1. **ETL (Extraction,Transformation,Loading)**

package pr1;

import java.sql.\*;

public class Bmi {

public void target() throws ClassNotFoundException, SQLException {

ResultSet rs;

Connection conn = null;

Class.*forName*("org.postgresql.Driver");

conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5433/postgres", "postgres", "matrix");

Statement st = conn.createStatement();

Statement s1 = conn.createStatement();

Statement s2 = conn.createStatement();

Statement s3 = conn.createStatement();

rs = st.executeQuery("SELECT MAX(sno) FROM table1");

rs.next();

int n = Integer.*parseInt*(rs.getString(1));

int i = 1;

System.*out*.println("the max is" + n);

ResultSet rs1 = st.executeQuery("SELECT \* FROM table1");

while (rs1.next()) {

int sno = Integer.*parseInt*(rs1.getString(1));

String sname = rs1.getString(2);

char c = rs1.getString(3).charAt(0);

String gender;

if (c == 'M' || c == 'm') {

gender = "male";

} else {

gender = "female";

}

double f = Float.*parseFloat*(rs1.getString(4));

double k = f \* 0.3048;

k = k \* k;

f = f \* 30.48;

double f1 = Float.*parseFloat*(rs1.getString(5));

double w;

w = f1;

f1 = f1 \* 1000;

double bmi = w / k;

System.*out*.println(sno + sname + gender + f + f1);

s1.execute("INSERT INTO integratedtable VALUES(" + sno + ",'" + sname + "','" + gender + "'," + f + "," + f1 + "," + bmi + ")");

}

ResultSet rs2 = s3.executeQuery("SELECT \* FROM table2");

while (rs2.next()) {

int sno = Integer.*parseInt*(rs2.getString(1));

String sname = rs2.getString(2);

int c = Integer.*parseInt*(rs2.getString(3));

String gender;

if (c == 1) {

gender = "male";

} else {

gender = "female";

}

sno = sno + n;

float f = Float.*parseFloat*(rs2.getString(5));

double h = (f \* f) / 10000;

double f1 = Float.*parseFloat*(rs2.getString(6));

double w2 = f1 \* 0.453592;

f1=f1\*453.592;

double bmi = w2 / h;

System.*out*.println(sno + sname + gender + f + f1);

s2.execute("INSERT INTO integratedtable VALUES(" + sno + ",'" + sname + "','" + gender + "'," + f + "," + f1 + "," + bmi + ")");

}

}

public static void main(String args[]) throws ClassNotFoundException, SQLException {

Bmi q = new Bmi();

q.target();

}

}

**Input Data:**

Table1

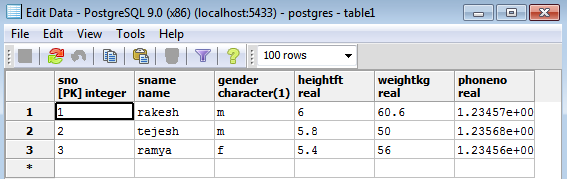
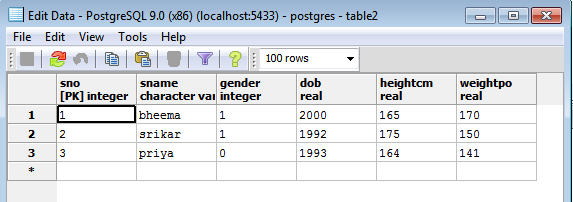
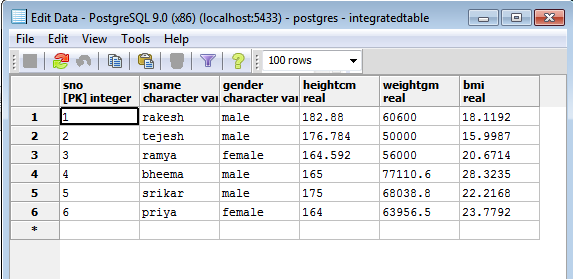


Table2



**Output:**



1. **DATA CLEANING (MISSING VALUES)**

package pr2;

import java.sql.\*;

public class MissingMean

{

public static void main(String args[])

{

Connection con=null;

ResultSet rs=null;

PreparedStatement pst=null;

Statement st=null;

int count=0,i=0;

float sum=0,avg;

float h[],w[];

try{

Class.*forName*("org.postgresql.Driver");

con=DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

st=con.createStatement();

rs=st.executeQuery("select count(\*) from msimp1 where wt<>0.0");

if(rs.next())

count=rs.getInt(1);

System.*out*.println(count);

h=new float[count];

w=new float[count];

rs=st.executeQuery("select \* from msimp1 where wt<>0.0");

while(rs.next())

{

w[i]=rs.getFloat("wt");

h[i]=rs.getFloat("ht");

sum=sum+w[i];

System.*out*.println(i+":"+w[i]+"-"+h[i]);

i++;

}

avg=sum/i;

rs=st.executeQuery("select \* from msimp1");

while(rs.next())

{

float wt=rs.getFloat("wt");

float ht=rs.getFloat("ht");

if(wt==0.0)

{

pst=con.prepareStatement("Update msimp1 set wt=? where wt =0.0 ");

pst.setFloat(1,avg);

int ps=pst.executeUpdate();

}

}

System.*out*.println("operation done");

st.close();

pst.close();

con.close();

}

catch(Exception e)

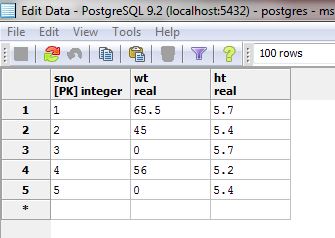
{

e.printStackTrace();

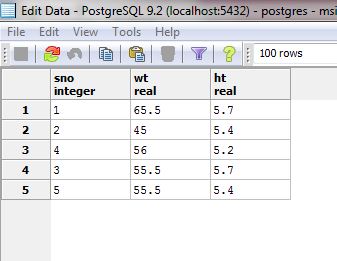
}

}}

**Input Data:**



**Output Data:**



1. **NORMALIZATION**
2. **MIN-MAX NORMALIZATION**

package pr3;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class MinMax {

public static void main(String[] args) {

// TODO Auto-generated method stub

Connection c=null;

Statement st1=null,st2=null,st3=null,st4=null;

int minRange=0;

int maxRange=1;

try{

Class.*forName*("org.postgresql.Driver");

c= DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

System.*out*.println("opened database successfully");

st1= c.createStatement();

ResultSet rsmin = st1.executeQuery("SELECT min(column1) from table5;");

rsmin.next();

int min = rsmin.getInt("min");

//System.out.println(min);

ResultSet rsmax = st1.executeQuery("SELECT max(column1) from table5;");

rsmax.next();

int max = rsmax.getInt("max");

st2= c.createStatement();

ResultSet rs = st2.executeQuery("SELECT \* from table5");

st3= c.createStatement();

while(rs.next())

{

int value = rs.getInt("column1");

double nValue = (((double)(value-min)/(max-min))\*(maxRange-minRange))+minRange;

st3.executeUpdate("UPDATE table5 set normalised\_values="+nValue+"where column1 = "+value);

}

st4= c.createStatement();

ResultSet rsfin = st4.executeQuery("SELECT \* from table5");

while(rsfin.next()){

System.*out*.println("column1="+rsfin.getInt("column1")+" normalised\_value = "+rsfin.getDouble("normalised\_values"));

}

rs.close();

st1.close();

c.close();

rsmin.close();

rsmax.close();

rsfin.close();

}

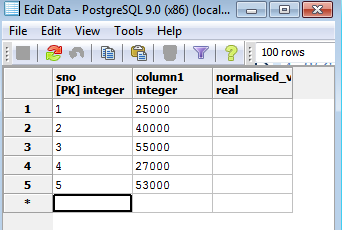
catch(Exception e){

e.printStackTrace();

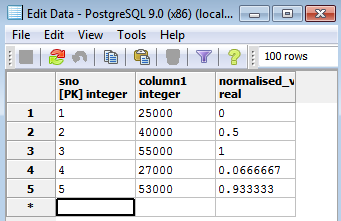
}

}}

**Input Data:**



**Output Data:**

****

1. **Z-SCORE NORMALIZATION**

package pr3;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class zscore {

public static void main(String[] args) {

// TODO Auto-generated method stub

Connection c = null;

Statement st=null, st2=null, st3=null, st4=null,st5=null,st6=null;

try{

Class.*forName*("org.postgresql.Driver");

c = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

System.*out*.println("database opened successfully");

st = c.createStatement();

//st1 = c.createStatement();

st2 = c.createStatement();

st3 = c.createStatement();

st4 = c.createStatement();

st5 = c.createStatement();

st6 = c.createStatement();

ResultSet rs = st.executeQuery("SELECT avg(col1) from zscore");

rs.next();

//mean of values

double mean = (double) rs.getInt("avg");

System.*out*.println("mean = "+mean);

/\* calculating standard deviation \*/

//count the number of rows

ResultSet rs1 = st2.executeQuery("SELECT count(col1) from zscore");

rs1.next();

int count = rs1.getInt("count");

ResultSet rs2 = st3.executeQuery("SELECT col1 from zscore");

double total = 0;

while(rs2.next())

{

int value = rs2.getInt("col1");

double x = (double) (value-mean) ;

x = x\*x;

total = total + x;

}

double sd = Math.*sqrt*((double)(total/count)); // standard deviation

System.*out*.println("standard deviation = "+sd);

ResultSet rs3 = st4.executeQuery("SELECT col1 from zscore");

while(rs3.next())

{

double zscore = Math.*abs*((double)(rs3.getInt("col1") - mean))/sd;

st5.executeUpdate("UPDATE zscore set col2 = "+zscore+"where col1 = "+rs3.getInt("col1"));

}

ResultSet rs4 = st6.executeQuery("SELECT \* from zscore");

while(rs4.next())

System.*out*.println("col1 = "+rs4.getInt("col1")+ " col2 = "+rs4.getDouble("col2"));

}

catch(Exception e){

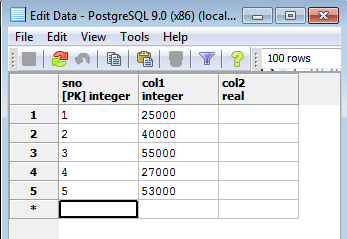
System.*out*.println("caught exception "+e);

}

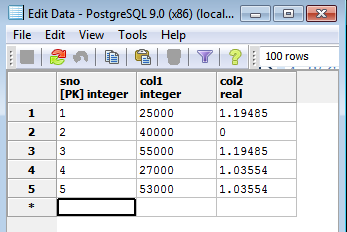
}

}

**Input Data:**



**Output Data:**

****

1. **DECIMAL SCALING**

package pr3;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.sql.Connection;

public class decimalscaling {

public static void main(String[] args) {

// TODO Auto-generated method stub

Connection c = null;

Statement st = null, st1=null, st2=null,st3=null;

try{

Class.*forName*("org.postgresql.Driver");

c=DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

System.*out*.println("opened database successfully");

st = c.createStatement();

ResultSet rs = st.executeQuery("SELECT MAX(col1) from decimal\_scaling");

rs.next();

int max = rs.getInt("MAX");

int count=0;

while(max!=0)

{

count++;

max=max/10;

}

st1 = c.createStatement();

st2 = c.createStatement();

ResultSet rs1 = st1.executeQuery("select col1 from decimal\_scaling");

while(rs1.next())

{

int x = rs1.getInt("col1");

double value = (double) (x / Math.*pow*(10,count));

st2.executeUpdate("UPDATE decimal\_scaling set col2 = "+ value + "where col1 = "+ x);

}

st3 = c.createStatement();

ResultSet rs2 = st3.executeQuery("SELECT \* from decimal\_scaling");

while(rs2.next())

System.*out*.println("col1 = "+ rs2.getInt("col1")+ " col2 = "+ rs2.getDouble("col2"));

rs.close();

st.close();

c.close();

}

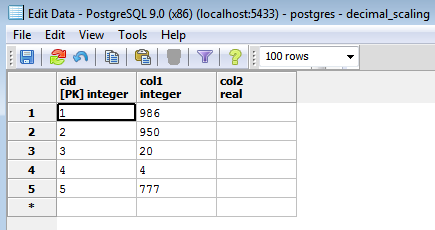
catch(Exception e){

System.*out*.println("caught exeception "+ e);

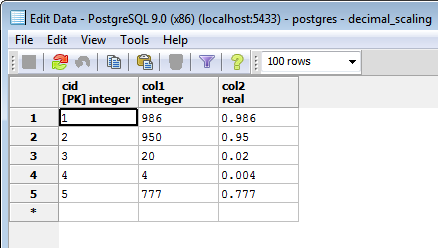
}

}}

**Input Data:**

****

**Output Data:**

****

1. **ASSOCIATION**
2. **APRIORI ALGORITHM**

package pr4;

import java.sql.\*;

import java.util.Scanner;

public class apriori {

public static void main(String[] args){

Connection conn=null;

int threshold,count1=0,count2=0,count3=0;

int c[]=new int[6];

int c1[][]=new int[10][10];

int c2[][][]=new int[10][10][10];

int c3[][][][]=new int[10][10][10][10];

try

{

System.*out*.println("\nAttempting to load JDBC Driver....");

Class.*forName*("org.postgresql.Driver");

System.*out*.println("JDBC Driver loaded...");

System.*out*.println("Connecting to database...");

conn=DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

System.*out*.println("Database connection established");

}//end of try

catch (Exception sqle)

{System.*out*.println("Unable to load driver...");}

try{

System.*out*.println("enter the minimum threshold");

Scanner sc=new Scanner(System.*in*);

threshold=sc.nextInt();

Statement s1=conn.createStatement();

Statement s2=conn.createStatement();

ResultSet rs1= s1.executeQuery("SELECT \* FROM apriori" );

/\*to count the itemsets\*/

while(rs1.next())

{

if(Integer.*parseInt*(rs1.getString(1))==1)

c[0]++;

if(Integer.*parseInt*(rs1.getString(2))==1)

c[1]++;

if(Integer.*parseInt*(rs1.getString(3))==1)

c[2]++;

if(Integer.*parseInt*(rs1.getString(4))==1)

c[3]++;

if(Integer.*parseInt*(rs1.getString(5))==1)

c[4]++;

}

int k=0,p;

while(k<5){

if(c[k]>=threshold)

{ p=k+1;

count1++;

System.*out*.println("itemset"+p+" :"+c[k]);

}

k++;

}

System.*out*.println("count1 is"+count1);

rs1.close();

rs1= s1.executeQuery("SELECT \* FROM apriori" );

if(count1!=0){

while(rs1.next()){

for(int l=1;l<5;l++)

{

for(int m=l+1;m<=5;m++)

{

if((Integer.*parseInt*(rs1.getString(l)))==1&&(Integer.*parseInt*(rs1.getString(m))==1))

{

c1[l][m]++;

}

}

}

}

}

for(int l=1;l<5;l++)

{

for(int m=l+1;m<=5;m++)

{

if(c1[l][m]>=threshold)

{

count2++;

System.*out*.println("itemset"+l+" "+m+" :"+c1[l][m]);

}

}

}

rs1.close();

rs1= s1.executeQuery("SELECT \* FROM apriori" );

if(count2!=0){

while(rs1.next()){

for(int l=1;l<4;l++)

{

for(int m=l+1;m<=4;m++)

{

for(int n=m+1;n<=5;n++)

{

if((Integer.*parseInt*(rs1.getString(l)))==1&&(Integer.*parseInt*(rs1.getString(m))==1&&(Integer.*parseInt*(rs1.getString(n))==1)))

{

c2[l][m][n]++;

}

}

}

}

}

}

for(int l=1;l<4;l++)

{

for(int m=l+1;m<=4;m++)

{

for(int n=m+1;n<=5;n++)

{

if(c2[l][m][n]>=threshold)

{

if((c1[l][m]>=threshold)&&(c1[m][n]>=threshold)&&(c1[m][n]>=threshold)){

System.*out*.println("itemset"+l+" "+m+" "+n+": "+c2[l][m][n]);

count3++;

}

}

}

}

}

System.*out*.println(count3);

rs1.close();

rs1= s1.executeQuery("SELECT \* FROM apriori" );

if(count3!=0){

while(rs1.next()){

for(int l=1;l<3;l++)

{

for(int m=l+1;m<=3;m++)

{

for(int n=m+1;n<=4;n++)

{

for(int o=n+1;o<=5;o++){

if((Integer.*parseInt*(rs1.getString(l)))==1&&(Integer.*parseInt*(rs1.getString(m))==1&&(Integer.*parseInt*(rs1.getString(n))==1))&&(Integer.*parseInt*(rs1.getString(o))==1))

{

c3[l][m][n][o]++;

}

}

}

}

}

}

}

for(int l=1;l<3;l++)

{

for(int m=l+1;m<=3;m++)

{

for(int n=m+1;n<=4;n++)

{

for(int o=n+1;o<=5;o++){

if(c3[l][m][n][o]>=threshold)

{

if((c2[l][m][n]>=threshold)&&(c2[m][n][o]>=threshold)){

System.*out*.println("itemset"+l+" "+m+" "+n+" "+o+":"+c3[l][m][n][o]);

}

}

}

}

}

}

}

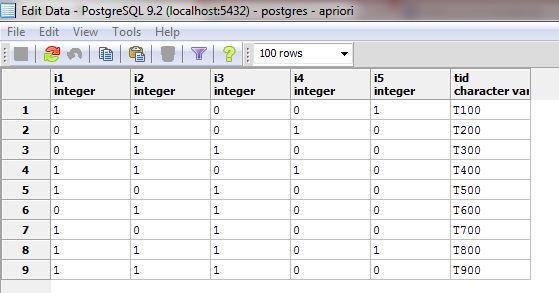
catch(Exception e){

e.printStackTrace();}

}

}

**Input Data:**



**Output Data:**

enter the minimum threshold

2

itemset1 :6

itemset2 :7

itemset3 :6

itemset4 :2

itemset5 :2

count1 is5

itemset 1 2 :4

itemset 1 3 :4

itemset 1 5 :2

itemset 2 3 :4

itemset 2 4 :2

itemset 2 5 :2

itemset 1 2 3: 2

itemset 1 2 5: 2

1. **ECLAT (EQUIVALENCE CLASS TRANSFORMATION)**

package pr4;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.Scanner;

interface finals

{

public final int *transactions*=9;

public final int *items*=5;

}

public class eclat1 implements finals

{

public static void main(String[] args)

{

int data[][]=new int[*items*][*transactions*];

int count\_data[]=new int[*items*];

Connection con;

Statement stmt;

ResultSet rs;

System.*out*.println("enter min support");

Scanner scan=new Scanner(System.*in*);

int min\_sup=scan.nextInt();

scan.close();

try

{

Class.*forName*("org.postgresql.Driver");

con=DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

stmt=con.createStatement();

rs=stmt.executeQuery("select \* from eclat;");

int i=0;

int j=0;

while(rs.next())

{

int k=1;

int num\_of\_transactions=0;

for(j=0;j<*transactions* && k<=9;j++,k++)

{

if(rs.getInt(k)!=0)

{

num\_of\_transactions++;

data[i][j]=1;

}

}

count\_data[i]=num\_of\_transactions;

i++;

}

//1st level of itemsets-inserting into 1st table

for(i=0;i<*items*;i++)

{

String trans="";

//int count=0;

for(j=0;j<*transactions*;j++)

{

if(data[i][j]==1)

{

trans=trans+" "+(j+1);

//System.out.println(trans);

// count++;

}

}

//if(count>=min\_sup)

if(count\_data[i]>=min\_sup)

{

stmt.executeUpdate("insert into eclat1 values('i"+(i+1)+"','"+trans+"');");

}

}

//2nd level of itemsets-inserting into 2nd table

SecondCount sc[]=new SecondCount[10];//5c2

for(i=0;i<10;i++)

{

sc[i]=new SecondCount();

sc[i].count=0;

sc[i].ids="";

}

int s=0;

for(i=0;i<*items*;i++)

{

for(j=i+1;j<*items*;j++)

{

sc[s].ids=Integer.*toString*(i+1)+(j+1);

String trans="";

String items="i"+(i+1)+" i"+(j+1);

int count=0;

for(int k=0;k<*transactions*;k++)

{

if(data[i][k]==1 && data[j][k]==1 && count\_data[i]>=min\_sup && count\_data[j]>=min\_sup)

{

trans=trans+" "+(k+1);

count++;

}

}

sc[s++].count=count;

if(count>=min\_sup)

{

stmt.executeUpdate("insert into eclat2 values('"+items+"','"+trans+"');");

}

}

}

/\*for(i=0;i<10;i++)

{

System.out.println(sc[i].ids);

System.out.println(sc[i].count);

}\*/

//3rd level of itemsets-inserting into 3rd table

for(i=0;i<*items*;i++)

{

for(j=i+1;j<*items*;j++)

{

for(int k=j+1;k<*items*;k++)

{

String trans="";

String items="i"+(i+1)+" i"+(j+1)+" i"+(k+1);

int count=0;

for(int a=0;a<*transactions*;a++)

{

if(data[i][a]==1 && data[j][a]==1 && data[k][a]==1 && SecondCount.*countInfo*(sc,10,(i+1),j+1,k+1) )

{

trans=trans+" "+(a+1);

count++;

}

}

if(count>=min\_sup)

{

stmt.executeUpdate("insert into eclat3 values('"+items+"','"+trans+"');");

}

}

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

class SecondCount

{

String ids;

int count;

static boolean countInfo(SecondCount sc[],int size,int i,int j,int k)

{

int a=0;

boolean r1,r2,r3;

r1=r2=r3=false;

String s1,s2,s3;

s1=Integer.*toString*(i)+j;

s2=Integer.*toString*(i)+k;

s3=Integer.*toString*(j)+k;

for(a=0;a<size;a++)

{

if(sc[a].ids.equalsIgnoreCase(s1))

{

r1=true;

}

if(sc[a].ids.equalsIgnoreCase(s2))

{

r2=true;

}

if(sc[a].ids.equalsIgnoreCase(s3))

{

r3=true;

}

}

if(r1 && r2 && r3)

return true;

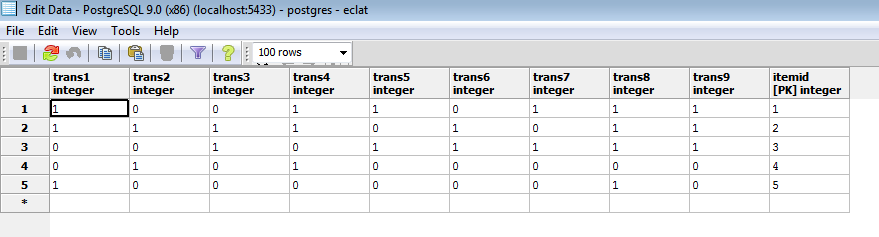
else

return false;

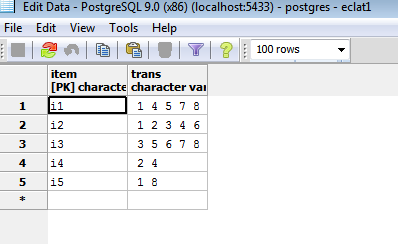
}

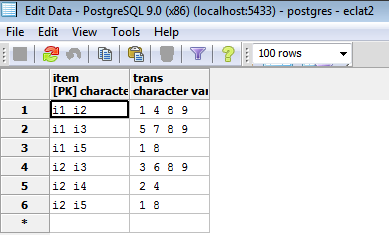
}

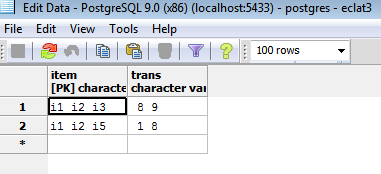
**Input Data:**

****

**Output Data:**

****

****

****

1. **CLUSTERING (K-Means)**

package pr4;

import java.sql.\*;

import java.lang.\*;

class clustering

{

public clustering()

{ }

int item[]=new int[30];int fat[]=new int[30];//Main array

Connection con=null;

ResultSet rs=null;

public void getdata()

{

int i=0;

String s1,s2;

try

{

System.out.println("\nAttempting to load JDBC Driver....");

Class.forName("org.postgresql.Driver");

System.out.println("JDBC Driver loaded...");

System.out.println("Connecting to database...");

con=DriverManager.getConnection("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

System.out.println("Database connection established");

}//end of try

catch (Exception sqle)

{System.out.println("Unable to load driver...");}

try

{

String queryString=("SELECT \* FROM food");

Statement stmt=con.createStatement();

rs=stmt.executeQuery(queryString);

while (rs.next())

{

s1=rs.getString("Item");

s2=rs.getString("Fat");

item[i]=Integer.parseInt(s1.trim());

fat[i]=Integer.parseInt(s2.trim());

i++;

} //end of while

}//end of try

catch (SQLException sqle)

{System.out.println("Some SQL error occured.");}

try

{if(con!=null)

{con.close();}

System.out.println("Connection to DB closed..Data Retrieved Successfully!");}

catch(Exception e)

{}

}//end of function

public void cluster()

{int m1,m2,m3;int i;int d1=0;int d2=0;int d3=0;int a=1;int b=1;int c=1;

int c11[]=new int[20];int c12[]=new int[20];//Cluster 1

int c21[]=new int[20];int c22[]=new int[20];//Cluster 2

int c31[]=new int[20];int c32[]=new int[20];//Cluster 3

c11[0]=item[0];c12[0]=fat[0]; //Randomly place one item in each cluster

c21[0]=item[1];c22[0]=fat[1];

c31[0]=item[2];c32[0]=fat[2];

m1=c12[0];m2=c22[0];m3=c32[0];//Initial Mean value of each cluster

for(i=3;i<20;i++)

{

d1=Math.abs(m1-fat[i]);

d2=Math.abs(m2-fat[i]);

d3=Math.abs(m3-fat[i]);

if(d1<=d2 && d1<=d3)

{c11[a]=item[i];

c12[a]=fat[i];

m1=(c12[a]+m1)/2;

a++;}

if(d2<=d1 && d2<=d3)

{c21[b]=item[i];

c22[b]=fat[i];

m2=(c22[b]+m2)/2;

b++;}

if(d3<=d1 && d3<=d2)

{c31[c]=item[i];

c32[c]=fat[i];

m3=(c32[c]+m3)/2;

c++;}

}//end of for...

System.out.println("\n Data is classified into 3 clusters as follows..");

System.out.println("\nCluster 1");

System.out.println("----------");

System.out.println("\nItem Fat\n");

for(i=0;i<10;i++)

{

if(c12[i]==0)

{break;}

System.out.print(c11[i]);

System.out.print(" ");

System.out.print(c12[i]);

System.out.println("");

}

System.out.println("\nCluster 2");

System.out.println("----------");

System.out.println("\nItem Fat\n");

for(i=0;i<10;i++)

{

if(c22[i]==0)

{break;}

System.out.print(c21[i]);

System.out.print(" ");

System.out.print(c22[i]);

System.out.println("");

}

System.out.println("\nCluster 3");

System.out.println("----------");

System.out.println("\nItem Fat\n");

for(i=0;i<10;i++)

{

if(c32[i]==0)

{break;}

System.out.print(c31[i]);

System.out.print(" ");

System.out.print(c32[i]);

System.out.println("");

}

}//end of function

public static void main(String args[])

{

clustering t=new clustering();

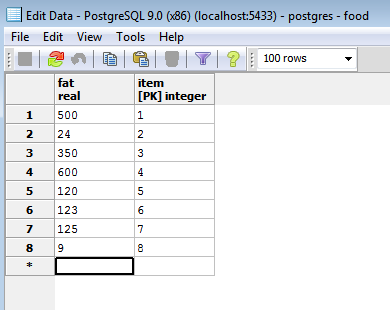
t.getdata();

t.cluster();

}//end of main

}//end of class

**Input Data:**



**Output Data:**

Data is classified into 3 clusters as follows..

Cluster 1

----------

Item Fat

1 500

4 600

Cluster 2

----------

Item Fat

2 24

5 120

6 123

7 125

8 9

Cluster 3

----------

Item Fat

1. 350
2. **CLASSIFICATION**
3. **BAYESIAN CLASSIFICATION**

package pr4;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.lang.Math.\*;

public class bayes {

public static void main( String args[] )

{

Connection c = null;

Statement stmt1 ;

Statement stmt2;

try {

Class.forName("org.postgresql.Driver");

c = DriverManager

.getConnection("jdbc:postgresql://localhost:5432/postgres","postgres", "cse");

System.out.println("Opened database successfully");

stmt1 = c.createStatement();

stmt2 = c.createStatement();

//stmt3 = c.createStatement();

double c3=0;

double pc1,pc2;

ResultSet rs = stmt1.executeQuery( "SELECT \* FROM bayes;" );

while(rs.next())

c3++;

ResultSet rs2= stmt2.executeQuery("SELECT \* FROM bayes where computer='y'");

double c1=0;

while(rs2.next())

c1++;

System.out.println("\n"+c1);

System.out.println("\n"+c3);

pc1=c1/c3;

pc2=1-pc1;

ResultSet rsc1,rsc2;

double rcc1=0,rcc2=0,rpc1=0,rpc2=0;

rsc1=stmt2.executeQuery( "SELECT \* FROM bayes WHERE age<35 AND computer='y';" );

while(rsc1.next())

rcc1++;

rcc1=rcc1/c1;

rsc2=stmt2.executeQuery( "SELECT \* FROM bayes WHERE age<35 AND computer='n';" );

while(rsc2.next())

rcc2++;

rcc2=rcc2/(c3-c1);

rsc1=stmt2.executeQuery( "SELECT \* FROM bayes WHERE income>='30000' AND computer='y';" );

while(rsc1.next())

rpc1++;

rpc1=rpc1/c1;

rsc2=stmt2.executeQuery( "SELECT \* FROM bayes WHERE income>='30000' AND computer='n';" );

while(rsc2.next())

rpc2++;

rpc2=rpc2/(c3-c1);

double hf,ha;

hf=rcc1\*rpc1;

ha=rcc2\*rpc2;

System.out.println("\n"+hf+" "+ha);

if(hf>ha)

System.out.println("\n Buys computer");

else

System.out.println("\n Does not buy computer");

rs.close();

stmt2.close();

c.close();

} catch ( Exception e ) {

System.err.println( e.getClass().getName()+": "+ e.getMessage() );

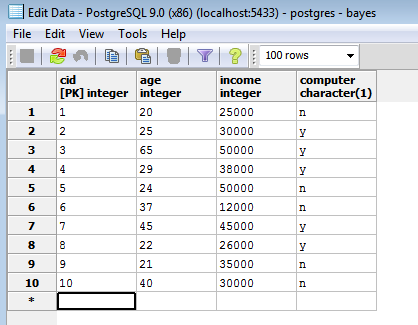
System.exit(0);

}

System.out.println("Operation done successfully");

}}

**Input Data:**

****

**Output Data:**

Opened database successfully

5.0

10.0

0.48 0.36

Buys computer

Operation done successfully

1. **K-NEAREST NEIGHBOR CLASSIFICATION**

package pr4;

import java.sql.\*;

import java.util.Scanner;

public class knearest {

Connection conn;

knearest() throws ClassNotFoundException, SQLException

{

Class.*forName*("org.postgresql.Driver");

conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","cse");

}

public void algo() throws SQLException

{

Scanner sc=new Scanner(System.*in*);

System.*out*.println("enter the colour");

String colour=sc.next();

System.*out*.println("enter the income");

int income=sc.nextInt();

System.*out*.println("enter the age");

String age=sc.next();

System.*out*.println("enter the value of k");

int k=sc.nextInt();

Statement s1=conn.createStatement();

Statement s2=conn.createStatement();

ResultSet rs1= s1.executeQuery("SELECT MAX(income) FROM table9" );

rs1.next();

int max=Integer.*parseInt*(rs1.getString(1));

ResultSet rs2= s1.executeQuery("SELECT MIN(income) FROM table9" );

rs2.next();

int min=Integer.*parseInt*(rs2.getString(1));

System.*out*.println(max+" "+min);

income=(income-min)/(max-min);

ResultSet rs3= s1.executeQuery("SELECT \* FROM table9" );

double f = 0;

double f1[]=new double[10];

String s[]=new String[10];

int i = 0;

int c ,c3;

while(rs3.next())

{f=0;

c=0;

c3=0;

if(colour.equals(rs3.getString(2)))

{

c=1;

}

if(age.equals(rs3.getString(4)))

{

c3=1;

}

if(c==0)

{ f=f+1;}

if(c3==0)

{ f=f+1;}

float income1=Integer.*parseInt*(rs3.getString(3));

income1=(income1-min)/(max-min);

float f2=income-income1;

System.*out*.println(f2);

f2=f2\*f2;

f=f+f2;

f1[i]=Math.*sqrt*(f);

s[i]=rs3.getString(5);

i++;

}

for(int j=0;j<10;j++){

System.*out*.println(f1[j]);}

//sorting algorithm

int p,j;

double temp;

String temp2;

for(i=0;i<f1.length;i++){

for(j=i+1;j<f1.length;j++){

if(f1[i]>f1[j]){

temp=f1[i];

temp2=s[i];

f1[i]=f1[j];

s[i]=s[j];

f1[j]=temp;

s[j]=temp2;

}

}

}

int c1 = 0,c2 = 0;

for( j=0;j<10;j++){

//System.out.println(f1[j]);

}

if(k==1)

{

// System.out.println("the class label is "+s[0]);

}

else

{

for(i=0;i<k;i++)

{

if(s[i].equals("yes"))

{

c1++;

}

if(s[i].equals("no"))

{

c2++;

}

}

if(c1>c2)

{

System.*out*.println("the class label is yes");

}

else{System.*out*.println("the class label is no"); }

}

}

public static void main(String args[]) throws SQLException, ClassNotFoundException

{

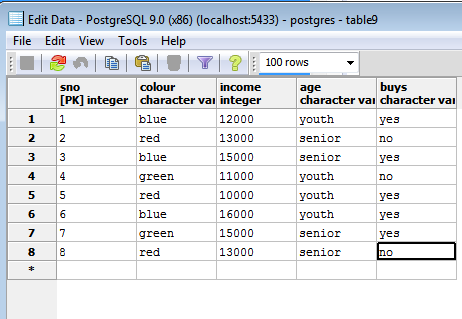
knearest kn=new knearest();

kn.algo();

}

}

**Input Data:**



**Output Data:**

enter the colour

red

enter the income

12000

enter the age

youth

enter the value of k

2

16000 10000

-0.33333334

-0.5

-0.8333333

-0.16666667

0.0

-1.0

-0.8333333

-0.5

1.0540925573162605

1.118033988749895

1.6414762922301045

1.0137937560704264

0.0

1.4142135623730951

1.6414762922301045

1.118033988749895

0.0

0.0

the class label is no

1. **WEKA**

* Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code.
* Weka contains tools for data pre-processing, classification, regression, clustering, association rules and visualization.
* It is also well-suited for developing new machine learning schemes.
* Weka can be found in weka website(Latest version 3.6): <http://www.cs.waikato.ac.nz/ml/weka/>

**Datasets in Weka**

* Each entry in a dataset is an instance of the java class: weka.core.Instance.
* Each instance consists of a number of attributes.

**Attributes**

* Attributes are of 5 types:

1.Nominal: one of a predefined list of values. Ex:red,green,blue

2.Numeric: A real or integer number.

3.String: Enclosed in “double quotes”.

4.Date

5.Relational

**ARFF Files**

* The external representation of an Instances class consists of:

->A header:Describes the attribute types.

->Data Section: Comma separated list of data

* Example for creation of .arff file:

@relation AllElectronics

@attribute colour {red,blue,green}

@attribute income real

@attribute age {youth,senior}

@attribute buys {yes,no}

@data

blue,12000,youth,yes

red,13000,senior,no

blue,15000,senior,yes

green,11000,youth,no

red,10000,youth,yes

blue,16000,youth,yes

green,15000,senior,yes

**Classifiers in Weka**

* Any .arff file is split into train and test set
* For Ex: Soybean-train.arff and Soybean-test.arff



1. **CLASSIFICATION**

It is necessary to provide a clear classification of data mining systems which may help users to distinguish between such systems and to identify them.

Data mining classification can be done in different ways:

1) Data mining can be classified according to the kinds of databases mined

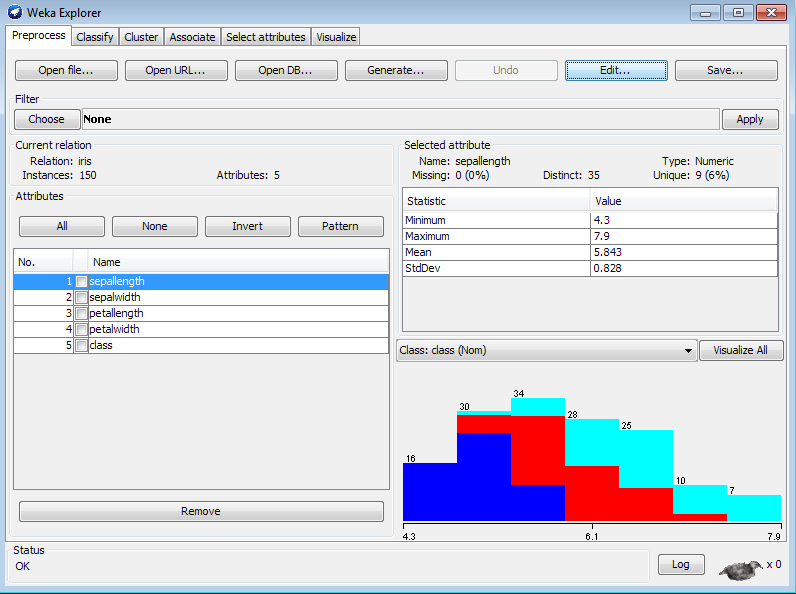
2) Data mining can be Classified according to the kinds of knowledge mined which is done based on the mining functionalities like characterization, discrimination etc…

3) We can also classify the data mining systems according to the kinds of techniques utilized, applications adapted.

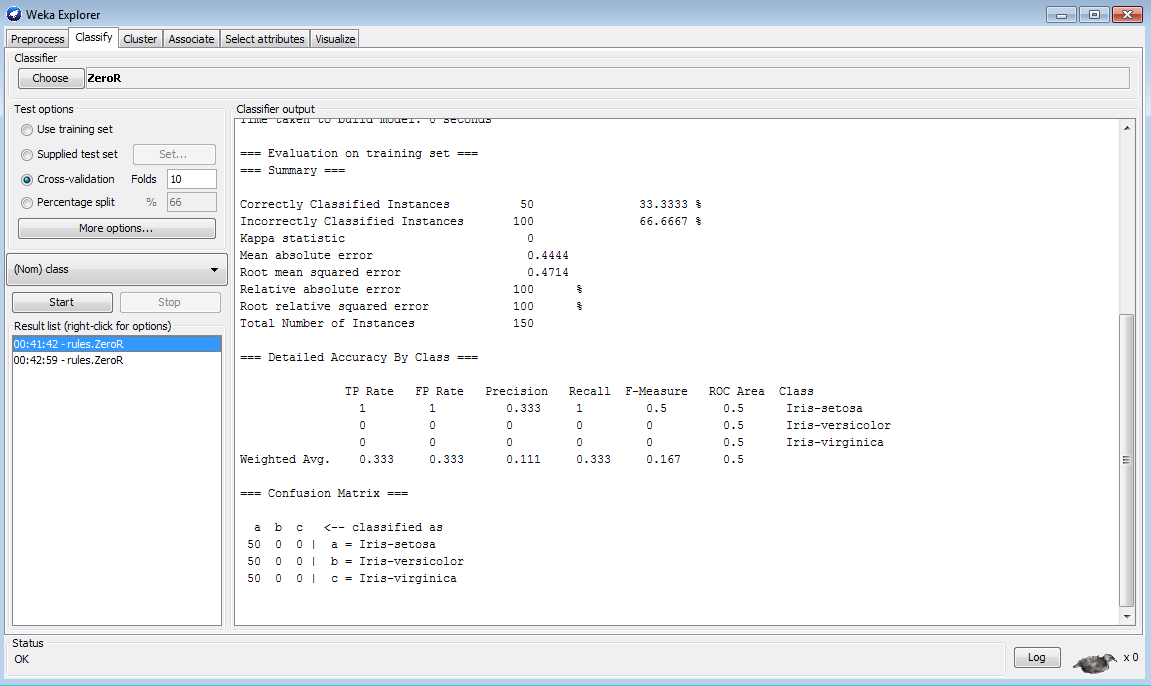
1. **ZeroR Algorithm**

Open the WEKA GUI Chooser from start menuall programs and click on the EXPLORER button.

Now click on the **Open File** button and choose the file named as “iris.arff”.

****

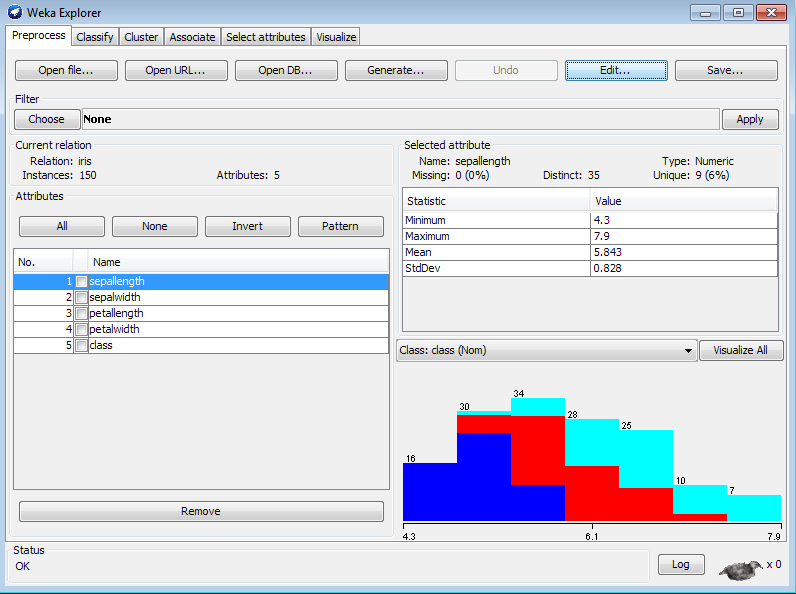
After loading the input choose the classify tab in the WEKA explorer window. Now select the “**use training set** “under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.

****

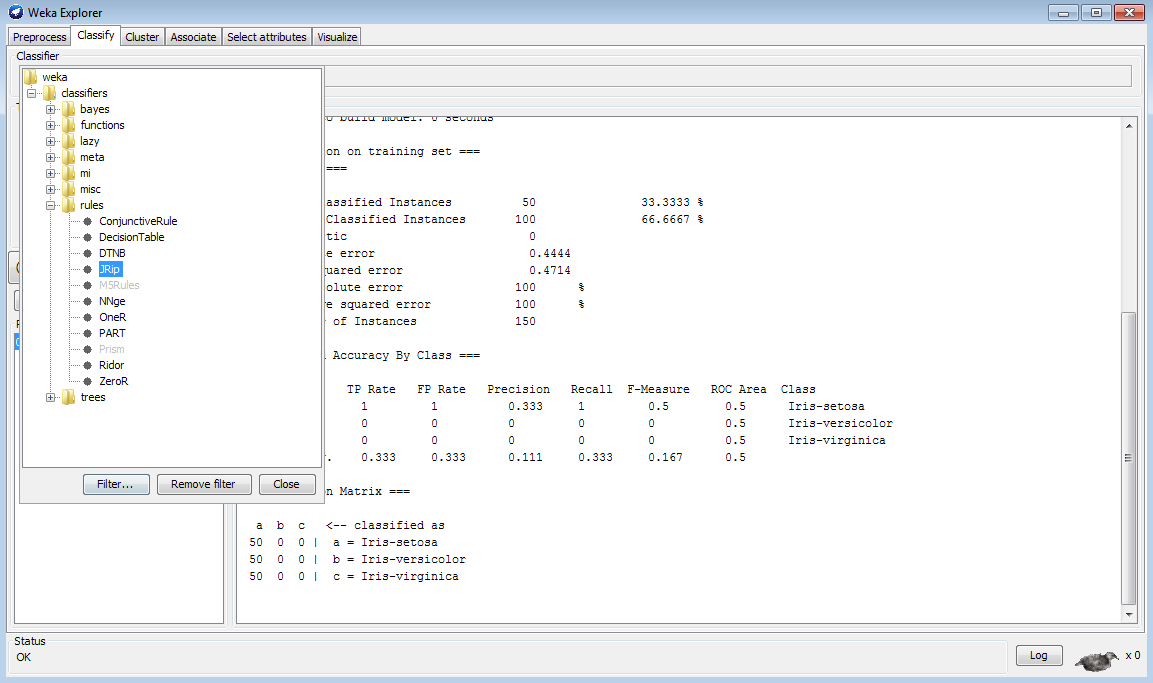
1. **JRip Algorithm**

Open the WEKA GUI Chooser from start menuall programs and click on the EXPLORER button.

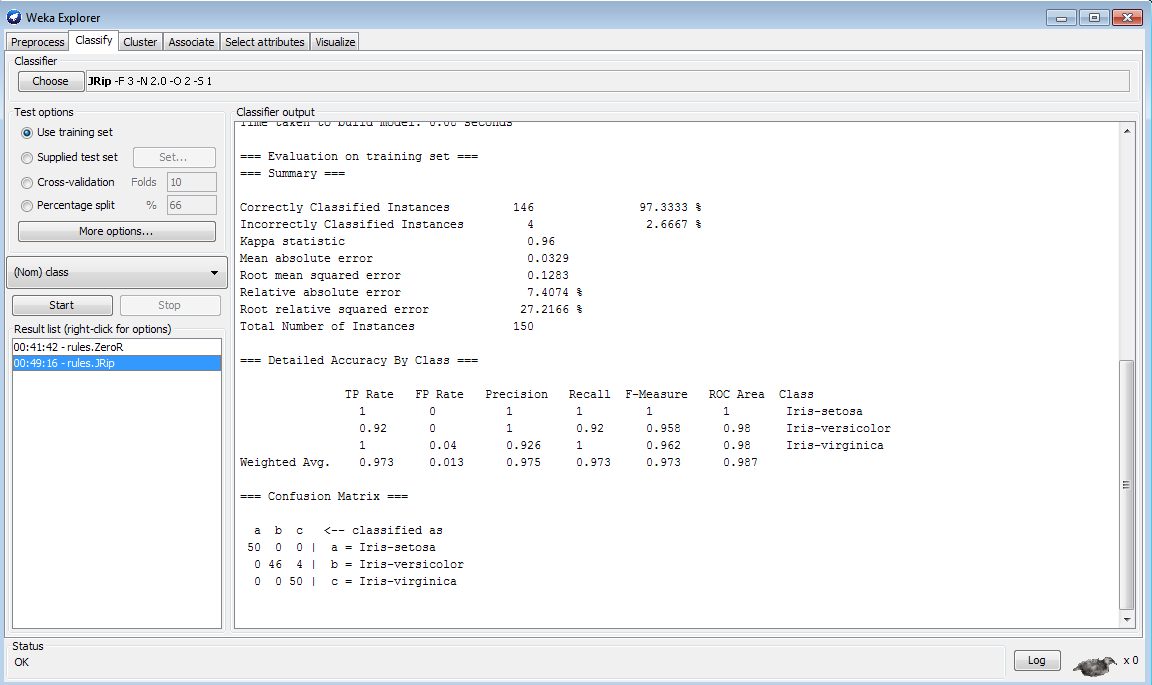
Now click on the **Open File** button and choose the file named as “iris.arff”.

****

After loading the input choose the classify tab in the WEKA explorer window. Under the classify tab click on choose button and select the JRip.

****

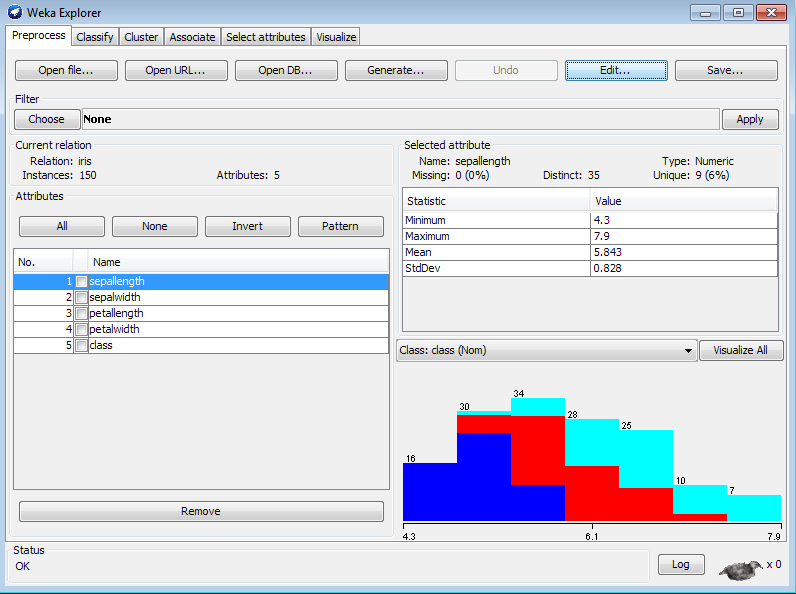
Now select the “**use training set** “under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.

****

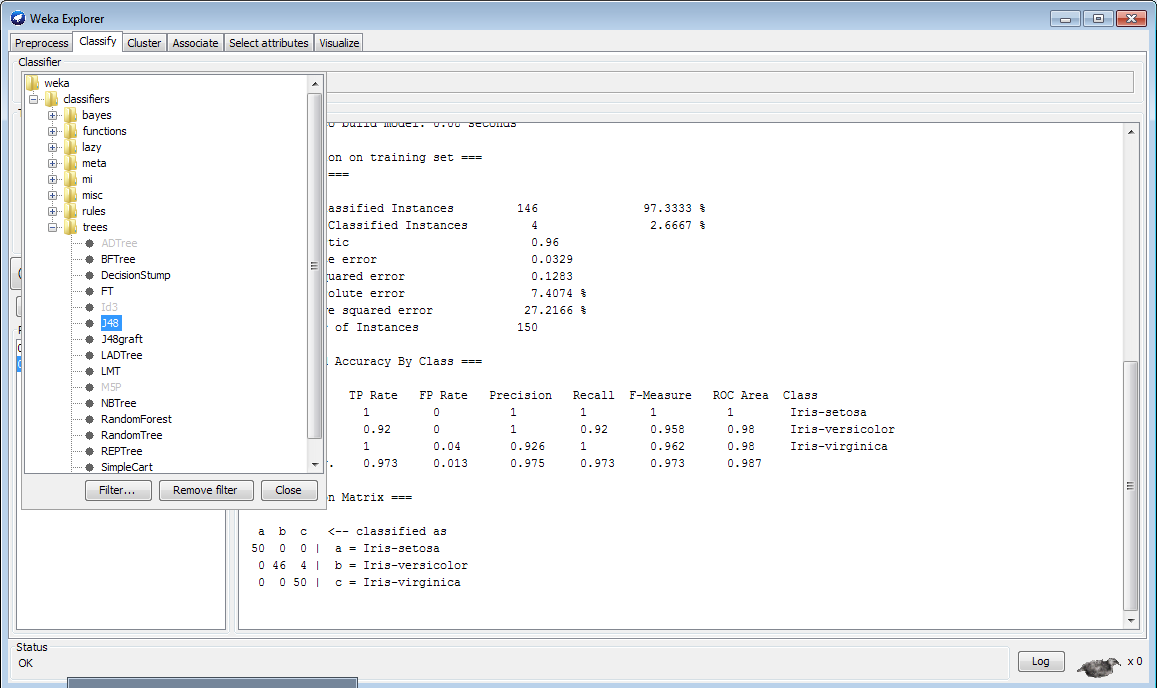
1. **J48 Algorithm**

Open the WEKA GUI Chooser from start menuall programs and click on the EXPLORER button.

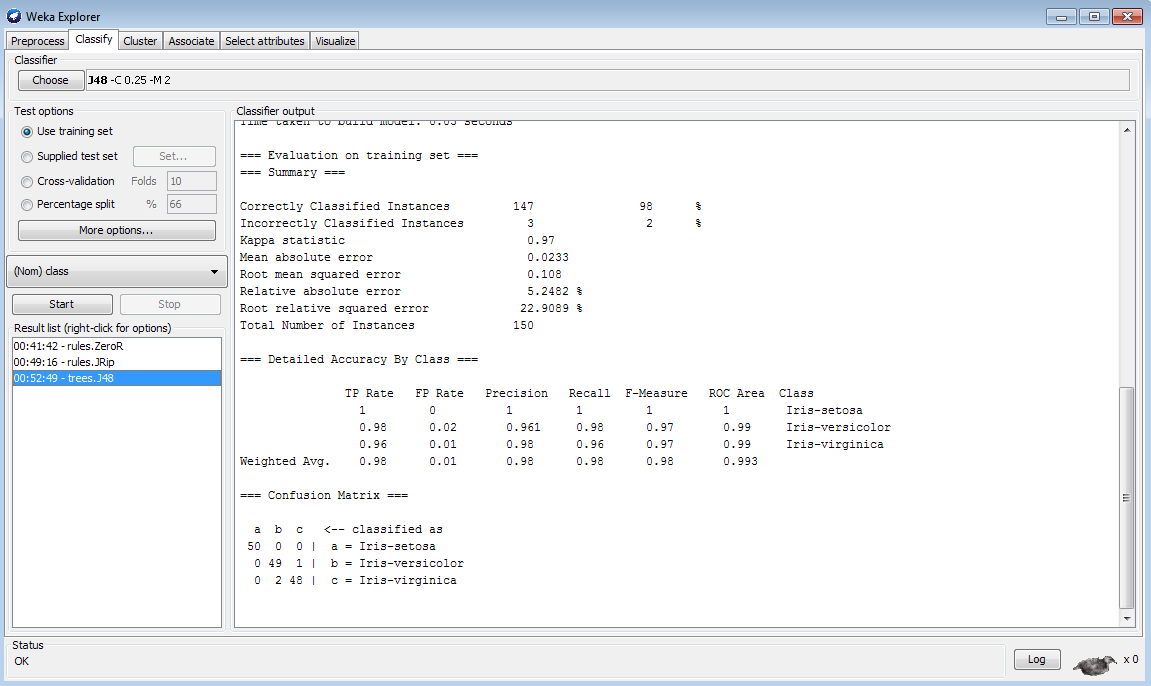
Now click on the **Open File** button and choose the file named as “iris.arff”.

****

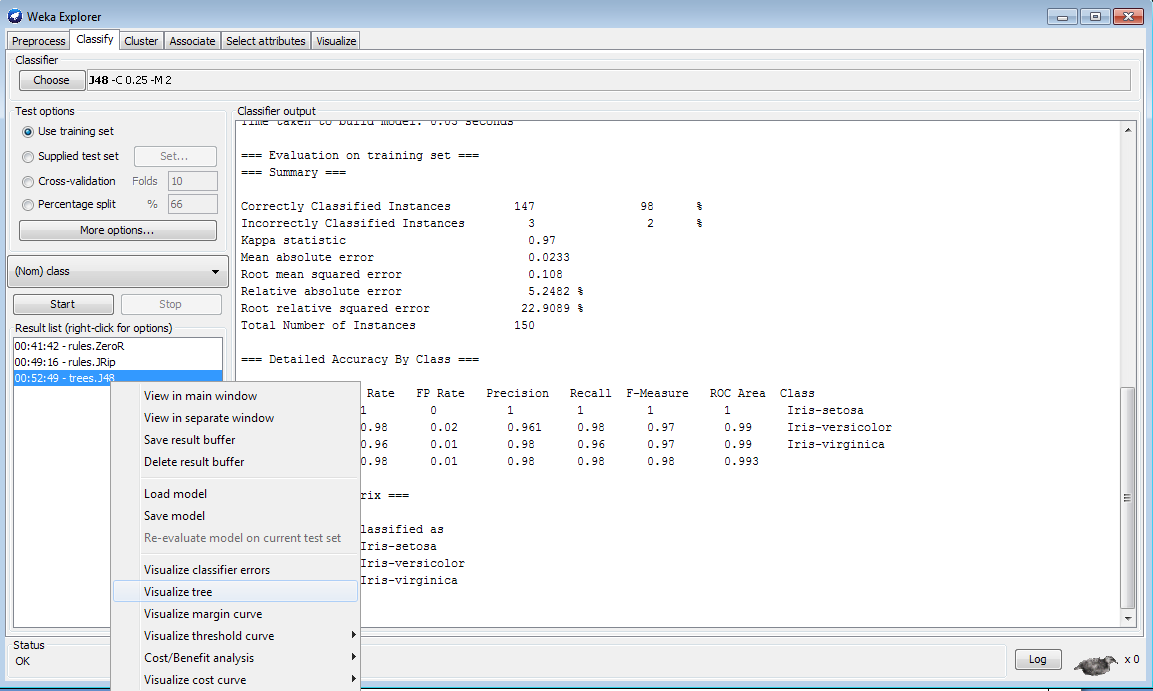
After loading the input choose the classify tab in the WEKA explorer window. Under the classify tab click on choose button and select the J48.

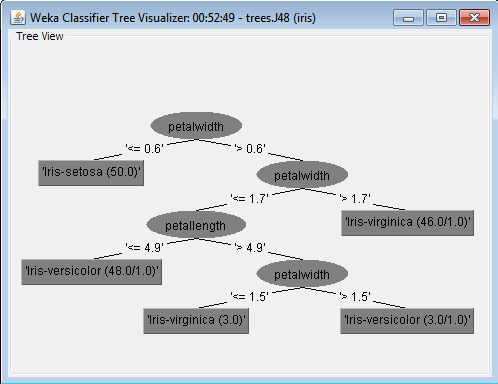


Now select the “**use training set** “under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.

****

Under the **Result List** right click the item to get the options and select the option “**visualize tree**” option.

****

****

1. **ASSOCIATION**

**ASSOCIATION RULE**: - It is a popular method for discovering interesting relations between variables in large databases.

It is used to identify the most frequent item sets or frequent patterns that occur frequently in a database.

To perform association rule mining we need two parameters like:

*Support*: - It is the total probability of the item sets that occurred in the transaction.

*Confidence*: - it is the ratio of probability of two item sets A,B and the probability of the item set A.

Association rule mining is used in different applications like basket analysis, catalog design, clustering, classification etc.

**Apriori Algorithm: -** To implement the association rule we use this algorithm.

Steps to follow while implementing the association rule are:

1. Find the frequent item sets where the items have the minimum support count.

2. Use the frequent item sets to generate association rules.

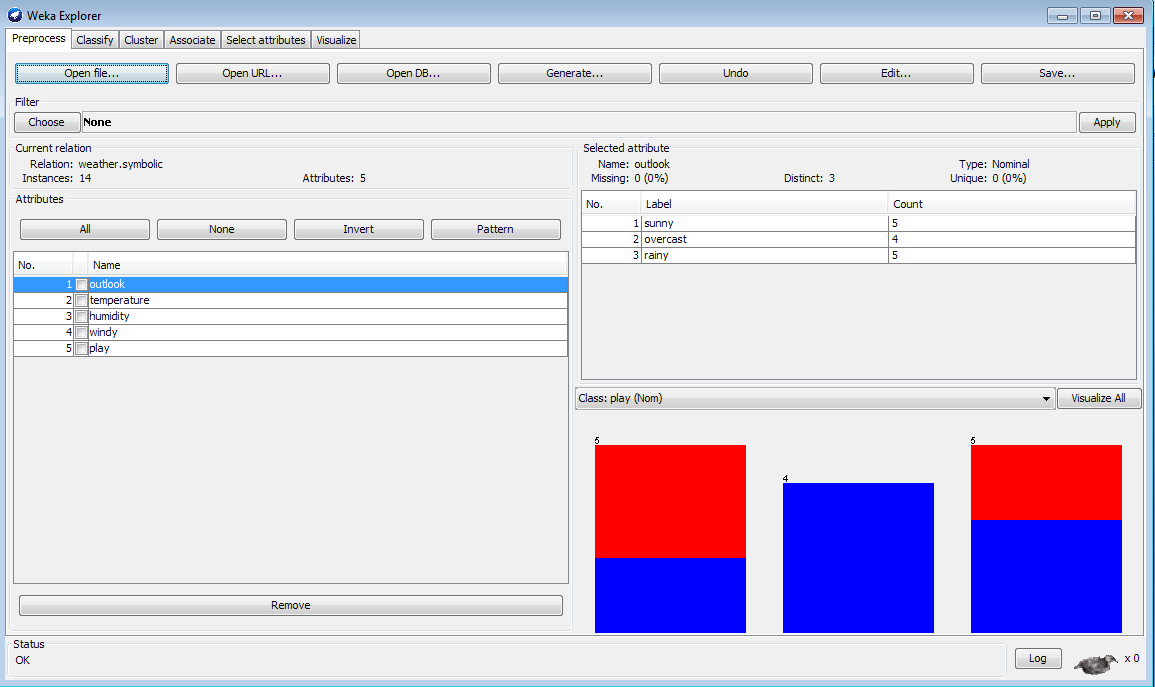
3. Generate the candidates by performing join operation.

4. Perform pruning operation i.e. if any one of the candidates that have a subset is not frequent in the set then delete it.

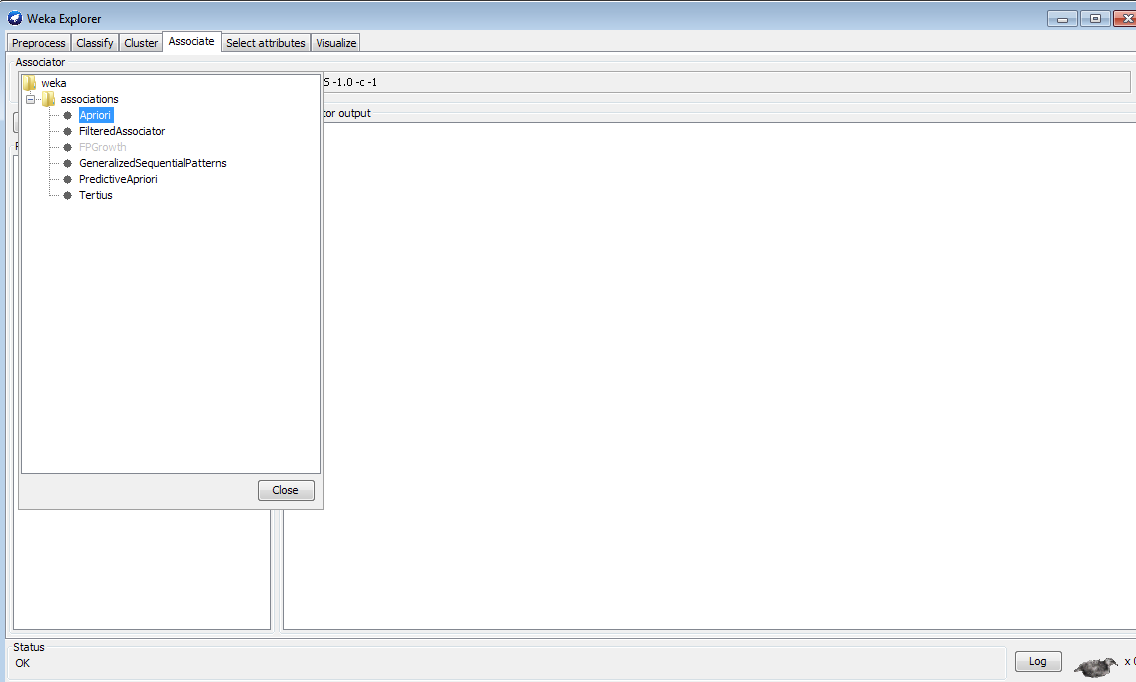
5. Also calculate support count for each candidate and delete those candidates whose minimum support count is not satisfied.

Open the WEKA GUI Chooser from start menuall programs and click on the EXPLORER button.

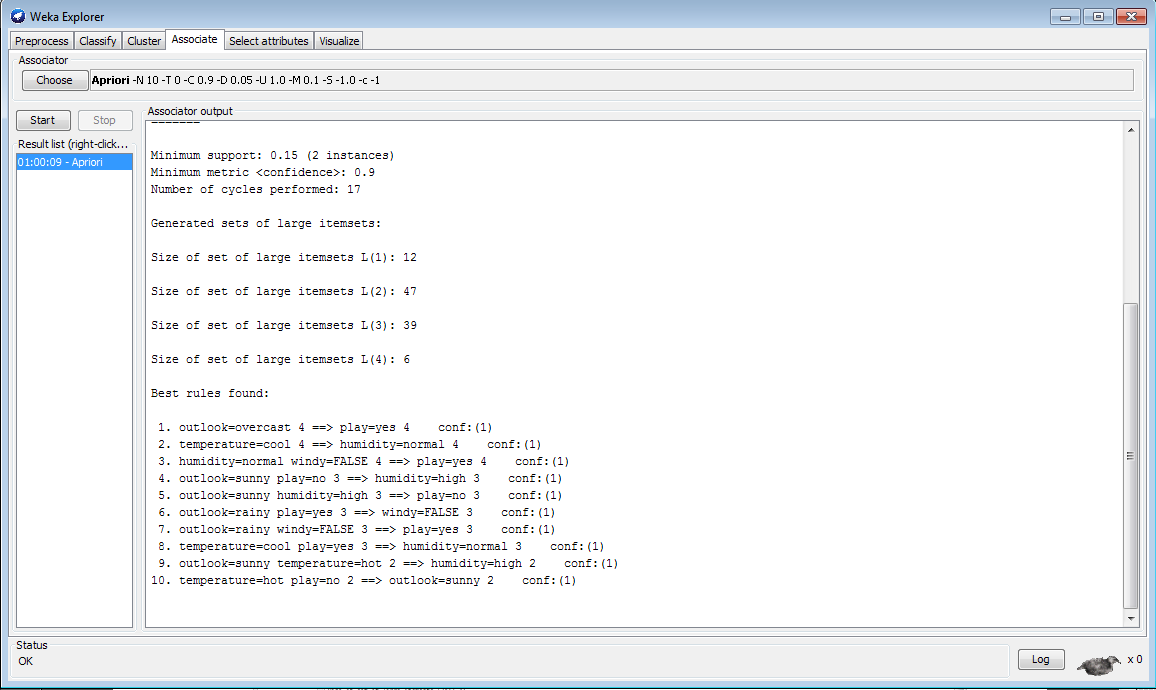
Now click on the **Open File** button and choose the file named as “weather.nominal.arff”.



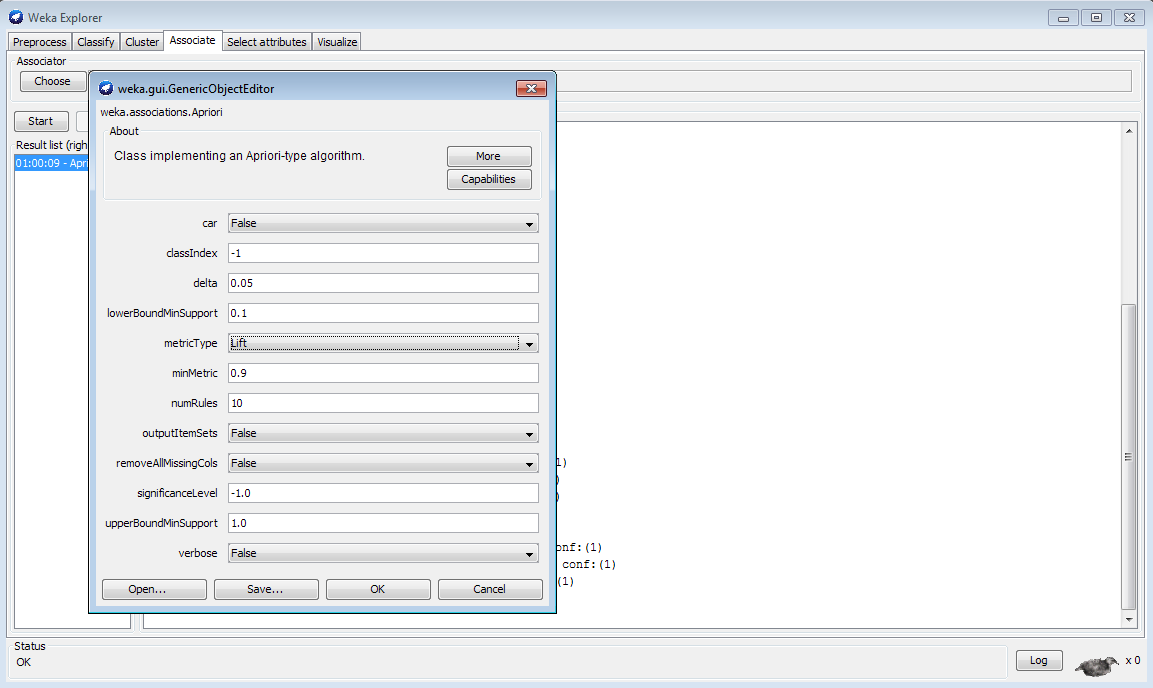
After loading the input file named association.csv as shown in figure 2, choose the associate tab in the WEKA explorer window. Under the associate tab click on choose button and select the **Apriori** Algorithm.



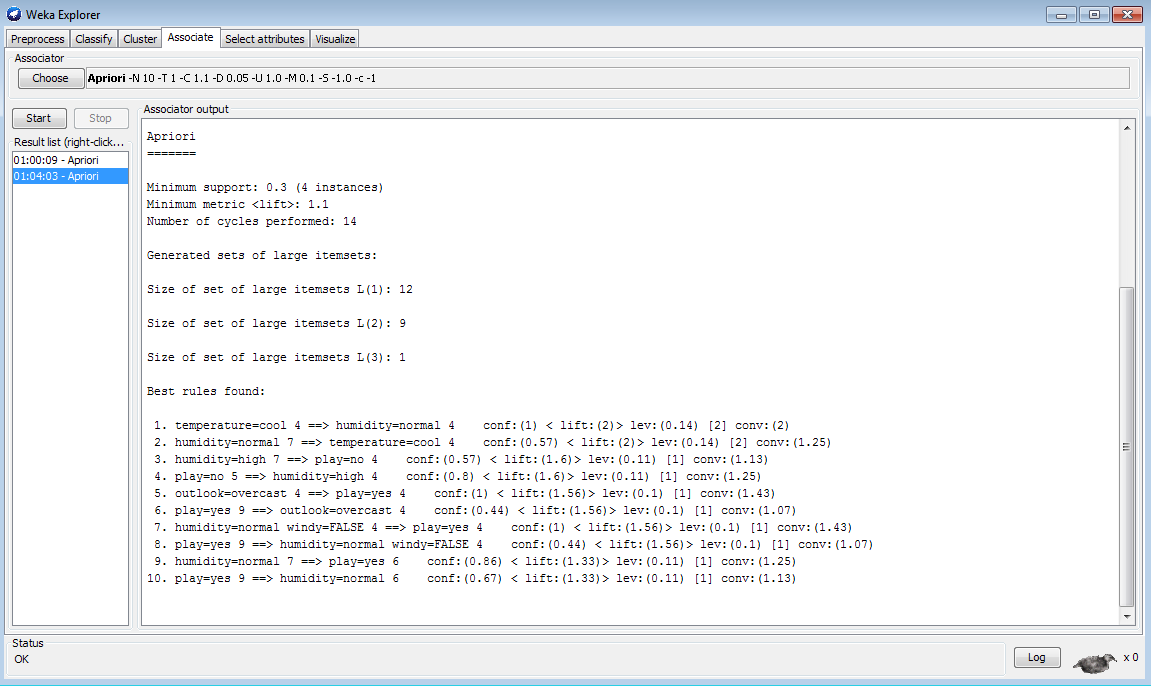
Now click on **start** button.



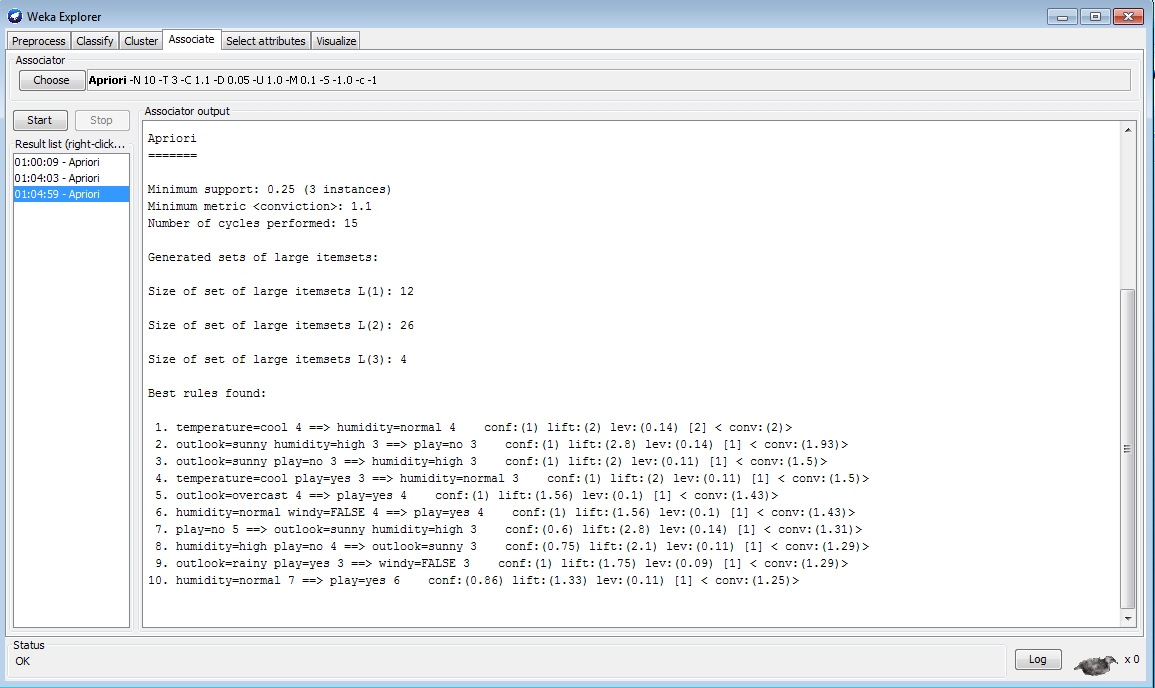
Now click on Apriori algorithm near choose button and change the metric from confidence to lift.

****

Click on start.

****

Change the metric to conviction and repeat.



1. **CLUSTERING**

Clustering is a task of assigning a set of objects into groups called as clusters. Clustering is also referred as cluster analysis where the objects in the same cluster are more similar to each other than to those objects in other clusters.

Clustering is the main task of Explorative Data mining and is a common technique for statistical data analysis used in many fields like machine learning, pattern recognition, image analysis, bio informatics etc…

Cluster analysis is not an algorithm but is a general task to be solved.

Clustering is of different types like hierarchical clustering which creates a hierarchy of clusters, partial clustering, and spectral clustering.

**SimpleK-Means: -**

It is a method of cluster analysis called as partial cluster analysis or partial clustering.

K-Means clustering partition or divides **n** observations into K clusters.

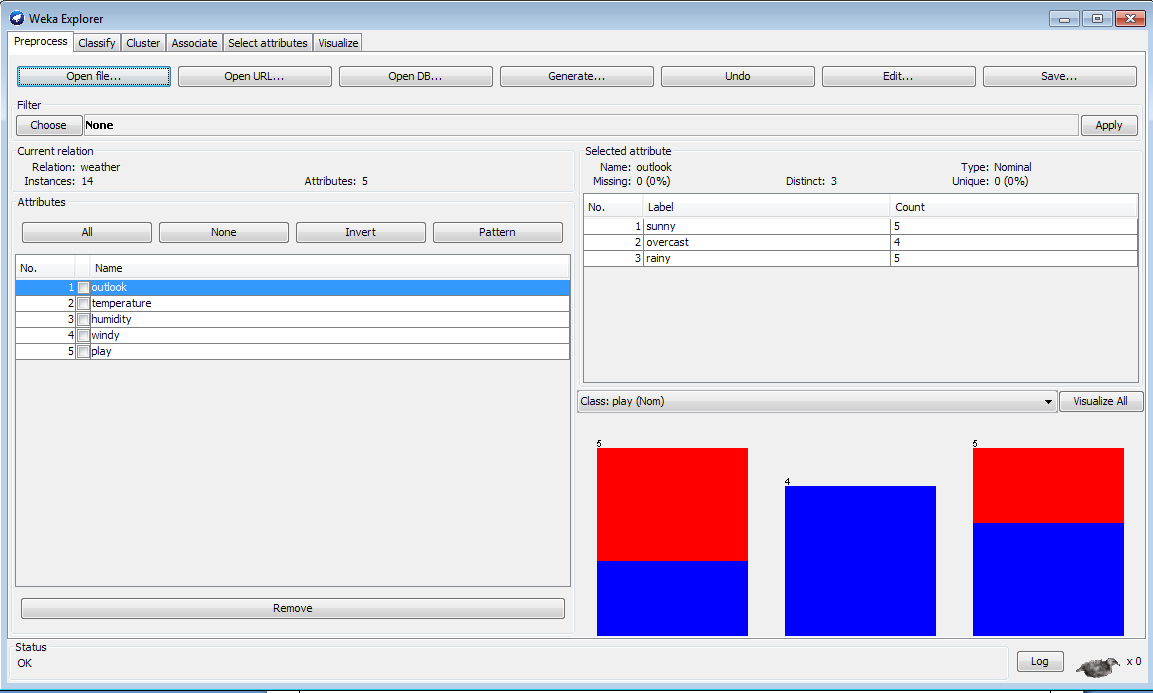
Each observation belongs to the cluster with the nearest mean.

K-means clustering is an algorithm to group the objects based on attributes/features into K number of groups where K is positive integer.

K-Means clustering is used in different types of applications like pattern recognition, artificial intelligent, image processing, etc.

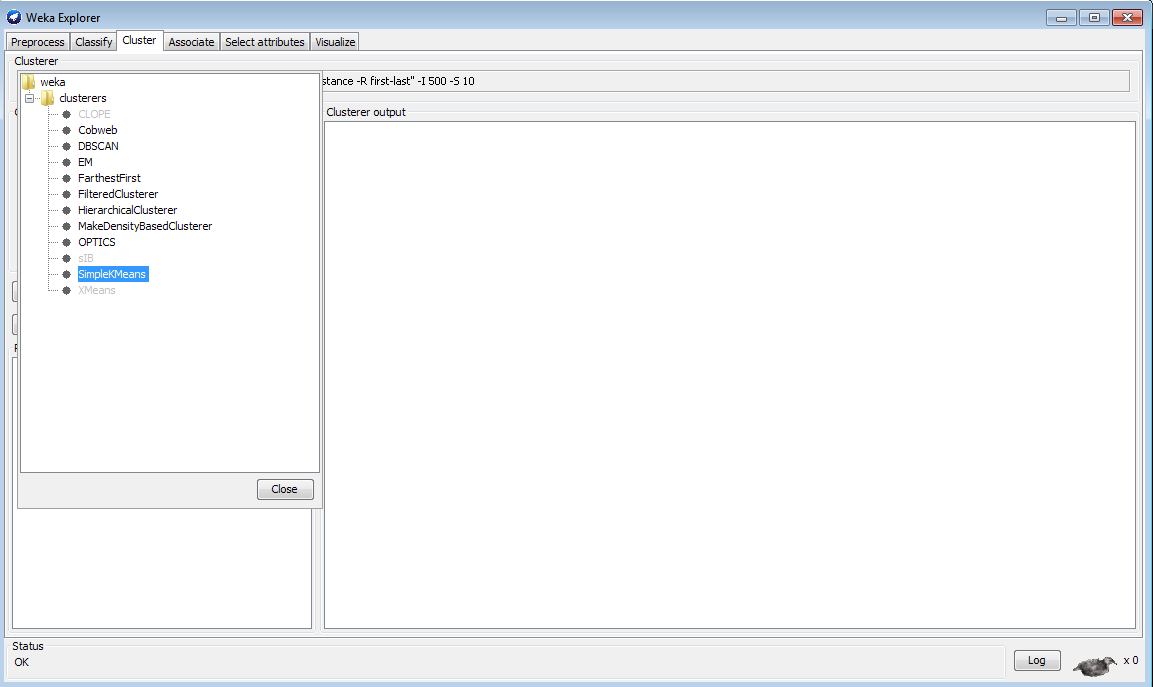
Open the WEKA GUI Chooser from start menuall programs and click on the EXPLORER button.

Now click on the **Open File** button and choose the file named as “weather.numeric.arff”.

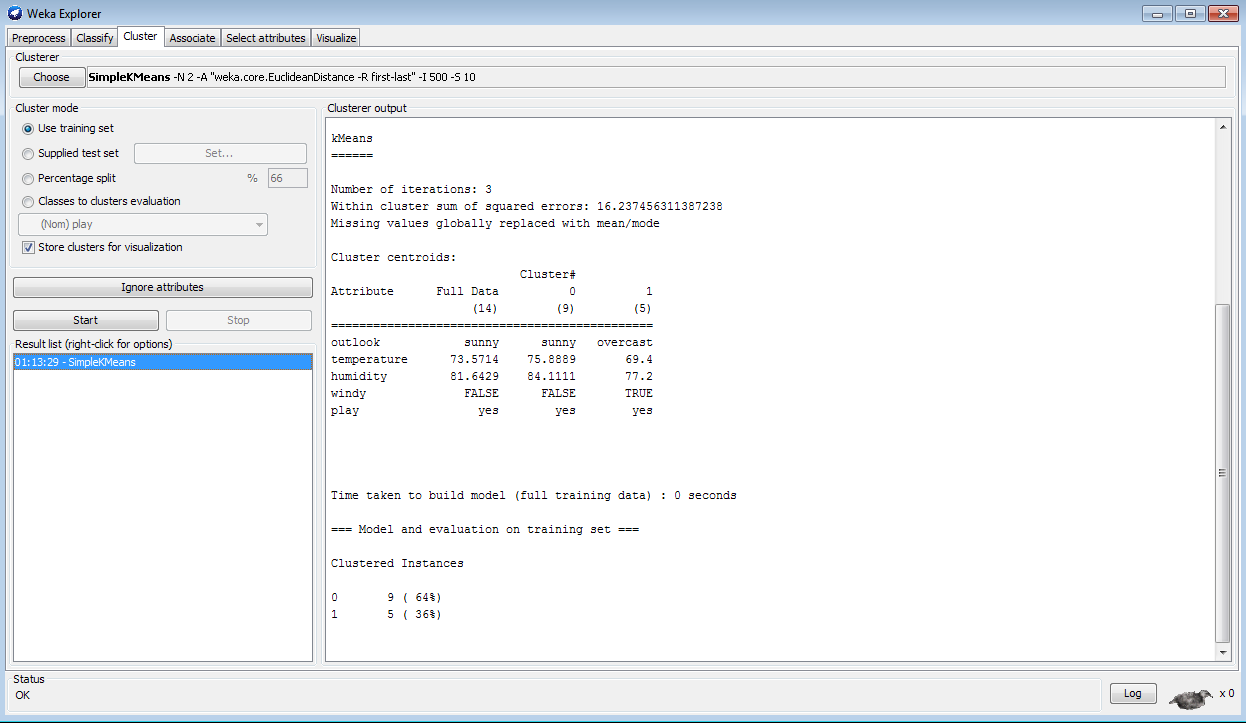


After loading the input file named cluster.csv as shown in figure 2, choose the cluster tab in the WEKA explorer window.

Under the cluster tab click on choose button and select the **SimpleKMeans** under **clusterers.**



Now select the “**use training set** “under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.



Under the **Result List** right click the item to get the options and select the option “**visualize cluster assignments**” option.

