Strategy:1

Initialize Centroid Randomly:

Centroid are selected at random and assigned from the given dataset. For different values

of k, there will be k different clusters.

```
# Initializing the centroid randomly
def centroid_init(k, data_sets):
    randomly = np.random.choice(data.shape[0], k, replace=False)
    centroids = data_sets[randomly]
    return centroids
```

Objective Function: Calculating the Objective function for the K-means algorithm

```
# Objective Function for the K-means Algorithm
def k_means_obj(data_sets, centroids):
    objectiveval = []
    for r in data_sets:
        objectiveval.append(((np.linalg.norm((r - centroids), axis=0) ** 2)))
    return np.sum(objectiveval)
```

Euclidean Distance: Calculating the Euclidean distance between 2 points

```
# Calculating the Euclidean Distance from the given Coordinates
def cal_euclidean_distance(x_cord, y_cord, x_cent, y_cent):
    x_new = (x_cent - x_cord) ** 2
    y_new = (y_cent - y_cord) ** 2
    disteuc = math.sqrt(x_new + y_new)
    return disteuc
```

K-means Algorithm:

1-Loop k for k = 2,3,4,5,6,7,8,9,10:

2- Initialize centroid

3- Loop until the centroids have converges

4- Loop for all the data sets:

a. Calculate the Euclidean-Distance of the data set and the centroid

b. Select the minimum distance calculated and assign to the centroid

c. Calculate the mean of the points under cluster of centroids to find new centroid

d. Compare both the new centroid and the old Centroid for convergence

5- Loop Ends

Plot for K=2:

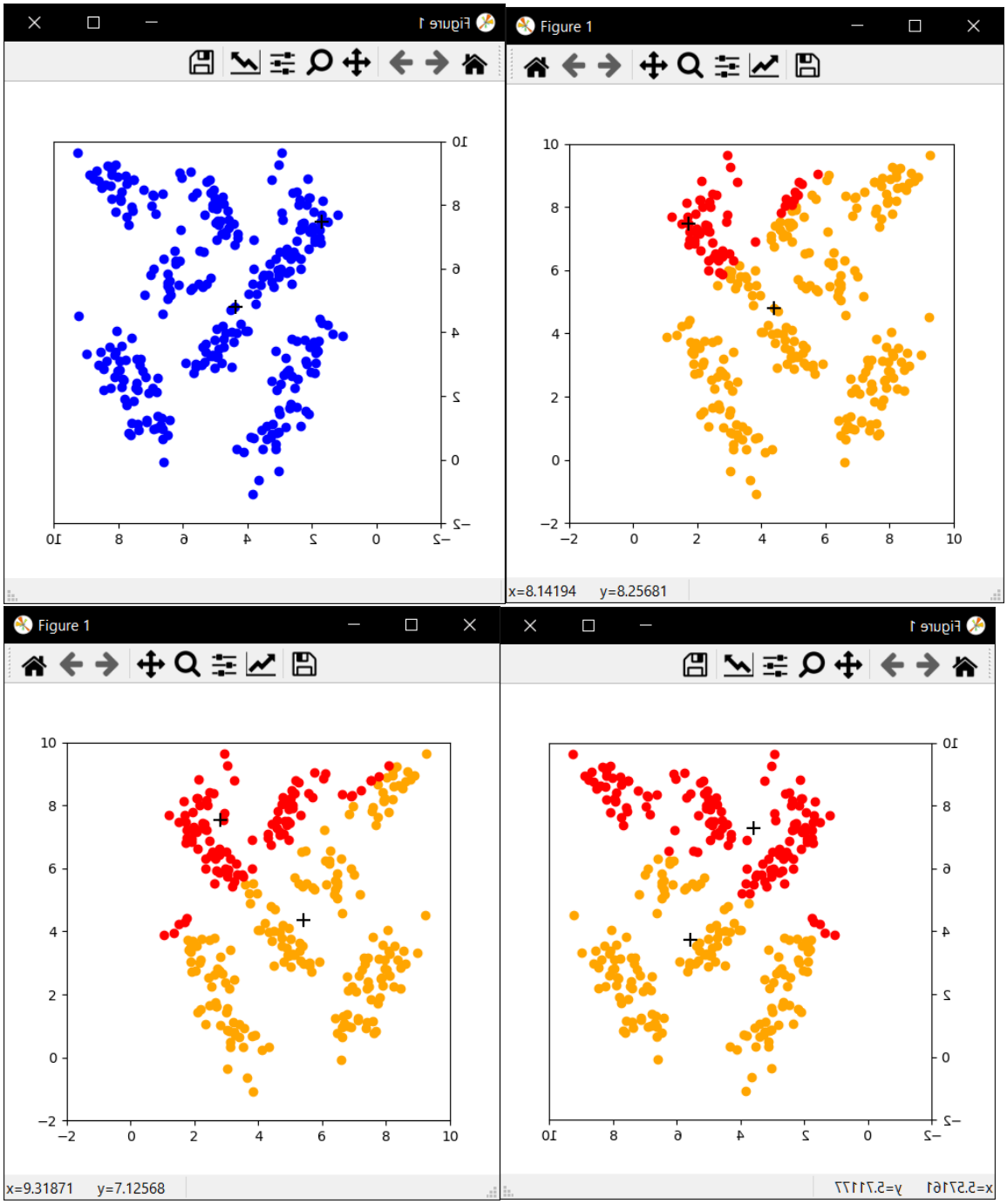
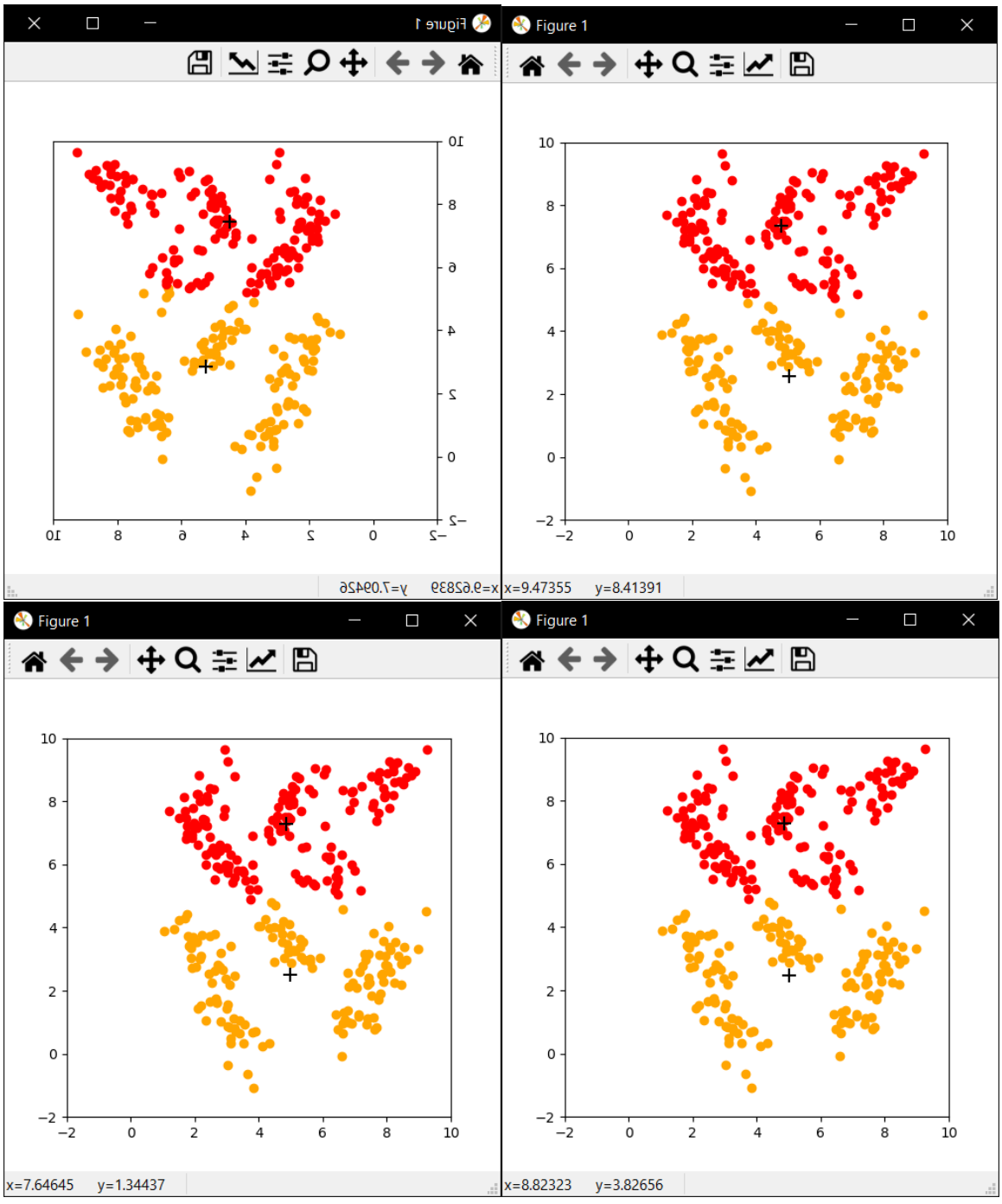
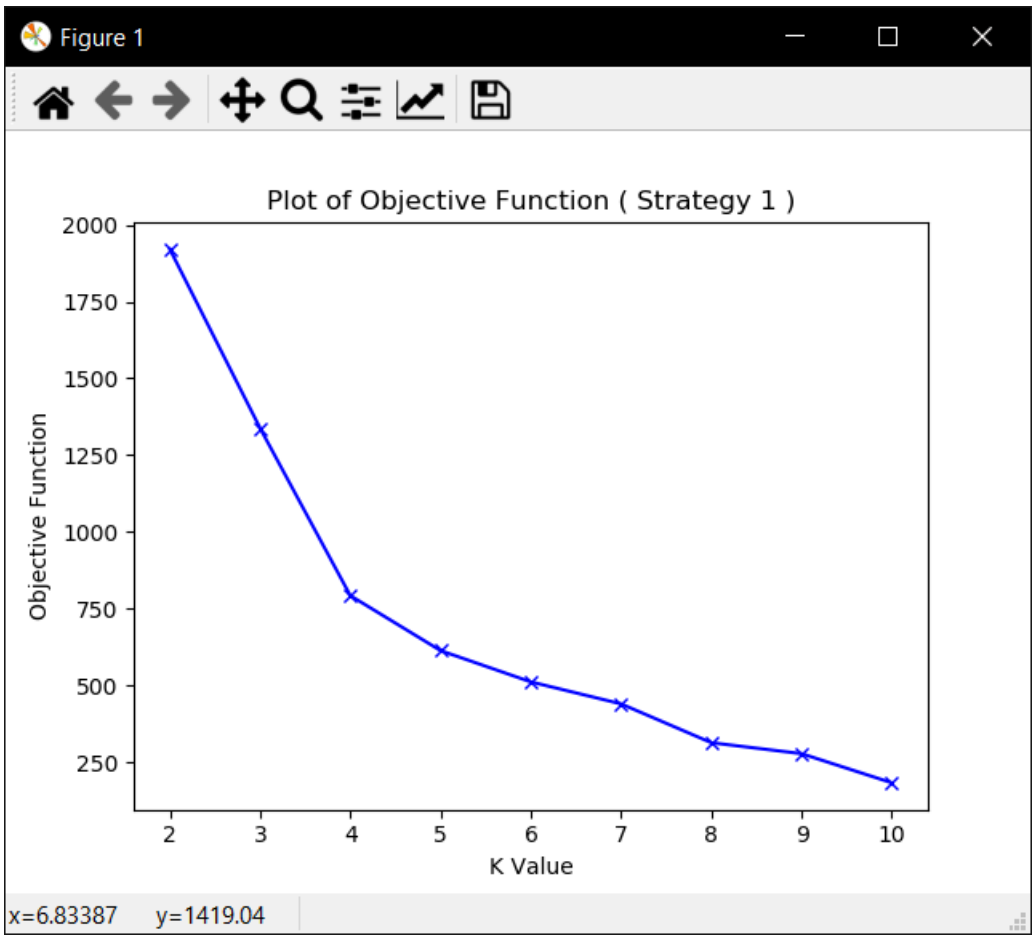


Figure 1
x=8.14194 y=8.25681
Figure 1
x=9.31871 y=7.12568



Because there are too many pictures afterwards, only the case when K = 2

Plot of Objective Function (Strategy 1):



Strategy 2:

Initializing Centroids: Initializing the first centroid randomly and then selecting other centroid whish are at maximum distance with other.

```
#Initializing randomly the First Centroid
def centroid_init(k, data_sets):
    index_list = []
    centroids = []
    temp = np.zeros([len(data_sets), k - 1])
    randomly = np.random.choice(data_sets.shape[0], 1, replace=False)
    index_list.append(randomly[0])
    centroids.append(data_sets[randomly])
    # Selecting Centroids which are at maximum distance from each other
    for i in range(k - 1):
        temp[:, i] = np.linalg.norm(centroids[i] - data_sets, axis=1)
        condition = True
        temp2 = np.mean(temp[:, :i + 1], axis=1)
        condition = True
        i = np.argmax(temp2)
        while (condition):
            if i in index_list:
                temp2[i] = 0
                i = np.argmax(temp2)
            else:
                condition = False
        centroids.append(np.asarray(data_sets[i]))
        index_list.append(i)
    centroidss = data_sets[index_list]
    return centroidss
```

K-means Algorithm

1-Loop k for k = 2,3,4,5,6,7,8,9,10:

2- Initialize centroid:

3- Loop for all the data sets:

a. Calculate the Euclidean-Distance of the data set and the centroid

b. Select the minimum distance belonging to the centroid

c. Calculate the mean under cluster of centroids to project new centroids

d. Compare both the new centroid and the old Centroid for convergence

5- Loop Ends

Plot for K=2:

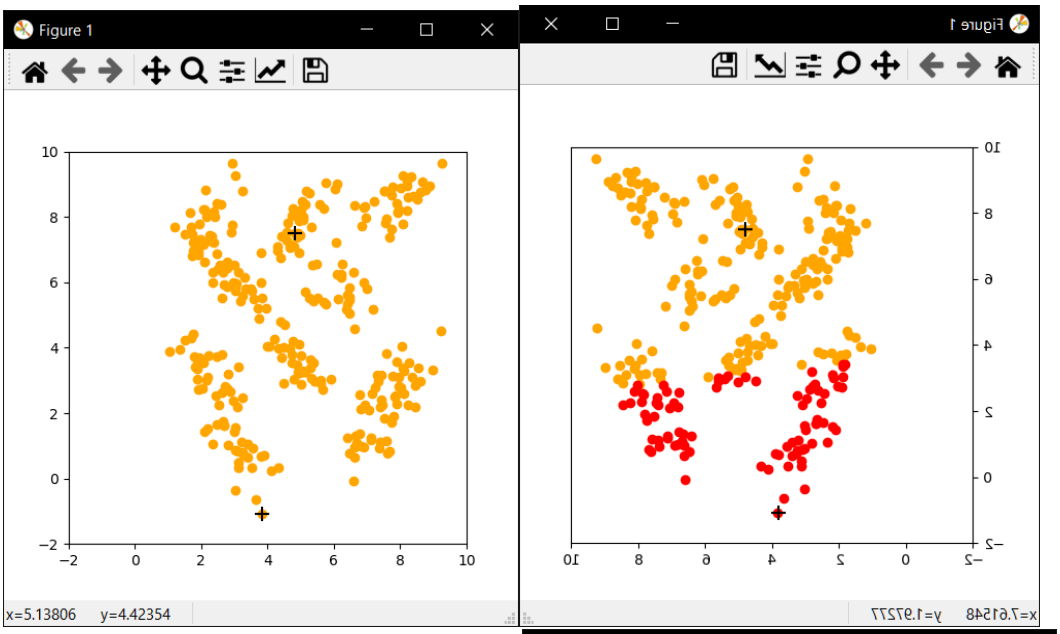


Figure 1
x=5.13806 y=4.42354

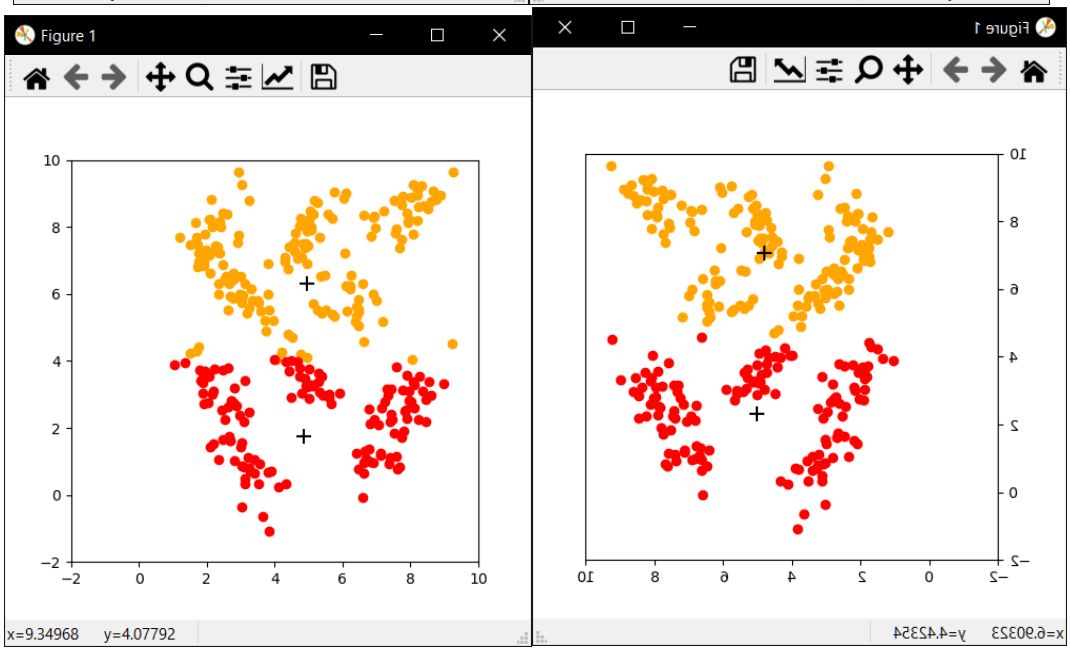
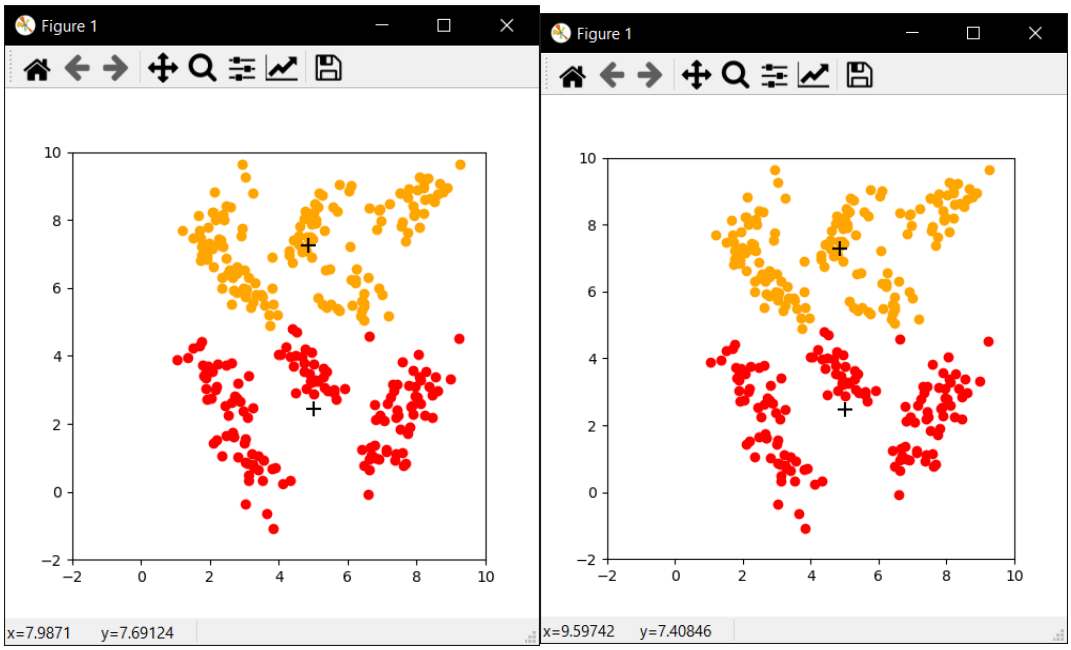
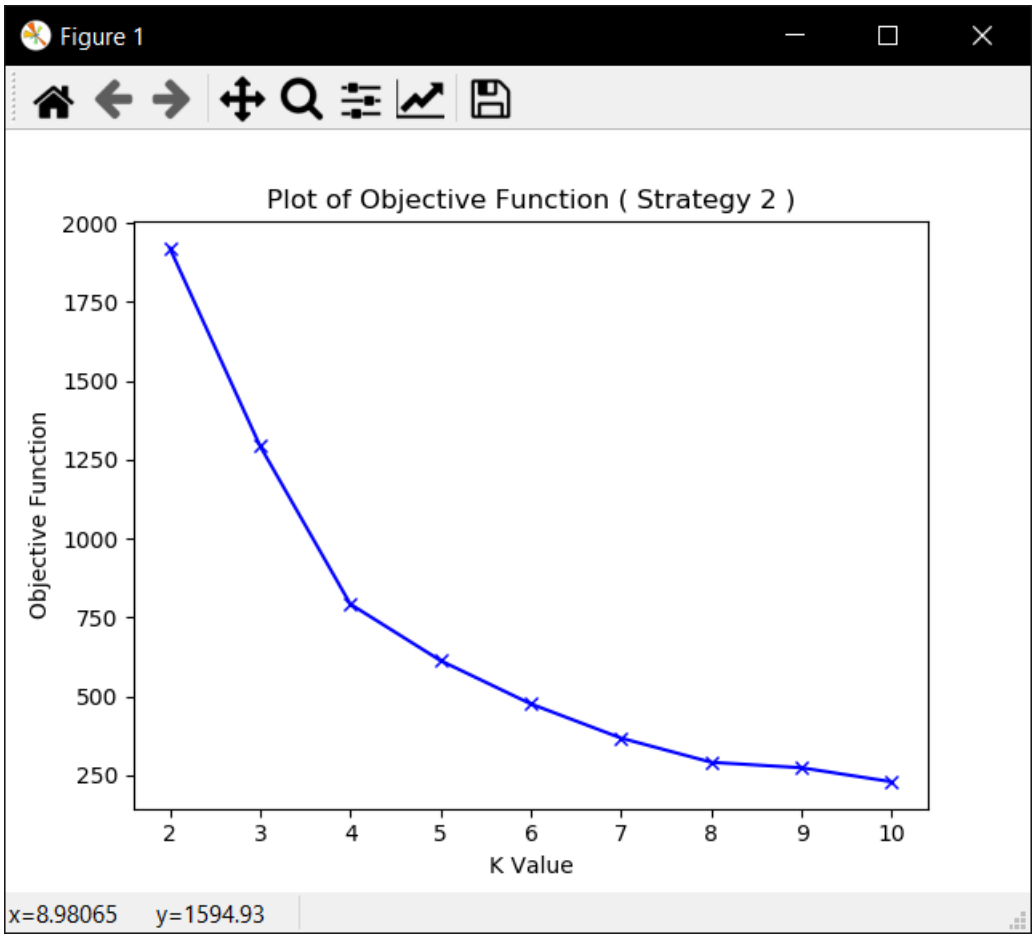


Figure 1
x=9.34968 y=4.07792



Because there are too many pictures afterwards, only the case when K = 2

Plot of Objective Function (Strategy 2):



Conclusion:

According to the graph of K value and objective function, as the number of k increases, the objective function decreases. In addition, after k = 4, the objective function decreases slowly. Therefore, it can be said that the additional optimal number k = 4.