Project Report

Student Id

1220905569

Strategy 1

Dataset Input and Size

```
3
[[7.39015357 1.13206806]
 [7.68097556 0.83542043]
 [2.81629029 3.1999725]]

5
[[7.52963009 8.79617112]
 [1.20162248 7.68639714]
 [7.1712312  5.16316266]
 [6.03237178 8.86195452]
 [6.63352332 0.98020705]]
```

300 – Data Points

Result Values

Final centroid for k1: [[6.49724962, 7.52297293]), [5.47740039, 2.25498103], [2.56146449, 6.08861338]]

Total Loss for k1: 1293.7774523911348

Final centroid for k2: [[7.75648325, 8.55668928]), [2.60123296, 6.91610506], [7.25262683, 2.40015826], [5.40252508, 6.73636175], [3.21257461, 2.49658087]]

Total Loss for k2: 613.2824392056041

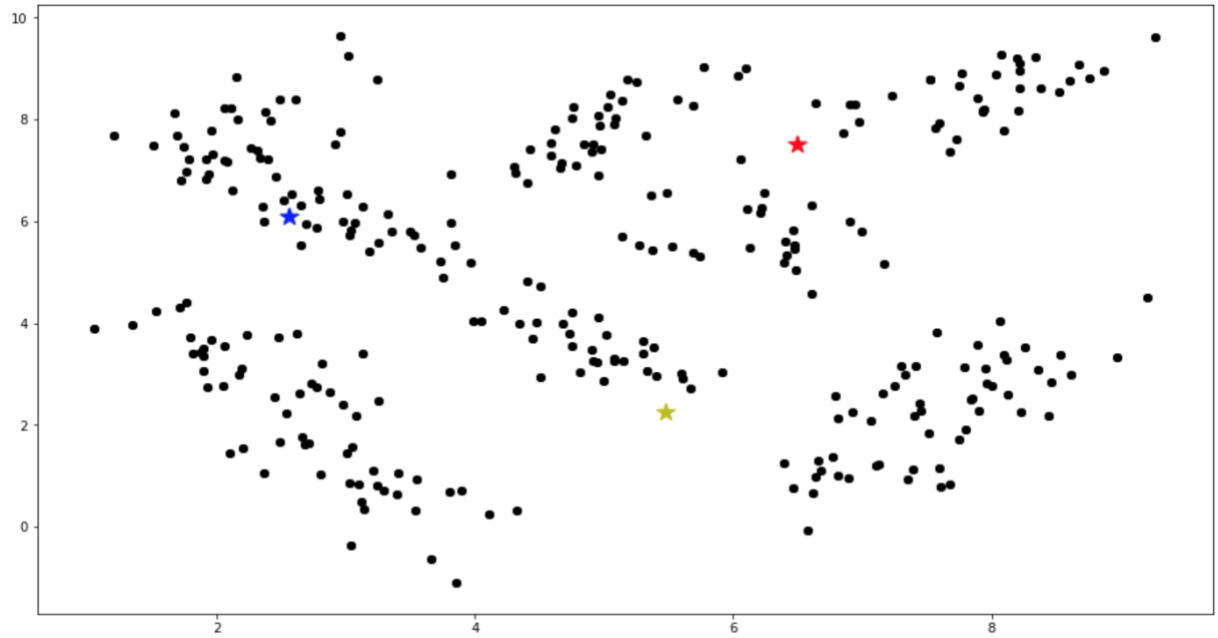

Fig 1. Strategy 1 K1 – Initial and Final Centroids

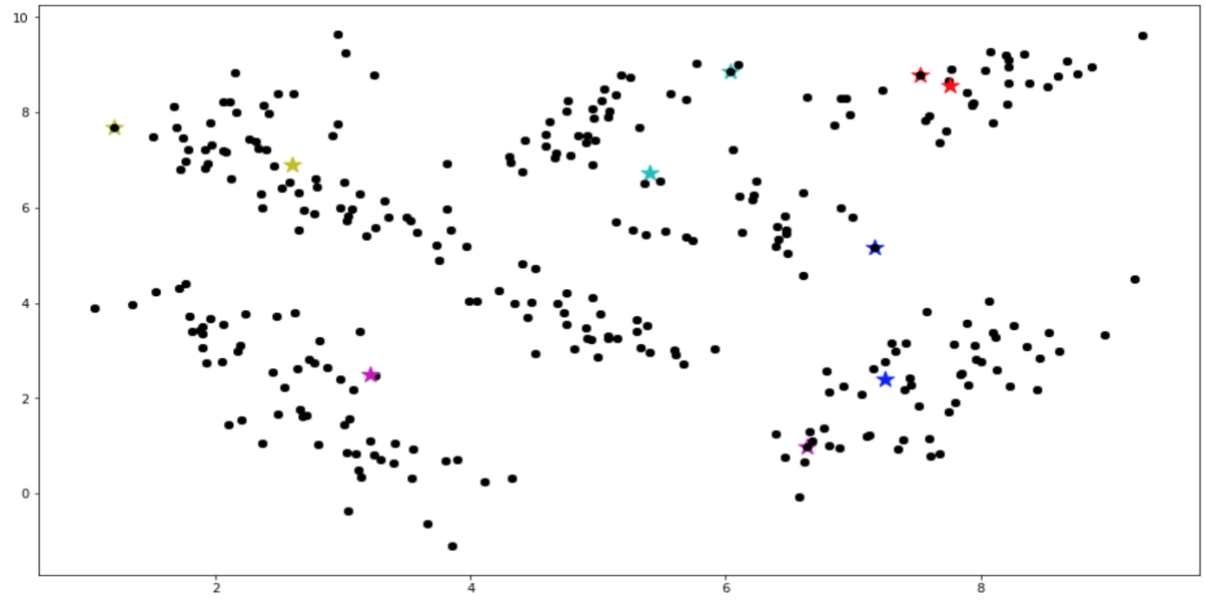

Fig 2. Strategy 1 K2 – Initial and Final Centroids

Strategy 2

Dataset Input and Size

```
4
[1.92561853 2.73857632]

6
[7.12751003 1.23747391]
```

300 – Data Points

Result Values

Final centroid for k1: [[3.339957483138508, 2.5921522375769444],
[6.603458393504191, 7.570421042158782],
[7.380762638700798, 2.332455315679148],
[2.8585923471789103, 6.931365250947319]]
Total Loss for k1: 788.2693490065566

Final centroid for k2: [[7.414192434680615, 2.3216911383868664],
[5.464277356727894, 6.837713536435891],
[7.756483249146484, 8.556689279063415],
[3.145061482959145, 0.9077065486588153],
[2.5633381461259046, 6.978224800606624],
[3.4955665791995627, 3.5661123157286907]]
Total Loss for k2: 476.11875167635293

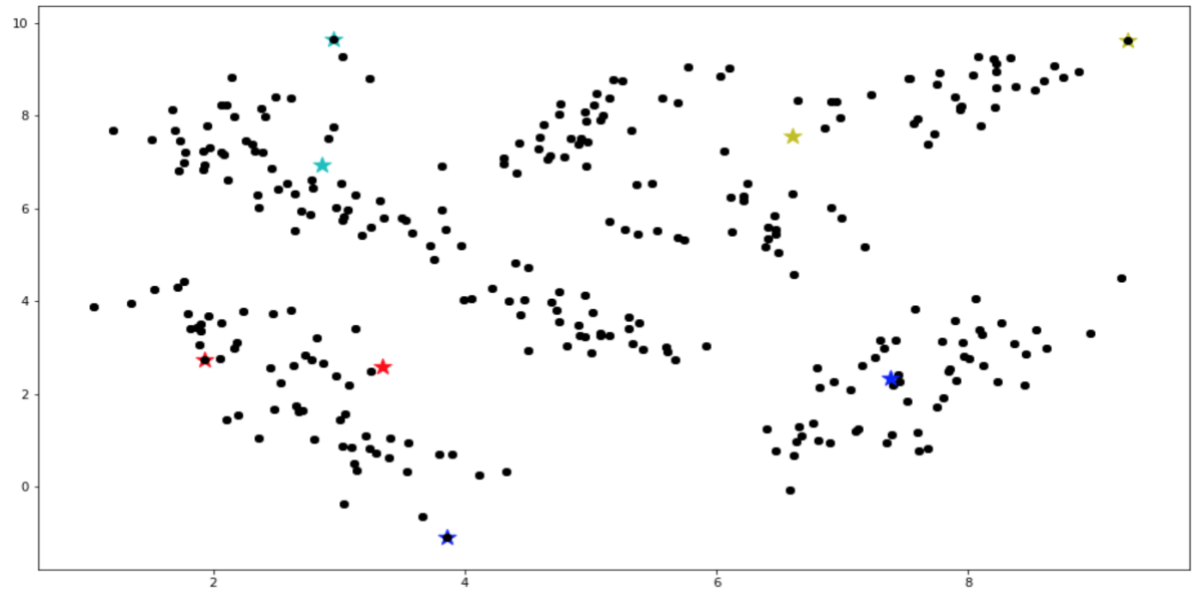

Fig 3. Strategy 2 K1 – Initial and Final Centroids

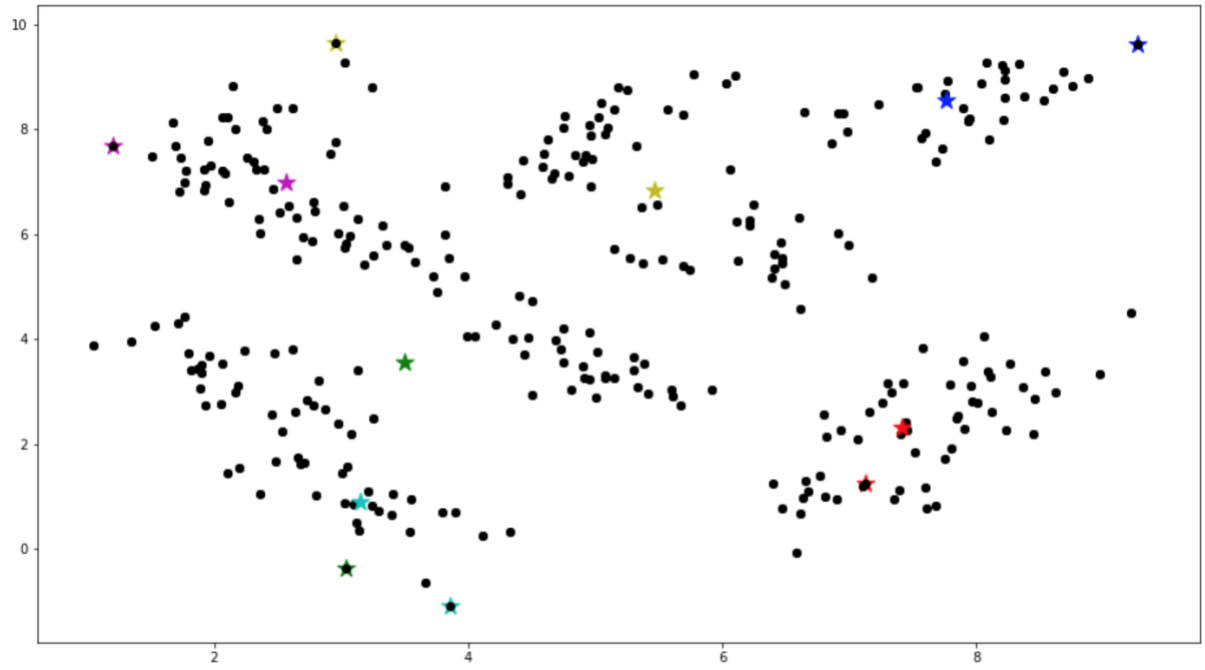

Fig 4. Strategy 2 K2 – Initial and Final Centroids

Approach

The projects that are done in CSE571 Artificial Intelligence and CSE572 Data Mining helped me to implement both NumPy and Dataframe approaches

I used a lot of print statements to check the data types and to understand the data.
Different numpy functions like "square", "sqrt", "sum" were helpful to achieve the tasks for this project.
I used matplotlib.pyplot to plot the points, initial and final centroids.

Observations/ Analysis

The data consists of an array of 2D points. I implemented the approach using both numpy and pandas dataframe separately.

I implemented a few common functions for this project –

- get_distance_from_centroid – This method takes the input data and selected centroids as the input, calculates the distance of each point from the selected centroids using Euclidean distance formula and returns 2 objects –
  - centroid_distance – An array of dictionaries that contains the distance of the point to the selected centroids and the last element is the average of all these distances. The length of this array is the same as data points as it contains the calculation for each data point
  - centroid_assignment – An array of the assigned centroid for each of the data points

- recalculate_centroids – This method takes the input data, selected centroids and centroid assignment for each data point as the input. It calculates the average of all the points assigned to a particular centroid and returns a new centroid based on the average.
- loss_function – This method takes the input data and selected centroids as the input, and calculates the loss for all the points from their centroids using the below formula
  $\sum_{i=1}^{k} \sum_{x \in D_i} ||x - \mu_i||^2))$
- printGraph – This method takes the input data and selected centroids as the input. It plots the data points and the centroids on the plot using pyplot library.
- create_centroids – This method takes the input data, k number of clusters value, and the initial centroids as the input. It iterates over k-1 times to calculate k-1 clusters. This method makes use of "get_distance_from_centroid" to calculate the distance of each point from all the centroids selected so far and take the mean of them. It then takes the point which has maximum mean and selects it as the new centroid. This method removes this newly selected centroid from the list of points to be used for the calculation of the next cluster.