Project Part 2
Unsupervised Learning (K-means)
CSE 575: Statistical Machine Learning

Submitted By:
Mit Patel (1217114054)

Introduction

In this project, I have used 300 two dimensional data points to apply K-means algorithms on them. K-means algorithm is an Unsupervised Machine Learning algorithm in which there is no label provided or available for input training data set. What K-means does is that it clusters all the data points into different groups to get an idea about how our dataset is scattered and what data points are related to each other.

For the sake of understanding the working process of K-means inside out, I have performed it without using any built-in libraries of Python. I generated it from scratch. I just have used Numpy and Matplotlib libraries to work with data and visualize them.

Unsupervised Learning

According to wiki, Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. Two of the main methods used in unsupervised learning are Principal Component and Cluster Analysis. In this project, I have implemented the Cluster Analysis using K-means algorithm. Cluster Analysis is a branch of Machine Learning that groups data that has not been labelled, classified or categorized. Instead of responding to feedback, cluster analysis identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. This approach helps detect anomalous data points that do not fit into either group.

K-means Algorithm

K-means clustering is popular for cluster analysis in the field of Machine Learning. The way it works is like it partitions data into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. K-means minimizes intra-cluster distance and maximizes inter-cluster distance.

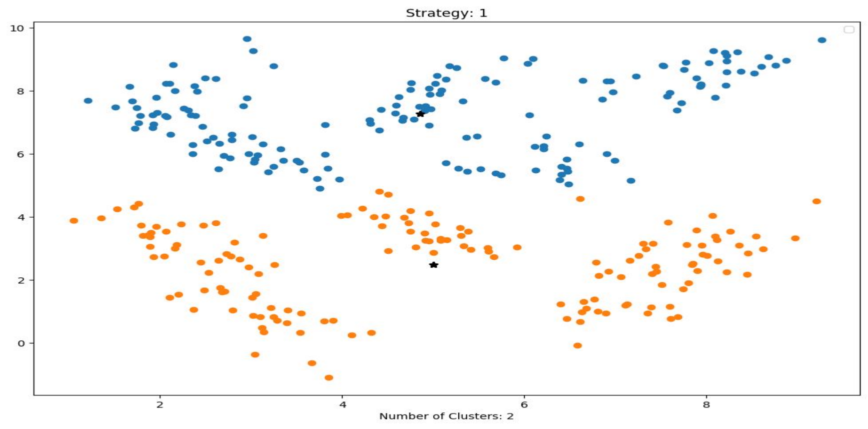
In this project, I have implemented two strategies for K-means algorithms. In the first strategy, the initial points of all the centroids are generated randomly. To generate the K random points (centroids) out of all data samples, I have used Python’s built-in library Numpy’s function: ‘numpy.random.choice()’. In the second strategy, only the first centroid out of K centroids is generated randomly. All other K-1 centroids are generated using specific technique which is described in the following paragraphs. The ‘strategy’ variable is passed while generating the object of K-means class.

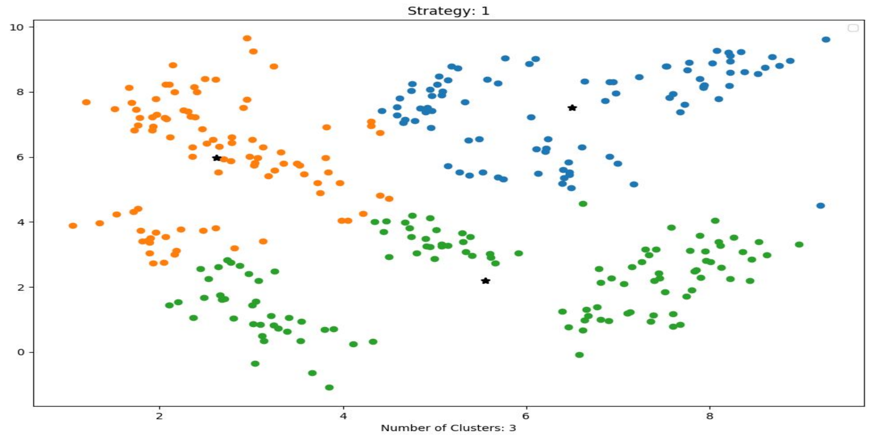
- *Strategy 1:*

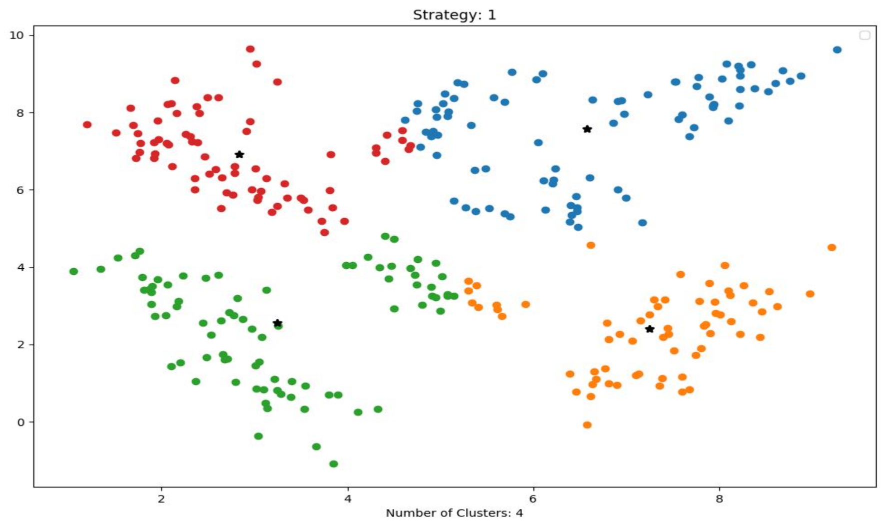
  In this part, all the initial centroids have been generated randomly. These points are some ‘K’ points from all the data samples. First, I have generated these ‘K’ centroids using Python’s built-in library. Then, all the data points are clustered into ‘K’ different clusters using these ‘K’ centroids as a mean for them. I have used Euclidean Distance as a measure for creating these clusters. All the data points are clustered with a particular centroid if their Euclidean Distance are nearer to that centroid than any other cnetroid.

  After this, I calculate the mean of all these clusters and make them as new centroids. Then again the above procedure is repeated until all the clusters are converged. And then finally, the cluster number for all these data samples is returned as an array.

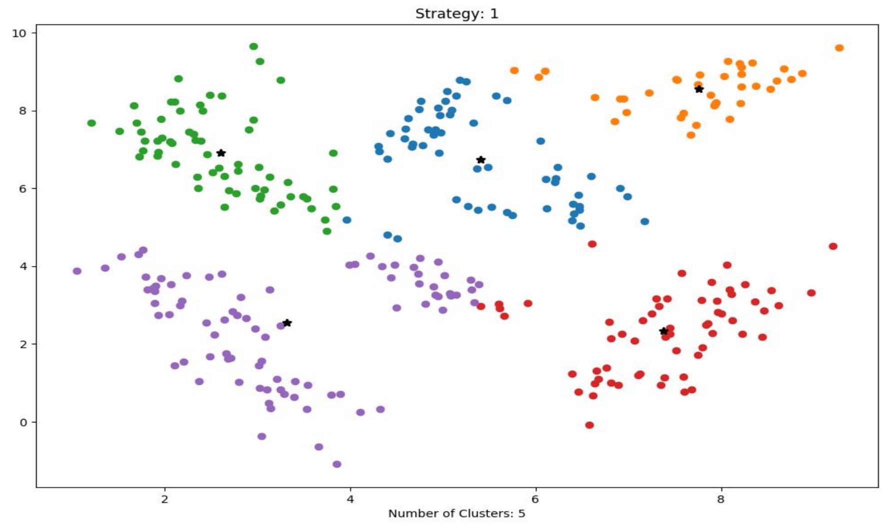
  I have repeated the above procedure for the ‘K’ value of 2 to 10. After every procedure for the particular ‘K’, I have gotten the following different clusters for the given data samples after convergence. Here, the “Black Star” represents the different centroids.



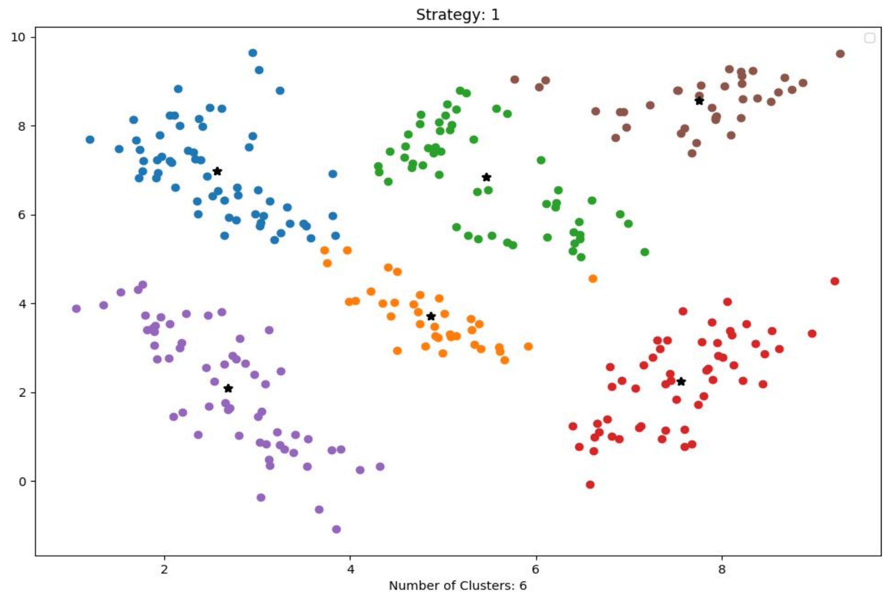




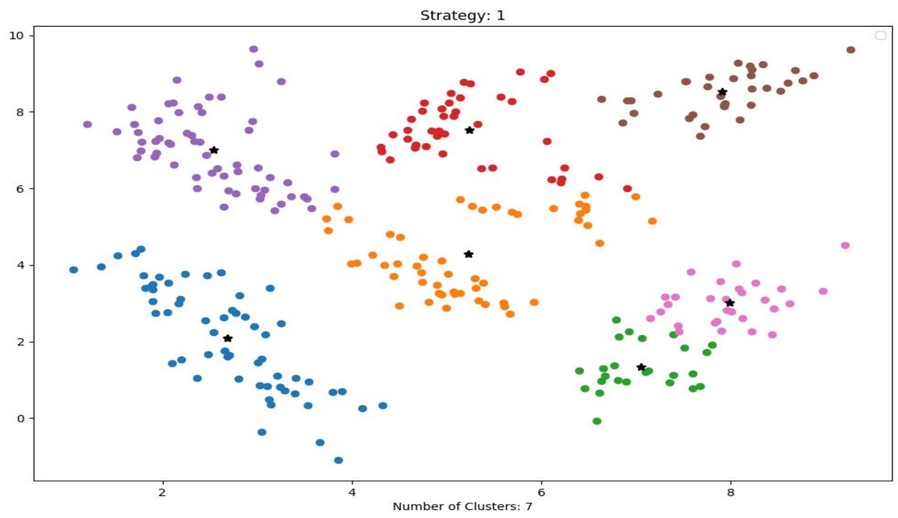
Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 4



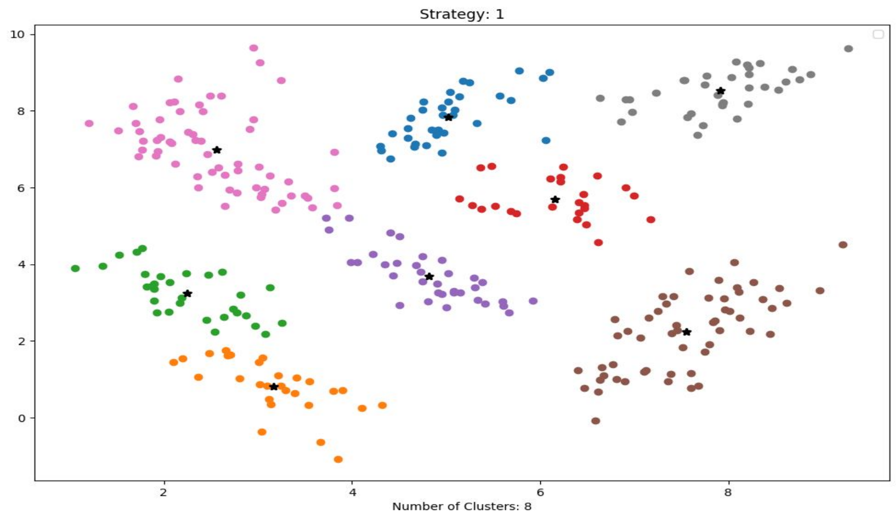
Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 5



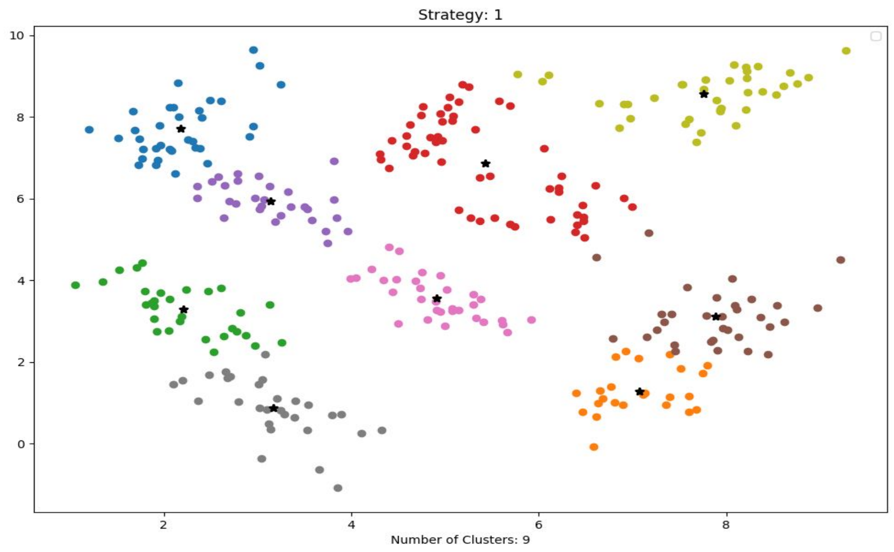
Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 6



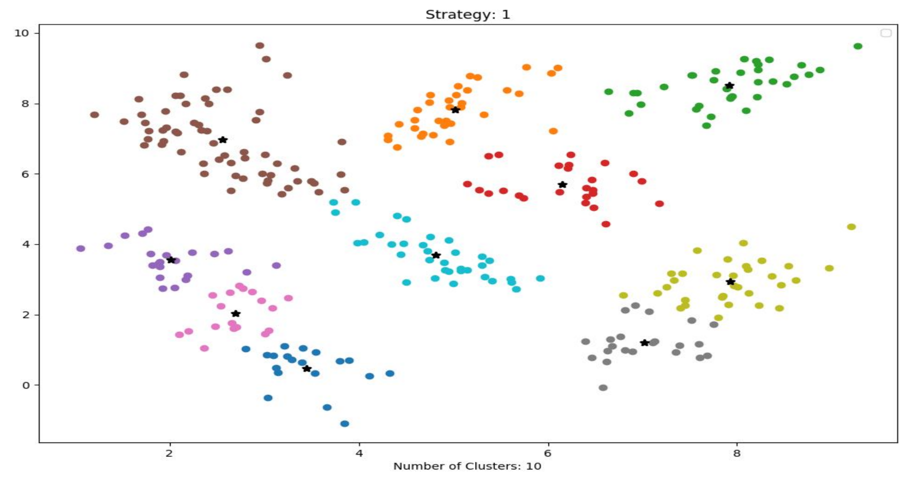
Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 7



Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 8



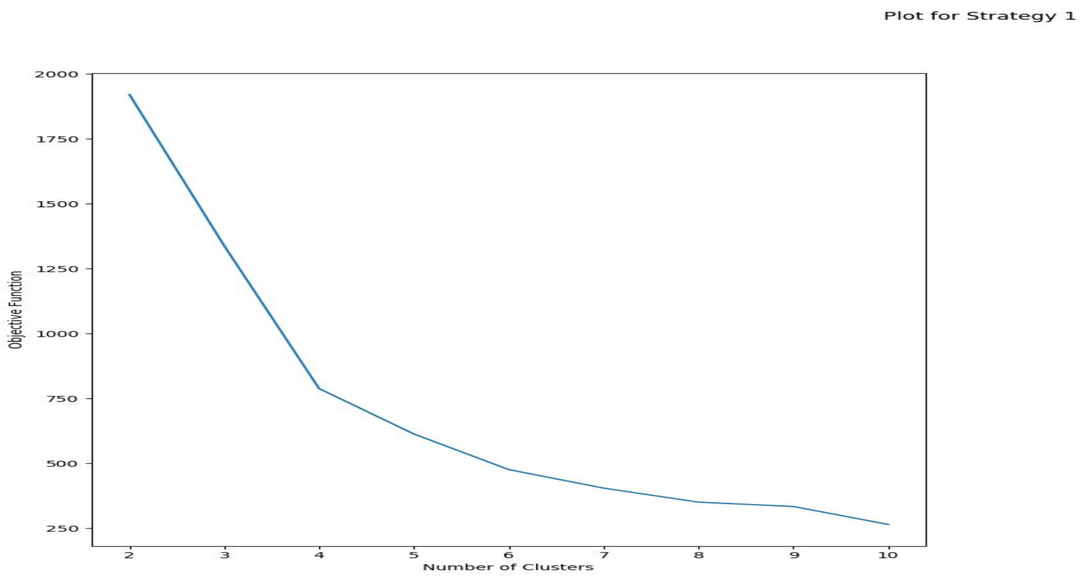
Strategy: 1
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 9

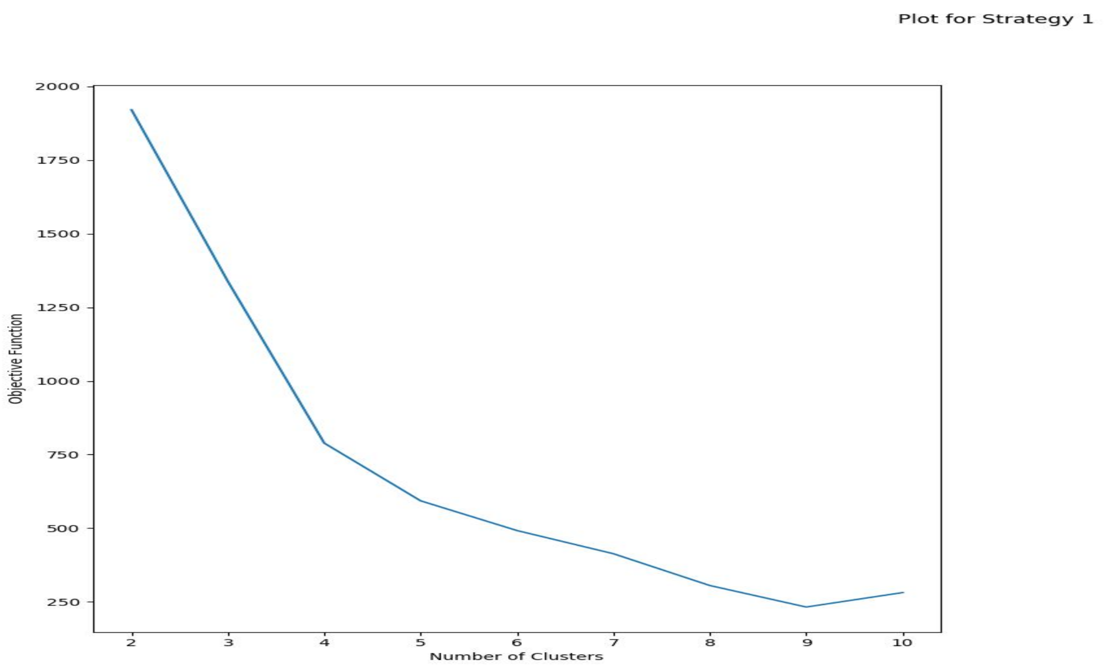


For all of the above procedures and for all the different ‘K’ values, I have calculated the objective function for each ‘K’ value. The objective function for K-means algorithm is as follow:

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2$$

Here, ***u*** is the mean value of i-th cluster and ***x*** is the particular data sample within that cluster. This objective function tries to minimize the intra-cluster distance. Below are two graphs for *‘objective function’* vs *‘K’* for two different runs of the program.



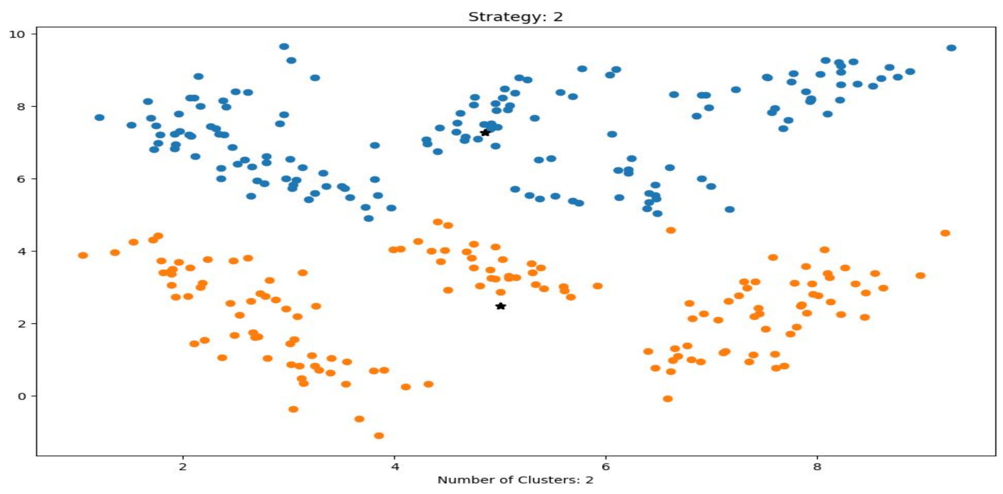


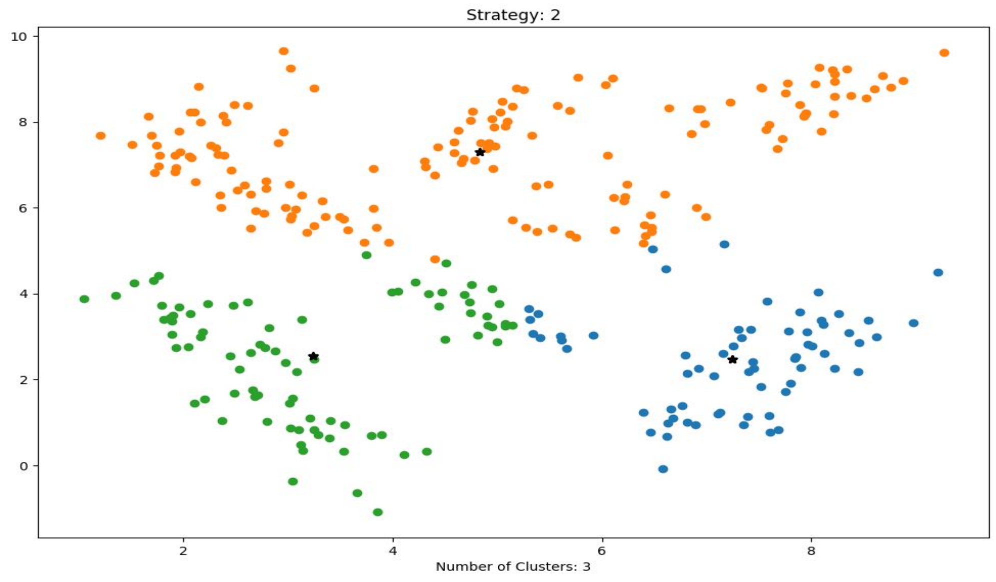
By using the **'Elbow method'** of clustering, I can say that for this strategy and for this dataset, the optimal value of clusters would be between 4 and 6, including.

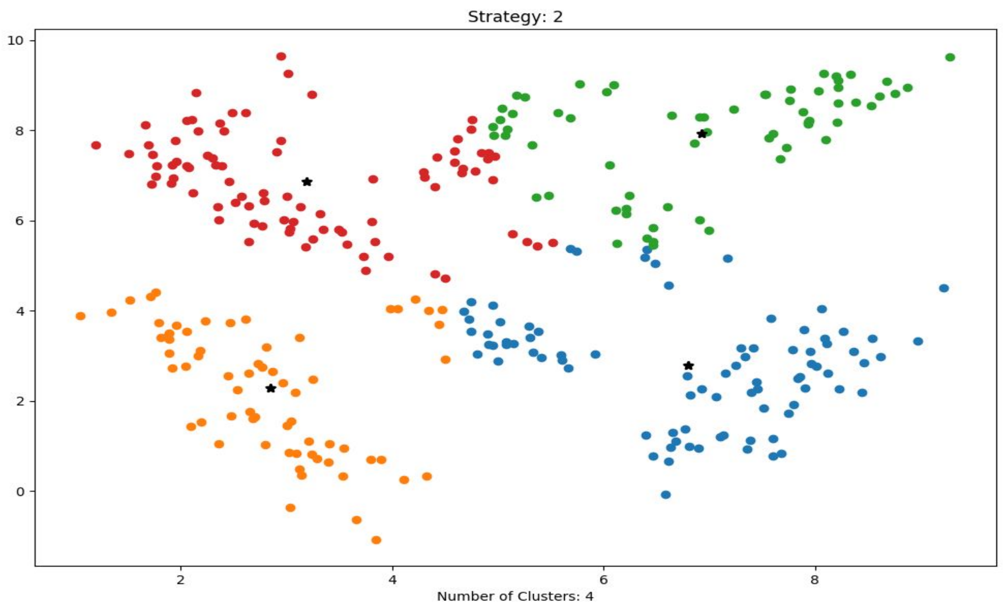
- *Strategy 2*

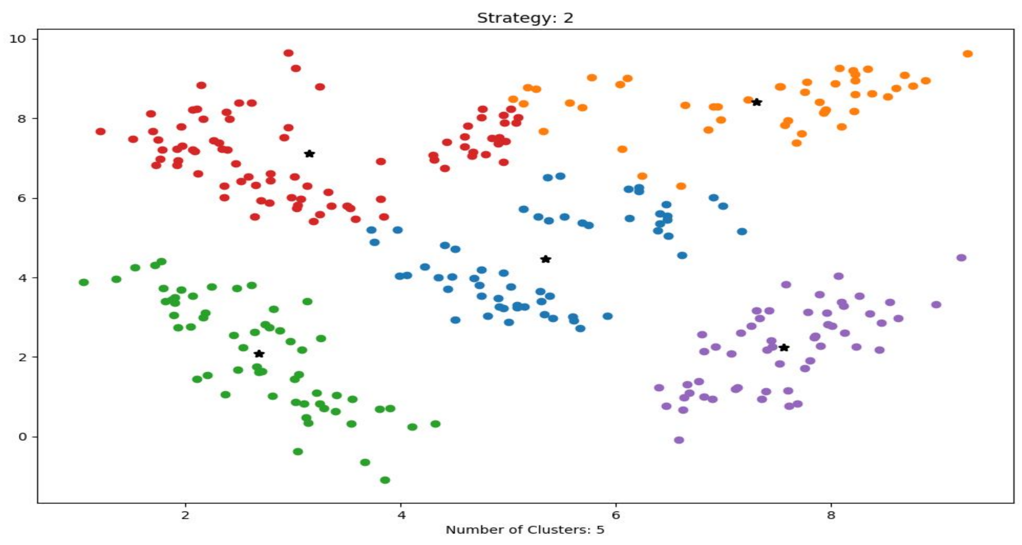
In this part, initially, only the first centroid has been generated randomly while all other 'K-1' centroids have been generated by maximizing the distance among all the centroids. First, the random centroid is generated using Python's built-in library function. Then, I have calculated the distance of all the other data samples to this random centroid and whichever point has maximum distance, I considered it as another centroid. This process carried out for generating all other centroids.

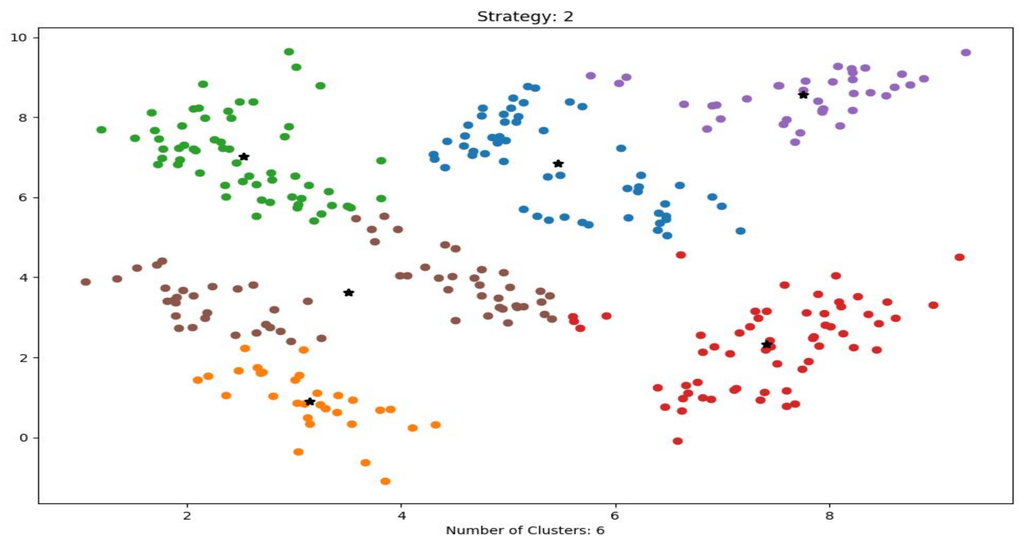
After this, the rest of the procedure is same as the strategy 1's procedure. For the value of 'K' from 2 to 10, I repeated the procedure and have gotten the different following clusters for the given data samples after the convergence. Black stars represent the different centroids.

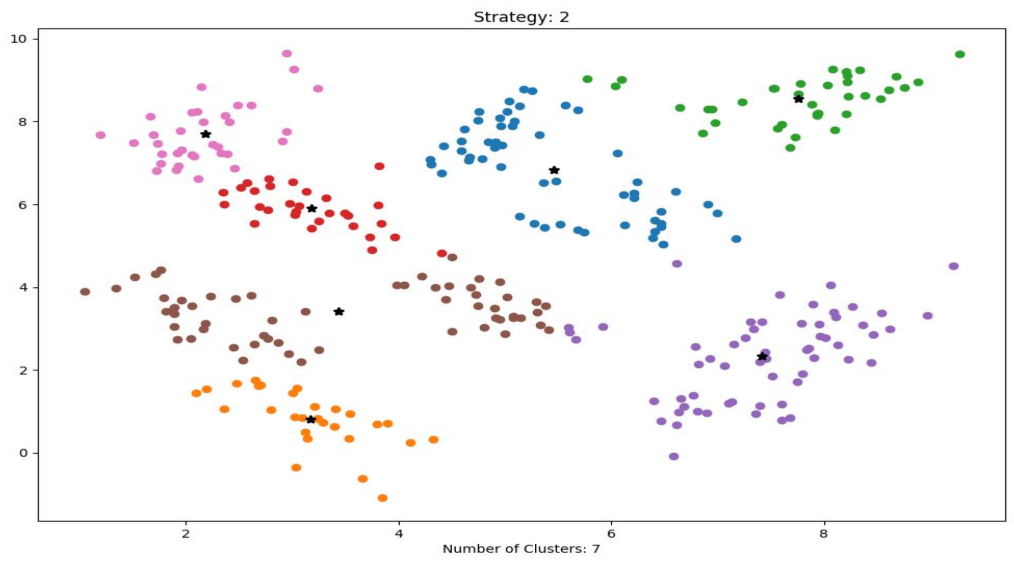

Strategy: 2
Number of Clusters: 2
10
8
6
4
2
0
2
4
6
8

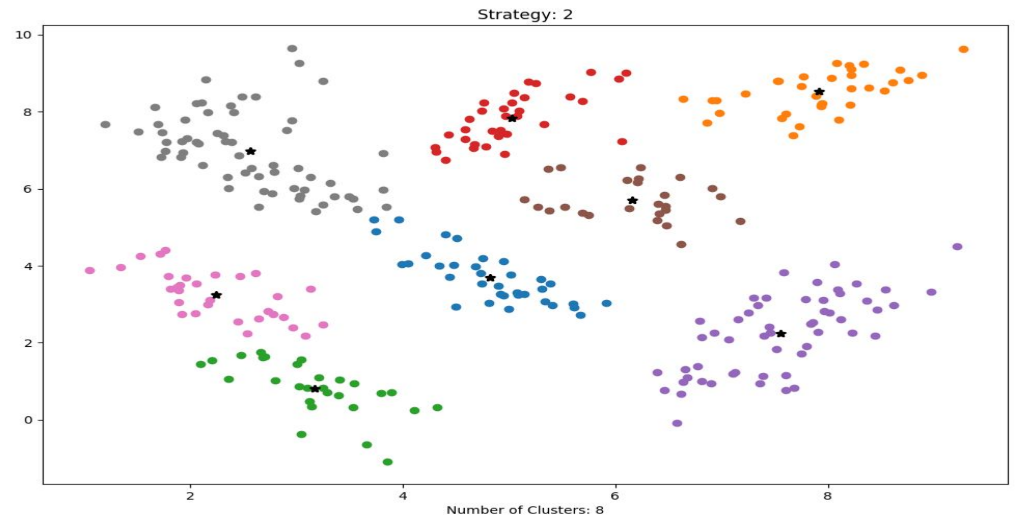

Strategy: 2
Number of Clusters: 3
10
8
6
4
2
0
2
4
6
8

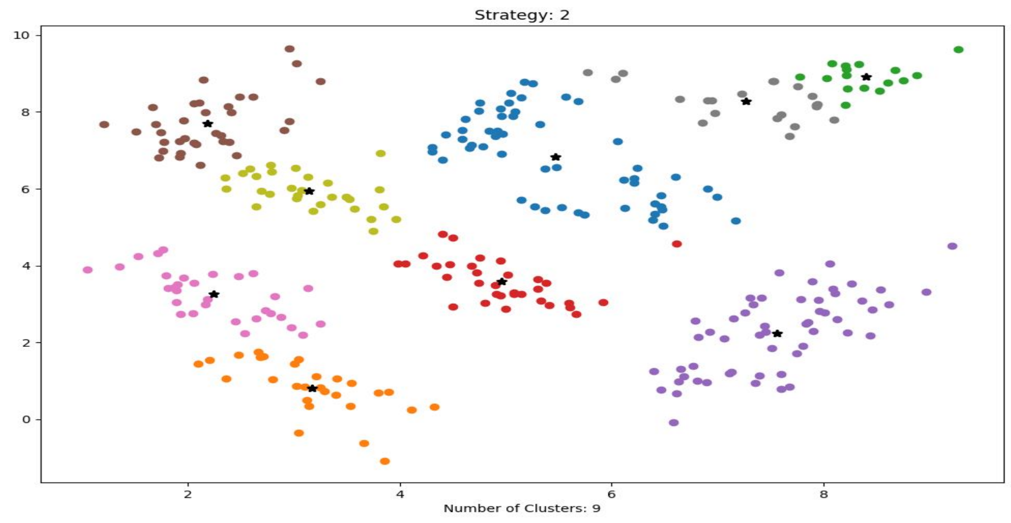

Strategy: 2
Number of Clusters: 4
10
8
6
4
2
0
2
4
6
8

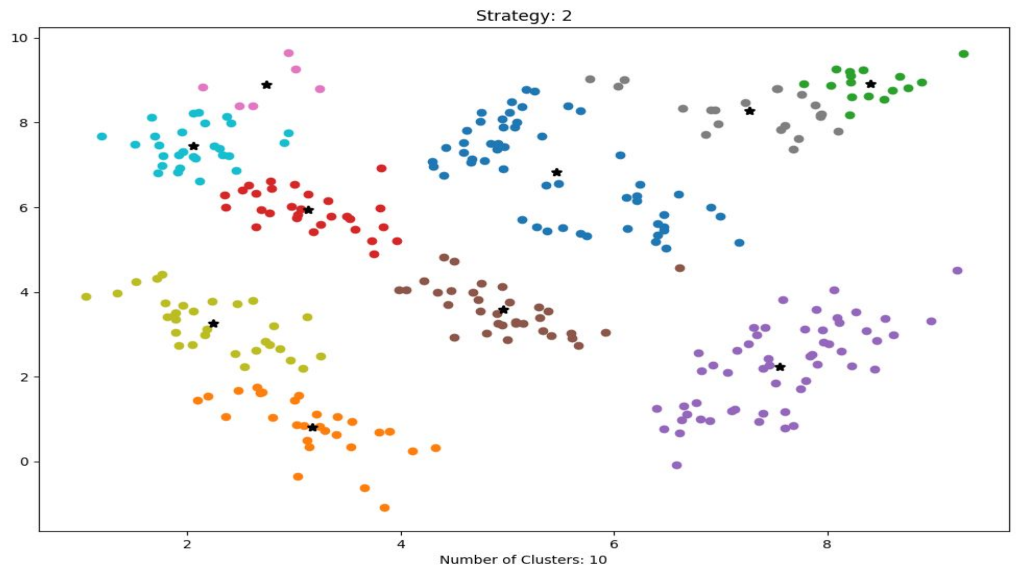

Strategy: 2
Number of Clusters: 5


Strategy: 2
Number of Clusters: 6

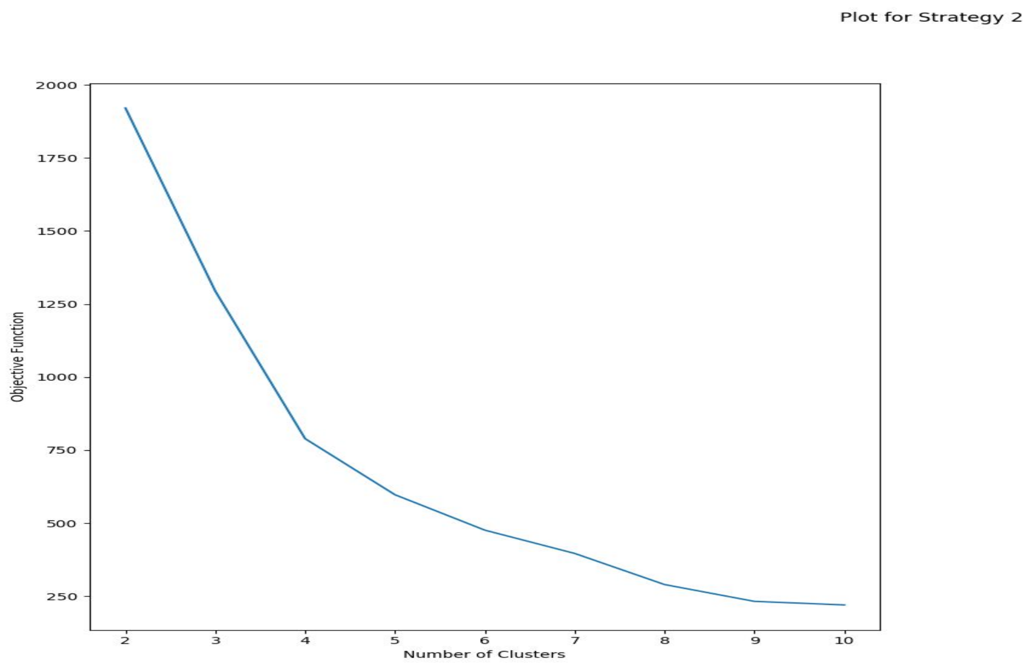

Strategy: 2
Number of Clusters: 7

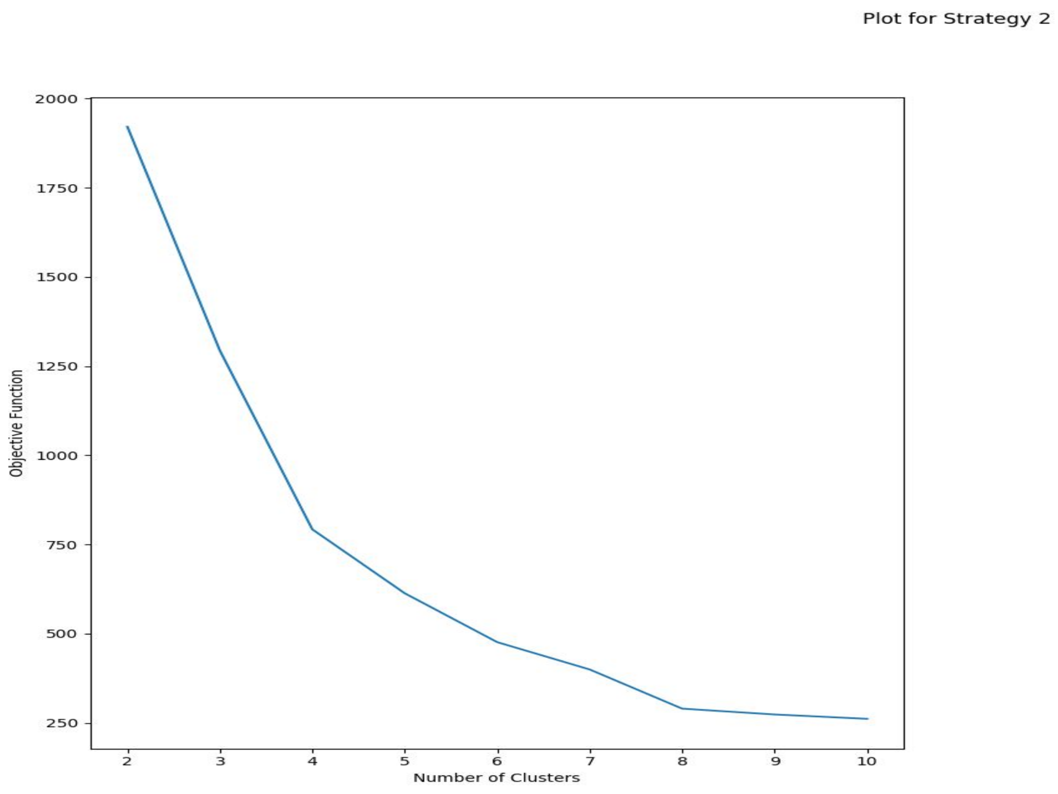

Strategy: 2
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 8


Strategy: 2
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 9


Strategy: 2
10
8
6
4
2
0
2
4
6
8
Number of Clusters: 10

As you can see, for strategy 2, all these graphs for different ‘K’ values are almost similar to strategy 1 after convergence. I have also generated ‘*objective function vs K’* graphs for two different runs for this strategy. They are as follows.





By using the same **‘Elbow Method’** for clustering, we can say that for this strategy and for this dataset, the optimal number of clusters would be the same as the strategy 1, i.e, between 4 and 6, including.

Conclusion

The ‘Elbow Method’ is a good measure for getting the optimal number of clusters for the particular dataset. In this scenario, I have not gotten the perfect but partial ‘Elbow’ curve as I have only 300 2-D data samples and I ran the algorithm for the K values of 2 to 10. If we could have a large dataset, the elbow curve would have been smoother for different values of the number of clusters. Still, the K-means algorithm has been able to minimize the intra-cluster distance for this small dataset also. And also in this case, the initial centroids position has not been able to change the cluster shape effectively because of the size of the dataset.