Practical No. 6

Roll No.: K041	Name: Anish Sudhan Nair
Class: B.Tech CSE Cybsersecurity	Batch: K2
Date of Practical: 26/02/2022	Date of Submission: 05/03/2022
Grade:	

Aim: To implement K Means Clustering Algorithm and use Visualization tab in WEKA to visualize the clusters created

Prerequisite:

- Working of K Means Clustering algorithm
- Understanding of fundamental programming constructs in C/C++/Java
- Basic features of WEKA tool

Outcome: After successful completion of this experiment students will be able to

- Understand the working of K Means Clustering algorithm through implementation and the various measures used for checking its performance
- Use Cluster tab in WEKA , apply K Means clustering algorithm and analyze the model created.
- Use the Visualization tab in WEKA and visualize the clusters created

Theory:

k-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set 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 or in other words centers do

not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

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

where,

'$||x_i - v_j||$' is the Euclidean distance between x_i and v_j.

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

'c' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let X = {x_1,x_2,x_3,……..,x_n} be the set of data points and V = {v_1,v_2,…….,v_c} be the set of centers.

1) Randomly select 'c' cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.

4) Recalculate the new cluster center using:

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

where, 'c_i' represents the 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 from step 3).

Advantages

1) Fast, robust and easier to understand.

2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

3) Gives best result when data set are distinct or well separated from each other.

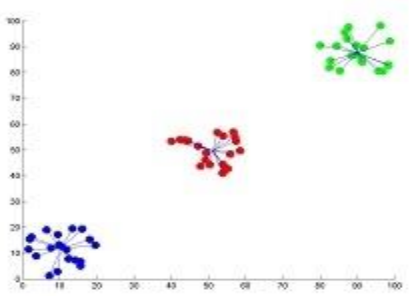


Fig I: Showing the result of k-means for *'N'* = 60 and *'c'* = 3

Disadvantages

1. The learning algorithm requires apriori specification of the number of cluster centers.
2. The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.
3. The learning algorithm is not invariant to non-linear transformations i.e., with different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).
4. Euclidean distance measures can unequally weight underlying factors.
5. The learning algorithm provides the local optima of the squared error function.
6. Randomly choosing of the cluster center cannot lead us to the fruitful result. Pl. refer Fig.
7. Applicable only when mean is defined i.e. fails for categorical data.
8. Unable to handle noisy data and outliers.
9. Algorithm fails for non-linear data set.

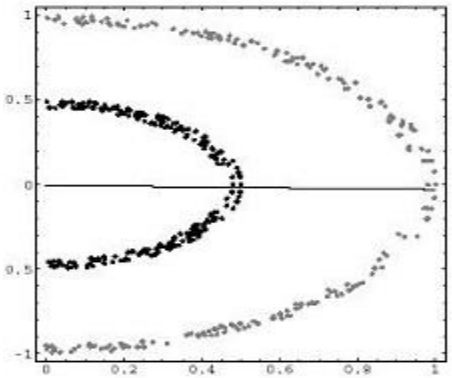


Fig II: Showing the non-linear data set where k-means algorithm fails

****

(TO BE COMPLETED BY STUDENTS)

Roll No.: K041	Name: Anish Sudhan Nair
Class: B.Tech CSE Cybsersecurity	Batch: K2
Date of Practical: 26/02/2022	Date of Submission: 05/03/2022
Grade:	

a. Implement a code to create clusters using K Means Clustering algorithm for the data set indicating the marks scored by 7 students in two subjects A and B. Create 2 clusters.

	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

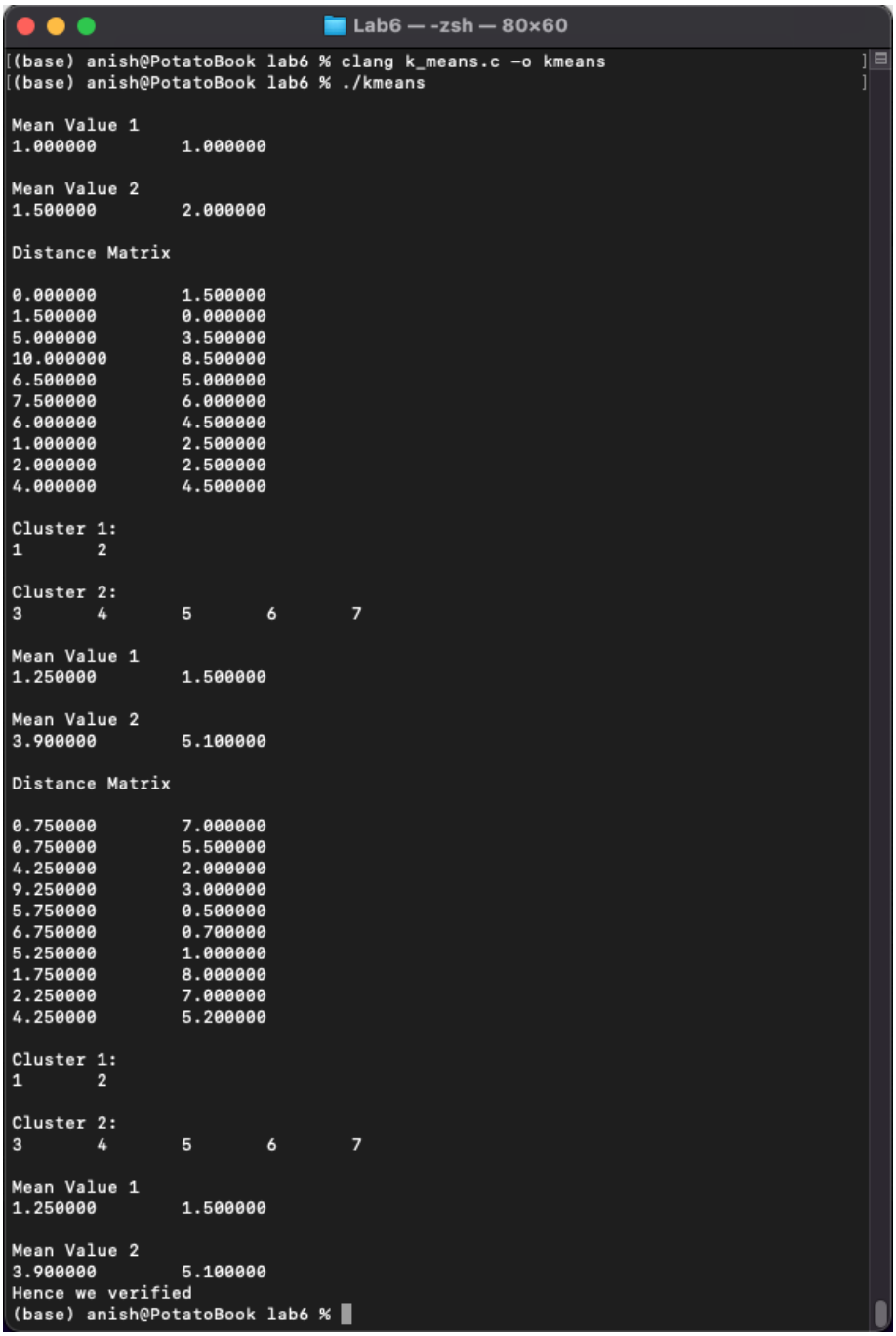
Code:

```c
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

float A[7]={1.0,1.5,3.0,5.0,3.5,4.5,3.5};
float B[7]={1.0,2.0,4.0,7.0,5.0,5.0,4.5};
int row_ID[7]={1,2,3,4,5,6,7};
float distances1[7],distances2[7];
float m1[2]={1.0,1.0};
float m2[2]={1.5,2.0};
float oldM1[2], oldM2[2];
int cluster1[7],cluster2[7];

void displayMean(float m[2], int j){
  printf("\nMean Value %d\n", j);
  for (int i = 0; i < 2; i++) {
    printf("%f\t",m[i] );
  }
  printf("\n");
}

void displayCluster(int cluster[]){

  for (int i = 0; i < 7; i++) {
```

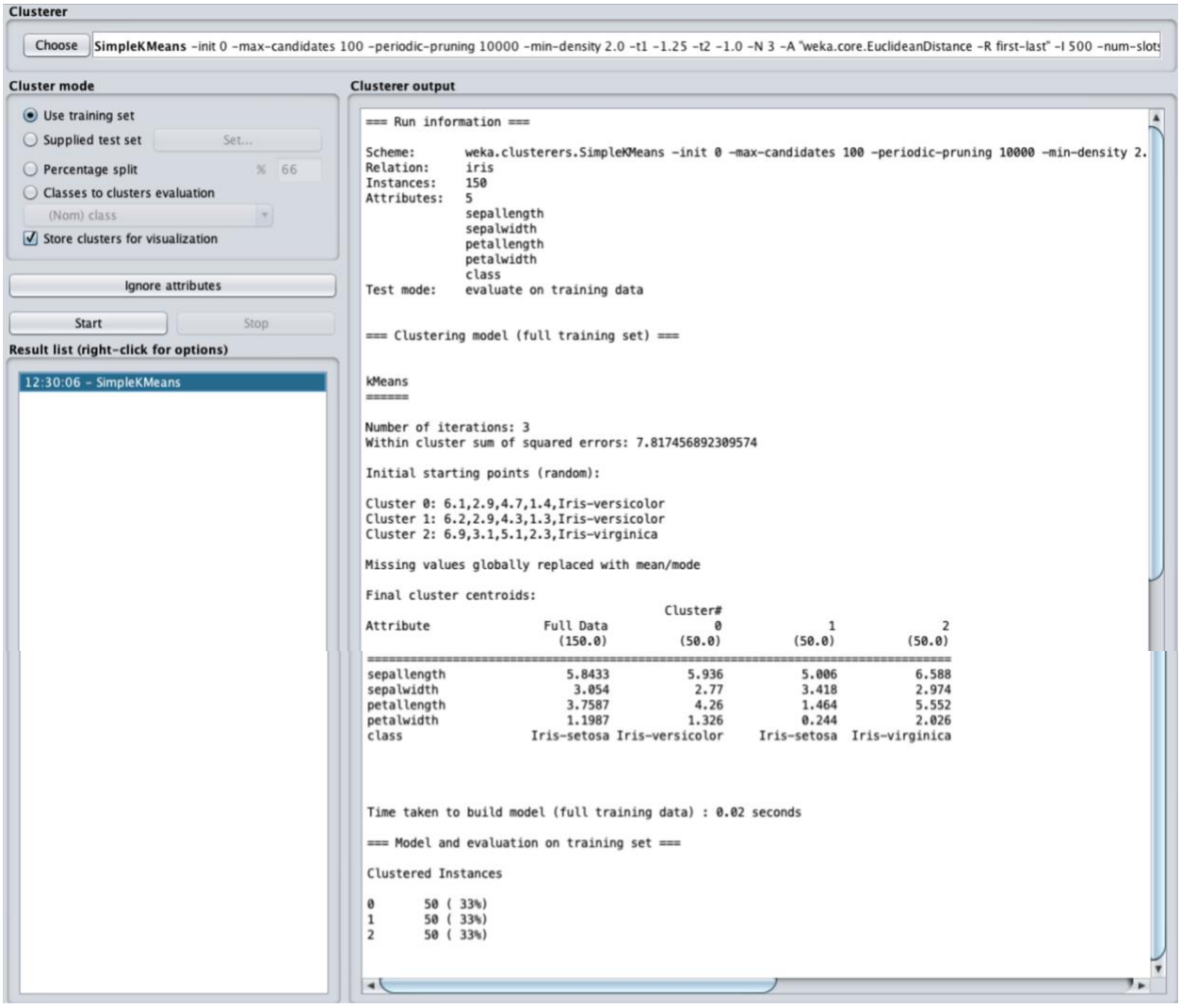
```
      if (cluster[i]==1) {
        printf("%d\t",row_ID[i] );
      }
    }
    printf("\n");
  }

  float distanceCalculator(float m[2], int i){
    float distance=0.0;
    distance=fabsf(m[0]-A[i])+fabsf(m[1]-B[i]);
    return distance;
  }

  void meanDistanceCalculator(float m1[2], float m2[2]){

    printf("\nDistance Matrix\n\n");

    for (int i = 0; i < 10; i++) {
      distances1[i]=distanceCalculator(m1,i);
      distances2[i]=distanceCalculator(m2,i);
      printf("%f\t%f\n",distances1[i],distances2[i]);
    }
  }

  void clusterValues(){

    for (int i = 0; i < 7; i++) {
      if (distances1[i]<distances2[i]) {
        cluster1[i]=1;
      }
      else {
        cluster2[i]=1;
      }
    }

    printf("\nCluster 1: \n");
    displayCluster(cluster1);
    printf("\nCluster 2: \n");
    displayCluster(cluster2);

  }

  void preserveMeanValues(float m[2], float oldM[2]) {
    for (int i = 0; i < 3; i++) {
      oldM1[i]=m1[i];
      oldM2[i]=m2[i];
    }
  }

  bool meanEqual(){

    int count=0;
```

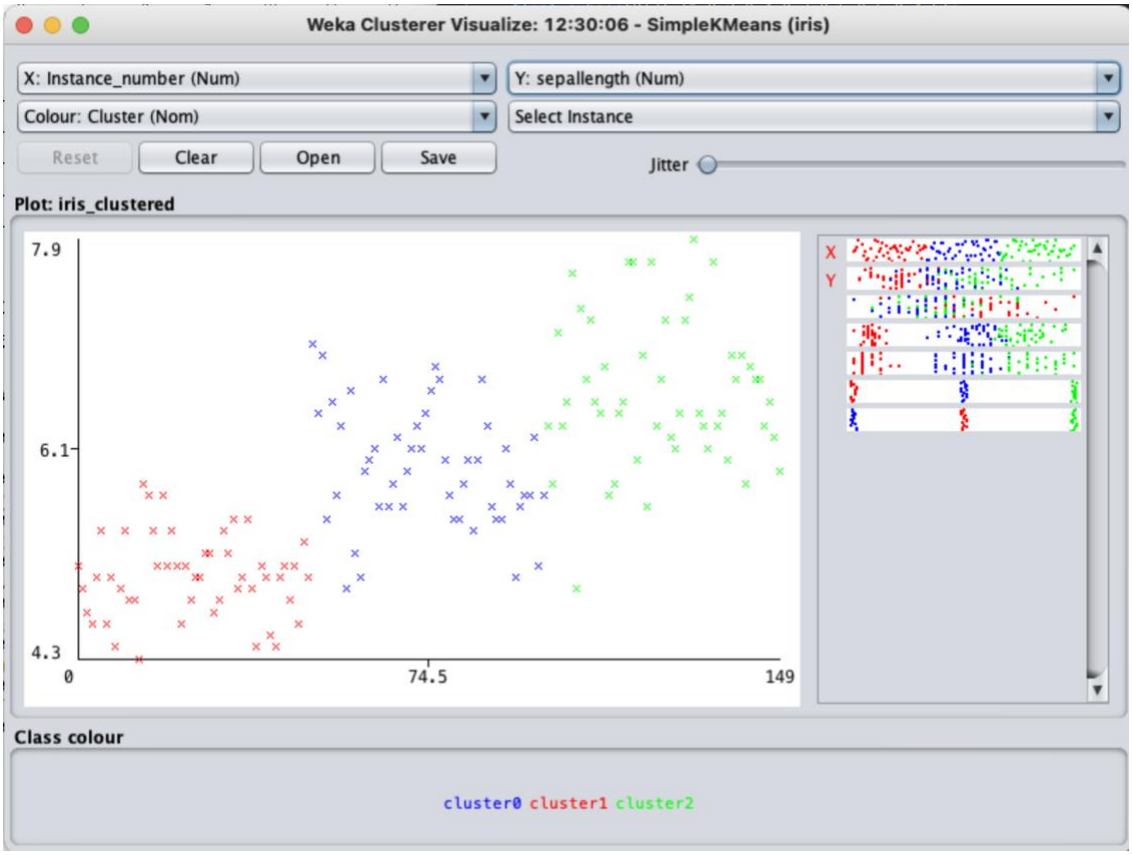
```
  for (int i = 0; i < 2; i++) {
    if ((m1[i]==oldM1[i])&&(m2[i]==oldM2[i])) {
      count++;
    }
    if (count==2) {
      return true;
    }
  }
  return false;
}

void newMeanValue(int cluster[7], float m[]){

  float sumA=0, sumB=0;
  int number=0;

  for (int i = 0; i < 7; i++) {
    if (cluster[i]==1) {
      sumA+=A[i];
      sumB+=B[i];
      number++;
    }
  }
  sumA/=number;
  sumB/=number;

  m[0]=sumA;
  m[1]=sumB;
}


int main(){

  do {
    displayMean(m1,1);
    displayMean(m2,2);
    meanDistanceCalculator(m1,m2);
    clusterValues();
    preserveMeanValues(m1, oldM1);
    preserveMeanValues(m2, oldM2);
    newMeanValue(cluster1, m1);
    newMeanValue(cluster2, m2);
  } while(!(meanEqual()));

  displayMean(m1,1);
  displayMean(m2,2);

  printf("Hence we verified \n");

  return 0;
}

```

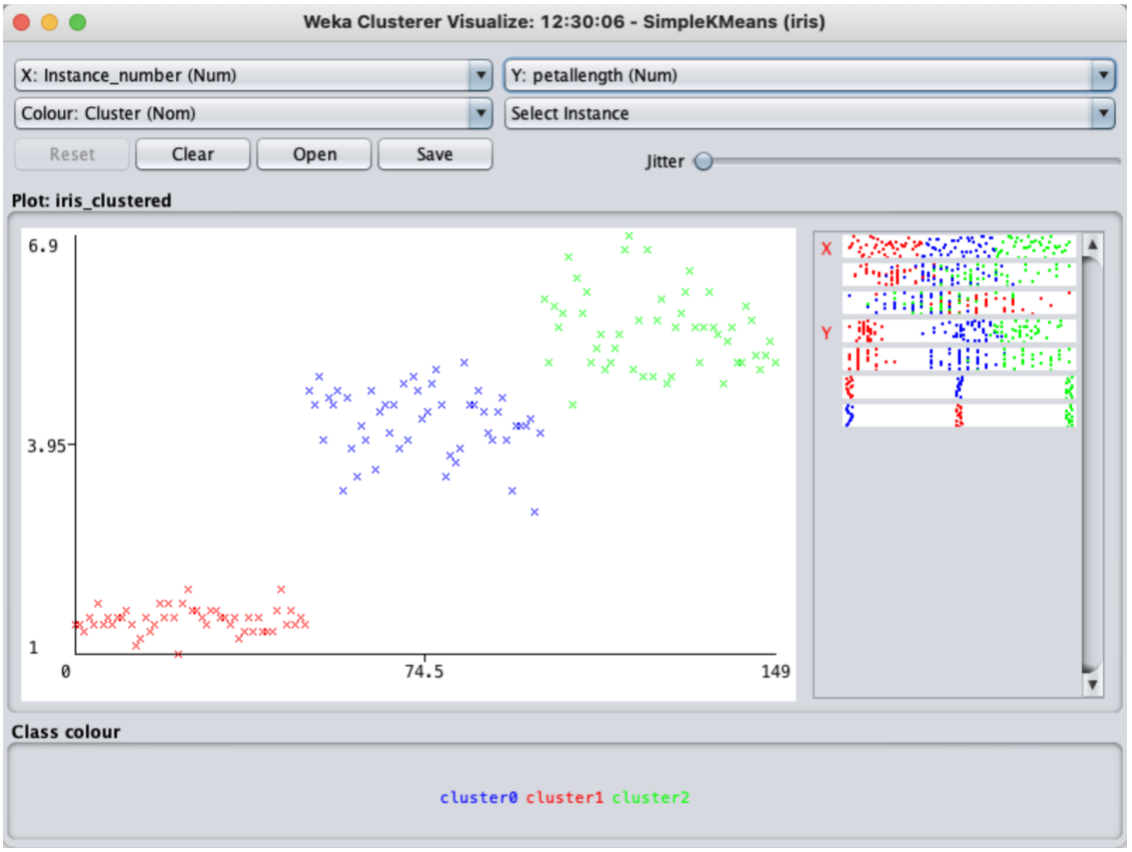
Output:

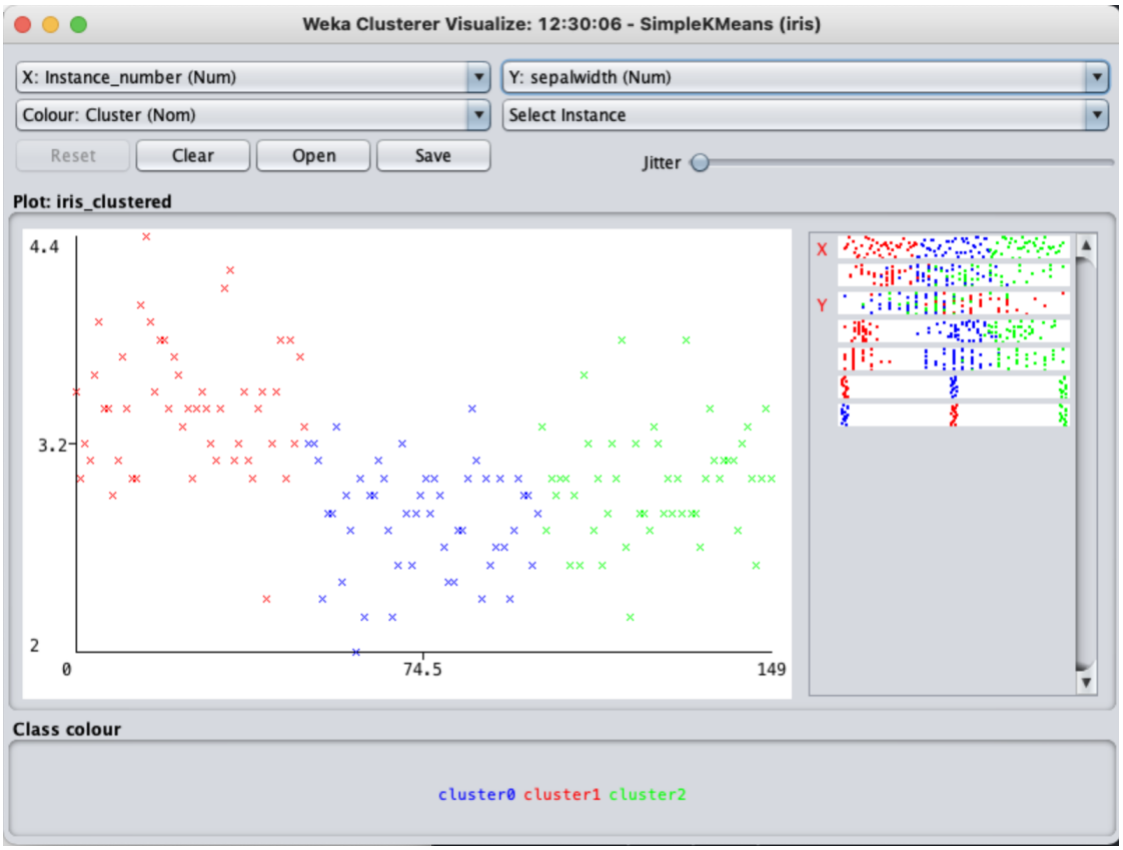

Lab6 — -zsh — 80×60
(base) anish@PotatoBook lab6 % clang k_means.c -o kmeans
(base) anish@PotatoBook lab6 % ./kmeans
Mean Value 1
1.000000 1.000000
Mean Value 2
1.500000 2.000000
Distance Matrix
0.000000 1.500000
1.500000 0.000000
5.000000 3.500000
10.000000 8.500000
6.500000 5.000000
7.500000 6.000000
6.000000 4.500000
1.000000 2.500000
2.000000 2.500000
4.000000 4.500000
Cluster 1:
1 2
Cluster 2:
3 4 5 6 7
Mean Value 1
1.250000 1.500000
Mean Value 2
3.900000 5.100000
Distance Matrix
0.750000 7.000000
0.750000 5.500000
4.250000 2.000000
9.250000 3.000000
5.750000 0.500000
6.750000 0.700000
5.250000 1.000000
1.750000 8.000000
2.250000 7.000000
4.250000 5.200000
Cluster 1:
1 2
Cluster 2:
3 4 5 6 7
Mean Value 1
1.250000 1.500000
Mean Value 2
3.900000 5.100000
Hence we verified
(base) anish@PotatoBook lab6 %

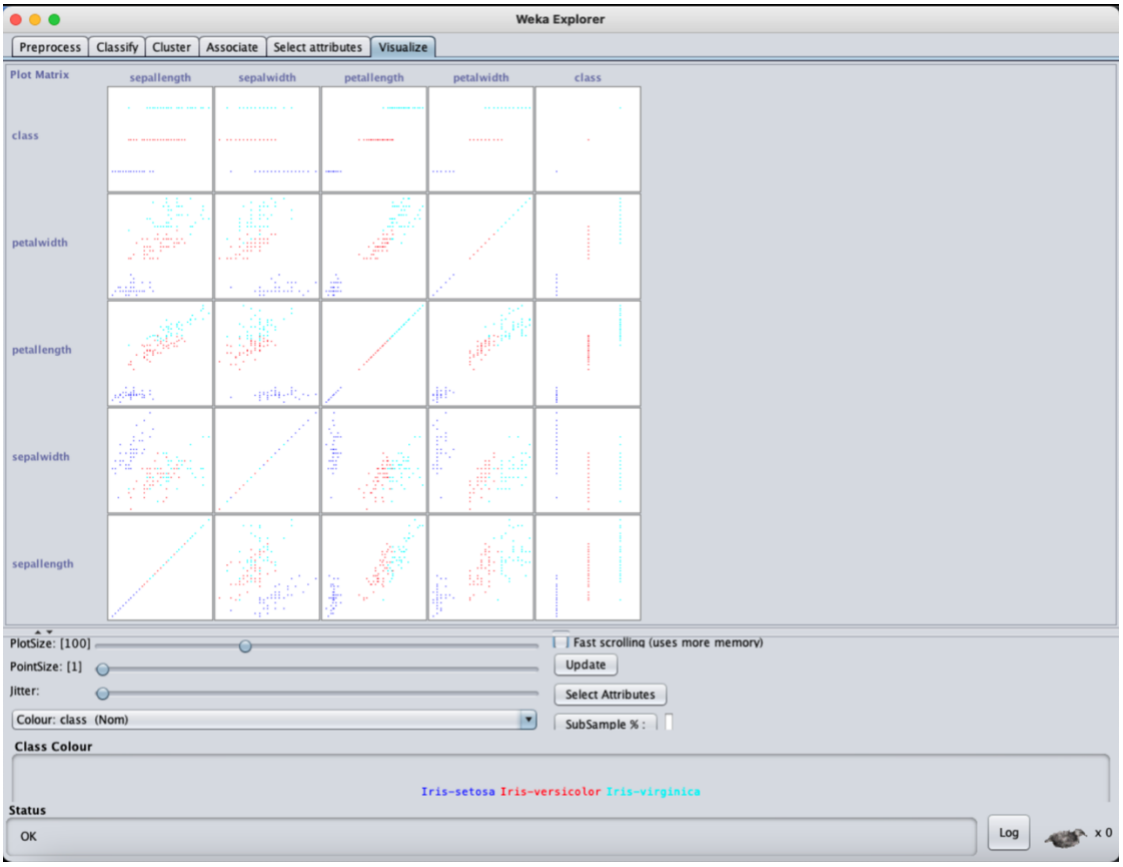
b. Using WEKA Tool – Create a K Means clustering model in WEKA for the Iris data set (iris.arff) and analyze the clusters with the Visualization tab.




Weka Clusterer Visualize: 12:30:06 - SimpleKMeans (iris)
X: Instance_number (Num)
Y: sepallength (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris_clustered
7.9
6.1
4.3
0
74.5
149
X
Y
Class colour
cluster0 cluster1 cluster2


Weka Clusterer Visualize: 12:30:06 - SimpleKMeans (iris)
X: Instance_number (Num)
Y: petallength (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris_clustered
6.9
3.95
1
0
74.5
149
X
Y
Class colour
cluster0 cluster1 cluster2


Weka Clusterer Visualize: 12:30:06 - SimpleKMeans (iris)
X: Instance_number (Num)
Y: sepalwidth (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris_clustered
4.4
3.2
2
0
74.5
149
X
Y
Class colour
cluster0 cluster1 cluster2


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Plot Matrix
sepallength
sepalwidth
petallength
petalwidth
class
PlotSize: [100]
PointSize: [1]
Jitter:
Colour: class (Nom)
Fast scrolling (uses more memory)
Update
Select Attributes
SubSample % :
Class Colour
Iris-setosa Iris-versicolor Iris-virginica
Status
OK
Log
x 0

Questions to be answered:

a. What is the benefit of using a K Medoid approach than a K Means method of cluster creation?

➔ The primary advantage K medoid clustering (such as Partitioning Around Medoids) has over K Means clustering is the superior ability to handle outliers. Medoids are essentially a more robust estimate of the representative point that the mean since it's an actual datapoint closer to the actual points over a virtual average point, and in k means, the mean will be centered with respect to the outliers (towards them) while in K medoids one of the more centered clustered members would be chosen. Hence, the pairwise distance measure is able to interpret outliers better.

b. Explain any one method of handling outliers in Clustering.

➔ The primary aspect of outlier handling is outlier identification, post which it can be removed, changed to another value, marked as irregular and removed form cluster etc.

➔ One of the methods of outlier handling (identification) is the Silhouette Analysis. This method essentially every data points similarity to its own cluster as opposed to the other clusters and assigns a value between 1 to -1.

➔ Negative value indicates wrong cluster or outlier, 0 indicates that the data pint is in the boundary decision of 2 neighbouring clusters (could belong to either) and a positive value indicates correct clustering (distance of sample from other clusters)

➔ Mathematically, the silhouette coefficient can be found by first finding the average distance of the chosen point with all the other points in the same cluster (an (i)), then average of the point with all the points in the closest neighbouring cluster (b(i)), and then inserting them into the formula below:

$$\mathbf{s(i)} = \frac{\mathbf{b(i)} - \mathbf{a(i)}}{\mathbf{max(b(i), a(i))}}$$

****