دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

Kasra Khalafi: 9531306

Homework 2

Data Mining

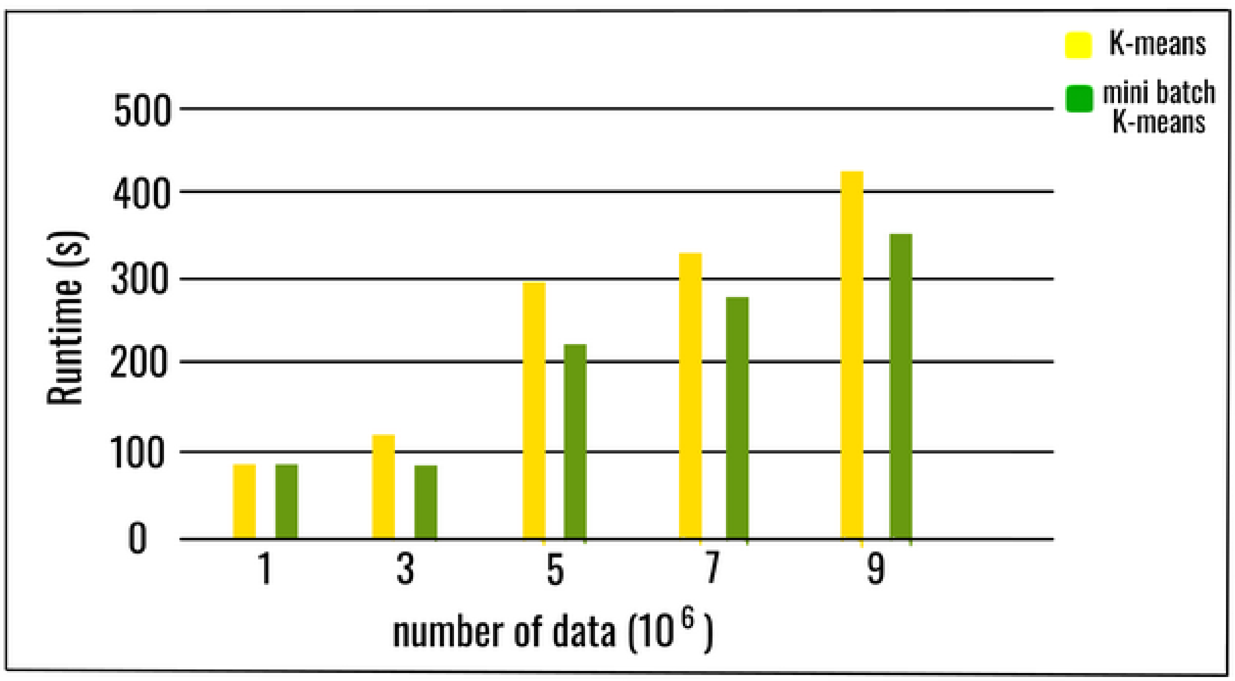
Dr.Ehsan Nazerfard

Winter Term 2019

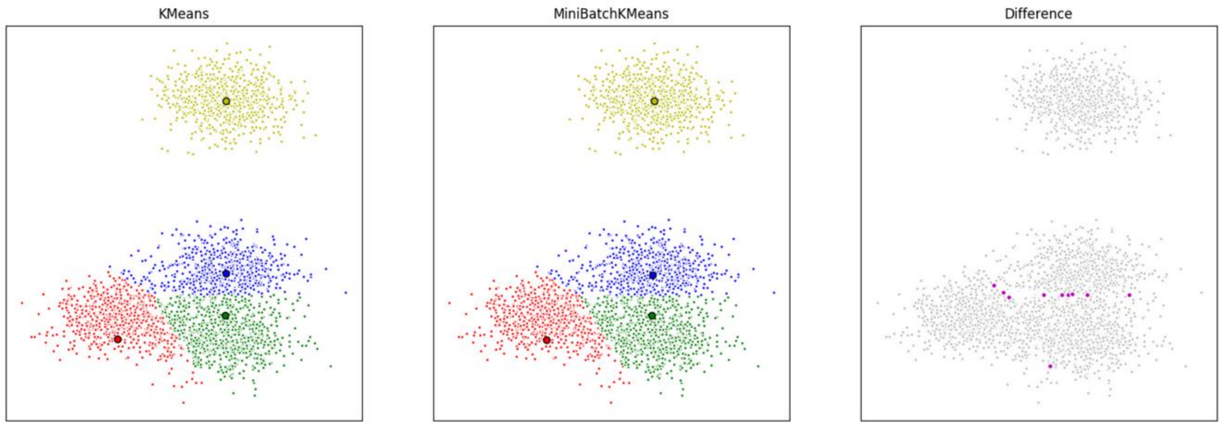
1- As written in question, K-Means is an instance based algorithm which means that it needs all data to compute its clustring, as it is obvious, when dataset is big, more computation and storage is required which makes it susceptible to big data.
For the computational problem, some different methods have been developed. One of these methods is Mini Bach K-Means:

In Mini Bach K-Means, small random batches of data of a fixed size is used to be stored in memory. Each iteration a new random sample from the dataset is obtained and used to update the clusters and this is repeated until convergence. Each mini batch updates the clusters using a convex combination of the values of the prototypes and the data, applying a learning rate that decreases with the number of iterations.

As the number of iterations increases, the effect of new data is reduced. As empirical studied showed, Mini Bach K-Means saves a great computational time but in instead we miss some cluster quality. As illustrated below, you could spectate the computational difference of the two K-Means and Mini Batch K-Means:



As mentioned above, Mini Batch K-Means is faster, but as nothing is free, it gives slightly different results as the normal K-Means. Below we have the illustrated picture of both methods for a given data set and the different clustered data are showed:



2- Different initialization of centroids in K-Means could result in different clustring. For a better initialization, cetroids could be chosen from the data rather than a random place. Also for a better clustring, initialization could be run for several times till the best one to be selected. Also as PAM, medoid could be used so it would be robuster than K-Means. Also K-Means ++ Method could be used to initialized.

3-

a) True: for data points to be in a cluster, they must be in a distance threshold to a core point. This threshold in DBSCAN is called as Eps.

b) False: DBSCAN does not have assumption for the distribution of data in dataspace because it works with distance of data together and could cluster a convex data properly, so shape of data in dataspace is not crucial.

c) False: time complexity of DBSCAN is $O(n^2)$. In lower dimention space it could be reduced to $O(n * \log(n))$.

d) True : unlike K-Means, DBSCAN does not need # clusters but instead it requires information about the maximum radious of neigbouhood (Eps) and minimum number of points in an Eps neigbourhood of that point (MinPts).

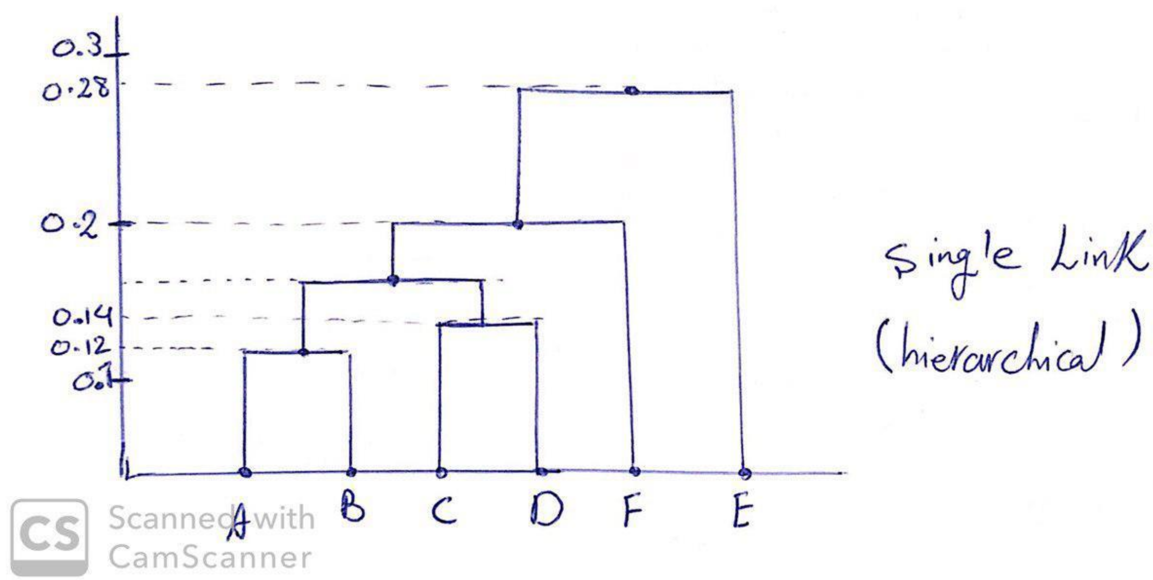
e) True: DBSCAN which is a Density Based Method can handle Noises, and because it clusters with the distance of the data together, so it does not get effected by outlier and outlier will be a separate cluster for itself

----

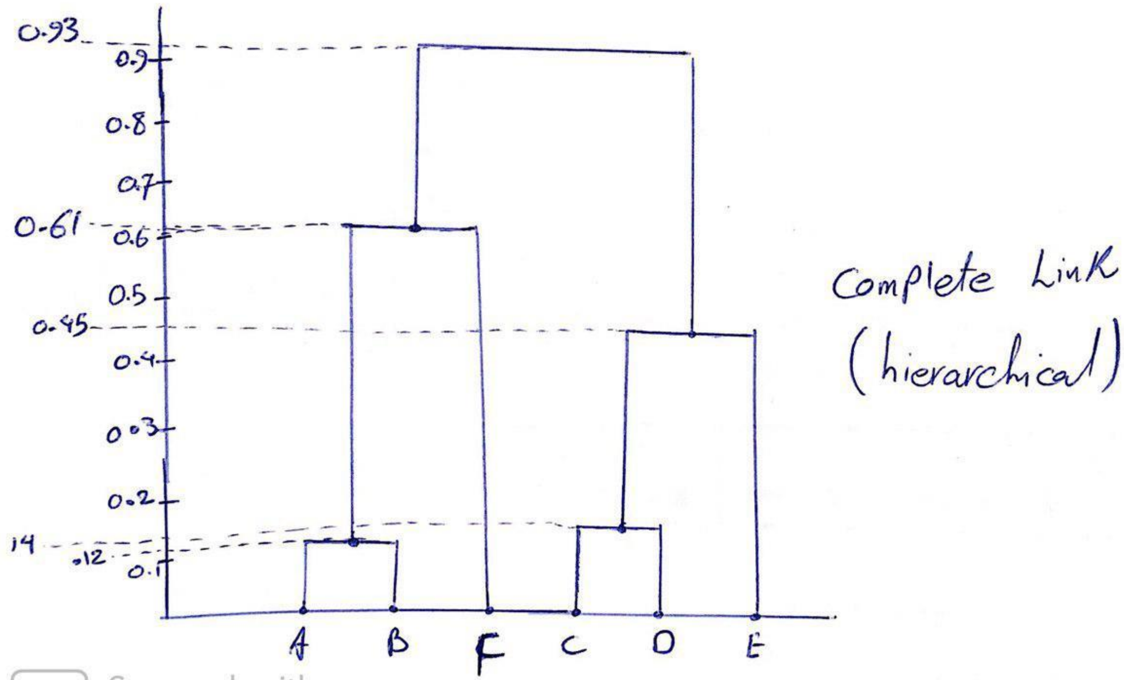
4- Distance Matrix for 6 data are as shown in table :

	A	B	C	D	E	F
A	0					
B	0.12	0				
C	0.51	0.25	0			
D	0.84	0.16	0.14	0		
E	0.28	0.77	0.70	0.45	0	
F	0.34	0.61	0.93	0.20	0.67	0

A) Dendogram of hierarchical clustring with single link will be like:



B) Dendogram of hierarchical clustring with complete link will be like:



5- Below are the Data given:

A. DBSCAN: because the shapes and points are not linearly seperable and are convex so for clustring DBSCAN which is a Density Based Clustring is used.if K-Means was used the clustring was seperated into a linear shaped clustring.

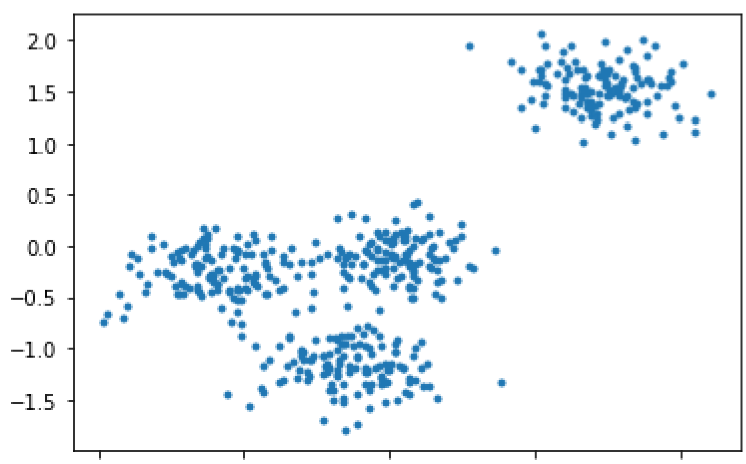
B. Both(DBSCAN & K-Means): This shape could be clustered by Both DBSCAN and K-Means. If we choose 3 cluster for K-Means we could have the clustring depicted. But if the number of clusters varies than 3 we would not have clustring depicted. Also for DBSCAN if Eps (Maximum radius of the neigbourhood) is greater than expected, then the two clusters in the right up of the picture would be shown as 1. But if it is choosed normal and goodly, we would have the depicted clustring.

C. K-Means: The clustring is linear and the data are belonging to their centers linearly. Number of clusters in this example is 4. But it can not be DBSCAN cause if it was DBSCAN, all the data shown above, would have been named to same cluster not 4 different clusters.

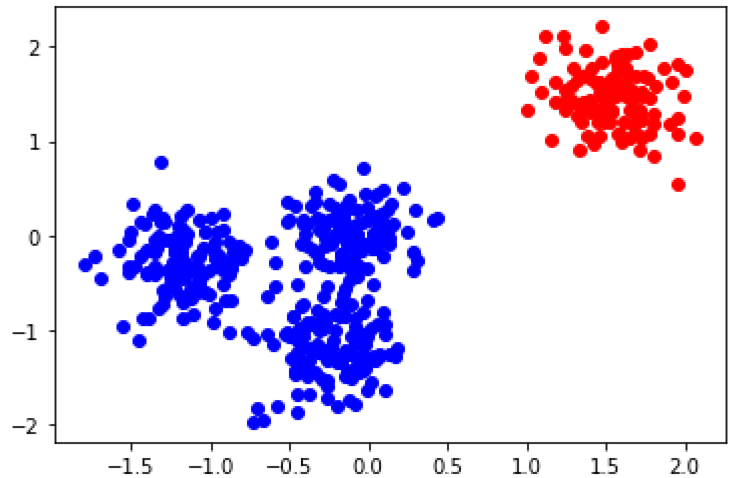
D. DBSCAN: This picture and data is convex and with Density Based Methods could be clustered. Because in DBSCAN we cluster with respect to the distance of data to each other which forms a Density Based, so it is possible to be clustered. But we can not cluster as shown with K-Means cause that is for Non-Convex data.

IMPLEMETATION: K-Means:

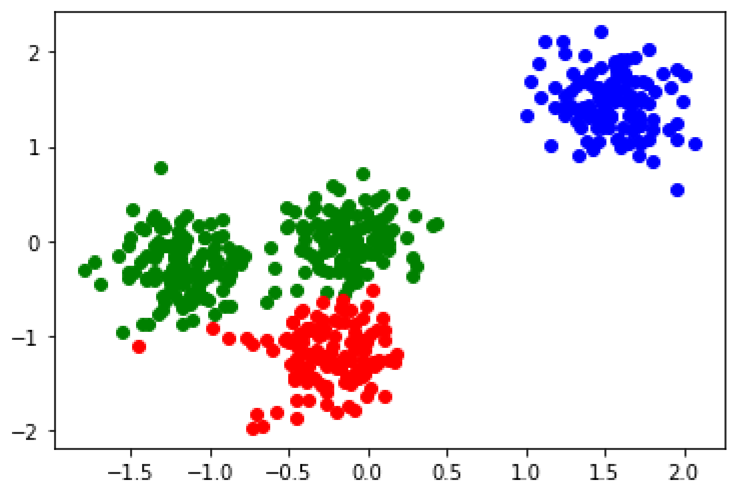
A. In this section, we will cluster the data with K=2, 3 and 4. First of all we illustrate the raw data in a 2D picture. Below is the raw data without being clustered:



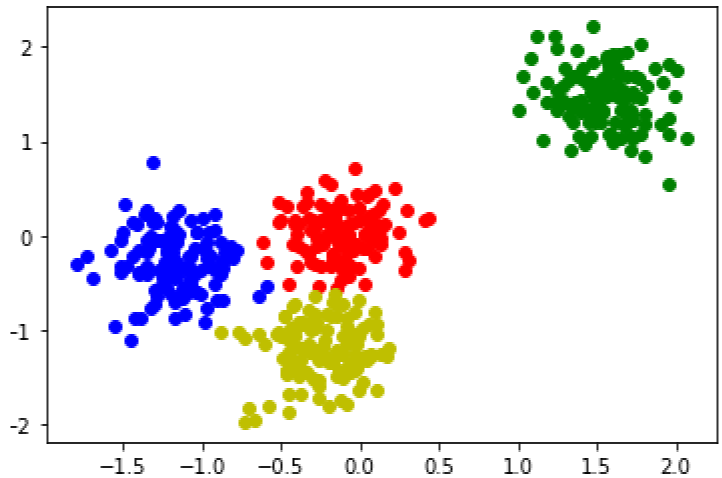
- K = 2: When K is choosed as 2, which means that we have 2 centroids (centers) and data should be clustered into 2 centers, the data will be clustered as depic



- K = 3: When K is choosed as 3, which means that we have 3 centroids (centers) and data should be clustered into 3 centers, the data will be clustered as depic:



- K = 4: When K is choosed as 4, which means that we have 4 centroids (centers) and data should be clustered into 4 centers, the data will be clustered as depic:



B. The average distance between the cluster center and the data points in that cluster or also called CLUSTER ERROR is :

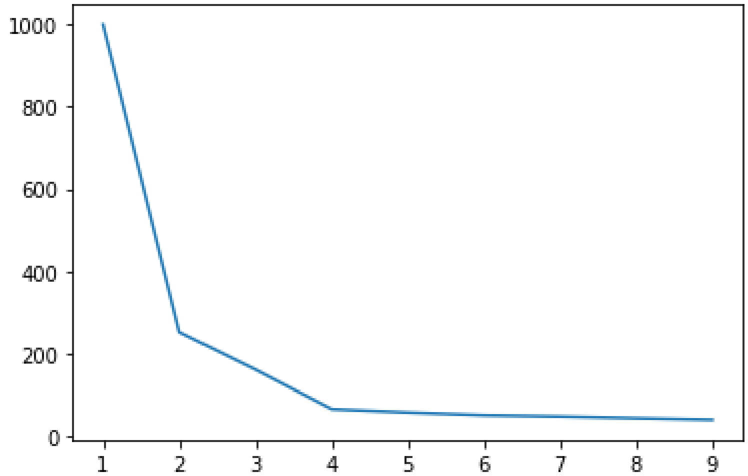
```
-----------------------------------------------------------------
Number of K = 2 ==> Error for Center 1 : 16.574429562386786
Number of K = 2 ==> Error for Center 2 : 234.54432910397898
-----------------------------------------------------------------
Number of K = 3 ==> Error for Center 1 : 103.80669511151572
Number of K = 3 ==> Error for Center 2 : 19.78808351897737
Number of K = 3 ==> Error for Center 3 : 16.574429562386786
-----------------------------------------------------------------
Number of K = 4 ==> Error for Center 1 : 18.05008119019905
Number of K = 4 ==> Error for Center 2 : 16.574429562386786
Number of K = 4 ==> Error for Center 3 : 16.76231341144023
Number of K = 4 ==> Error for Center 4 : 13.249071471981477
-----------------------------------------------------------------
```

C. Average cluster error as cluster error is :

```
-----------------------------------------------------------------
Number of K = 2 ==> Error 251.11875866636578
-----------------------------------------------------------------
Number of K = 3 ==> Error 140.16920819287986
-----------------------------------------------------------------
Number of K = 4 ==> Error 64.63589563600755
-----------------------------------------------------------------
```

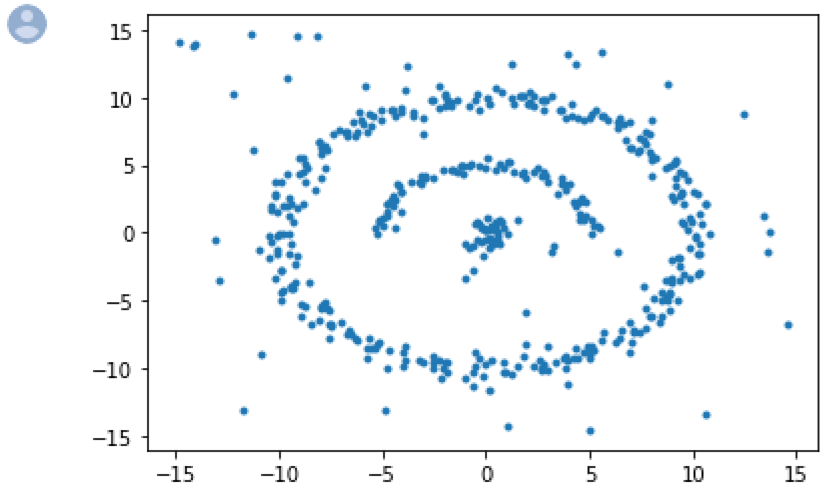
D. Running the k-means with 0<k<10 on Dataset1 and compute the clustering error and plot these errors are as below:

```
------------- PART D ----------------------------------------------
```



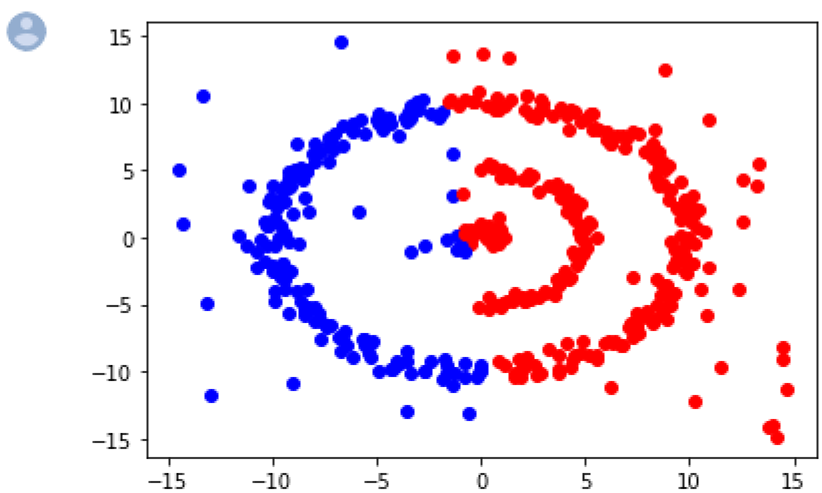

```
For K = 1 the cluster error is : 999.9999999999995
For K = 2 the cluster error is : 251.11875866636578
For K = 3 the cluster error is : 160.58467235336292
For K = 4 the cluster error is : 64.63589563600755
For K = 5 the cluster error is : 56.64319739158328
For K = 6 the cluster error is : 50.110090716719355
For K = 7 the cluster error is : 47.13932662079585
For K = 8 the cluster error is : 43.03147967797507
For K = 9 the cluster error is : 39.16549221272648
```

E. Elbow algorithms tells that where there is a big change in slope, that K should be selected as K. In this method cause for 1 cluster there is big cluster error, with k = 2 this error has been decreased a lot. Also with K = 4 we have a big change in the slope of the cluster errors so K = 2 and K = 4 both could be chosen but K = 4 is decent because we have a huge error with K = 1 which causes big change in slope of cluster error when we increase K = 2.
So K = 4 is more decent in this example.
F. This all above methods will be again run in Dataset2.
The raw data in a 2D space is illustrated below:

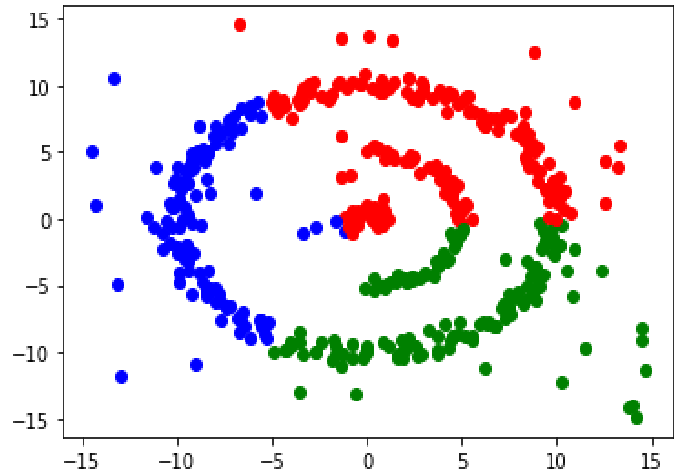


For K = 2, 3 and 4 clustrings are as shown below:

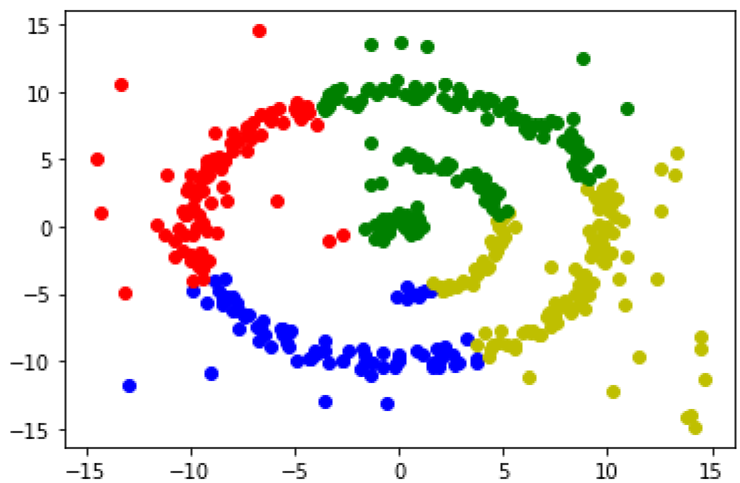
- K = 2:



- K = 3:



- K = 4:



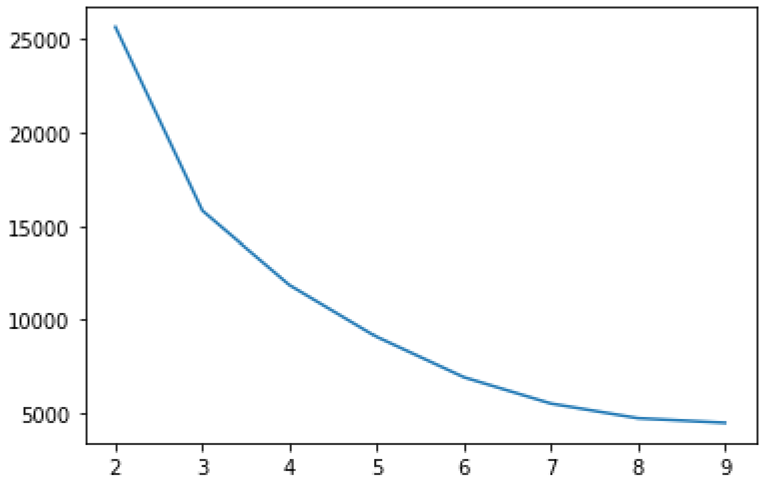
- B) for Dataset2 :

```
--------------------------------------------------------------
Number of K = 2 ==> Error for Center 1 : 14079.604509004399
Number of K = 2 ==> Error for Center 2 : 11645.544161842341
--------------------------------------------------------------
Number of K = 3 ==> Error for Center 1 : 5254.615071103115
Number of K = 3 ==> Error for Center 2 : 6697.768379301496
Number of K = 3 ==> Error for Center 3 : 4093.1021725723167
--------------------------------------------------------------
Number of K = 4 ==> Error for Center 1 : 2120.889002283974
Number of K = 4 ==> Error for Center 2 : 4220.703756717153
Number of K = 4 ==> Error for Center 3 : 1998.3079717373403
Number of K = 4 ==> Error for Center 4 : 3249.9983425575824
--------------------------------------------------------------
```

- C) for Dataset2:

```
----------------------------------------------------------------
Number of K = 2 ==> Error 25698.728221815847
----------------------------------------------------------------
Number of K = 3 ==> Error 16045.485622976928
----------------------------------------------------------------
Number of K = 4 ==> Error 11484.704578066108
----------------------------------------------------------------
```

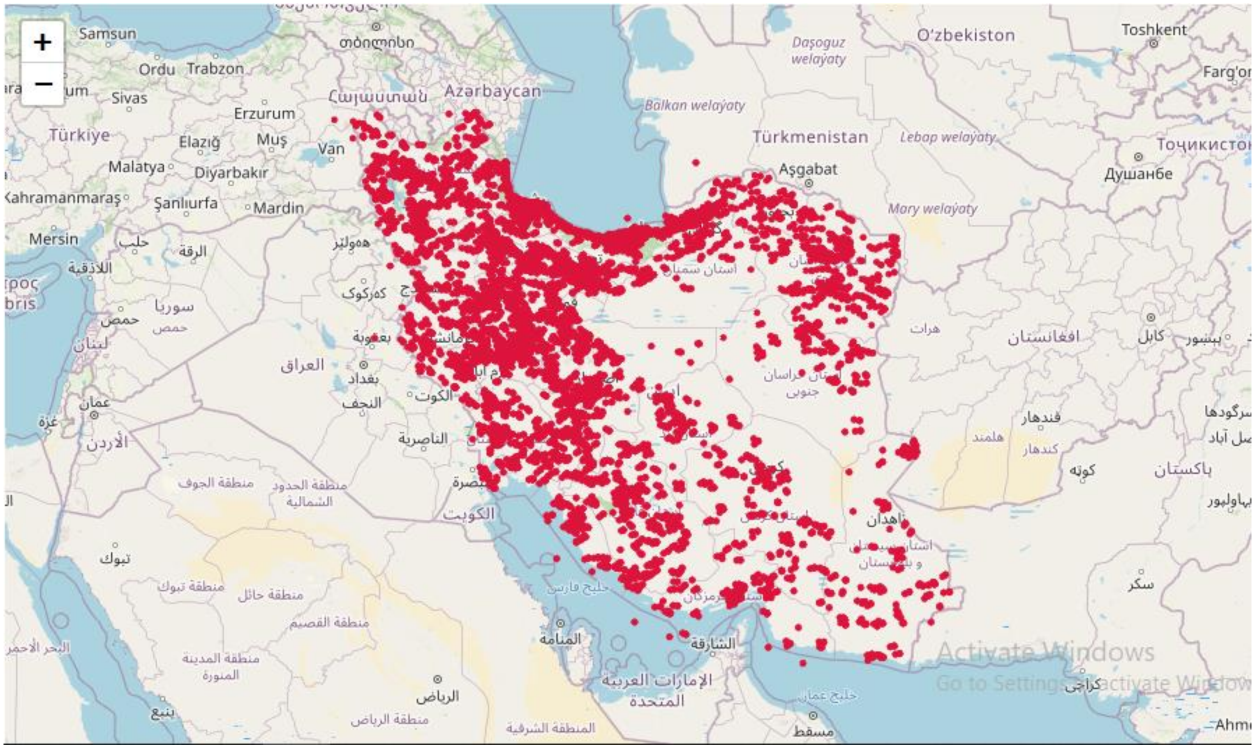
- D) for Dataset2:



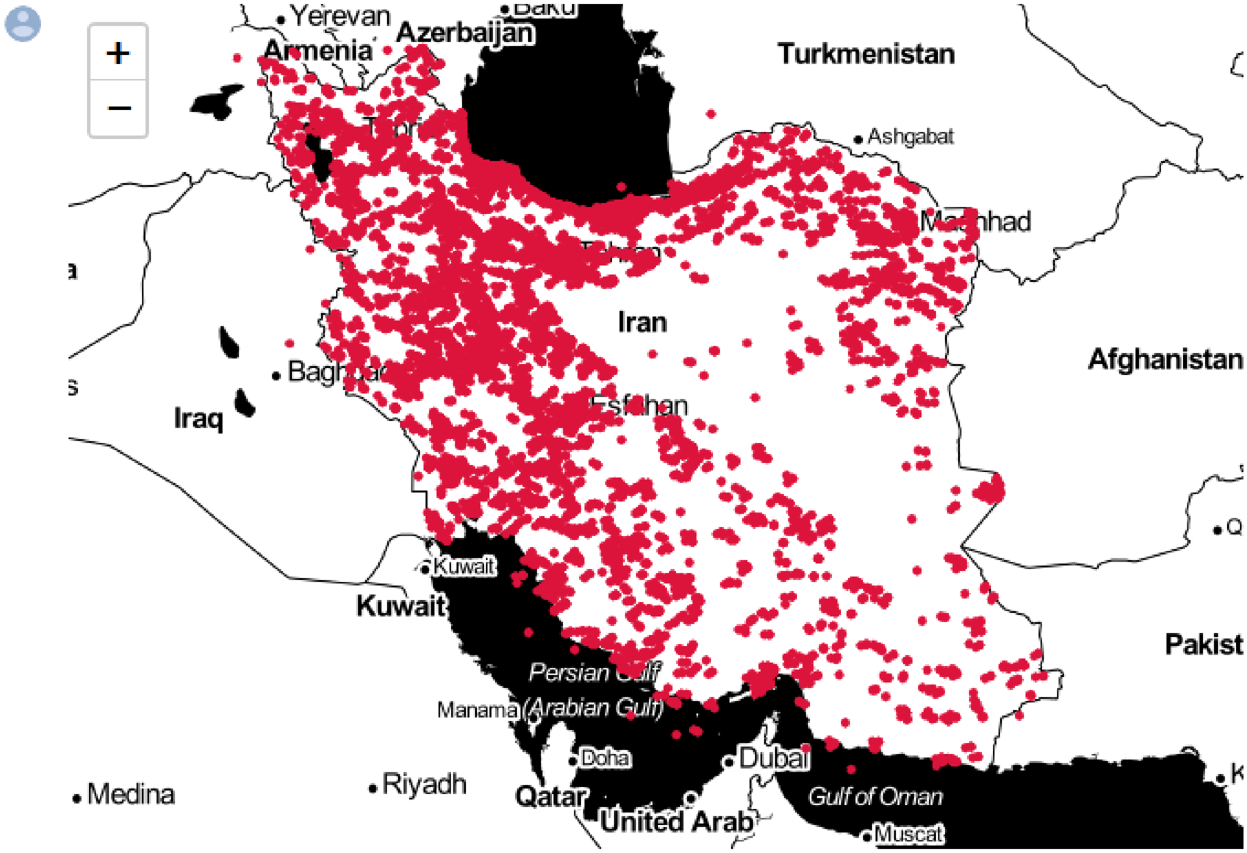
The reason that in Dataset2 we could not cluster properly like Dataset1 is that Dataset2 is Convex data and could not be separated and clustered linearly. For solving the problem in clustering we could use density based algorithms like DBSCAN to cluster properly.

IMPLEMETATION: DBSCAN:

a) The data set with name of "covid.csv" was loaded and with folium, the data set was illusterated on the real map. Each data was showed as a red point on the map:



For better showing the data on the map, map color was taken to white and black demostration so the red data could be illustrated properly:



b) In this part, DBSCAN was loaded from Sklearn library with Esp of 5 and minPts of 5. In this condition, there would be one cluster and there wouldn't be any outlier:

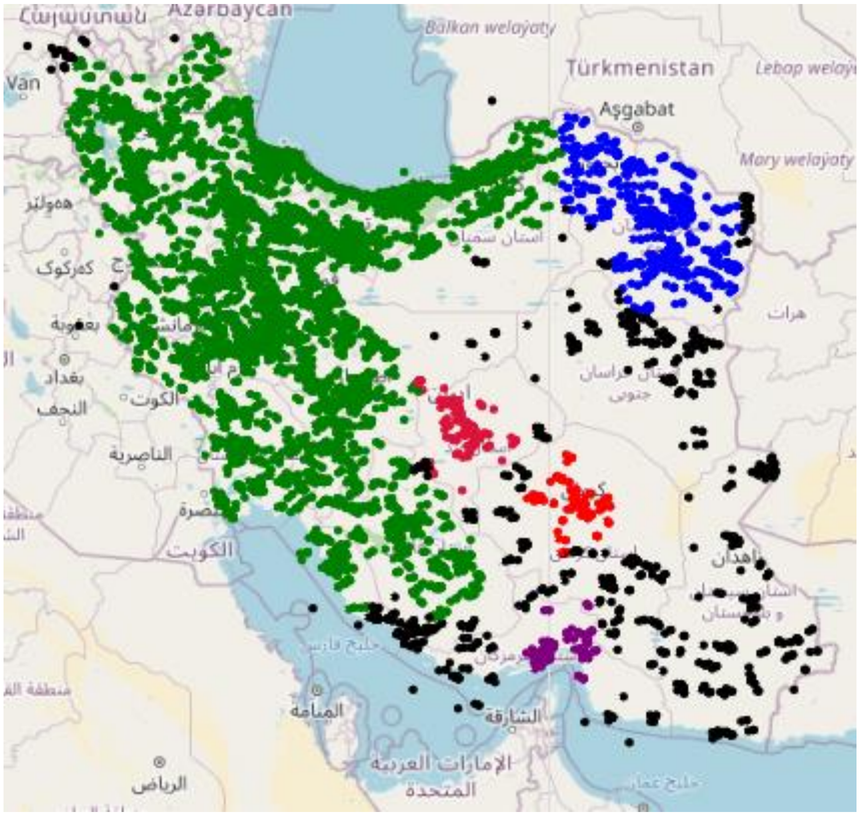
```
Eps is chosen 5 and MinPts is 5
Estimated number of clusters is: 1
Estimated number of outliers is: 0
```

C & D) Eps and MinEps was tuned manually, as the MinEps increased, it meant that there should exist more concentration to be as a center and cluster and when it decreased it meant that there could be lots of clusters as they pass the test of numbers in a radius. Also by increasing eps, the radius of neigbourhood increased so clusters would have join together.

Below are some examle of data with a specified Eps and MinEps, number of clusters, number of oulier and the illustrated data on the map:

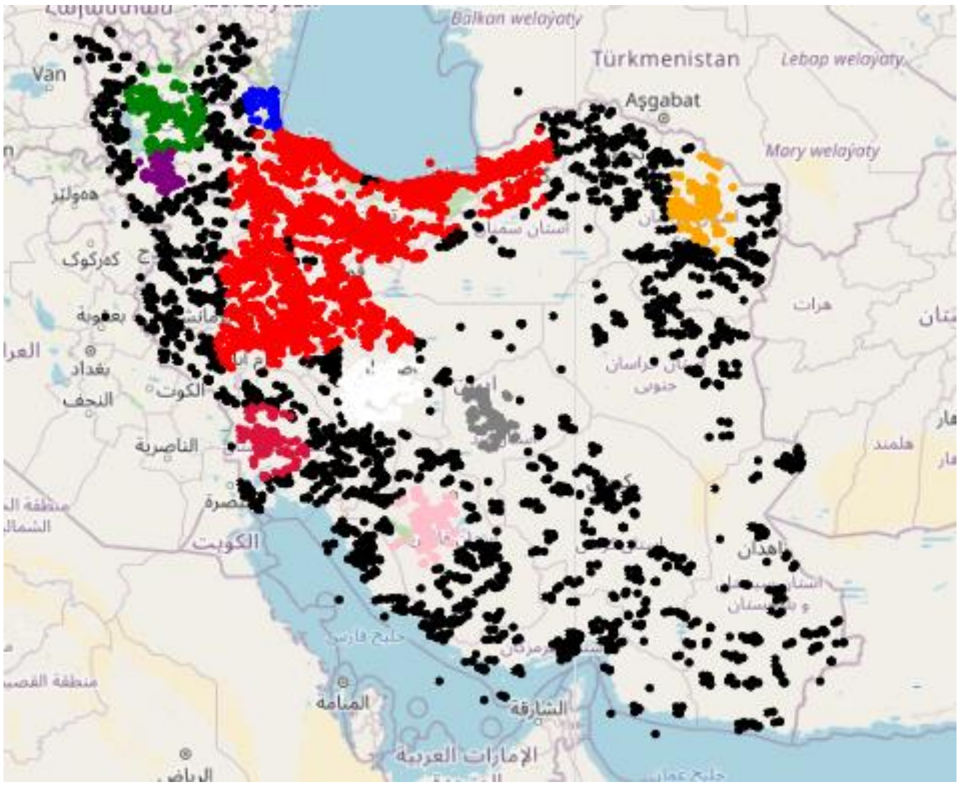
----------------------UNTUNED VERSION---------------------

```
Eps is chosen 0.7 and MinPts is 120
Estimated number of clusters is: 5
Estimated number of outliers is: 1132
```



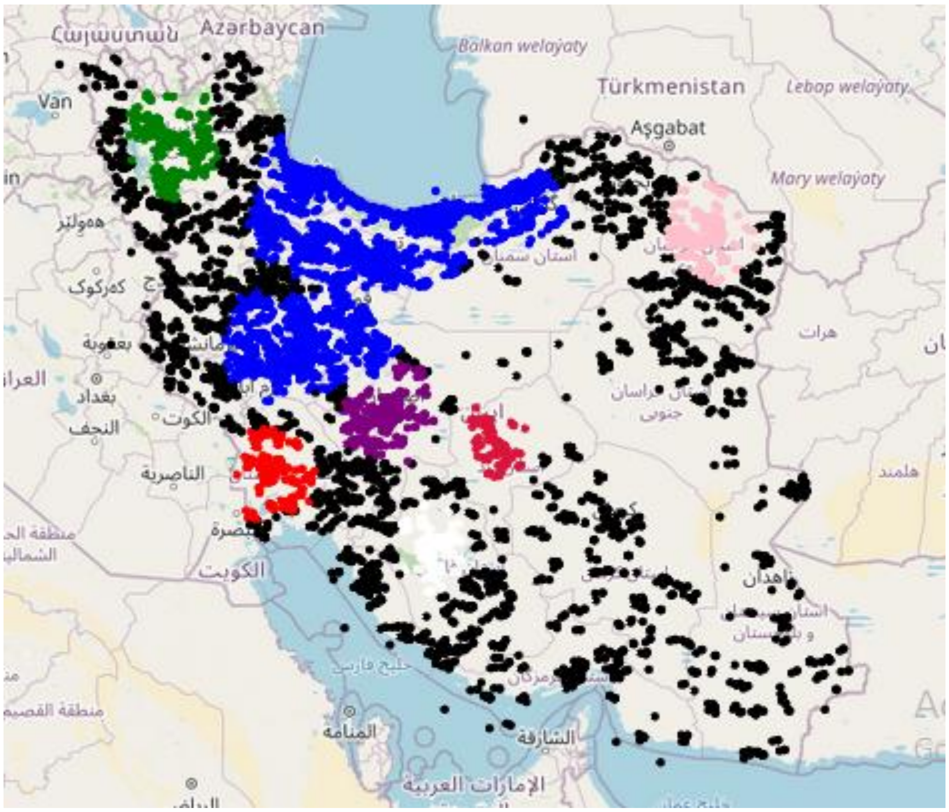
----------------------UNTUNED VERSION---------------------

```
Eps is chosen 0.5 and MinPts is 200
Estimated number of clusters is: 9
Estimated number of outliers is: 5096
```



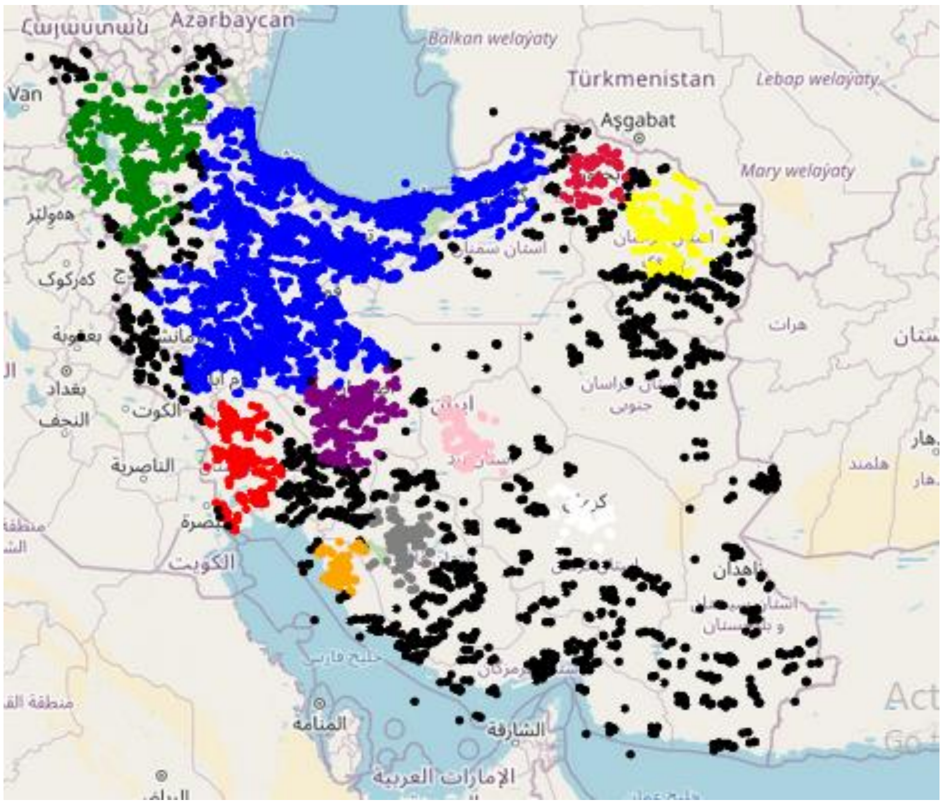
-----------UNTUNED VERSION-----------

```
Eps is chosen 0.6 and MinPts is 300
Estimated number of clusters is: 7
Estimated number of outliers is: 5118
```



--------AND BELOW IS THE TUNED VERSION--------

```
Eps is chosen 0.55 and MinPts is 150
Estimated number of clusters is: 10
Estimated number of outliers is: 2906
```



IMPLEMETATION: Image Compression:

In this implementation, we will compress an image with K-Means. We'll substitute the data with center of the cluster so there will be less colors saved and resolution will be reduced, but as a trade-off, size of the file will be reduced too.

As it is mentioned, firstly K should be chosen as 16. And the compressed pictured and also real picture would be like below:

---------LEFT ONE IS THE COMPRESSED AND THE RIGHT ONE IS ORIGINAL ONE--------

Because the code was so slow to run, I decided to use the K-Means from sklearn library for a better implementation and optimality.

A) K was set to 16 and K-Means algorithm was run in python:

```
################# A ######################
newImg = np.reshape(img, (img.shape[0] * img.shape[1], img.shape[2]))
kmeans = KMeans(n_clusters=16, random_state=10).fit(newImg)
mask = kmeans.labels_
centers = kmeans.cluster_centers_
```

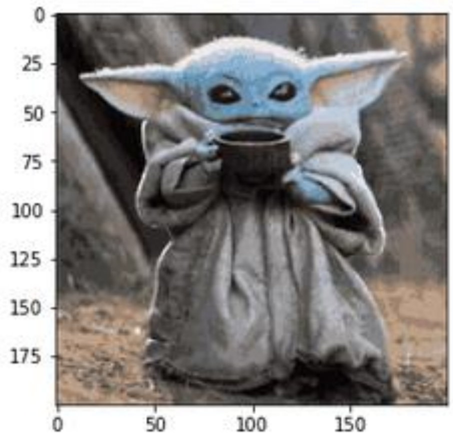
B) Each RGB pixel was replaced by center of the cluster:

```
################# B ########################
for i in range(m):
  newImg[i] = centers[int(mask[i])]
```

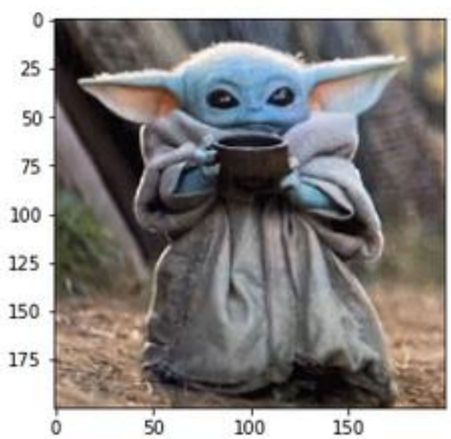
C) K was chosen 256 and A & B was run again

D) For k = 16 and original photo we have:

---------LEFT ONE IS THE CLUSTERED WITH K = 16 AND THE RIGHT ONE IS ORIGINAL ONE--------



---------LEFT ONE IS THE CLUSTERED WITH K = 256 AND THE RIGHT ONE IS ORIGINAL ONE--------



For byte size comparison, OS library was used to compute the byte size of the files and the sizes were:

```
---------------------------------------------------
Size of original small photo is 32.542K bytes
Size of photo with K = 16 is 1.273K bytes
Size of photo with K = 256 is 112.791K bytes
---------------------------------------------------
```