CMPT 459 E100 Data Mining

Assignment 2

Report

Prof: Martin Ester

Student Name: Zeyong Jin

SFU ID: 301353174

Computing ID: zeyongj@sfu.ca

Date: March 15th, 2021

Contents

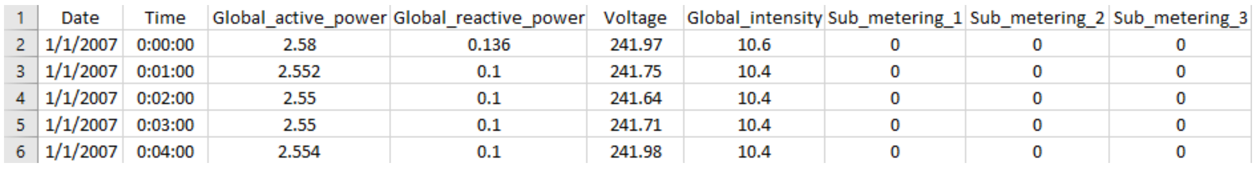
Task a. Preprocessing

For the preprocessing process, I first replaced all the values shown as "?" in the dataset with "np. nan". And then I used the dropna () function to drop all of these missing values. For the remaining values, I used the min-max scaling formula from the lecture note:

$$v' = \frac{v - min_a}{max_a - min_a}$$ [1]

After scaling, the data would be more centralized, which would improve the performance of density clustering.

Given the first several rows and features of the dataset are shown as follows:



1	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
2	1/1/2007	0:00:00	2.58	0.136	241.97	10.6	0	0	0
3	1/1/2007	0:01:00	2.552	0.1	241.75	10.4	0	0	0
4	1/1/2007	0:02:00	2.55	0.1	241.64	10.4	0	0	0
5	1/1/2007	0:03:00	2.55	0.1	241.71	10.4	0	0	0
6	1/1/2007	0:04:00	2.554	0.1	241.98	10.4	0	0	0

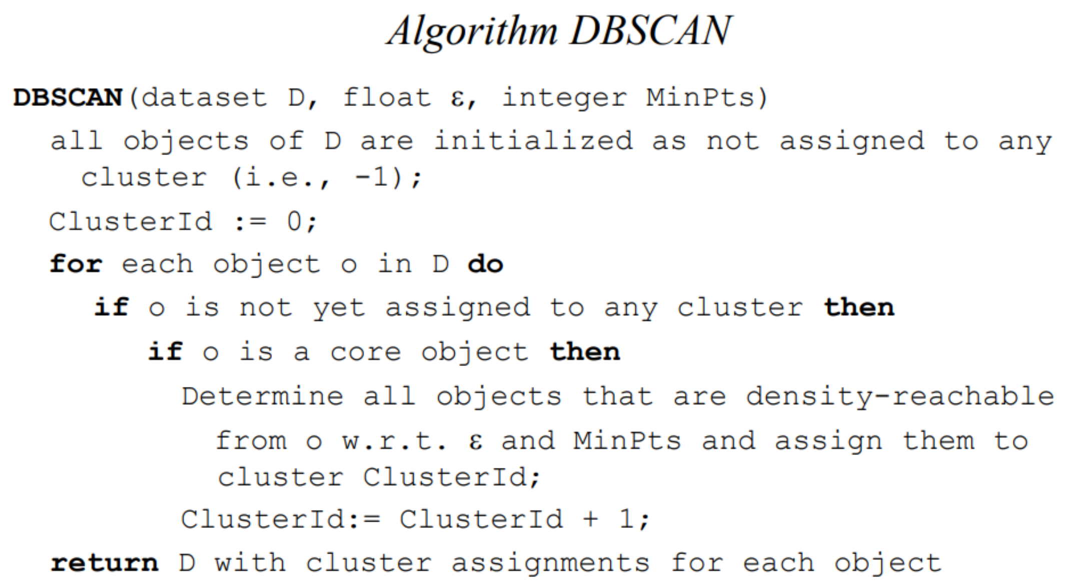
Figure 1

The first two features are date and time, which are all about a specific time. It is not so useful if dealing with these two columns via a function calculating the difference of time. Time shows a significant periodicity, meaning that if at the end the clustering could also represent the time's feature, then the clustering would be effective. And for the last three columns, all of them are sub-meters, whose values are usually 0. For the sake of convenience and to reduce the running time of the program, I only took the middle 4 features, which are "Global_active_power", "Global_reactive_power", "Voltage" and "Global_intensity", into further implementation in task b.

Also, as suggested by the teaching assistant, I only used the January data to complete all the three tasks in this assignment.

Task b. Implementation of DBSCAN

During the implementation, I used the pseudocode from the lecture note [2], to first defined a class named DBSCAN with different functions including fit (), and then initialized this class in the main function:



Algorithm DBSCAN

```
DBSCAN(dataset D, float ε, integer MinPts)
  all objects of D are initialized as not assigned to any
    cluster (i.e., -1);
  ClusterId := 0;
  for each object o in D do
    if o is not yet assigned to any cluster then
      if o is a core object then
        Determine all objects that are density-reachable
          from o w.r.t. ε and MinPts and assign them to
          cluster ClusterId;
        ClusterId:= ClusterId + 1;
  return D with cluster assignments for each object
```

Figure 2

The number of clusters, cluster labels' list and the frequency of each cluster label would be printed during the execution of the main function. And I also made the function automatically generate a text file to show cluster labels as required in the instruction.

The DBSCAN algorithm depends on the two parameters, which are epsilon and MinPts. Given the dimension of the processed dataset is 4, I first chose MinPts as 2×4, which is 8. But adapting this parameter, the running time was extremely long, and the result showed there would be a large number of clusters, which meant this clustering is bad. And the value of epsilon needed to be gained from the k-distance diagram.

So, combining the analysis in the next process, I chose epsilon as 0.030, MinPts as 90. And the result showed that there are 7 clusters (including noise), and each cluster has a relatively balanced frequency, indicating the clustering is good. And the running time was 2 hours and 28 minutes on a CSIL workstation with Intel Core i9-9900 and 64 GB RAM.

Task c. Testing Implementation on The Dataset

Part 1. Epsilon And K-Distance-Diagram

I did not use the heuristic approach in the class, in this task I used a method introduced by Tara Mullin on the website of Medium [3].

And I decided to find out the best value of epsilon first.

By importing the sklearn.neighbors package, I successfully draw out the k-distance-diagram.

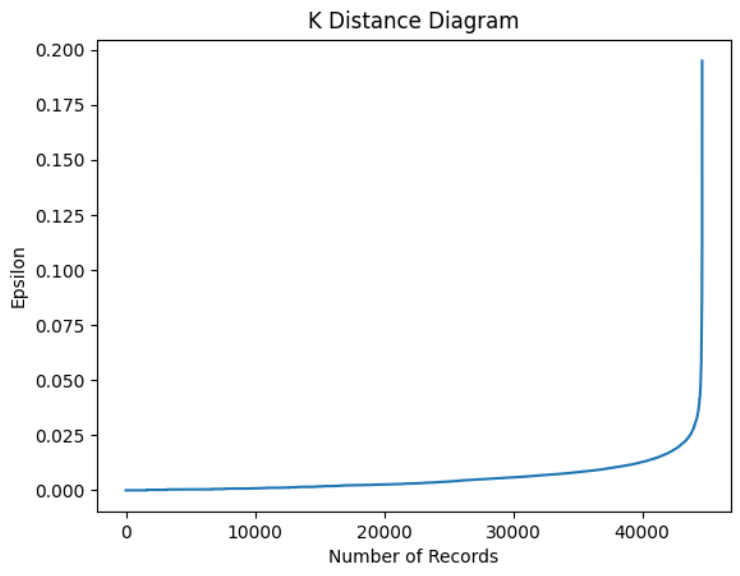


Figure 3

Clearly, the label lies in the interval of (0.025, 0.050). So, I zoomed in this part, and the result showed as follows:

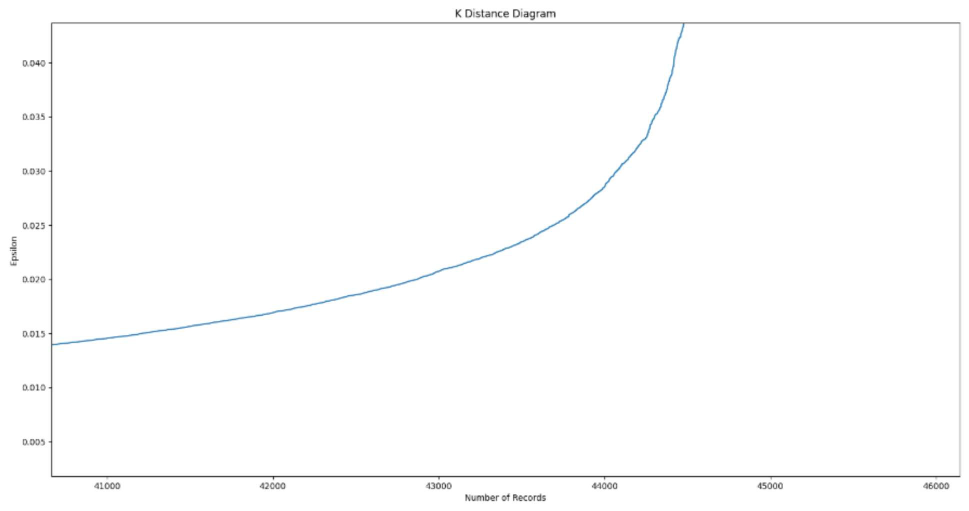


Figure 4

After the point of 0.030, the graph grows up rapidly. And therefore, I set the value of epsilon as 0.030.

Part 2. MinPts And Statistics

Given the dimension is 4 as mentioned in the previous task, I first chose the MinPts as 8. But it took an extremely long time to execute, and the result showed there are a number of clusters, meaning that the clustering using this combination of parameters is not good.

So, I tried to enlarge the value of MinPts. And I found that when MinPts equals 90, 100, 110, and the epsilon remains 0.030, the results of clustering are the following:

Epsilon	MinPts	Number of Noise	Number of Non-noise Clusters	Frequency of Each Cluster
0.03	90	15131	6	10250, 5545, 5413, 4902, 2111, 1286
0.03	100	15897	7	10155, 5473, 5383, 4603, 1781, 1200, 146
0.03	110	16670	6	10055, 5367, 5331, 4445, 1633, 1137

Figure 5

When MinPts equals 90 and 110, there are only 6 non-noise clusters. And when MinPts is 90, the number of noises is the smallest among these three values. Besides, when MinPts is 90, each cluster has a relatively balanced frequency. So, executing the DBSCAN with epsilon is 0.03 and MinPts is 90 is a better choice.

After executing the DBSCAN with epsilon is 0.03 and MinPts is 90, the running time is 2 hours and 28 minutes. After executing the DBSCAN with epsilon is 0.03 and MinPts is 110, the running time is 2 hours and 30 minutes. So, the running time does not change too much.

Combining all the analysis above, I decided to use the DBSCAN with epsilon is 0.03 and MinPts is 90, and drew a histogram of clusters as follows.

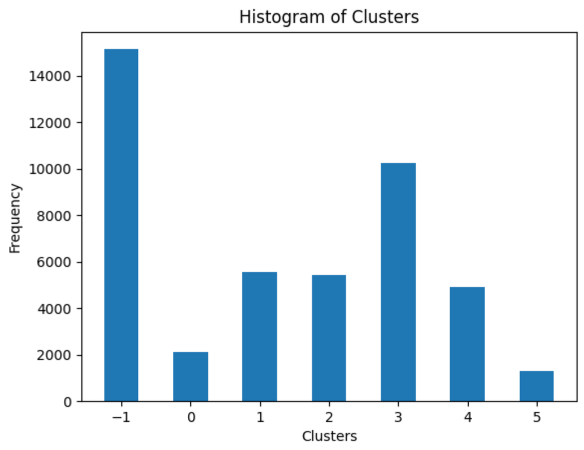


Figure 6

Including the noise, there are 7 clusters, namely -1, 0, 1, 2, 3, 4, 5. And the number of objects in each cluster is 15131, 2111, 5545, 5413, 10250, 4902, 1286 respectively.

Part 3. Discussion

The resulting clustering is good because of two reasons as follows.

A. The running time can be reduced to less than 2.5 hours when using Intel Core i9-9900 and 64 GB RAM.

B. The number of objects in each cluster is relatively balanced, and the number of clusters is reasonable.

C. The clustering can also represent the time's feature, meaning the clustering is effective.

But due to the limit of time and my knowledge, I was not able to write a function to find out the best combination of epsilon and MinPts. Perhaps there would be a better choice of these two parameters.

And the running time is still around 2.5 hours, showing this algorithm is time-consuming. Some further research needed to be made to improve the DBSCAN algorithm.

Task d. Acknowledgements

Part 1. References

[1] M. Ester, "CMPT 459 E100: Lecture Notes/Data Preprocessing," 21 January 2021. [Online]. Available: https://coursys.sfu.ca/2021sp-cmpt-459-e1/pages/DP. [Accessed 15 March 2021].

[2] M. Ester, "CMPT 459 E100: Lecture Notes/Cluster Analysis," 11 March 2021. [Online]. Available: https://coursys.sfu.ca/2021sp-cmpt-459-e1/pages/Clus. [Accessed 15 March 2021].

[3] T. Mullin, "DBSCAN Parameter Estimation Using Python," Medium, 2020 9 July. [Online]. Available: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd. [Accessed 2021 15 March].

Part 2. ReadMe

[1] When drawing the k-distance-diagram, I used sklearn library.

[2] The codes of drawing the k-distance-diagram are inspired and modified from the website: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd.

[3] The following libraries are needed for executing codes:

- numpy
- pandas
- math
- queue
- matplotlib
- sklearn # Only for task c.1.

[4] ***The name of the dataset file of Assignment is "houshold2007.csv", MISSING AN "r". But for the consideration of consistency, when the program was reading the file, I still set the name of the file to be read as "data_file = "houshold2007.csv"".***