INDEX

VISHWAKARMA GOVERNMENT ENGINEERING COLLEGE, CHANDKHEDA

Subject: Data Mining 3160714 (6th Semester)
Faculty Coordinator Prof. K.R.Raval, Prof. J.J.Jadav
Computer Engineering Department

Practical List
Even-2020-2021

No.	Name of Experiment	
1.	Identify how data mining is an interdisciplinary field by an Application.	**CO3**
2.	**Write programs to perform the following tasks (any language).** **2.1 Noisy data handling** 1) Equal Width Binning 2) Equal Frequency /Depth Binning **2.2 Normalization Techniques** 1) Min max normalization 2) Z score normalization 3) Decimal scaling 4)Five Number Summary	**CO1**
3.	To perform hand on experiments of data preprocessing with sample data on Orange tool.	**CO1, CO4**
4.	Implement Apriori algorithm of association rule data mining technique in any Programming language.	**CO2**
5.	Apply association rule data mining technique on sample data sets in XLMiner.	**CO2, CO4**
6.	Apply Classification data mining technique on sample data sets in Weka.	**CO4, CO5**
7.	A) Implement Classification technique with quality Measures in any Programming language B) Implement Regression technique in any Programming language.	**CO5**
8.	Apply Clustering on sample Dataset Using Matlab	**CO2, CO4**
9.	Perform hands on experiment on Web mining tool	**CO4**
10.	Solve Real world problem using Data Mining Techniques.	**CO3**

AIM: Identify how data mining is an interdisciplinary field by an Application.

- Data mining (DM) is an **interdisciplinary field** including various scientific disciplines such as: **database systems, statistics, machine learning, artificial intelligence and the others.** The main goal of data mining is the knowledge exploration from large data sets. The methods of data mining give the possibility to find the unknown regularities in the accumulated data sets. The experimental data from laboratory tests and observations, as well as the data recorded in industrial conditions may possibly be the source of significant information for modelling, control and optimization of the complex processes also in metallurgy. There is the knowledge included in the data sets, and yet it is very often unused because of the difficulties in the analysis of the large data sets. It is very complicated to discover the knowledge from the huge data sets and it is impossible to do it manually. Therefore, there is necessity of the scientific research in the area of both the data analysis and the data mining techniques, which are able to support the analysis of the data in the field of metallurgy and materials science. Data mining is one of the stages of the process of knowledge discovery in databases (KDD). Knowledge discovery process consists of the following steps: data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation, and knowledge presentation. Data mining is the essential stage of the knowledge discovery process, where the intelligent methods are applied in order to extract data patterns.

DATA MINING IN TELECOMMUNICATIONS

INTRODUCTION:

The telecommunications industry generates and stores a tremendous amount of data. These data include call detail data, which describes the calls that traverse the telecommunication networks, network data, which describes the state of the hardware and software components in the network, and customer data, which describes the telecommunication customers. The amount of data is so great that manual analysis of the data is difficult, if not impossible. The need to handle such large volumes of data led to the development of knowledge-based expert systems. These automated systems performed important functions such as identifying fraudulent phone calls and identifying network faults. The problem with this approach is that it is time-consuming to obtain the knowledge from human experts (the "knowledge acquisition bottleneck") and, in many cases, the experts do not have the requisite knowledge. The advent of data mining technology promised solutions to these problems and for this reason the telecommunications industry was an early adopter of data mining technology.

TYPES OF TELECOMMUNICATION DATA:

1. Call Detail Data

Every time a call is placed on a telecommunications network, descriptive information about the call is saved as a call detail record. The number of call detail records that are generated and stored is huge. For example, AT&T long distance customers alone generate over 300 million call detail records per day. Given that several months of call detail data is typically kept online, this means that tens of billions of call detail records will need to be stored at any time. Call detail

records include sufficient information to describe the important characteristics of each call. At a minimum, each call detail record will include the originating and terminating phone numbers, the date and time of the call and the duration of the call. Call detail records are generated in real-time and therefore will be available almost immediately for data mining. This can be contrasted with billing data, which is typically made available only once per month. Call detail records are not used directly for data mining, since the goal of data mining applications is to extract knowledge at the customer level, not at the level of individual phone calls. Thus, the call detail records associated with a customer must be summarized into a single record that describes the customer's calling behavior. The choice of summary variables (i.e., features) is critical in order to obtain a useful description of the customer. Below is a list of features that one might use when generating a summary description of a customer based on the calls they originate and receive over some time period P:

- average call duration
- % no-answer calls
- % calls to/from a different area code
- % of weekday calls (Monday – Friday)
- % of daytime calls (9am – 5pm)
- average # calls received per day
- average # calls originated per day
- # unique area codes called during P

These eight features can be used to build a customer profile. Such a profile has many potential applications. For example, it could be used to distinguish between business and residential customers based on the percentage of weekday and daytime calls. Most of the eight features listed above were generated in a straightforward manner from the underlying data, but some features, such as the eighth feature, required a little more thought and creativity. Because most people call only a few area codes over a reasonably short period of time (e.g., a month), this feature can help identify telemarketers, or telemarketing behavior, since telemarketers will call many different area codes.

2. Network Data

Telecommunication networks are extremely complex configurations of equipment, comprised of thousands of interconnected components. Each network element is capable of generating error and status messages, which leads to a tremendous amount of network data. This data must be stored and analyzed in order to support network management functions, such as fault isolation. This data will minimally include a timestamp, a string that uniquely identifies the hardware or software component generating the message and a code that explains why the message is being generated. For example, such a message might indicate that "controller 7 experienced a loss of power for 30 seconds starting at 10:03 pm on Monday, May 12." Due to the enormous number of network messages generated, technicians cannot possibly handle every message. For this reason expert systems have been developed to automatically analyze these messages and take appropriate action, only involving a technician when a problem cannot be automatically resolved. As described in Section 3, data mining technology is now helping identify network faults by automatically extracting knowledge from the network data. As was the case with the call detail data, network data is also generated in real-time as a data stream and must often be

summarized in order to be useful for data mining. This is sometimes accomplished by applying a time window to the data. For example, such a summary might indicate that a hardware component experienced twelve instances of a power fluctuation in a 10-minute period.

3. Customer Data

Telecommunication companies, like other large businesses, may have millions of customers. By necessity this means maintaining a database of information on these customers. This information will include name and address information and may include other information such as service plan and contract information, credit score, family income and payment history. This information may be supplemented with data from external sources, such as from credit reporting agencies. Because the customer data maintained by telecommunication companies does not substantially differ from that maintained in most other industries, the applications described in Section 3do not focus on this source of data. However, customer data is often used in conjunction with other data in order to improve results. For example, customer data is typically used to supplement call detail data when trying to identify phone fraud.

DATA MINING APPLICATIONS:

The telecommunications industry was an early adopter of data mining technology and therefore many data mining applications exist.

1. Fraud Detection

Fraud is a serious problem for telecommunication companies, leading to billions of dollars in lost revenue each year. Fraud can be divided into two categories: ***subscription*** fraud and ***superimposition*** fraud.

- Subscription fraud occurs when a customer opens an account with the intention of never paying for the account charges.
- Superimposition fraud involves a legitimate account with some legitimate activity, but also includes some "superimposed" illegitimate activity by a person other than the account holder.

Superimposition fraud poses a bigger problem for the telecommunications industry and for this reason we focus on applications for identifying this type of fraud. These applications should ideally operate in real-time using the call detail records and, once fraud is detected or suspected, should trigger some action. This action may be to immediately block the call and/or deactivate the account, or may involve opening an investigation, which will result in a call to the customer to verify the legitimacy of the account activity.

2. Marketing/Customer Profiling

Telecommunication companies maintain a great deal of data about their customers. In addition to the general customer data that most businesses collect, telecommunication companies also store call detail records, which precisely describe the calling behavior of each customer. This information can be used to profile the customers and these profiles can then be used for marketing and/or forecasting purposes.

A serious issue with telecommunication companies is **customer churn**. Customer churn involves a customer leaving one telecommunication company for another. Customer churn is a significant problem because of the associated loss of revenue and the high cost of attracting new customers.

Some of the worst cases of customer churn occurred several years ago when competing long distance companies offered special incentives, typically $50 or $100, for signing up with their company—a practice which led to customers repeatedly switching carriers in order to earn the incentives. Data mining techniques now permit companies the ability to mine historical data in order to predict when a customer is likely to leave. These techniques typically utilize billing data, call detail data, subscription information (calling plan, features, and contract expiration data) and customer information (e.g., age). Based on the induced model, the company can then take action

AIM: Write programs to perform the following tasks (any language).

2.1 Noisy data handling

1) Equal Width Binning
2) Equal Frequency /Depth Binning

2.2 Normalization Techniques

1) Min max normalization
2) Z score normalization
3) Decimal scaling
4) Five Number Summary

```
//================================================================
========
// Name        : DataM.cpp
// Author      : SidPro
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
// step-1: create new configuration
// step-2: change Toolchain
// step-3: build 'Debug' for projrct_name
// step-4: select path in run configuration
// step-5: Run
//================================================================
========

#include <iostream>
#include <algorithm>
#include <tuple>
#include <cmath>
using namespace std;

class lolMath{
	public:
		int n=0;

		lolMath(int length){
			n = length;
		}

	double Mean(int data[]){
		double sum = 0.0;
		for(int i=0;i<n;++i){
			sum+=data[i];
		}
		return sum/n;
```

```
        }


        double Standard_deviation(int data[],double mean){
                double temp;
                double sum = 0.0;
                for(int i=0;i<n;++i){
                        temp = data[i] - mean;
                        sum += temp*temp;
                }
                return sqrt(sum/n);
        }
};

/*-------------------------EQUAL FREQUENCY / DEPTH------------------------ */

tuple<int **,int,int,int> equalfrequency(int data[],int n,int noBins=0){
        int **bins,size,r=0,i;                                          // no. bins X
size
        sort(data,data + n);                                            // sort
        if(noBins == 0)noBins = (data[n - 1] - data[0]) / n;// no. bins
        size = n/noBins;                                                        // size
of bin
        bool see = false;
        if(n%noBins!=0){
                r = n%noBins;
                noBins+=1;
                see = true;
        }
        bins = new int*[noBins];

        int k=0;
        for(i=0;i<noBins;++i){
                bins[i] = new int[size];
                for(int j=0;j<size&&k<n;++j,++k){
                        bins[i][j] = data[k];
                }
        }
        if(see){
                bins[i] = new int[r];
                for(int j=k;j<n;++j)bins[i][j]=data[k];
                for(int j=r;j<size-r;++j)bins[i][j]=0;
        }

        return make_tuple(bins,noBins,size,r);
}
```

```
void equalfrequencyMean(int data[],int n,int row,int col,int r){
	int i;
	if(r>0)row-=1;
	int k=0;
	for(i=0;i<row;++i){
		double sum=0;
		for(int j=0;j<col;++j,++k){
			sum+=data[k];
		}
		sum = sum/col;
		cout<<"Bin"<<(i+1)<<": ";
		for(int j=0;j<col;++j){
			cout<<(int)sum<<" ";
		}
		cout<<"\n";
	}
	if(r>0){
		double sum=0;
		for(int j=k;j<n;++j){
			sum+=data[k];
		}
		sum = sum/r;
		cout<<"Bin"<<(i+1)<<": ";
		for(int j=k;j<n;++j){
			cout<<(int)sum<<" ";
		}
		cout<<"\n";
	}
}

void equalfrequencyBoundary(int **data,int n,int row,int col,int r){
	int i;
	if(r>0)row-=1;
	for(i=0;i<row;++i){
		cout<<"Bin"<<(i+1)<<": ";
		for(int j=0;j<col;++j){
			if(j==0 || j==(col-1))cout<<data[i][j]<<" ";
			else{
				int l= (abs(data[i][0]-data[i][j])<abs(data[i][col-1]-
data[i][j]))?data[i][0]:data[i][col-1];
				cout<<l<<" ";
			}
		}
		cout<<"\n";
	}
```

```
        if(r>0){
                cout<<"Bin"<<(i+1)<<": ";
                for(int j=0;j<r;++j){
                        if(data[i][j]==0)break;
                        if(j==0 || j==(r-1))cout<<data[i][j]<<" ";
                        else{
                                int l= (abs(data[i][0]-data[i][j])<abs(data[i][r-1]-
data[i][j]))?data[i][0]:data[i][r-1];
                                cout<<l<<" ";
                        }
                }
                cout<<"\n";
        }
}

void print2D(int **equalWidth,int row,int col,int size){
        int k=0;
        for(int i=0;i<row;++i){
                cout<<"Bin"<<(i+1)<<": ";
                for(int j=0;j<col&&k<size;++j,++k){
                        cout<<equalWidth[i][j]<<" ";
                }
                cout<<"\n";
        }
}

/*----------------------------EQUAL WIDTH------------------------------- */

void EqualWidth(int data[],int n,int noBins = 0){

        if(noBins == 0)noBins = (data[n - 1] - data[0]) / n;// no. bins

        int W = (data[n-1]-data[0])/noBins;
        int min = data[0];
        int k=0;
        for(int i=0;i<noBins;++i){
                int Bin = min + (i+1)*W;
                cout<<"Bin"<<(i+1)<<": ";
                while(data[k]<Bin && k<n){
                        cout<<data[k]<<" ";
                        ++k;
                }
                cout<<"\n";
        }
}
```

```
/*------------------------- Z-score Normalization -------------------------*/
void Z_score(int data[],int n){
	lolMath l(n);
	double X = l.Mean(data);
	double S = l.Standard_deviation(data,X);
	for(int i=0;i<n;++i){
		cout<<((data[i]-X)/S)<<"\n";
	}
}

/*---------------------- Decimal Scaling Normalization ----------------------*/
void DecimalScale(int data[],int n){
	string s = to_string(data[n-1]);
	int l = s.length();
	double k = pow(10,l);
	for(int i=0;i<n;++i){
		cout<<(data[i]/k)<<"\n";
	}
}

/*------------------------- Min-Max Normalization -------------------------*/
void MinMax(int data[],int n,double NewMin,double NewMax){
	double Min = data[0],Max=data[n-1];
	for(int i=0;i<n;++i){
		cout<<( ((data[i]-Min)/(Max-Min))*(NewMax-NewMin)+NewMin )<<"\n";
	}
}

/*------------------------- Five Number Summary -------------------------*/
void FivenumberSummary(int data[],int n){
	double M,Q1,Q3,IQR;
	int Min = data[0],Max = data[n - 1];
	if(n%2==0){
		M=(data[n/2] + data[n/2 - 1])/2.0;
		Q1 = data[n/4];
		Q3 = data[n/4 + n/2];
		if(n/2 % 2==0){
			Q1 = (data[n/4]+data[n/4 - 1])/2.0;
			Q3 = (data[n/4 + n/2]+data[n/4 + n/2 - 1])/2.0;
		}
	}else{
		M=data[n/2];
		Q1 = (data[n/4]+data[n/4 - 1])/2.0;
		Q3 = (data[n/4 + n/2]+data[n/4 + n/2 + 1])/2.0;
	}
	lolMath l(n);
```

```
        double X = l.Mean(data);
        double S = l.Standard_deviation(data,X);
        IQR = Q3 - Q1;
        bool see =true;
        double q1 = Q1 - 1.5 * IQR;
        double q3 = Q3 + 1.5 * IQR;
        //cout<<"\nThe Count: "<<n;
        //cout<<"\nThe Mean: "<<X;
        //cout<<"\nThe Standard deviation: "<<S;
        cout<<"\nThe Minimum: "<<Min;
        cout<<"\nQ1 (the first quartile or 25%): "<<Q1;
        cout<<"\nThe Median: "<<M;
        cout<<"\nQ3 (the third quartile or 75%): "<<Q3;
        cout<<"\nThe Maximum: "<<Max;
        cout<<"\nOutliers: ";
        for(int i=0;i<n;++i){
                if(data[i]<q1 || data[i]>q3){
                        cout<<data[i]<<" ";
                        see = false;
                }
        }
        if(see)cout<<"0";
}

int main() {
        //int data[] = {1,2,3,4,5,6,7,8,9,11};
        int data[] = {5,10,11,13,15,35,50,55,72,92,204,215};
        //int data[] = {3 ,5 ,7 ,8 ,12 ,14 ,14 ,15 ,18 ,21};
        int size = sizeof(data)/sizeof(data[0]);
        int row,col,**equalDepth,r;
        tie(equalDepth,row,col,r) = equalfrequency(data,size,3);
        cout<<"Equal Width Binning\n";
        EqualWidth(data,size,3);
        cout<<"Equal Frequency /Depth Binning\n";
        print2D(equalDepth,row,col,size);
        cout<<"Equal Frequency /Depth by Means Binning\n";
        equalfrequencyMean(data,size,row,col,r);
        cout<<"Equal Frequency /Depth by Boundary Binning\n";
        equalfrequencyBoundary(equalDepth,size,row,col,r);
        cout<<"Min-Max normalization"<<"\n";
        double Min=0.0,Max=1.0;
        MinMax(data,size,Min,Max);
        cout<<"Z-score normalization"<<"\n";
        Z_score(data,size);
        cout<<"Normalization by decimal scaling"<<"\n";
        DecimalScale(data,size);
```

```
cout<<"five number summary";
FivenumberSummary(data,size);
return 0;}
```

```
Equal Width Binning
Bin1: 5 10 11 13 15 35 50 55 72
Bin2: 92
Bin3: 204
Equal Frequency /Depth Binning
Bin1: 5 10 11 13
Bin2: 15 35 50 55
Bin3: 72 92 204 215
Equal Frequency /Depth by Means Binning
Bin1: 9 9 9 9
Bin2: 38 38 38 38
Bin3: 145 145 145 145
Equal Frequency /Depth by Boundary Binning
Bin1: 5 13 13 13
Bin2: 15 55 55 55
Bin3: 72 72 215 215
Min-Max normalization
0
0.0238095
0.0285714
0.0380952
0.047619
0.142857
0.214286
0.238095
Z-score normalization
-0.855259
-0.783689
-0.769375
-0.740747
-0.712119
-0.42584
-0.211131
-0.139561
0.103776
0.390055
1.99322
2.15067
Normalization by decimal scaling
0.005
0.01
0.011
0.013
0.015
0.035
0.05
0.055
0.072
0.092
0.204
0.215
five number summary
The Minimum: 5
Q1 (the first quartile or 25%): 12
The Median: 42.5
Q3 (the third quartile or 75%): 82
The Maximum: 215
Outliers: 204 215
```

AIM: To perform hand on experiments of data preprocessing with sample data on Orange tool.

AS we know Knowledge discovery as a process is depicted in and consists of an iterative sequence of the following steps:

1. Data cleaning (to remove noise and inconsistent data)
2. Data integration (where multiple data sources may be combined)
3. Data selection (where data relevant to the analysis task are retrieved from the database)
4. Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance)
5. Data mining (an essential process where intelligent methods are applied in order to extract data patterns)
6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures)
7. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user)

- In this experiment we are going to use **iris dataset** which is already available in Orange tools inbuilt datasets.
- For that we will add a file widget and then select the iris dataset. As shown in fig1.

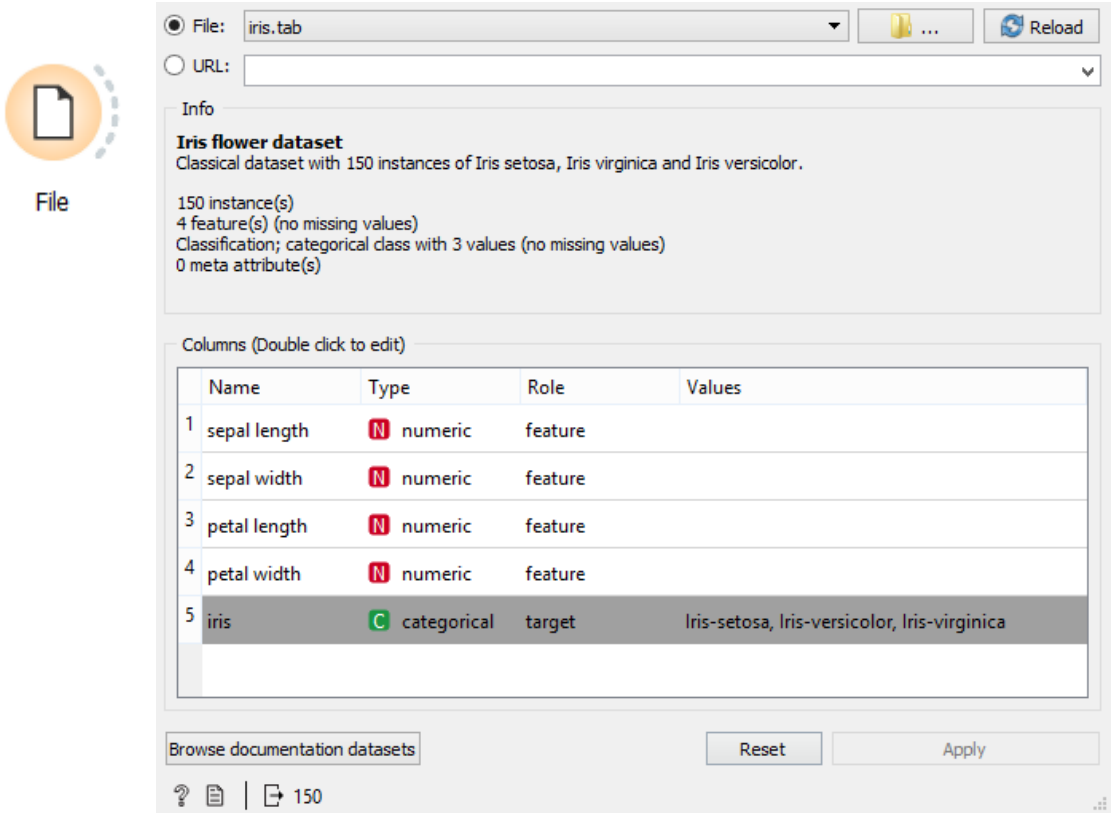
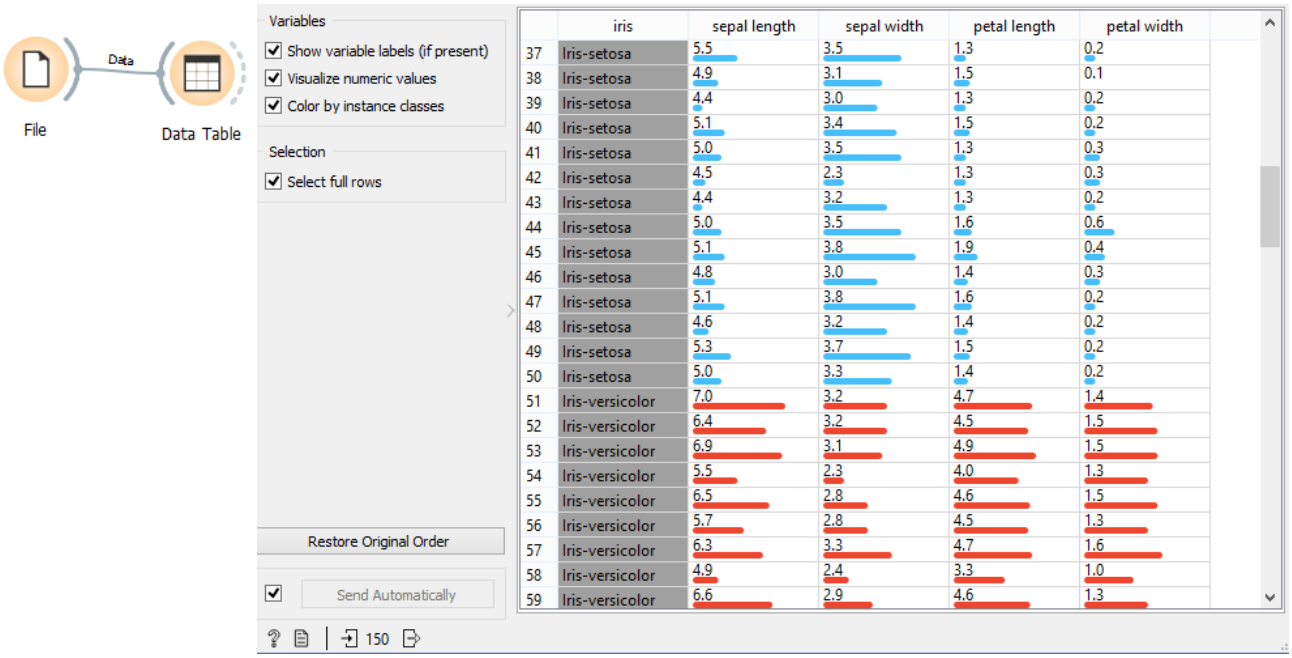


Fig1- Load iris dataset

- Now we want to see the data then it can be done by using data table widget as shown in Fig1.2. It seems that **data is clean** there is no need to remove noise and inconsistent data.



	iris	sepal length	sepal width	petal length	petal width
37	Iris-setosa	5.5	3.5	1.3	0.2
38	Iris-setosa	4.9	3.1	1.5	0.1
39	Iris-setosa	4.4	3.0	1.3	0.2
40	Iris-setosa	5.1	3.4	1.5	0.2
41	Iris-setosa	5.0	3.5	1.3	0.3
42	Iris-setosa	4.5	2.3	1.3	0.3
43	Iris-setosa	4.4	3.2	1.3	0.2
44	Iris-setosa	5.0	3.5	1.6	0.6
45	Iris-setosa	5.1	3.8	1.9	0.4
46	Iris-setosa	4.8	3.0	1.4	0.3
47	Iris-setosa	5.1	3.8	1.6	0.2
48	Iris-setosa	4.6	3.2	1.4	0.2
49	Iris-setosa	5.3	3.7	1.5	0.2
50	Iris-setosa	5.0	3.3	1.4	0.2
51	Iris-versicolor	7.0	3.2	4.7	1.4
52	Iris-versicolor	6.4	3.2	4.5	1.5
53	Iris-versicolor	6.9	3.1	4.9	1.5
54	Iris-versicolor	5.5	2.3	4.0	1.3
55	Iris-versicolor	6.5	2.8	4.6	1.5
56	Iris-versicolor	5.7	2.8	4.5	1.3
57	Iris-versicolor	6.3	3.3	4.7	1.6
58	Iris-versicolor	4.9	2.4	3.3	1.0
59	Iris-versicolor	6.6	2.9	4.6	1.3

Fig1.2 – iris Data Table

- We have successfully identified a few important features from the dataset and would like to create a new feature from existing features. We will **integrate new feature** into existing dataset.
- There are times where there is a need to create a new feature from existing features. For example, you can create a BMI feature using the height and weight features of person. This is usually referred as feature engineering.
- Orange comes with a built-in widget called feature constructor that allows us to create a new feature using existing features as shown in Fig2.
- We will create Two features:
    1) one is $sepal\_length^2$
    2) One will group the iris into small, medium and large based on their sepal length.

    0 if sepal_length < 5
    else 1 if sepal_length < 6
    else 2

    We represents the label based on condition like,
    0: S
    1: M
    2: L

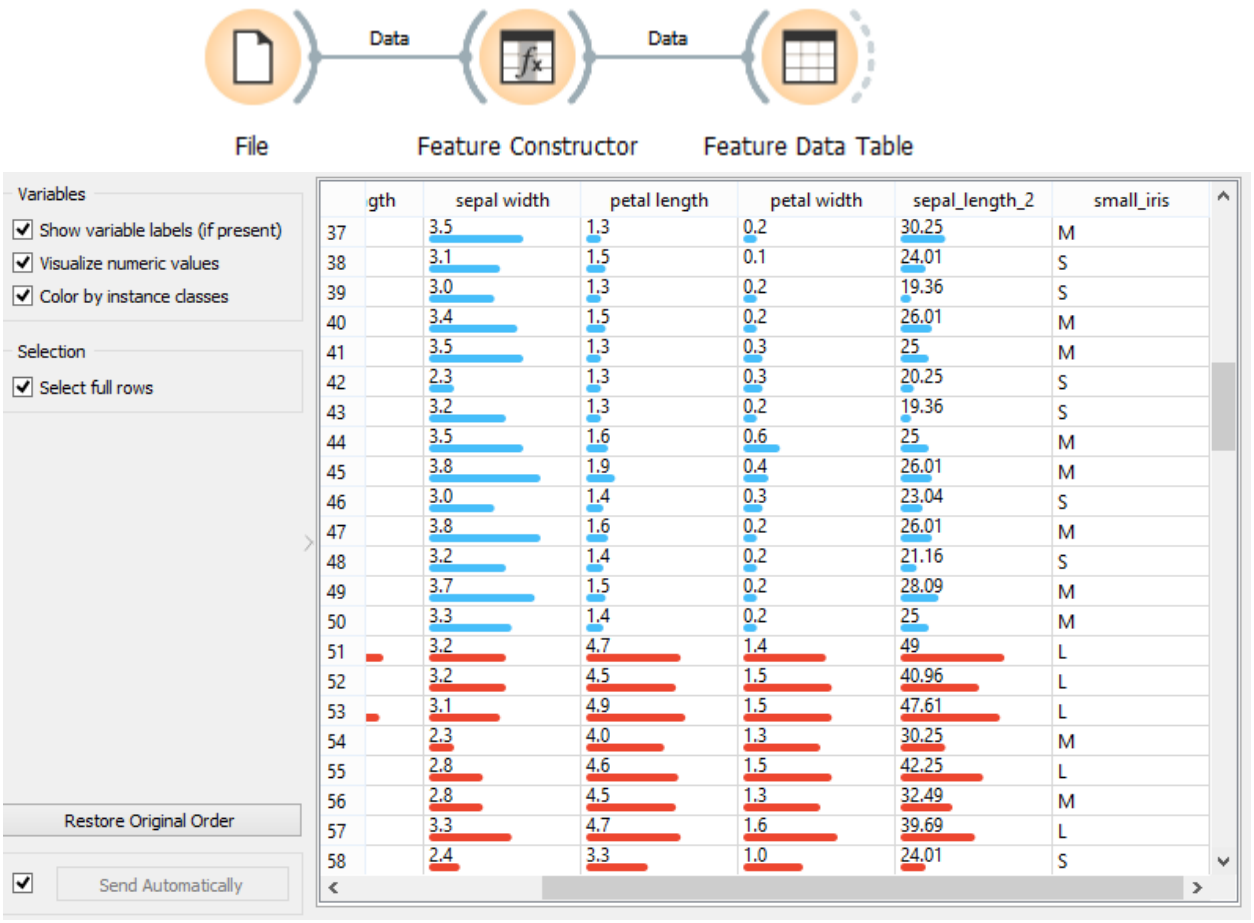


Fig2 – Feature Data Table

- We will **Select data relevant to the analysis task** we can easily do so with the select columns widget. Left represents unwanted features while right represents selected features.
- Orange comes with a built-in widget called select columns.
- We will select columns: sepal width, sepal length, petal width, petal length.

- We can set our conditions using the select rows interface to filter out the desired data. We can filter out data that are not from specific class. We can also set condition for the features such as the sepal length must be greater than certain figure.
- We set condition to select rows that have sepal width is greater than 3 and we are getting 67 instances as shown in Fig3.
- Orange comes with a built-in widget called select rows.

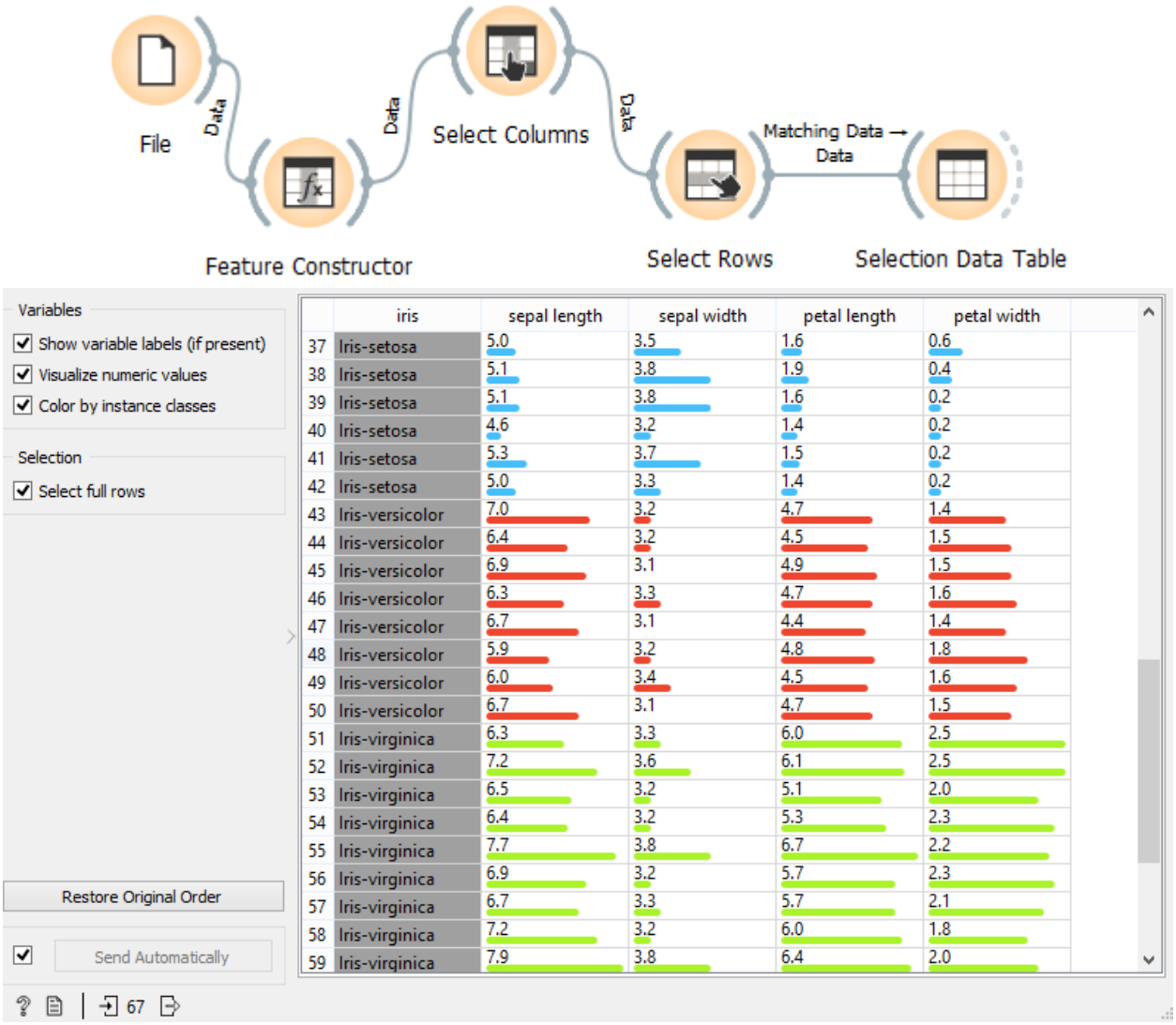
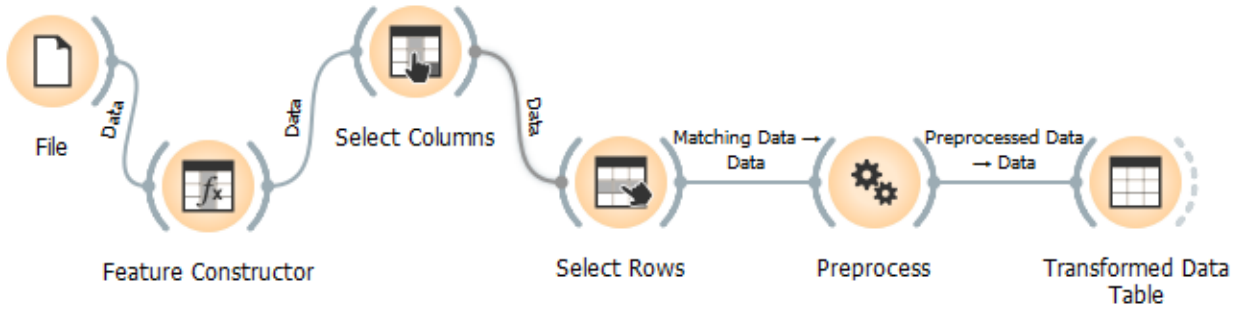


Fig3 – Selection Data Table

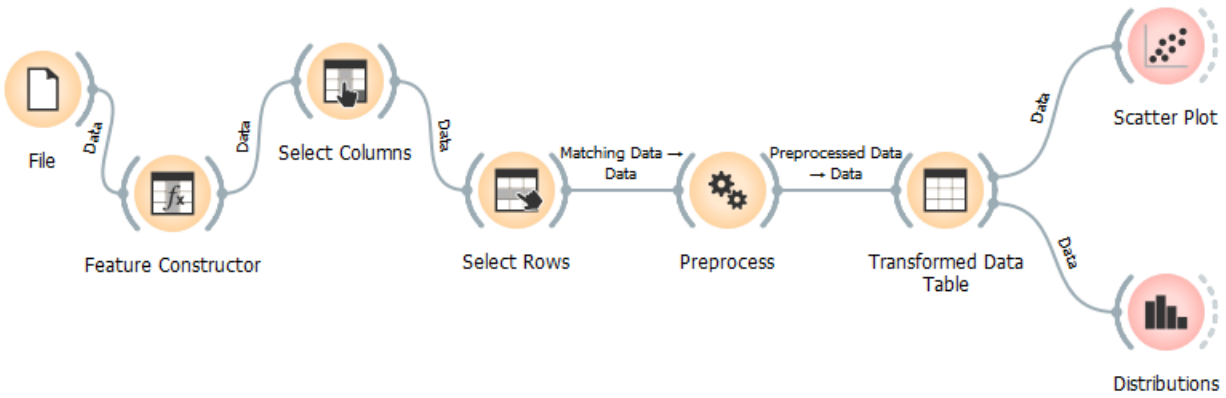
- Now to **transform** the given data we will add a widget called **preprocess** and connect it with data table widget.
- For transforming we will normalize our iris dataset to the interval of [0, 1] as shown in Fig4.



	iris	sepal length	sepal width	petal length	petal width
37	Iris-setosa	0.171429	0.307692	0.105263	0.208333
38	Iris-setosa	0.2	0.538462	0.157895	0.125
39	Iris-setosa	0.2	0.538462	0.105263	0.0416667
40	Iris-setosa	0.0571429	0.0769231	0.0701754	0.0416667
41	Iris-setosa	0.257143	0.461538	0.0877193	0.0416667
42	Iris-setosa	0.171429	0.153846	0.0701754	0.0416667
43	Iris-versicolor	0.742857	0.0769231	0.649123	0.541667
44	Iris-versicolor	0.571429	0.0769231	0.614035	0.583333
45	Iris-versicolor	0.714286	0	0.684211	0.583333
46	Iris-versicolor	0.542857	0.153846	0.649123	0.625
47	Iris-versicolor	0.657143	0	0.596491	0.541667
48	Iris-versicolor	0.428571	0.0769231	0.666667	0.708333
49	Iris-versicolor	0.457143	0.230769	0.614035	0.625
50	Iris-versicolor	0.657143	0	0.649123	0.583333
51	Iris-virginica	0.542857	0.153846	0.877193	1
52	Iris-virginica	0.8	0.384615	0.894737	1
53	Iris-virginica	0.6	0.0769231	0.719298	0.791667
54	Iris-virginica	0.571429	0.0769231	0.754386	0.916667
55	Iris-virginica	0.942857	0.538462	1	0.875
56	Iris-virginica	0.714286	0.0769231	0.824561	0.916667
57	Iris-virginica	0.657143	0.153846	0.824561	0.833333
58	Iris-virginica	0.8	0.0769231	0.877193	0.708333
59	Iris-virginica	1	0.538462	0.947368	0.791667

Fig4 – Normalized Data Table

- With Orange tool we can also visualize our data. It can be done by adding widgets like Scatter plot, Box Plot etc. as shown in Fig5.1, Fig5.2



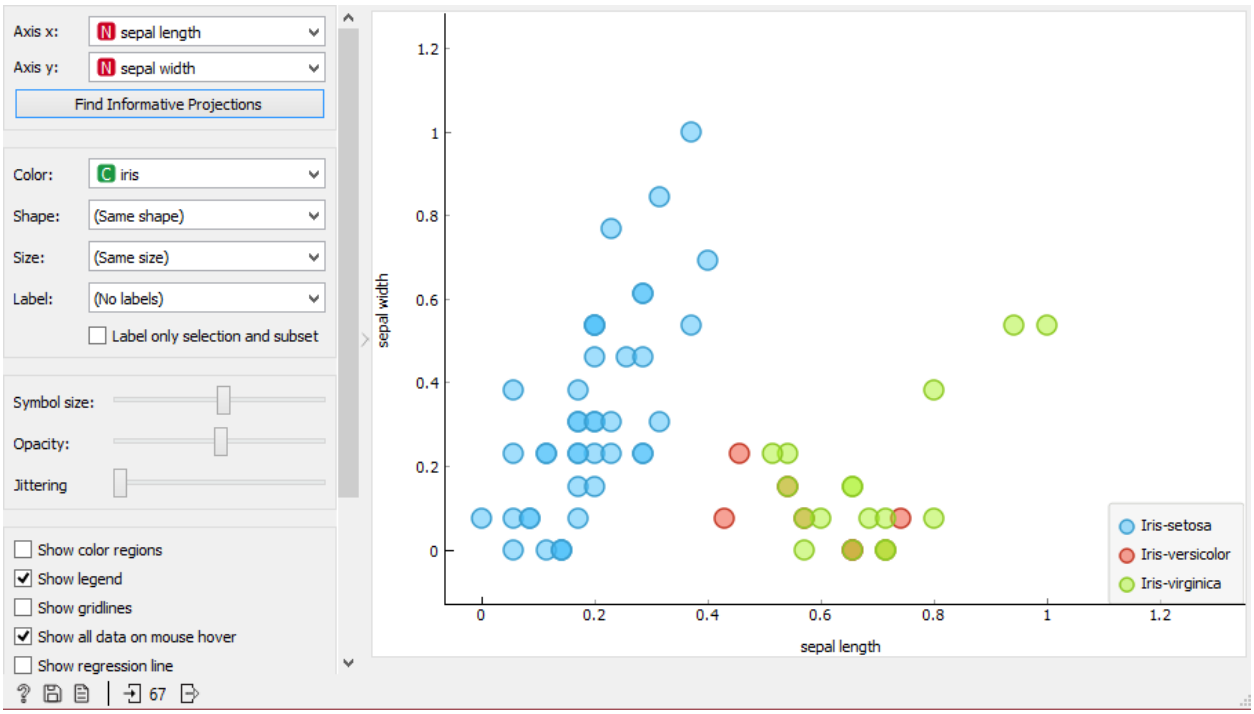


Fig5.1 - Scatter Plot

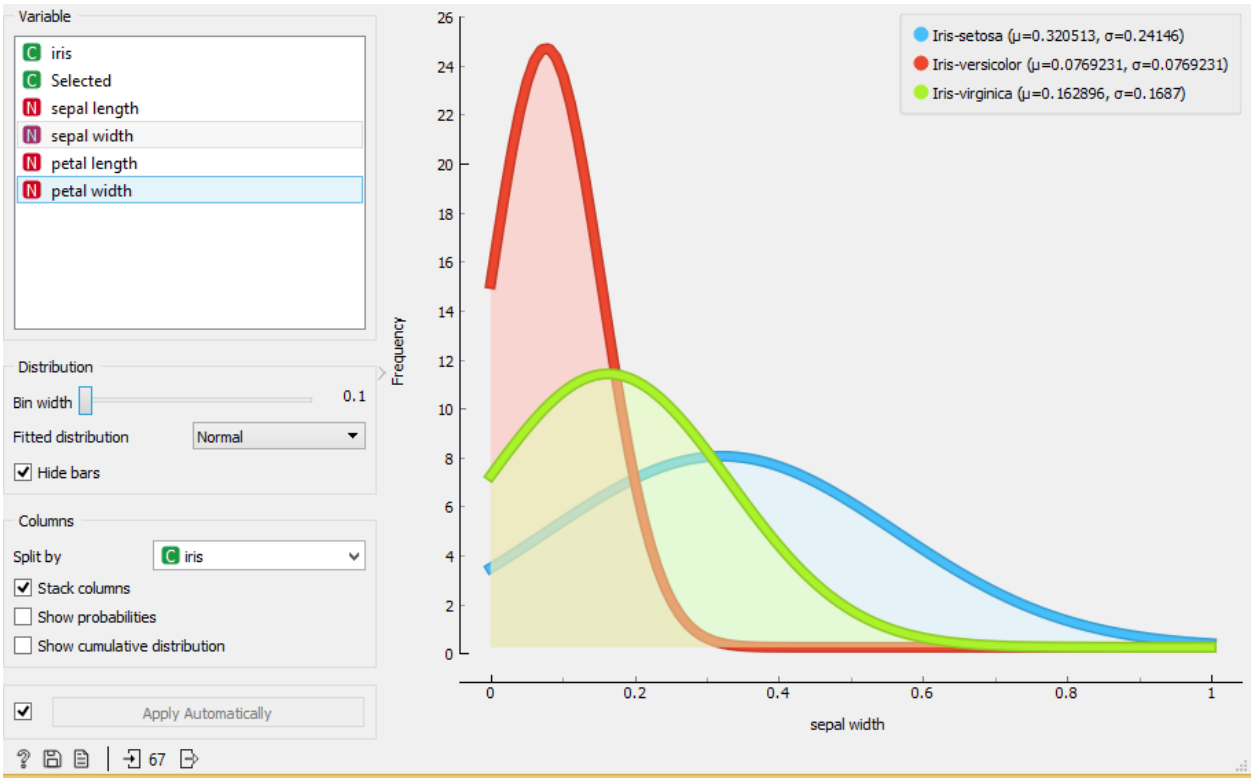


Fig5.2 – Distribution

AIM: Implement Apriori algorithm of association rule data mining technique in any Programming language.

ABSTRACT:

There are several mining algorithms of association rules. One of the most popular algorithms is Apriori that is used to extract frequent item sets from large database and getting the association rule for discovering the knowledge.

ASSOCIATION RULE MINING:

Association Mining is one of the most important data mining's functionalities and it is the most popular technique has been studied by researchers. Extracting association rules is the core of data mining. It is mining for association rules in database of sales transactions between items which is important field of the research in dataset. The benefits of these rules are detecting unknown relationships, producing results which can perform basis for decision making and prediction. The discovery of association rules is **divided into two** phases: **detection the frequent item sets** and **generation of association rules**.

In the first phase, every set of items is called
Item set, if they occurred together greater than the minimum support threshold, this item set is called frequent item set. Finding frequent item sets is easy but costly so this phase is more important than second phase. In the second phase, it can generate many rules from one item set as
in form, if item set {I1, I2, I3}, its rules are {I1->I2,I3}, {I2->I1,I3}, {I3->I1,I2}, {I1, I2->I3}, {I1,I3->I1}, {I2,I3->I1}, number of those rules is n^2-1 where n = number of items. To validate the rule (e.g. X → Y), where X and Y are items, based on confidence threshold which determine the ratio of the transactions which contain X and Y to the transactions A% which contain X, this means that A% of the transactions which contain X also contain Y. minimum support and confidence is defined by the user which represents constraint of the rules. So the support and confidence thresholds should be applied for all the rules to prune the rules which it values less than thresholds values. The problem that is addressed into association mining is finding the correlation among different items from large set of transactions efficiency.

The research of association rules is motivated by more applications such as telecommunication, banking, health care and manufacturing, etc.

RELATED WORK:

Mining of frequent item sets is an important phase in association mining which discovers frequent item sets in transactions database. It is the core in many tasks of data mining that try to find interesting patterns from datasets, such as association rules, episodes, classifier, clustering and correlation, etc.
Many algorithms are proposed to find frequent item sets, but all of them can be catalogued into two classes: candidate generation or pattern growth.

Apriori is a representative the candidate generation approach. It generates length (k+1)
Candidate item sets based on length (k) frequent item sets. The frequency of item sets is defined by counting their occurrence in transactions.

FP-growth, is proposed by Han in 2000, represents
pattern growth approach, it used specific data structure (FP-tree), FP-growth discover the frequent item sets by finding all frequent in 1-itemsets into condition pattern base , the condition pattern base is constructed efficiently based on the link of node structure that association with FP-tree. FP-growth does not generate candidate item sets explicitly.

LIMITATIONS OF APRIORI ALGORITHM:
Apriori algorithm suffers from some weakness in spite of being clear and simple. The main limitation is costly wasting of time to hold a vast number of candidate sets with much frequent item sets, low minimum support or large item sets. For example, if there are 104 from frequent 1-itemsets, it need to generate more than 107 candidates into 2-length which in turn they will be tested and accumulate. Furthermore, to detect frequent pattern in size 100 (e.g.) v1, v2…v100, it have to generate 2^100 candidate item sets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate item sets, also it will scan database many times repeatedly for finding candidate item sets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

List the techniques to improve the efficiency of Apriori algorithm

1. Hash based technique
2. Reduction
3. Portioning
4. Sampling
5. Dynamic item counting

Implement Apriori:

```
#include <iostream>
#include <vector>
#include <string>
#include <set>
#include <string>
#include <fstream>
#include <utility>
#include <algorithm>
#include <sstream>
using namespace std;

// this structure is for storing L3:{I1,I2,I3 => sup}
struct four{
   int I1,I2,I3,sup;
};
```

```
float found2(int a,int b,vector<pair<int,pair<int,int>>> L2){
    int length = L2.size();
    for(int i=0;i<length;++i){
        if(L2[i].second.first == a && L2[i].second.second == b )
            return (float)L2[i].first;
    }
    return 0;
}
float found1(int a,vector<pair<int,int>> L1){
    int length = L1.size();
    for(int i=0;i<length;++i){
        if(L1[i].first == a)
            return (float)L1[i].second;
    }
    return 0;
}


int main(){
    int TD=0,i=0,j,length;
    int Min_sup;
    string line,s;
    // store unique items
    set<int> item;

    /* Input format
    first Total number of Transaction: TD
    second Minimum support
    then next TD lines with space separated Items
    */
    ifstream Tfile("T4.txt");
    if(!(Tfile.is_open())){
        cout<<"Unable to open";
        exit(1);
    }
    getline(Tfile,line);
    TD = stoi(line);
    // 2D vector to store transaction with items
    vector<int> v[TD];

    getline(Tfile,line);
    Min_sup = stoi(line);
    while(getline(Tfile,line)){
        stringstream ss(line);
        while(ss>>s){
```

```
            v[i].push_back((stoi(s)));
            item.insert(stoi(s));
        }
        ++i;
    }
    Tfile.close();

//============================ THE APRIORI ==============================//
    // Frequent item set L1
    vector<pair<int,int>> L1;
    for(auto it=item.cbegin(); it != item.cend(); ++it){
        int item_count = 0;
       for(i=0;i<TD;++i){
            // find check only once
            if(find(v[i].begin(),v[i].end(),*it) != v[i].end())
                ++item_count;
       }
        if(item_count>=Min_sup)
            L1.push_back(make_pair(*it,item_count));
    }
    // free memory of set
    item.clear();

    // Frequent item set L2
    vector<pair<int,pair<int,int>>> L2;
    length = L1.size();
    for(i=0;i<length-1;++i){
        for(j=i+1;j<length;++j){
            int item_count=0;
            for(int k=0;k<TD;++k){
                if(find(v[k].begin(),v[k].end(),L1[i].first) != v[k].end() &&
                   find(v[k].begin(),v[k].end(),L1[j].first) != v[k].end() )
                    ++item_count;
            }
            if(item_count>=Min_sup)
                L2.push_back(make_pair(item_count,make_pair(L1[i].first,L1[j].first)));
        }
    }

    // Frequent item set L3
    vector<four> L3;
    vector<string> check2;
    length = L2.size();
    for(i=0;i<length-1;++i){
        for(j=i+1;j<length;++j){
```

```
        if(L2[i].second.first==L2[j].second.first ||
          L2[i].second.second==L2[j].second.second ||
          L2[i].second.second==L2[j].second.first){
            set<int> check1;
            check1.insert(L2[i].second.first);
            check1.insert(L2[i].second.second);
            check1.insert(L2[j].second.first);
            check1.insert(L2[j].second.second);
            string gs = "";
            for(auto it=check1.cbegin(); it != check1.cend(); ++it){
                gs += to_string(*it);
            }
            if(!(find(check2.begin(),check2.end(),gs) != check2.end())){
                check2.push_back(gs);

                int item_count=0;
                for(int k=0;k<TD;++k){
                   if(find(v[k].begin(),v[k].end(),(gs[0]-'0')) != v[k].end() &&
                     find(v[k].begin(),v[k].end(),(gs[1]-'0')) != v[k].end() &&
                     find(v[k].begin(),v[k].end(),(gs[2]-'0')) != v[k].end())
                      ++item_count;
                }
                if(item_count>=Min_sup)
                   L3.push_back({(gs[0]-'0'),(gs[1]-'0'),(gs[2]-'0'),item_count});
            }
        }
      }
   }
//============================                PRINT                EVERYTHING
================================//
   // PRINT Transactions
   cout<<"Transactions: \n";
   for(i=0;i<TD;++i){
        length=v[i].size();
      for(j=0;j<length;++j){
         cout<<"I"<<v[i][j]<<" ";
      }
      cout<<"\n";
   }
   cout<<"\nMinimum Support: "<<Min_sup<<"\n";
   // PRINT L1
   cout<<"\nFrequent item set L1:\n";
   length = L1.size();
   for(i=0;i<length;++i)
      cout<<"I"<<L1[i].first<<" "<<L1[i].second<<"\n";
```

```
   // PRINT L2
   if(L2.size()>0){
      cout<<"\nFrequent item set L2:\n";
      length = L2.size();
      for(i=0;i<length;++i){
         cout<<"{I"<<L2[i].second.first<<", I"<<L2[i].second.second<<"} "<<L2[i].first<<"\n";
      }
      if(L3.size()<1){
         cout<<"\nSo rules can be:\n";
         for(i=0;i<length;++i){
            cout<<"I"<<L2[i].second.first<<"      =>      I"<<L2[i].second.second<<"          "<<(
((float)L2[i].first/found1(L2[i].second.first,L1))*100 )<<" %\n";
            cout<<"I"<<L2[i].second.second<<"      =>      I"<<L2[i].second.first<<"          "<<(
((float)L2[i].first/found1(L2[i].second.second,L1))*100 )<<" %\n";
         }
      }
   }

   // PRINT L3
   if(L3.size()>0){
      cout<<"\nFrequent item set L3:\n";
      length = L3.size();
      for(i=0;i<length;++i){
         cout<<"{I"<<L3[i].I1<<",I"<<L3[i].I2<<",I"<<L3[i].I3<<"} "<<L3[i].sup<<"\n";
      }
      cout<<"\nSo rules can be:\n";
      for(i=0;i<length;++i){
         cout<<"I"<<L3[i].I1<<",I"<<L3[i].I2<<"       =>       I"<<L3[i].I3<<"                "<<(
((float)L3[i].sup/found2(L3[i].I1,L3[i].I2,L2))* 100.0 )<<" %\n";
         cout<<"I"<<L3[i].I1<<",I"<<L3[i].I3<<"       =>       I"<<L3[i].I2<<"                "<<(
((float)L3[i].sup/found2(L3[i].I1,L3[i].I3,L2))* 100.0 )<<" %\n";
         cout<<"I"<<L3[i].I2<<",I"<<L3[i].I3<<"       =>       I"<<L3[i].I1<<"                "<<(
((float)L3[i].sup/found2(L3[i].I2,L3[i].I3,L2))* 100.0 )<<" %\n";
         cout<<"I"<<L3[i].I1<<"       =>       I"<<L3[i].I2<<",I"<<L3[i].I3<<"                "<<(
((float)L3[i].sup/found1(L3[i].I1,L1))* 100.0 )<<" %\n";
         cout<<"I"<<L3[i].I2<<"       =>       I"<<L3[i].I1<<",I"<<L3[i].I3<<"                "<<(
((float)L3[i].sup/found1(L3[i].I2,L1))* 100.0 )<<" %\n";
         cout<<"I"<<L3[i].I3<<"       =>       I"<<L3[i].I1<<",I"<<L3[i].I2<<"                "<<(
((float)L3[i].sup/found1(L3[i].I3,L1))* 100.0 )<<" %\n";
         cout<<"\n";
      }
   }
   return 0;
}
```

```
Transactions:
I1 I2 I5
I2 I4
I2 I4
I1 I2 I4
I1 I3
I2 I3
I1 I3
I1 I2 I3 I5
I1 I2 I3

Minimum Support: 3

Frequent item set L1:
I1 6
I2 7
I3 5
I4 3

Frequent item set L2:
{I1, I2} 4
{I1, I3} 4
{I2, I3} 3
{I2, I4} 3

So rules can be:
I1 => I2  66.6667 %
I2 => I1  57.1429 %
I1 => I3  66.6667 %
I3 => I1  80 %
I2 => I3  42.8571 %
I3 => I2  60 %
I2 => I4  42.8571 %
I4 => I2  100 %
```

```
Transactions:
I1 I2 I5
I2 I4
I2 I3
I1 I2 I4
I1 I3
I2 I3
I1 I3
I1 I2 I3 I5
I1 I2 I3

Minimum Support: 2

Frequent item set L1:
I1 6
I2 7
I3 6
I4 2
I5 2

Frequent item set L2:
{I1, I2} 4
{I1, I3} 4
{I1, I5} 2
{I2, I3} 4
{I2, I4} 2
{I2, I5} 2

Frequent item set L3:
{I1,I2,I3} 2
{I1,I2,I5} 2

So rules can be:
I1,I2 => I3   50 %
I1,I3 => I2   50 %
I2,I3 => I1   50 %
I1 => I2,I3   33.3333 %
I2 => I1,I3   28.5714 %
I3 => I1,I2   33.3333 %

I1,I2 => I5   50 %
I1,I5 => I2   100 %
I2,I5 => I1   100 %
I1 => I2,I5   33.3333 %
I2 => I1,I5   28.5714 %
I5 => I1,I2   100 %
```

```
Transactions:
I1 I3 I4
I2 I3 I5
I1 I2 I3 I5
I2 I5

Minimum Support: 2

Frequent item set L1:
I1 2
I2 3
I3 3
I5 3

Frequent item set L2:
{I1, I3} 2
{I2, I3} 2
{I2, I5} 3
{I3, I5} 2

Frequent item set L3:
{I2,I3,I5} 2

So rules can be:
I2,I3 => I5   100 %
I2,I5 => I3   66.6667 %
I3,I5 => I2   100 %
I2 => I3,I5   66.6667 %
I3 => I2,I5   66.6667 %
I5 => I2,I3   66.6667 %
```

AIM: Apply association rule data mining technique on sample data sets in XLMiner.

Introduction to XLMiner:

XLMiner SDK offers comprehensive facilities for deep data analysis for exploration/educational purposes, as well as for high-performance production applications. Usage can range from a single line of code with minimal user intervention (XLMiner SDK finds sensible defaults when possible), to a customized workflow where users can manually control all parts of exploration, fitting, and scoring processes. XLMiner SDK's easy-to-use API allows the combination of all available data mining techniques into a single, all-inclusive application, or pipeline. For example, your pipeline could first draw a sample from Big Data stored across an Apache Spark cluster, then use Feature Selection to determine the best inputs to a supervised algorithm, partition the data, fit a model, and score new data.
For exploration purposes, XLMiner SDK offers advanced algorithms that provide deep insights about your data/models. Using this information, you can fine-tune the parameters and search for optimal choices to prepare the application for deployment into a production application. At the deployment stage, “auxiliary” informative outputs might not be required, so the user can choose to remove this ancillary code for lightning fast performance.

- XLMiner is a comprehensive data mining add-in for Excel.
- It offers a variety of methods to analyse data.
- It has extensive coverage of statistical and machine learning techniques for classification, prediction, affinity analysis and data exploration and reduction.
- Here XLMiner is used to view and find the association rules over a dataset to find out the associating patterns from the data.
- Here the values in the data set are as follows from which the patterns are to be recognised.

	A	B	C	D	E	F	G	H	I
1	Bread	PenutBut	Milk	Fruit	Cheese	Vegtable	Cream	Soda	
2	1	1	1	1	0	0	0	0	
3	1	1	1	1	0	1	0	1	
4	0	0	0	1	0	0	1	0	
5	1	0	0	1	0	0	0	1	
6	0	1	1	1	0	0	0	1	
7	1	0	1	1	0	0	0	1	
8	0	0	1	1	0	0	0	1	
9	0	1	1	1	0	0	0	1	
10	0	0	0	1	1	1	0	0	
11	0	0	0	0	0	0	0	0	
12									

Fig5.1 – Transaction Dataset

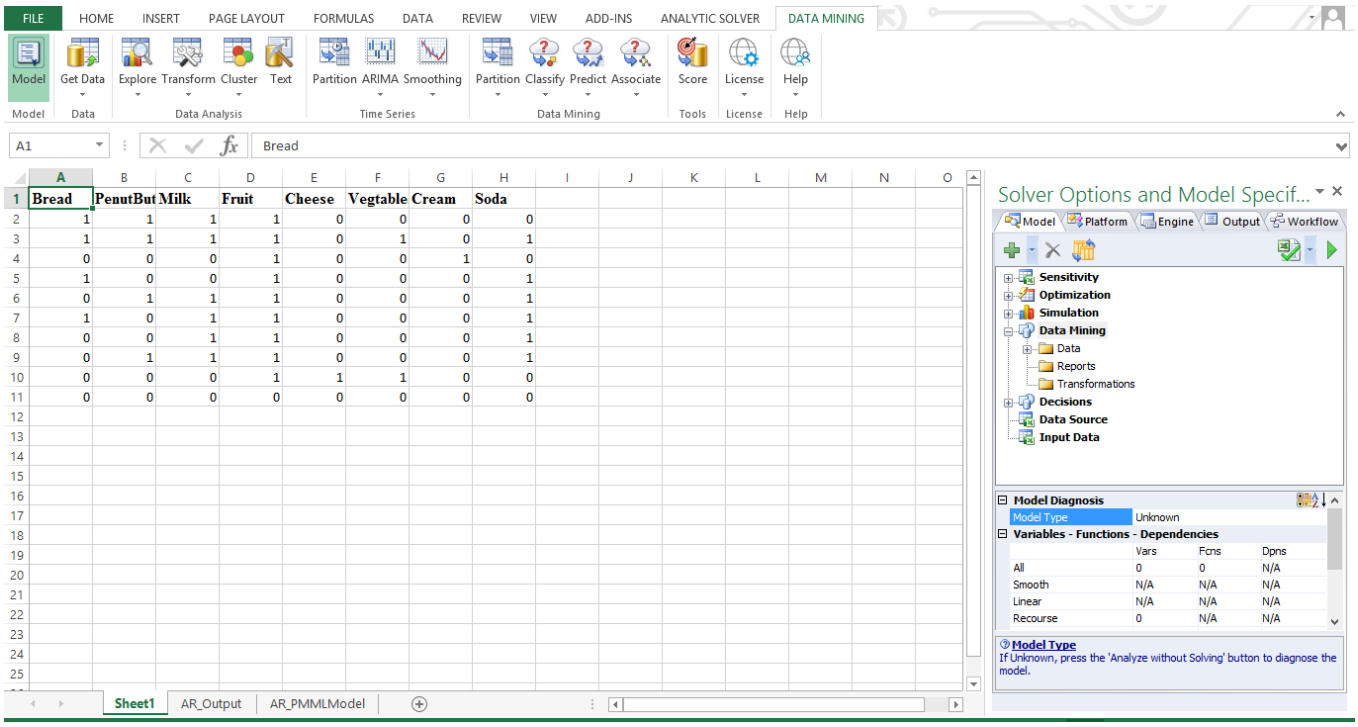


Fig5.2 – Xlminer

- Above Dataset is in form of Binary matrix where first row is label of items
- It contains 10 Transactions and 8 items

Let's find association rule, click on in ***DATA MINING*** Tab

Associate > Association rule > ok

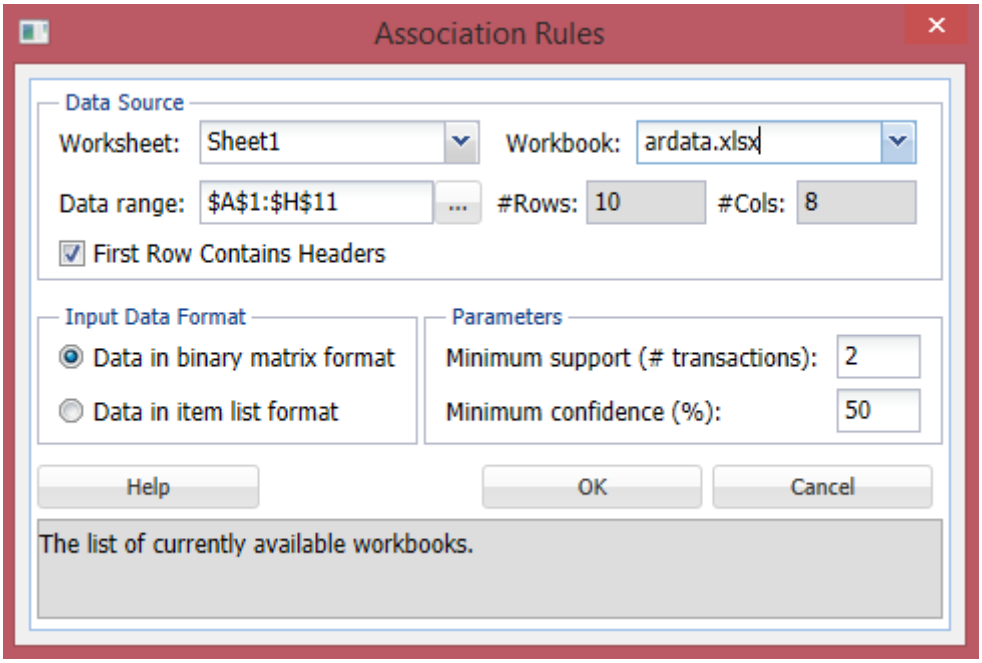


Fig 5.3 – Association rule configuration

Data Mining: Association Rules

Date: 13-Apr-2021 12:48:58

Output Navigator			
Inputs	Summary	Rules	PMML Model

Elapsed Times in Milliseconds			
Data Reading Time	gorithm Tin	Report Time	Total
0	499	75	574

Inputs

Data	
Workbook	transaction.xlsx
Worksheet	Sheet1
Range	$A$1:$H$11
# Records in the input data	10

Variables							
# Selected Variables	8						
Selected Variables	Bread	PenutButter	Milk	Fruit	Cheese	Vegtables	Cream	Soda

Association Rules: Fitting Parameters	
Method	Apriori
Min support	2
Min confidence	50

Association Rules: Reporting Parameters	
Data Format	Binary

Summary

Metric	Value
# Transacti	10
# Items	8
# Rules	95

Fig5.4 – Association rule output

Fig5.3 shows that what inputs are given for the processing and describes the inputs provided.

- Data format is binary,
- we have set min support is 2 and
- Min confidence is 50
- Total rules are 95

➢ We will see all rules at last. First sort all rules to find most relevant by sorting them with **min support** and **lift ratio** in descending order.

Rules

Rule ID	A-Support	C-Support	Support	Confidence	Lift-Ratio	Antecedent	Consequent
Rule 8	4	6	4	100	1.66666667	[PenutButter]	[Milk]
Rule 22	2	6	2	100	1.66666667	[Bread,PenutButter]	[Milk]
Rule 43	4	6	4	100	1.66666667	[PenutButter]	[Milk,Fruit]
Rule 46	4	6	4	100	1.66666667	[PenutButter,Fruit]	[Milk]
Rule 52	3	6	3	100	1.66666667	[PenutButter,Soda]	[Milk]
Rule 67	2	6	2	100	1.66666667	[Bread,PenutButter]	[Milk,Fruit]
Rule 73	2	6	2	100	1.66666667	[Bread,PenutButter,Fruit]	[Milk]
Rule 88	3	6	3	100	1.66666667	[PenutButter,Soda]	[Milk,Fruit]
Rule 94	3	6	3	100	1.66666667	[PenutButter,Fruit,Soda]	[Milk]
Rule 5	4	9	4	100	1.11111111	[Bread]	[Fruit]
Rule 10	4	9	4	100	1.11111111	[PenutButter]	[Fruit]
Rule 13	6	9	6	100	1.11111111	[Milk]	[Fruit]
Rule 17	2	9	2	100	1.11111111	[Vegtables]	[Fruit]
Rule 19	6	9	6	100	1.11111111	[Soda]	[Fruit]
Rule 27	2	9	2	100	1.11111111	[Bread,PenutButter]	[Fruit]
Rule 32	3	9	3	100	1.11111111	[Bread,Milk]	[Fruit]
Rule 41	3	9	3	100	1.11111111	[Bread,Soda]	[Fruit]
Rule 45	4	9	4	100	1.11111111	[PenutButter,Milk]	[Fruit]
Rule 57	3	9	3	100	1.11111111	[PenutButter,Soda]	[Fruit]
Rule 63	5	9	5	100	1.11111111	[Milk,Soda]	[Fruit]
Rule 72	2	9	2	100	1.11111111	[Bread,PenutButter,Milk]	[Fruit]
Rule 81	2	9	2	100	1.11111111	[Bread,Milk,Soda]	[Fruit]

Fig5.5 – confidence 100%

From fig5.5,
Confidence = support of {antecedent and consequent} / support of antecedent
Lift-Ratio = confidence / support of consequent

➢ There is a 100% chance that a customer will buy a Milk if he/she already bought PenutButter.
➢ So **Antecedent is if part** of rule and **consequent is than** part of rule.

Now let's all 95 rules.

Rules

Rule ID	A-Support	C-Support	Support	Confidence	Lift-Ratio	Antecedent	Consequent
Rule 1	4	4	2	50	1.25	[Bread]	[PenutButter]
Rule 2	4	4	2	50	1.25	[PenutButter]	[Bread]
Rule 3	4	6	3	75	1.25	[Bread]	[Milk]
Rule 4	6	4	3	50	1.25	[Milk]	[Bread]
Rule 5	4	9	4	100	1.11111111	[Bread]	[Fruit]
Rule 6	4	6	3	75	1.25	[Bread]	[Soda]
Rule 7	6	4	3	50	1.25	[Soda]	[Bread]
Rule 8	4	6	4	100	1.66666667	[PenutButter]	[Milk]
Rule 9	6	4	4	66.66666667	1.66666667	[Milk]	[PenutButter]
Rule 10	4	9	4	100	1.11111111	[PenutButter]	[Fruit]
Rule 11	4	6	3	75	1.25	[PenutButter]	[Soda]
Rule 12	6	4	3	50	1.25	[Soda]	[PenutButter]
Rule 13	6	9	6	100	1.11111111	[Milk]	[Fruit]
Rule 14	9	6	6	66.66666667	1.11111111	[Fruit]	[Milk]
Rule 15	6	6	5	83.33333333	1.38888889	[Milk]	[Soda]
Rule 16	6	6	5	83.33333333	1.38888889	[Soda]	[Milk]
Rule 17	2	9	2	100	1.11111111	[Vegtables]	[Fruit]
Rule 18	9	6	6	66.66666667	1.11111111	[Fruit]	[Soda]
Rule 19	6	9	6	100	1.11111111	[Soda]	[Fruit]
Rule 20	4	4	2	50	1.25	[Bread]	nutButter,Milk]
Rule 21	4	3	2	50	1.66666667	[PenutButter]	[Bread,Milk]
Rule 22	2	6	2	100	1.66666667	d,PenutButter]	[Milk]
Rule 23	3	4	2	66.66666667	1.66666667	[Bread,Milk]	[PenutButter]
Rule 24	4	4	2	50	1.25	utButter,Milk]	[Bread]
Rule 25	4	4	2	50	1.25	[Bread]	utButter,Fruit]
Rule 26	4	4	2	50	1.25	[PenutButter]	[Bread,Fruit]
Rule 27	2	9	2	100	1.11111111	d,PenutButter]	[Fruit]
Rule 28	4	4	2	50	1.25	[Bread,Fruit]	[PenutButter]
Rule 29	4	4	2	50	1.25	utButter,Fruit]	[Bread]
Rule 30	4	6	3	75	1.25	[Bread]	[Milk,Fruit]
Rule 31	6	4	3	50	1.25	[Milk]	[Bread,Fruit]
Rule 32	3	9	3	100	1.11111111	[Bread,Milk]	[Fruit]
Rule 33	4	6	3	75	1.25	[Bread,Fruit]	[Milk]
Rule 34	6	4	3	50	1.25	[Milk,Fruit]	[Bread]
Rule 35	4	5	2	50	1	[Bread]	[Milk,Soda]
Rule 36	3	6	2	66.66666667	1.11111111	[Bread,Milk]	[Soda]
Rule 37	3	6	2	66.66666667	1.11111111	[Bread,Soda]	[Milk]
Rule 38	4	6	3	75	1.25	[Bread]	[Fruit,Soda]
Rule 39	6	4	3	50	1.25	[Soda]	[Bread,Fruit]
Rule 40	4	6	3	75	1.25	[Bread,Fruit]	[Soda]
Rule 41	3	9	3	100	1.11111111	[Bread,Soda]	[Fruit]
Rule 42	6	4	3	50	1.25	[Fruit,Soda]	[Bread]
Rule 43	4	6	4	100	1.66666667	[PenutButter]	[Milk,Fruit]
Rule 44	6	4	4	66.66666667	1.66666667	[Milk]	utButter,Fruit]
Rule 45	4	9	4	100	1.11111111	utButter,Milk]	[Fruit]
Rule 46	4	6	4	100	1.66666667	utButter,Fruit]	[Milk]

Rule 47	6	4	4	66.66666667	1.66666667	[Milk,Fruit]	[PenutButter]
Rule 48	4	5	3	75	1.5	[PenutButter]	[Milk,Soda]
Rule 49	6	3	3	50	1.66666667	[Milk]	ıutButter,Soda]
Rule 50	6	4	3	50	1.25	[Soda]	nutButter,Milk]
Rule 51	4	6	3	75	1.25	ıutButter,Milk]	[Soda]
Rule 52	3	6	3	100	1.66666667	utButter,Soda]	[Milk]
Rule 53	5	4	3	60	1.5	[Milk,Soda]	[PenutButter]
Rule 54	4	6	3	75	1.25	[PenutButter]	[Fruit,Soda]
Rule 55	6	4	3	50	1.25	[Soda]	ıutButter,Fruit]
Rule 56	4	6	3	75	1.25	ıutButter,Fruit]	[Soda]
Rule 57	3	9	3	100	1.11111111	utButter,Soda]	[Fruit]
Rule 58	6	4	3	50	1.25	[Fruit,Soda]	[PenutButter]
Rule 59	6	6	5	83.33333333	1.38888889	[Milk]	[Fruit,Soda]
Rule 60	9	5	5	55.55555556	1.11111111	[Fruit]	[Milk,Soda]
Rule 61	6	6	5	83.33333333	1.38888889	[Soda]	[Milk,Fruit]
Rule 62	6	6	5	83.33333333	1.38888889	[Milk,Fruit]	[Soda]
Rule 63	5	9	5	100	1.11111111	[Milk,Soda]	[Fruit]
Rule 64	6	6	5	83.33333333	1.38888889	[Fruit,Soda]	[Milk]
Rule 65	4	4	2	50	1.25	[Bread]	ıtter,Milk,Fruit]
Rule 66	4	3	2	50	1.66666667	[PenutButter]	ead,Milk,Fruit]
Rule 67	2	6	2	100	1.66666667	d,PenutButter]	[Milk,Fruit]
Rule 68	3	4	2	66.66666667	1.66666667	[Bread,Milk]	ıutButter,Fruit]
Rule 69	4	4	2	50	1.25	[Bread,Fruit]	nutButter,Milk]
Rule 70	4	4	2	50	1.25	ıutButter,Milk]	[Bread,Fruit]
Rule 71	4	3	2	50	1.66666667	ıutButter,Fruit]	[Bread,Milk]
Rule 72	2	9	2	100	1.11111111	ıutButter,Milk]	[Fruit]
Rule 73	2	6	2	100	1.66666667	ıutButter,Fruit]	[Milk]
Rule 74	3	4	2	66.66666667	1.66666667	ead,Milk,Fruit]	[PenutButter]
Rule 75	4	4	2	50	1.25	tter,Milk,Fruit]	[Bread]
Rule 76	4	5	2	50	1	[Bread]	Milk,Fruit,Soda]
Rule 77	3	6	2	66.66666667	1.11111111	[Bread,Milk]	[Fruit,Soda]
Rule 78	4	5	2	50	1	[Bread,Fruit]	[Milk,Soda]
Rule 79	3	6	2	66.66666667	1.11111111	[Bread,Soda]	[Milk,Fruit]
Rule 80	3	6	2	66.66666667	1.11111111	ead,Milk,Fruit]	[Soda]
Rule 81	2	9	2	100	1.11111111	ead,Milk,Soda]	[Fruit]
Rule 82	3	6	2	66.66666667	1.11111111	ead,Fruit,Soda]	[Milk]
Rule 83	4	5	3	75	1.5	[PenutButter]	Milk,Fruit,Soda]
Rule 84	6	3	3	50	1.66666667	[Milk]	tter,Fruit,Soda]
Rule 85	6	4	3	50	1.25	[Soda]	ıtter,Milk,Fruit]
Rule 86	4	6	3	75	1.25	ıutButter,Milk]	[Fruit,Soda]
Rule 87	4	5	3	75	1.5	ıutButter,Fruit]	[Milk,Soda]
Rule 88	3	6	3	100	1.66666667	utButter,Soda]	[Milk,Fruit]
Rule 89	6	3	3	50	1.66666667	[Milk,Fruit]	ıutButter,Soda]
Rule 90	5	4	3	60	1.5	[Milk,Soda]	ıutButter,Fruit]
Rule 91	6	4	3	50	1.25	[Fruit,Soda]	nutButter,Milk]
Rule 92	4	6	3	75	1.25	tter,Milk,Fruit]	[Soda]
Rule 93	3	9	3	100	1.11111111	tter,Milk,Soda]	[Fruit]
Rule 94	3	6	3	100	1.66666667	ter,Fruit,Soda]	[Milk]
Rule 95	5	4	3	60	1.5	Milk,Fruit,Soda]	[PenutButter]

AIM: Apply Classification data mining technique on sample data sets in Weka.

We will be using C4.5, a successor of ID3, uses an extension to information gain known as gain ratio.

- The **information gain** measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values. For example, consider an attribute that acts as a unique identifier, such as product ID. A split on product ID would result in a large number of partitions (as many as there are values), each one containing just one tuple. Because each partition is pure, the information required to classify data set D based on this partitioning would be Info product ID (D) = 0. Therefore, the information gained by partitioning on this attribute is maximal. Clearly, such a partitioning is useless for classification. **C4.5 attempts to overcome this bias.**

Why it's called J48 in Weka

- An Australian computer scientist called Ross Quinlan. He started out with a system called ID3 and then C4.5, a successor of ID3 became quite famous. This kind of morphed through various versions into C4.5. It became famous; He continued to work on this system. It went up to C4.8, and then he went commercial. Up until then, these were all open source systems. When they built Weka, they took the latest version of C4.5, which was C4.8, and they rewrote it. Weka's written in Java, so they called it J48.

We will be using weka's built in diabetes dataset: **Open file> data > weather.nominal.arff**

- It's got 14 instances, 14 days, and for each of these days, have recorded the values of 5 attributes.
- 4 are to do with the weather: Outlook, Temperature, Humidity, and Windy.
- The fifth, Play, is whether or not we're going to play a particular, unspecified game. Actually, what we're going to be doing is predicting the Play attribute from the other attributes.
- The Pre-process panel tells you how many data rows, or "instances" are in the currently loaded data set.
- The "Selected attribute" box shows a summary of the currently selected attribute. By selecting different attributes in the "Attributes" box you can quickly investigate their possible values.

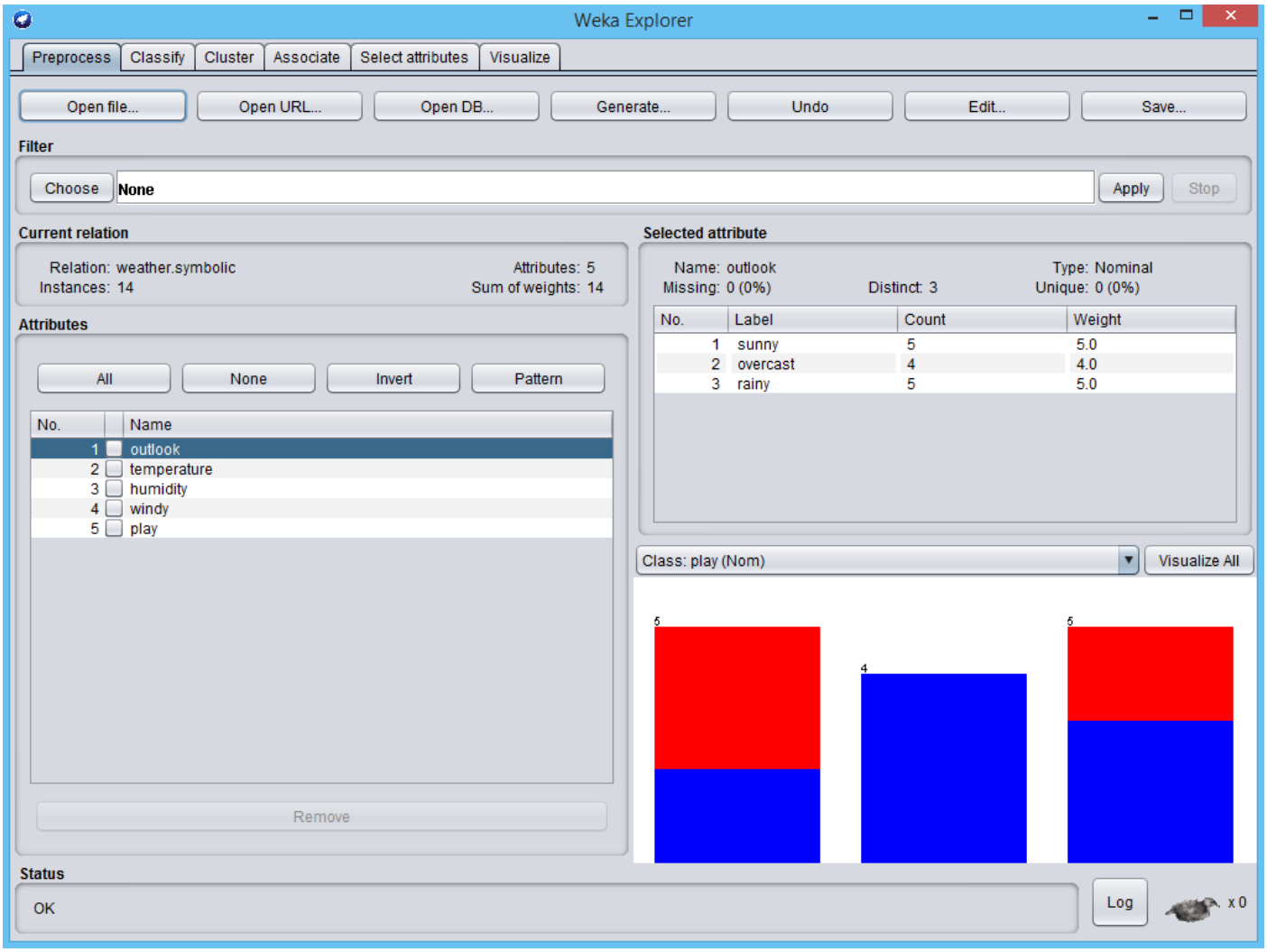


Fig1 – weather data

- Click on edit we can edit and view data.
- By default, the last attribute in Weka is always the class value. You can change this if you like.

We've already explored dataset. It's a **classification** problem, sometimes called a "**supervised learning**" problem. "Supervised" because you get to know the class values of the training instances. We take as input a data set as classified examples; these examples are independent examples with a class value attached. The idea is to produce automatically some kind of model that can classify new examples. That's the "classification" problem.

These **attributes, or features**, can be **discrete or continuous**. What we looked at in the weather data were discrete; we call them "**nominal**" attribute values when they belong to a certain fixed set. Or they can be numeric, or "continuous", values. Also, the class can be discrete or continuous. We're looking at a discrete class, "yes" or "no", in the case of the weather data. Another kind of machine learning problem would involve continuous classes, where you're trying to predict a number. That's called a "regression" problem in the trade.

Let's use J48 to whether or not we're going to play a particular unspecified game.

- classify > Choose J48 > start
- by clicking on "J48-C 0.25 -M 2" we get configuration tab
- by right clicking on one of result in result list we can visualize tree

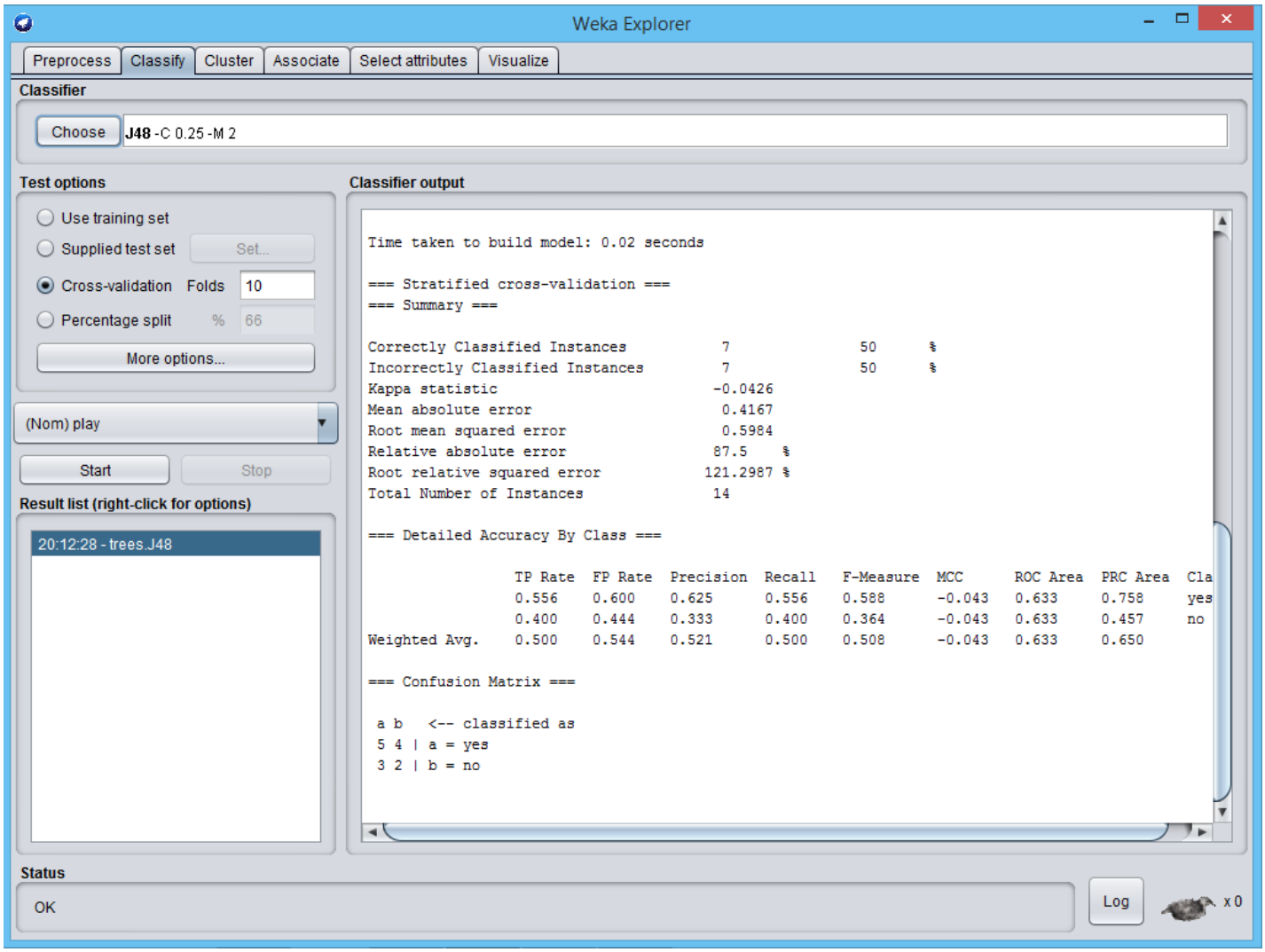


Fig2 – J48 classifier

We got 50% accuracy, confusion matrix shows that 7 instances are incorrectly classified. And 7 instances are correctly classified. We can see which instances are incorrectly classified by visualize classifier errors.

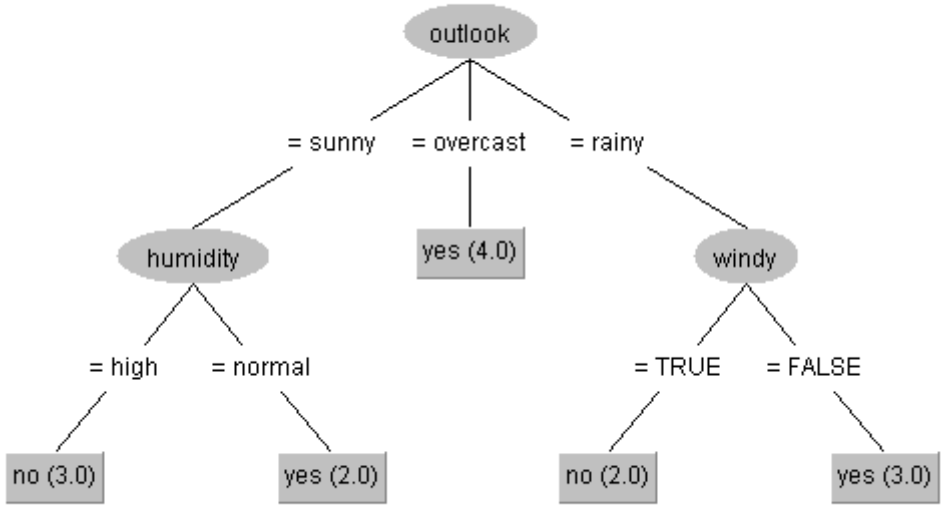


Fig3 – visualize decision tree

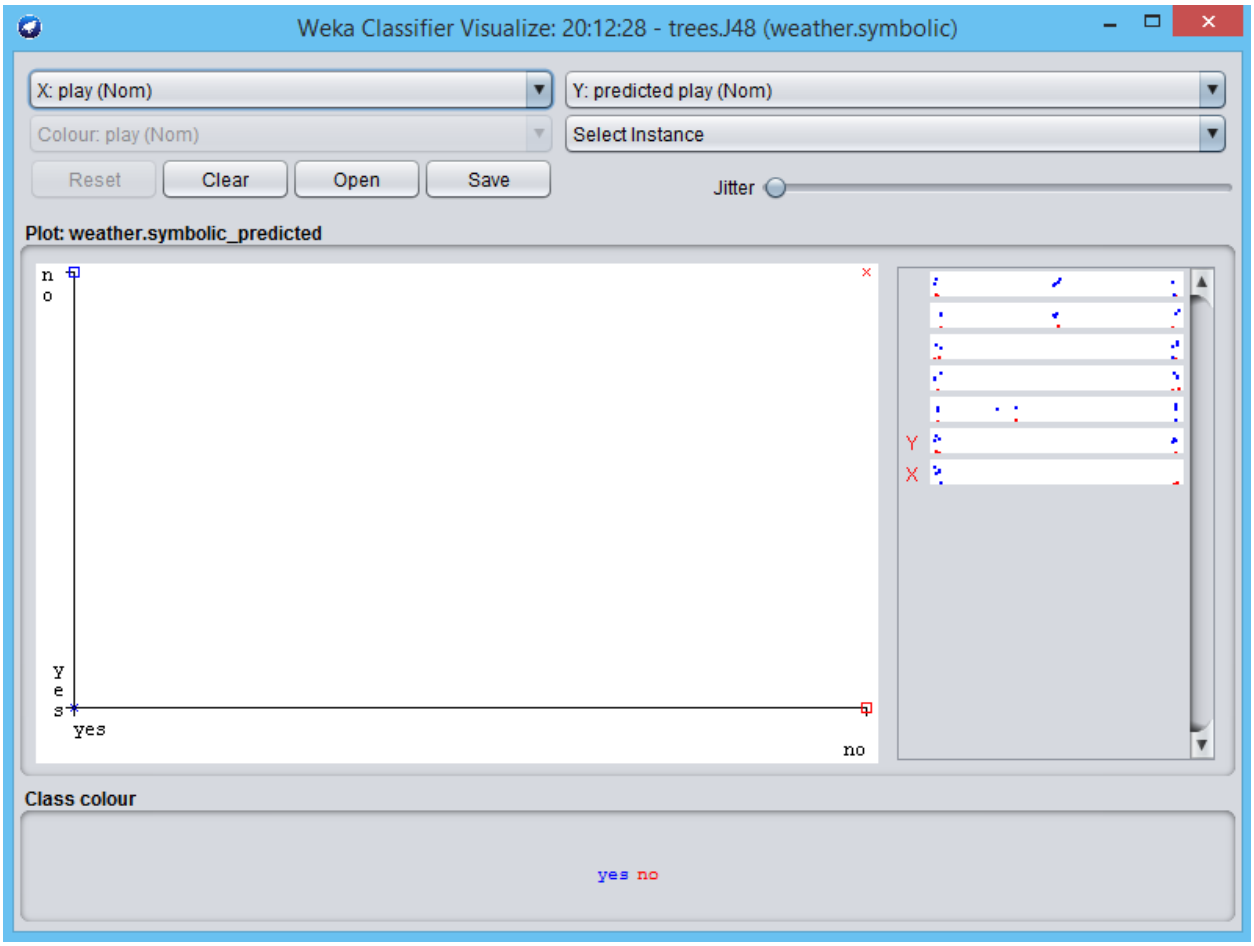


Fig4 - visualize classifier errors

- **Misclassified instances** are shown as **square boxes** on the visualization plot, and clicking the box shows their instance number. If multiple instances are shown, be sure to choose only those who's predicted Class differs from their class.

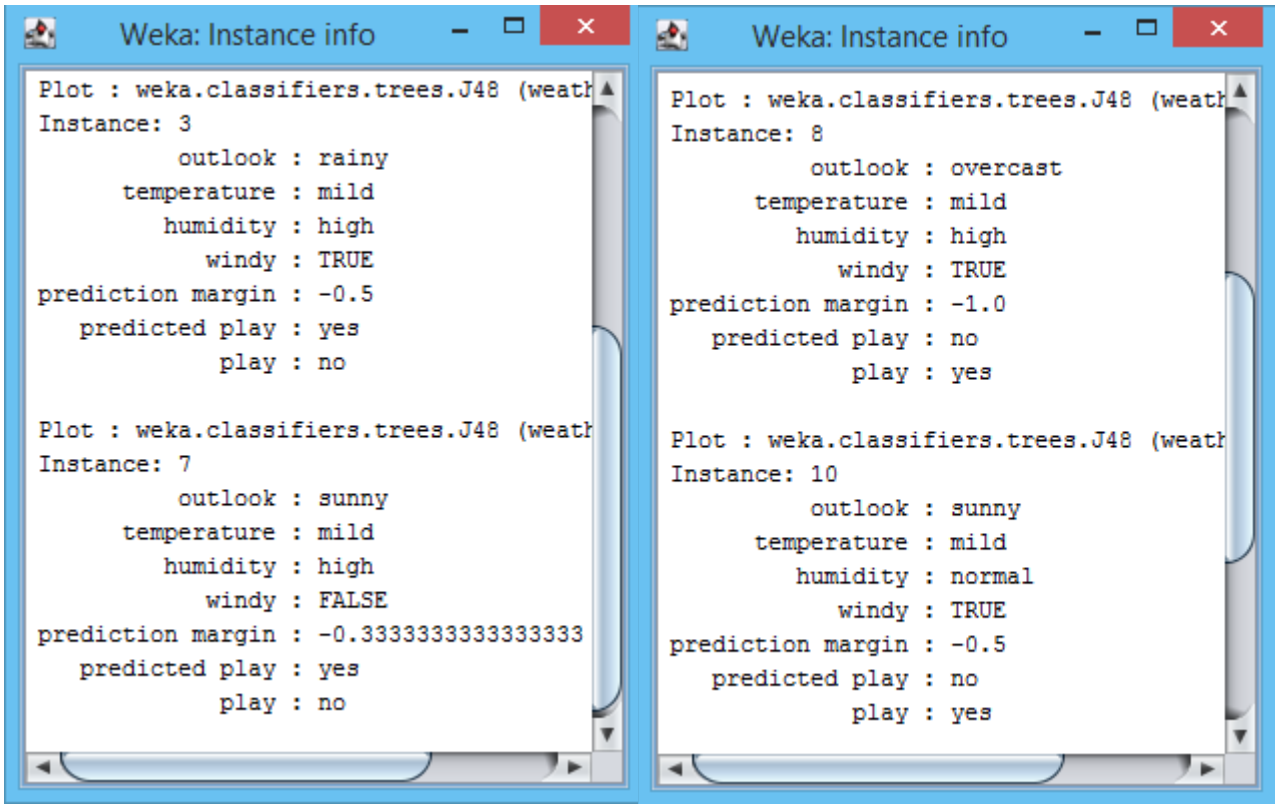


Fig5 - Misclassified instances

- Let us try to increase accuracy by setting MinNumObj=1 in configuration tab of J48.
- It sets the minimum number of instances per leaf node. It's set the minimum amount of data separation per branching.
- For example, separating one instance from 100 instances doesn't give much information so we set minimum amount of separation.

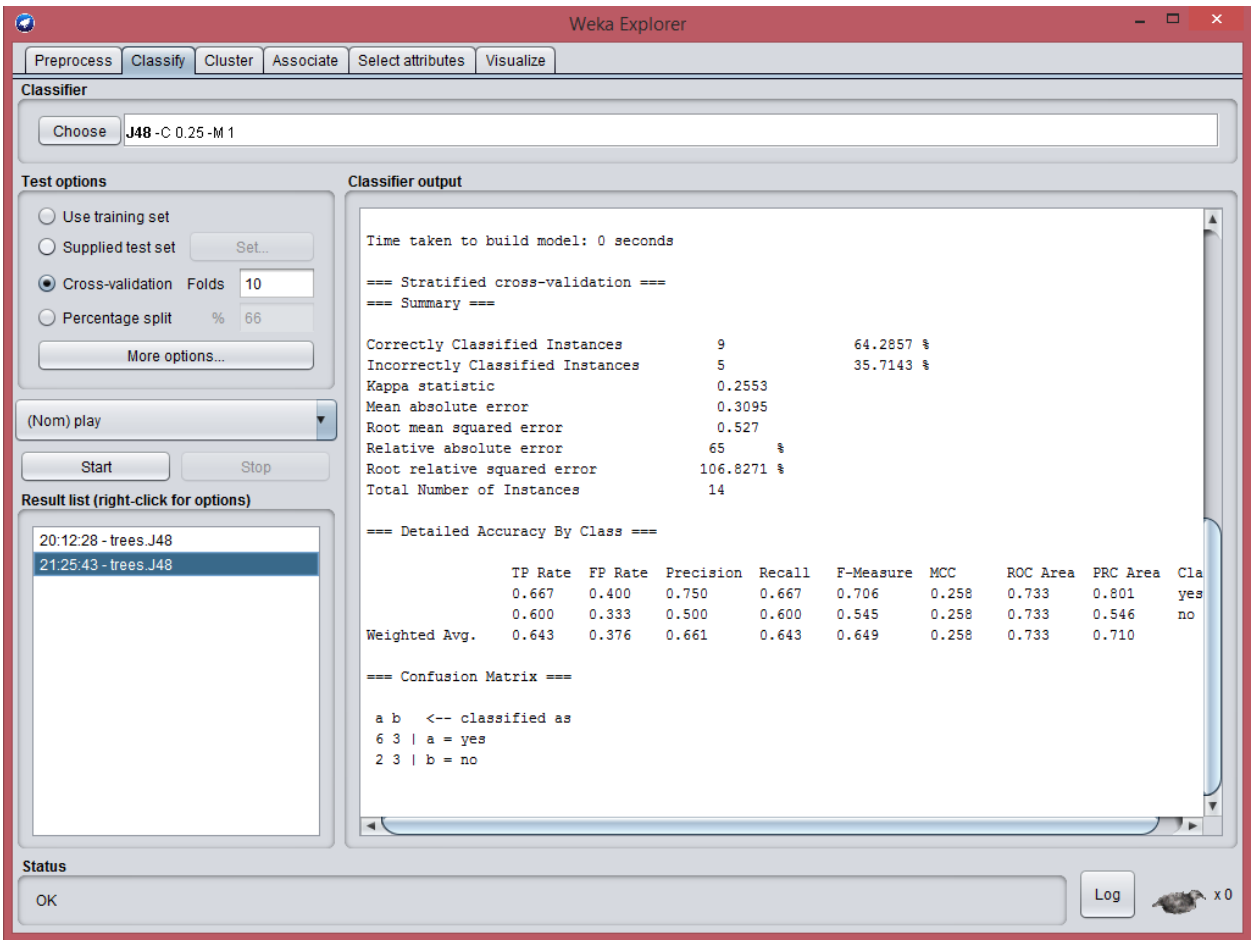


Fig6 – increase accuracy

10-fold cross-validation:

- Divide dataset into 10 parts (folds)
- Hold out each part in turn
- Average the results
- Each data point used once for testing and 9 times for training
- In **stratified cross-validation** we ensure that each fold has the right proportion of each class value because there is many ways to divide dataset in 10 parts.

AIM: A) Implement Classification technique with quality Measures in any Programming language.

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n-dimensional space. In this way, all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k "nearest neighbors" of the unknown tuple. "Closeness" is defined in terms of a distance metric, such as Euclidean distance.

- Here firstly the Nearest-neighbor classifiers is used to train over iris dataset with target values of the different species of iris flower.
- Accuracy of the trained classifier over test set is measured and predictions for the test data are made.

Nearest-neighbor classifiers:

```
import random
from scipy.spatial import distance

def euc(a,b):
    return distance.euclidean(a,b)

################# our k-nearest neighbor #############
class ScrappyKNN():
    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def predict(self, X_test):
        prediction = [] # store prediction
        for row in X_test:
            label = self.closest(row) # it's returen closest
            prediction.append(label)
        return prediction

    def closest(self , row):
        best_dist = euc(row, self.X_train[0])
        best_index = 0
        for i in range(1, len(self.X_train)):
            dist = euc(row , self.X_train[i])
            if dist < best_dist :
                best_dist = dist
                best_index = i
        return self.y_train[best_index]


import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
```

```
X = iris.data
y = iris.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5) #
0.5 for half train and half test

clf = ScrappyKNN()

clf.fit(X_train,y_train)
prediction = clf.predict(X_test)
#print(prediction)

from sklearn.metrics import accuracy_score
print(f"k-nearest neighbor: {accuracy_score(y_test,prediction)} ")
```

DecisionTreeClassifier:

```
################# DecisionTreeClassifier #############
from sklearn import tree
test_idx = [0,50,100]

# training data
train_target = np.delete(iris.target,test_idx)
train_data = np.delete(iris.data,test_idx,axis=0)

# testing data
test_target = iris.target[test_idx]
test_data = iris.data[test_idx]

clf = tree.DecisionTreeClassifier()
clf = clf.fit(train_data,train_target)
print(f"DecisionTreeClassifier: {accuracy_score(y_test,prediction)} ")

#viz code
from six import StringIO
import pydotplus
dot_data = StringIO()
tree.export_graphviz(clf,out_file=dot_data,
                         feature_names=iris.feature_names,
                         class_names=iris.target_names,
                         filled=True,rounded=True,
                         impurity=False)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("iris_KNN.pdf")
```

k-nearest neighbor: 0.9466666666666667
DecisionTreeClassifier: 0.9466666666666667

Let’s visualize Decision tree,

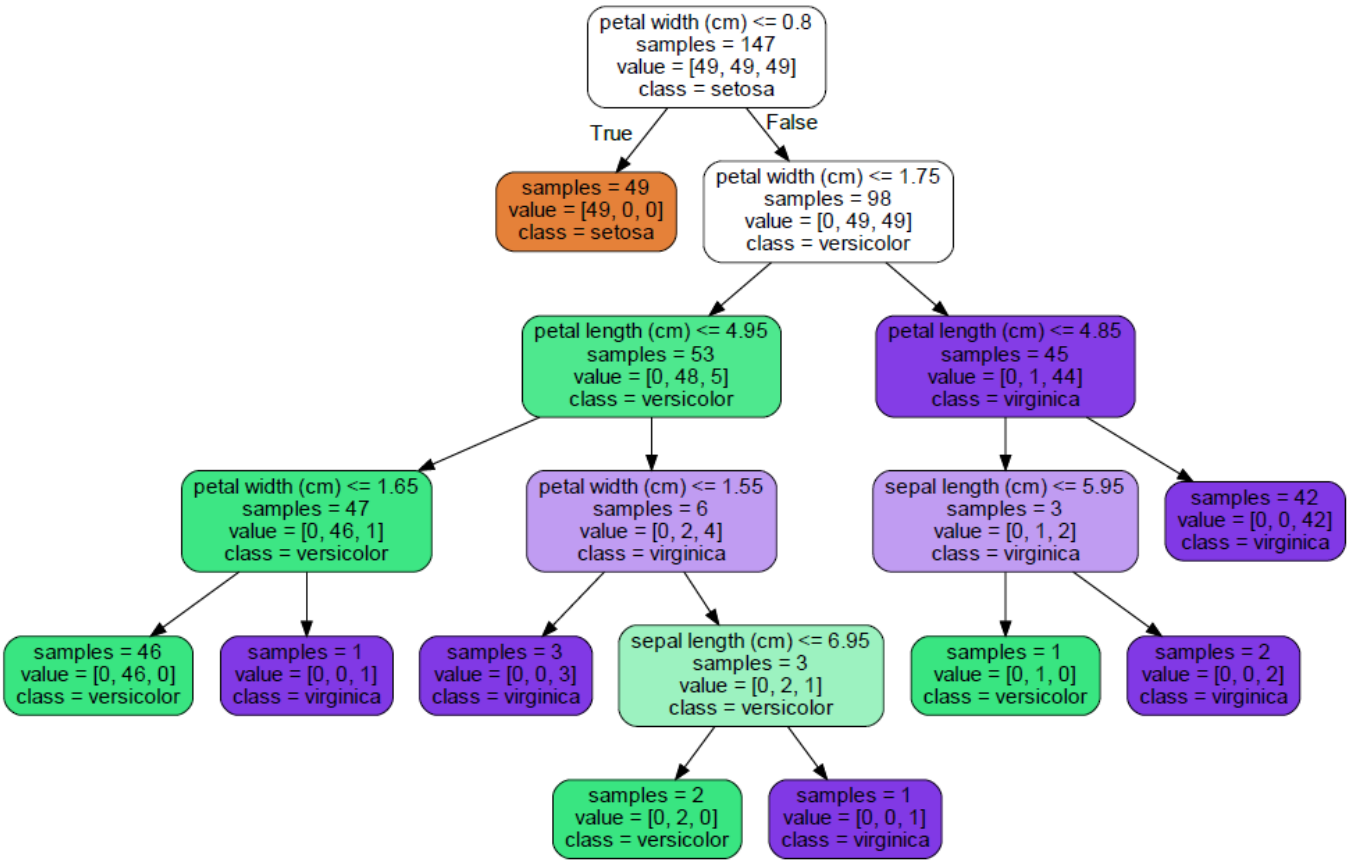


Fig7.1 Decision tree - iris classification

AIM: B) Implement Regression technique in any Programming language.

- The prediction of continuous values can be modeled by a statistical technique called regression.
- The objective of regression analysis is to determine the best model that can relate the output variable to various input variables. More formally, regression analysis is the process of determining how a variable Y is related to one or more other variables x1, x2,…, Xn. Y is usually called the response output, or dependent variable and x is are inputs, regressors, explanatory variables, or independent variables.

Here we are performing Linear Regression to find out how Head Size is relate to Brain weight.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (8.0,6.0)

data = pd.read_csv('headbrain.csv')
print(data.shape)
```

```
data.head()

X = data['Head Size(cm^3)'].values
Y = data['Brain Weight(grams)'].values

# mean X and Y
mean_x = np.mean(X)
mean_y = np.mean(Y)

# Total number of values
n = len(X)


'''
y = mx +c

m = E(x-mean_x)(y-mean_y)/E(x-mean_x)^2
m = a1/a2

c = y - mx
'''

# using the formula
a1 =0
a2 =0

for i in range(n):
    a1 += (X[i] - mean_x) * (Y[i] - mean_y)
    a2 += (X[i] - mean_x) ** 2

m = a1/a2
c = mean_y - (m*mean_x)

print(m,c)

# plotting values and regression line

max_x = np.max(X) + 100
min_x = np.min(X) - 100

x = np.linspace(min_x,max_x,1000)
y = m*x + c

plt.plot(x,y,color='#58b970',label='Regression Line')
plt.scatter(X,Y,c='#ef5423',label='Scatter Plot')
plt.xlabel('Head Size(cm^3)')
plt.ylabel('Brain Weight(grams)')
plt.legend()
plt.show()


'''
R-squared value is a statistical
R^2 = E(y_p-mean_y)^2/E(y-mean_y)^2
R^2 = b1/b2
'''
```

```
b1 = 0
b2 = 0
for i in range(n):
    y_p = m*X[i]+c
    b1 += (Y[i] - mean_y) ** 2
    b2 += (Y[i] - y_p) ** 2
r2 = 1 - (b2/b1)
print(r2)
```

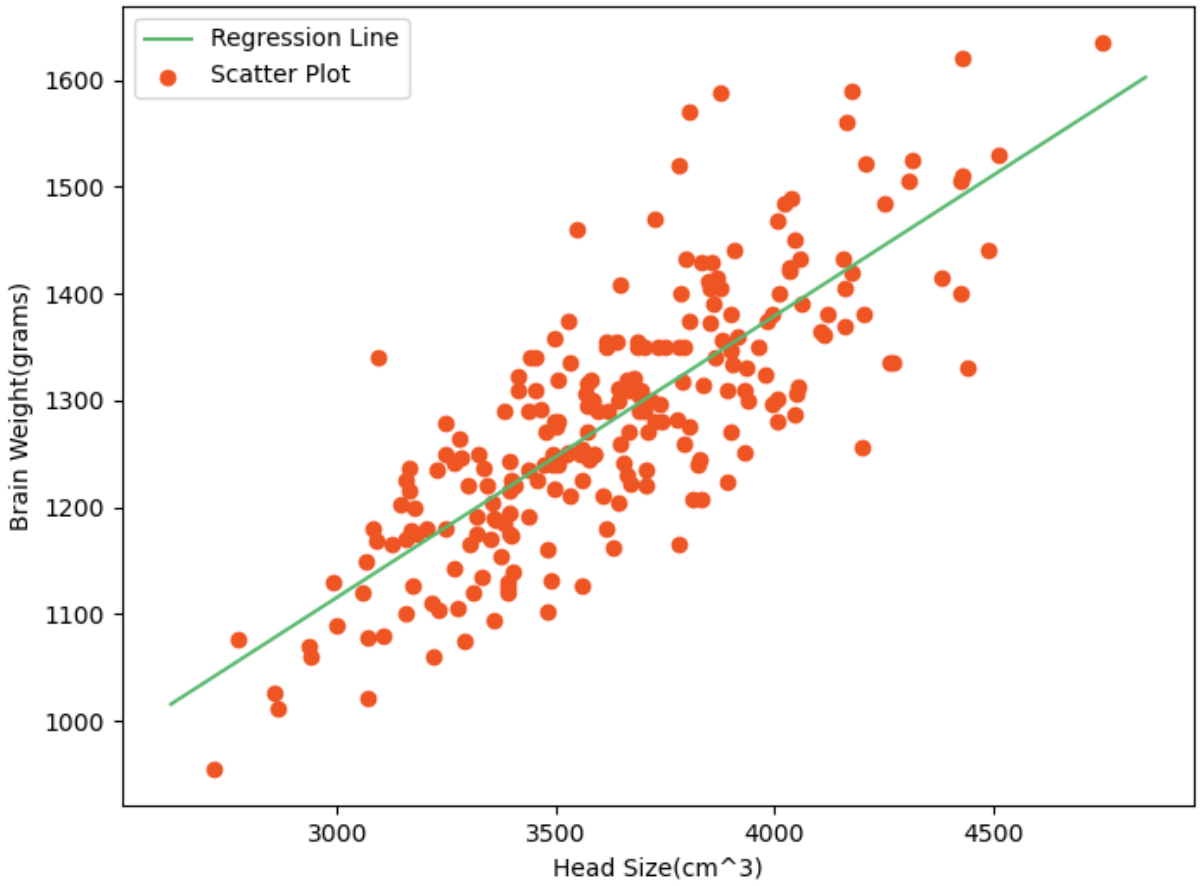


Fig7.2 – Scatter plot and Regression line

R^2 = 0.6393117199570003

R-squared value is a statistical measure of how close the data are to the fitted regression line it is also known as coefficient of determination or the coefficient of multiple determination.

AIM: Apply Clustering on sample Dataset Using Matlab.

- What is Matlab?

MATLAB is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

- What is Matlab used for?

Millions of engineers and scientists worldwide use MATLAB for a range of applications, in industry and academia, including deep learning and machine learning, signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology.

- **What is GNU octave?**

GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. Octave helps in solving linear and nonlinear problems numerically, and **for performing other numerical experiments using a language that is mostly compatible with MATLAB**. It may also be used as a batch-oriented language. Since it is part of the GNU Project, it is free software under the terms of the GNU General Public License.

Other free alternatives to MATLAB include Scilab and FreeMat. **Octave is more compatible with MATLAB** than Scilab and FreeMat has not been updated since June 2013.

We are going to use Octave Online.

Octave online is a **web** UI for GNU **Octave**, the open-source alternative to MATLAB. Thousands of students, educators, and researchers from around the world use **Octave Online** each day for studying machine learning, control systems, numerical methods, and more. https://octave-online.net/

- Now let's apply k-means clustering on sample dataset using Octave Online.
- Octave Online provides in-built functions to do data mining techniques so in this we will use
- K-means in-built function to do clustering and will plot clusters.

```
pkg load statistics
load HWMar26data;  %Use Matrix A for HW problem 1
```

```
clear B C

[idx,C,disterr]=kmeans(A,3);


% Plotting routines
idx1=find(idx==1);
idx2=find(idx==2);
idx3=find(idx==3);

plot(A(idx1,1),A(idx1,2),'r^');
hold on
plot(A(idx2,1),A(idx2,2),'b*');
plot(A(idx3,1),A(idx3,2),'ko');
hold off
```

Idx is a vector of predicted cluster indices corresponding to the observations. C is a 3-by-2 matrix containing the final centroid locations.
Use k-means to compute the distance from each centroid to points on a grid. To do this, pass the Centroids (C) and points on a grid to k-means, and implement one iteration of the algorithm.

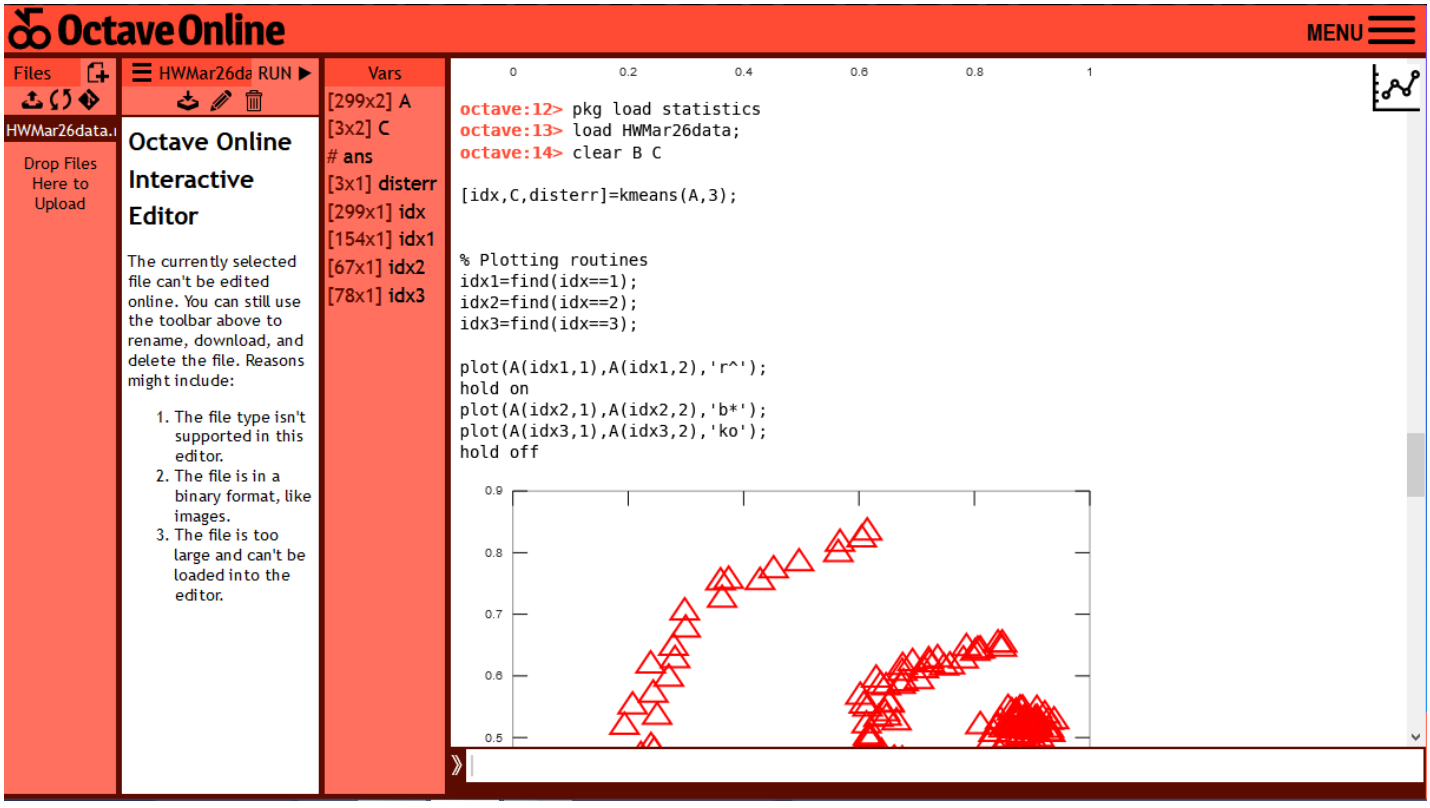
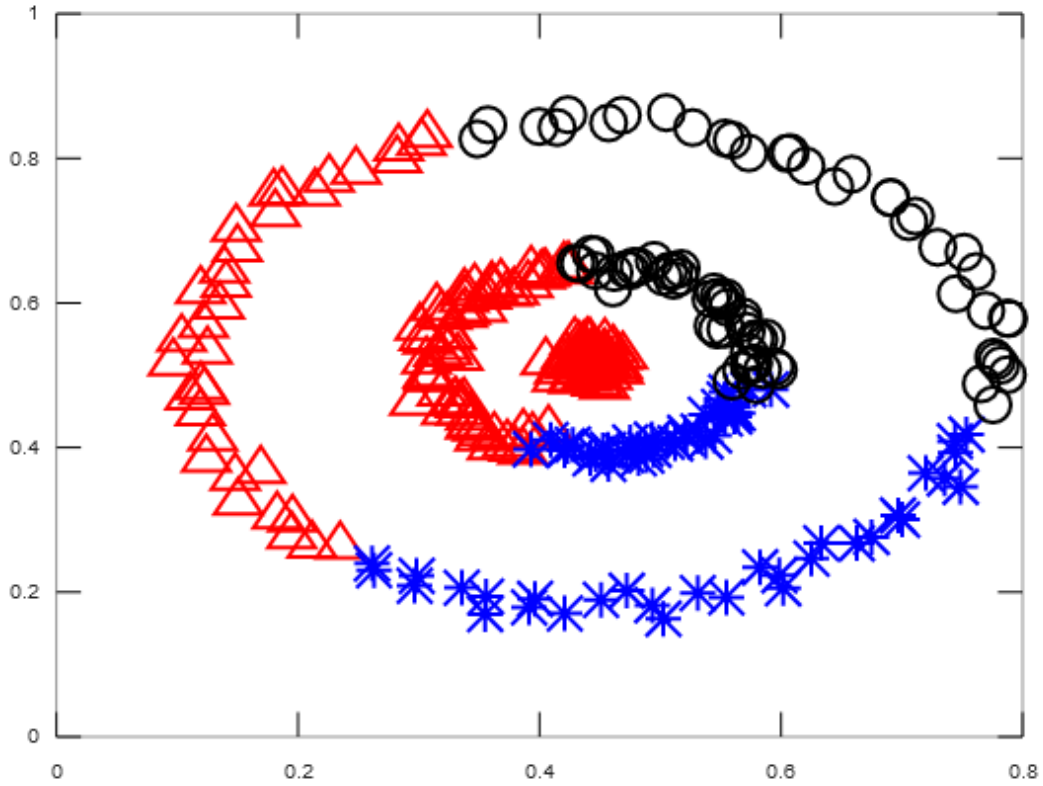
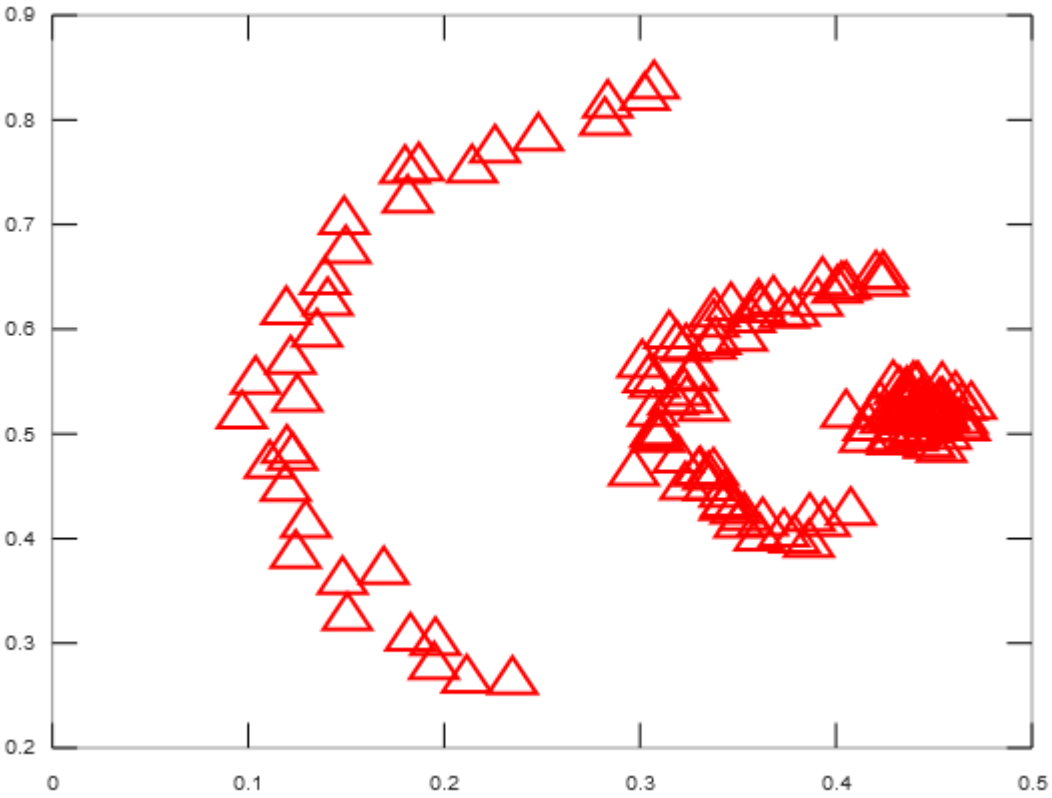


Fig8.1 – Octave Online


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
0.1
0.2
0.3
0.4
0.5

AIM: Perform hands on experiment on Web mining tool.

What is Web mining?

Web mining is the application of data mining techniques to discover patterns from the World Wide Web. As the name proposes, this is information gathered by mining the web. It makes utilization of automated apparatuses to reveal and extricate data from servers and web2 reports, and it permits organizations to get to both organized and unstructured information from browser activities, server logs, website and link structure, page content and different sources.

What is ProWebScrapper?

ProWebScraper is an incredible web content mining and web scraping tool. Its breathtaking features, uniquely uncomplicated process and unrivalled customer service make it the market champion of web scraping services. It eliminates your biggest fear- getting blocked. With ProWebScraper, you are never going to get blocked. You can simply relax and continue scraping web data. If you have bulk web data scraping in mind, ProWebScraper is the tool for it. In fact, it's designed for scraping vast quantities of data. It's easily scalable and yet produces clean and actionable data. It doesn't matter if the website is dynamic or its structure is complicated; ProWebScraper invariably ensures the extraction of data that you need. Icing on the cake is that it provides free custom set-up; you don't need to bother how to set it up. Leave the technicalities to ProWebScraper, you can just peg away at web data!

Scrapping laptops info from flipcart.com

We can go to our desired URL by adding it in the text box and hitting GO. Now, we are directed to the target URL. Here, it is noticeable that one can select almost all crucial elements like links, texts, images, etc. of the website. When the mouse pointer has green add symbol, by clicking on the element can scrap all the similar data in that column.

By selecting on the laptop link, we can add that. The columns can be renamed for better contexts.

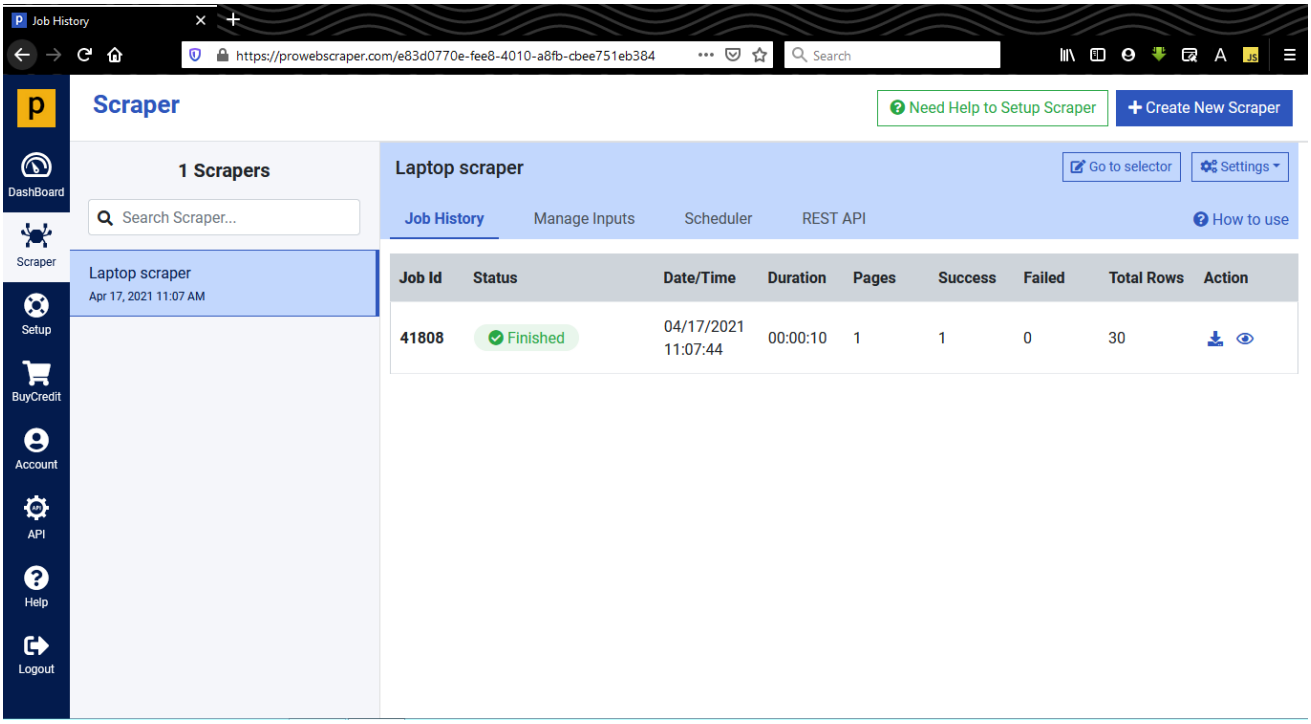
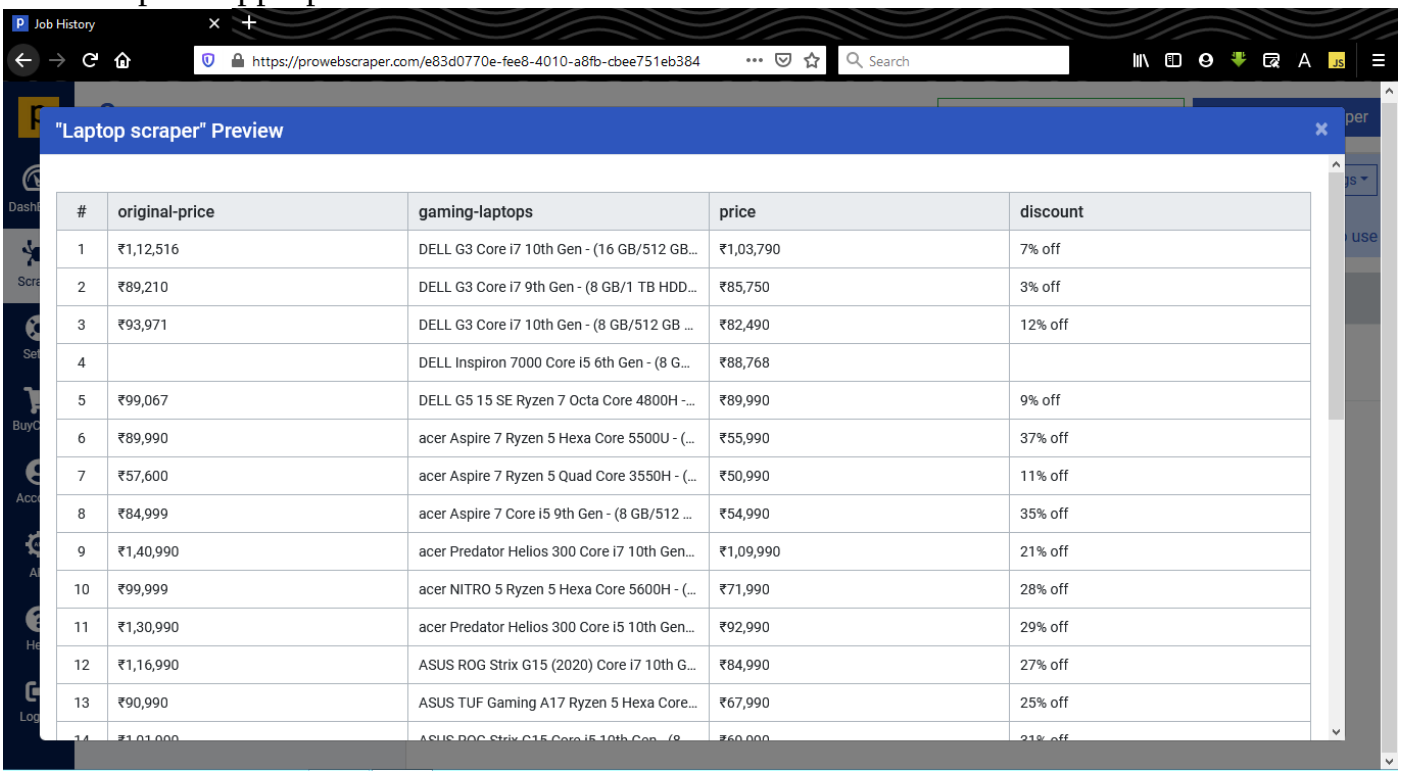


Fig9.1 - After selecting appropriate attributes to scrap, we can run the scrapper to scrap the appropriate data



#	original-price	gaming-laptops	price	discount
1	₹1,12,516	DELL G3 Core i7 10th Gen - (16 GB/512 GB...	₹1,03,790	7% off
2	₹89,210	DELL G3 Core i7 9th Gen - (8 GB/1 TB HDD...	₹85,750	3% off
3	₹93,971	DELL G3 Core i7 10th Gen - (8 GB/512 GB ...	₹82,490	12% off
4		DELL Inspiron 7000 Core i5 6th Gen - (8 G...	₹88,768	
5	₹99,067	DELL G5 15 SE Ryzen 7 Octa Core 4800H -...	₹89,990	9% off
6	₹89,990	acer Aspire 7 Ryzen 5 Hexa Core 5500U - (...	₹55,990	37% off
7	₹57,600	acer Aspire 7 Ryzen 5 Quad Core 3550H - (...	₹50,990	11% off
8	₹84,999	acer Aspire 7 Core i5 9th Gen - (8 GB/512 ...	₹54,990	35% off
9	₹1,40,990	acer Predator Helios 300 Core i7 10th Gen...	₹1,09,990	21% off
10	₹99,999	acer NITRO 5 Ryzen 5 Hexa Core 5600H - (...	₹71,990	28% off
11	₹1,30,990	acer Predator Helios 300 Core i5 10th Gen...	₹92,990	29% off
12	₹1,16,990	ASUS ROG Strix G15 (2020) Core i7 10th G...	₹84,990	27% off
13	₹90,990	ASUS TUF Gaming A17 Ryzen 5 Hexa Core...	₹67,990	25% off

Fig9.2 - After scrapping is done, it allows to preview the scrapped data. It is clear that it has fetched significantly accurate data from that particular page

AIM: Solve real world problem using Data mining Techniques.

Team Members:
180170107030 – Gabu Siddharth
180170107033 – Gohil Chandrarajsinh
180170107046 – Kava Chirag
180170107048 – kosrekar Adarsh

Color Identification using K-Means

- This opens the doors for many superior applications such as searching for colors in a Search Engine, or looking for a piece of clothing that has a certain color in it.

Some Real-world Applications

- In self-driving car, to detect the traffic signals.
- Multiple colour detection is used in some industrial robots, to performing pick-and-place task in separating different colored objects.

First, we resize the image to the size 600 x 400. It is not required to resize it to a smaller size but we do so to lessen the pixels which'll reduce the time needed to extract the colors from the image. K-Means expects the input to be of two dimensions, so we use Numpy's reshape function to reshape the image data.

K-Means algorithm creates clusters based on the supplied count of clusters. In our case, it will form clusters of colors and these clusters will be our top colors. We then fit and predict on the same image to extract the prediction into the variable labels.

We use **Counter** to get count of all labels. To find the colors, we use ***clf.cluster_centers_***. The ***ordered_colors*** iterates over the keys present in count, and then divides each value by 255. We could have directly divided each value by 255 but that would have disrupted the order.

Next, we get the hex and rgb colors. As we divided each color by 255 before, we now multiply it by 255 again while finding the colors. If show_chart is True, we plot a pie chart with each pie chart portion defined using count.values(), labels as hex_colors and colors as ordered_colors. We finally return the rgb_colors which we'll use at a later stage.

```
from sklearn.cluster import KMeans,MiniBatchKMeans
import matplotlib.pyplot as plt
import numpy as np
import cv2
from collections import Counter
```

```
from skimage.color import rgb2lab, deltaE_cie76
import os

# On reading the color which is in RGB space, we return a string.
# {:02x} simply displays the hex value for the respective color.
def RGB2HEX(color):
    return "#{:02x}{:02x}{:02x}".format(int(color[0]), int(color[1]),
int(color[2]))

# method that will help us get an image into Python in the RGB space.
def get_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image

def get_colors(image,number_of_colors,show_chart):
    modified_image = cv2.resize(image, (400, 400), interpolation =
cv2.INTER_AREA)
    modified_image =
modified_image.reshape(modified_image.shape[0]*modified_image.shape[1], 3)
    clf = MiniBatchKMeans(n_clusters =
number_of_colors,max_iter=200,batch_size=200)
    #clf = KMeans(n_clusters = number_of_colors)
    labels = clf.fit_predict(modified_image)
    counts = Counter(labels)

    center_colors = clf.cluster_centers_
    # We get ordered colors by iterating through the keys
    ordered_colors = [center_colors[i] for i in counts.keys()]
    hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.keys()]
    rgb_colors = [ordered_colors[i] for i in counts.keys()]
    if (show_chart):
        plt.figure(figsize = (8, 6))
        plt.pie(counts.values(), labels = hex_colors, colors = hex_colors)
        print("Press Enter")
        plt.waitforbuttonpress()
        plt.close()
    return rgb_colors

image = get_image('F:\movie\color_extracter1.jpg')
plt.imshow(image)
print("Press Enter")
plt.waitforbuttonpress()
plt.close()
import time
start = time.time()
cO= get_colors(get_image('F:\movie\color_extracter1.jpg'), 8, True)
end = time.time()
print("Elapsed time is {} seconds.".format((end - start)))
```

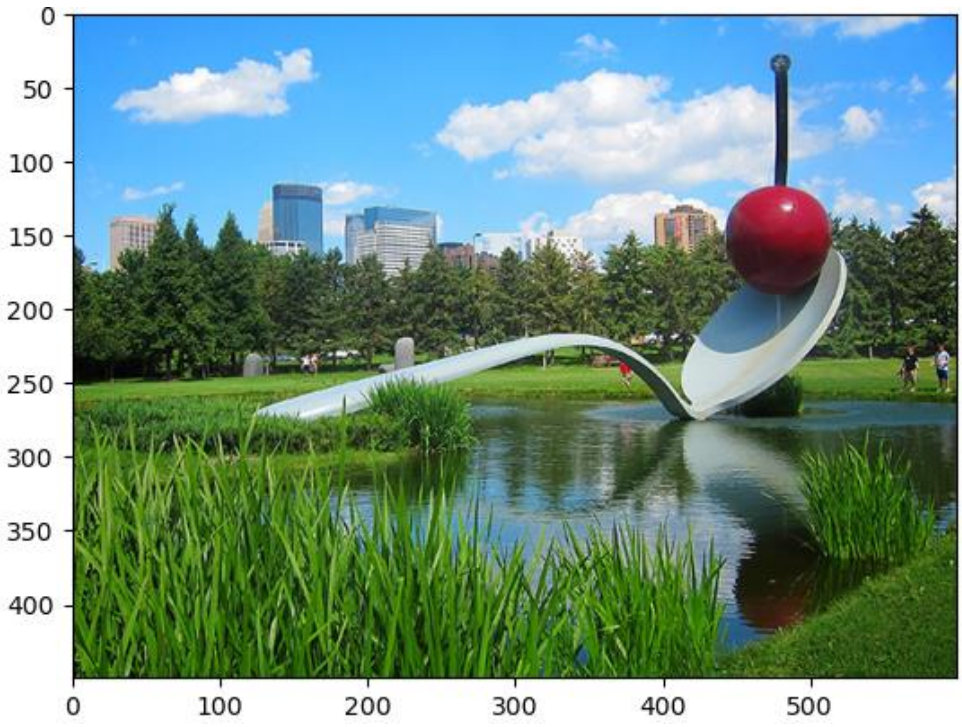


Fig10.1

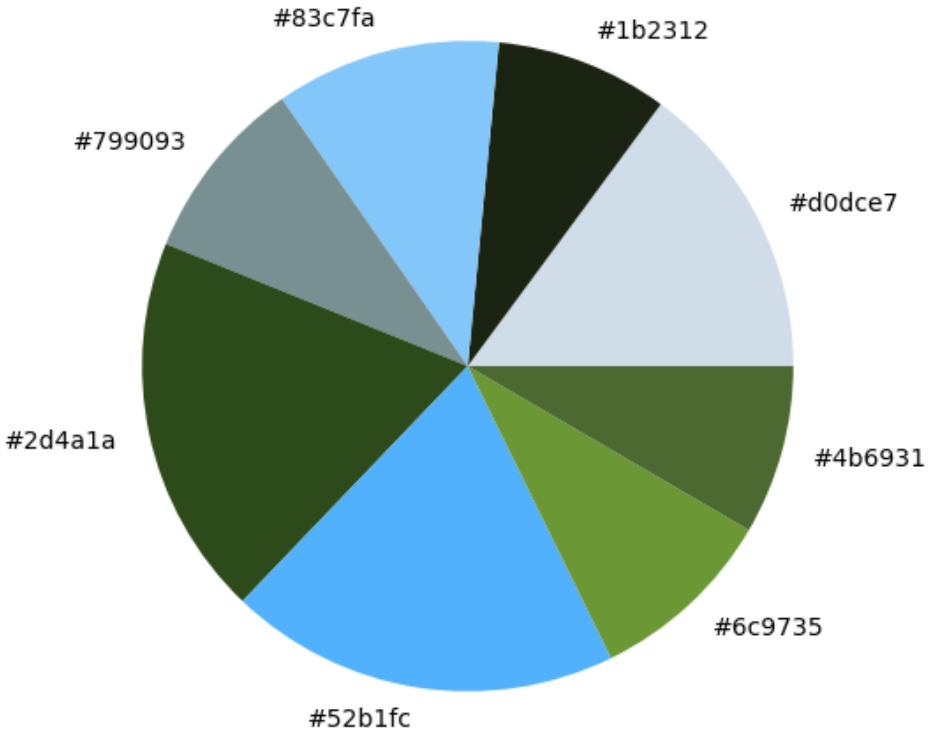


Fig10.2 – Extracted colors from Fig10.1

Fig10.3

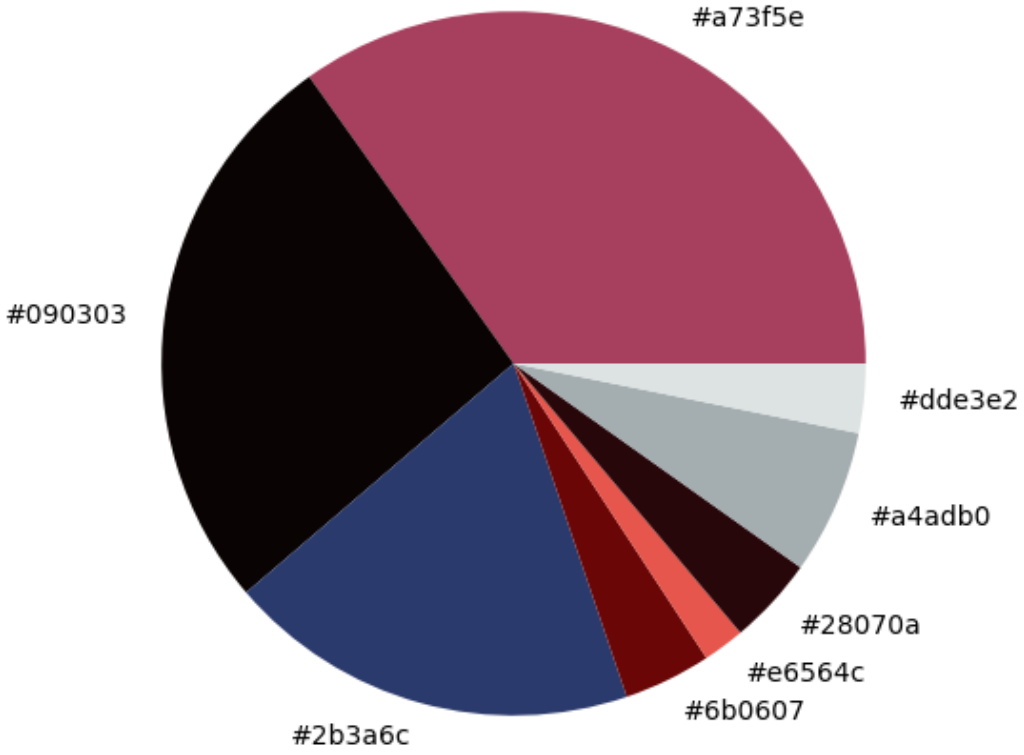


Fig10.4 - Extracted colors from Fig10.3

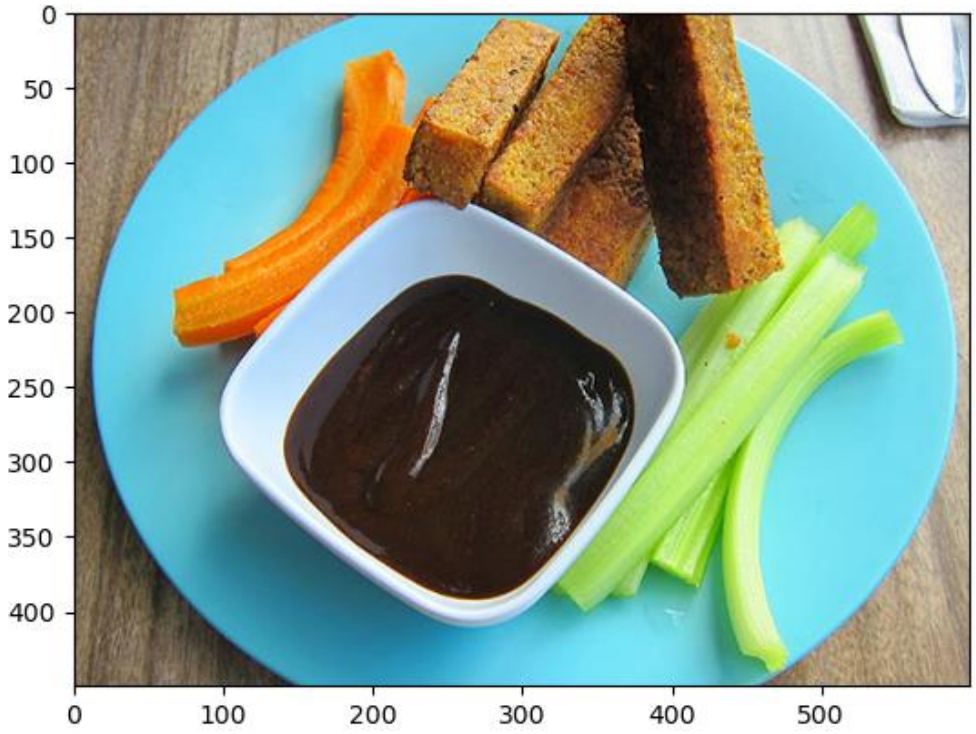


Fig 10.5

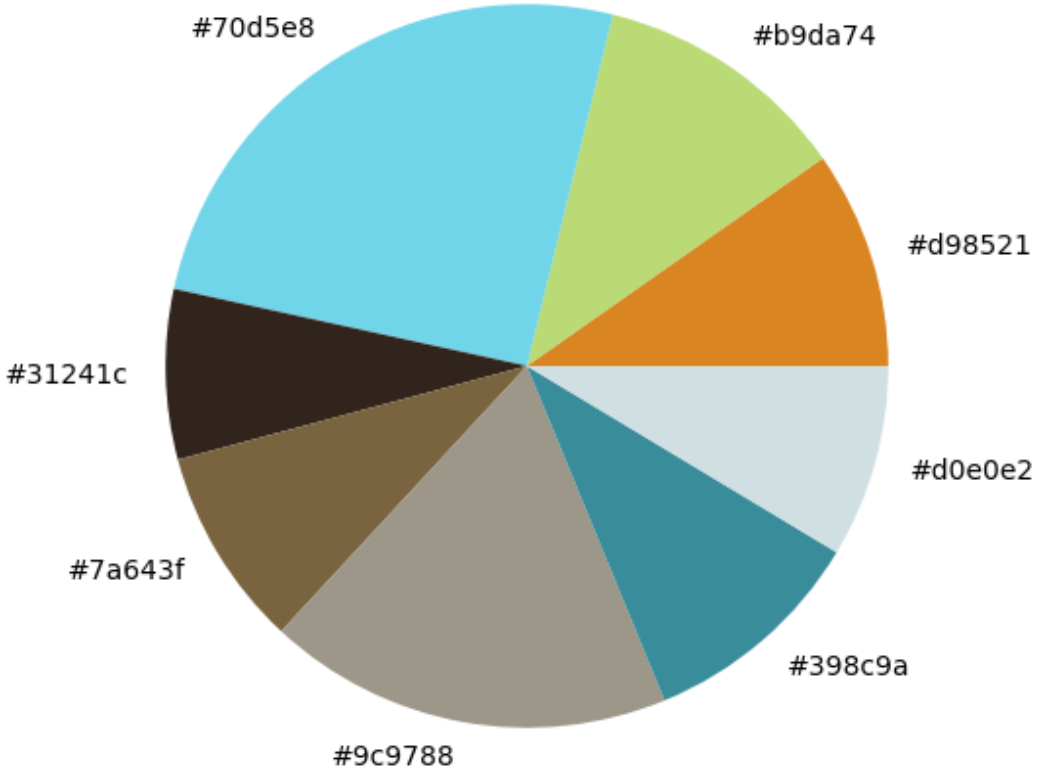


Fig 10.6 - Extracted colors from Fig10.5