1. Perform k-means clustering with this dataset for values of k to be 3, 4, 5, 6, 7, and 8. For each case of k run the clustering algorithm with three different initial cluster centers and select the one with the lowest total SSE value of all clusters in the clustering. Report the following in the submitted work: (Use Matlab kmeans function or any other similar toolbox)

**#Importing necessary packages in sciItLearn**

from sklearn.cluster import KMeans

import numpy as np

import pandas as pd

from scipy.spatial.distance import cdist

from sklearn.metrics import silhouette\_samples, silhouette\_score

from matplotlib import pyplot as plt

from Hierarchical import Hierarchical,computeRandIndex

from sklearn.decomposition import PCA

**#Function to computer each cluster Sum Of Squared Errors**

def computeIndividualSSE (k\_cluster,labels,centroids,X):

Indi\_SSE=[];

for Kval in range(0,k\_cluster):

indices = [i for i, l in enumerate(labels) if l == Kval]

A0=np.array(np.take(X[:,0],indices));

A1=np.array(np.take(X[:,1],indices));

A=np.column\_stack((A0, A1))

C\_temp=centroids[Kval,:];

B0=np.array(np.tile(C\_temp[0],len(indices)));

B1=np.array(np.tile(C\_temp[1],len(indices)));

B=np.column\_stack((B0, B1));

indSSE=((A-B)\*\*2).sum();

Indi\_SSE.append(indSSE);

return Indi\_SSE;

**# Importing the data and setting up precision**

plt.rcParams['figure.figsize'] = (16, 9)

plt.style.use('ggplot')

np.set\_printoptions(precision=5, suppress=True)

data = pd.read\_csv('HW3-StudentData2.csv')

print('The data dimensions',data.shape)

data.head()

**# Getting the values and plotting it**

d0=data['StudentId'].values;

d1 = data['Phys'].values

d2 = data['Maths'].values

d3=data['English'].values;

d4=data['Music'].values;

X = np.array(list(zip(d1, d2,d3,d4)))

**#PCA analysis for dimensionality reduction**

pca=PCA(n\_components=2)pca.fit(X);

print('PCA variance ratio is ',pca.explained\_variance\_ratio\_);

X=pca.transform(X);

**#plotting data and initializing all arrays and variables**

data.iloc[:,1:].plot()

plt.figure();

TotalSSE=0;

**#final**

FSSE=[];

Fdistortions=[];

Findi\_clusters=[];

FSilh\_avg=[];

Flabels=[];

Karray=[3,4,5,6,7,8];

itr=-1;

**#This is the Main Logic of the program. Here I have done the below requirement**

**#For each cluster values(3,4,5,6,7,8), I have done 3 iterations and choses the best iteration**

**#Arrays starting with “J” specifies iterative arrays and among them the best one is stored #in “F” arrays.**

for i in Karray:

itr=itr+1;

JSilh\_avg=[];

JSSE=[];

Jdistortions=[];

Jindi\_clusters=[];

Jlabels=[];

Jcentroids=[];

for j in range(0,3): #his is for running 3 times for each K value

**# Number of clusters**

kmeans = KMeans(n\_clusters=i)

# Fitting the input data

kmeans = kmeans.fit(X)

**# Centroid values**

centroids = kmeans.cluster\_centers\_

if(j==0 or JSSE[0] >=kmeans.inertia\_):

JSilh\_avg=[];

JSSE=[];

Jdistortions=[];

Jindi\_clusters=[];

Jlabels=[];

Jcentroids=[];

**# Getting the cluster labels**

labels=kmeans.predict(X);

Jlabels.append(kmeans.predict(X));

Jindi\_clusters.append(computeIndividualSSE(i,labels,centroids,X));

JSSE.append(kmeans.inertia\_);

#print('JSSE size',len(JSSE));

print('clustering centers k =',i,' Iteration is ',j);

print('Total SSE ',JSSE);

print('centroids ',centroids);

print('Individual clusters SSE values are',Jindi\_clusters)

silhouette\_avg = silhouette\_score(X, labels);

JSilh\_avg.append(silhouette\_avg);

Jcentroids.append(centroids);

Jdistortions.append(sum(np.min(cdist(X, kmeans.cluster\_centers\_, 'euclidean'), axis=1)) / X.shape[0])

**#For each K values cluster labels, Centroids, Individual Cluster SSE ,Silhoutte Coefficient #are stored in “F” Arrays and displayed the results**

Flabels.append(Jlabels);

Findi\_clusters.append(Jindi\_clusters);

FSSE.append(JSSE);

Fdistortions.append(Jdistortions);

#print('FSSE shape',len(FSSE));

FSilh\_avg.append(JSilh\_avg);

print(' ');

print(' ');

print ('Selected Iteration Details for clusters',i);

print('The Final lables are ',Flabels[itr]);

print('The final cluster centers are',Fcentroids[itr]);

print('Individual clusters SSE values are',Findi\_clusters[itr]);

print('Final Total SSE is ',FSSE[itr]);

print('The Final average silhouette\_score is :', FSilh\_avg[itr]);

print(' ');

print(' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

**#The following plots are shown**

**#a. ClusterArray vs Sum Of Squared Errors**

**#b. ClusterArray vs Average Silhouette Coefficients**

**#c. Elbow method for choosing optimal K**

f1 = plt.figure()

f2 = plt.figure()

f3 = plt.figure()

ax1 = f1.add\_subplot(111)

ax1.plot(Karray,FSSE)

ax2 = f2.add\_subplot(111)

ax2.plot(Karray,FSilh\_avg)

ax3 = f3.add\_subplot(111)

plt.plot(Karray, Fdistortions, 'bx-')

plt.xlabel('k')

plt.ylabel('Distortion')

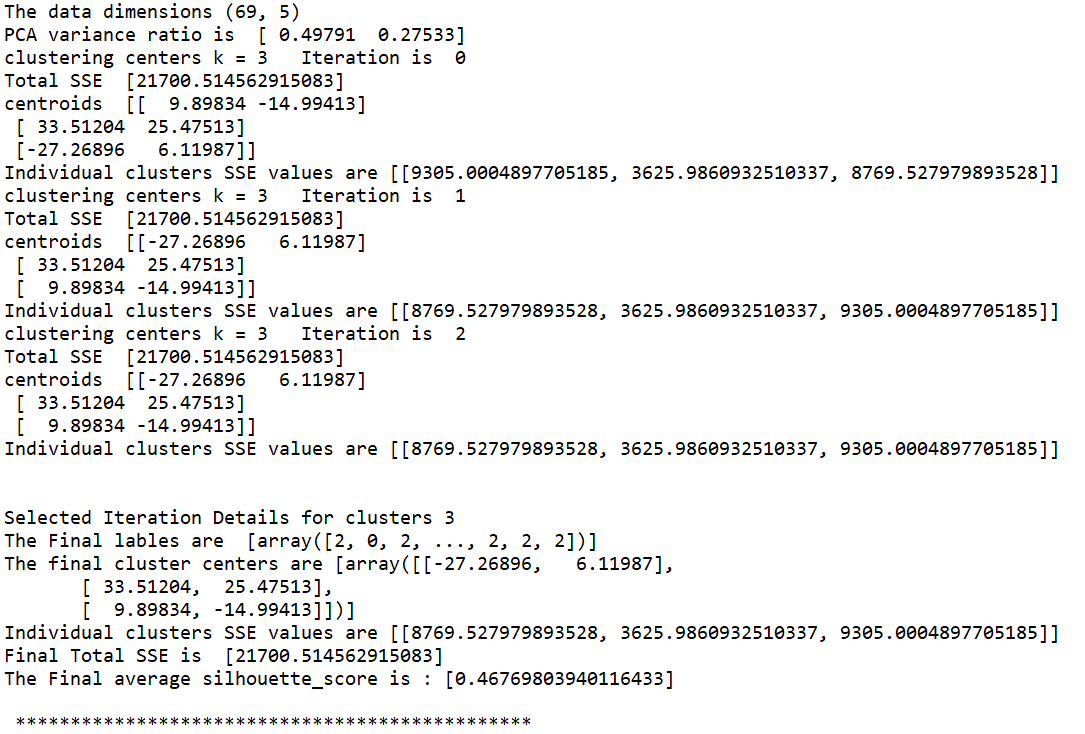
plt.title('The Elbow Method showing the optimal k')

plt.show()

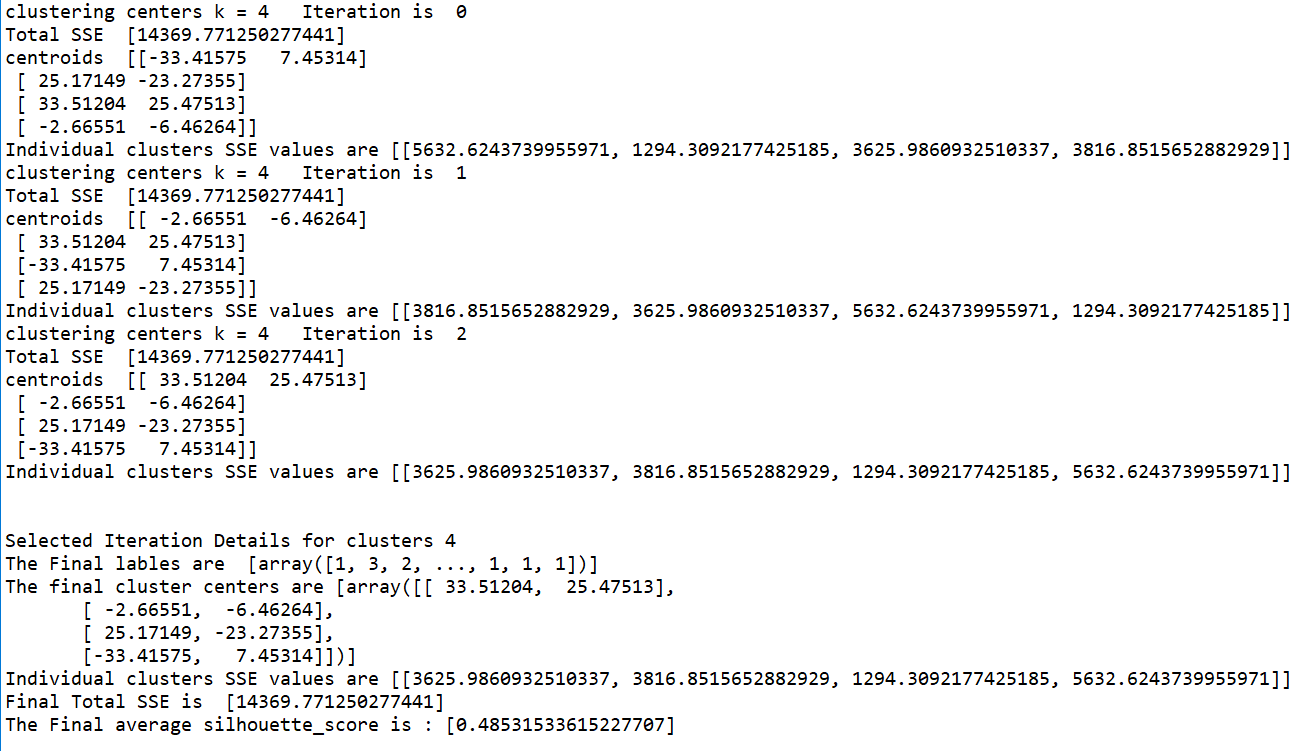
* 1. Show the cluster centers, SSE values of the clusters, and the total SSE value for the clustering for each value of k.

**Output**

**Cluster K=3**

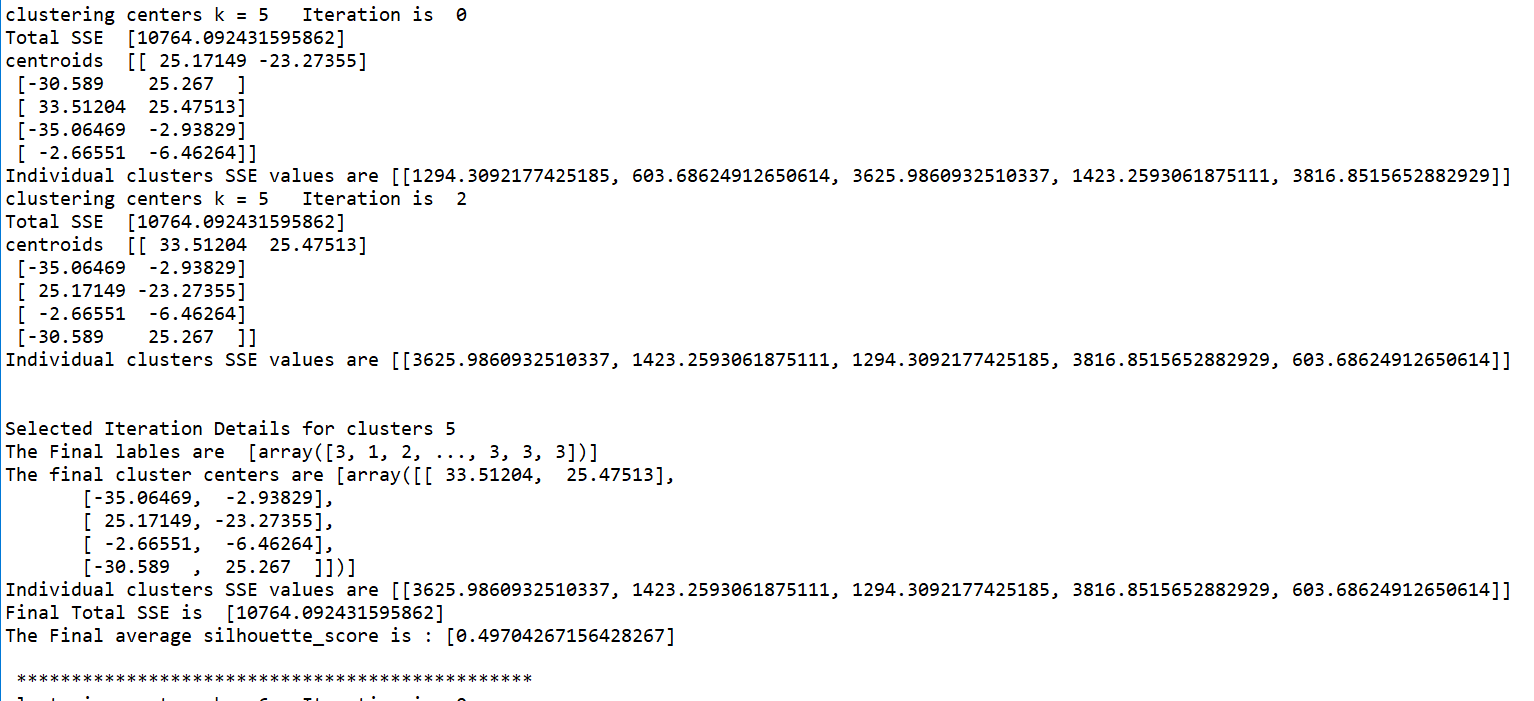


**Cluster K=4**



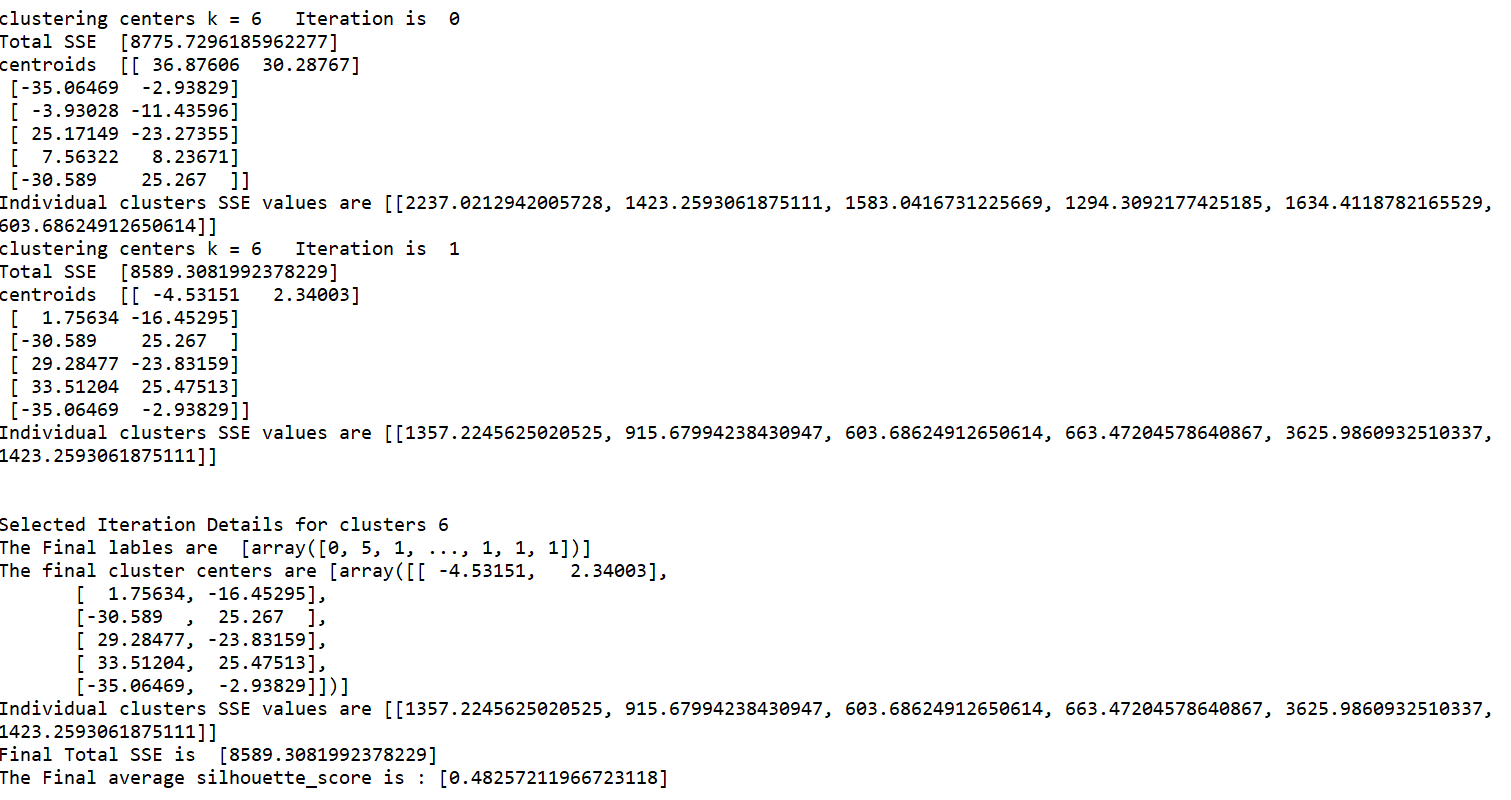
**Cluster k=5**

**There are two iterations because of the “IF” condition in code specifying to take least SSE value**

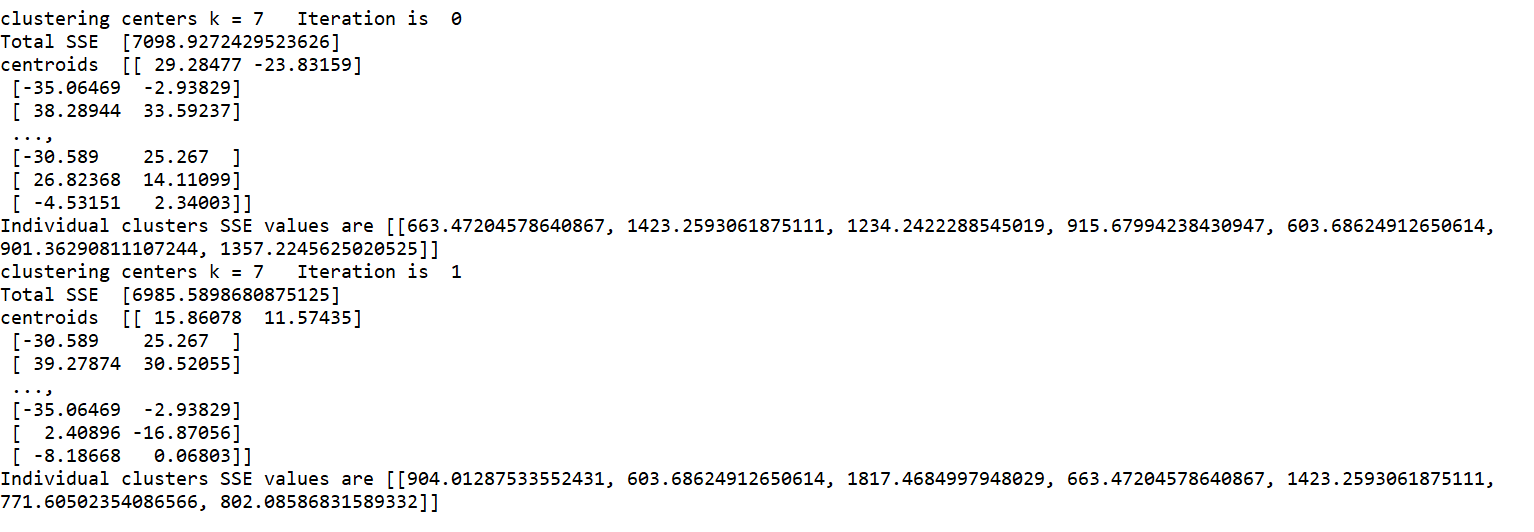


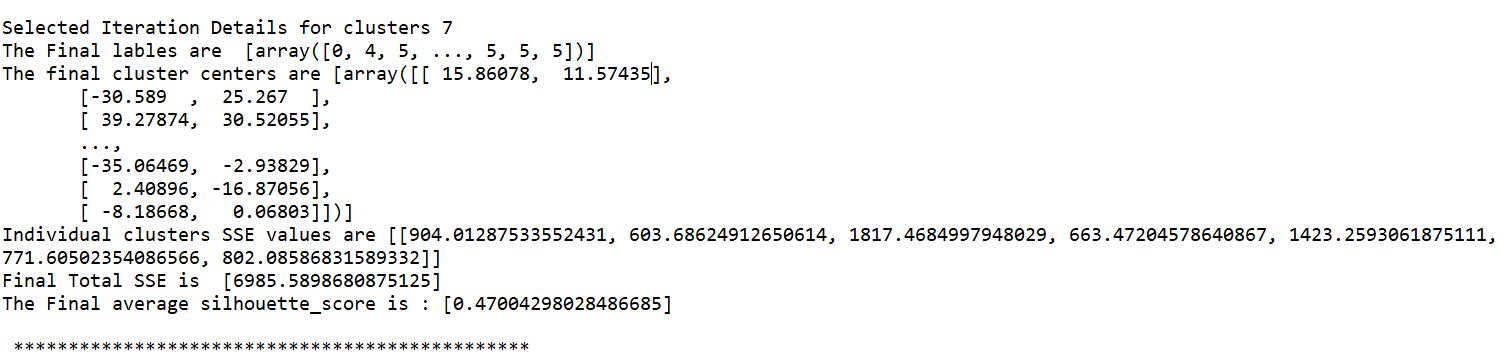
**Cluster K=6**

**There is only one iteration because of the “IF” condition in code specifying to take least SSE value**



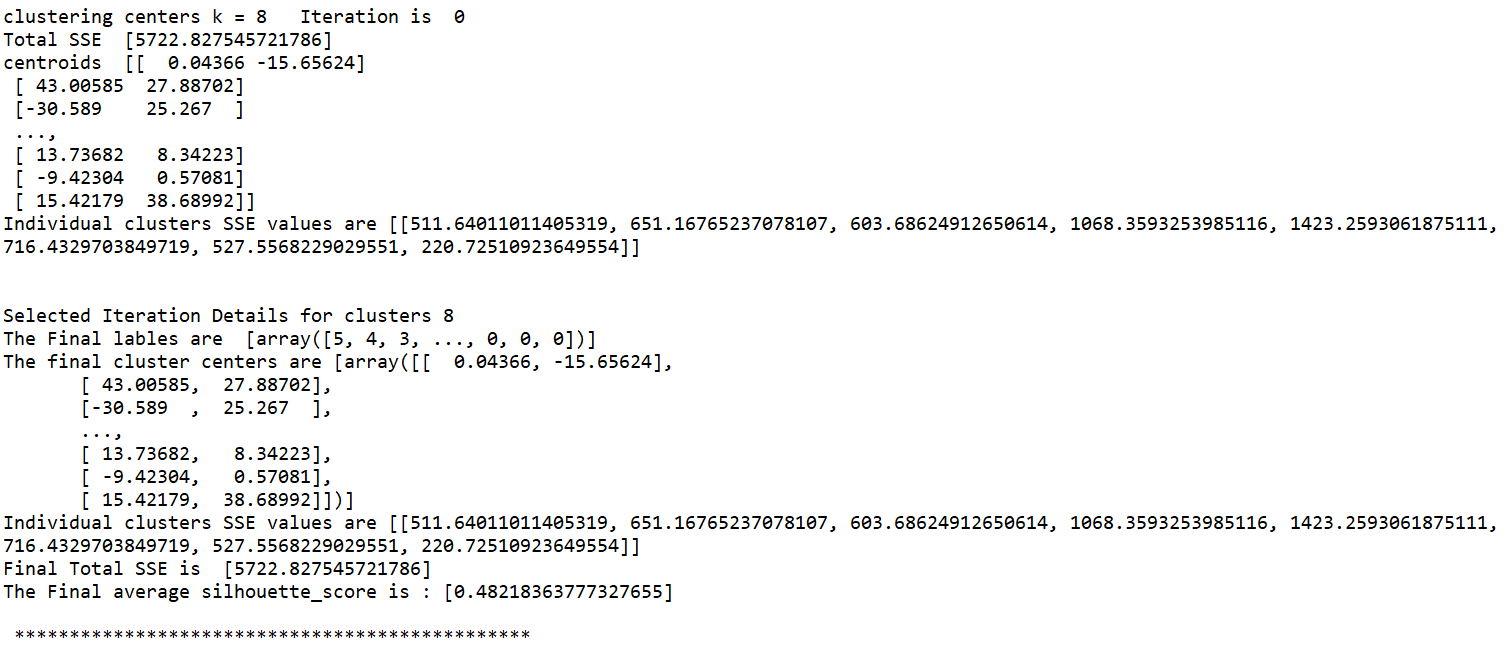
**Cluster K=7**



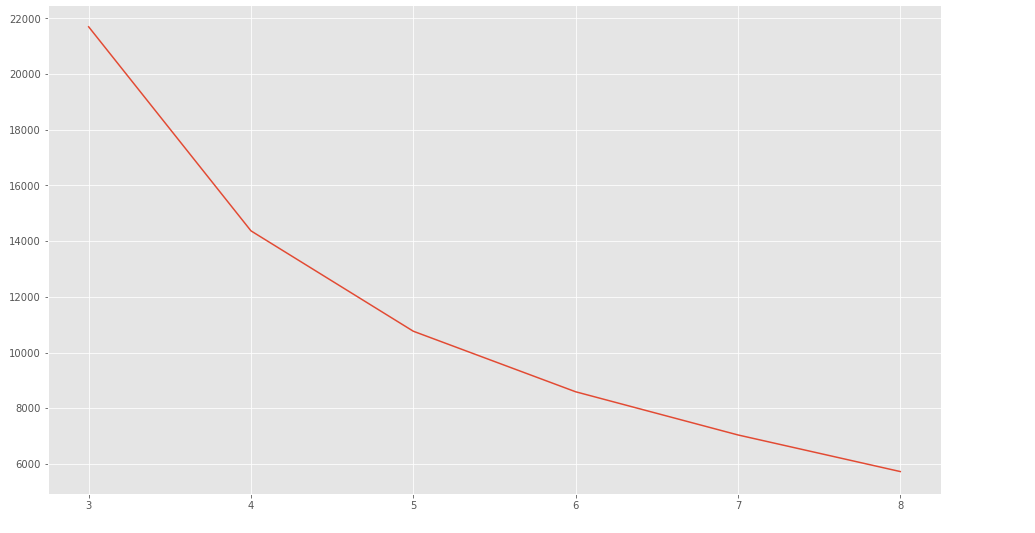


**Cluster k=8**

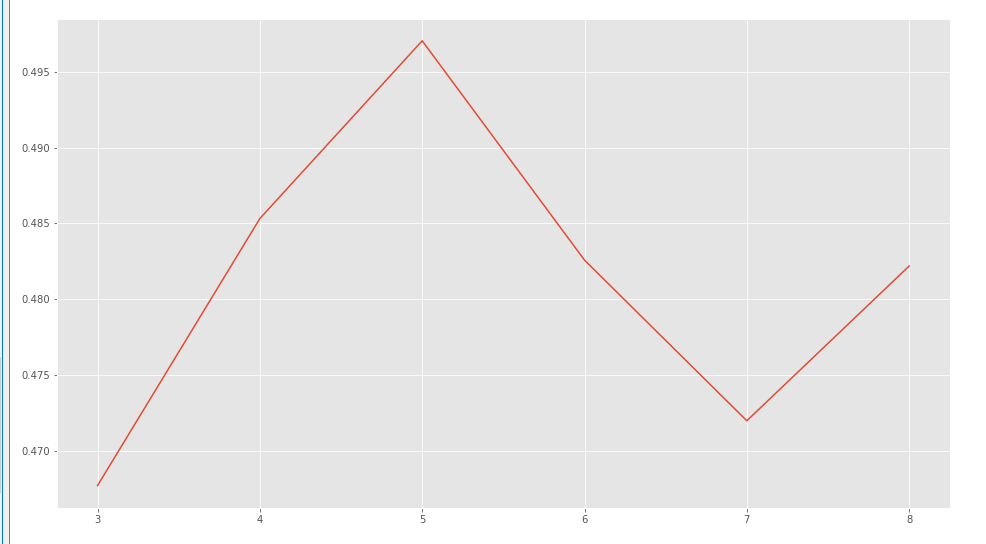
**There is only one iteration because of the “IF” condition in code specifying to take least SSE value**



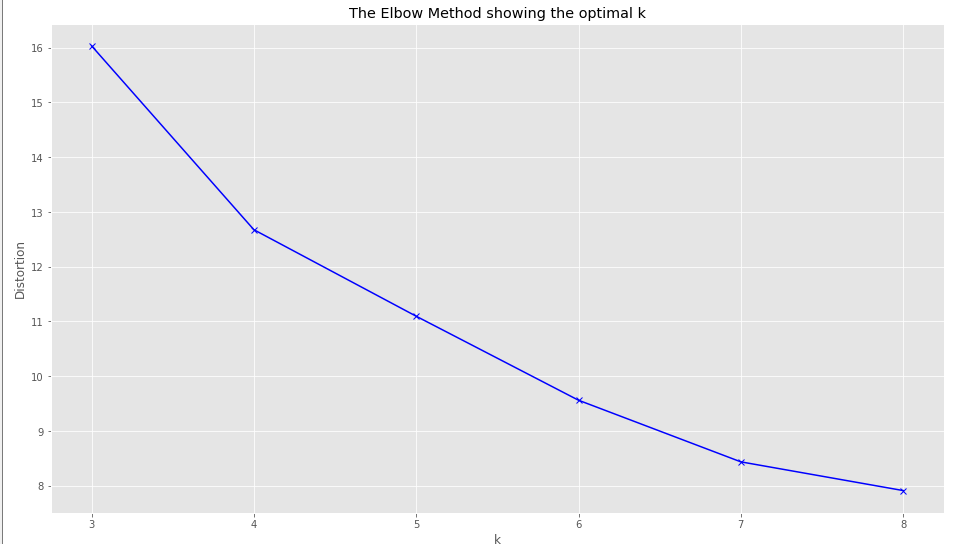
* 1. Plot the total SSE value against the values of k.



* 1. Plot showing Silhouette coefficient against the K values



Plot showing K against distortions using elbow method to get optimal k



* 1. How many clusters would you form in this dataset? Justify your answer. For your choice of the best number of clusters, report the centroids of all the clusters and their SSE values (Call this as Clustering-1).

**#Analyzing best cluster and printing SSE and Total SSE**

print('SO whats the best clusters needed?????')

selectedkindex=FSilh\_avg.index(max(FSilh\_avg));

selectedkval=Karray[FSilh\_avg.index(max(FSilh\_avg))];

print('The best clustering is selcted based on Silhoutte coefficeient and the value is ',selectedkval)

print('The Final lables are ',Flabels[selectedkindex]);

print('The final cluster centers are',Fcentroids[selectedkindex]);

print('Individual clusters SSE values are',Findi\_clusters[selectedkindex]);

print('Final Total SSE is ',FSSE[selectedkindex]);

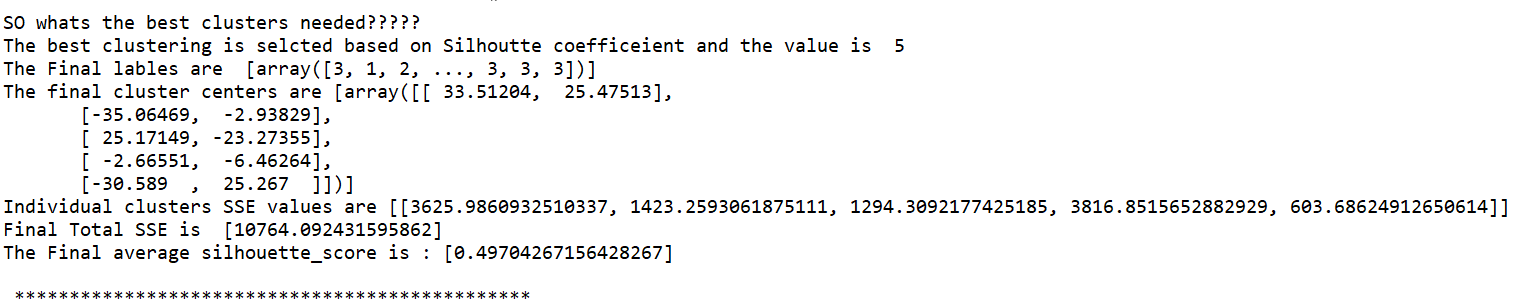
print('The Final average silhouette\_score is :', FSilh\_avg[selectedkindex]);

print(' ');

print(' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

**Output**

I think 5 clusters gives better result and I have deduced this based on Elbow method evaluation and plot showing K values vs Silhouette coefficients.



* 1. Generate 100 random 4-dimensional random data points such that each attribute can take values between 0 and 100. With this dataset form the same number of clusters as selected by you in (d) above. Report the centroids and populations of the clusters. Compare the total SSE for this random dataset with the SSE for the clustering of the provided dataset. Compare and comment on the differences between the two total SSE values.

**#generating random data points**

D1=np.random.uniform(low=0,high=100,size=100)

D1=np.int\_(D1);

D2=np.random.uniform(low=0,high=100,size=100)

D2=np.int\_(D2);

D3=np.random.uniform(low=0,high=100,size=100)

D3=np.int\_(D3);

D4=np.random.uniform(low=0,high=100,size=100)

D4=np.int\_(D4);

NewX = np.array(list(zip(D1, D2,D3,D4)));

#PCA analysis for dimensionality reduction

pcaNew=PCA(n\_components=2)

pcaNew.fit(NewX);

print('PCA variance ratio is ',pcaNew.explained\_variance\_ratio\_);

NewX=pcaNew.transform(NewX);

kmeansNew = KMeans(n\_clusters=selectedkval)

kmeansNew = kmeansNew.fit(NewX)

labelsNew = kmeansNew.predict(NewX)

centroidsNew = kmeansNew.cluster\_centers\_

SSENew=kmeans.inertia\_;

print('clustering centers k =',selectedkval,' for random data points');

print('RD-The Final lables are ',labelsNew);

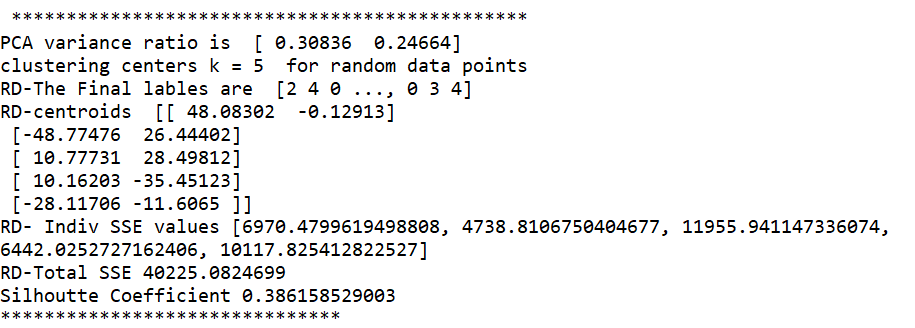
print('RD-centroids ',centroidsNew);

print ('RD-Total SSE',SSENew);

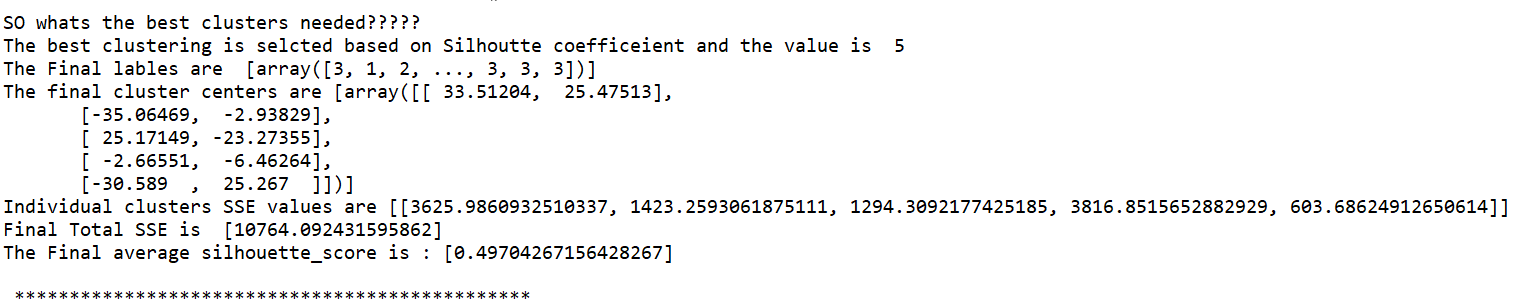
print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

**Output: RD stands for Random Data**

**A)**



**B) Best K=5 cluster values**



**Comment:** Based on the Total SSE and Silhouette coefficient, the data points are closed to their centroid in k=5 cluster existing data (B) screenshot when compared with A screenshot which is having more SSE and less Silhouette coefficient.

1. Perform hierarchical clustering for the students’ scores dataset. Generate and show dendrograms for the cases (i) Single-Linkage clustering (Clustering-2), and (ii) Complete-Linkage clustering (Clustering-3). Use Euclidean distance for computing distance between data points. Report the following in the submitted work: (Use Matlab functions pdist and linkage, or any other similar toolbox.) Make sure the dendrogram shows all points at its lowest level.

**#Import all the necessary packages**

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from scipy.cluster.hierarchy import cophenet

from scipy.spatial.distance import pdist

from scipy.cluster.hierarchy import dendrogram, linkage

from scipy.cluster.hierarchy import fcluster

from sklearn.decomposition import PCA

**#compute rand index for cluster 2 and cluster 3**

def computeRandIndex(A,B,rCount):

a=0;

b=0;

c=0;

d=0;

for i in range(rCount):

for j in range(i+1,rCount):

if(A[i]==A[j] and B[i]==B[j]):

a=a+1;

elif(A[i]!=A[j] and B[i]==B[j]):

b=b+1;

elif (A[i]!=A[j] and B[i]!=B[j]):

d=d+1;

else:

c=c+1;

print('Rand Index A, B , C , D values down here ');

print('a,b,c,d',a,b,c,d);

randIndex=(a+d)/(a+b+c+d);

return randIndex;

def Hierarchical():

plt.rcParams['figure.figsize'] = (16, 9)

plt.style.use('ggplot')

np.set\_printoptions(precision=5, suppress=True)

**# Importing the dataset**

data = pd.read\_csv('HW3-StudentData2.csv')

print(data.shape)

data.head()

**# Getting the values and plotting it**

d1 = data['Phys'].values

d2 = data['Maths'].values

d3=data['English'].values;

d4=data['Music'].values;

X = np.array(list(zip(d1, d2,d3,d4)))

data.iloc[:,1:].plot()

plt.figure();

**#Single Linkage and Complete link clustering method**

Z\_single = linkage(X, 'single')

Z\_complete = linkage(X, 'complete')

c\_single, coph\_dists\_single = cophenet(Z\_single, pdist(X));

c\_complete, coph\_dists\_complete = cophenet(Z\_complete, pdist(X));

**# calculate full dendrogram**

f1 = plt.figure(figsize=(25, 10))

ax1 = f1.add\_subplot(111)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('sample index')

plt.ylabel('distance')

dendrogram(

Z\_single,

leaf\_rotation=90., # rotates the x axis labels

leaf\_font\_size=8.,

# font size for the x axis labels

)

f2 = plt.figure(figsize=(25, 10))

ax2 = f2.add\_subplot(111)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('sample index')

plt.ylabel('distance')

dendrogram(

Z\_complete,

leaf\_rotation=90., # rotates the x axis labels

leaf\_font\_size=8., # font size for the x axis labels

)

plt.show()

max\_c = 4

clusters\_single = fcluster(Z\_single, max\_c, criterion='maxclust')

print(clusters\_single)

max\_c = 4

clusters\_complete = fcluster(Z\_complete, max\_c, criterion='maxclust')

print(clusters\_complete)

**#single link**

index\_single\_one =[i for i, x in enumerate(clusters\_single) if x == 1]

index\_single\_two =[i for i, x in enumerate(clusters\_single) if x == 2]

index\_single\_three =[i for i, x in enumerate(clusters\_single) if x == 3]

index\_single\_four =[i for i, x in enumerate(clusters\_single) if x == 4]

centroid\_single\_one=calculateAvg(index\_single\_one,X);

centroid\_single\_two=calculateAvg(index\_single\_two,X);

centroid\_single\_three=calculateAvg(index\_single\_three,X);

centroid\_single\_four=calculateAvg(index\_single\_four,X);

print('single link centroids for 4 clusters');

print(centroid\_single\_one);

print(centroid\_single\_two);

print(centroid\_single\_three);

print(centroid\_single\_four);

print('')

print('')

**#complete link**

index\_complete\_one =[i for i, x in enumerate(clusters\_complete) if x == 1]

index\_complete\_two =[i for i, x in enumerate(clusters\_complete) if x == 2]

index\_complete\_three =[i for i, x in enumerate(clusters\_complete) if x == 3] index\_complete\_four =[i for i, x in enumerate(clusters\_complete) if x == 4]

centroid\_complete\_one=calculateAvg(index\_complete\_one,X);

centroid\_complete\_two=calculateAvg(index\_complete\_two,X);

centroid\_complete\_three=calculateAvg(index\_complete\_three,X);

centroid\_complete\_four=calculateAvg(index\_complete\_four,X);

print('complete link centroids for 4 clusters');

print(centroid\_complete\_one);

print(centroid\_complete\_two);

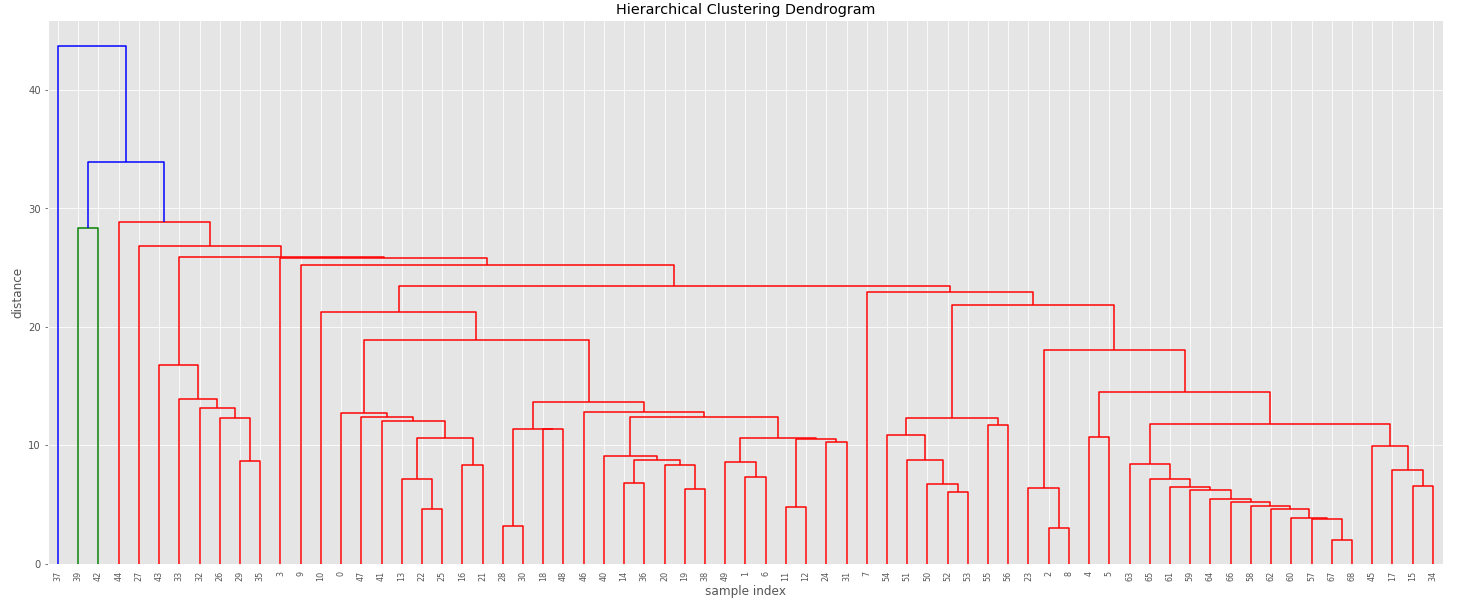
print(centroid\_complete\_three);

print(centroid\_complete\_four);

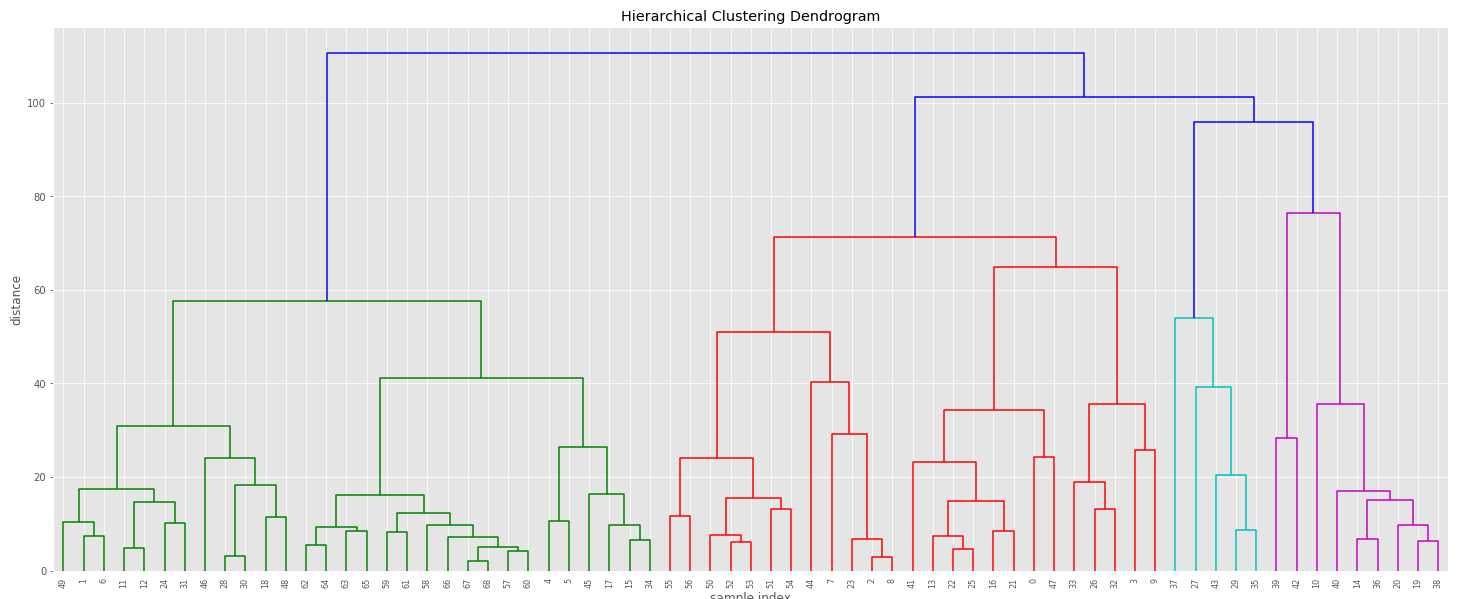
return clusters\_single;

1. Dendrograms for the two clusterings (Clustering-2 and Clustering-3)

**Single Link**

****

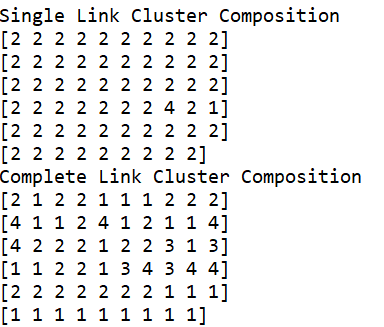
**Complete Link**

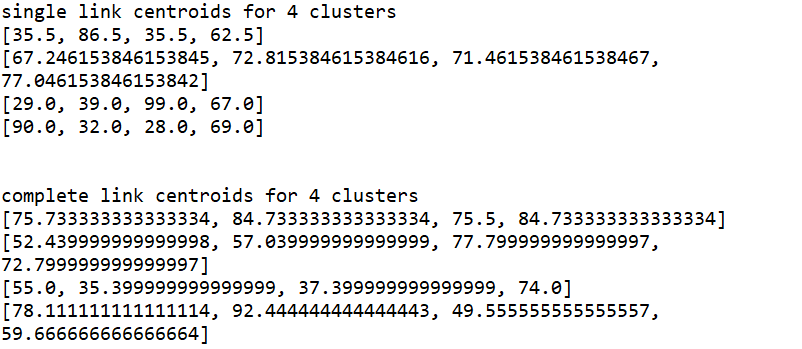


1. Cluster compositions for each case when we need only four clusters. Write ALL the data points included in each cluster and compute their centroids.

**Cluster Compositions**

**Single Link & Complete Link**





1. Comment on any differences in the cluster centres and cluster compositions for the two different clustering’s as performed in (b) above.

**Answer**:

In Single Link, the data points are more concentrated in cluster 2 and while complete link they are diversified

Also, the cluster centroids are close to each other in complete link whereas they are far from each other in single link.

From the above we can say that there are no joining data points between second cluster and the remaining clusters in single link.

Complete link did a better job of clustering data points to form 4 clusters as they are evenly poised.

Since the data points belong to 4-dimensional space, it is very difficult to scatter plot them to show the difference in clustering

1. Compute Rand Index for the comparison of Clustering-2 and Clustering-3 and show the counts a, b, c, and d as determined for computing the Rand index. Explain the meaning of each count and why such counts have been obtained for this dataset and their clustering’s

**Answer**:

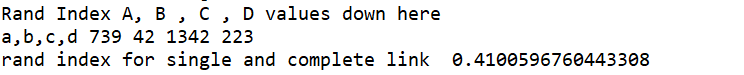
Randi Index measures similarity between two data clustering.

R= (a+b)/(a+b+c+d)

* a value measures the number of pair of data in S that are in same subset in X and in same subset in Y.
* b value measures the number of pair of data in S that are in different subset in X and in different subset in Y.
* c value measures the number of pair of data in S that are in same subset in X and in different subset in Y.
* d value measures the number of pair of data in S that are in different subset in X and in same subset in Y.

a+b consider as number of agreements between X and Y , c+d is consider as number of disagreements between Xand Y.

Denominator represents total number of pairs.so, rand index is frequency of occurrence of agreements over the total pairs



Here Rand Index is 0.41 means that both clustering algorithm agreed for 41% among the data in clustering regions. Both Single and complete link are hierarchical based clustering and hence the score is approximately 40%.

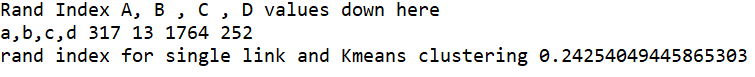
1. Compute Rand Index for the comparison of Clustering-1 and Clustering-2 and show the counts a, b, c, and d as determined for computing the Rand index. Explain the meaning of each count and why such counts have been obtained for this dataset and these clusterings in this comparison.

**Answer**

Randi Index measures similarity between two data clustering.

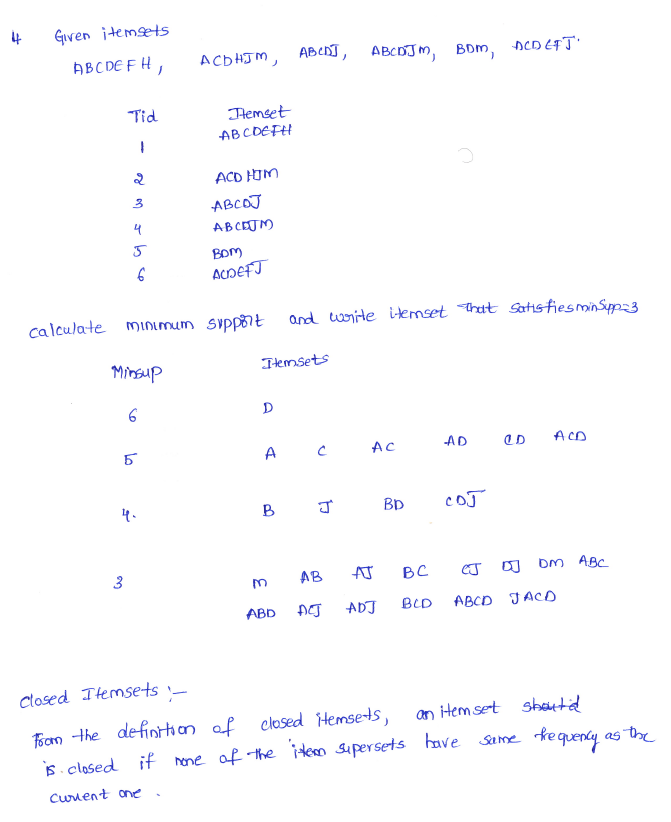
R= (a+b)/(a+b+c+d)

* a value measures the number of pair of data in S that are in same subset in X and in same subset in Y.
* b value measures the number of pair of data in S that are in different subset in X and in different subset in Y.
* c value measures the number of pair of data in S that are in same subset in X and in different subset in Y.
* d value measures the number of pair of data in S that are in different subset in X and in same subset in Y.



Here Rand Index is 0.24 means that both clustering algorithm agreed for 24% among the data in clustering regions which means that they are clustering differently. Since K-means depend on the initial cluster centres where single link is density based algorithm and hence the index is very less.

1. Show the execution tree for the CHARM algorithm for finding all the closed itemsets for the dataset containing the following transactions: ABCDEFH, ACDHJM, ABCDJ, ABCDJM, BDM, ACDEFJ



1. For the same data as in #4 above, show execution of the algorithm for finding all the maximal itemsets.

