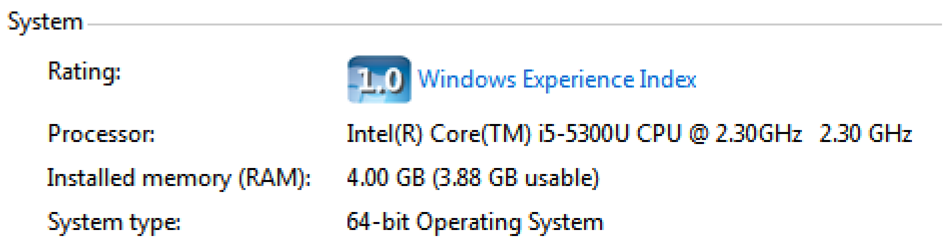
Assignment 3 Report

COMP 4331

PHAM Trung Kien

20553388

1. Environment



System

Rating:	1.0 Windows Experience Index
Processor:	Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz 2.30 GHz
Installed memory (RAM):	4.00 GB (3.88 GB usable)
System type:	64-bit Operating System

Besides, I use Python 3. Other than these, there are some libraries imported in the source codes such as Pandas, NumPy or Matplotlib

2. Result

1. K-Means Clustering

- Method of initializing the means: K-Means++ initialization algorithm. In details, K-means++ is just the standard K-Means but coupled with a smarter initialization of the centroids. The initialization process is as below:
    1. Randomly select the first centroid from the data points.
    2. For each data point compute its distance from the nearest, previously chosen centroid.
    3. Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its distance from the nearest, previously chosen centroid. (i.e. the point having

maximum distance from the nearest centroid is most likely to be selected next as a centroid)

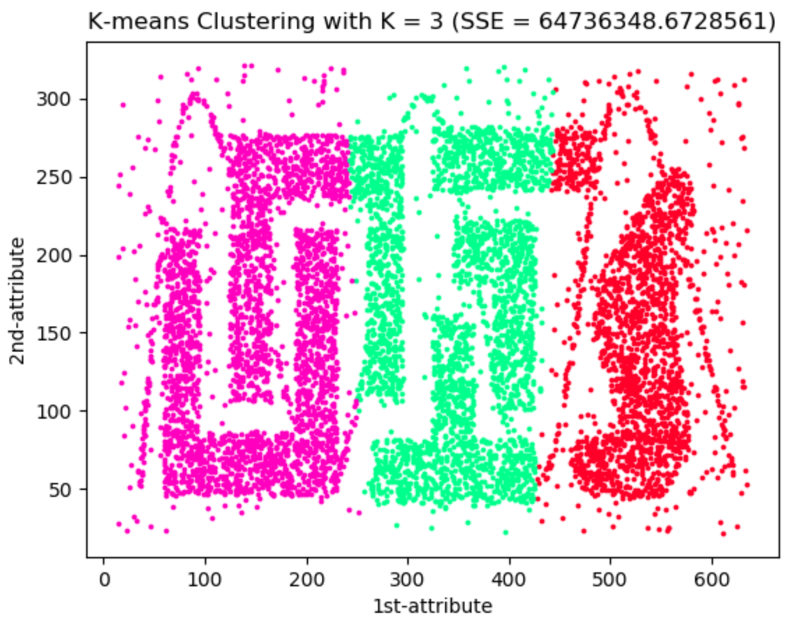
4. Repeat steps 2 and 3 until K centroids have been sampled

The reasons why I choose this method is that it helps initially pick up centroids which are far away from one another, increasing the probability that they lie in different clusters already. As a result, the run time for convergence to optimum can be reduced drastically (K-Means++ is tested to converge in just a few iterations and often twice as fast as the standard K-Means algorithm).

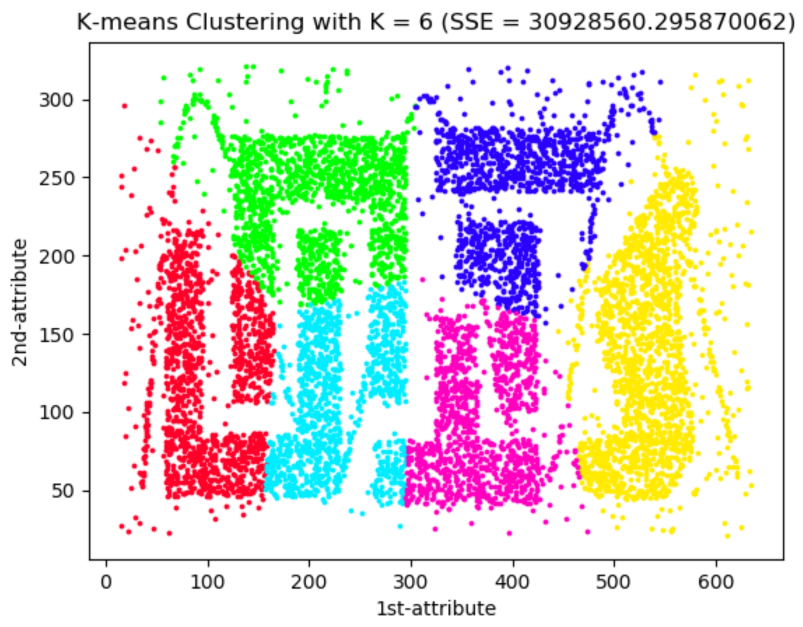
- Below are the scatter plots detailing the clusters as well as the Sum of Squared Error for:

  Note: Due to the fact that I use K-mean++ to initialize the means with first step of randomly selecting the first centroid from the data points, the clustering plots obtained for a K value may vary among different trials. These scatter plots below are the best chosen among many trials with lowest SSE score for each K value.

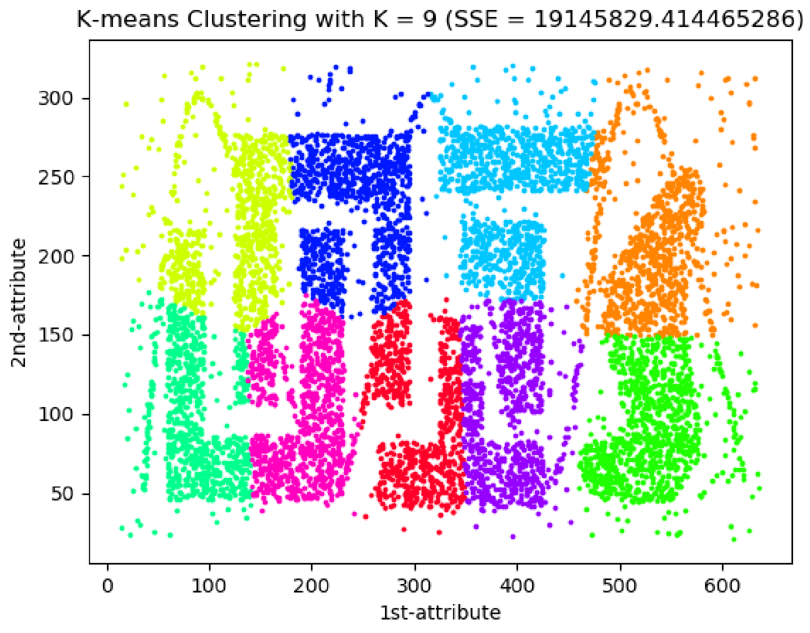
  - K = 3:



- K = 6:



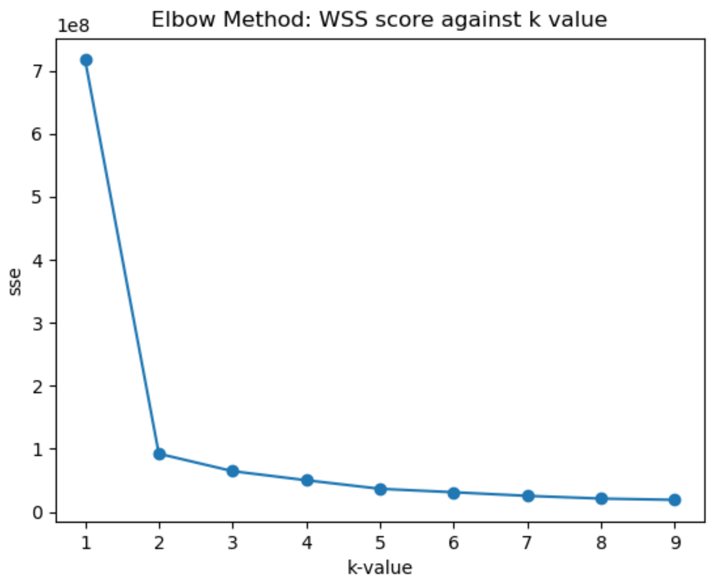
- K = 9:



- In order to decide which value of K is the best, I use the Elbow Method, which is one of the well-known methods for determining the optimal number of clusters.

  In details, after calculating the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, the best K is chosen for which WSS becomes first starts to diminish. In the plot of WSS-versus-K, this is visible as an elbow.

From the given dataset and my program, the aforementioned graph is provided below:

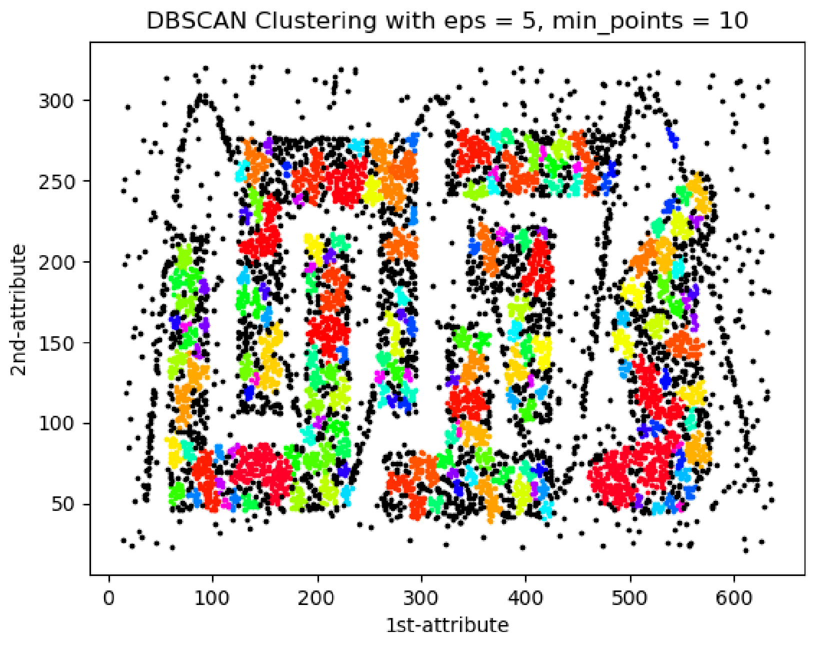


From the above graph, we can conclude that among K values of 3, 6 and 9, K = 3 is the best option. This conclusion is also supported by the fact that observing the above three clustering scatter plots, K = 3 generates the most appropriate good-looking clusters when comparing with the clumsy un-uniform distribution of points into clusters for K = 6 and 9.

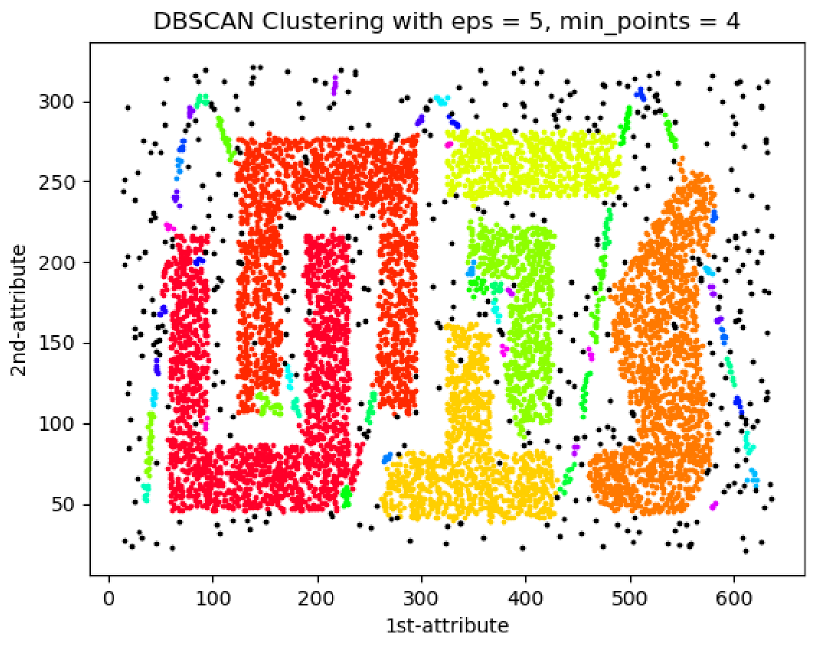
2. DBScan

- Below are the scatter plots detailing the clusters for:

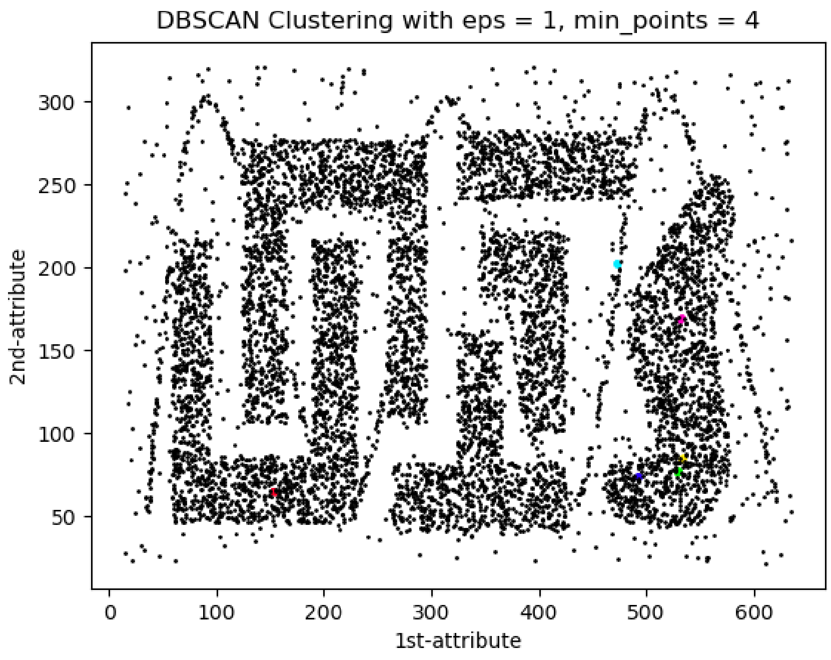
  - ε = 5 and MinPoints = 10



- ε = 5 and MinPoints = 4

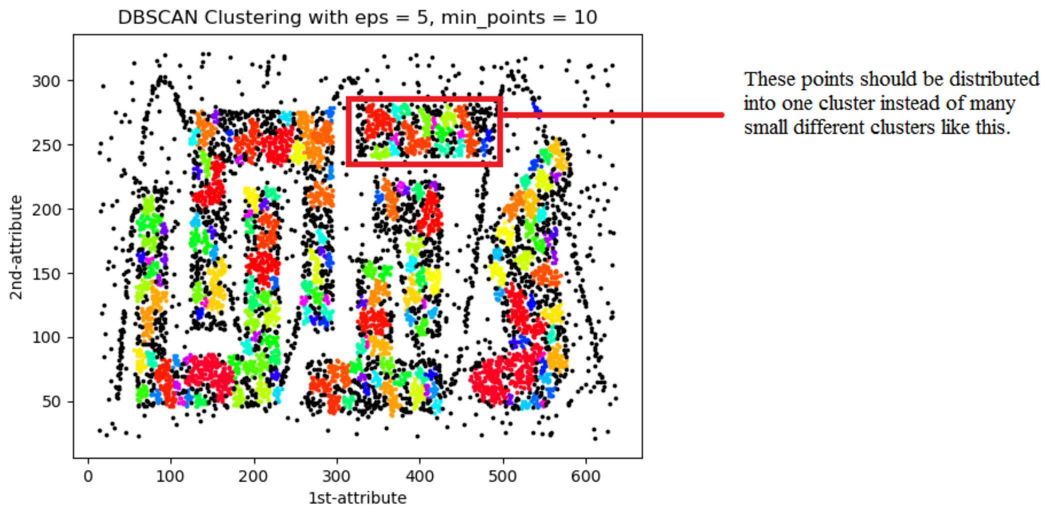


- ε = 1 and MinPoints = 4



- Having observed the above three scatter plots, it is persuasive that with parameters ε = 5 and MinPoints = 4, DBScan algorithm generates the best clustering result. The reasons are:

  - In the first scatter plot of ε = 5 and MinPoints = 10, many points which are supposed to be core points and density-reachable from others, were classified as border points or noise due to MinPoints parameter is too large. Consequently, many points that should belong to one cluster, were divided into many separately different small clusters instead. For instance:



- In the last scatter plot of ε = 1 and MinPoints = 4, because the value of ε parameter is too small, the majority of points (nearly every points) was considered as noise. As a result, the majority of points was not clustered and the scatter plot is covered by black points.

- In the second scatter plot of ε = 5 and MinPoints = 4, the values of ε and MinPoints parameters are both reasonable and supporting each other. As illustrated, six significant high-density clusters have been detected and the problems happened for the other two cases of parameters have been minimized.

3. Compare between K-Means and DBScan for given dataset

- I believe that DBScan, if the parameters are suitably chosen, will outperform K-Means algorithm in clustering the given dataset for these below reasons:

  - K-Means is highly sensitive to noise while DBScan is great with handling outliers (noise) within the dataset.

    - Observing the scatter plots in K-Means and DBScan, it is clear that the given dataset contains quite a lot of noise.
    - K-Means algorithm performs clustering for every point without differentiating which one is outlier. Consequently, it resulted in bad clustering with massive Sum or Squared Error just like in the K-Means scatter plots of given dataset mentioned above.
    - DBScan has the ability to detect which one is noise and exclude them during the clustering process. Therefore, it helps improve the correctness of the clustering result, highlighting the robustness of the algorithm itself.

  - DBScan can divide the data into different shapes and dimensions in order to find the best clusters while K-Means can only perform well when the data is distributed into pairwise linearly separable clusters. Having observed the dataset from those above scatter plots, we can affirm that there exists some large groups of data which can not be linearly separated to form suitable clusters. Therefore, DBScan will be a better method than K-Means to deal with this dataset.

- For K-Means algorithm, it may perform equal to or better than DBScan when:
  - The parameters chosen for DBScan are terrible, such as ε is too small (like when performing DBScan clustering for the given dataset with ε = 1 and MinPoints = 4, nearly every data points were not clustered). Meanwhile, the centroids for K-Means have to be appropriately initialized and the value for K is determined suitably. As a result, K-Means can perform equal to or better than DBScan.

  - The dataset is modified so that the number of noise is minimized and all the data points are arranged in a way such that they form different clusters and those clusters are pairwise linearly separable. In this situation, with proper initialization of means and proper choice of K value, K-Means algorithm outperforms DBScan in term of runtime and computational cost (faster in the runtime and less expensive computational cost).