The code for this assignment is uploaded below :
https://github.com/nimaarvin83/CAP5610/blob/main/HW4/KMeans_With_Distance_and_Centroids.ipynb

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.

First we should know how to calculate Manhattan distance and Euclidean distance.
We are calculating these two as below:
Euclidean: Take the square root of the sum of the squares of the differences of the coordinates.
For example, if x=(a,b) and y=(c,d), the Euclidean distance between xy is

$$\text{Euclidean Distance between XY} = \sqrt{(a-c)^2 + (b-d)^2}$$

Manhattan: Take the sum of the absolute values of the differences of the coordinates.
For example, if x=(a,b) and y=(c,d), the Manhattan distance between xy and yy is

$$\text{Manhattan Distance between XY} = |a-c|+|b-d|$$

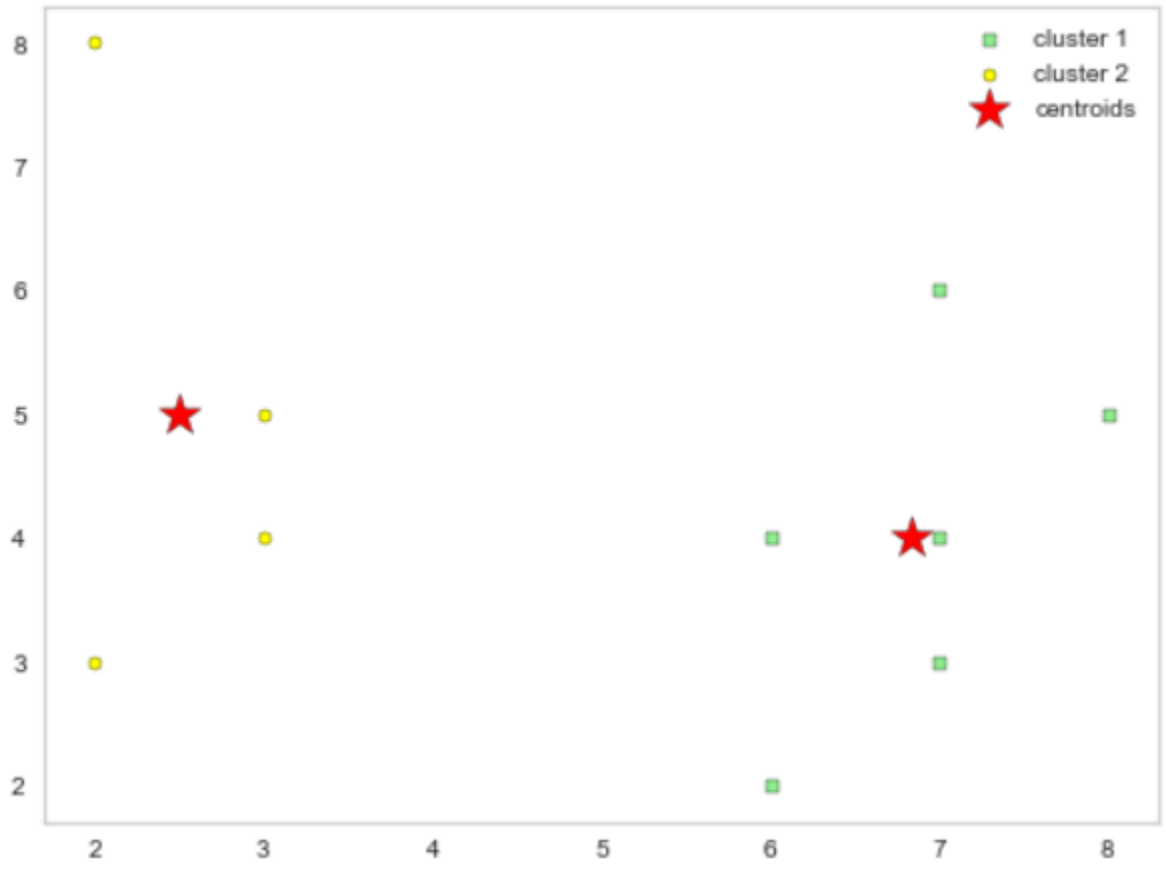
(1) Initialize with two centroids, (4, 6) and (5, 4). Use Manhattan distance as the distance metric. First, perform one iteration of the K-means algorithm and report the coordinates of the resulting centroids. Second, please use K-Means to find two clusters.

Below we calculated the Manhattan distance of each node from the center1 and center 2 and assign that node to its closest center. For calculating the new center we then got the average of the X coordinates of all the nodes that are close to the center 1 as X coordinate for the new center and got the average Y value of the nodes that are close to the center 1 for the Y coordinate for the new center and did the same thing for point close to center 2

New center 1 : (4, 6.33)
New center 2 : (5.571, 3.571)

Team	# wins in Season 2016 (x-axis)	# wins in Season 2017 (y-axis)	Center1 X axis	Center1 Y axis	Center 2 X axis	Center2 Y Axis	Manhattan Dist from Center 1	Manhattan Dist from Center 2	Picked Cluster	Cluster 1 New Center X	Cluster 1 New Center Y	Cluster 2 New Center X	Cluster 2 New Center Y
X1	3	5	4	6	5	4	2	3	Cluster 1				
X2	3	4	4	6	5	4	3	2	Cluster 2				
X3	2	8	4	6	5	4	4	7	Cluster 1				
X4	2	3	4	6	5	4	5	4	Cluster 2				
X5	6	2	4	6	5	4	6	3	Cluster 2				
X6	6	4	4	6	5	4	4	1	Cluster 2	4	6.33	5.571	3.571
X7	7	3	4	6	5	4	6	3	Cluster 2				
X8	7	4	4	6	5	4	5	2	Cluster 2				
X9	8	5	4	6	5	4	5	4	Cluster 2				
X10	7	6	4	6	5	4	3	4	Cluster 1				

Running the KMeans with initial centers of (4, 6) and (5, 4) and distance metric of Manhattan results in:
Two Clusters of [4, 5, 6, 7, 8, 9], [0, 1, 2, 3]
With centers of [6.833333333333333, 4.0], [2.5, 5.0]



(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.

Below we calculated the Euclidean distance of each node from the center1 and center 2 and assign that node to its closest center. For calculating the new center we then got the average of the X coordinates of all the nodes that are close to the center 1 as X coordinate for the new center and got the average Y value of the nodes that are close to the center 1 for the Y coordinate for the new center and did the same thing for point close to center 2

New center 1 : (2.5, 6.5)
New center 2 : (5.75, 3.875)

Team	# wins in Season 2016 (x-axis)	# wins in Season 2017 (y-axis)	Center1 X axis	Center1 Y axis	Center 2 X axis	Center2 Y Axis	Euclidean Dist from Center 1	Euclidean Dist from Center 2	Picked Cluster	Cluster 1 New Center X	Cluster 1 New Center Y	Cluster 2 New Center X	Cluster 2 New Center Y
X1	3	5	4	6	5	4	1.41421356	2.236068	Close to Center 1				
X2	3	4	4	6	5	4	2.23606798	2	Close to Center 2				
X3	2	8	4	6	5	4	2.82842712	5	Close to Center 1				
X4	2	3	4	6	5	4	3.60555128	3.1622777	Close to Center 2				
X5	6	2	4	6	5	4	4.47213595	2.236068	Close to Center 2	2.5	6.5	5.75	3.875
X6	6	4	4	6	5	4	2.82842712	1	Close to Center 2				
X7	7	3	4	6	5	4	4.24264069	2.236068	Close to Center 2				
X8	7	4	4	6	5	4	3.60555128	2	Close to Center 2				
X9	8	5	4	6	5	4	4.12310563	3.1622777	Close to Center 2				
X10	7	6	4	6	5	4	3	2.8284271	Close to Center 2				

Running the KMeans with initial centers of (4, 6) and (5, 4) and distance metric of Euclidean results in:
Two Clusters of [4, 5, 6, 7, 8, 9], [0, 1, 2, 3]
With centers of [6.833333333333333, 4.0], [2.5, 5.0]

(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.

Below we calculated the Manhattan distance of each node from the center1 (3,3) and center 2 (8,3) and assign that node to its closest center. For calculating the new center we then got the average of the X coordinates of all the nodes that are close to the center 1 as X coordinate for the new center and got the average Y value of the nodes that are close to the center 1 for the Y coordinate for the new center and did the same thing for point close to center 2

New center 1 : (2.5, 5)
New center 2 : (6.833, 4)

Team	# wins in Season 2016 (x-axis)	# wins in Season 2017 (y-axis)	Center1 X axis	Center1 Y axis	Center 2 X axis	Center2 Y Axis	Manhattan Dist from Center 1	Manhattan Dist from Center 2	Picked Cluster	Cluster 1 New Center X	Cluster 1 New Center Y	Cluster 2 New Center X	Cluster 2 New Center Y
X1	3	5	3	3	8	3	2	7	Close to Center 1				
X2	3	4	3	3	8	3	1	6	Close to Center 1				
X3	2	8	3	3	8	3	6	11	Close to Center 1				
X4	2	3	3	3	8	3	1	6	Close to Center 1				
X5	6	2	3	3	8	3	4	3	Close to Center 2	2.5	5	6.833	4
X6	6	4	3	3	8	3	4	3	Close to Center 2				
X7	7	3	3	3	8	3	4	1	Close to Center 2				
X8	7	4	3	3	8	3	5	2	Close to Center 2				
X9	8	5	3	3	8	3	7	2	Close to Center 2				
X10	7	6	3	3	8	3	7	4	Close to Center 2				

Running the KMeans with initial centers of (3, 3) and (8, 3) and distance metric of Manhattan results in:
Two Clusters of [4, 5, 6, 7, 8, 9], [0, 1, 2, 3]
With centers of [6.833333333333333, 4.0], [2.5, 5.0]

(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.

Below we calculated the Manhattan distance of each node from the center1 (3,2) and center 2 (4,8) and assign that node to its closest center. For calculating the new center we then got the average of the X coordinates of all the nodes that are close to the center 1 as X coordinate for the new center and got the average Y value of the nodes that are close to the center 1 for the Y coordinate for the new center and did the same thing for point close to center 2

New center 1 : (4.857, 3.571)
New center 2 : (5.667, 6.333)

Team	# wins in Season 2016 (x-axis)	# wins in Season 2017 (y-axis)	Center1 X axis	Center1 Y axis	Center 2 X axis	Center2 Y Axis	Manhattan Dist from Center 1	Manhattan Dist from Center 2	Picked Cluster	Cluster 1 New Center X	Cluster 1 New Center Y	Cluster 2 New Center X	Cluster 2 New Center Y
X1	3	5	3	2	4	8	3	4	Close to Center 1				
X2	3	4	3	2	4	8	2	5	Close to Center 1				
X3	2	8	3	2	4	8	7	2	Close to Center 2				
X4	2	3	3	2	4	8	2	7	Close to Center 1				
X5	6	2	3	2	4	8	3	8	Close to Center 1	4.857	3.571	5.667	6.333
X6	6	4	3	2	4	8	5	6	Close to Center 1				
X7	7	3	3	2	4	8	5	8	Close to Center 1				
X8	7	4	3	2	4	8	6	7	Close to Center 1				
X9	8	5	3	2	4	8	8	7	Close to Center 2				
X10	7	6	3	2	4	8	8	5	Close to Center 2				

Running the KMeans with initial centers of (3, 2) and (4, 8) and distance metric of Manhattan results in:
Two Clusters of [0, 1, 3, 4, 5, 6, 7], [2, 8, 9]
With centers of [4.85, 3.57], [5.67, 6.33]

Task 2 K-Means Clustering with Real World Dataset

First, download a simulated dataset: hw4_kmeans_data.zip from Modules->Datsets. Then, Implement the K-means algorithm **from scratch**. 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 (refer to: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/jaccard.htm).

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

Kmeans clustering algorithm is developed from scratch and ran with number of clusters=10
Kmeans ran using the Euclidean as distance method and SSE = 5600272
Kmeans ran using the Cosine as distance method and SSE=5612804
Kmeans ran using the Cosine as distance method and SSE=6679331

From the first look at the SSE factor we can conclude that the Euclidean metho d creates better clusters with lower Sum of Squared Errors

Q2: Compare the accuracies of Euclidean-K-means Cosine-K-means, Jarcard-K-means. First, label each cluster using the majority vote label of the data points in that cluster. Later, compute the predictive accuracy of Euclidean-K-means, Cosine-K-means, Jarcard-K-means. Which metric is better?

Kmeans ran using the Euclidean as distance method and used the majority vot ing to find the most voted cluster during the process and using that most voted resulted in increasing the SSE from 5600272 to 5606337. Looks like majority v oting in this method did not result in improvement.

Kmeans ran using the Cosine as distance method and used the majority voting to find the most voted cluster during the process and using that most voted resulted in increasing the SSE from 5612804 to 5649884. Looks like majority voting in this method did not result in improvement in this case also.

Kmeans ran using the Jaccard as distance method and used the majority votin g to find the most voted cluster during the process and using that most voted did not change the SSE the most voted identified the same clustering labels.

Q3: Set up the same stop criteria: "when there is no change in centroid position OR when the SSE value increases in the next iteration OR when the maximum preset value (e.g., 500, you can set the preset value by yourself) of iteration is complete", for Euclidean-K-means, Cosine-Kmeans, Jarcard-K-means. Which method requires more iterations and times to converge?

Euclidean K-Means

- when there is no change in centroid position → converged in 42 Iterations
- when the SSE value increases in the next iteration → converged in 42 iterations
- when the maximum preset value of iteration is complete → ran for n=300 and the sse didn't improve and centroids didn't change after the 42nd iteration

Cosine K-Means

- when there is no change in centroid position → converged in 85 Iterations
- when the SSE value increases in the next iteration → converged in 54 iterations
- when the maximum preset value of iteration is complete → ran for n=300 and the sse didn't improve and centroids didn't change after the 54th iteration

Jaccard K-Means

- when there is no change in centroid position → converged in 3 Iterations
- when the SSE value increases in the next iteration → converged in 3 iterations
- when the maximum preset value of iteration is complete → ran for n=300 and the sse didn't improve and centroids didn't change after the 3rd iteration

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 (e.g., 100) of iteration is complete**

Euclidean-K-means

• when there is no change in centroid position → SSE = 5600272
• when the SSE value increases in the next iteration → SSE = 5600272
• when the maximum preset value (e.g., 100) of iteration is complete → SSE=5600272
SSE didn't change after converging in all three methods

Cosine-K-means

• when there is no change in centroid position → SSE = 5612804
• when the SSE value increases in the next iteration → SSE=5611334

• when the maximum preset value (e.g., 100) of iteration is complete → SSE=5612804

Jarcard-K-means

• when there is no change in centroid position → SSE = 6679331
• when the SSE value increases in the next iteration → SSE=6679331
• when the maximum preset value (e.g., 100) of iteration is complete →SSE = 6679331

Q5: What are your summary observations or takeaways based on your algorithmic analysis?

The Euclidean Kmeans method resulted in having the lowest SSE among all methods.

Using different stop criteria's didn't result in significant changes in the SSE when the algorithm converged.

Using different distance methods result in significant difference in the way that kmeans clusters.

Majority voting didn't necessarily improve the clustering in this example.

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.

In order to calculate the distance between two clusters, first we need to calculate the center of each cluster and then calculate the distance between the centers.

The X coordinate of the red cluster center point is calculated by averaging the X coordinates of all red nodes and the Y coordinate of the red cluster is calculated by averaging all the Y coordinates of the red nodes.

X Coordinate of the Red cluster center = AVERAGE(4.7,4.9,5,4.6) = 4.8

Y Coordinate of the Red cluster center = AVERAGE(3.2,3.1,3,2.9) = 3.05

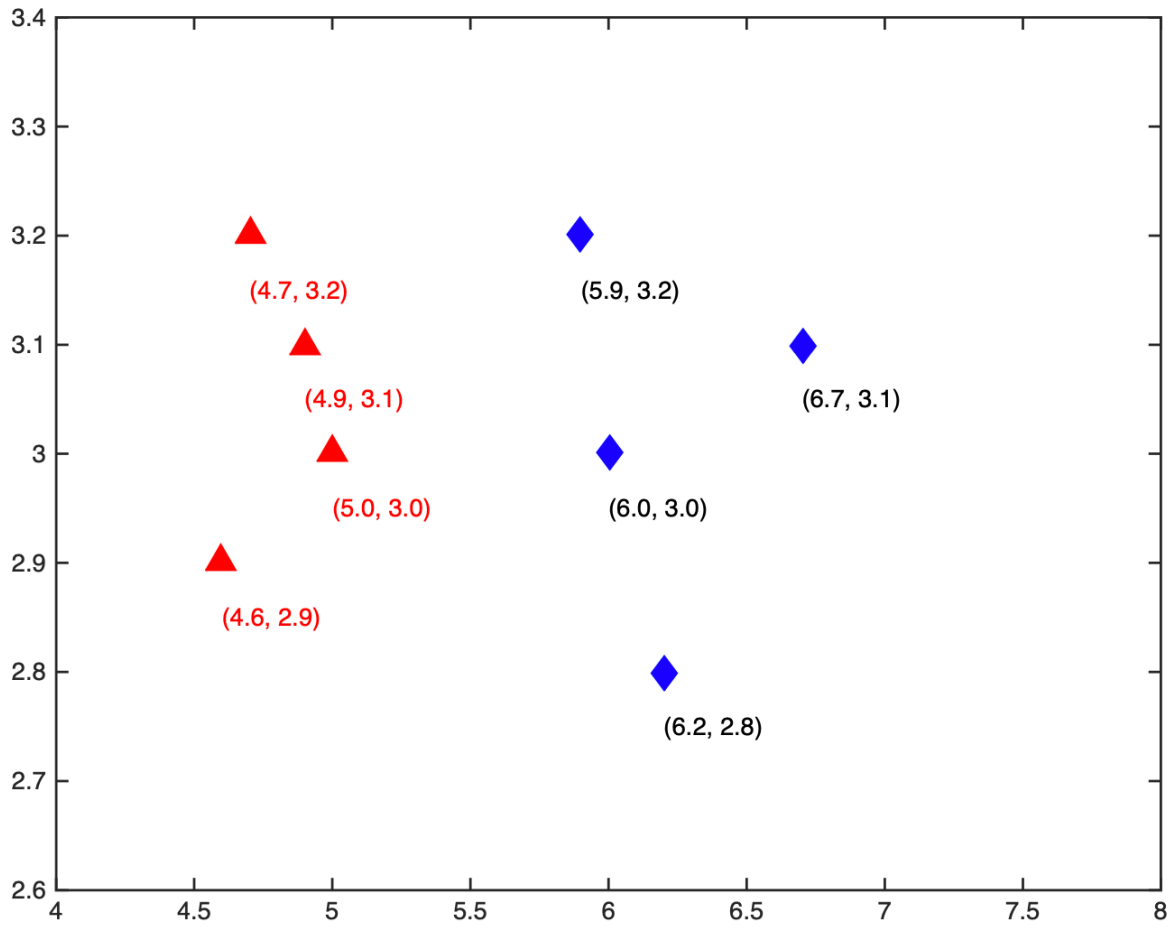
X Coordinate of the Blue cluster center = AVERAGE(5.9,6.7,6,6.2) = 6.2

Y Coordinate of the Blue cluster center = AVERAGE(3.2,3.1,3,2.8) = 3.025

Red center = (4.8,3.05)

Blue center = (6.2,3.025)

Euclidean distance of two clusters = $\sqrt{(6.2-4.8)^2+(3.05-3.025)^2} = 1.40022$



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

The farthest points are the two points of $(4.6, 2.9)$ and $(6.7, 3.1)$ and their Euclidean distance is $\sqrt{(6.7-4.6)^2+(3.1-2.9)^2} = 2.1095$

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

The closest points are the two points of $(4.9, 3.1)$ and $(5\ , 3)$ and their Euclidean distance is $\sqrt{(4.9-5)^2+(3.1-3)^2} = 0.1414$

C. What is the average distance between all pairs?

I calculated the distance between all pairs as below:

```
array([0.2236068 , 0.36055513, 0.31622777, 1.2       , 2.00249844,
       1.31529464, 1.55241747, 0.14142136, 0.36055513, 1.00498756,
       1.8       , 1.1045361 , 1.33416641, 0.41231056, 0.92195445,
       1.70293864, 1.        , 1.21655251, 1.33416641, 2.10950231,
       1.40356688, 1.60312195, 0.80622577, 0.2236068 , 0.5       ,
       0.70710678, 0.58309519, 0.28284271])
```

The average distance between all these pairs = 0.9829

D. Discuss which distance (A, B, C) is more robust to noises in this case?
The distance C is the most robust to the noises, because it is the average of all distances. The Distance A and B are very sensitive to the noises because if there is a random point added to the graph, the random point may completely change the closest points and the farthest points.

Additional Questions:

- **Approximately how many hours did you spend on this assignment?**

I spent about 10 hours in total on this assignment to develop my kmeans algorithm, write the answers and run the kmeans through different requested scenarios

- **Which aspects of this assignment did you find most challenging? Were there any significant stumbling blocks?**

The task 2 was the most challenging and took most of the time. The stumbling block was on calculating the distance using the Jaccard method. Since it was using the percentage from the ratio It couldn't divide to 10 clusters properly and it was assigning all the data points to a single cluster

- **Which aspects of this assignment did you like? Is there anything you would have changed?**

I liked the task 2 which was the most challenging one because it engaged heavy coding which I had to challenge with and overcome issues. Also task 1 was interesting because it gave me ground understanding to the kmeans algorithm.