**Task 1** Suppose we have 10 college football teams X1 to X10. We want to cluster them into 2

groups. For each football team, we have two features: One is # wins in Season 2016, and the

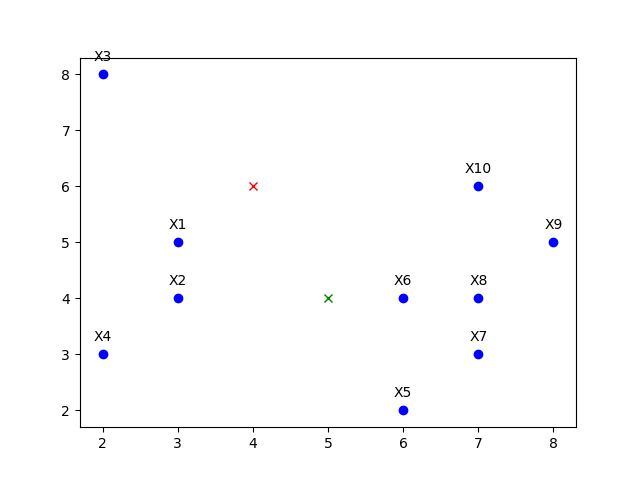
other is # wins in Season 2017.

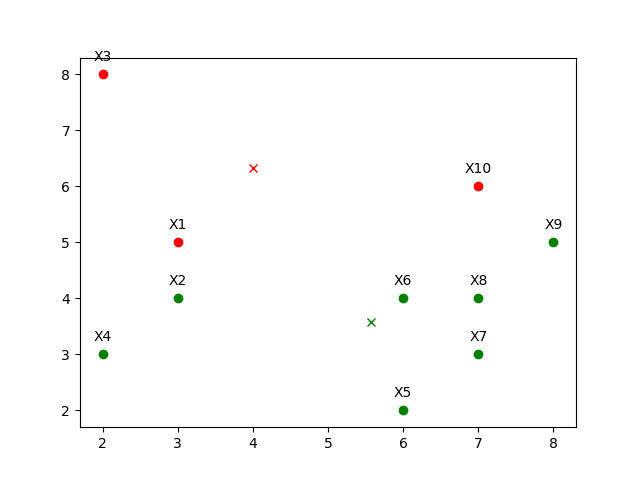
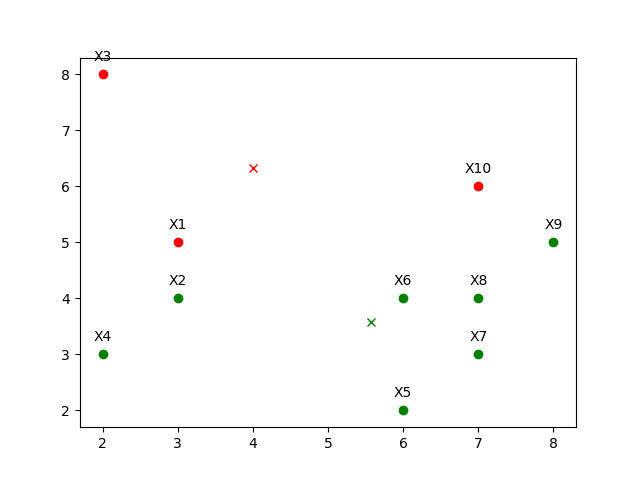
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
| (3,5) | (3,4) | (2,8) | (2,3) | (6,2) | (6,4) | (7,3) | (7,4) | (8,5) | (7,6) |

To solve this problem, I wrote a python script to print the initialization, the results after the first iteration and the results after several iterations. The stopping criteria I used was the Euclidean distance between the new centroid and the old centroid being less than 0.0001.

**(1) Initialize with two centroids, (4, 6) and (5, 4). Use Manhattan distance as the distance metric. First, perform one iteration of the K-means algorithm and report the coordinates of the resulting centroids. Second, please use K-Means to find two clusters.**

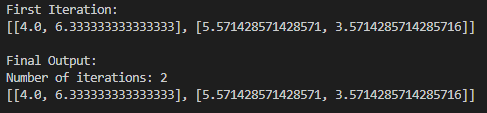
The initialization is shown below:



The results after the first iteration are shown below on the left and the final results shown on the right:

Iteration 1

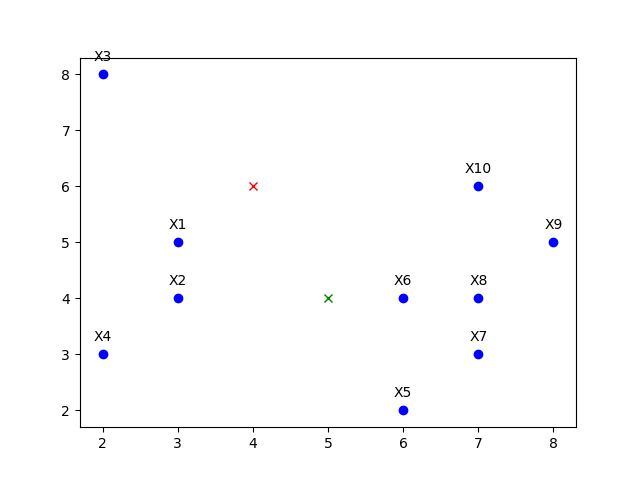
Final Results

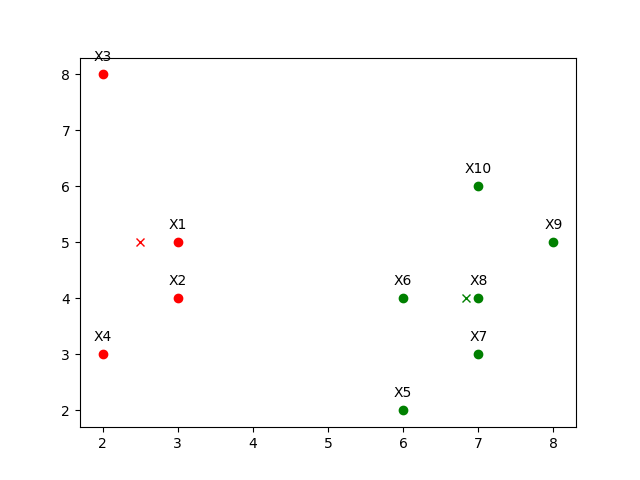
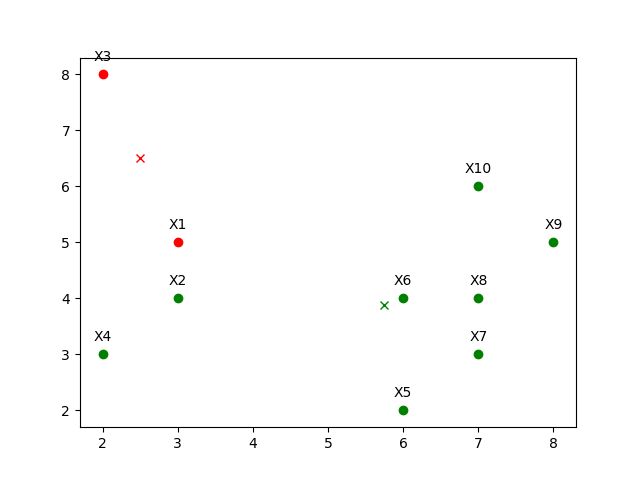


The results above show the centroids after the first iteration and the centroids after the stopping criteria was met. The results also show the number of iterations required to reach the stopping criteria.

**(2) Initialize with two centroids, (4, 6) and (5, 4). Use Euclidean distance as the distance metric. First, perform one iteration of the K-means algorithm and report the coordinates of the resulting centroids. Second, please use K-Means to find two clusters.**

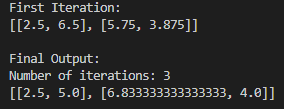
The initialization is shown below:



The results after the first iteration are shown below on the left and the final results shown on the right:

Iteration 1

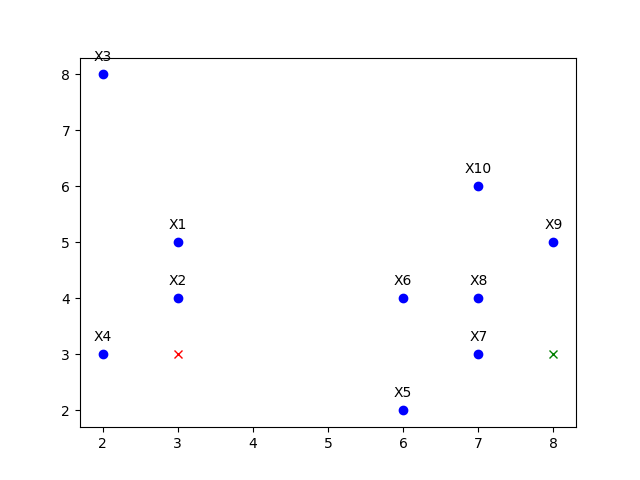
Final Results

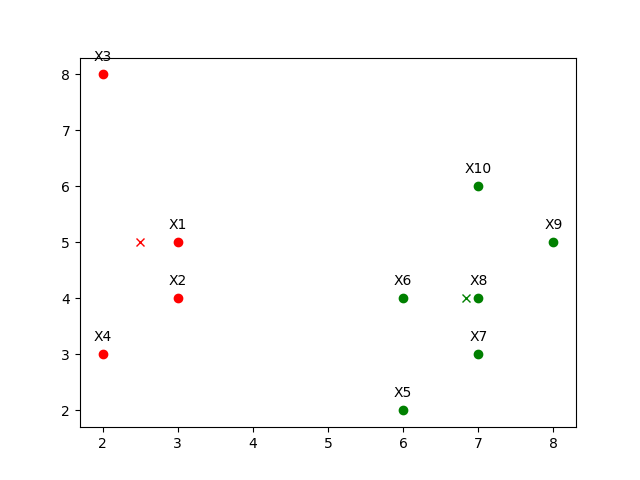
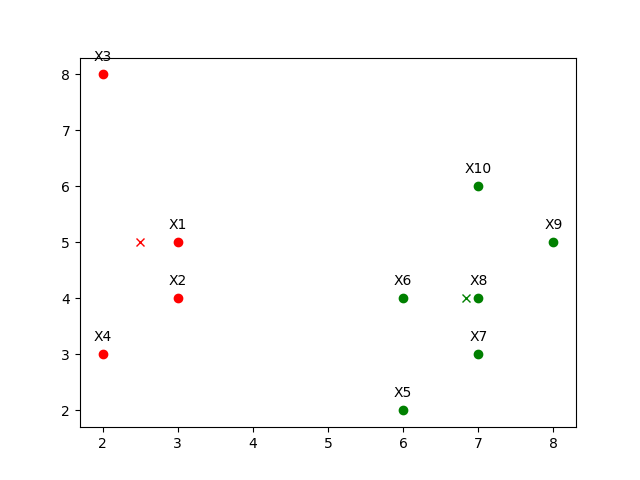


The results above show the centroids after the first iteration and the centroids after the stopping criteria was met. The results also show the number of iterations required to reach the stopping criteria.

**(3) Initialize with two centroids, (3, 3) and (8, 3). Use Manhattan distance as the distance metric. First, perform one iteration of the K-means algorithm and report the coordinates of the resulting centroids. Second, please use K-Means to find two clusters.**

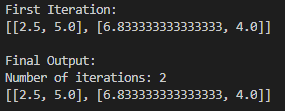
The initialization is shown below:



The results after the first iteration are shown below on the left and the final results shown on the right:

Iteration 1

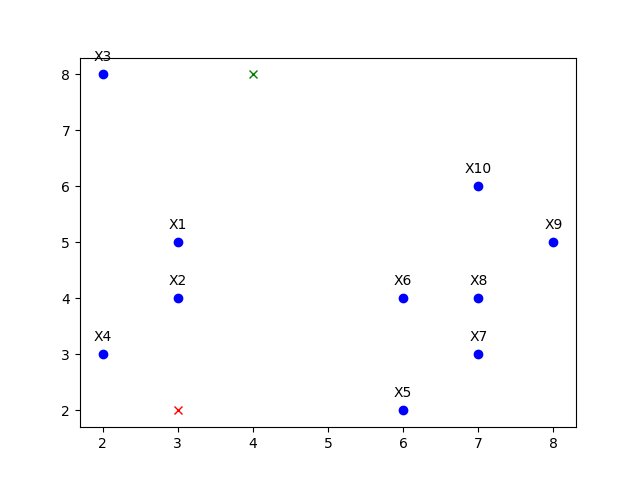
Final Results

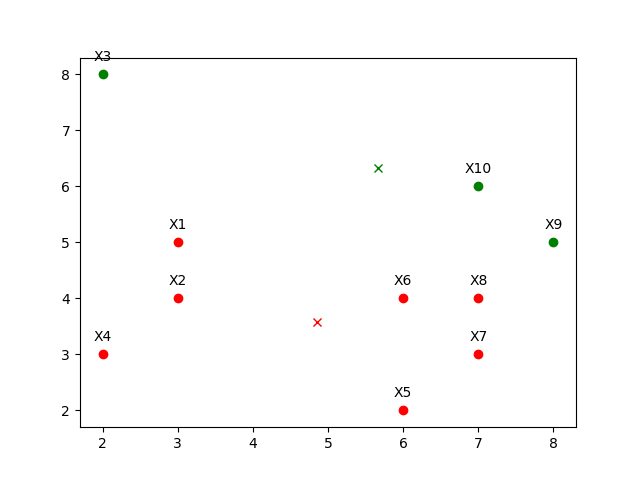
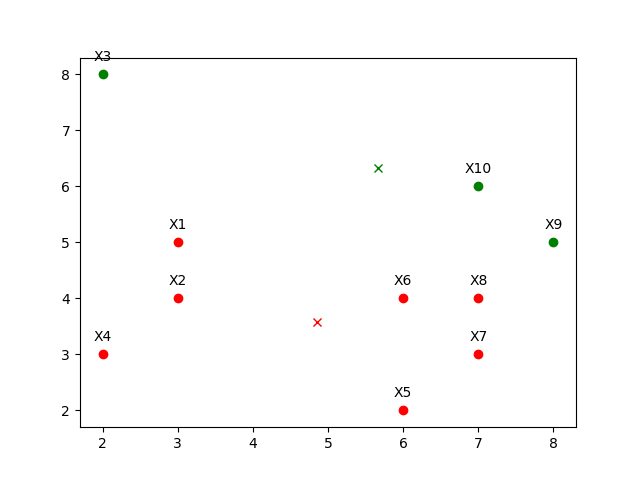


The results above show the centroids after the first iteration and the centroids after the stopping criteria was met. The results also show the number of iterations required to reach the stopping criteria.

**(4) Initialize with two centroids, (3, 2) and (4, 8). Use Manhattan distance as the distance metric. First, perform one iteration of the K-means algorithm and report the coordinates of the resulting centroids. Second, please use K-Means to find two clusters.**

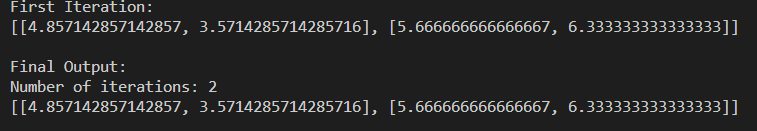
The initialization is shown below:



The results after the first iteration are shown below on the left and the final results shown on the right:

Iteration 1

Final Results

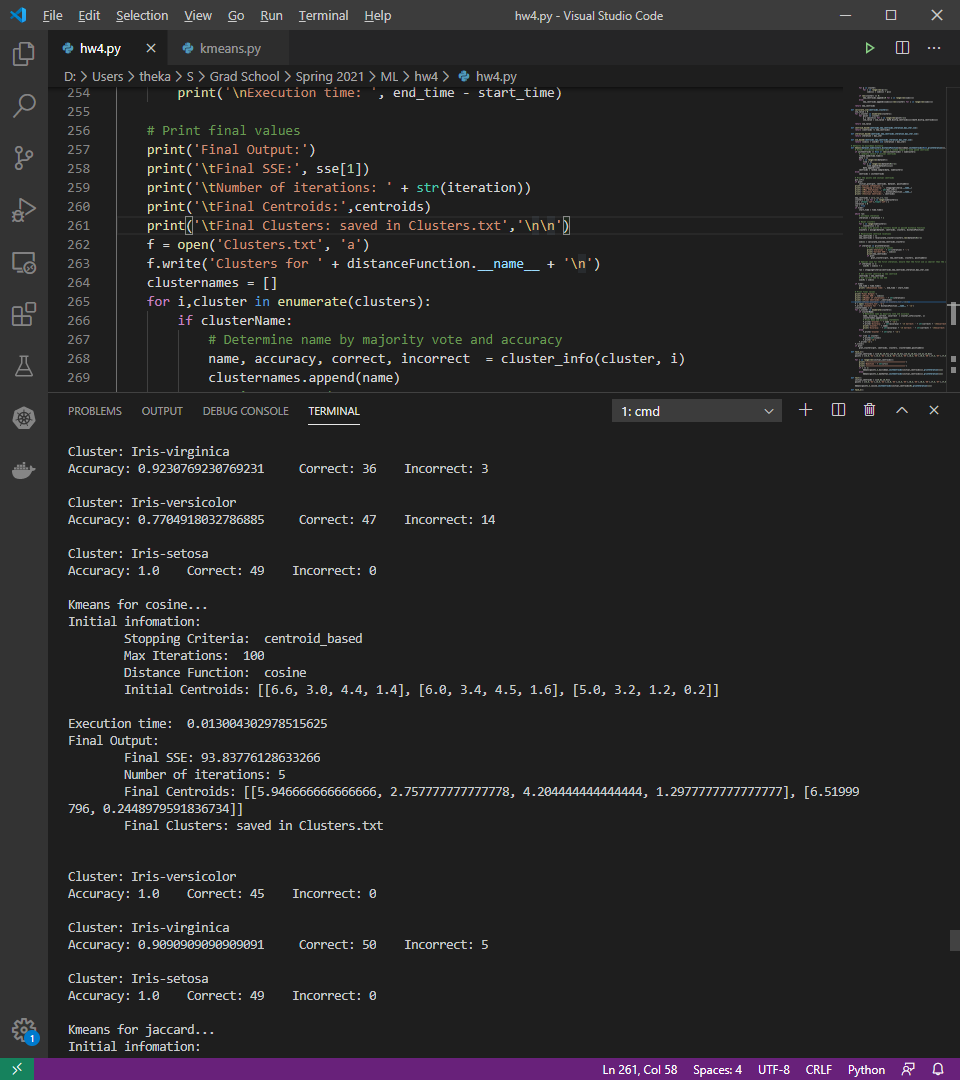


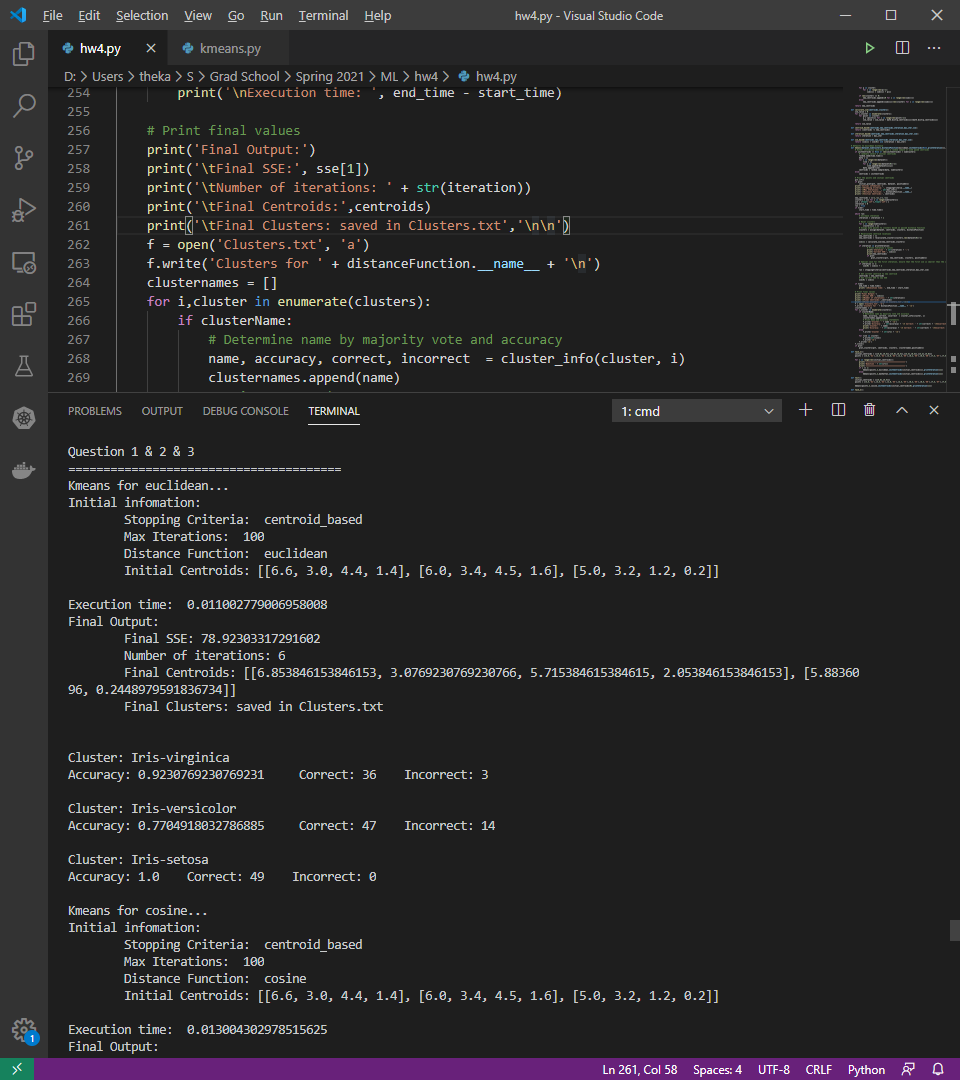
The results above show the centroids after the first iteration and the centroids after the stopping criteria was met. The results also show the number of iterations required to reach the stopping criteria.

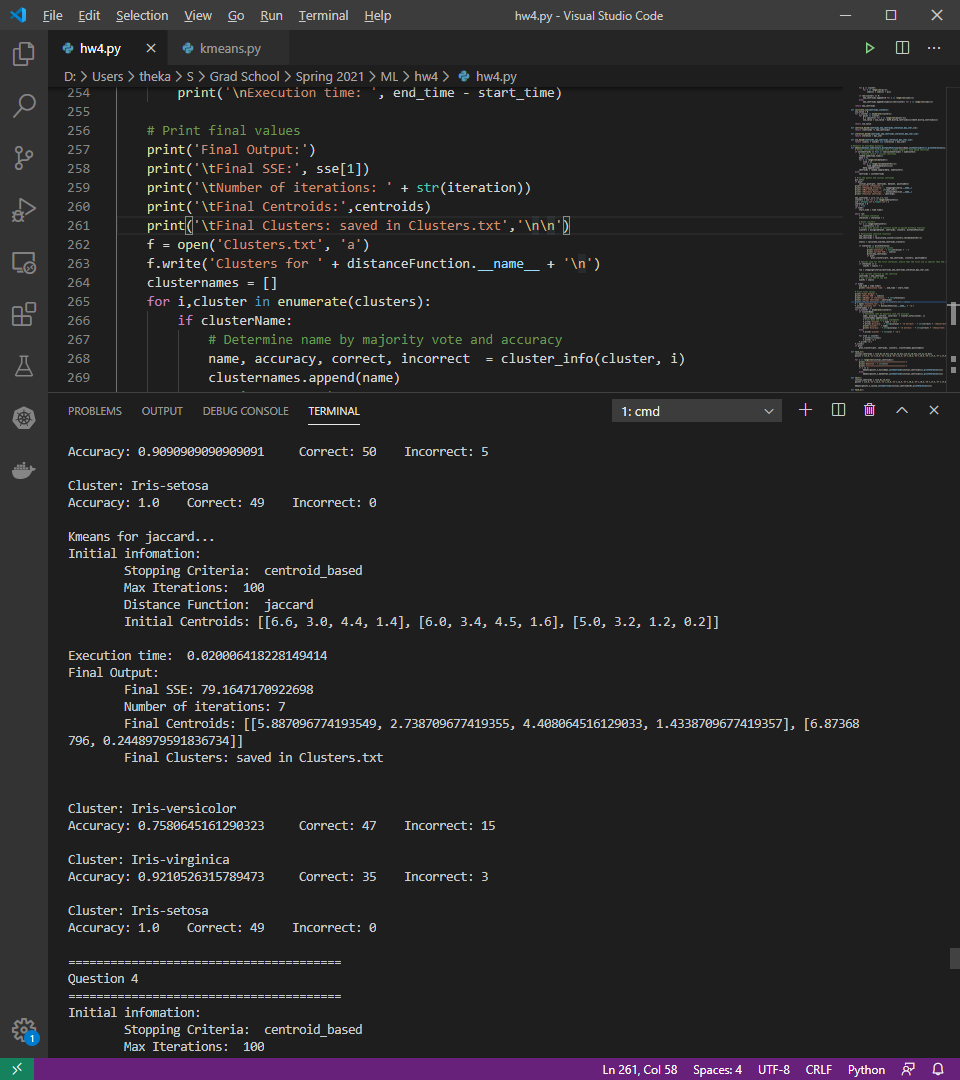
**Task 2 K-Means Clustering with Real World Dataset**

First, download the Iris data set from: https://archive.ics.uci.edu/ml/datasets/Iris. Then, implement the K-means algorithm. K-means algorithm computes the distance of a given data point pair. Replace the distance computation function with Euclidean distance, 1- Cosine similarity, and 1 – the **Generalized** Jarcard similarity

(<https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/jaccard.htm>).

For Q1 and Q3, I will refer to the images below.





**Q1: Run K-means clustering with Euclidean, Cosine and Jarcard similarity. Specify K= the number of categorical values of y (the variable of label). Compare the SSEs of Euclidean-Kmeans Cosine-K-means, Jarcard-K-means. Which method is better?**

The K I specified was 3 for the three different labels: Iris-setosa, Iris-versicolor, Iris-virginica. The data in the table was pulled from the image shown on the previous page.

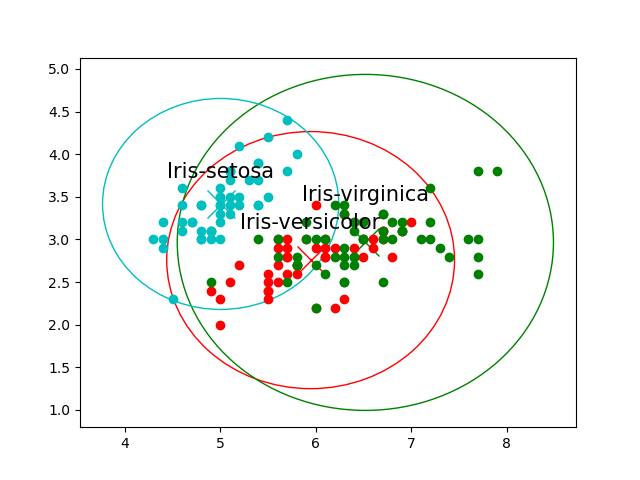
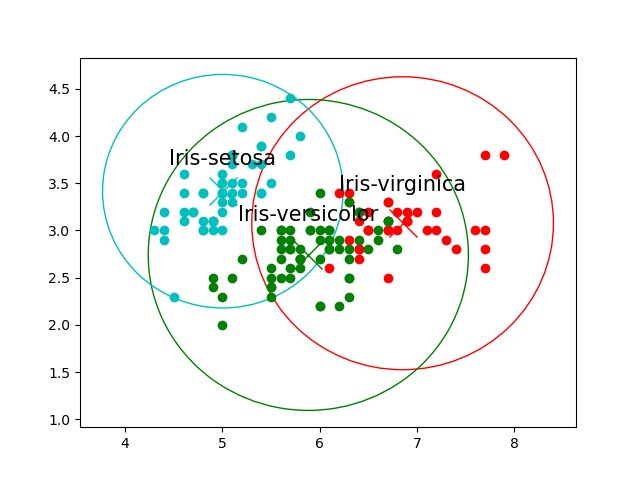
|  |  |
| --- | --- |
| **Distance Method** | **SSE** |
| Euclidean | 78.92303317291602 |
| Cosine | 93.83776128633266 |
| Jaccard | 79.1647170922698 |

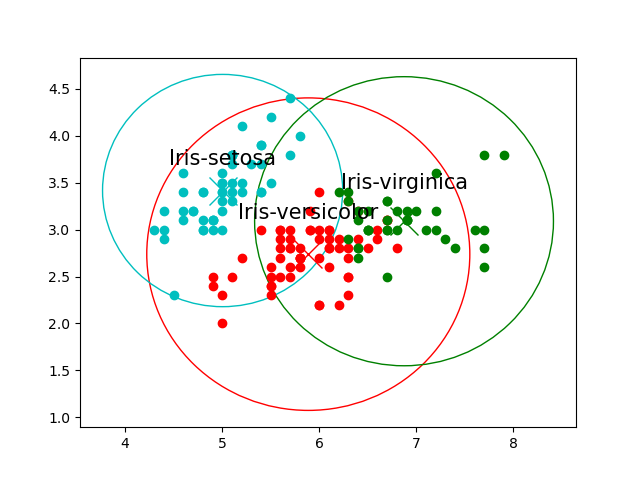
From this run, Euclidean performed best. Jaccard was only 1 higher and Cosine was 15 higher. The final SSE was heavily depended on the starting centroid. The starting centroids were determined based on the same function that was given in the kmeans.py.

**Q2: Compare the accuracies of Euclidean-K-means Cosine-K-means, Jarcard-K-means. First, label each cluster with the label of the highest votes. Later, compute the accuracy of the Kmeans with respect to the three similarity metrics. Which metric is better?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Distance Method** | **Cluster 1 Accuracy** | **Cluster 2 Accuracy** | **Cluster 3 Accuracy** |
| Euclidean | 0.9230769230769231 | 0.7704918032786885 | 1.0 |
| Cosine | 1.0 | 0.9090909090909091 | 1.0 |
| Jaccard | 0.7580645161290323 | 0.9210526315789473 | 1.0 |

Based on this run, Cosine performed the best. I noticed that depending on the starting centroids, the accuracies for both Euclidean and Cosine can be very bad. Jaccard seemed to be the most consistent over different runs.



Euclidean

Cosine

Jaccard

**Q3: Which of Euclidean-K-means, Cosine-K-means, Jarcard-K-means requires more iterations and times?**

|  |  |  |
| --- | --- | --- |
| **Distance Method** | **Time** | **Iterations** |
| Euclidean | 0.011002779006958008 | 6 |
| Cosine | 0.013004302978515625 | 5 |
| Jaccard | 0.020006418228149414 | 7 |

Based on this run, Euclidean completed in the fastest time, Cosine completed in the fewest iterations.

**Q4: Compare the SSEs of Euclidean-K-means Cosine-K-means, Jarcard-K-means with respect to the following three terminating conditions:**

**• when there is no change in centroid position**

**• when the SSE value increases in the next iteration**

**• when the maximum preset value (100) of iteration is complete**

**Which method requires more time or more iterations?**

**Table: SSE comparison**

|  |  |  |  |
| --- | --- | --- | --- |
| **Distance Method** | **Centroid Stopping** | **SSE Stopping** | **Iteration Stopping** |
| Euclidean | 78.92303317291602 | 78.92303317291602 | 78.92303317291602 |
| Cosine | 93.83776128633266 | 93.83776128633266 | 93.83776128633266 |
| Jaccard | 79.1647170922698 | 79.1647170922698 | 79.1647170922698 |

**Table: Euclidean Timing**

|  |  |  |
| --- | --- | --- |
| **Stopping Criteria** | **Iterations** | **Time** |
| Centroid | 6 | 0.009002685546875 |
| SSE | 6 | 0.010002851486206055 |
| Iteration | 100 | 0.15405011177062988 |

**Table: Cosine Timing**

|  |  |  |
| --- | --- | --- |
| **Distance Method** | **Iterations** | **Time** |
| Centroid | 5 | 0.012002706527709961 |
| SSE | 5 | 0.013004302978515625 |
| Iteration | 100 | 0.2440793514251709 |

**Table: Jaccard Timing**

|  |  |  |
| --- | --- | --- |
| **Distance Method** | **Iterations** | **Time** |
| Centroid | 7 | 0.018005847930908203 |
| SSE | 7 | 0.0180051326751709 |
| Iteration | 100 | 0.25208163261413574 |

Based on the above tables, regardless of the stopping criteria the final SSE was all the same. Centroid based and SSE based stopping criteria both resulted in the same number of iterations. Centroid based having slightly better times than SSE based.

Iteration based took the most time and iterations because the number was fixed. However, if the number of iterations required were to be above the fixed value, then the stopping criteria was stop early resulting in worse SSE.

**Task 3,** There are two clusters A (red) and B (blue), each has four members and plotted in Figure. The coordinates of each member are labeled in the figure. Compute the distance between two clusters using Euclidean distance.

****

**The distance were calculated with the equation Euclidean Distance = on an excel document. The tables group similar starting points together.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Point** | **Point** | **Distance** | **Point** | **Point** | **Distance** | **Point** | **Point** | **Distance** |
| **(4.7,3.2)** | **(4.9,3.1)** | **0.2236** | **(4.9,3.1)** | **(5.0,3.0)** | **0.1414** | **(5.0,3.0)** | **(4.6,2.9)** | **0.4123** |
| **(4.7,3.2)** | **(5.0,3.0)** | **0.3606** | **(4.9,3.1)** | **(4.6,2.9)** | **0.3606** | **(5.0,3.0)** | **(5.9,3.2)** | **0.9220** |
| **(4.7,3.2)** | **(4.6,2.9)** | **0.3162** | **(4.9,3.1)** | **(5.9,3.2)** | **1.0050** | **(5.0,3.0)** | **(6.0,3.0)** | **1.0000** |
| **(4.7,3.2)** | **(5.9,3.2)** | **1.2000** | **(4.9,3.1)** | **(6.0,3.0)** | **1.1045** | **(5.0,3.0)** | **(6.7,3.1)** | **1.7029** |
| **(4.7,3.2)** | **(6.0,3.0)** | **1.3153** | **(4.9,3.1)** | **(6.7,3.1)** | **1.8000** | **(5.0,3.0)** | **(6.2,2.8)** | **1.2166** |
| **(4.7,3.2)** | **(6.7,3.1)** | **2.0025** | **(4.9,3.1)** | **(6.2,2.8)** | **1.3342** |  |  |  |
| **(4.7,3.2)** | **(6.2,2.8)** | **1.5524** |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Point** | **Point** | **Distance** | **Point** | **Point** | **Distance** | **Point** | **Point** | **Distance** |
| **(4.6,2.9)** | **(5.9,3.2)** | **1.3342** | **(5.9,3.2)** | **(6.0,3.0)** | **0.2236** | **(6.0,3.0)** | **(6.7,3.1)** | **0.7071** |
| **(4.6,2.9)** | **(6.0,3.0)** | **1.4036** | **(5.9,3.2)** | **(6.7,3.1)** | **0.8062** | **(6.0,3.0)** | **(6.2,2.8)** | **0.2828** |
| **(4.6,2.9)** | **(6.7,3.1)** | **2.1095** | **(5.9,3.2)** | **(6.2,2.8)** | **0.5000** |  |  |  |
| **(4.6,2.9)** | **(6.2,2.8)** | **1.6031** |  |  |  | **(6.7,3.1)** | **(6.2,2.8)** | **0.5831** |

A. What is the distance between the two farthest members? (round to four decimal places here, and next 2 problems);

Using the tables from above, the farthest distance is 2.1095 between points (4.6,2.9) and (6.7,3.1).

B. What is the distance between the two closest members?

The closet distance is 0.1414 between points (4.9,3.1) and (5.0,3.0).

C. What is the average distance between all pairs?

Averaging everything in the table, the result is 0.9830.

D. Discuss which distance (A, B, C) is more robust to noises in this case?

Each case can be influenced by noise. The farthest and shortest will have the most influence if noise is one of the points as they will report a value that is not intended. The average will have the least influence from a given noise point. However, if there are too many extreme noise then the average will be greater influenced. Therefore, the average is the most robust to noise.