**COMP 4331 Data Mining: Assignment 3**

Name: YEUNG, Man Yin Michael

SID: 20603418 / mymyeung

**Environment:**

OS: Windows 10 家用版

CPU: Intel Core i5-7200U

RAM: 8.00 GB

Model: ASUS UX410UQK

**Running Time:**

**K-Means Clustering:**

K = 3: 3 min 31 sec

K = 6: 3 min 38 sec

K = 9: 3 min 51 sec

**DBScan:**

= 5 and MinPoints = 10: 9 min 7 sec

= 5 and MinPoints = 4 : 9 min 42 sec

= 1 and MinPoints = 4 : 6 min 35 sec

**K-Means Clustering Report (2.1)**

**Method of initializing means:**

I think the best way to initialize means is to maximize the distance between them, because it can:

1. avoid bias to certain data

2. avoid having exactly same means

3. aviod “centriod of a cluster = initial mean(s)”

(which hinder the clustering progress, as there may be less clusters formed as the distances are equal)

From the reason above, I think the **ideal method** to initialize means is to find 5 data point that are furthest apart (find 5 data points such that the sum of combinational distances between them is maximized). However, this ideal method has a very high complexity and incurs an extremely significant run-time, which is not practical. Thus, I chose another method that is much less complex and can achieve a rather good result (close to the ideal method). The method is as follows:

**Chosen Method - Partition and take relatively far points, shown below:**

Even number means has:

x-coordinate = (max(x)-min(x) / (K - 1)

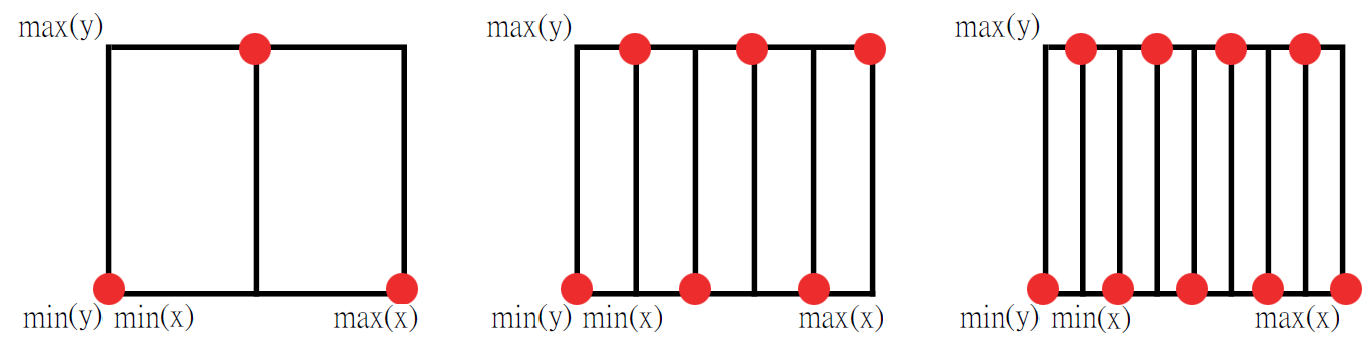
y-coordinate = min(y)

Odd number means has:

x-coordinate = (max(x)-min(x) / (K - 1)

y-coordinate = max(y)

Graphically:



In python codes:

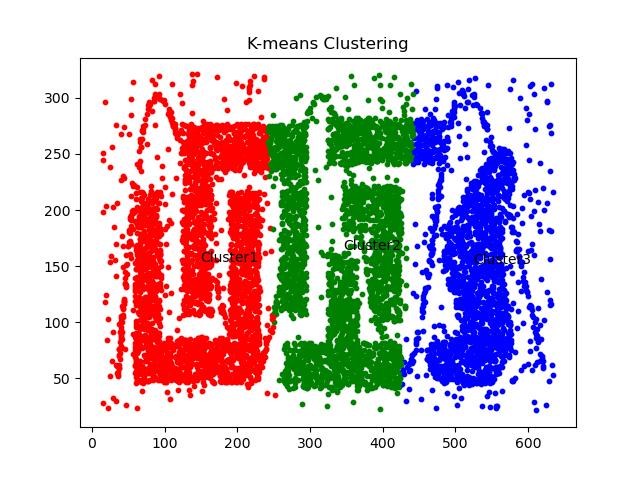
|  |
| --- |
| def init\_centriod():  centriods\_here = []  odd = 0  for i in range(0, K\_CONST):  if odd == 0:  y\_coor = MIN[1]  else:  y\_coor = MAX[1]  partition\_width = (MAX[0]-MIN[0]) / (K\_CONST - 1)  x\_coor = MIN[0] + i \* partition\_width  centriods\_here.append([x\_coor, y\_coor])  odd = 1 - odd # switch odd / even  return centriods\_here |

**Scatter Plots and Sum of Squared Error**

K = 3:

Sum of Squared Error = 64,736,348.67285633

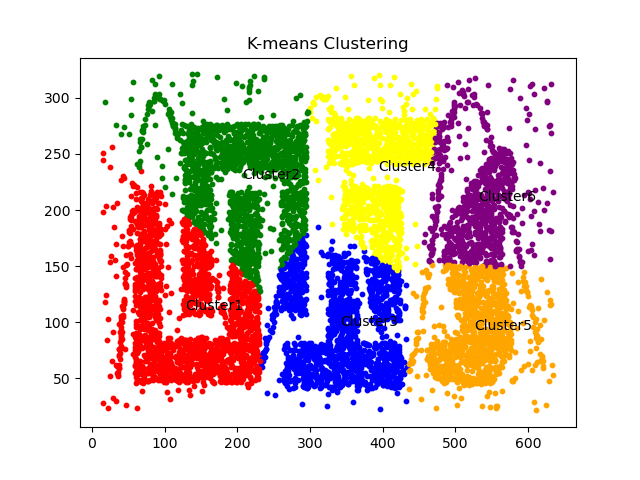
Scatter Plot:



K = 6:

Sum of Squared Error = 31,228,055.891633432

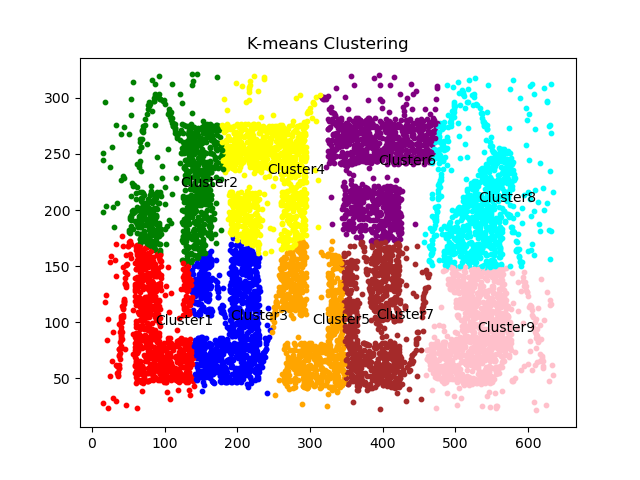
Scatter Plot:



K = 9:

Sum of Squared Error = 19,145,835.914759893

Scatter Plot:



**Best Available Choice**

I believe that K= 9 is the best available choice because of the following reason.

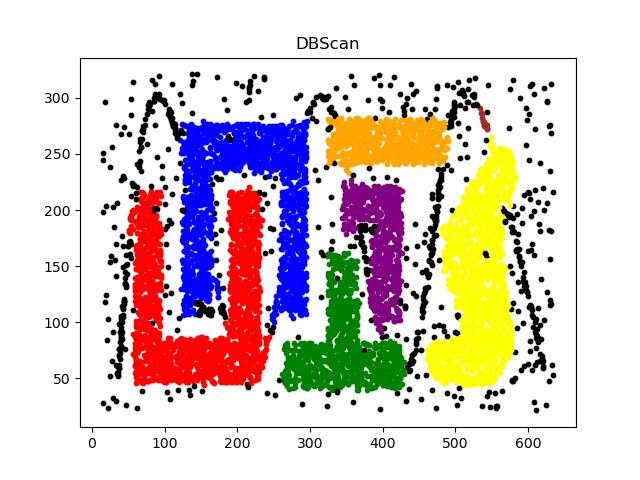
**It has the least Sum of Squared Error (SSE) with reasonable run-time.**

Having less sum of squared error meaning that we gain more information from the clustering method (i.e. we know more about the data). While at the same time, the runtime did not increase significantly (with not much extra time cost). We can also refer to the *Elbow method* [1] to strike a balance between SSE and running time. Viewing the graph below, the slope is still steep and I believe that K=9 have not reached the “elbow” (turning point) yet (or K=9 itself is the turning point), thus is the best available choice as it is the closest to the elbow (have a small SSE with reasonable run-time).

|  |
| --- |
|  |

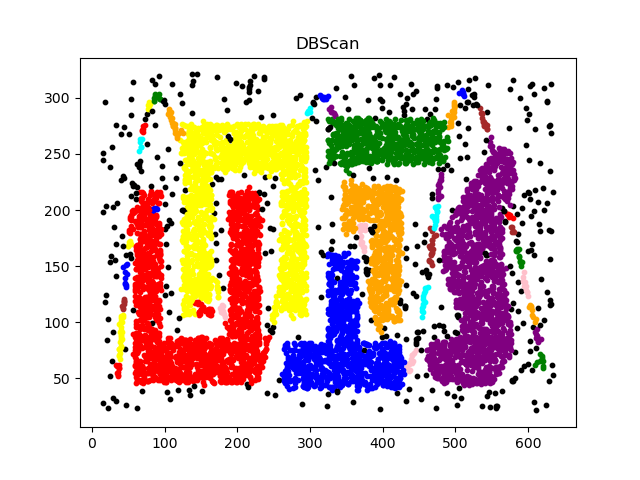
**DBScan Report (2.2)**

**= 5 and MinPoints = 10:**



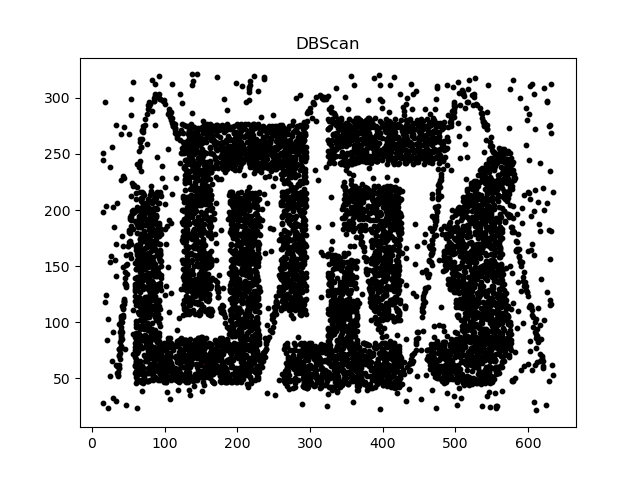
**= 5 and MinPoints = 4:**

*Note that: the nine colors are reused so there are different clusters using the same color.*



**= 1 and MinPoints = 4:**

*Note that: there is one cluster indicated in red (near (150, 60)) if you look at the graph very closely.*



**Best Available Choice**

I believe that  **= 5 and MinPoints = 10** is the best available choice because of the following reason:

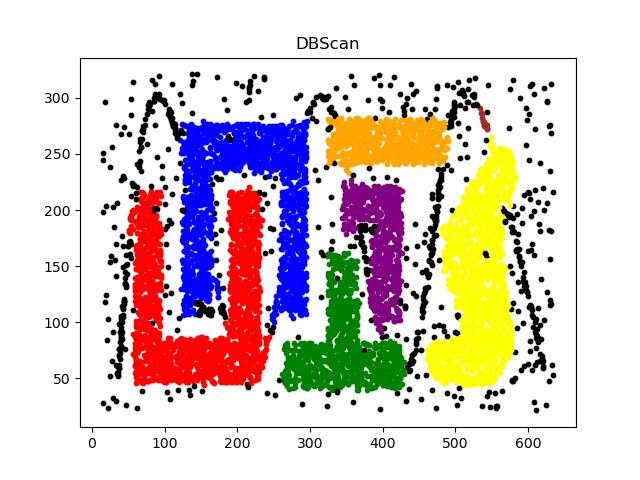
1. The scatter plot shows that it does not include “noise data points” as small clusters. It is not overly sensitive to noise data. We can see that in another scatter plot,  **= 5 and MinPoints = 4**, some noise data (colored) formed small clusters that we may not be interested in (the “sin-wave-shaped” **noise** in the background). However, if we consider including the “sin-wave-shaped” object as a **targeted pattern**,  **= 5 and MinPoints = 4**, will in turn become the best available choice as the pattern is included in the clusters. Only small MinPoints are able to recognize line-shaped or thin patterns / clusters. (I did not choose these parameters because I regard the wave as noise.)

2. The epsilon is not too small, in a sense that, (if it is too small) it does not fit the density behavior of clusters. For example,  **= 1 and MinPoints = 4** has an overly small epsilon so the majority of data points are regarded as “outliers” (indicated in black). Though the data points seem to form solid shapes / polygons, there are spaces between the data points. When epsilon (radius) is too small, the circle N(p) does not include (or include insufficient amount of) neighboring close data points, which hinders the construction of clusters (data points are not density-reachable to each other).

**Discussion (2.3)**

I believe that **DBScan** produced the best result for the given dataset.

**Superior Final Product (DBScan, = 5 and MinPoints = 10):**



It is clear that the “solid shapes” are indicated in different colors (clusters) and the noise data points (outliers) are not indicated in the clusters. In other scatter plots of DBScan, it reveals that either the “shapes” are not colored or some noise data points are included as clusters. In K-Means Clustering scatter plots, it is clear that the data points are not colored / clustered depending on the “shapes”, which is less desirable and less conclusive.

**Detailed Explanations:**

1. The data set contains a lot of noise data points, including randomly-scattered “background” data points and a “sin-wave shape”. DBScan is more robust to noise data and is able to spot them as “outliers (indicated in black)”. DBScan cares density thus may spot outliers, while K-Means Clustering does not care density thus includes outliers in the clusters. There are a large amount of outliers in the given dataset, thus DBScan is a better choice in this sense.

2. The dataset have close data points forming rather solid irregular shapes (non-globular), which can only be detected and recognized using DBScan. K-Means clustering is suitable for globular shapes only. As shown in the scatter plot, there are irregular (non-globular) shapes like “U-shaped”, “N-shaped” and “T-shaped” clusters etc, which makes DBScan more suitable in indentifying these “shapes” (clusters).

3. K-Means clustering tend to create clusters of the same size (radius equal half of displacement between two neighbor clusters), and which DBScan does not have this tendency. As shown in the scatter plot, the “shapes” (desired clusters) have sizes that do not seem similar. That’s why K-Means clustering is relatively not suitable for this dataset to use.

**Cases when K-Means Clustering perform better:**

1. The “shapes” of clusters are globular. As globular shapes meaning that data points are generally denser when it is closer (with small radius) to a certain point (centriod). This coincides with the concept that K-Means Clustering uses the distance (radius, epsilon) between data points and centriods to form clusters.

2. “Shapes” / potential clusters have similar sizes. As mentioned before, K-Means Clustering tends to create clusters of the same size. So, if the “shapes” / potential clusters have similar sizes, then K-Means Clustering method will “divide” them into clusters better (as there will be less difference between actual (data behavioral) cluster size and the equally-sized clusters (clustering implementation result).

3. There are few noises data points (actually this only reduces the disadvantage of K-Means but not a comparative advantage to DBScan, so two methods may become equal when no noise). As mentioned before, K-Means Clustering method includes all points in the clusters thus noise points are also included, while DBScan does not do so. Thus, if there is no noise data points, K-Means Clustering do not have a disadvantage in this regard.

**References:**

[1] K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. *Towards Data Science*

https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

**END OF REPORT**