**NAME OF EXPERIMENT:**

Implementation of K-Means Clustering in Python.

**INTRODUCTION:**

K-Means Algorithm is a centroid based clustering algorithm which groups the data into K clusters, where K is a predefined number. Each data point is assigned to the closed cluster centroid based on the distance between the point and centroid. The centroids are then recalculated until their convergence. It uses iterative relocation techniques.

**ALGORITHM:**

1. Randomly initialize K cluster centroids.
2. Calculate the Euclidean distance between each point and each centroid using formula:

d =

1. Assign each point to the cluster associated with the nearest centroid.
2. Recalculate the cluster centroids by taking the mean(x̅) of all data points assigned to each cluster. This will give you K new cluster centroids.
3. Repeat steps 3 and 4 until the cluster assignments of the data points no longer change, or until a maximum number of iterations is reached.

**SOLUTION:**

Clusters the following instances of given data with the help of K means algorithm (Take K=2).

|  |  |  |
| --- | --- | --- |
| **Instance** | **X** | **Y** |
| 1 | 31 | 31.5 |
| 2 | 31 | 34.5 |
| 3 | 32 | 31.5 |
| 4 | 32 | 33.5 |
| 5 | 33 | 32.5 |
| 6 | 33 | 34 |

→ Solution, here, K = 2.

Let the initial cluster centers are: C1 = (31, 31.5) and C2 = (32, 31.5).

**Iteration 1:** Calculate distance between cluster centers and each data point using Euclidean distance.

d(C1,2) = = 3

d(C2,2) = = 3.163

Here, d(C1,2) < d(C2,2). So, data point 1 belongs to cluster C1.

Similarly,

|  |  |  |  |
| --- | --- | --- | --- |
| **Points** | **Distance to** | | |
| **C1(31, 31.5)** | **C2(32, 31.5)** | **Clusters** |
| (31,34.5) | 3 | 3.163 | C1 |
| (32,33.5) | 2.236 | 2 | C2 |
| (33,32.5) | 2.236 | 1.414 | C2 |
| (33,34) | 3.2 | 2.7 | C2 |

Now, after iteration:

Cluster C1 = {(31, 31.5), (31, 34.5)}

Cluster C2 = {(32, 31.5), (32, 33.5), (33, 32.5), (33, 34)}

**Iteration 2:** New cluster centers are:

C1 = = (31,33)

C2 = ) = (32.5,32.8753)

Again, calculating distance between new cluster centers and each data point using Euclidean distance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Points** | **Distance to** | | |
| **C1(31, 33)** | **C2(32.5, 32.875)** | **Clusters** |
| (31,31.5) | 1.5 | 2.035 | C1 |
| (31,34.5) | 1.5 | 2.22 | C1 |
| (32,31.5) | 1.8 | 1.463 | C2 |
| (32,33.5) | 1.12 | 0.8 | C2 |
| (33,32.5) | 2.06 | 0.625 | C2 |
| (33,34) | 2.236 | 0.718 | C2 |

Now, after iteration:

Cluster C1 = {(31, 31.5), (31, 34.5)}

Cluster C2 = {(32, 31.5), (32, 33.5), (33, 32.5), (33, 34)}

The cluster of data points obtained in the **2nd** iteration is same as of **1st**iteration. So, terminate.

**ADVANTAGES OF K-MEANS ALGORITHM:**

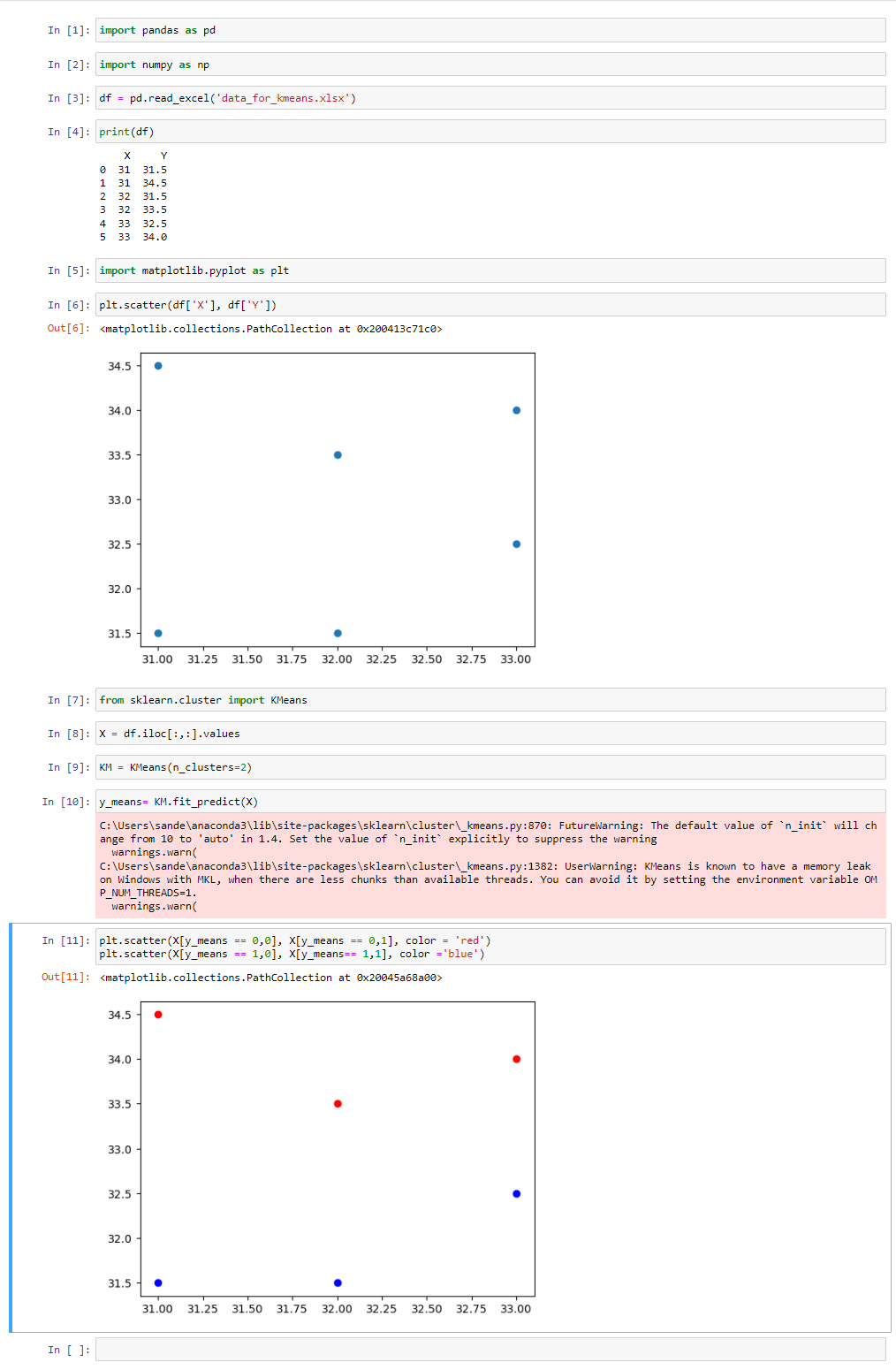
* It is easy to implement.
* It is scalable to huge data set and faster.

**DISADVANTAGES OF K-MEANS ALGORITHM:**

* It is sensitive to outliers.
* Choosing the K values manually is a tough job.

**PROGRAM IMPLEMENTATION IN PYTHON:**

Requirement: Anaconda Navigator



**CONCLUSION:**

Hence, we have successfully implemented the K-Means algorithm in Python to cluster the instance of given data.