

Experiment No :- 04

19_Sanket Chandrashekhhar Harvande

Experiment No. 04

PAGE NO.	
DATE	/ /

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 $X = (x_1, x_2, x_3, \dots, x_n)$ be the set of data points & $V = \{v_1, v_2, \dots, v_c\}$ be the set of centers.

- 1) Randomly select 'c' cluster centers
- 2) Calculate 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 ith 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 call 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

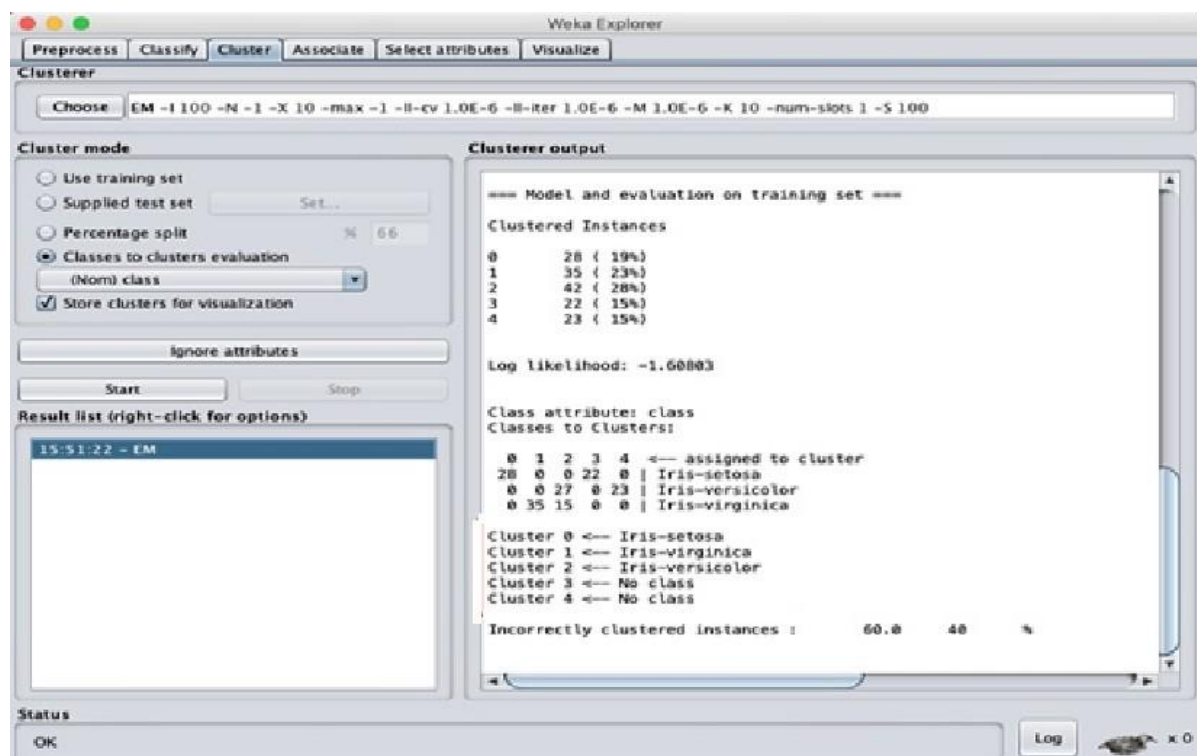
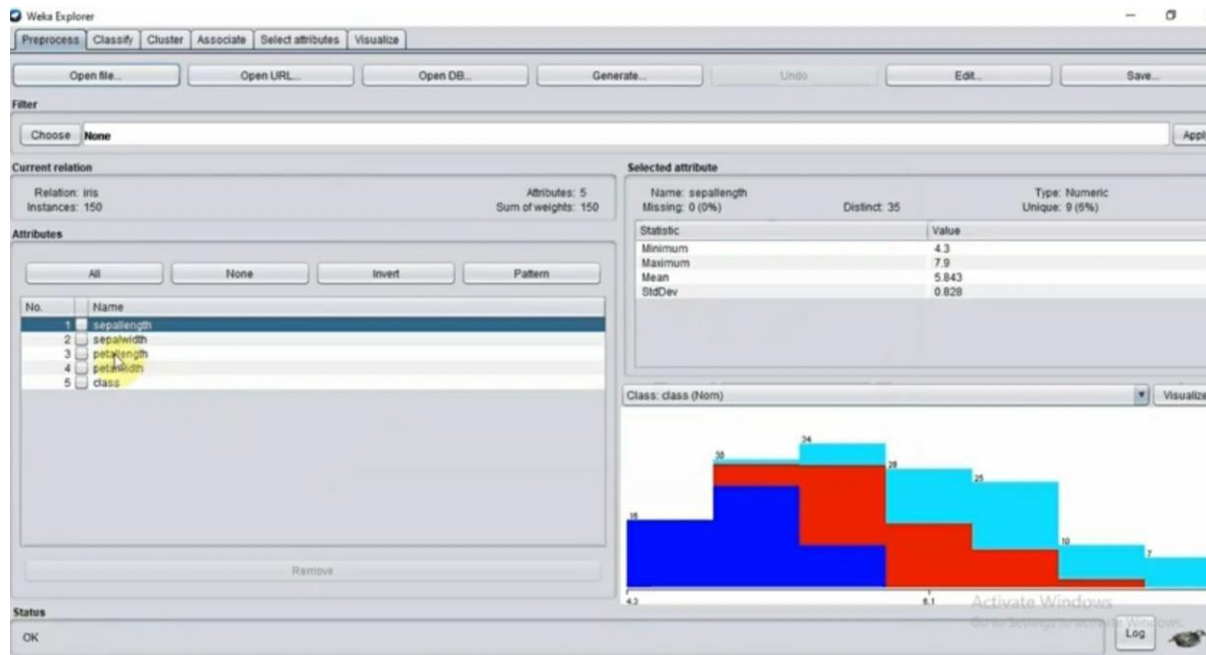
m1=2.0 m2=5.666666666666667 m3=16.75

At this step

Value of clusters

K1{ 2 3 }

Using WEKA:-



Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose EM -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode

☐ Use training set
☐ Supplied test set Set...
☐ Percentage split % 66
☒ Classes to clusters evaluation
 (Nom) class
☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

15:51:22 - EM

Status

OK Log x 0

Cluster output

=== Clustering model (full training set) ===

EM

Number of clusters selected by cross validation: 5
Number of iterations performed: 16

Attribute	Cluster 0 (0.18)	Cluster 1 (0.23)	Cluster 2 (0.28)	Cluster 3 (0.15)	Cluster 4 (0.15)
sepal.length					
mean	4.7748	6.8585	6.1613	5.2823	5.5432
std. dev.	0.2485	0.5228	0.4138	0.2487	0.3159
sepal.width					
mean	3.1789	3.0862	2.8547	3.7037	2.5786
std. dev.	0.2599	0.2891	0.2687	0.2857	0.2512
petal.length					
mean	1.4194	5.7859	4.7484	1.5173	3.863
std. dev.	0.1692	0.4745	0.3193	0.1592	0.3516
petal.width					
mean	0.1948	2.1327	1.5757	0.3028	1.1696
std. dev.	0.0557	0.2359	0.2196	0.1212	0.1351

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose EM -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode

☐ Use training set
☐ Supplied test set Set...
☐ Percentage split % 66
☒ Classes to clusters evaluation
 (Nom) class
☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

15:51:22 - EM

- View in main window
- View in separate window
- Save result buffer
- Delete result buffer(s)
- Load model
- Save model
- Re-evaluate model on current test set
- Re-apply this model's configuration
- Visualize cluster assignments**
- Visualize tree

Status

OK Log x 0

Cluster output

=== Clustering model (full training set) ===

EM

Number of clusters selected by cross validation: 5
Number of iterations performed: 16

Attribute	Cluster 0 (0.18)	Cluster 1 (0.23)	Cluster 2 (0.28)	Cluster 3 (0.15)	Cluster 4 (0.15)
sepal.length					
mean	4.7748	6.8585	6.1613	5.2823	5.5432
std. dev.	0.2485	0.5228	0.4138	0.2487	0.3159
sepal.width					
mean	3.1789	3.0862	2.8547	3.7037	2.5786
std. dev.	0.2599	0.2891	0.2687	0.2857	0.2512
petal.length					
mean	1.4194	5.7859	4.7484	1.5173	3.863
std. dev.	0.1692	0.4745	0.3193	0.1592	0.3516
petal.width					
mean	0.1948	2.1327	1.5757	0.3028	1.1696
std. dev.	0.0557	0.2359	0.2196	0.1212	0.1351

