Walchand College Of Engineering, Sangli

Data Mining LAB BOOK
(2022-23)-Sem VII

Name :**Shripad Rajesh Bhat**
PRN :**2019BTEIT00030**

Sr.No	Title	Page No
1	Study and use different graphs and charts (use MS-XLS).	2
2	Perform Normalization of data (Min-max and Z-score)	10
3	Find Info Gain of an attribute from given data.	23
4	Find t and d weight of a data	34
5	Find 5 no summary of a dataset (Box Plot)	41
6	Find frequent itemset from given transaction data	48
7	Extend program 6, to find association rule.	60
8	Find correlation between items/entities.	75
9	Distance and cluster	80
10	Write a program for Agglomerative Hierarchical clustering using single linkage method	91

11	Attribute for classification .Write a program to find Gini index For categorical and numerical values	103
12	WAP for Baye's classification	117

1. Study and use different graphs and charts (use MS-XLS).

Expt No 1

Study & use of different types of graph

For this experiment, I have used Hungarian Chickenpox Cases Data set, downloaded from UCI machine learning respository

Algorithm:-

Step 1 :- Compute monthwise cases of chicken pox

Step 2 :- Compute percentage cases of chickenpox for each month

Step 3 :- Take month & % two attributes & draw various graphs.

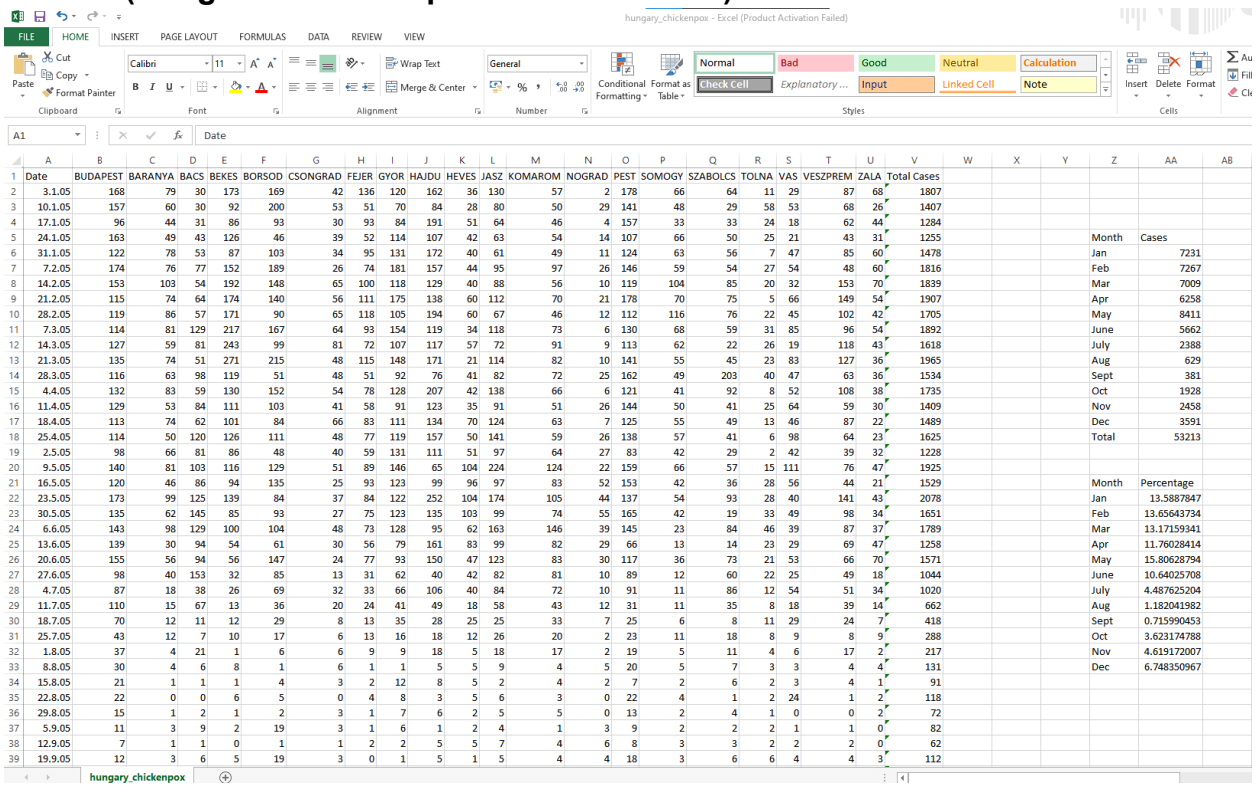
Solutions:-

P.T.O.

	Months	Cases	Percentage
1.	January	7231	13.89
2.	February	7267	13.66
3.	March	7009	13.17
4.	April	6258	11.76
5.	MAY	8411	15.81
6.	June	5662	10.64
7.	July	2388	4.49
8.	August	629	1.18
9.	September	381	0.72
10.	October	1928	3.62
11.	November	2458	4.62
12.	December	3591	6.44

Conclusion :- By using hungary chickenpox dataset provided by UCI machine learning repository we are able to understand various graphs.

Data set(Hungarian Chickenpox Cases Data Set)



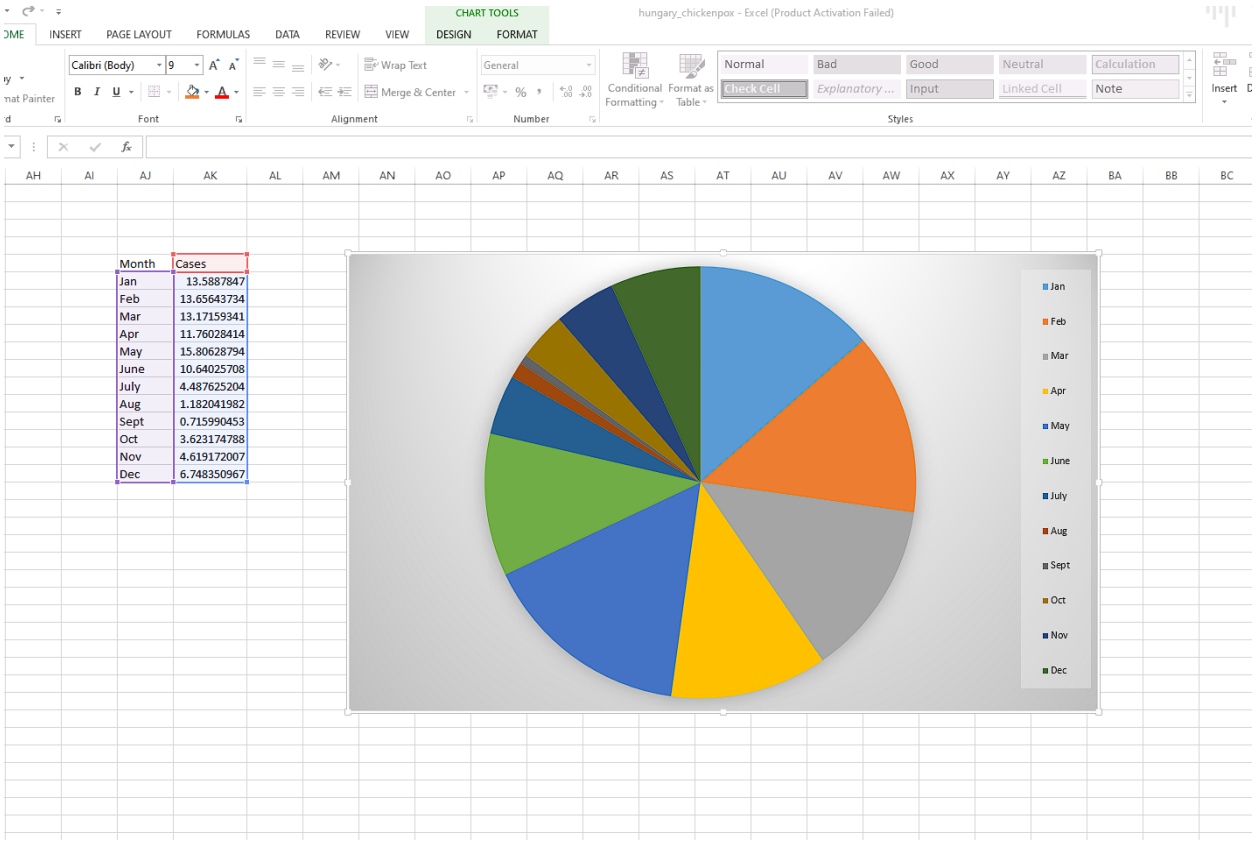
Date	BUDAPEST	BARANYA	BACS	BEKES	BORSOD	CSONGRAD	FEJER	GYOR	HAJDU	HEVES	JASZ	KOMAROM	NOGRAD	PEST	SOMOGY	SZABOLCS	TOLNA	VAS	VESZPREM	ZALA	Total Cases
3.1.05	168	79	30	173	169	42	136	120	162	36	130	57	2	178	66	64	11	29	87	68	1807
10.1.05	157	60	30	92	200	53	51	70	84	28	80	50	29	141	48	29	58	53	68	26	1407
17.1.05	96	44	31	86	93	30	93	84	191	51	64	46	4	157	33	33	24	18	62	44	1284
24.1.05	163	49	43	126	46	39	52	114	107	42	63	54	14	107	66	50	25	21	43	31	1255
31.1.05	122	78	53	87	103	34	95	131	172	40	61	49	11	124	63	56	7	47	85	60	1478
7.2.05	174	76	77	152	189	26	74	181	157	44	95	97	26	146	59	54	27	54	48	60	1816
14.2.05	153	103	54	192	148	65	100	118	129	40	88	56	10	119	104	85	20	32	153	70	1839
21.2.05	115	74	64	174	140	56	111	175	138	60	112	70	21	178	70	75	5	66	149	54	1907
28.2.05	119	86	57	171	90	65	118	105	194	60	67	46	12	112	116	76	22	45	102	42	1705
7.3.05	114	81	129	217	167	64	93	154	119	34	118	73	6	130	68	59	31	85	96	54	1892
14.3.05	127	59	81	243	99	81	72	107	117	57	72	91	9	113	62	22	26	19	118	43	1618
21.3.05	135	74	51	271	215	48	115	148	171	21	114	82	10	141	55	45	23	83	127	36	1965
28.3.05	116	63	98	119	51	48	51	92	76	41	82	72	25	162	49	203	40	47	63	36	1534
4.4.05	132	83	59	130	152	54	78	128	207	42	138	66	6	121	41	92	8	52	108	38	1735
11.4.05	129	53	84	111	103	41	58	91	123	35	91	51	26	144	50	41	25	64	59	30	1409
18.4.05	113	74	62	101	84	66	83	111	134	70	124	63	7	125	55	49	13	46	87	22	1489
25.4.05	114	50	120	126	111	48	77	119	157	50	141	59	26	138	57	41	6	98	64	23	1625
2.5.05	98	66	81	86	48	40	59	131	111	51	97	64	27	83	42	29	2	42	39	32	1228
9.5.05	140	81	103	116	129	51	89	146	65	104	224	124	22	159	66	57	15	111	76	47	1925
16.5.05	120	46	86	94	135	25	93	123	99	96	97	83	52	153	42	36	28	56	44	21	1529
23.5.05	173	99	125	139	84	37	84	122	252	104	174	105	44	137	54	93	28	40	141	43	2078
30.5.05	135	62	145	85	93	27	75	123	135	103	99	74	55	165	42	19	33	49	98	34	1651
6.6.05	143	98	129	100	104	48	73	128	95	62	163	146	39	145	23	84	46	39	87	37	1789
13.6.05	139	30	94	54	61	30	56	79	161	83	99	82	29	66	13	14	23	29	69	47	1258
20.6.05	155	56	94	56	147	24	77	93	150	47	123	83	30	117	36	73	21	53	66	70	1571
27.6.05	98	40	153	32	85	13	31	62	40	42	82	81	10	89	12	60	22	25	49	18	1044
4.7.05	87	18	38	26	69	32	33	66	106	40	84	72	10	91	11	86	12	54	51	34	1020
11.7.05	110	15	67	13	36	20	24	41	49	18	58	43	12	31	11	35	8	18	39	14	662
18.7.05	70	12	11	12	29	8	13	35	28	25	25	33	7	25	6	8	11	29	24	7	418
25.7.05	43	12	7	10	17	6	13	16	18	12	26	20	2	23	11	18	8	9	8	9	288
1.8.05	37	4	21	1	6	6	9	9	18	5	18	17	2	19	5	11	4	6	17	2	217
8.8.05	30	4	6	8	1	6	1	1	5	5	9	4	5	20	5	7	3	3	4	4	131
15.8.05	21	1	1	1	4	3	2	12	8	5	2	4	2	7	2	6	2	3	4	1	91
22.8.05	22	0	0	6	5	0	4	8	3	5	6	3	0	22	4	1	2	24	1	2	118
29.8.05	15	1	2	1	2	3	1	7	6	2	5	5	0	13	2	4	1	0	0	2	72
5.9.05	11	3	9	2	19	3	1	6	1	2	4	1	3	9	2	2	2	1	1	0	82
12.9.05	7	1	1	0	1	1	2	2	5	5	7	4	6	8	3	3	2	2	2	0	62
19.9.05	12	3	6	5	19	3	0	1	5	1	5	4	4	18	3	6	6	4	4	3	112

Month	Cases
Jan	7231
Feb	7267
Mar	7009
Apr	6258
May	8411
June	5662
July	2388
Aug	629
Sept	381
Oct	1928
Nov	2458
Dec	3591
Total	53213

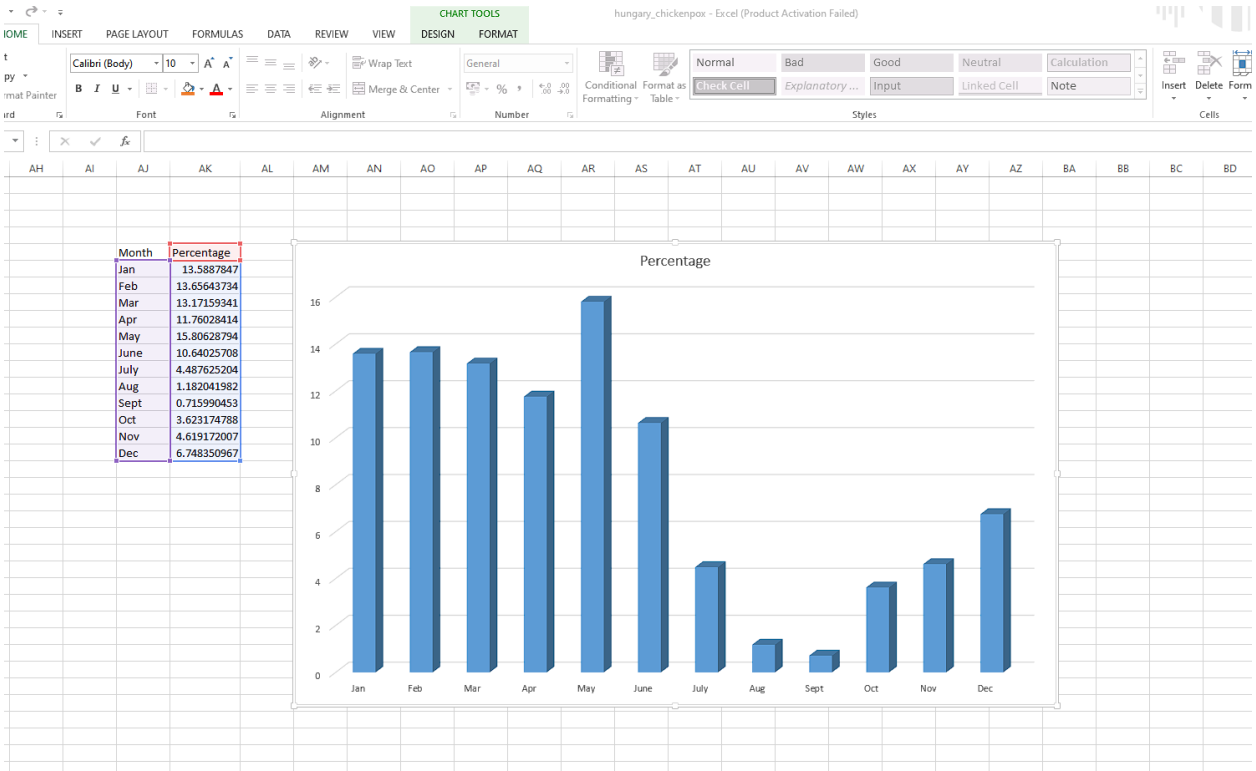
Month	Percentage
Jan	13.5887847
Feb	13.65643734
Mar	13.17159341
Apr	11.76028414
May	15.80628794
June	10.64025708
July	4.487625204
Aug	1.182041982
Sept	0.715990453
Oct	3.623174788
Nov	4.619172007
Dec	6.748350967

Output:

1)Pie chart

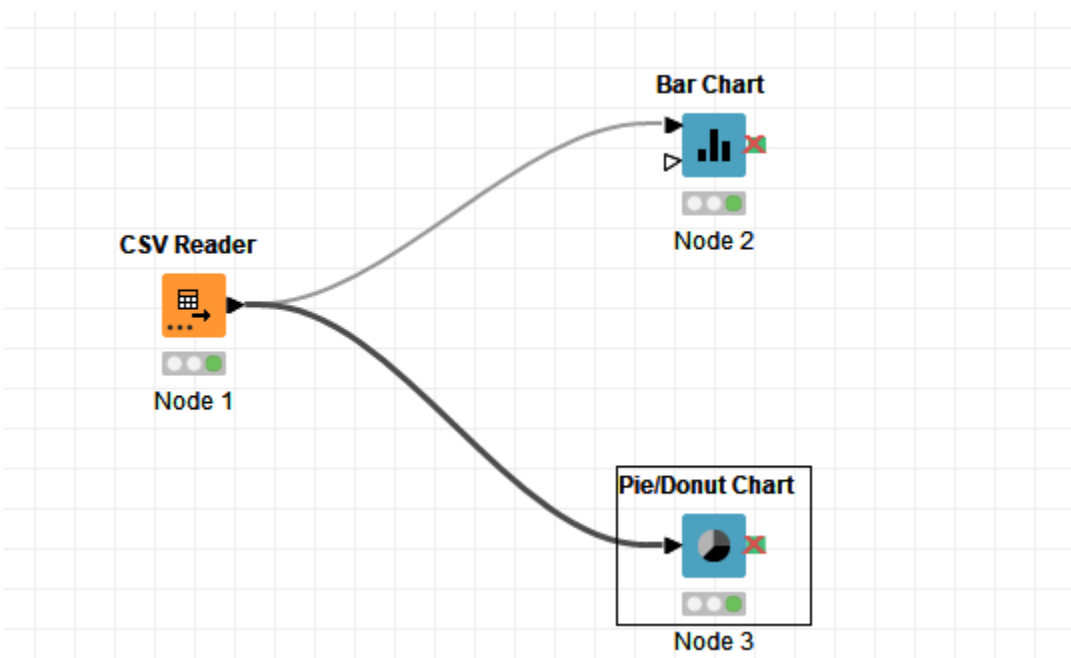

hungary_chickenpox - Excel (Product Activation Failed)
CHART TOOLS
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
DESIGN
FORMAT
Calibri (Body)
9
Wrap Text
Merge & Center
General
Conditional Formatting
Format as Table
Normal
Bad
Good
Neutral
Calculation
Check Cell
Explanatory ...
Input
Linked Cell
Note
Insert
Font
Alignment
Number
Styles
Month
Cases
Jan 13.5887847
Feb 13.65643734
Mar 13.17159341
Apr 11.76028414
May 15.80628794
June 10.64025708
July 4.487625204
Aug 1.182041982
Sept 0.715990453
Oct 3.623174788
Nov 4.619172007
Dec 6.748350967
Jan
Feb
Mar
Apr
May
June
July
Aug
Sept
Oct
Nov
Dec

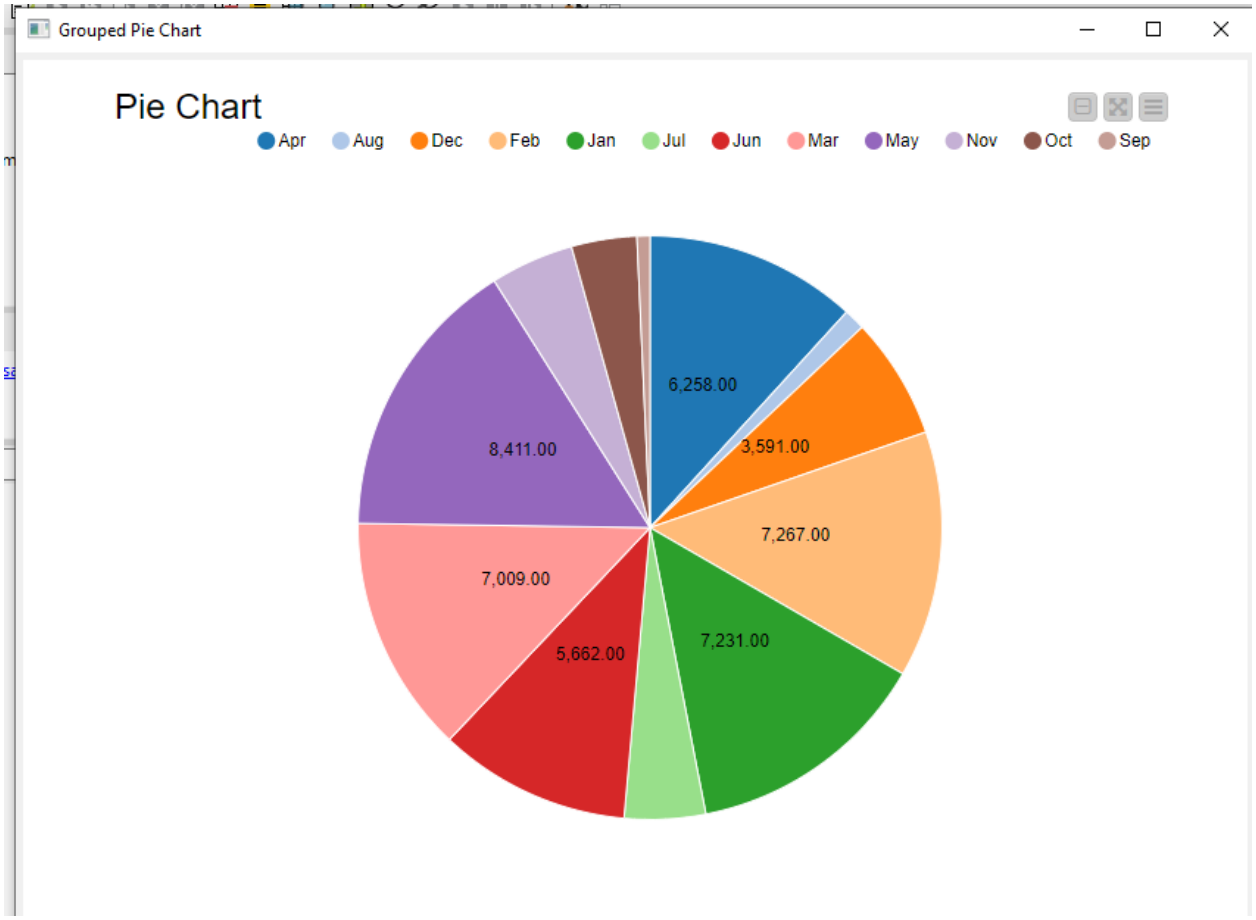
2)bar chart

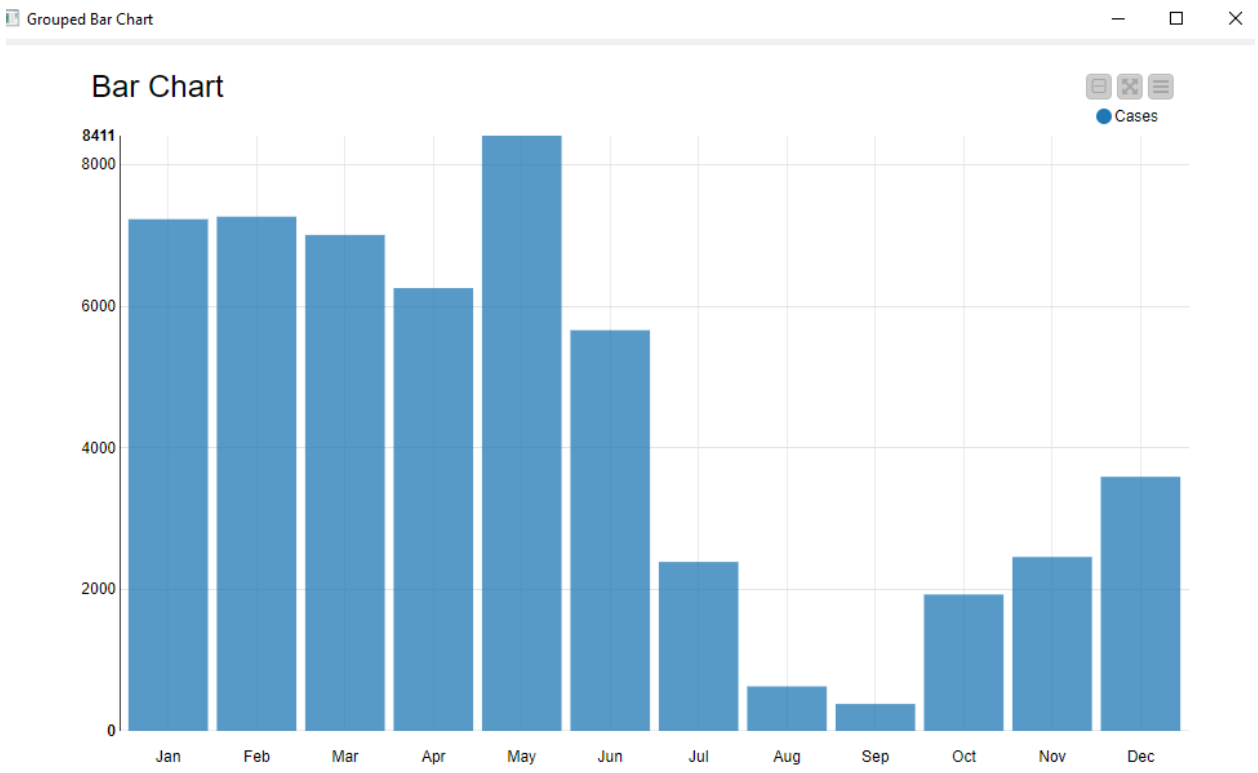


Month	Percentage
Jan	13.5887847
Feb	13.65643734
Mar	13.17159341
Apr	11.76028414
May	15.80628794
June	10.64025708
July	4.487625204
Aug	1.182041982
Sept	0.715990453
Oct	3.623174788
Nov	4.619172007
Dec	6.748350967

KNIME OUTPUTS:




Grouped Pie Chart
Pie Chart
Apr
Aug
Dec
Feb
Jan
Jul
Jun
Mar
May
Nov
Oct
Sep
6,258.00
3,591.00
7,267.00
7,231.00
5,662.00
7,009.00
8,411.00



Conclusion:

By using the Hungarian Chickenpox Cases dataset provided by UCI machine learning repository, I was able to understand the various graphs, charts. I also drew a graph for a huge data set by performing some calculations.

Expt 2: Perform Normalization of data (Min-max and Z-score).

Expt No. 2:- Perform Normalisation of data (Min-max & z-score)

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

Min-Max Normalisation:- In this technique, linear transformation is performed on original data. Minimum & maximum value is replaced according to formula.

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

Example:-

marks = [3, 5, 7, 8, 9]

min(A) = minimum value = 3

max(A) = maximum value = 9

new-min = 0 new-max = 1

For marks 3:-

$$min\text{-}max = \frac{3-3}{9-3} \times (1-0) + 0 = 0$$

For marks 5:

$$min\text{-}max = \frac{5-3}{9-3} \times (1-0) + 0 = 0.3333$$

For marks 7

min-max $\Rightarrow \frac{7-3}{9-3} \times (1-0) + 0 = 0.6667$

For marks 8 :-

min-max $\Rightarrow \frac{8-3}{9-3} \times (1-0) + 0 = 0.8333$

For marks 9 :-

min-max $\Rightarrow \frac{9-3}{9-3} \times (1-0) + 0 = \underline{1}$

Marks	Marks after min-max Normalisation
3	0
5	0.3333
7	0.6667
8	0.8333
9	1

Algorithm for min-max Normalisation

Step 1 :- Find the maximum value from given attribute set.

Step 2 :- Find attribute minimum value from given attribute set.

Step 3 :- Calculate & show the average value from array & larger from average.

Step 4 :- Calculate & show normalised value of original attribute set.

* Z score Normalisation :-

It helps in normalisation of data. If we normalize data into simpler form with help of z-score normalisation then it

is very easy to understand.

Z-score formula

$$Z = \frac{x-\mu}{\sigma}$$

$x \rightarrow$ score

$\mu \rightarrow$ mean

$\sigma \rightarrow$ std deviation

Marks
3
5
7
8
9

$$S.D = \sqrt{\frac{(x-\bar{x})^2}{n}}$$

mean $(\bar{x})$ of marks $= \frac{3+5+7+8+9}{5} = 6.4$

$$S.D = \sqrt{\frac{(3-6.4)^2+(5-6.4)^2+(7-6.4)^2+(8-6.4)^2+(9-6.4)^2}{5}}$$

$$= \sqrt{4.64} = \underline{\underline{2.15407}}$$

Z score for 3 $= \frac{x-\mu}{\sigma} = \frac{3-6.4}{2.15407} = -1.57840$

Z score for 5 $= \frac{x-\mu}{\sigma} = \frac{5-6.4}{2.15407} = -0.64993$

Z score for 7 $= \frac{x-\mu}{\sigma} = \frac{7-6.4}{2.15407} = 0.27854$

Z score for 8 $= \frac{x-\mu}{\sigma} = \frac{8-6.4}{2.15407} = 0.74277$

Z score for 9 $= \frac{x-\mu}{\sigma} = \frac{9-6.4}{2.15407} = 1.20701$

Algorithm for z-score :-

Step 1 :- Calculate the mean of attribute set

Step 2 :- Calculate std deviation of given attribute set

Step 3 :- Now, find z score value for each current value in attribute

Step 4 :- formula for z score

$$z\text{-score} = \frac{x-\mu}{\sigma}$$

Conclusion :-

1. The transformed values obtained by calculation and by Knime for min-max identical.
2. There is slight difference between z-score solved and Knime output
3. So, successfully performed transformation of given data using min-max and z-score

CODE :

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

double mean_ofArray(vector<double> MinMaxArray, int
num_elements){
		double mean;
		double  sum=0, valuesAbovemean=0;

		for (int i=0; i<num_elements; i++)
			sum+=MinMaxArray[i];

		mean=sum/num_elements;

		for (int i=0; i<num_elements; i++)
			if (MinMaxArray[i]>mean)
				valuesAbovemean++;


		cout<<"Values above the mean are:
"<<valuesAbovemean<<endl;
		cout<<"mean: "<< mean<<endl;
		return mean;
}
```

```
double Maximum_fromArray(vector<double> MinMaxArray, int
num_elements){
		double maxi=INT_MIN;
			for (int i=0; i<num_elements; i++){
				if (MinMaxArray[i]>maxi)
maxi=MinMaxArray[i];
			}
		return maxi;
}
```

```
double Minimun_fromArray( vector<double> MinMaxArray, int
num_elements){
		double mini=INT_MAX;
		for (int i=0; i<num_elements; i++){
				if(MinMaxArray[i]<mini)
					mini=MinMaxArray[i];
		}
	return mini;
}
```

```
vector<double> norm_by_minmax(int min, int max, int
new_min,int new_max, vector<double>MinMaxArray, int
num_elements){
```

```
            cout<<"\n\nNormalisation using min-max \n";
            vector<double> normalized(num_elements);
                for (int i=0; i<num_elements; i++){
                    normalized[i]=(((MinMaxArray[i]
-min)/(max-min))*(new_max-new_min)) + new_min;
                }
            cout<<"\nNormalized Values are :";
            for (int i=0; i<num_elements; i++)
                {
                cout<<normalized[i]<<" ";

            }


            cout<<"\n";

            return normalized;
}

int main(){

 vector<double> MinMaxArray;
   string fname="input2.csv";
//vector<vector<string>> content;
string line, word;

fstream file ;
file.open(fname);
if(file.is_open())
{
```

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


stringstream str(line);

while(getline(str, word, ','))
MinMaxArray.push_back(stod(word));

}
}

        for(auto i:MinMaxArray)
        cout<<i<<" ";
        cout<<endl;


        int num_of_elements=MinMaxArray.size();
        vector<double>alias=MinMaxArray;
        int new_max, new_min,old_min,old_max;

        cout<<"Enter New Min: ";
        cin>>new_min;
        cout<<"Enter New Max: ";
        cin>> new_max;

        double mean;
        old_max =
Maximum_fromArray(MinMaxArray,num_of_elements);
```

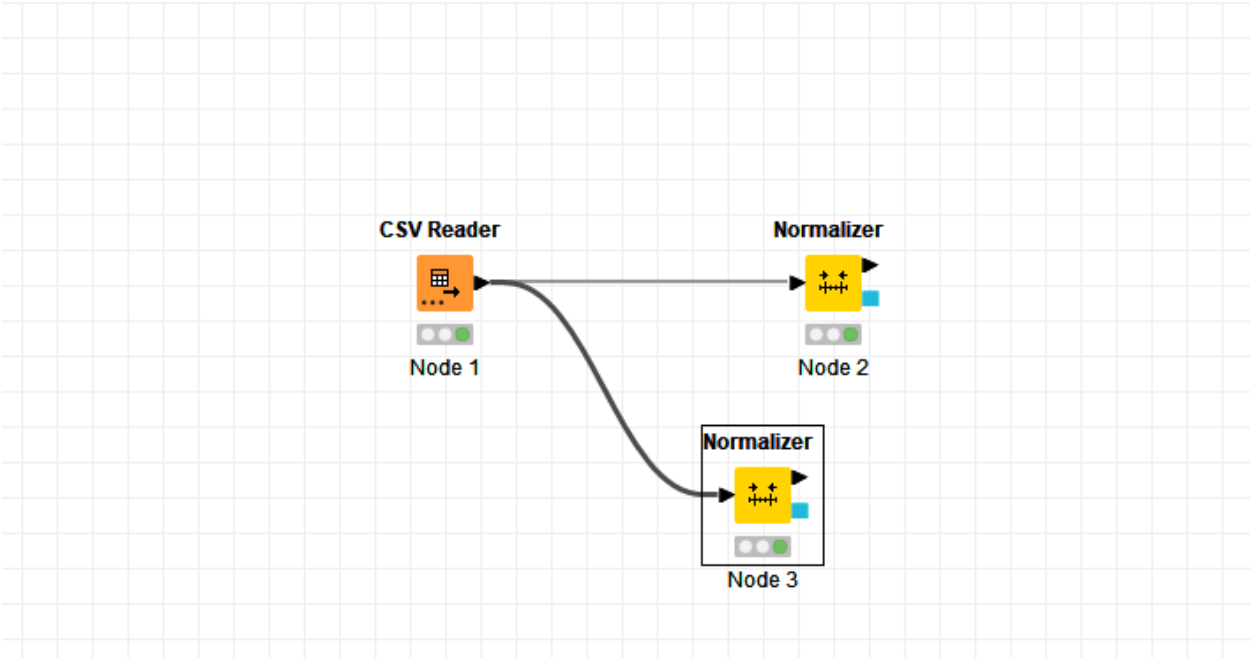
```
	old_min =
Minimun_fromArray(MinMaxArray,num_of_elements);



	mean = mean_ofArray(MinMaxArray,num_of_elements);

	vector<double>v=norm_by_minmax(old_min, old_max,
new_min,new_max,MinMaxArray,num_of_elements);



	cout<<"\n\nz-score Normalisation\n";

	int n=alias.size();
	double sd=0.0;

	for(int i=0;i<n;i++)
			sd+= pow(alias[i] - mean, 2);


	double std_dev=sqrt(sd/n);
	// cout<<"||"<<std_dev<<" "<<mean<<"|";
	vector<double> newArr=alias;
	for(int i=0;i<n;i++){
		newArr[i]=(newArr[i]-mean)/std_dev;
	}
```

```
	cout<<"\nZ-score Values are :";
	for(int i=0;i<n;i++)
		cout<<newArr[i]<<" ";

	ofstream writeFile("output2.csv");
	  writeFile<<"Marks"<<","<<"Normalized Marks(Min-Max
)"<<","<<"Normalized Marks (Z Score)"<<endl;

		for(int i=0;i<MinMaxArray.size();i++)
		writeFile <<MinMaxArray[i]  << "," <<
v[i]<<","<<newArr[i]<<endl;



	return 0;
}
```

OUTPUT :

```
C:\WINDOWS\system32\cmd.exe
3 5 7 8 9
Enter New Min: 0
Enter New Max: 1
Values above the mean are: 3
mean: 6.4


Normalisation using min-max

Normalized Values are :0 0.333333 0.666667 0.833333 1


z-score Normalisation

Z-score Values are :-1.57841 -0.649934 0.278543 0.742781 1.20702

Press any key to continue . . .
```

	A	B	C	D
1	Marks	Normalized Marks(Min-Max)	Normalized Marks (Z Score)	
2	3	0	-1.57841	
3	5	0.333333	-0.649934	
4	7	0.666667	0.278543	
5	8	0.833333	0.742781	
6	9	1	1.20702	
7				



Normalized table - 3:2 - Normalizer

File Edit Hilite Navigation View

Table "default" - Rows: 5 | Spec - Column: 1 | Properties | Flow Variables

Row ID	D Number
Row0	0
Row1	0.333
Row2	0.667
Row3	0.833
Row4	1

Normalized table - 3:3 - Normalizer

File Edit Hilite Navigation View

Table "default" - Rows: 5 | Spec - Column: 1 | Properties | Flow Variables

Row ID	D Number
Row0	-1.412
Row1	-0.581
Row2	0.249
Row3	0.664
Row4	1.08

Conclusion: Successfully performed min-max and z-score normalization with code implementation.

Expt 3. Find Info Gain of an attribute from given data.

Experiment :- 3

Title :- Find the info gain of attribute from given data

Entropy :-

It is an information theory matrix, that measures the impurity or uncertainity in a group of abreviation. It determine how a decision tree chooses to split data.

$$E = -\sum_{i=1}^{N} p_i \log(p_i)$$

where, p_i is the probability of randomly selecting on example ~~of~~ in class.

+ Information gain :- It can be defined as measure of how much information features provided about a class. It helps to determine the order of attributes in the nodes of decision tree.

+ Algorithm :-

(1) Read data from csv file
(2) Defined function to calculate entropy
(3) Calculate frequency of distinct values in particular column.
(4) Calculate probability of distinct values in particular column.
(5) calculate entropy using following formula

$$\Sigma = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

(6) Similar manner calculate info gain and find the entropy of child column.
(7) Return info gain.

Example :-

RID	Age	Income	Student	Credit rating	Buy computer
1	Youth	High	No	Fair	No
2	Youth	High	No	Excellent	No
3	middle age	High	No	Fair	Yes
4	Senior	Medium	No	Fair	Yes
5	Senior	Low	Yes	Fair	Yes
6	Senior	Low	Yes	Excellent	No
7	middle age	Low	Yes	Excellent	Yes
8	Youth	medium	No	Fair	No
9	Youth	low	Yes	fair	Yes
10	Senior	Medium	Yes	fair	Yes
11	Youth	medium	Yes	Excellent	Yes
12	middle age	Medium	No	Excellent	Yes
13	middle age	High	Yes	fair	Yes
14	Senior	Medium	No	Excellent	No

Here target class, "Buy computer"
Probability of "Yes" = 9/14
Probability of "No" = 5/14

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

$$= 0.940286$$

For attribute "Income"

	Yes	No
High	2	2
Low	3	1
Medium	4	2

$$\text{Income} = -\frac{4}{14} \times \left[\frac{2}{4}\log\left(\frac{2}{4}\right) + \frac{2}{4}\log\left(\frac{2}{4}\right)\right] - \frac{4}{14}\left[\frac{3}{4}\log\left(\frac{3}{4}\right) + \frac{1}{4}\log\left(\frac{1}{4}\right)\right] - \frac{6}{14}\left[\frac{4}{6}\log\left(\frac{4}{6}\right) + \frac{2}{6}\log\left(\frac{4}{6}\right)\right]$$

$$= 0.91106$$

$$\text{Info Gain} = 0.940286 - 0.91106$$
$$= 0.0292226$$

Conclusion :-

1. From decision tree output on Knime the splitting attribute
2. Thus, information gain can be used to find the best splitting attribute from set of attribute.

CODE :

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

double getentropy(int p,int n){

    if(p==0||n==0) return 0;
    double first = (-1*log2((1.0)*p/(p+n))*p)/(p+n);
    double second = (-1*log2((1.0)*n/(p+n))*n)/(p+n);

    return first + second;
}

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

    vector<vector<string>> content;
    vector<string> row;
    string line, word;

    fstream file (fname, ios::in);
    ofstream myFile("output.csv",ios::out);
    if(file.is_open()){
        while(getline(file, line)){
            row.clear();
            stringstream str(line);
            while(getline(str, word, ',')){
                row.push_back(word);
            }
```

```
        content.push_back(row);
    }
}
int cnt=0;
map<int,string> attribute_mapping;
bool isdone=false;
cout<<"\nDATA :\n";
for(auto vec : content){
    for(auto str: vec){
        if(!isdone){
            attribute_mapping[cnt++]=str;
        }
        cout<<str<<" ";
    }
    isdone=true;
    cout<<endl;
}

int totalYes = 0,totalNo=0,total=0;
vector<pair<int,int>> v;
int columns=content[0].size();
for(int i=1;i<content.size();i++){
    if(content[i][columns-1]=="Yes"){
        totalYes++;
    }
    else{
        totalNo++;
    }
    total++;
}
```

```
    cout<<"\n\nNumber\tAttribute\n";
    for(auto mp:attribute_mapping){
        cout<<mp.first<<"\t"<<mp.second<<"\n";
    }

    int choice,class_attribute=cnt-1;
    cout<<"\nEnter Attribute Number whose info gain is to be
calculated :";
    cin>>choice;

    cout<<"\nTotal Number of No's :"<<totalNo<<"\nTotal Number
of yes's :"<<totalYes;
    cout<<"\nTotal number of entries : "<<total<<endl;

    map<string , pair<int,int>> m;

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

        if(content[i][class_attribute]=="Yes")
            m[content[i][choice]].first++;
        if(content[i][class_attribute]=="No")
            m[content[i][choice]].second++;
    }

    cout<<"\n\n<ATTRIBUTE's yes and no count>\n";
    map<int,string> mp;
    cnt=0;
    for(auto i : m){
```

```
        cout<<i.first<<" :"<<(i.second).first<<"
"<<(i.second).second<<"\n";
        mp[cnt]=i.first;
        v.push_back(i.second);
        cnt++;
    }


    // calculating entropy
    double attributeEntropy = 0;
    myFile<<" Value , Number of yes , Number of No , Entropy \n";
    cnt=0;
    for(auto i : v){
        double entropy = getentropy(i.first,i.second) ;

myFile<<mp[cnt]<<","<<i.first<<","<<i.second<<","<<entropy<<en
dl;
        double s =( entropy * (i.first+i.second)*(1.0) )/total;
        attributeEntropy+=s;
        cnt++;
    }

    double classentropy=getentropy(totalYes,totalNo);
    cout<<"\nClass Entropy :"<<classentropy<<endl;

    cout<<"\nAttribute Entropy :"<<attributeEntropy<<endl;
    myFile<<"\n \n ";
    double info_gain = getentropy(totalYes,totalNo) -
attributeEntropy ;
```

```
    cout<<"\nInfo Gain :"<<info_gain<<"\n";
    myFile<<"information gain , "<<info_gain<<endl;

    return 0;
}
```

OUTPUT SS:

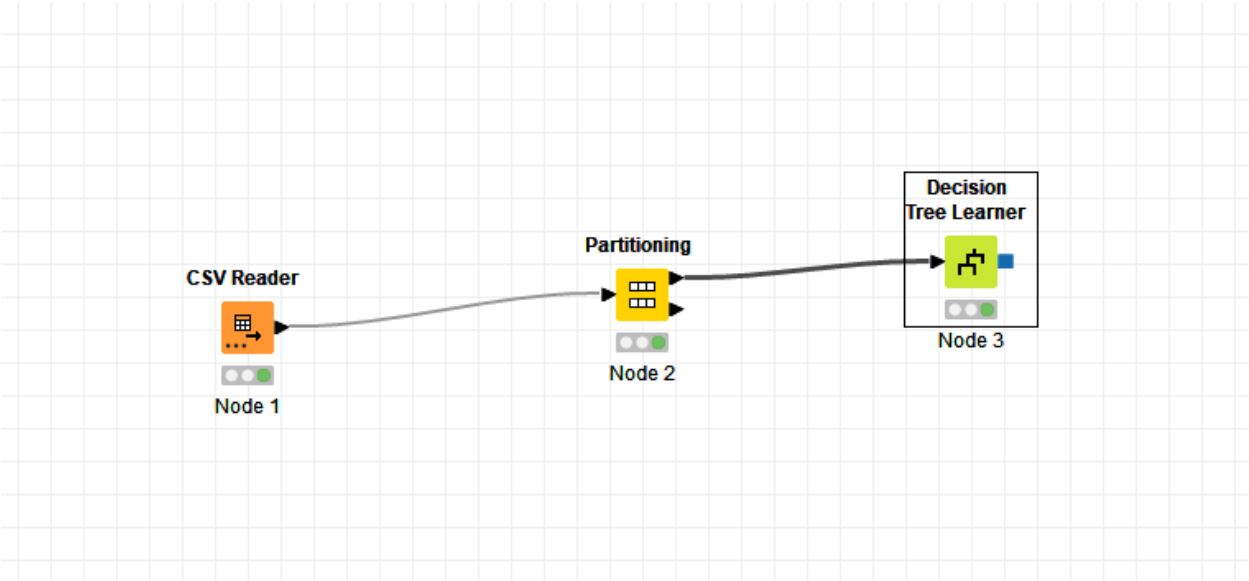
```
C:\WINDOWS\system32\cmd.exe
14 Senior Medium No Excellent No


Number  Attribute
0       RID
1       Age
2       Income
3       Student
4       Credit rating
5       Class: buys computer

Enter Attribute Number whose info gain is to be calculated :4

Total Number of No's :5
Total Number of yes's :9
Total number of entries : 14


<ATTRIBUTE's yes and no count>
Excellent :3 3
Fair :6 2

Class Entropy :0.940286

Attribute Entropy :0.892159

Info Gain :0.048127


Press any key to continue . . .
```

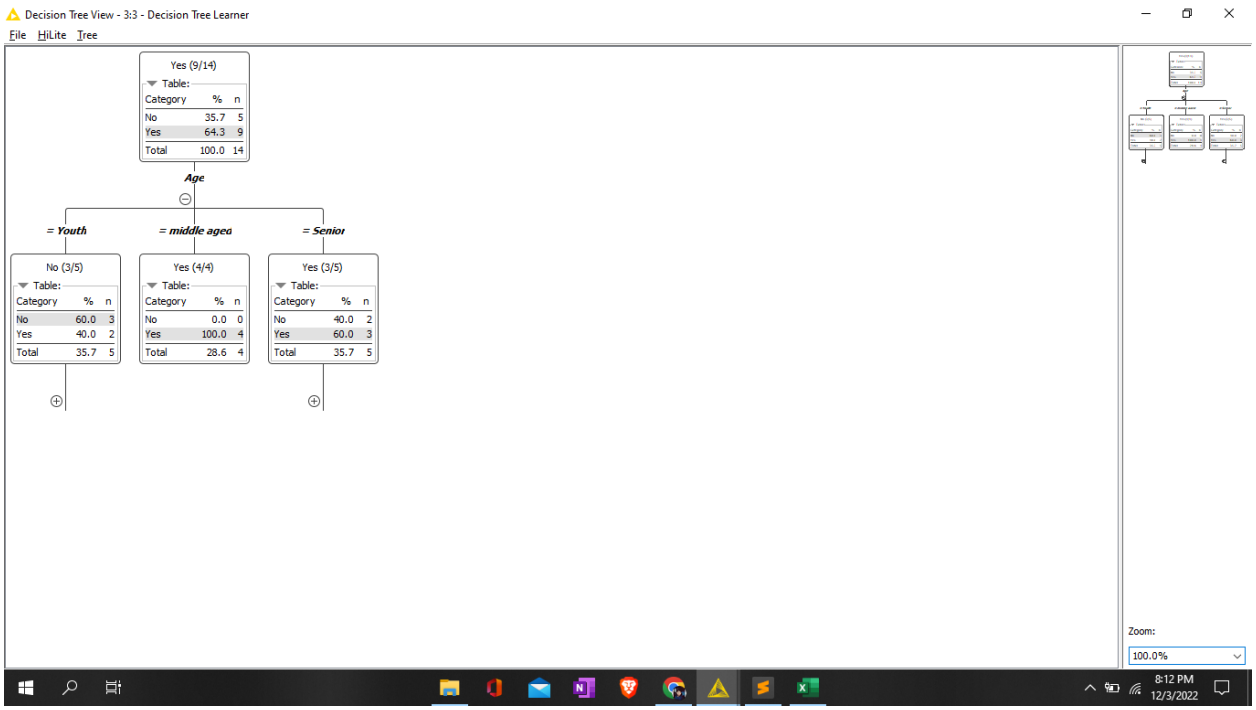
	A	B	C	D
1	Value	Number of yes	Number of No	Entropy
2	Excellent	3	3	1
3	Fair	6	2	0.811278
4				
5				
6	information gain	0.048127		

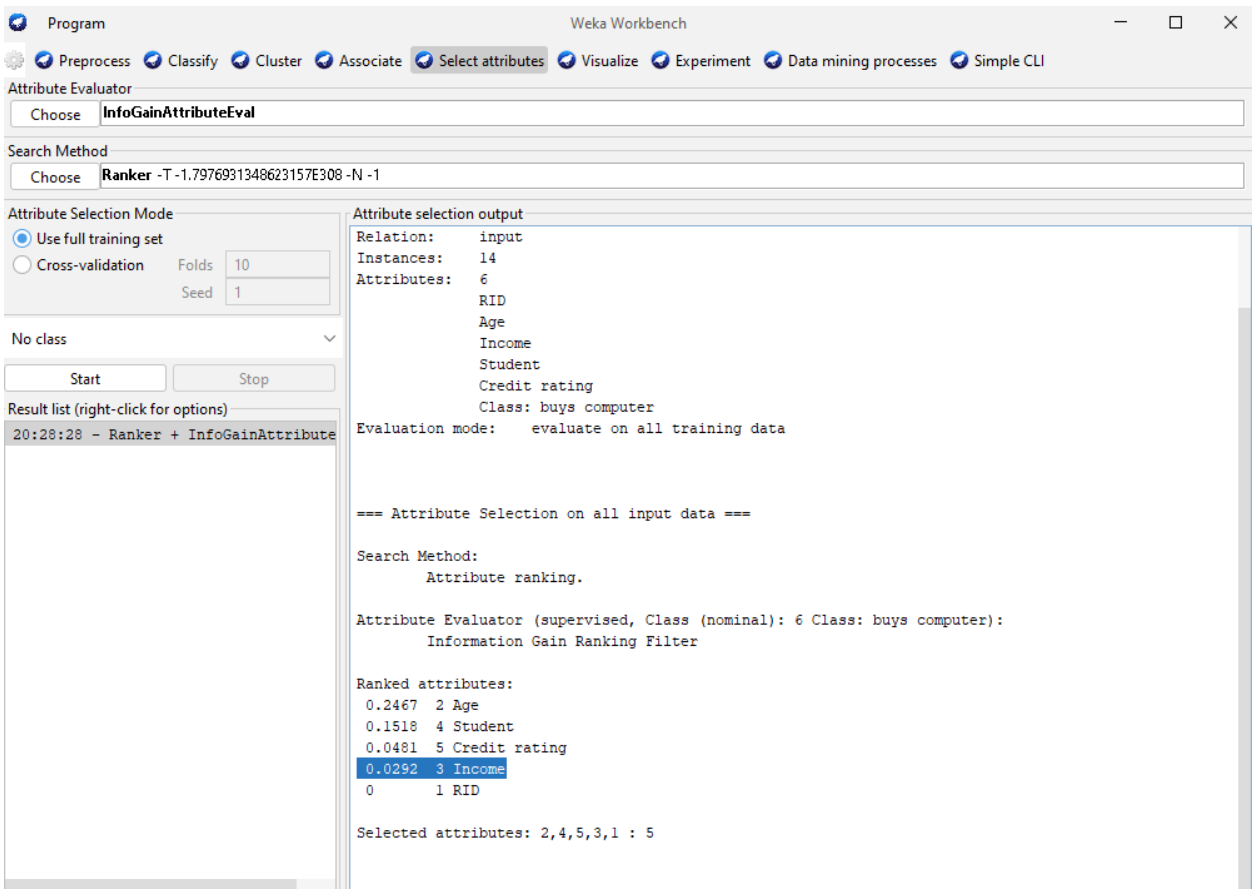
USED DATASET :

	A	B	C	D	E	F
1	RID	Age	Income	Student	Credit rating	Class: buys computer
2	1	Youth	High	No	Fair	No
3	2	Youth	High	No	Excellent	No
4	3	middle aged	High	No	Fair	Yes
5	4	Senior	Medium	No	Fair	Yes
6	5	Senior	Low	Yes	Fair	Yes
7	6	Senior	Low	Yes	Excellent	No
8	7	middle aged	Low	Yes	Excellent	Yes
9	8	Youth	Medium	No	Fair	No
10	9	Youth	Low	Yes	Fair	Yes
11	10	Senior	Medium	Yes	Fair	Yes
12	11	Youth	Medium	Yes	Excellent	Yes
13	12	middle aged	Medium	No	Excellent	Yes
14	13	middle aged	High	Yes	Fair	Yes
15	14	Senior	Medium	No	Excellent	No
16						

KNIME OUTPUT:




Decision Tree View - 3:3 - Decision Tree Learner
File HiLite Tree
Yes (9/14)
Table:
Category % n
No 35.7 5
Yes 64.3 9
Total 100.0 14
Age
= Youth
= middle aged
= Senior
No (3/5)
Table:
Category % n
No 60.0 3
Yes 40.0 2
Total 35.7 5
Yes (4/4)
Table:
Category % n
No 0.0 0
Yes 100.0 4
Total 28.6 4
Yes (3/5)
Table:
Category % n
No 40.0 2
Yes 60.0 3
Total 35.7 5
Zoom:
100.0%
8:12 PM
12/3/2022


Program
Weka Workbench
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Attribute Evaluator
Choose
InfoGainAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds 10
Seed 1
No class
Start
Stop
Result list (right-click for options)
20:28:28 - Ranker + InfoGainAttribute
Attribute selection output
Relation: input
Instances: 14
Attributes: 6
RID
Age
Income
Student
Credit rating
Class: buys computer
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 6 Class: buys computer):
Information Gain Ranking Filter
Ranked attributes:
0.2467 2 Age
0.1518 4 Student
0.0481 5 Credit rating
0.0292 3 Income
0 1 RID
Selected attributes: 2,4,5,3,1 : 5

Expt 4. Find t and d weight of a data

Expt 4:- T and D weight

Theory :-

The d-weight for attribute is the ratio of number of tuples from initial target class working relative that are covered by attribute to the total number of tuples in both initial target class and controling class working relation that are covered by attribute.

T-weight :- within class
D-weight :- across class.

Example:-

	Class	Year	Count
1]	IT	FY SY	50 75
2]	CSE	FY SY	80 120

Solution:

P.T.O.

Class	First Year FY			Second Year SY			Total		
	count	T	D	count	T	D	count	T	D
IT	50	40%	38%	75	60%	38%	125	100%	38%
CSE	80	40%	62%	120	60%	62%	200	100	62%
Total	130	40	100%	195	60%	100%	325	100%	100%

Algo :- Step 1 : Find total count for each attribute.

Step 2 :- for each value find T-weight take row-wise count for value & find %.

Step 3 :- Again for each value. find D weight take column wise count & find percentage

Step 4 :- find T&D weight for total attribute.

Conclusion :-

① D-weight helps us to find the weight of an attribute across the class

② T weight helps us to find weight of attribute within class.

CODE:

```
#include<bits/stdc++.h>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>

using namespace std;

int main(){
string fname="TWeight&DWeightN.csv";
vector<vector<string>> content;
vector<string> row;
string line, word;

fstream file ;
file.open(fname);
if(file.is_open())
{


while(getline(file, line))
{

row.clear();

stringstream str(line);
```

```
while(getline(str, word, ','))
row.push_back(word);
content.push_back(row);
}
}
else
cout<<"Could not open the file\n";
 cout<<endl;

int c1_tot = 0;
int c2_tot = 0;
int r1_tot = 0;
int r2_tot = 0;
for(int i=1;i<content.size();i++)
{
for(int j=1;j<content[i].size();j++)
{
 if(i==1)
  r1_tot+=stoi(content[i][j]);
 if(i==2)
  r2_tot+=stoi(content[i][j]);
 if(j==1)
  c1_tot+=stoi(content[i][j]);
 if(j==2)
  c2_tot+=stoi(content[i][j]);

}

}
```

```
ofstream writeFile("output4.csv");

vector<vector<double> >
tweight(4,vector<double>(4)),dweight(4,vector<double>(4));

for(int i = 1; i < content.size(); i++) {

        for(int j = 1; j < content[i].size(); j++) {
            if(i == 1)
                tweight[i][j]=((double)stoi(content[i][j]) / r1_tot) * 100;
            if(i == 2)
                 tweight[i][j]=((double)stoi(content[i][j]) / r2_tot) * 100;
            if(j == 1)
                dweight[i][j]=((double)stoi(content[i][j]) / c1_tot) * 100;
            if(j == 2)
                dweight[i][j]=((double)stoi(content[i][j]) / c2_tot) * 100;
        }
}

writeFile<<"T-weight"<<"\n";
writeFile<<content[0][0]<<","<<content[0][1]<<","<<content[0][2]<<
"\n";
for(int i=1;i<=2;i++){
    writeFile<<content[i][0]<<",";
    for(int j=1;j<=2;j++)
        writeFile<<tweight[i][j]<<"%"<<",";
    writeFile<<"\n";
}

writeFile<<"\n\n\n";
```

```
writeFile<<"D-weight"<<"\n";
writeFile<<content[0][0]<<","<<content[0][1]<<","<<content[0][2]<<
"\n";
for(int i=1;i<=2;i++){
    writeFile<<content[i][0]<<",";
    for(int j=1;j<=2;j++)
        writeFile<<dweight[i][j]<<"%"<<",";
    writeFile<<"\n";
}
return 0;
}
```

OUTPUT:

	A	B	C	D
1	T-weight			
2	City	FY	SY	
3	IT	40%	60%	
4	CSE	40%	60%	
5				
6				
7				
8	D-weight			
9	City	FY	SY	
10	IT	38.46%	38.46%	
11	CSE	61.54%	61.54%	
12				

Conclusion: Successfully performed t and d weight data analysis in data mining from the table in CSV file

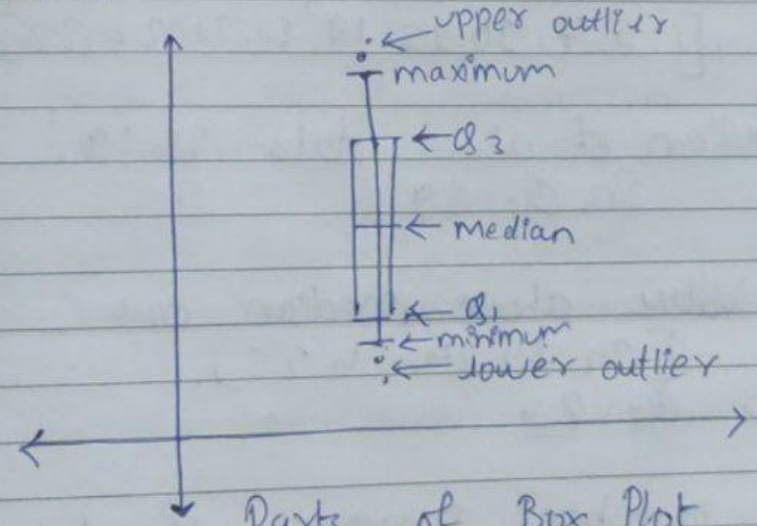
Expt 5 : Find 5 no summary of a dataset (Box Plot)

Expt 5:- Find 5 number summary of a dataset

Box Plot:-

Theory:-

A box plot is also known as whisker plot is created to display the summary of the set of values having properties like minimum, first quartile, median, third quartile and maximum.



Parts of Box Plot

Q_1 → the first quartile or 25% mark
Q_3 → the third quartile or 75% mark
IQR → Inter Quartile Range is difference b/w Q_3 & Q_1

An outlier is data point that lies outside the overall pattern

Upper Outlier = $Q_3 + 1.5 \times IQR$

Lower outlier = $Q_1 - 1.5 \times IQR$

Sample Data :-

data = [5, 7, 10, 15, 19, 21, 21, 22, 22, 23, 23, 23, 23, 23, 24, 24, 24, 24, 25]

Median → As number of element is 19 ie. odd
Median will be 10th value, ie. 23
Median = 23

Q_1 → Values less than median are
[5, 7, 10, 15, 19, 21, 21, 22, 22]
& median of above data is 19.
So, $Q_1 = 19$

Q_3 → Value above median are
[24, 24, 24, 24, 25]
So, $Q_3 = \underline{\underline{24}}$

So, five number summary of data set is
minimum : 5
maximum : 25
median = 23
$Q_1 = 19$
$Q_3 = 24$

$IQR = Q_3 - Q_1 = 24 - 19 = \underline{\underline{5}}$

lower outlier = $Q_1 - 1.5 \times IQR = 19 - 1.5 \times 5 = 11.5$
Upper outlier = $Q_1 + 1.5 \times IQR = 24 + 1.5 \times 5 = 31.5$

CODE :

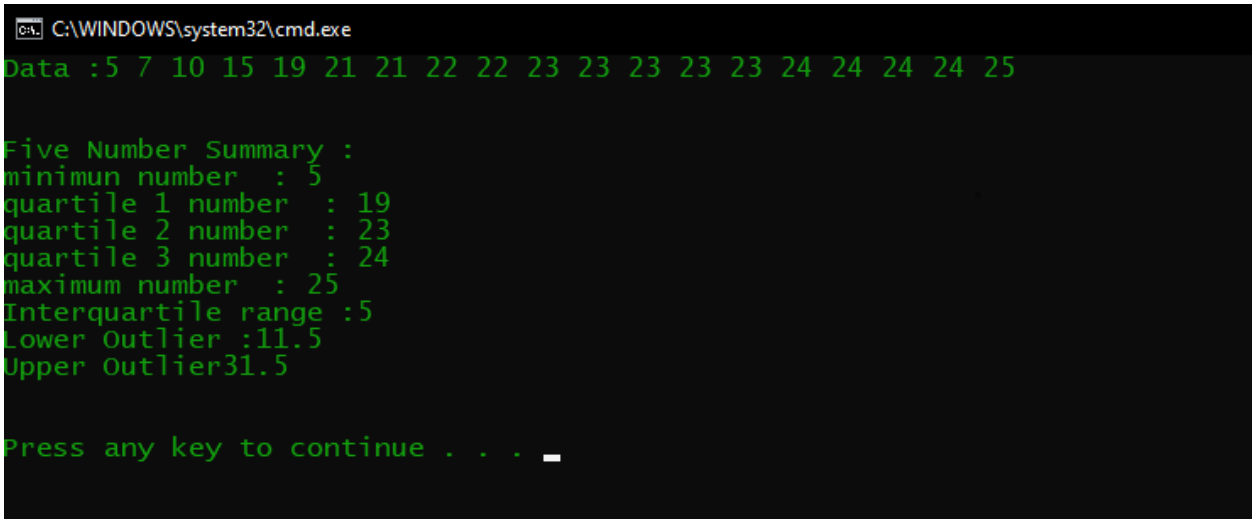
```
#include<bits\stdc++.h>
using namespace std;

double median(vector<int>& a,int i,int j) {
        int n = j-i+1;
    if((n%2)==1)
        return a[(i+j)/2];

    double f1=(double)a[((i+j)/2)]+(double)a[((i+j+1)/2)];
    return f1/2.0;
}

int main(){
    vector<int> v;
    ifstream readFile("input5.csv");
        int min=INT_MAX;
        string marks;
```

```
cout<<"Data :";
   while(readFile.good())
   {
         getline(readFile,marks,'\n');

         stringstream obj;
         int value;
         obj<<marks;
         obj>>value;
         cout<<value<<" ";
         v.push_back(value);
   }
   cout<<endl;
sort(v.begin(),v.end());
int n=v.size();
vector<int>v1;
v1.push_back(v[0]);
v1.push_back(v[n-1]);
int min_number=v[0];
int max_number=v[n-1];
int q1=median(v,0,(n-1)/2-1);
int q2=median(v,0,n-1);
int q3=median(v,(n-1)/2+1,n-1);

cout<<"\n\nFive Number Summary :\n";
cout<<"minimun number  : "<<min_number<<endl;
```

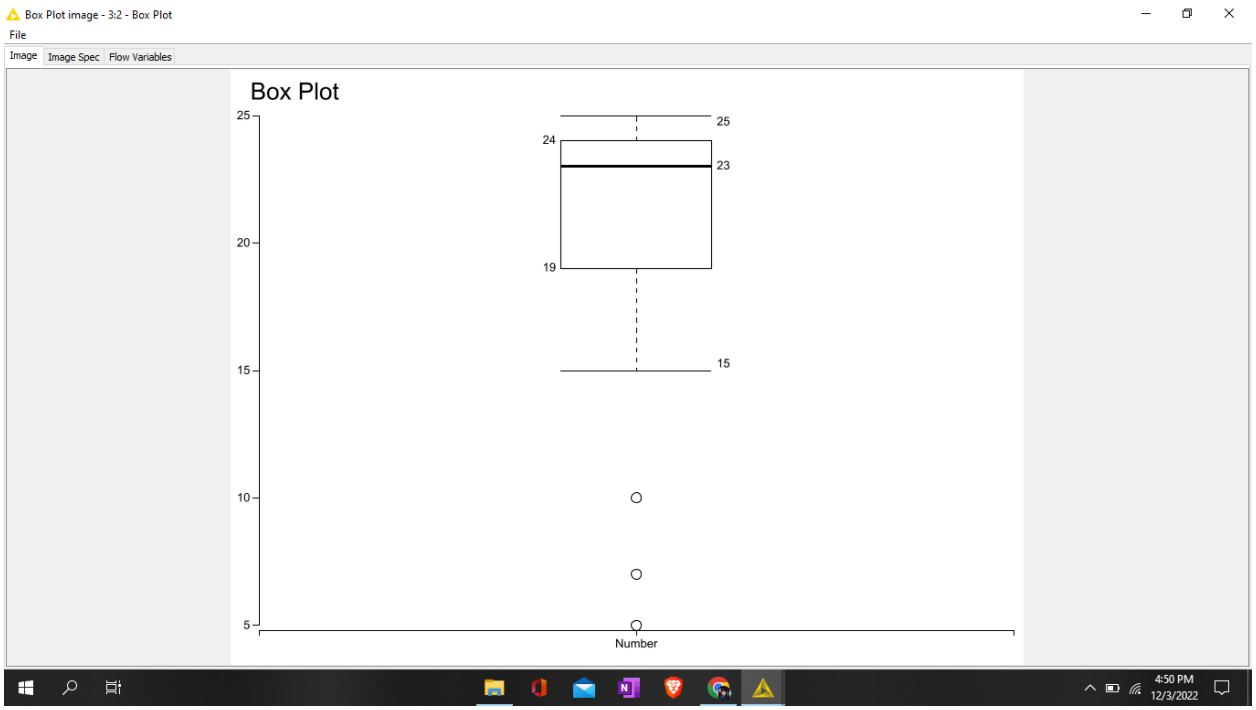
```
    cout<<"quartile 1 number  : "<<q1<<endl;
    cout<<"quartile 2 number  : "<<q2<<endl;
    cout<<"quartile 3 number  : "<<q3<<endl;
    cout<<"maximum number  : "<<max_number<<endl;

    int iqr=q3-q1;

    double lower_outlier=q1-(1.5)*iqr;
    double upper_outlier=q3+(1.5)*iqr;

    cout<<"Interquartile range :"<<iqr<<"\n";
    cout<<"Lower Outlier :"<<lower_outlier<<"\n";
    cout<<"Upper Outlier"<<upper_outlier<<"\n";
    ofstream writeFile("output5.csv");
    writeFile<<"minimun number "<<','<<min_number<<endl;
    writeFile<<"maximum number
"<<','<<max_number<<endl;
    writeFile<<"median "<<','<<q2<<endl;
    writeFile<<"quartile one"<<','<<q1<<endl;
    writeFile<<"quartile three "<<','<<q3<<endl;
    writeFile<<"lower Outlier"<<','<<lower_outlier<<endl;
    writeFile<<"upper Outlier"<<','<<upper_outlier<<endl;

    return 0;
```

}

OUTPUT SS:



KNIME



CONCLUSION :Successfully performed 5 number summary on array of numbers.

Expt 6 : Find frequent itemset from given transaction data.

Experiment 6

Find frequent itemsets from given transaction data

When items are grouped, they from an itemset. A frequent itemset combines elements that often appear together. A frequent itemset is a subset of an itemset that occurs in a dataset with a particular frequency.

Support :- It refers to the popularity of a product in a transaction database a measure of interesting

$$\text{Support} = \frac{\text{transaction comprising a product}}{\text{total transaction}}$$

$$= \text{count}(A \cup B)$$

Confidence :- reference shows the possibilities that the customer bought two items together in a single transaction

$$\text{Confidence} = \frac{\text{support count of both items}}{\text{support count of 1st item}}$$

$$= \frac{\text{sup count}(A \cup B)}{\text{sup count}(A)}$$

+ Maximal itemset :-

If none of an itemset's supersel is frequency the itemset is said to be maximal

* frequent itemset are determined by Apriori FP growth algorithm

* Apriori algorithm is the commonly used frequent itemset mining algorithm. It works well for association rule learning over transactional and relational databases.

* Application of Apriori :- i) Market basket
ii) Cross Marketing
iii) Sale compain analysis

* Algorithm :-
i) Take min-sup as input from user
ii) Read excel file
iii) calculate the min. freq ~~from~~ for dataset
iv) For each level calculate freq. of itemset. Now if frequency of itemset of itemset is greater than minimum frequency
Declare it as frequent itemset & print

Expt 6.

Transaction	Set
1	I1, I2, I3
2	I2, I5
3	I4, I5
4	I1, I2, I5
5	I2, I3, I4

Let min-sup = 40%. ie. 2 transaction

Item	Count
I1	2
I2	4
I3	2
I4	2
I5	3

Frequent 2-item set :

Item	Count
I1, I2	2
I1, I3	1
I1, I4	0
I1, I5	1
I2, I3	2
I2, I4	1
I2, I5	2
I3, I4	1
I3, I5	0
I4, I5	1

Frequent 2 item set

{I2, I3}, {{I1, I2}, {I2, I5}

3 itemset	count
$\{I_1, I_2, I_3\}$	1
$\{I_2, I_3, I_5\}$	0

∴ frequent itemset :- $\{I_2, I_3\}$, $\{I_1, I_2\}$, $\{I_2, I_5\}$

Knime :- gives o/p $\{I_2, I_3\}$, $\{I_1, I_2\}$, $\{I_2, I_5\}$
weka don't show number of frequent itemset at particular level.

Conclusion :-

1. By providing transaction set, we can get frequent item set, so in real world application, we can say that which product is mostly sold.
2. minimum support criteria is used to calculate frequent itemset from list of transaction.

CODE:

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

ifstream fin;
double minfre;
```

```
vector<set<string>> records;
set<string> items;
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 (auto 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 (auto it1 = sets.begin(); it1 != sets.end(); it1++)
   {
      auto 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 main(){
    fin.open("input.csv", ios::in);

    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]);
    records.push_back(tmpset);

    for (auto it = tmpset.begin(); it != tmpset.end(); it++)
    {
        items.insert(*it);
        freq[*it]++;
    }
}
fin.close();
 cout << "No of transactions: " << records.size() << endl;
minfre = minfre * (records.size()-1) / 100;
cout << "Min frequency:" << minfre << endl;

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

while (q.size() > 0)
{
    items.erase(*q.front());
```

```
    q.pop();
}

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

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

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

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

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

        int tot = 0;
        for (int j = 0; j < records.size(); j++)
        {
            bool pres = true;
            for (int k = 0; k < arr.size() && pres; k++)
```

```
            if (records[j].find(arr[k]) == records[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();
}

bool flag = true;

for (auto 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 (auto it = cur.begin(); it != cur.end(); it++)
            cout << "{" << *it << "} " << freq[*it] << endl;

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

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


    for (auto it = prev.begin(); it != prev.end(); it++)
    {


        stringstream ss(*it);
        string word;
        while(getline(ss,word,' '))
            fw<<word<<",";

        fw<<endl;
    }

    return 0;
}
```

DATASET:

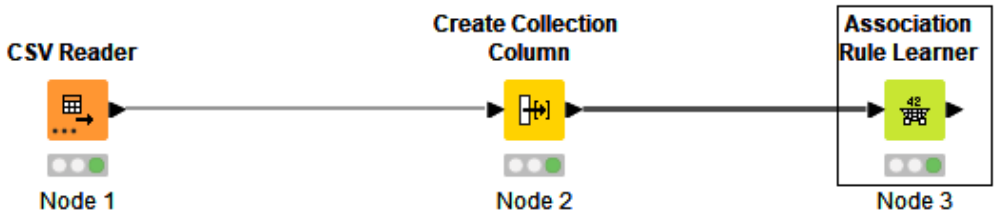
	A	B	C	D
1	I1	I2	I3	
2	I2	I5		
3	I4	I5		
4	I1	I2	I5	
5	I2	I3	I4	
6				
7				

OUTPUT SS:

```
C:\WINDOWS\system32\cmd.exe
Frequency % :40
No of transactions: 6
Min frequency:2

Frequent 1 -item set :
{I1} 2
{I2} 4
{I3} 2
{I4} 2
{I5} 3


Frequent 2 -item set :
{I1 I2} 2
{I2 I3} 2
{I2 I5} 2


Press any key to continue . . .
```

A	B
I1	I2
I2	I3
I2	I5

KNIME :



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

File Edit Hilite Navigation View

Table "default" - Rows: 8 | Spec - Columns: 2 | Properties | Flow Variables

Row ID	D Suppor...	[...] Items
item set 0	0.4	[I1]
item set 1	0.4	[I1,I2]
item set 2	0.8	[I2]
item set 3	0.4	[I2,I3]
item set 4	0.4	[I2,I5]
item set 5	0.4	[I3]
item set 6	0.6	[I5]
item set 7	0.4	[I4]

Expt 7 :Extend program 6, to find association rule.

Expt 7:- Generate Association Rule

Theory:

Apriori algorithm is one of algorithm used for transaction data in association rule learning. It allows us to mine the frequent itemset in order to generate association rule b/w them.

Principle behind apriori algorithm:-

i) Subset of frequent itemset are frequent itemset.

ii) Superset of infrequent itemset are infrequent itemset.

Support $S(I)$ = Number of transaction containing item / total no. of transaction

Confidence : Number of transactions containing I_1 & I_2 / Number of transaction combining I_1

Example :-

Transaction ID	Item purchased
T_1	I_1, I_2, I_3
T_2	I_2, I_5
T_3	I_4, I_5
T_4	I_1, I_2, I_5
T_5	I_2, I_3, I_4

Items	Support count
I_1	2
I_2	4
I_3	2
I_4	1
I_5	3

Step 2: Find all superset with 2 itemset of all items present in last step.

Itemset	support-count
I_1, I_2	2
I_1, I_3	1
I_1, I_5	1
I_2, I_3	2
I_2, I_4	3
I_3, I_5	1

Now, remove all those itemset which has support count less than minimum support count.

Itemset	Support_count
I_1, I_2	2
I_2, I_3	2
I_2, I_5	2

Step 3: In this case if we select $\{I_1, I_2\}$ we must have all subset that is $\{I_1, I_2\}$, $\{I_2, I_3\}$, $\{I_2, I_5\}$

All possible association rules can be

1)	$I_1 \rightarrow I_2$	100%
2)	$I_2 \rightarrow I_3$	50%
3)	$I_2 \rightarrow I_5$	50%
4)	$I_2 \rightarrow I_1$	50%
5)	$I_3 \rightarrow I_2$	100%
6)	$I_5 \rightarrow I_2$	66.67%

So, all these association rules has confidence $\geq 50\%$ then all can be considered as strong association.

Expt 7

Conclusion :-

① There is slight diffrence b/w result of Knime and weka.

② By using this concept, we canvey association b/w products, e.g. If we buy A, how likely B will buy by customer.

CODE :

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

ifstream fin;
double minfre;
vector<set<string>> records;
set<string> items;
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 (auto 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 (auto it1 = sets.begin(); it1 != sets.end(); it1++)
  {
     auto 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 : records)
  {
    bool present = true;

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

```
        }
    }

    if (present)
        count++;
  }

  return count;
}

ofstream fw1("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)
    {
```

```
        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("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]);
    records.push_back(tmpset);

    for (auto it = tmpset.begin(); it != tmpset.end(); it++)
    {
        items.insert(*it);
        freq[*it]++;
    }
}
fin.close();

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

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

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

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

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

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

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

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

        int tot = 0;
        for (int j = 0; j < records.size(); j++)
        {
            bool pres = true;
            for (int k = 0; k < arr.size() && pres; k++)
                if (records[j].find(arr[k]) == records[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();
    }


    bool flag = true;

    for (auto 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 (auto it = cur.begin(); it != cur.end(); it++)
        cout << "{" << *it << "} " << freq[*it] << endl;

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

  generateAssociationRules(prev);

  return 0;
}
```

DATASET :

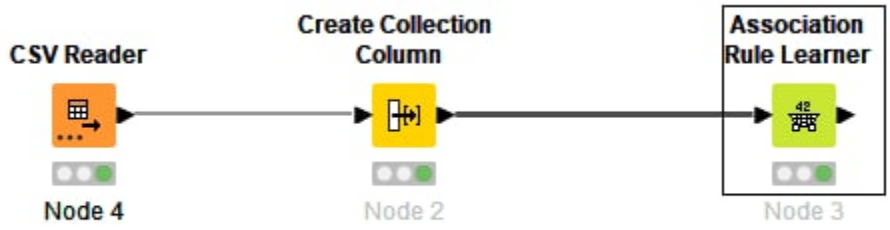
	A	B	C	D
1	I1	I2	I3	
2	I2	I5		
3	I4	I5		
4	I1	I2	I5	
5	I2	I3	I4	
6				
7				

OUTPUT :

J8

	A	B	C	D	E
1	I1	->	I2	100	
2	I2	->	I1	50	
3	I2	->	I3	50	
4	I3	->	I2	100	
5	I2	->	I5	50	
6	I5	->	I2	66.6667	
7					
8					
9					

KNIME :



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

File Edit Hilite Navigation View

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

Row ID	D Support	D Confide...	D Lift	S Conseq...	S implies	[...] Items
rule0	0.4	0.5	1.25	I1	<---	[I2]
rule1	0.4	1	1.25	I2	<---	[I1]
rule2	0.4	1	1.25	I2	<---	[I3]
rule3	0.4	0.5	1.25	I3	<---	[I2]
rule4	0.4	0.667	0.833	I2	<---	[I5]
rule5	0.4	0.5	0.833	I5	<---	[I2]

Expt 8 :Find correlation between items/entities.

Expt 8 : Correlation Coefficient

Correlation is an alternative framework for finding relationships between data itemset

- The data occurance of itemset A is independent of occurance of itemset B if $P(A \cup B) = P(A) \cdot P(B)$
- Otherwise, itemset A and B are independent and correlated as events. This defination can be easily be extended to more than itemset.
- Correlation b/w A&B can be measured by computing $corr_{A,B} = \frac{P(A \cup B)}{P(A) \cdot P(B)}$
- If resulting value of $corr_{A,B}$ is less than 1, then occurrance of A is negatively correlated to B.
- If resulting value of $corr_{A,B}$ is greater than 1, then occurrance of B is positively correlated to B.
- If it is equal to 1, then A&B are independent & there is no relation.

A	1	1	1	1	0	0	0	0
B	1	1	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1

$$corr_{A,B} = \frac{2/8}{4/8 \times 2/8} = 2.0$$

$$corr_{B,C} = \frac{1/8}{7/8 \times 2/8} = \frac{8}{14} = 0.571429$$

$corr_{B,C} = 0.571429$

$corr_{A,C} = \frac{3/8}{4/8 \times 7/8} = \frac{24}{28} = 0.8571$

Conclusion:

① As $corr_{A,B} > 1$, it is positively correlated.

② As $corr_{B,C}$ & $corr_{A,C}$ is negatively correlated.

CODE :

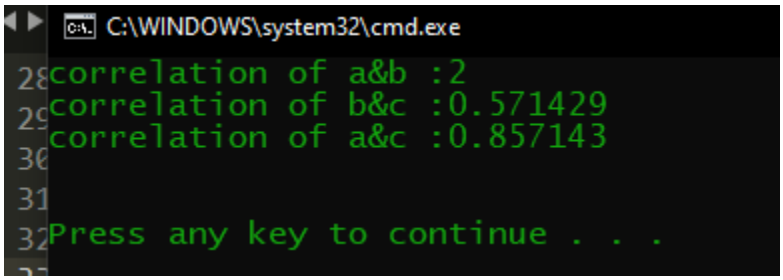
```
#include<bits/stdc++.h>
using namespace std;


double correlation(vector<int>&a,vector<int>&b){
	int prob_aUb=0,prob_a=0,prob_b=0;

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

```
	for(int i=0;i<n;i++){
		if(a[i]==1&&b[i]==1)
			prob_aUb++;

		if(a[i]==1)
			prob_a++;

		if(b[i]==1)
			prob_b++;
	}

	double num=(prob_aUb*1.0)/n;
	double deno=((prob_a*1.0)/n)*((prob_b*1.0)/n);
	// double coef=((prob_aUb*1.0)/)/(prob_a*prob_b);

	return num/deno;
}


int main(){
	vector<int> a,b,c;

	fstream file;
	file.open("input.csv");
	if(file.is_open()){
	string word,line;
	bool isdone=false;
	while(getline(file, line)){
	stringstream str(line);
```

```
        int cnt=0;
        while(getline(str, word, ',')){
                if(cnt==0&&isdone)
                        a.push_back(stoi(word));
                else if(cnt==1&&isdone)
                        b.push_back(stoi(word));
                else if(cnt==2&&isdone)
                        c.push_back(stoi(word));
                cnt++;
        }
        isdone=true;

}
}



cout<<"correlation of a&b :"<<correlation(a,b)<<endl;
cout<<"correlation of b&c :"<<correlation(b,c)<<endl;
cout<<"correlation of a&c :"<<correlation(a,c)<<endl;

ofstream writeFile("output.csv");


writeFile<<"correlation of a&b :"<<","<<correlation(a,b)<<endl;
writeFile<<"correlation of b&c :"<<","<<correlation(b,c)<<endl;
writeFile<<"correlation of a&c :"<<","<<correlation(a,c)<<endl;
return 0;
```

}

OUTPUT SS:



	A	B
1	correlation of a&b :	2
2	correlation of b&c :	0.571429
3	correlation of a&c :	0.857143
4		

Expt 9 : Distance and cluster

Expt 9:-

Distance & Cluster:-

* Algorithm:-

Step 1:- Consider some sample points which are in same cluster.

Step 2:- To find centroid of cluster from given points, find mean of x-coordinated and y-coordinates individually

Step 3:- centroid coordinate will be (x-mean, y-mean)

Step 4:- Compute distance of all given points from this centroid

Step 5:- The point which has a least distance from centroid is assigned as a new centre point

Step 6:- And again distance of each points is calculated from this new centre and store distance in form of upper or lower triangular matrix.

* Examples:-

Points: A(2,1), B(1,4), C(2,2), D(1,1)

Let centroid be 'O'

$$\therefore x_0 = \frac{x_A + x_B + x_C + x_D}{4} = \frac{2+1+2+1}{2}$$

$$x_0 = 1.5$$

$$y_0 = \frac{y_A + y_B + y_C + y_D}{4} = \frac{1+4+2+1}{4} = 2$$

$$O(x_0, y_0) \Rightarrow O(1.5, 2)$$

Distance ⇒ Calculate using Eucledian distance

OA = 1.118 OC = 0.5

OB = 2.061 OD = 1.118

Here, we can see that "C" point is at minimum distance from "O".
Hence, "C" point is considered as new centre point of cluster.

* Distance matrix from "O".

	O	A	B	C	D
O	0	1.1803	2.0615	0.5	1.1803
A	0	0	3.1628	1	1
B	0	0	0	2.2360	3
C	0	0	0	0	1.4142
D	0	0	0	0	0

* ~~Conclusion:-~~
~~Successfully formed the dissimilar matrix using Euclidean distance formula.~~
Now, Here C is closest from O, so, it new center.

Distance matrix from "C".

	C	A	B	C	D
C	0	1	2.23607	0	1.41421
A	0	0	3.16228	1	1
B	0	0	0	2.23607	3
C	0	0	0	0	1.41421
D	0	0	0	0	0

Center of cluster :- (2,2)
Radius of cluster :- 2.23607

Conclusion :- Successfully formed the dissimilar matrix using Euclidean distance formula.

CODE :

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


double distance(double x1,double y1,double x2,double y2) {
   double sq_x = (x2 - x1) * (x2 - x1);
   double sq_y = (y2 - y1) * (y2 - y1);
   return sqrt(sq_x + sq_y);
}
```

```
vector<vector<double>> solve(double new_x,double new_y,double
n,vector<double>&x,vector<double>&y,double&final_radius){

    vector<double> x_cord1;
    x_cord1.push_back(new_x);
    for(double i = 0 ; i < n ; ++i) {
        x_cord1.push_back(x[i]);
    }

    vector<double> y_cord1;
    y_cord1.push_back(new_y);
    for(double i = 0 ; i < n ; ++i) {
        y_cord1.push_back(y[i]);
    }

    vector<vector<double>> res1(n+1,vector<double>(n+1,0));
    for(double i = 0 ; i <= n; ++i) {
        for(double j = i + 1 ; j <= n; ++j) {
            double dis = distance(x_cord1[i],y_cord1[i],x_cord1[j],y_cord1[j]);
            if(i==0&&final_radius<dis)
                final_radius=dis;
            res1[i][j] = dis;
        }
    }
    return res1;

}

double getSum(vector<double>arr) {
    double sum = 0;
```

```
    for(auto&el:arr)
        sum+=el;
    return sum;
}

int main() {
    vector<double> x,y;

    fstream file;
    file.open("input.csv");
    if(file.is_open()){
    string word,line;
    bool isdone=false;
    while(getline(file, line)){
    stringstream str(line);
        double cnt=0;
        while(getline(str, word, ',')){
            if(cnt==1&&isdone)
                x.push_back(stod(word));
            else if(cnt==2&&isdone)
                y.push_back(stod(word));
            cnt++;
        }
        isdone=true;

    }
}
    double sumX = getSum(x);
    double sumY = getSum(y);

    double n=x.size();
```

```
double midX = sumX / n;
double midY = sumY / n;

ofstream writeFile("output.csv");
cout<<"\nCentroid :"<<"("<<midX<<","<<midY<<")\n";
writeFile<<"\nCentroid(C) :"<<","<<midX<<" "<<midY<<"\n";
   writeFile<<"\nmatrix from C\n";
vector<double> x_cord;
x_cord.push_back(midX);
for(double i = 0 ; i < n ; ++i) {
   x_cord.push_back(x[i]);
}

vector<double> y_cord;
y_cord.push_back(midY);
for(double i = 0 ; i < n ; ++i) {
   y_cord.push_back(y[i]);
}

double minD = 1e7+9;;
double new_x,new_y;
int idx;

vector<vector<double>> res(n+1,vector<double>(n+1,0));
for(double i = 0 ; i <= n; ++i) {
   for(double j = i + 1 ; j <= n; ++j) {
      double dis = distance(x_cord[i],y_cord[i],x_cord[j],y_cord[j]);
      res[i][j] = dis;
      if(i==0&&dis < minD) {
         minD = dis;
         new_x=x_cord[j];
```

```
            new_y=y_cord[j];
            idx=j;
        }
    }
}



for(double i = 0 ; i <= n ; ++i) {
    for(double j = 0 ; j <= n ; ++j) {
        cout << res[i][j] << ' ';
    }
    cout << endl;
}

writeFile<<" "<<","<<"C"<<",";
for(int i=1;i<=n;i++)
    writeFile<<"P"<<i<<",";
writeFile<<"\n";
for(double i = 0 ; i <= n ; ++i) {
    if(i==0) writeFile<<"C "<<",";
    else writeFile<<"P"<<(i)<<",";
    for(double j = 0 ; j <= n ; ++j) {
        writeFile << res[i][j] << ',';
    }
    writeFile << endl;
}

cout<<"\n\nNew Coordinate :";
cout<<new_x<<" "<<new_y<<"\n";
```

```
writeFile<<"\n\nMatrix from :("<<new_x<<" "<<new_y<<")\n\n";
double final_radius=0;
vector<vector<double>> res1=solve(new_x,new_y,n,x,y,final_radius);



for(double i = 0 ; i <= n ; ++i) {
    for(double j = 0 ; j <= n ; ++j) {
        cout << res1[i][j] << ' ';
    }
    cout << endl;
}
 writeFile<<" "<<","<<"P"<<idx<<",";
 for(int i=1;i<=n;i++)
    writeFile<<"P"<<i<<",";
writeFile<<"\n";

  for(double i = 0 ; i <= n ; ++i) {
    if(i==0)
        writeFile<<"P"<<idx<<",";
    else
        writeFile<<"P"<<(i)<<",";


    for(double j = 0 ; j <= n ; ++j) {
        writeFile << res1[i][j] << ',';
    }
    writeFile << endl;
}
```

```
    cout<<"\n\nCENTER of cluster :"<<"("<<new_x<<" "<<new_y<<")\n";
    cout<<"RADIUS of cluster :"<<final_radius;


    writeFile<<"\n\nCENTER of cluster :"<<"("<<new_x<<"
"<<new_y<<")\n";
    writeFile<<"RADIUS of cluster :"<<final_radius;



    return 0;
}
```

INPUT DATASET :

points	x	y
1	2	1
2	1	4
3	2	2
4	1	1

OUTPUT :

```
C:\WINDOWS\system32\cmd.exe

Centroid :(1.5,2)
0 1.11803 2.06155 0.5 1.11803
0 0 3.16228 1 1
0 0 0 2.23607 3
0 0 0 0 1.41421
0 0 0 0 0


New Coordinate :2 2
0 1 2.23607 0 1.41421
0 0 3.16228 1 1
0 0 0 2.23607 3
0 0 0 0 1.41421
0 0 0 0 0


CENTER of cluster :(2 2)
RADIUS of cluster :2.23607

Press any key to continue . . .
```

	A	B	C	D	E	F	G	H
4	matrix from C							
5		C	P1	P2	P3	P4		
6	C	0	1.11803	2.06155	0.5	1.11803		
7	P1	0	0	3.16228	1	1		
8	P2	0	0	0	2.23607	3		
9	P3	0	0	0	0	1.41421		
10	P4	0	0	0	0	0		
11								
12								
13	Matrix from :(2 2)							
14								
15		P3	P1	P2	P3	P4		
16	P3	0	1	2.23607	0	1.41421		
17	P1	0	0	3.16228	1	1		
18	P2	0	0	0	2.23607	3		
19	P3	0	0	0	0	1.41421		
20	P4	0	0	0	0	0		
21								
22								
23	CENTER of cluster :(2 2)							
24	RADIUS of cluster :2.23607							

Expt 10:Write a program for Agglomerative Hierarchical clustering using single linkage method

Experiment 10:-

Title: Agglomerative heirarchical clustering using single linkage method

Theory:-

A Heirarchical clustering method works via grouping data into a tree of clusters. Heirarchical clustering begins by treating every data points as separate. Then it repeatedly executes the subsequent steps. Identify the cluster which can be closed & merge. The maximum comparable clusters we need to continue these steps until all the cluster are merged together. In heirarchical clustering, the aim is to produce a heirarchical series of nested cluster. The basic method to generate heirarchical clustering are:-

1. Agglomerative clustering:-

Initially consider all data points as an individual cluster & at every step merge nearest pairs of cluster. At first every dataset is considered as entity or cluster. At every iteration, the cluster merge with different cluster until one cluster is formed

Algorithm:-

① Calculate the similarity of one cluster with all the other cluster

② Consider every data point as individual cluster

③ Merge the cluster which are highly similar to close to each other.
④ Recalculate the proximate matrix for each cluster
⑤ Repeat step ③ & ④ until only a single cluster remains

Problems:- Points

	x	y
A	7	8
B	4	1
C	3	4
D	2	7
E	14	10

~~New~~ Distance matrix

	A	B	C	D	E
A	0				
B	10	0			
C	8	$\boxed{4}$	0		
D	6	8	4	0	
E	9	14	17	15	0

Minimum = 4, merge $\boxed{BC}$

	A	D	E	(B, C)
A	0			
D	6	0		
E	9	15	0	
(B,C)	8	$\boxed{4}$	17	0

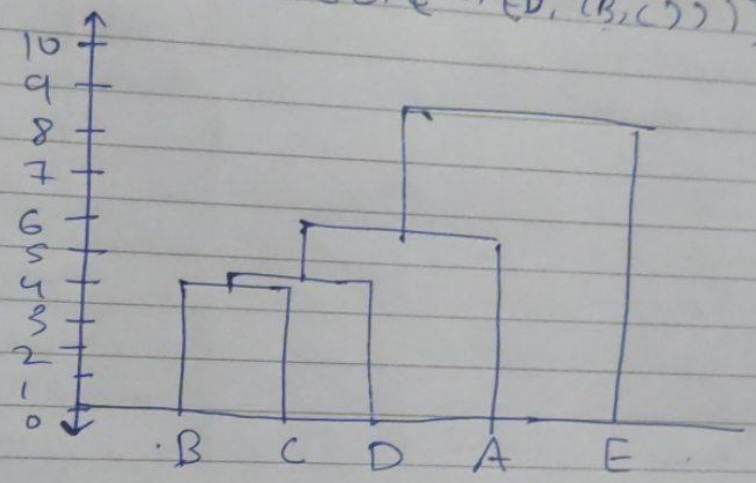
merge D & (B,C)

	A	E	(D,(B,C))
A	0		
E	9	0	0
(D,(B,C))	$\boxed{6}$	15	

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

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

Final cluster:- (E, (A, (D, (B, C))))



Conclusion :-

Thus, agglomerative clustering helps to analyse the grouping of data sharing common characteristics.

CODE:

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

```
void agglo(int itr, int n, int tot_itr, vector<vector<int>> dist,
vector<string> header)
{
	if (itr == tot_itr)
	{
		cout << "\nFinal Cluster: " << header[0] << "\n";
		return;
	}

	cout << "\n"
		 << itr << "th iteration:----------------------------------------\n";
	if (itr != 1)
		cout << "New ";
	cout << "Distance Matrix:" << endl;

	for (int i = 0; i < header.size() - 1; i++)
		printf("%5s", header[i].c_str());
	cout << "  " << header[header.size() - 1] << endl;

	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j <= i; j++)
			printf("%5d", dist[i][j]);
		cout << endl;
	}
```

```
int m = INT_MAX, row, column;
for (int i = 1; i < n; i++)
{
	for (int j = 0; j < i; j++)
		if (dist[i][j] < m)
		{
			m = dist[i][j];
			row = i;
			column = j;
		}
}
cout << "\nMinimum: " << m << endl;
vector<vector<int>> new_dist_matrix;
vector<int> cd;
for (int i = 0; i < n; i++)
{
	vector<int> r;
	m = INT_MAX;
	for (int j = 0; j < n; j++)
	{
		if (i != row && i != column)
		{
			m = min(dist[i][row], dist[i][column]);
			if (j != row && j != column)
				r.push_back(dist[i][j]);
		}
	}
```

```
        if (m != INT_MAX)
        {
                cd.push_back(m);
                r.push_back(m);
        }

        if (!r.empty())
                new_dist_matrix.push_back(r);
}
cd.push_back(0);
new_dist_matrix.push_back(cd);

string str = "(";
if (row < column)
{
        str += header[row] + ", " + header[column] + ")";
}
else
{
        str += header[column] + ", " + header[row] + ")";
}

header.erase(header.begin() + row);
header.erase(header.begin() + column);
header.push_back(str);
```

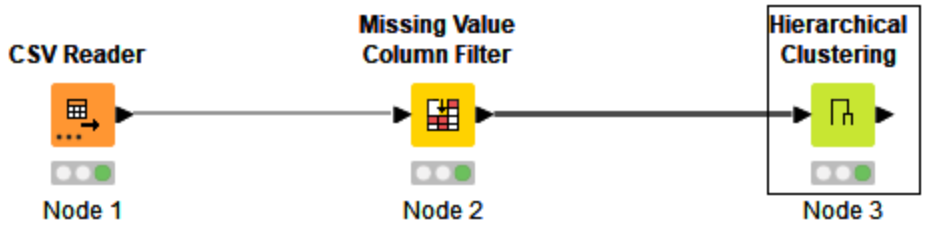
```
	agglo(++itr, --n, tot_itr, new_dist_matrix, header);
}

vector<vector<int>> dist_matrix(vector<int> Ax, vector<int>
Ay)
{
	vector<vector<int>> res;
	for (int i = 0; i < Ax.size(); i++)
	{
		vector<int> row;
		for (int j = 0; j < Ax.size(); j++)
			row.push_back(abs(Ax[i] - Ax[j]) + abs(Ay[i] -
Ay[j]));
		res.push_back(row);
	}
	return res;
}

int main()
{
	fstream fin("input.csv", ios::in);
	vector<vector<string>> content;
	vector<string> row;
	string line = "", word = "";

	if (fin.is_open())
	{
```

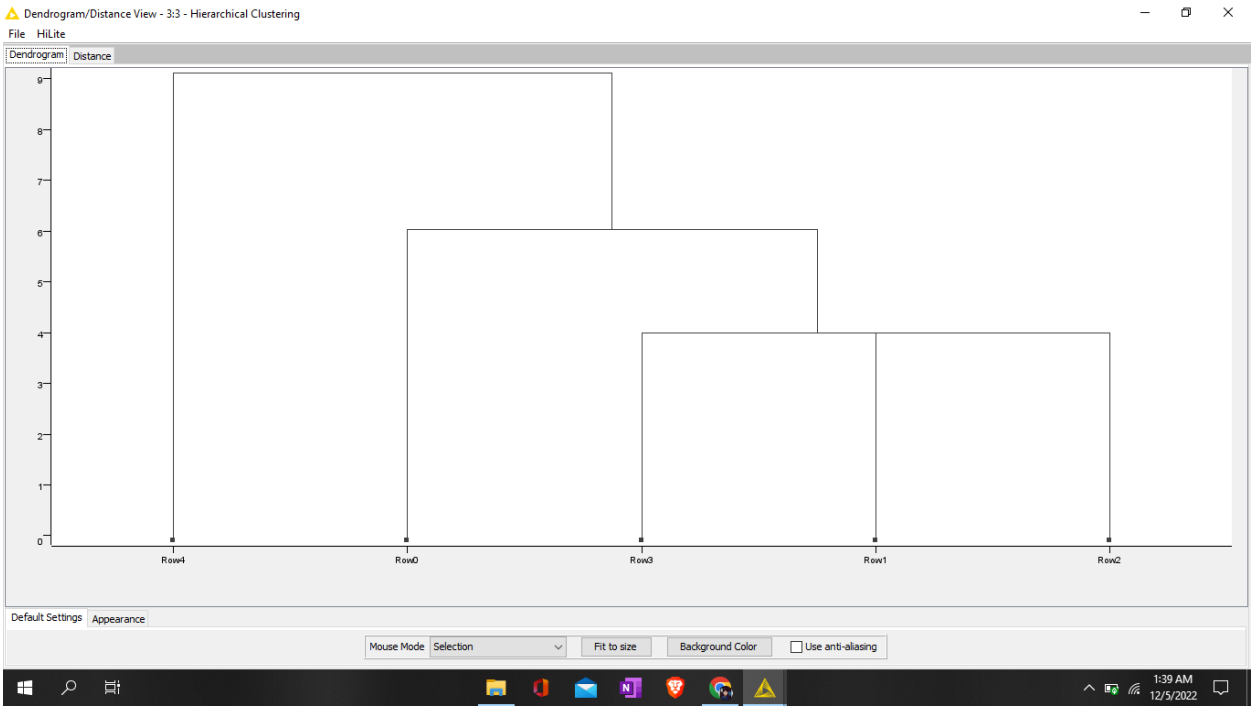
```
        while (getline(fin, line))
        {
                row.clear();
                stringstream s(line);
                while (getline(s, word, ','))
                {
                        row.push_back(word);
                }
                content.push_back(row);
        }
}

vector<int> Ax, Ay;
for (int i = 1; i < content.size(); i++)
{
        Ax.push_back(stoi(content[i][0]));
        Ay.push_back(stoi(content[i][1]));
}

vector<string> header;
char c = 'A';
for (int i = 0; i < Ax.size(); i++)
{
        string str(1, c + i);
        header.push_back(str);
}
```

```
    vector<vector<int>> d = dist_matrix(Ax, Ay);
    agglo(1, Ax.size(), Ax.size(), d, header);
    fin.close();
}
```

OUTPUT:

```
C:\WINDOWS\system32\cmd.exe

1th iteration:----------------------------------------
Distance Matrix:
    A    B    C    D    E
    0
   10    0
    8    4    0
    6    8    4    0
    9   19   17   15    0

Minimum: 4

2th iteration:----------------------------------------
New Distance Matrix:
    A    D    E (B, C)
    0
    6    0
    9   15    0
    8    4   17    0

Minimum: 4

3th iteration:----------------------------------------
New Distance Matrix:
    A    E      (D, (B, C))
    0
    9    0
    6   15    0

Minimum: 6

4th iteration:----------------------------------------
New Distance Matrix:
    E   (A, (D, (B, C)))
    0
    9    0

Minimum: 9

Final Cluster: (E, (A, (D, (B, C))))
```





Expt 11 :Attribute for classification .Write a program to find Gini index For categorical and numerical values

Expt 11

Title: Write a program to verify Gini Index for numerical & categorical data

Theory:-

Gini Index also known as gini impurity calculates the amount of probability of a specific features that is classified in correctly when selected randomly. If all the elements are linked with single class it can be called as pure. Gini Index varies b/w 0 and 1. 0 expresses purity of classification and 1 indicates random distribution of elements across various classes. The values of 0's shows equal distribution. Gini Index can be expressed by formula

Gini Index: $1 - \sum_{i=1}^{2} (p_i)^2$ where

$p_i \Rightarrow$ probability of distinct class

Classification & regression tree algorithm deploys the method of the gini Index to originate binary split.

Example:-

Day	Outlook	Temp	Humidity	Wind	Play Tennis
1	Sunny	85	High	weak	No
2	Sunny	80	High	Strong	NO
3	Overcast	83	High	weak	Yes
4	Rain	70	High	weak	Yes
5	Rain	68	Normal	weak	Yes
6	Overcast	65	Normal	Strong	No
7	Sunny	64	Normal	Strong	Yes
8	Sunny	72	High	weak	NO
9	Rain	69	Normal	weak	Yes
10	Rain	75	Normal	weak	Yes
11	Sunny	75	Normal	Strong	Yes
12	Overcast	72	High	Strong	Yes
13	Overcast	81	Normal	weak	Yes
14	Rain	71	High	Strong	No

Humidity	No	Yes
High	4	3
Normal	1	6

$$Gini(Humidity) = \frac{7}{14}\left[1-\left(\frac{4}{7}\right)^2-\left(\frac{3}{7}\right)^2\right]+\frac{7}{14}\left[1-\left(\frac{1}{7}\right)^2-\left(\frac{6}{7}\right)^2\right]$$

$$= 0.367347$$

For temperature: let's the threshold temp be 72. then left ⇒ (values less than or equal 72) & right ⇒ (values greater 72).

temp	No	Yes
left	3	5
right	2	4

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

$$= 0.45833$$

Conclusion:-

Hence, we obtained the selected attribute as ~~outlook~~ Humidity (categorical) & temperature (numerical) ~~as~~ we get gini index

CODE:

```
#include<bits/stdc++.h>
using namespace std;
vector<vector<string>> content;
int n;
double calcEntropy(int y,int n){
   if(y==0||n==0)
   return 0;
   double s_entropy=-(y/(float)(y+n))*(log2(y/(float)(y+n)));
```

```
s_entropy=s_entropy+(-(n/(float)(y+n))*(log2(n/(float)(y+n)
)));
    return s_entropy;
}
```

```
double calcgain(map<string,pair<int,int> >tmp,double
s_entropy,int total){

        for(auto itr=tmp.begin();itr!=tmp.end();++itr){
        double
p_entropy=calcEntropy(itr->second.second,itr->second.firs
t);
        double mul=(itr->second.second+itr->second.first);
        mul=mul/total;
        s_entropy-=((mul)*p_entropy);
        }
```

```
        return s_entropy;
}
double giniIndex(map<string,pair<int,int>>tmp,double
s_entropy,int total){

        double gini=0.0;

    for(auto itr=tmp.begin();itr!=tmp.end();++itr){
     double ans=1;
     // cout<<itr->first<<" :"<<itr->second.first<<"
"<<itr->second.second<<"\n";
     int tot=itr->second.second+itr->second.first;
     ans-=pow(((float)itr->second.second/tot),2);
     ans-=pow(((float)itr->second.first/tot),2);
     gini+=(ans*((float)tot/total));
    }


 return gini;
}

double sqx(int num,int deno){

    return (num*num*1.0)/(deno*deno);
```

```
}

double giniIndexN(map<string,pair<int,int>>tmp,double
s_entropy,int total){


    int inequality;
    cout<<"\nEnter Splitting Criteria :";
    cin>>inequality;

    int a1=0,a2=0,b1=0,b2=0;
    //a for <= && b for >
    //1 for NO and 2 for YES

    for(auto p:tmp){
        if(stoi(p.first)<=inequality){
            if(p.second.first)
                a1+=p.second.first;
            if(p.second.second)
                a2+=p.second.second;
        }else{
            if(p.second.first)
                b1+=p.second.first;
            if(p.second.second)
                b2+=p.second.second;
        }
```

```
    }

    // cout<<a1<<" "<<a2<<"\n"<<b1<<" "<<b2<<"\n";
    double
first_term=((a1+a2)*1.0/n)*(1-sqx(a1,a1+a2)-sqx(a2,a1+a2
));
    double
second_term=((b1+b2)*1.0/n)*(1-sqx(b1,b1+b2)-sqx(b2,b1
+b2));
    return first_term+second_term;
}


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


    string line, word;
    int cnty=0;
    int cntn=0;
    if(fin.is_open()){
        while(getline(fin,line)){
            row.clear();
```

```
        stringstream s(line);
        while(getline(s,word,',')){
                row.push_back(word);
        }
        content.push_back(row);
            }
}


n=content.size()-1;
map<string,map<string,pair<int,int>> >entireData;

for(int i=1;i<content.size();i++)
{

    if(content[i][content[i].size()-1]=="No")
    {
        cntn++;
        for(int j=1;j<content[0].size()-1;j++){
            entireData[content[0][j]][content[i][j]].first+=1;

                    }


    }
    else
```

```
        {
            cnty++;

            for(int j=1;j<content[0].size()-1;j++){
                entireData[content[0][j]][content[i][j]].second+=1;
                        }

        }
    }

    map<int,string> indexing;
    int idx=0;
    for(auto s:entireData){
        indexing[idx]=s.first;idx++;
    }
    double s_entropy=calcEntropy(cnty,cntn);

    cout<<"Number\t"<<"Atttribute\n";
    for(auto p:indexing)
        cout<<p.first<<"\t"<<p.second<<"\n";

    int choice;
    cout<<"\nEnter Atttribute for which GiniIndex is to be
found :";
    cin>>choice;
```

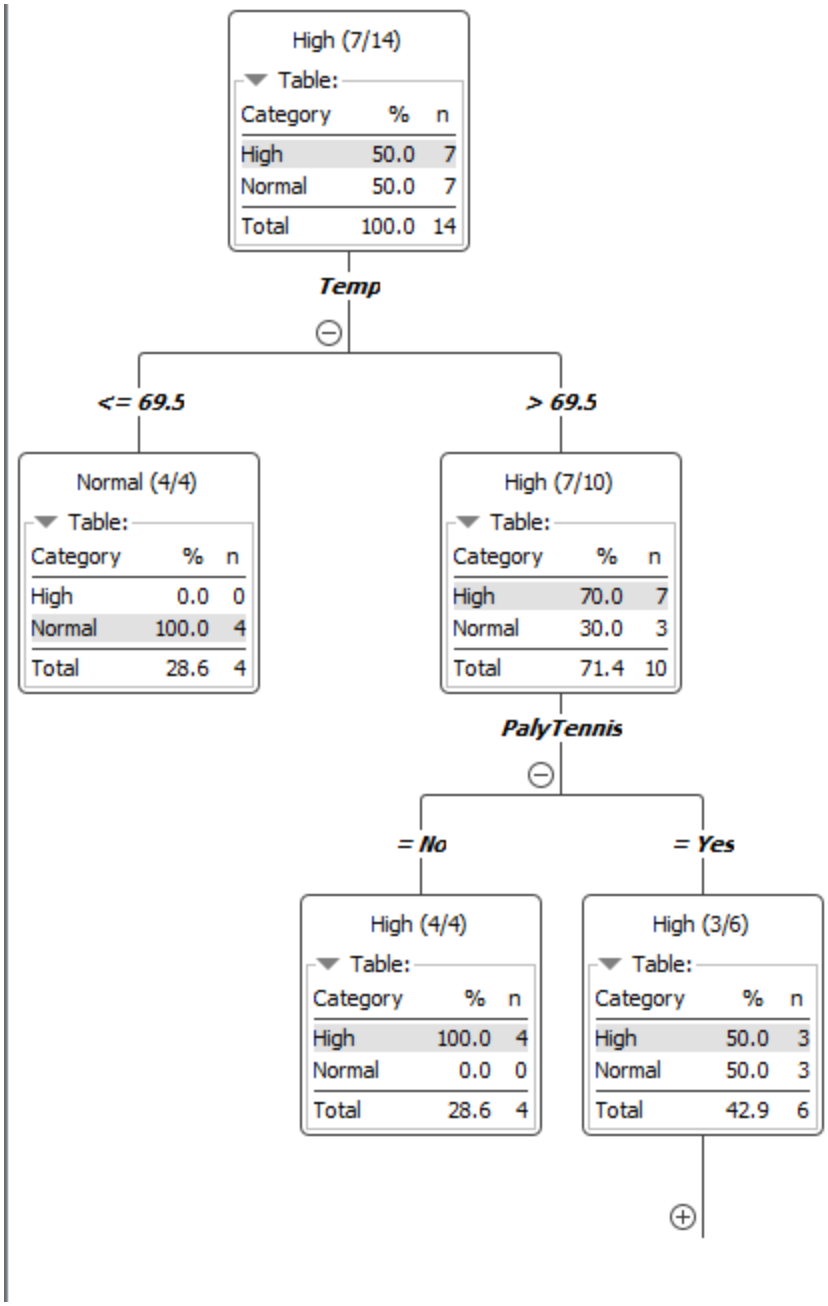
```
   bool isnumerical=false;
   cout<<"Is entered attribute numerical ? :";
   cin>>isnumerical;

   if(isnumerical==0)
   cout<<"GiniIndex for "<<indexing[choice]<<":
"<<giniIndex(entireData[indexing[choice]],s_entropy,conten
t.size()-1)<<"\n";
       else
       cout<<"GiniIndex for "<<indexing[choice]<<":
"<<giniIndexN(entireData[indexing[choice]],s_entropy,conte
nt.size()-1)<<"\n";


   cout<<"\nEnter Atttribute for which GiniIndex is to be
found :";
   cin>>choice;


   cout<<"Is entered attribute numerical ? :";
   cin>>isnumerical;

   if(isnumerical==0)
   cout<<"GiniIndex for "<<indexing[choice]<<":
"<<giniIndex(entireData[indexing[choice]],s_entropy,conten
t.size()-1)<<"\n";
```

```
    else
        cout<<"GiniIndex for "<<indexing[choice]<<":
"<<giniIndexN(entireData[indexing[choice]],s_entropy,conte
nt.size()-1)<<"\n";


        return 0;
}
```

OUTPUT:

```
C:\WINDOWS\system32\cmd.exe
Number   Atttribute
0        Humidity
1        Temp
2        Wind
3        outlook

Enter Atttribute for which GiniIndex is to be found :2
Is entered attribute numerical ? :0
GiniIndex for Wind: 0.428571

Enter Atttribute for which GiniIndex is to be found :1
Is entered attribute numerical ? :1
GiniIndex for Temp:
Enter Splitting Criteria :72
0.458333


Press any key to continue . . .
```

CSV Reader
Node 2
Decision
Tree Learner
Node 3


High (7/14)
Table:
Category % n
High 50.0 7
Normal 50.0 7
Total 100.0 14
Temp
<= 69.5
> 69.5
Normal (4/4)
Table:
Category % n
High 0.0 0
Normal 100.0 4
Total 28.6 4
High (7/10)
Table:
Category % n
High 70.0 7
Normal 30.0 3
Total 71.4 10
PalyTennis
= No
= Yes
High (4/4)
Table:
Category % n
High 100.0 4
Normal 0.0 0
Total 28.6 4
High (3/6)
Table:
Category % n
High 50.0 3
Normal 50.0 3
Total 42.9 6

12.WAP for Baye's classification

Expt No:- 12

* **Title** :- Baye's Classification

* **Theory** :-

Baye's theorem describes the probability of an event based on precedent knowledge of condition which might related to the event. In other words Baye's theorem is the add on of conditional probability.
with the help of conditional probability one confined out probability of X given it and it is denoted by $P(X/H)$
Baye's theorem states that if you know conditional probability, then we can find out $P(H|X)$.
Baye's theorem has 2 types of probability
⇒ Prior Probability $[P(H)]$
⇒ Posterion Probability $[P(H/X)]$

X = data-type
H = Hypothesis.

According to conditional probability
$P(X|H) = [P(X)$ and $P(H)] / P(H)$ — ①

Similarly,
$P(H/X) = [P(H)$ and $P(X)] / P(X)$
$P(X \text{ and } H) = P(H/X) * P(X)$ — ②

From ① and ②
$P(X|H) * P(H) = P(H|X) * P(X)$
$P(X/H) = P(H/X) * P(X) / P(H)$

Example :

Day	outlook	temp	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no

Total Yes = 3 Total No = 3

temp	Yes	No
Hot	1	2
mild	1	0
cool	1	1

outlook	Yes	No
Sunny	0	2
overcast	1	0
rainy	2	1

humidity	Yes	No
High	2	2
normal	1	1

wind	Yes	NO
Strong	0	2
weak	3	1

Condition X : wind = weak
temp = mild
outlook = overcast
humidity = yes

P(weak & Play Yes) = 3/3 = 1
P(weak & Play No) = 1/3 = 0.33

P(High & Yes) = $\frac{2}{5}$ = 0.667

P(High & No) = 2/3 = 0.667

P(mild & Yes) = 1/3 = 0.337

P(mild & No) = 0/3 = 0

P(overcast & Yes) = 1/3 = 0.337

P(overcast & No) = 0/3 = 0

P(x & Yes) = $\frac{3}{6} \times \frac{1}{3} \times 1/3 \times 2/3 \times 3/3$ =

= ~~0.03~~ = 0.037037

P(x & No) = 0

Conclusion :- Baye's classification helps in decision making based on ~~the~~ conditional probability.

Code :

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

void fun(vector<string> ip , vector<vector<string>> data ){
    cout<<ip.size()<<" "<<data[0].size()<<endl;
    if((ip.size()+1)!=data[0].size()){
        cout<<" no proper input given ! "<<endl;
        return ;
    }
    int n = data.size(),m=data[0].size() , ycount=0 , ncount=0;
    double p_yes=0,p_no=0;

    for(int i=1;i<n;i++){
        if(data[i][m-1]=="yes")
            ycount++;
        else
            ncount++;
    }
```

```
cout<<ycount<<" "<<ncount<<endl;
p_yes = (ycount*1.0) / (ycount+ncount);
p_no = (ncount*1.0) / (ycount+ncount);
ofstream myFile("output.csv",ios::out);

for(int i=1;i<ip.size();i++){
    int tmpyc=0,tmpnc=0;
    for(int j=1;j<n;j++){
        if(data[j][i]==ip[i]){
            if(data[j][m-1]=="yes")
                tmpyc++;
            else
                tmpnc++;
        }
    }
    double Py_tmp = (tmpyc*1.0)/ycount;
    double Pn_tmp = (tmpnc*1.0)/ncount;
    myFile<<data[0][i]<<","<<"P(yes),P(no)"<<endl;
    myFile<<ip[i]<<","<<Py_tmp<<","<<Pn_tmp<<endl;
    myFile<<endl;
```

```
        p_yes*=Py_tmp;
        p_no*=Pn_tmp;
    }
    myFile<<endl;
    myFile<<endl<<endl;
    myFile<<"P(yes),"<<p_yes<<endl;
    myFile<<"P(no)"<<","<<p_no<<endl;
    cout<<p_yes<<" "<<p_no<<endl;
}

int main()
{
    vector<vector<string>> input;
    fstream file;
    file.open("input1.csv");
    if(file.is_open()){
        string word,line;
        while(getline(file, line)){
        stringstream str(line);
            double cnt=0;
```

```
        vector<string> tmp;
        while(getline(str, word, ',')){
            tmp.push_back(word);
        }
        input.push_back(tmp);
    }
}
else {
    cout<<" could not open file "<<endl;
    return 0;
}
cout<<"Enter Values for respective values : ";
vector<string> v;
for(int i=0;i<input[0].size()-1;i++){
    cout<<input[0][i]<<"\t";
    string s ;
    cin>>s;
    v.push_back(s);
}
fun(v,input);
```

}

Input :

	A	B	C	D	E	F
1	day	outlook	temp	humidity	wind	play
2	1	sunny	hot	high	weak	no
3	2	sunny	hot	high	strong	no
4	3	overcast	hot	high	weak	yes
5	4	rain	mild	high	weak	yes
6	5	rain	cool	normal	weak	yes
7	6	rain	cool	normal	strong	no
8						

Output :

	A	B	C	D
4	temp	P(yes)	P(no)	
5	mild	0.333333	0	
6				
7	humidity	P(yes)	P(no)	
8	high	0.666667	0.666667	
9				
10	wind	P(yes)	P(no)	
11	weak	1	0.333333	
12				
13				
14				
15				
16	P(yes)	0.037037		
17	P(no)	0		
18				

Using Knime :

Naive Bayes Learner View - 5:11 - Nai...

File

Class counts for play

Class:	no	yes
Count:	3	3

Total count: 6

Threshold to used for zero probabilities: 1.0E-4

Missing values are ignored during learning and prediction phase.

Gaussian distribution for day per class value

	no	yes
Count:	3	3
Mean:	3	4
Std. Deviation:	2.64575	1
Rate:	50%	50%

P(humidity | class=?)

Class/humidity	high	normal
no	2	1
yes	2	1
Rate:	67%	33%

P(outlook | class=?)

Class/outlook	overcast	rain	sunny
no	0	1	2
yes	1	2	0
Rate:	17%	50%	33%

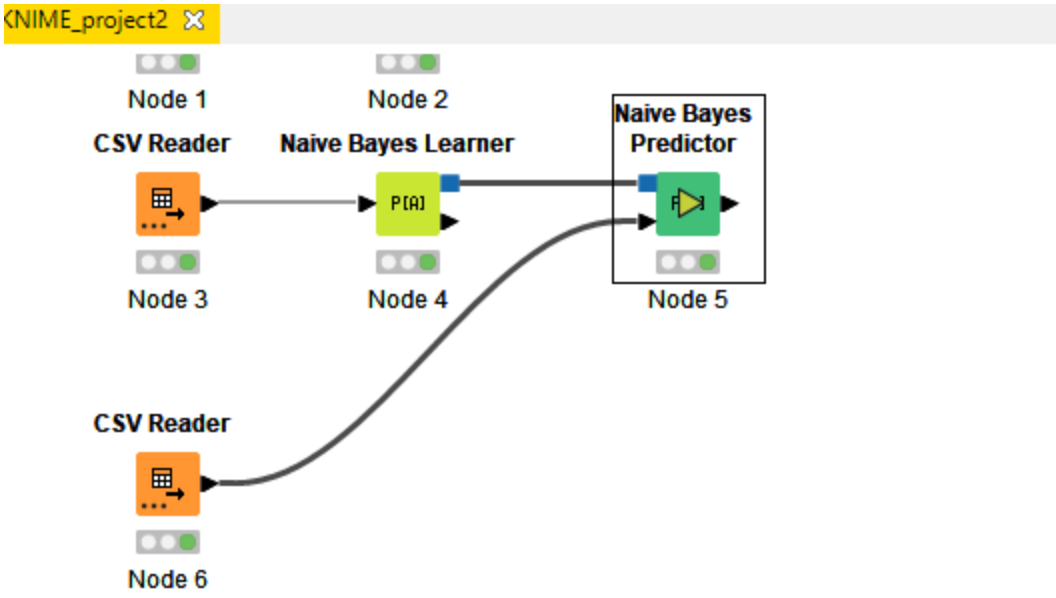
P(temp | class=?)

Class/temp	cool	hot	mild
no	1	2	0
yes	1	1	1
Rate:	33%	50%	17%

P(wind | class=?)

Class/wind	strong	weak
no	2	1
yes	0	3

WORKFLOW:



FOR QUERY:

Table "default" - Rows: 1 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	I day	S outlook	S temp	S humidity	S wind	S Predicti...
Row0	1	overcast	mild	high	weak	yes