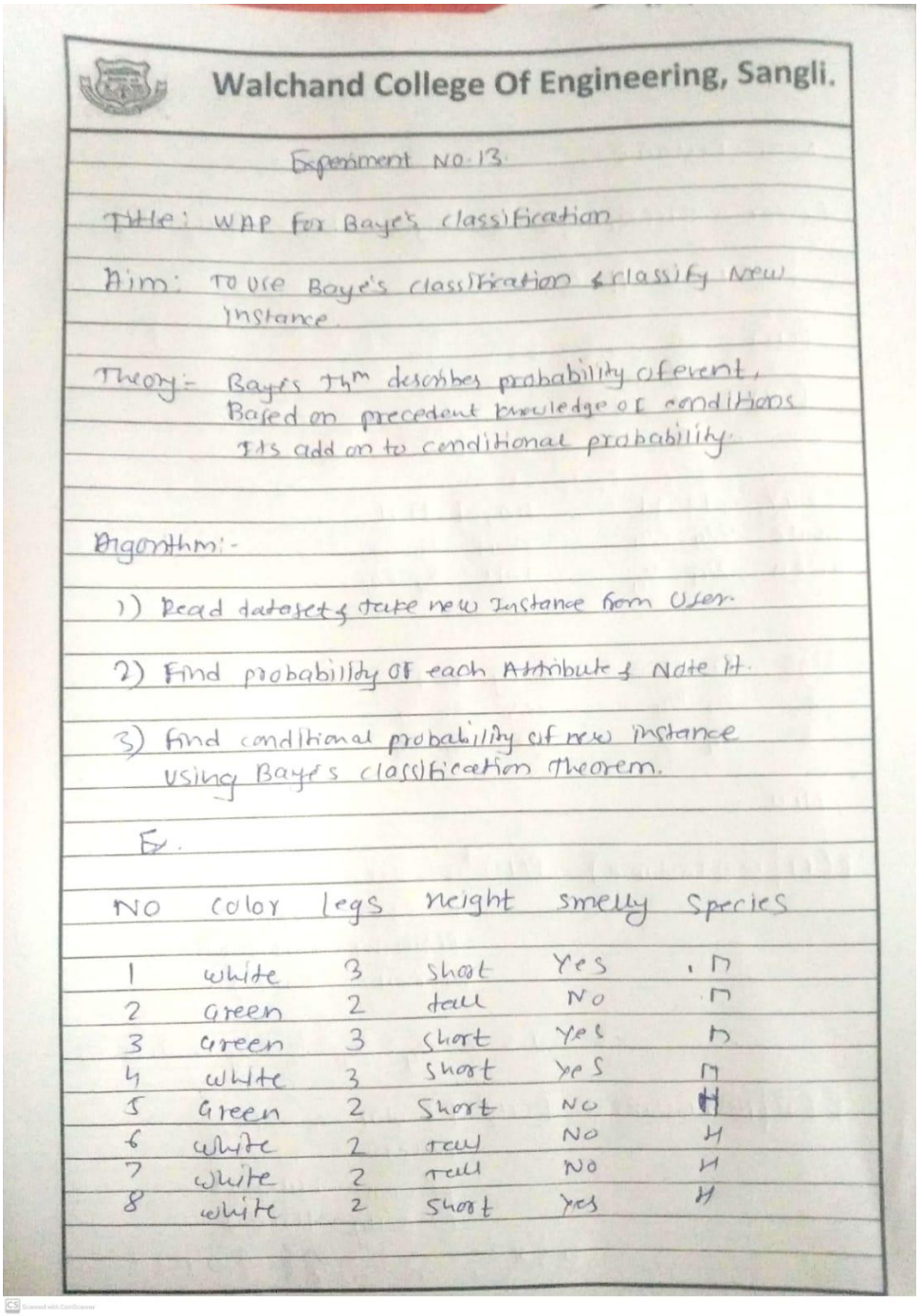
Output:

```
Enter Child Column Number : 2
Parent Entropy: 0.940286
Child Entropy * Their proportion : 0.78845
Info gain : 0.151836
```

Experiment No 13

Title :WAP for Baye's classification

Theory:Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.



Walchand College Of Engineering, Sangli.

Experiment No: 13

Title: WAP for Baye's classification

Aim: To use Baye's classification & classify new instance.

Theory:- Bayes Thm describes probability of event, Based on precedent knowledge of conditions Its add on to conditional probability.

Algorithm:-

1) Read dataset & take new instance from User.

2) Find probability of each Attribute & Note it.

3) Find conditional probability of new instance using Baye's classification theorem.

Ex.

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, Smelly = NO)

Here M = 4, N = 4

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

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

color	M	H
White	2/4	2/4
Green	2/4	2/4

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

height	M	H
short	3/4	2/4
tall	1/4	2/4

Smelly	M	H
Yes	3/4	1/4
No	1/4	3/4

Now

$P(M \mid \text{New Instance}) = P(M) * P(\text{color} = \text{Green} \mid M) * P(\text{legs} = 2 \mid M) * P(\text{height} = \text{tall} \mid M) * P(\text{Smelly} = \text{NO} \mid M)$

$= 0.5 \times \frac{2}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} = 0.0039$

$P(H \mid \text{New Instance}) = P(H) * P(\text{color} = \text{Green} \mid H) * P(\text{legs} = 2 \mid H) * P(\text{height} = \text{tall} \mid H) * P(\text{Smelly} = \text{No} \mid H)$

$= 0.5 \times \frac{1}{4} \times \frac{1}{4} \times \frac{2}{4} \times \frac{3}{4} = 0.0468$

Here

$P(H/\text{New Instance}) > P(M/\text{New Instance})$

∴ New Instance belongs to species 'H'.

Conclusion:-

Naive Baye's classification is probabilistic classifier. It is useful for making predictions & forecasting data based on historical results.

Program Code

:#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("bayes.csv");

string color, legs, height, smelly, species;

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, color, ','))

{

columns.push_back(color);

}

int n = columns.size();

parent[columns[n - 1]]++;

for (int i = 1; i < n - 1; i++)

{


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

}

count++;

}

vector<string> resultclass;

for (auto it : parent)

{

resultclass.push_back(it.first);
```

```
}
vector<double> output(resultclass.size(), 1);
for (auto it : child)
{
string input;
again:
cout << "Enter " << it.first << " condition \n";
cin >> input;
auto curr = child[it.first].find(input);
if (curr == child[it.first].end())
{
cout << "no match\n";
goto again;
}
for (int i = 0; i < resultclass.size(); i++)
{
cout << child[it.first][input][resultclass[i]] << " / " <<


parent[resultclass[i]] << endl;


double val = child[it.first][input][resultclass[i]] /


parent[resultclass[i]];
output[i] *= val;
cout << output[i] << endl;
}
```

```
}
for (int i = 0; i < resultclass.size(); i++)
{
output[i] *= parent[resultclass[i]] / count;
}
double sum = accumulate(output.begin(), output.end(), 0.0f);
cout << "sum " << sum << endl;
cout << "output---------" << endl;
for (int i = 0; i < resultclass.size(); i++)
{
cout << resultclass[i] << " " << output[i] << endl;
cout << "Percentage " << (output[i] / sum) * 100 << endl;
}
}


else
{
cout << "Could not open the file\n";
}
return 0;
}
```

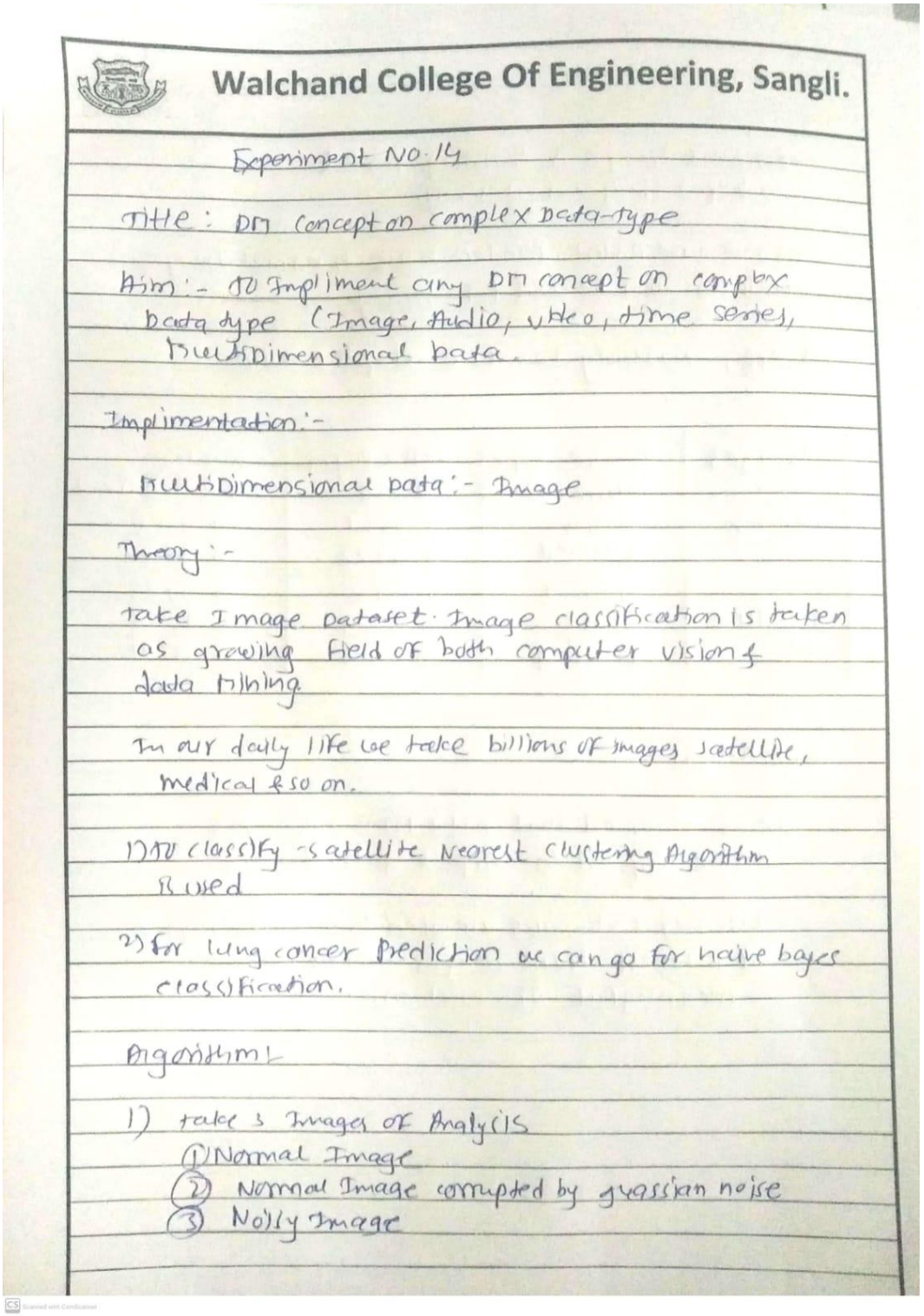
output:

```
Enter color condition
green
1 / 4
0.25
2 / 4
0.5
Enter height condition
tall
2 / 4
0.125
1 / 4
0.125
Enter legs condition
2
4 / 4
0.125
1 / 4
0.03125
Enter smelly condition
no
3 / 4
0.09375
1 / 4
0.0078125
sum 0.0507812
output---------
H 0.046875
Percentage 92.3077
M 0.00390625
Percentage 7.69231
```

Experiment No 14

Title :Data Mining concepts on complex data types

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



Walchand College Of Engineering, Sangli.

Experiment No. 14

Title : DM concept on complex Data-type

Aim :- To Impliment any DM concept on complex Data type (Image, Audio, video, time series, Multidimensional Data.

Implimentation :-

Multidimensional Data :- Image

Theory :-

Take Image Dataset. Image classification is taken as growing field of both computer vision & data mining

In our daily life we take billions of images satellite, medical & so on.

1) To classify - satellite Nearest Clustering Algorithm is used

2) For lung cancer Prediction we can go for naive bayes classification.

Algorithm :-

1) take 3 Images of Analysis
   (1) Normal Image
   (2) Normal Image corrupted by gaussian noise
   (3) Noisy Image

2) Normal Image → Training Model
Other 2 Images → Testing.

3) Now we have Added Random noise to normal Image
Now this will be used for testing.

4) Apply smoothing to reduce adaptive noise.

Images	Image type	Naive Bayes	Random Forest 0.0258
Cancer	Normal	0.216	
	Noisy	0.4331	0.2918
	Filtered	0.4282	0.2527
Satellite	Normal	0.1205	0.0099
	Noisy	0.5229	0.3539
	Filtered	0.5073	0.3371

Conclusion:-

It is observed that data mining concepts can be Applied to complex dataset

Classifier techniques are used to classify region of interest from images inorder to get meaningful observation.