Experiment No :- 04

19_Sanket Chandrashekhar Harvande

PAGE No.
DATE / /

Experiment No. 04

Aim :- Implemented k-means clustering algorithm in Java and using WEKA tool.

Theory & Concept :-

K-means algorithm is very simplest unsupervised learning Algorithm that is used to solve clustering problem in data mining.

K-means is one of the simplest unsupervised learning algorithm that solve the well known clustering problem. so the better choice is to place them as much as possible far away from each other. After we have k-new centroids from each other. After we have some data points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k-centers change their location step by step until no more changes are done. in other words centers do not move any more.

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (||x_i - v_j||)^2$$

Where

$||(x_i - v_j)||$ is the euclidean distance between x_i & v_j

c_i is the number of data points in i^{th} cluster

'c' is the number of cluster centers.

PAGE NO.
DATE: / /

Algorithm steps for K-means clustering :-

Let $x = (x_1, x_2, x_3 \cdots x_n)$ be the set of data points & $v = \{v_1, v_2 \cdots v_c\}$ be the set of centers.

1) Randomly select 'c' cluster centers.
2) Calculates the distance between each data point
3) Assign the data point to the cluster center whose distance from the cluster minimum of all the cluster centers
4) Recalculate the new cluster center using

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_j$$

where 'c_i' represent number of data points in i^{th} cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop. otherwise repeat.

Conclusion :-

Thus we have implemented k-means clustering algorithm in java & weka tool.

Using Java :-

```
//Program
import java.util.*;
class k_means
{
static int count1,count2,count3;
static int d[];
static int k[][];
static int tempk[][];
static double m[];
static double diff[];
static int n,p;

static int cal_diff(int a) // This method will determine the cluster in which an element go at a particular step.
{
int  temp1=0;
for(int i=0;i<p;++i)
{
if(a>m[i])
diff[i]=a-m[i];
else
diff[i]=m[i]-a;
}
int val=0;
double temp=diff[0];
for(int i=0;i<p;++i)
{
if(diff[i]<temp)
{
temp=diff[i];
val=i;
}
}//end of for loop
return val;
}

static void cal_mean() // This method will determine intermediate mean values
{
for(int i=0;i<p;++i)
m[i]=0; // initializing means to 0
int cnt=0;
for(int i=0;i<p;++i)
{
cnt=0;
for(int j=0;j<n-1;++j)
{
if(k[i][j]!=-1)
```

```
{
m[i]+=k[i][j];
++cnt;
}}
m[i]=m[i]/cnt;
}
}
static int check1() // This checks if previous k ie. tempk and current k are same.Used as terminating case.
{
for(int i=0;i<p;++i)
for(int j=0;j<n;++j)
if(tempk[i][j]!=k[i][j])
{
return 0;
}
return 1;
}
public static void main(String args[])
{
Scanner scr=new Scanner(System.in);
/* Accepting number of elements */
System.out.println("Enter the number of elements ");
n=scr.nextInt();
d=new int[n];
/* Accepting elements */
System.out.println("Enter "+n+" elements: ");
for(int i=0;i<n;++i)
d[i]=scr.nextInt();
/* Accepting num of clusters */
System.out.println("Enter the number of clusters: ");
p=scr.nextInt();
/* Initialising arrays */
k=new int[p][n];
tempk=new int[p][n];
m=new double[p];
diff=new double[p];
/* Initializing m */
for(int i=0;i<p;++i)
m[i]=d[i];

int temp=0;
int flag=0;
do
{
for(int i=0;i<p;++i)
for(int j=0;j<n;++j)
{
k[i][j]=-1;
}
```

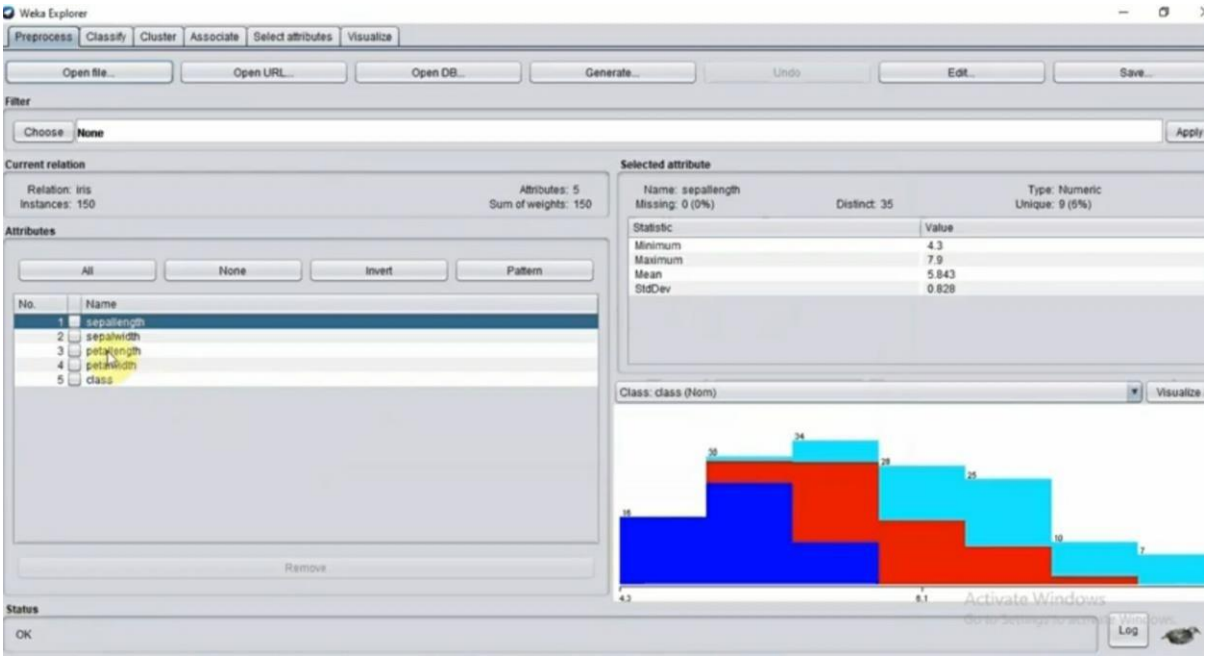
```
for(int i=0;i<n;++i) // for loop will cal cal_diff(int) for every element.
{
temp=cal_diff(d[i]);
if(temp==0)
k[temp][count1++]=d[i];
else
if(temp==1)
k[temp][count2++]=d[i];
else
if(temp==2)
k[temp][count3++]=d[i];
}
cal_mean(); // call to method which will calculate mean at this step.
flag=check1(); // check if terminating condition is satisfied.
if(flag!=1)
/*Take backup of k in tempk so that you can check for equivalence in next step*/
for(int i=0;i<p;++i)
for(int j=0;j<n;++j)
tempk[i][j]=k[i][j];

System.out.println("\n\nAt this step");
System.out.println("\nValue of clusters");
for(int i=0;i<p;++i)
{
System.out.print("K"+(i+1)+"{ ");
for(int j=0;k[i][j]!=-1 && j<n-1;++j)
System.out.print(k[i][j]+" ");
System.out.println("}");
}//end of for loop
System.out.println("\nValue of m ");
for(int i=0;i<p;++i)
System.out.print("m"+(i+1)+"="+m[i]+"  ");

count1=0;count2=0;count3=0;
}
while(flag==0);

System.out.println("\n\n\nThe Final Clusters By Kmeans are as follows: ");
for(int i=0;i<p;++i)
{
System.out.print("K"+(i+1)+"{ ");
for(int j=0;k[i][j]!=-1 && j<n-1;++j)
System.out.print(k[i][j]+" ");
System.out.println("}");
}
}
}
```

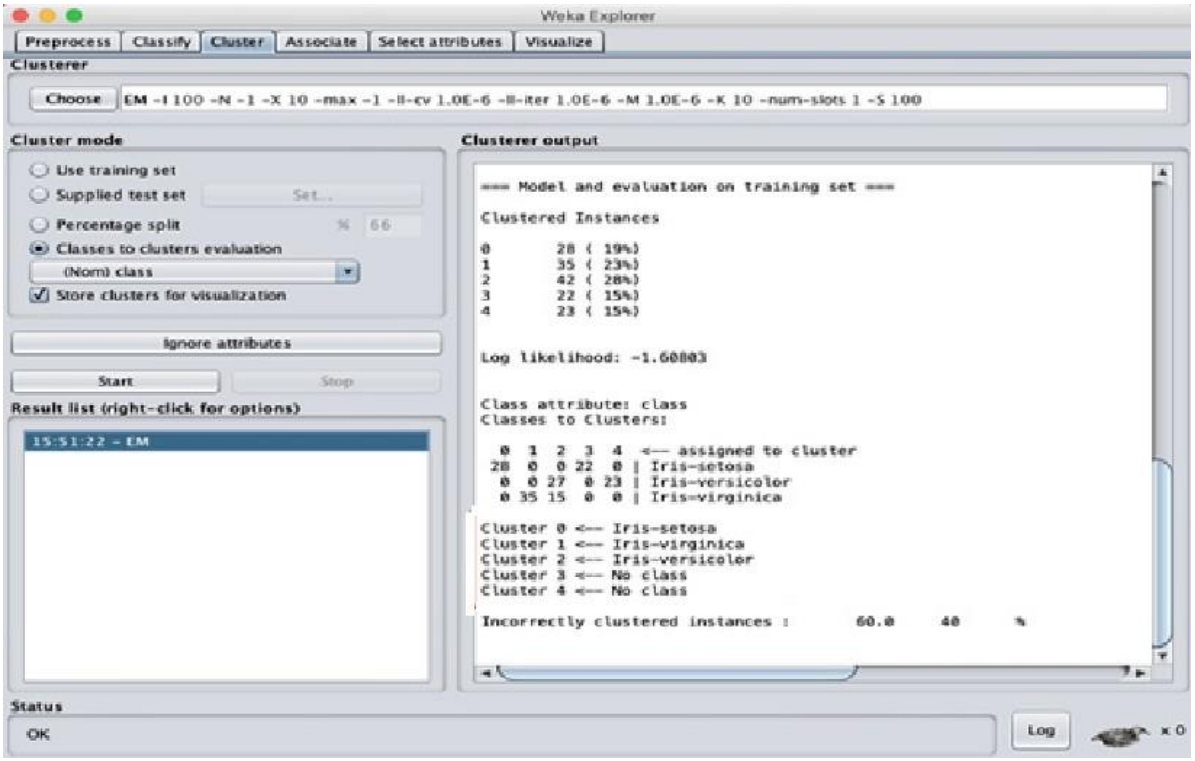
OUTPUT

```
Enter the number of elements
8
Enter 8 elements:
2 3 6 8 12 15 18 22
Enter the number of clusters:
3

At this step
Value of clusters
K1{ 2 }
K2{ 3 }
K3{ 6 8 12 15 18 22 }
Value of m
m1=2.0  m2=3.0  m3=13.5

At this step
Value of clusters
K1{ 2 }
K2{ 3 6 8 }
K3{ 12 15 18 22 }
Value of m
m1=2.0  m2=5.666666666666667  m3=16.75

At this step
Value of clusters
K1{ 2 3 }
```

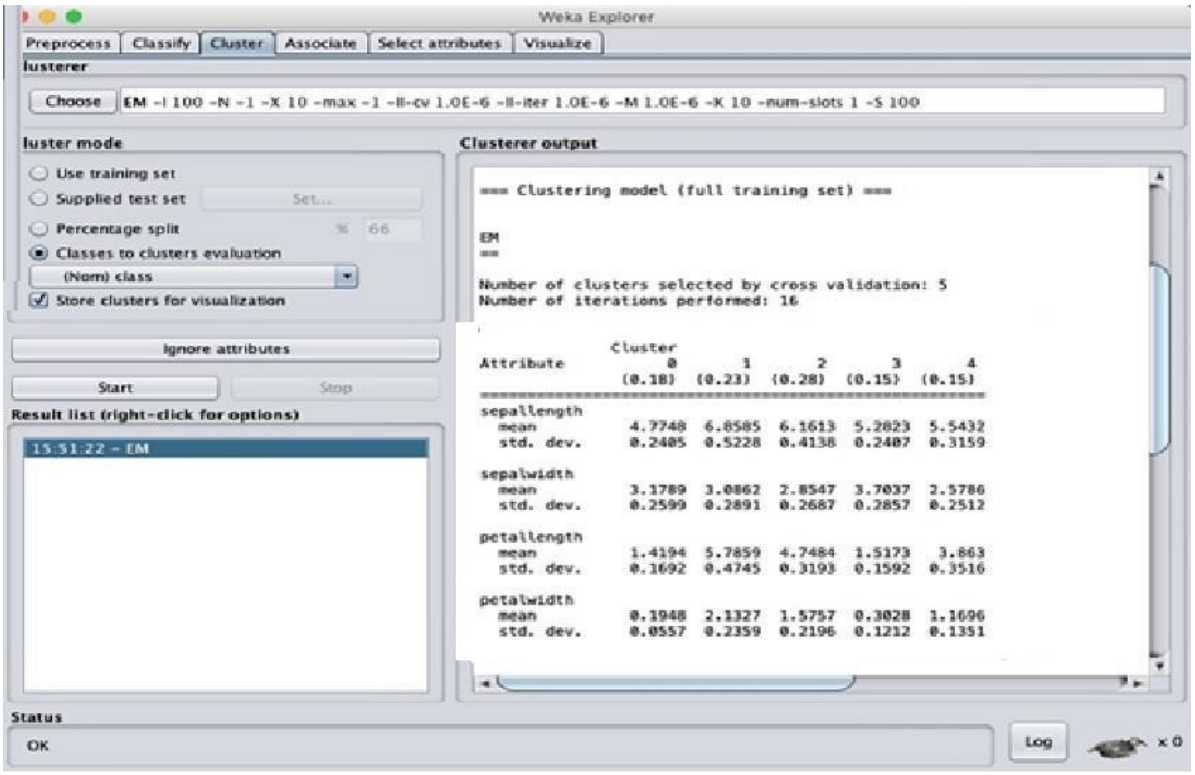
Using WEKA:-

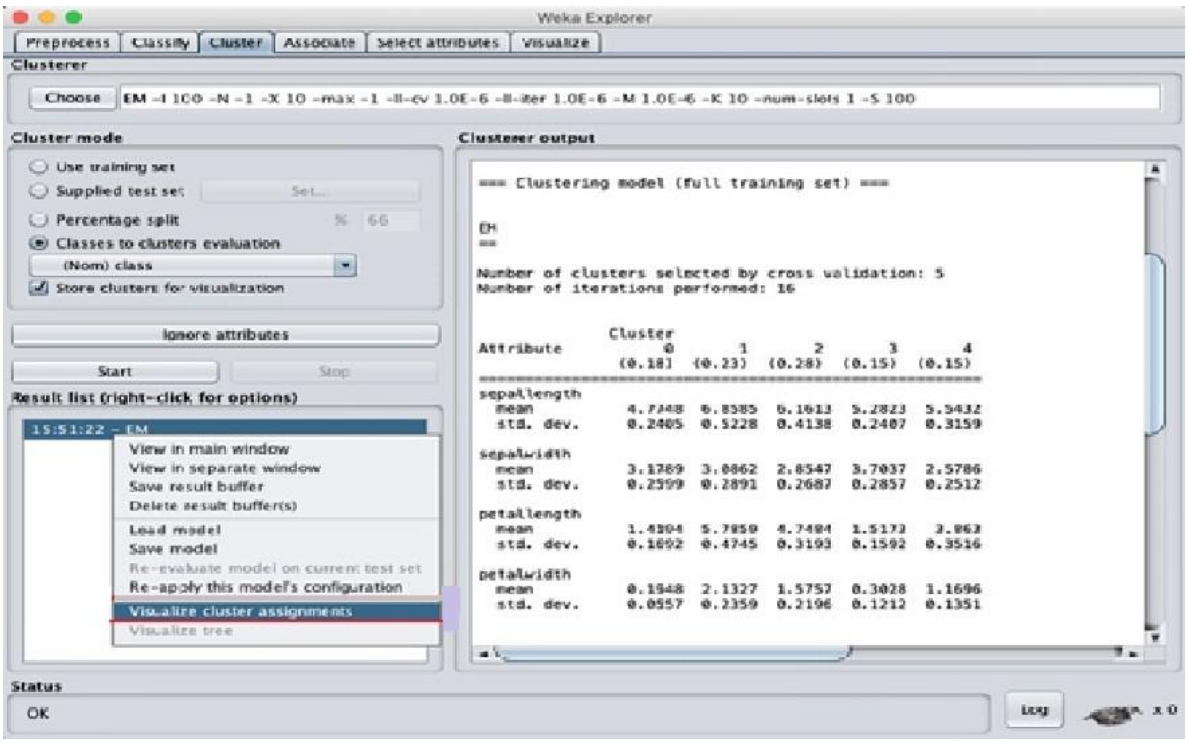
19_Sanket Chandrashekhar Harvande





19_Sanket Chandrashekhar Harvande





19_Sanket Chandrashekhar Harvande