# CMPE 481 ASSIGNMENT 1


## ADALET VEYİS TURGUT - 2017400210


## Table of Contents


**1. Generating a 2D simple dataset suitable for clustering. 2**

**2. k-Means algorithm for the first three iterations and for the last iteration. 3**

2.1. K=4 3

2.2. K=7 5

**3. Objective function vs iteration count plot for all iterations. 7**

3.1 K=4 8

3.2 K=7 8

**4. Comparing with scikit-learn library 9**

4.1. K=4 9

4.2. K=7 10

**5. Finding the best k automatically. 11**

**6. Nonconvex Dataset 13**

6.1 Generating a 2D nonconvex dataset suitable for clustering. 13

6.2 k-Means algorithm for the first three iterations and for the last iteration. 14

6.2.1. K=4 14

6.2.2. K=7 16

6.3. Objective function vs iteration count plot for all iterations. 19

6.3.1 K=4 19

6.3.2 K=7 19

6.4. Comparing with scikit-learn library 20

6.4.1. K=4 20

6.4.2. K=7 20

6.5. Finding the best k automatically. 21

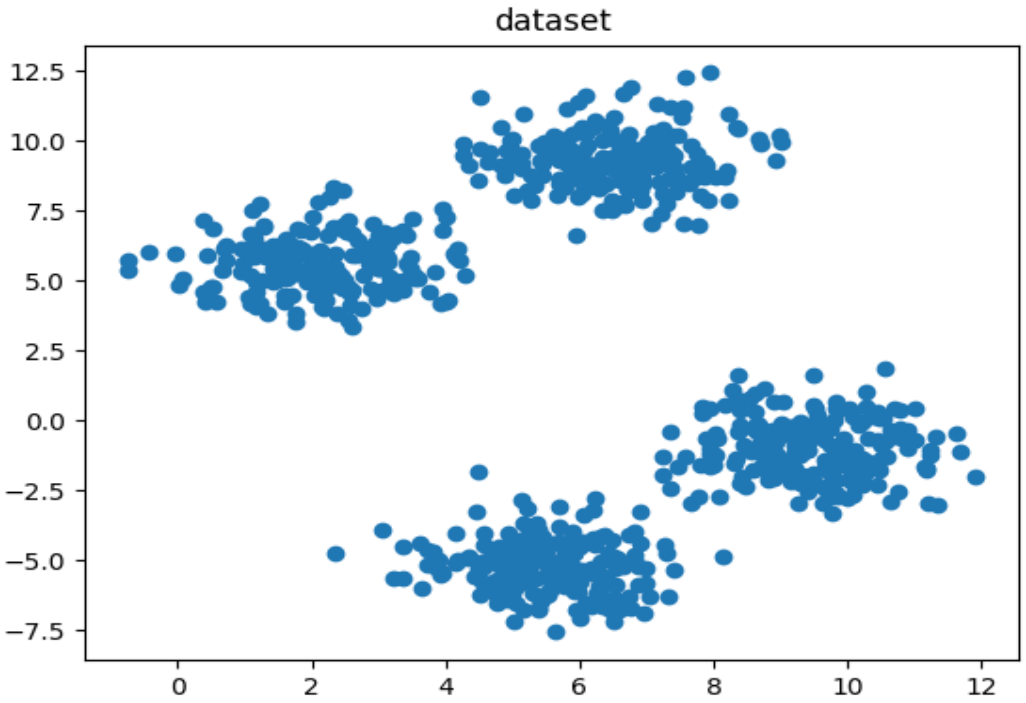
**7. References 21**

**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

# 1. Generating a 2D simple dataset suitable for clustering.

I used the make_blobs() function of the sklearn.dataset library.
Sample size is 800, dimension is 2, number of cluster centers is 4, cluster_std is 1 and seed is 13.
Corresponding code is:

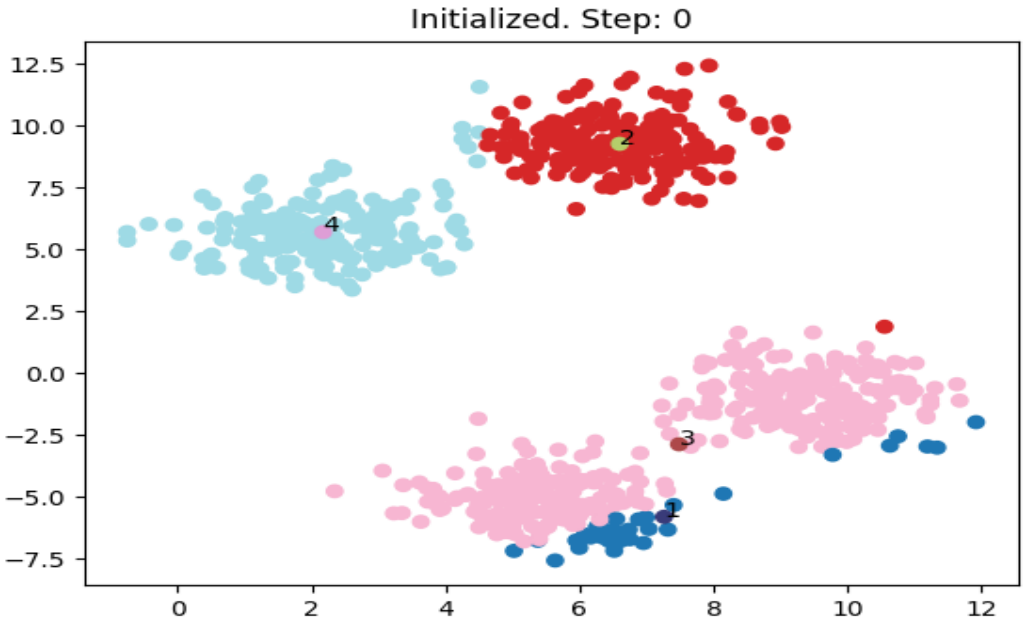
```
dataset, label = dt.make_blobs(n_samples=size, n_features=2,
centers=cluster_count, cluster_std=1, random_state=seed)
```

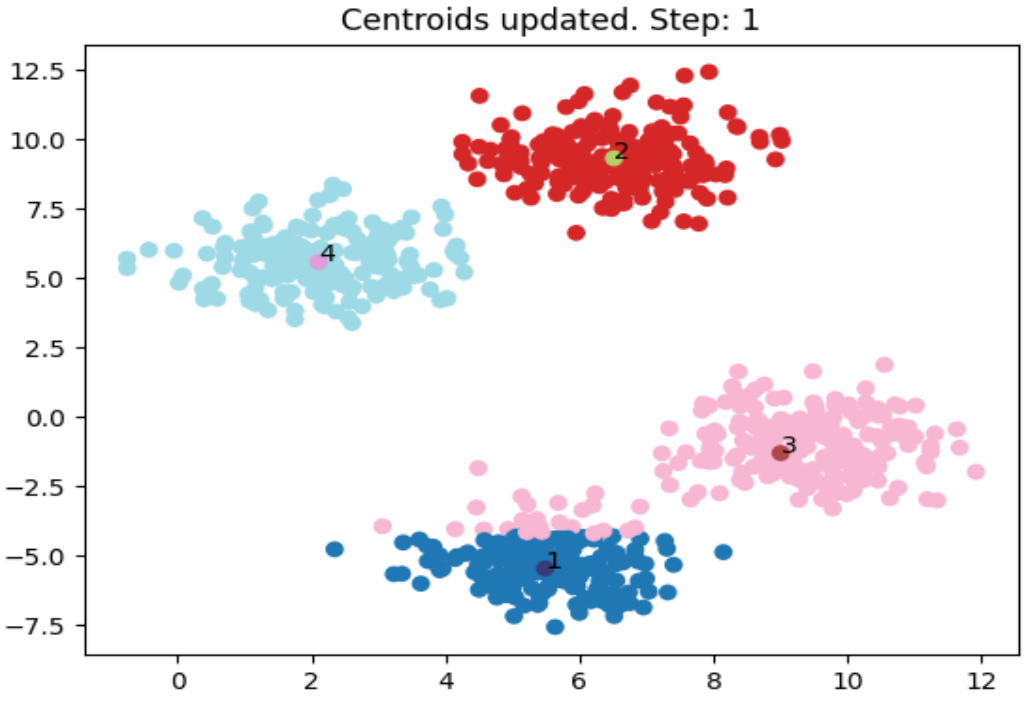


**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

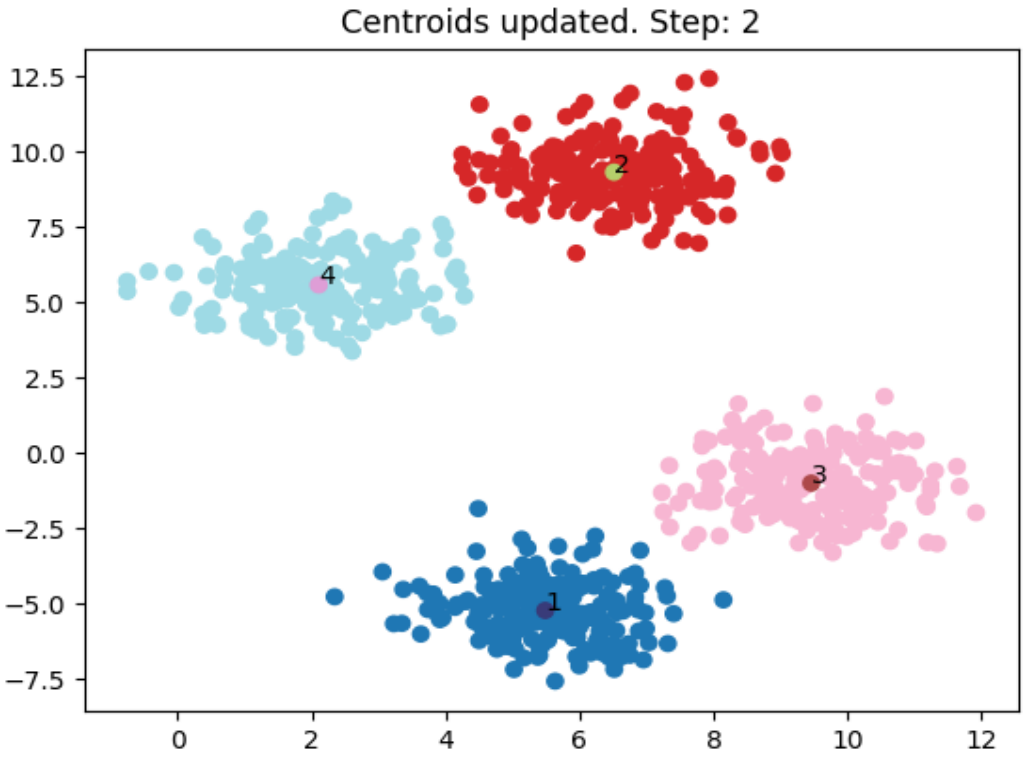
# 2. k-Means algorithm for the first three iterations and for the last iteration.

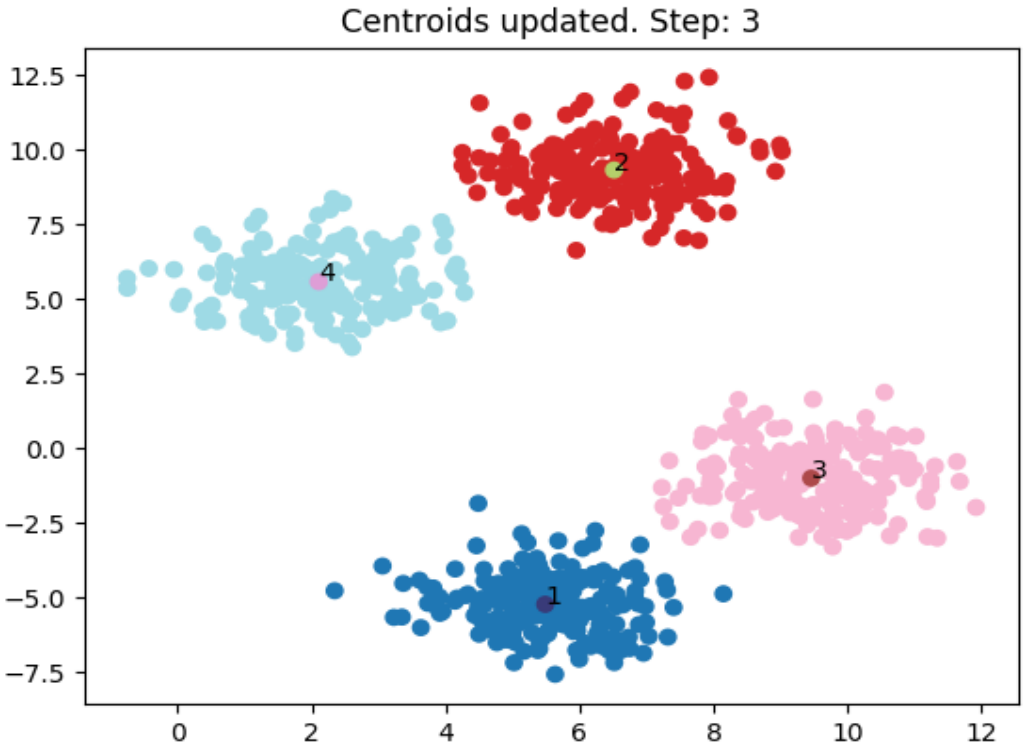
## 2.1. K=4



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

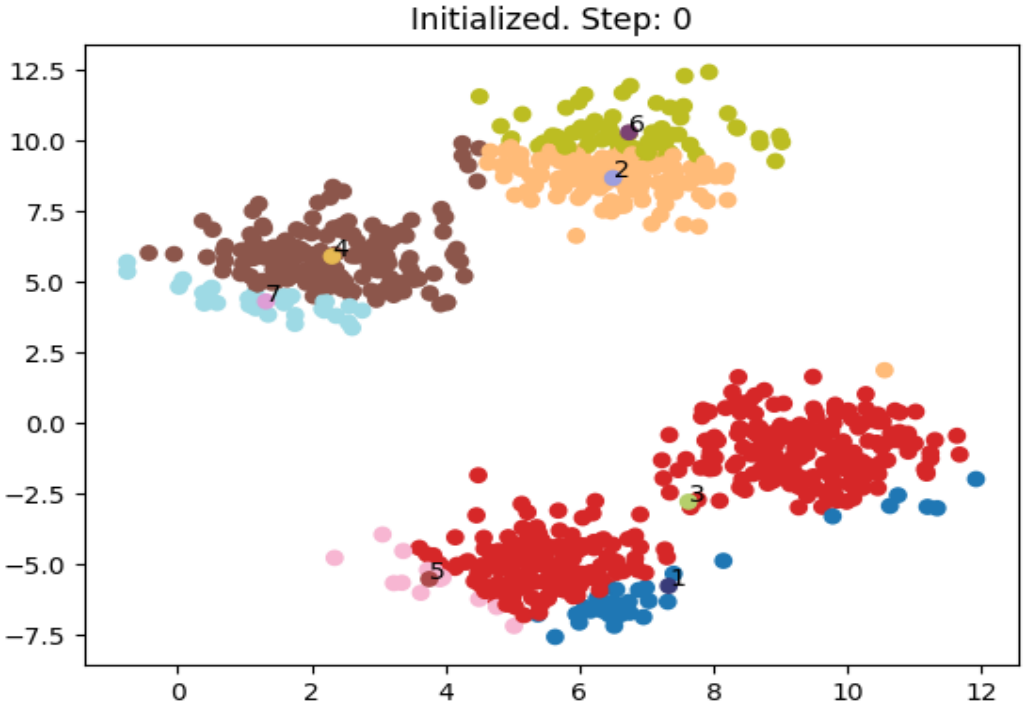


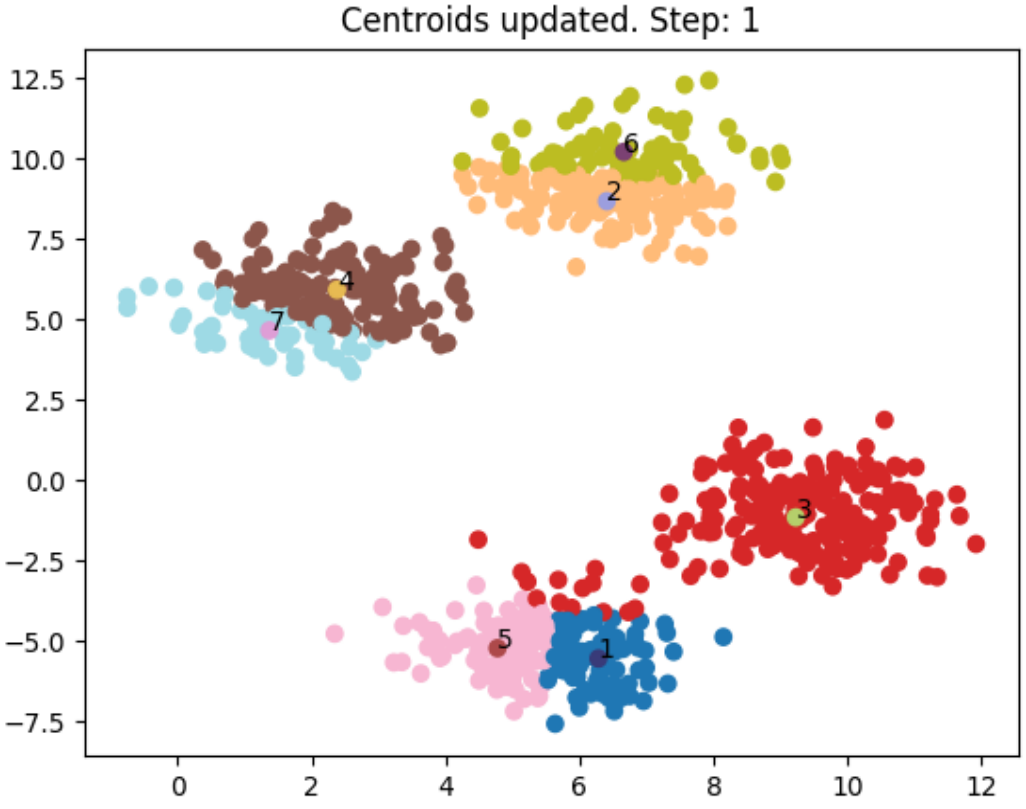




**This is also the final iteration.

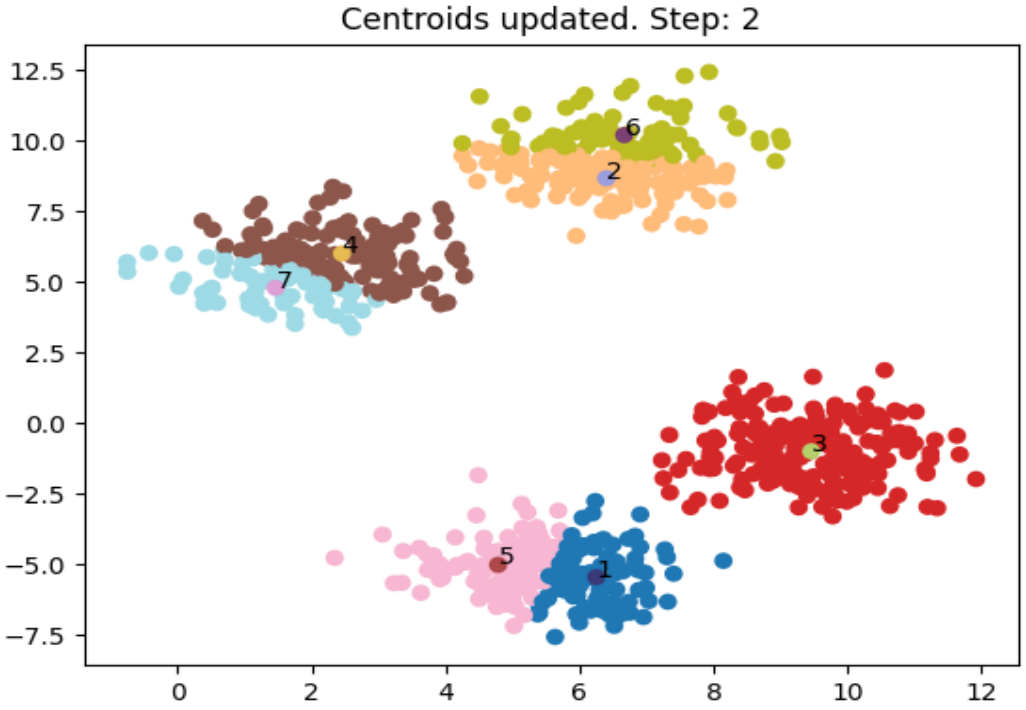
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

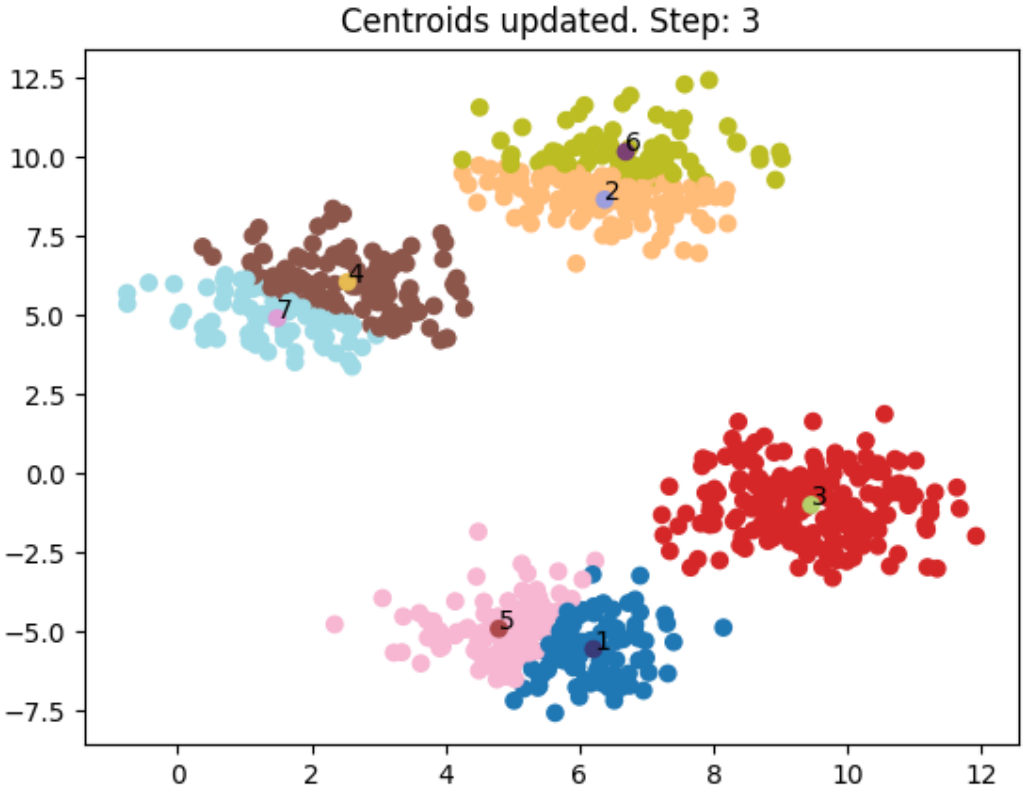
## 2.2. K=7



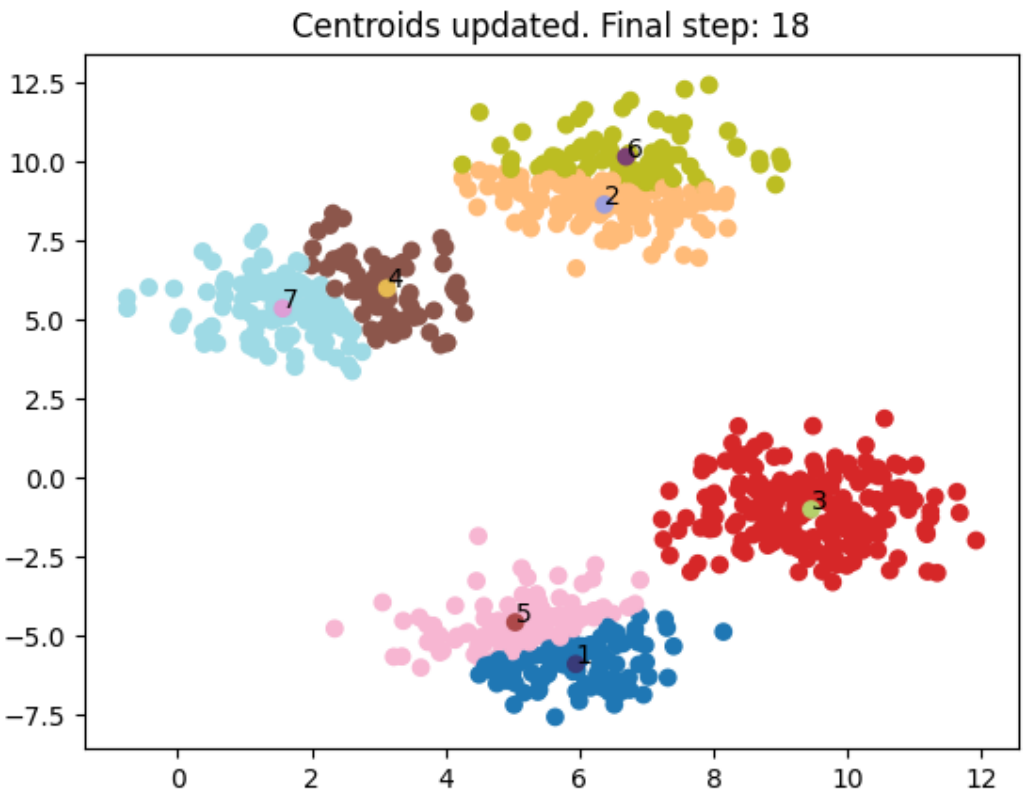



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

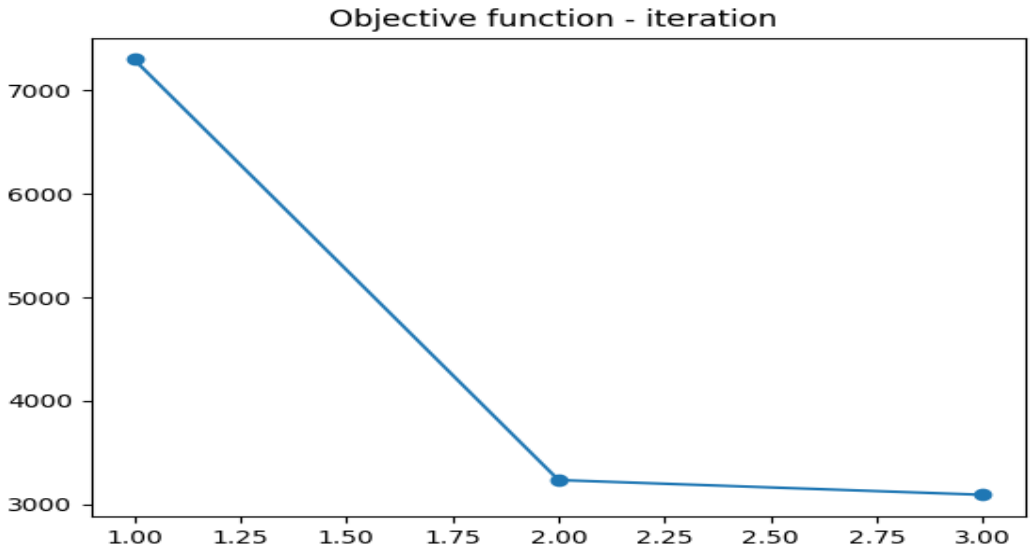


**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

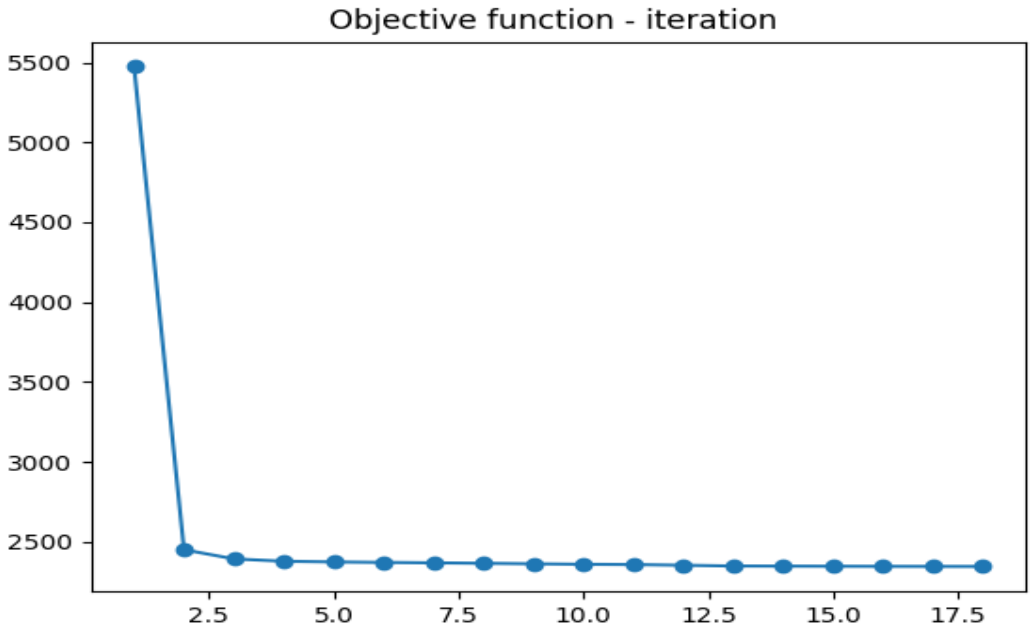
# 3. Objective function vs iteration count plot for all iterations.

It can be seen that objective function is monotonically decreasing.

## 3.1 K=4




## 3.2 K=7

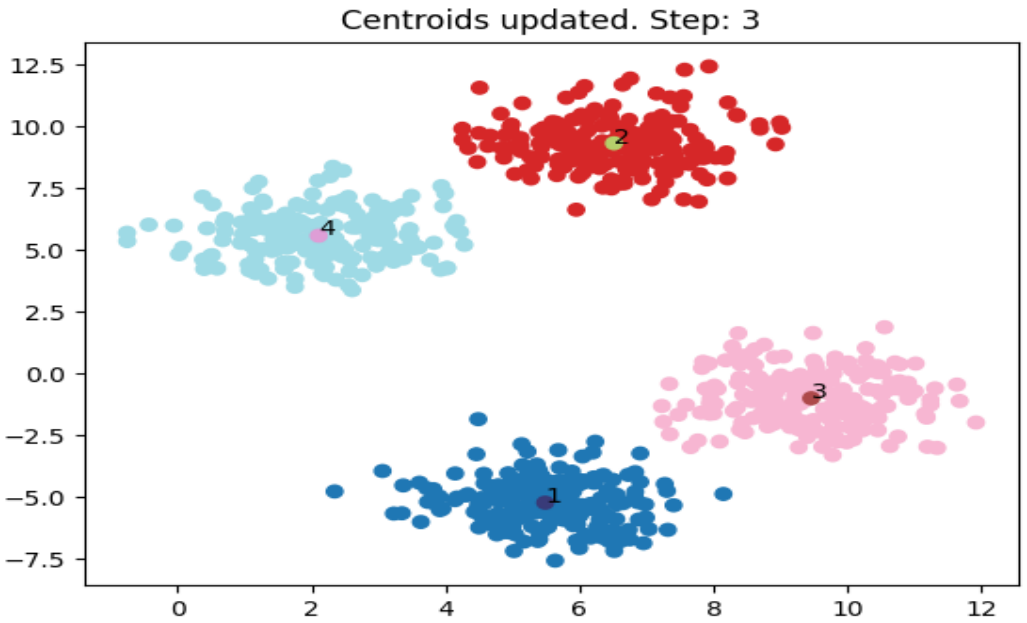


**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

# 4. Comparing with scikit-learn library

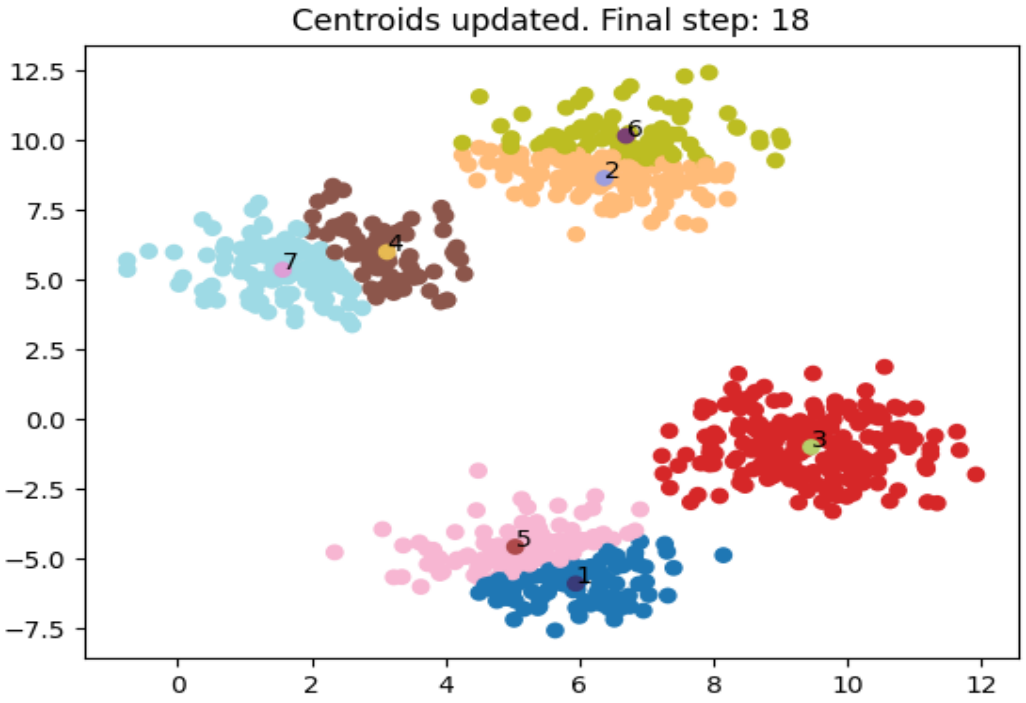
When K=4, results are the same. When K=7, results differ greatly.
When K is larger than the number of natural clusters in the dataset, the algorithm divides existing natural clusters. Since I initialized centroids randomly, scikit-learn and I divide different natural clusters in order to have 7 clusters.

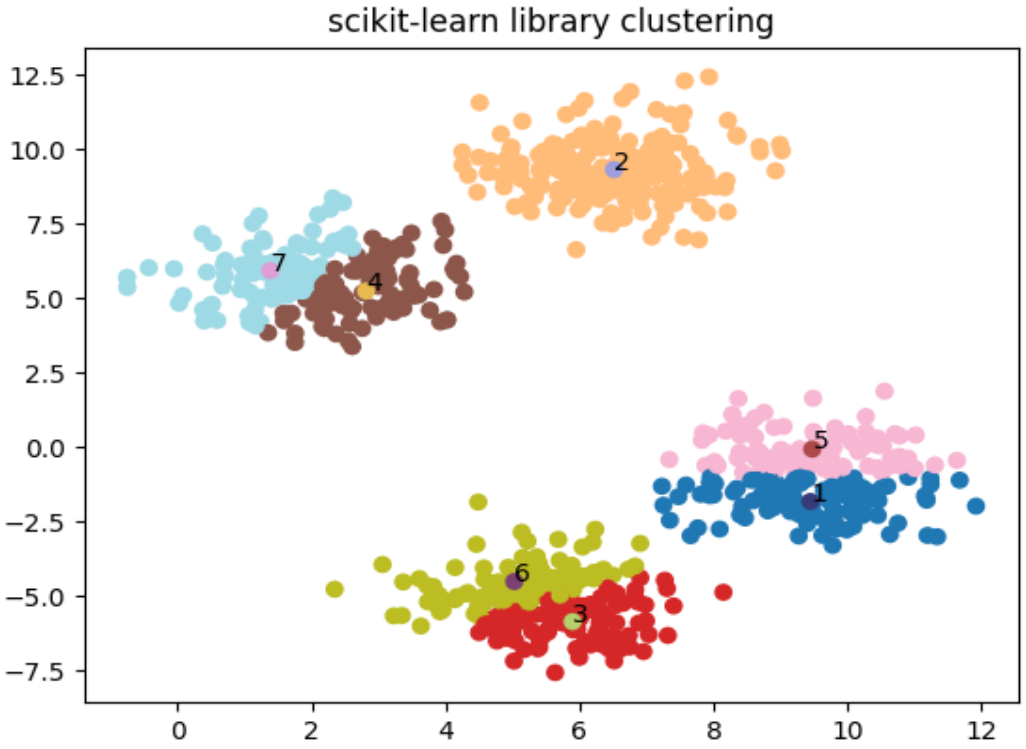
## 4.1. K=4

**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

## 4.2. K=7



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

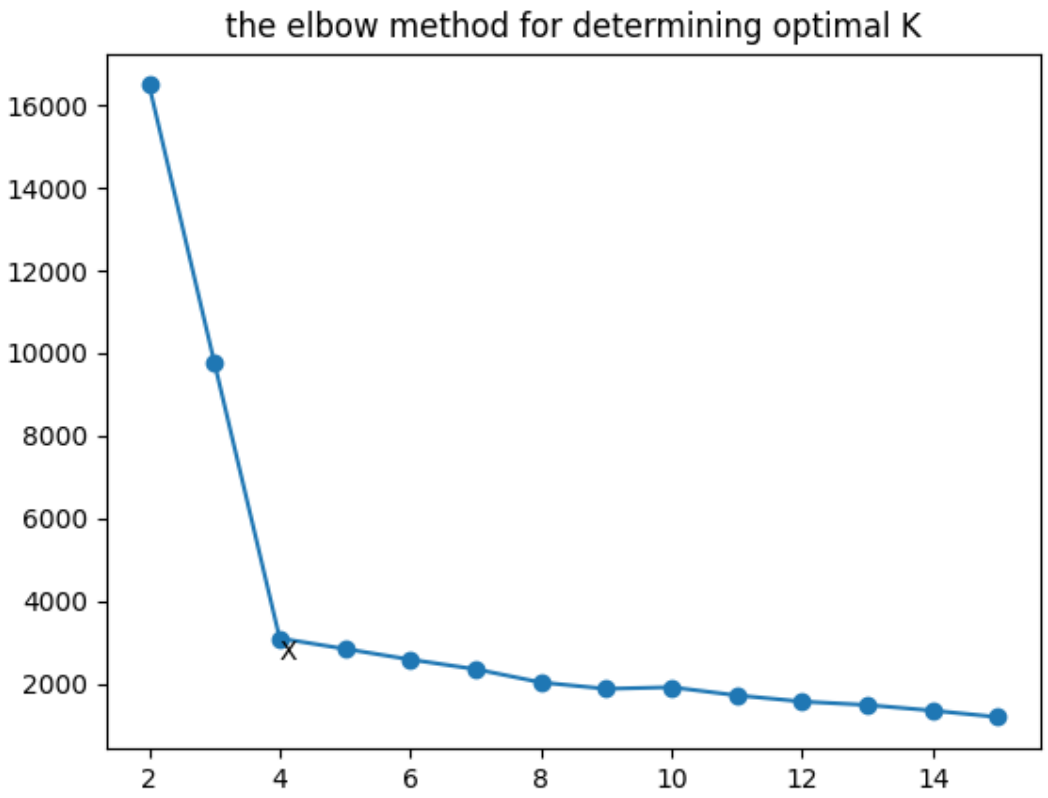
# 5. Finding the best k automatically.

I researched about this subject and found two main methods: Elbow and Silhouette. I immediately figured out that in the spectral clustering lecture you hinted at the elbow method. So, I firstly implemented the Elbow method.

In the Elbow method, we plot objective function vs K and find the corner point where the plot is getting linear. Since I was calculating objective function values at each iteration already, it was easy to integrate this method.
Main idea of this methodology is calculating the within cluster sum of squared errors and finding the optimal number of clusters where adding one more cluster does not worth it --in other words the extra information we gained does not worth the effort.
Picking this point with bare eyes is very simple, but automating it may be troublesome. I came up with an idea that takes the ratio of previous slope to next slope for each K value and takes the K which makes that ratio maximum. I think this approach detects the sudden linearity in the plot. But there is a drawback: if objective function peaks once again like point 9 in section 6.5., the result may not be correct. For example if objective value were 10000 for the K=10 in the below figure --which may occur if 9 centroids are very close to each other and span one natural cluster, and one centroid span 9 natural cluster, again which may occur due to random initialization-- elbow method could have find K=11 as optimal.

To avoid this drawback, I implemented the Silhouette method too. Main idea of this method is measuring how well a data point is assigned to its cluster compared to other clusters. We first find the sum of the distances between a point and other points within its cluster, then we find the closest neighbor cluster which has the minimum sum of distances between this point and those cluster's points. 1 - ratio of these values gives us the "s" value for that point. Mean of these "s" points for the dataset gives us the silhouette score. The result is always between -1 and 1. Highest s value means the optimal number of clusters.
Unfortunately, I could neither correctly implement this method nor found the source of the error. S value is increasing with K in my implementation. Also, it was slowing down the program since it has a high complexity. So, I commented out this method.

**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**



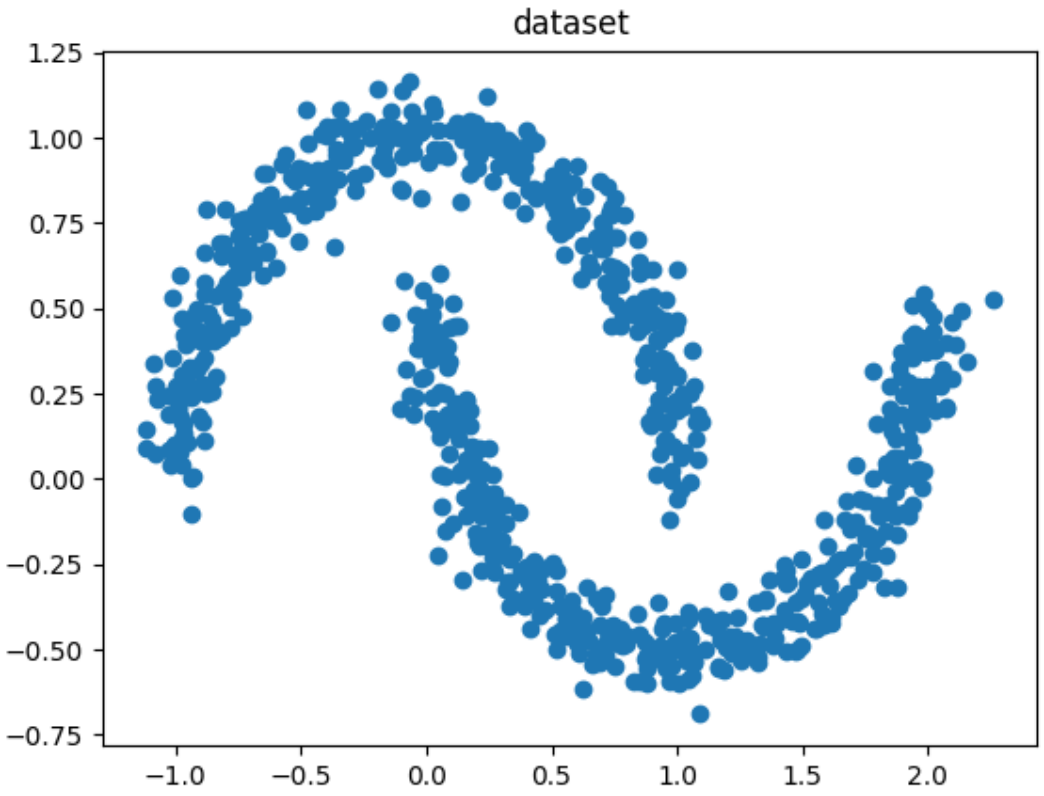
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

# 6. Nonconvex Dataset

## 6.1 Generating a 2D nonconvex dataset suitable for clustering.

I used the make_moons() function of the sklearn.dataset library.
Sample size is 800, dimension is 2, noise is 0.07.
Corresponding code is:

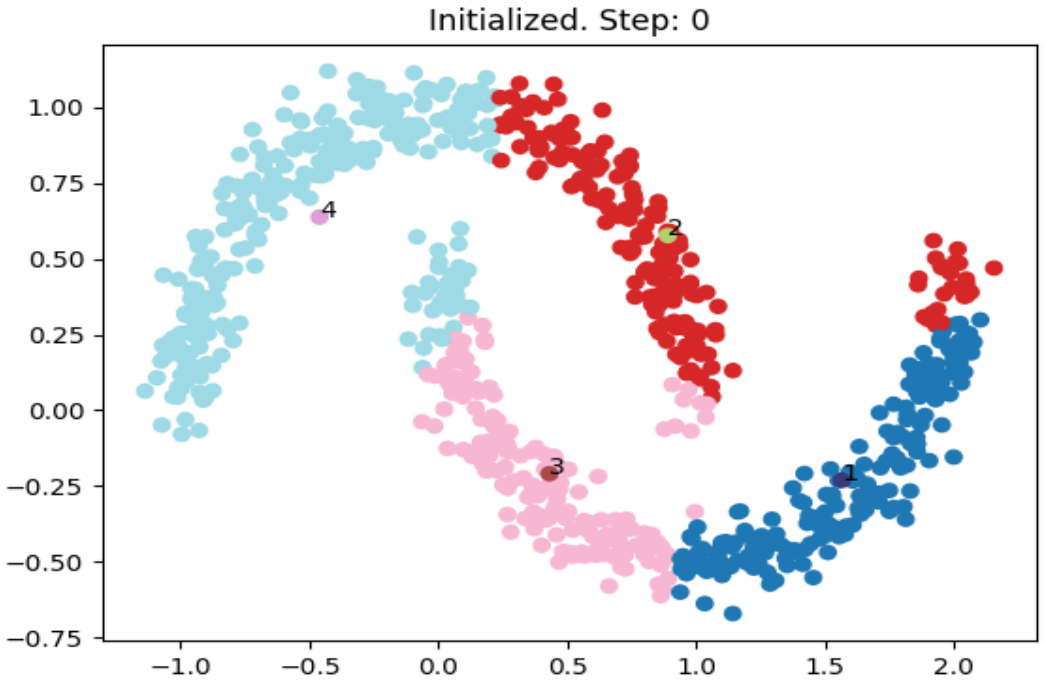
```
dataset, label = dt.make_moons(n_samples=size, noise=0.07)
```

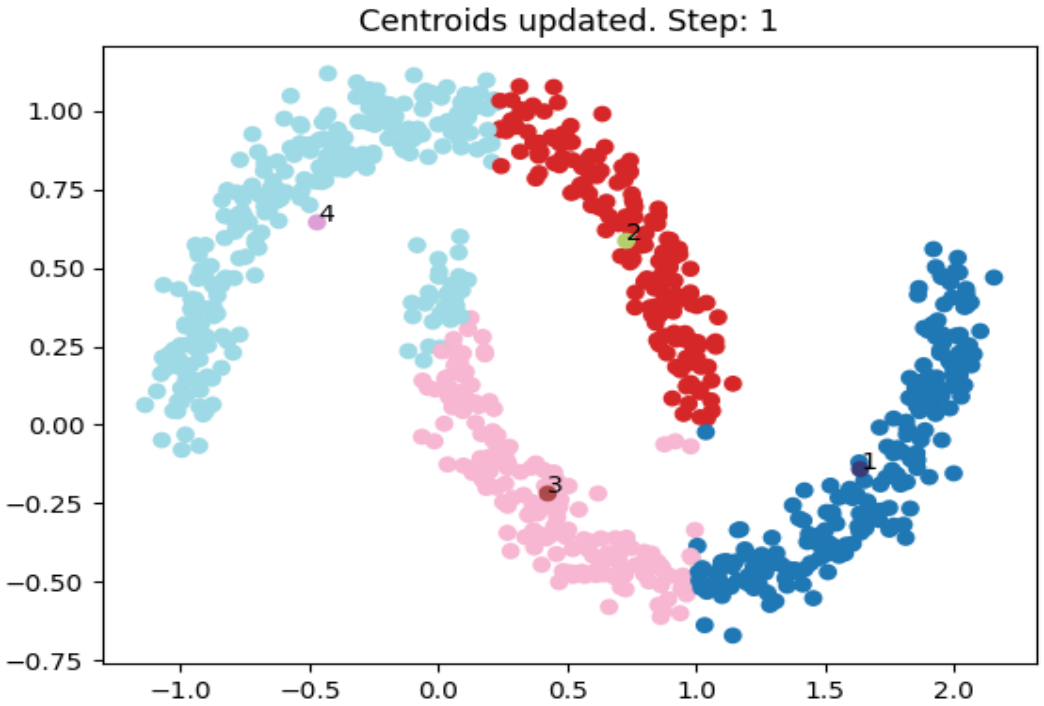


**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

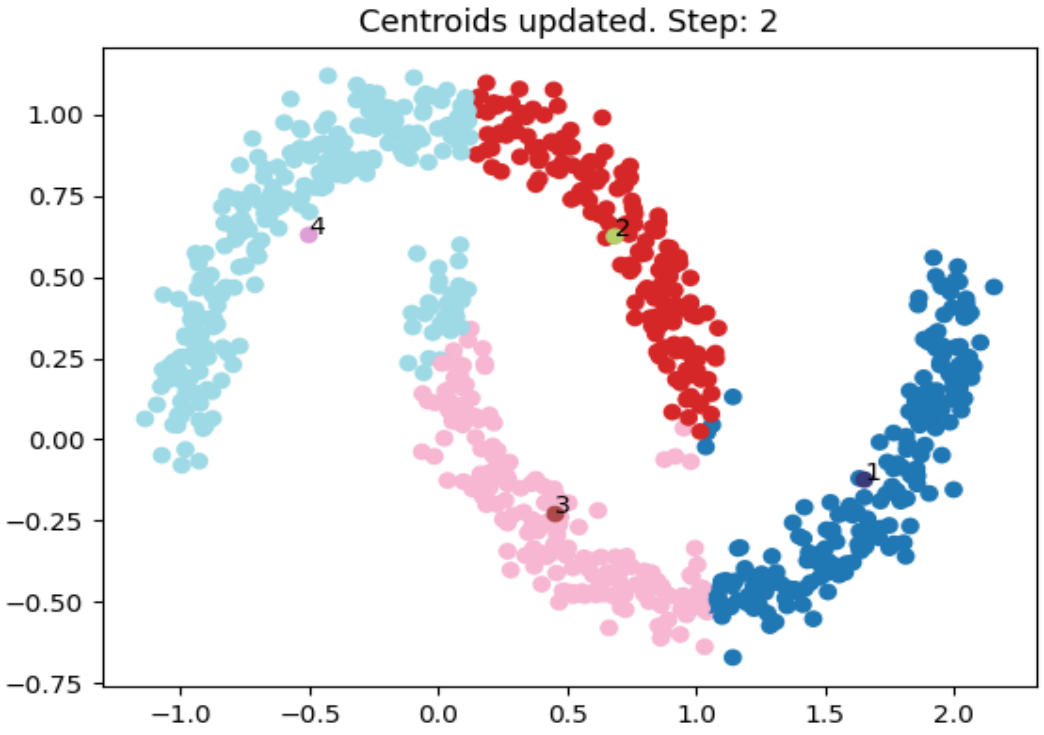
## 6.2 k-Means algorithm for the first three iterations and for the last iteration.

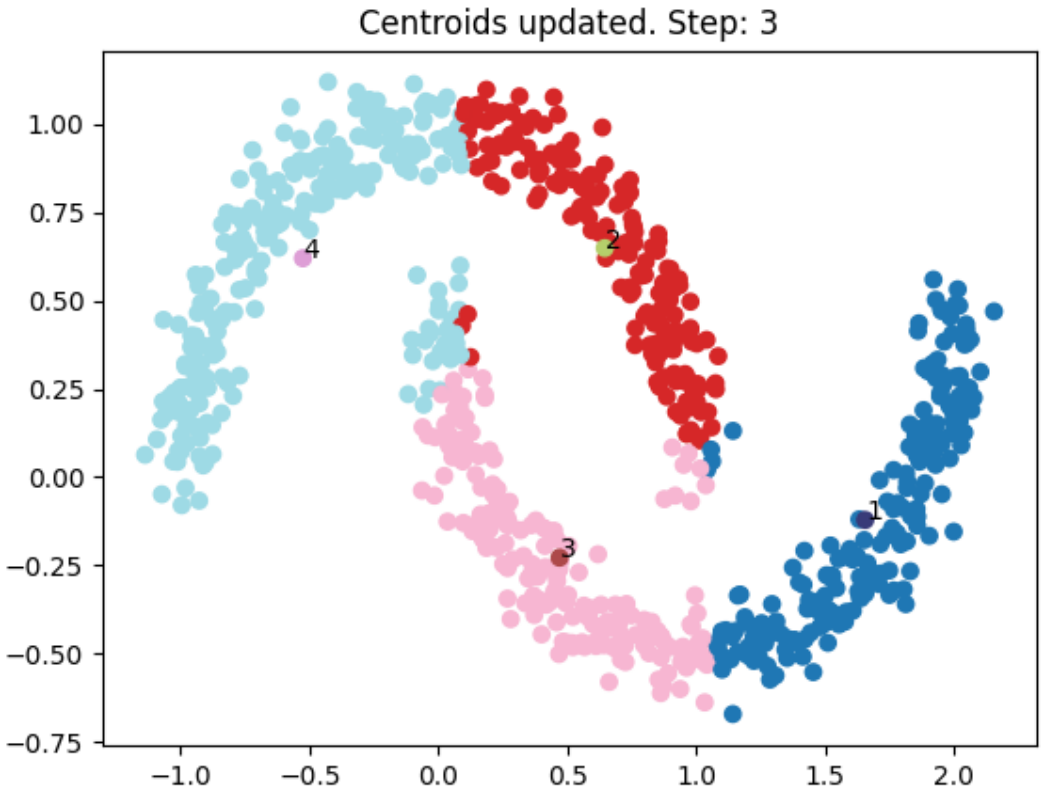
### 6.2.1. K=4



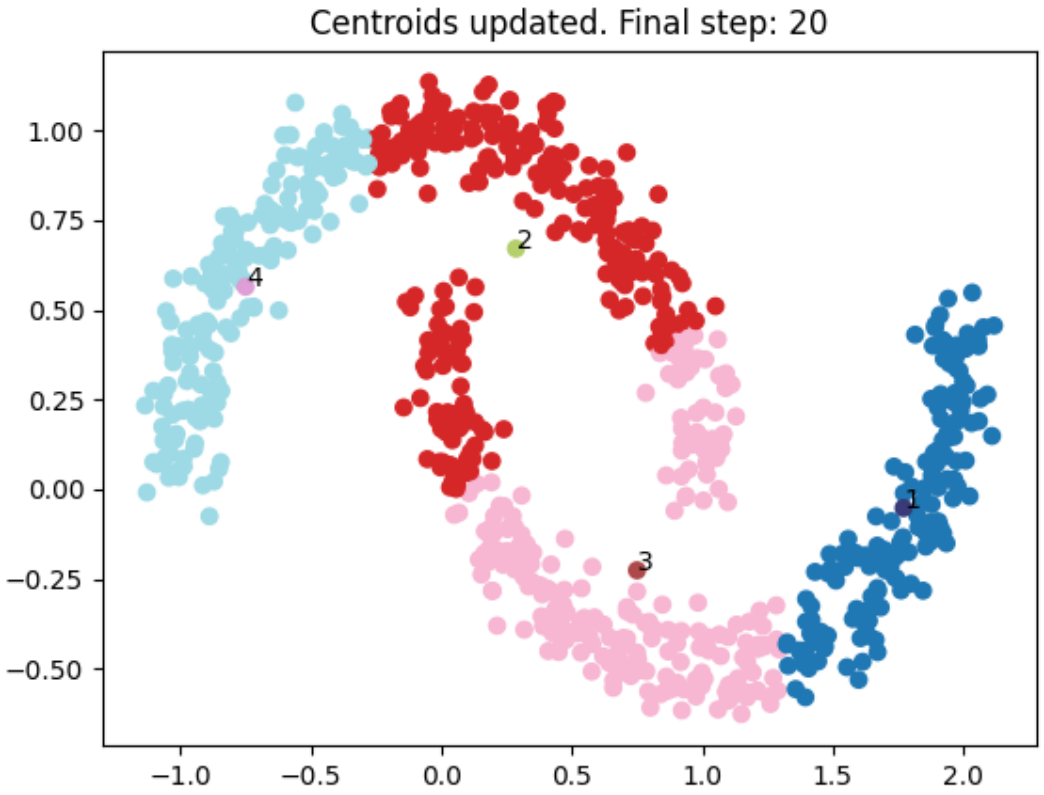



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

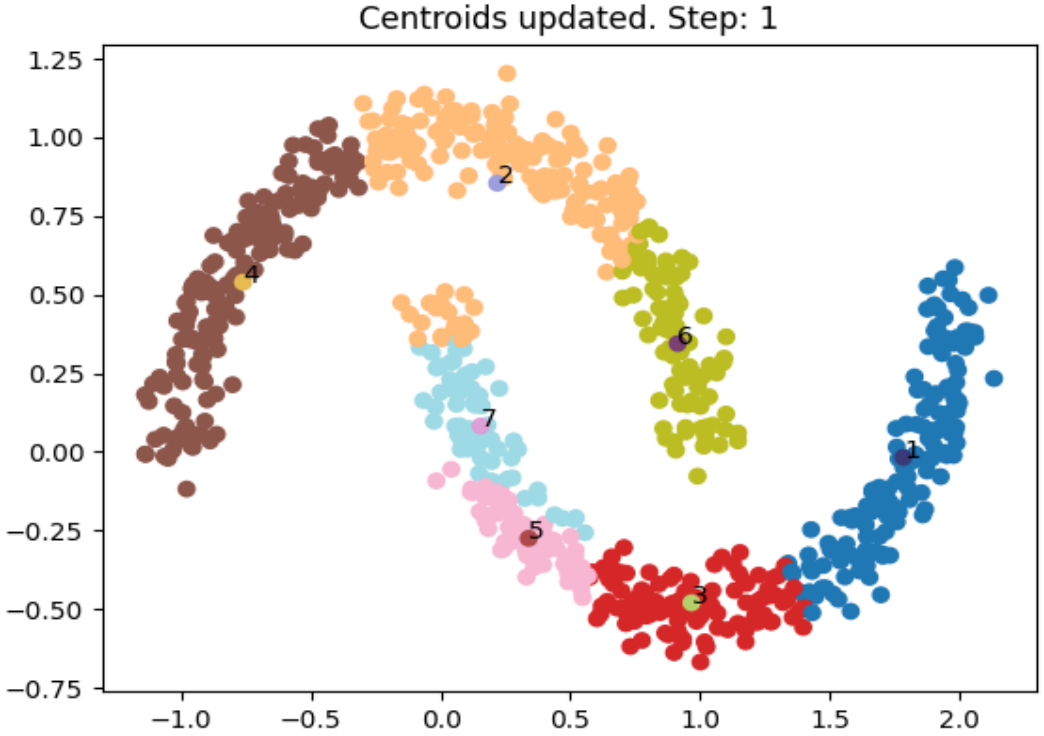


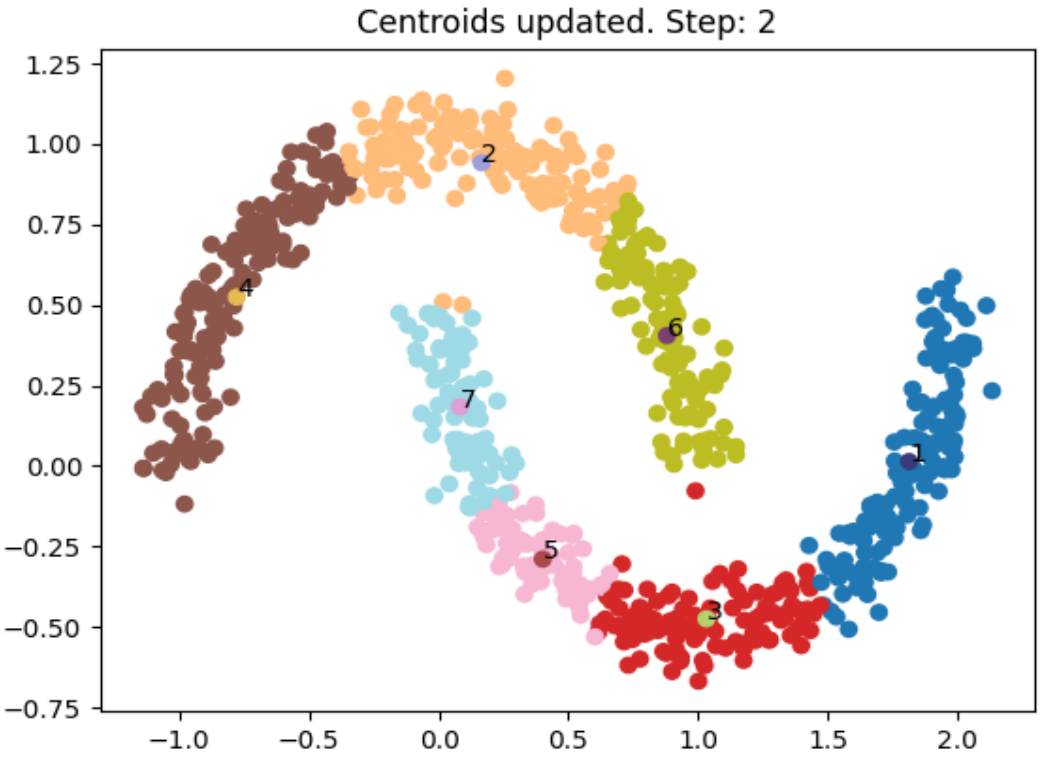
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

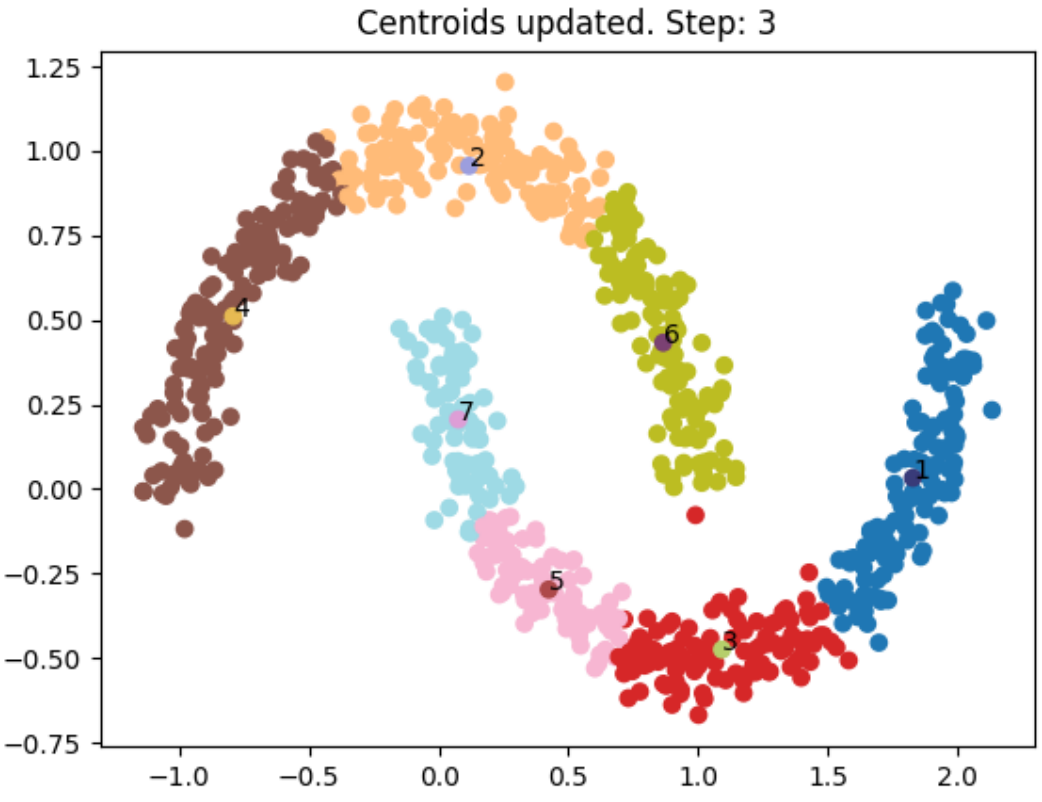



### 6.2.2. K=7

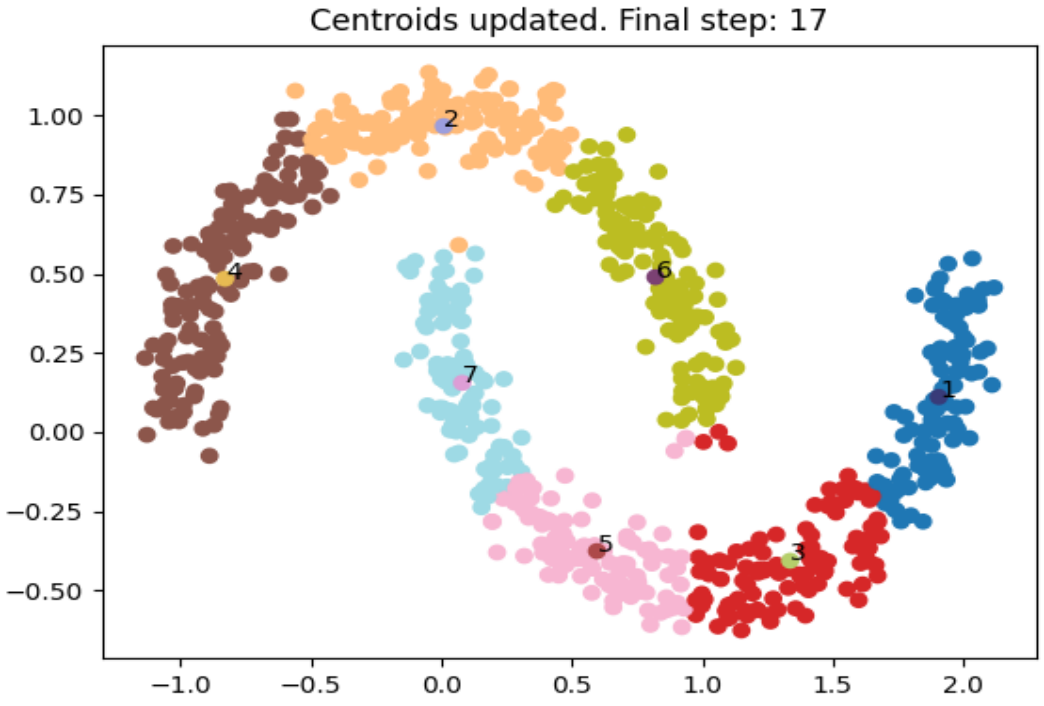


**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**



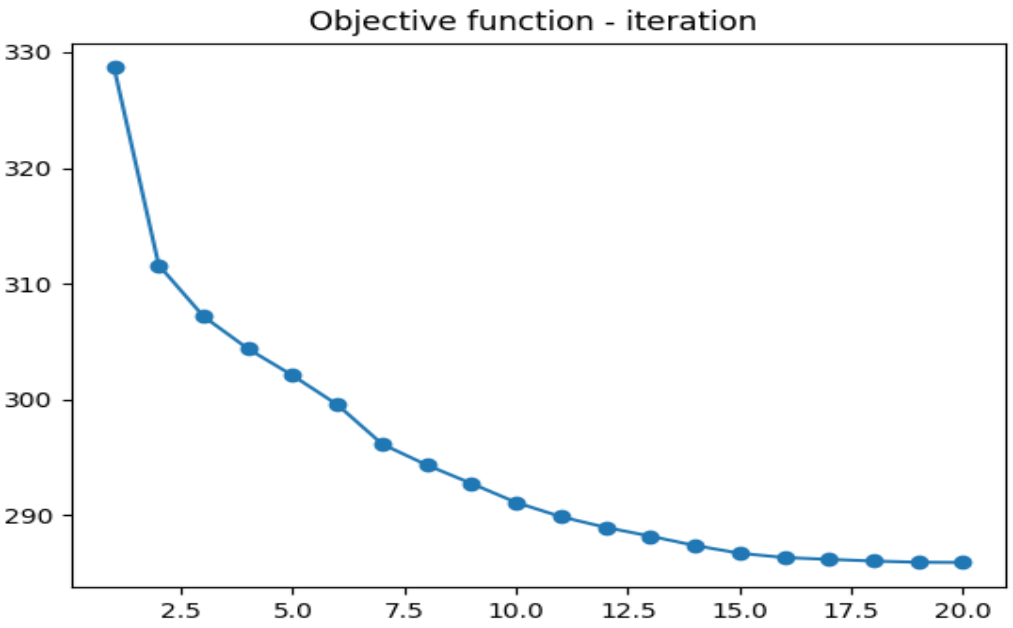
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**



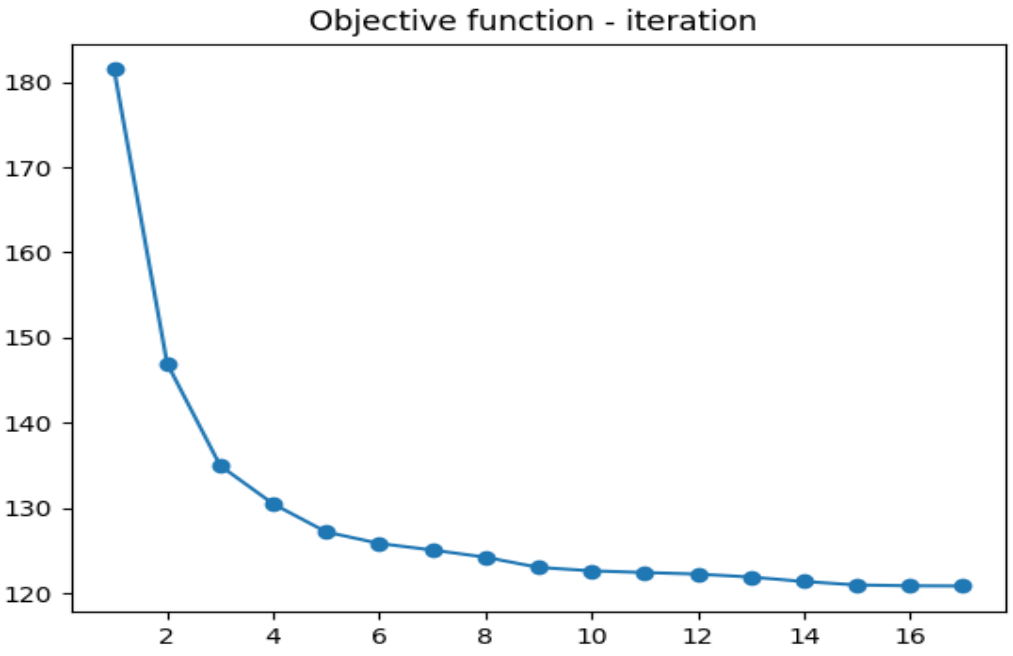
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

## 6.3. Objective function vs iteration count plot for all iterations.

### 6.3.1 K=4



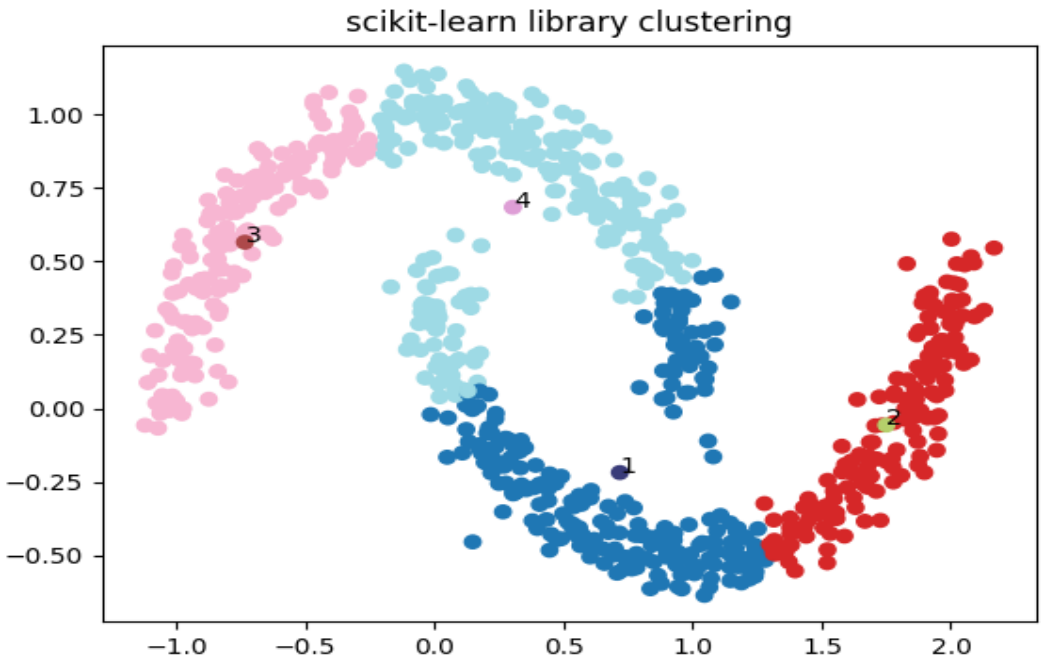

### 6.3.2 K=7



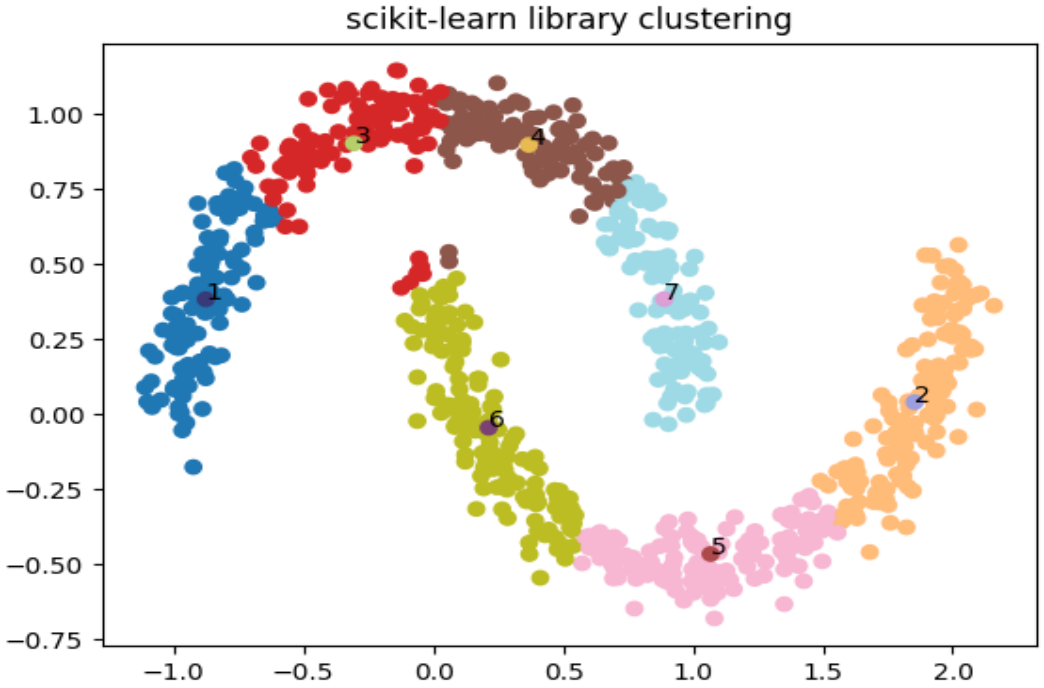
**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

## 6.4. Comparing with scikit-learn library

### 6.4.1. K=4



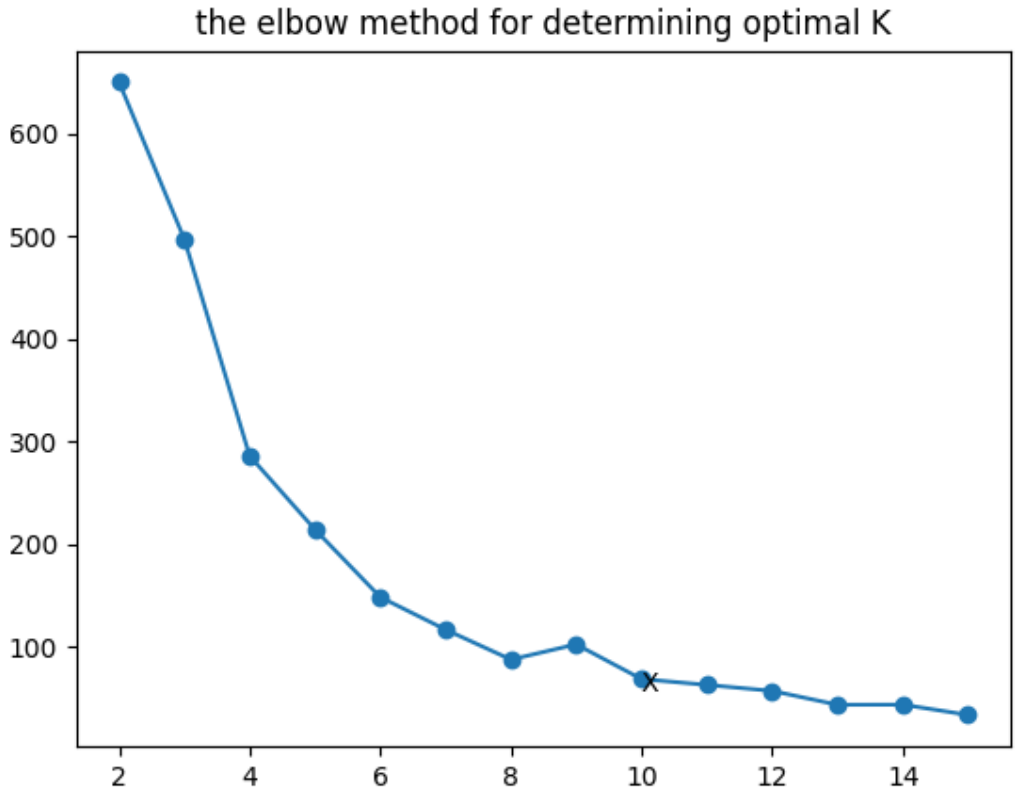

### 6.4.2. K=7



**Adalet Veyis Turgut**
**2017400210**
**21.01.2021**

### 6.5. Finding the best k automatically.

Since this is a nonconvex dataset, the elbow method does not give a satisfying result.




# 7. References

1. https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb
2. https://en.wikipedia.org/wiki/Silhouette_(clustering)
3. https://en.wikipedia.org/wiki/Elbow_method_(clustering)