Matthew Hess

Red-ID: 818299658

**Assignment 07**

I would first like to state the following.

Line 48: ‘centroid\_points = np.random.randint(samples\_size, size=(k\_value, data\_dimensions))’

This method is an inaccurate way choosing initial random cluster values. Notice that for each value in an array shape (k\_value, data\_dimensions), the range of each of the possible value is minimum 0 to the max 329 which is the sample\_size.

If k\_value = 1

Initial cluster would be [[0-329, 0-329, 0-329, 0-329, 0-329, 0-329, 0-329, 0-329, 0-329]]

Scanning through the csv file I notice that the initial cluster should actually be a random vector between the following ranges.

[[105-910, 5159-23604, 43-7850, 308-2498, 1145-8625, 1701-3781, 52-56745, 300-4800, 3045-9980]]

data.min(axis=0) [ 105 5159 43 308 1145 1701 52 300 3045]

data.max(axis=0) [ 910 23640 7850 2498 8625 3781 56745 4800 9980]

In fact when I run the program with line 48 these are my results for when k\_value is 3:

Found after **2** iterations

k\_value: 3

number of points assigned to cluster 1: 329

number of points assigned to cluster 2: 0

number of points assigned to cluster 3: 0

cluster #0 coordinates:

[ 539. 8347. 1186. 961. 4210. 2815. 3151. 1846. 5525.]

cluster #1 coordinates:

[0. 0. 0. 0. 0. 0. 0. 0. 0.]

cluster #2 coordinates:

[0. 0. 0. 0. 0. 0. 0. 0. 0.]

Notice how two centroids are assigned to [0. 0. 0. 0. 0. 0. 0. 0. 0.] and that the number of points assigned to these centroids is 0. This is definitely not what the results should look like, the data isn’t spread out to centroids as the model is intended to do.

When using the sum of square errors formula to calculate the overall distances from the clusters these were my results:

K = 3

SSE = 10635737016.96

K = 6

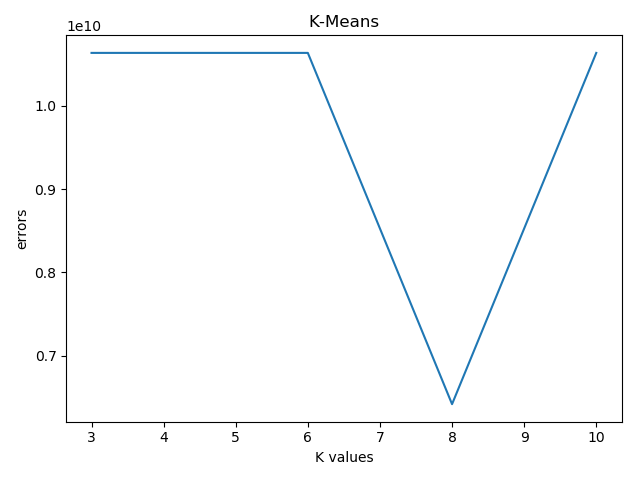
SSE = 10635737016.96

K = 8

SSE = 6418357530.11

K = 10

SSE = 10635737016.96



I then changed the clusters to be initialized the following way:

‘

random\_index\_samples = random.sample(range(samples\_size), k\_value)

for index in random\_index\_samples:

centroid\_points.append(xy\_samples[index, :])

centroid\_points = np.array(centroid\_points)

’

These are my results for when k\_value is 3:

Found after **8** iterations

k\_value: 3

number of points assigned to cluster 1: 70

number of points assigned to cluster 2: 6

number of points assigned to cluster 3: 253

cluster #0 coordinates:

[ 599. 10897. 2136. 1107. 5139. 2976. 6825. 2442. 5846.]

cluster #1 coordinates:

[ 654. 11964. 5598. 1514. 6942. 3427. 27528. 2834. 5655.]

cluster #2 coordinates:

[ 519. 7555. 818. 908. 3888. 2756. 1556. 1658. 5434.]

When using the sum of square errors formula to calculate the overall distances from the clusters these were my results:

K = 3

SSE = 4385767709.19

K = 6

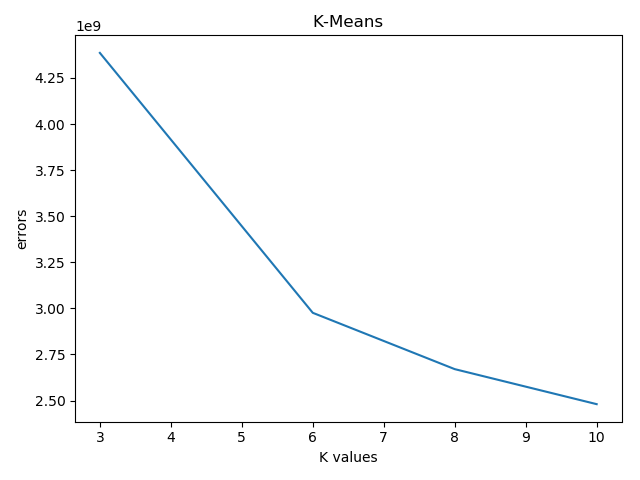
SSE = 2975909345.32

K = 8

SSE = 2670448319.15

K = 10

SSE = 2480443445.50



Overall I noticed that the greater the k\_value the less error was produced. In addition selecting k\_value random samples as centroids made the program converge after more iterations oppose to the first method discussed which was faulty. Please run the code to see a more detailed report.