NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4032 Data Analytics and Mining

Project 2

Group 45

Group Members:

Lee Cheng Han (U1920206L)
Ryan Phuay Qian Hao (U1922240H)
Koh Yi Joshua (U1922368F)
Nur Lydia Afiqah Binte Rozali (U1922339C)

1. Abstract

This paper will take a look into the usage of different clusters, namely the KMeans and DBSCAN methods for the clustering of two main datasets, shopping mall customers and their spending scores, as well as various statistics for different countries. It will take into consideration the various pros and cons of each method as well as their limitations into consideration to come up with a conclusion on the datasets. Here, we take reference to ROBERT KWIATKOWSKI and Tofti's works on KMeans, DBSCAN and AP. To further our explorations, we came up with various data analytic tools including, but not limited to histograms, scatterplots, as well as visualizations for each clustering technique.

2. Introduction

This project focuses on the task of clustering, comparing KMeans clustering and DBSCAN clustering on two datasets of firstly, shopping mall customers and their spending scores, as well as various statistics for different countries.

3. Related Work

Works used as reference to build our project include:

Python Implementation of k-means clustering
(https://github.com/tofti/python-kmeans)

Customers clustering: K-Means, DBSCAN and AP
(https://www.kaggle.com/datark1/customers-clustering-k-means-dbscan-and-ap)

4. Methods

One of the methods that we have used is KMeans clustering, The most well-known partitional clustering algorithm is K-Means. It was independently developed in many places in the 50s and 60s and gained great popularity because of its ease of implementation, simplicity and many empirical successes (e.g. in business, medicine and science).

There are 3 main steps in K-Means algorithm (known also as Lloyd's algorithm):

1. Split samples into initial groups by using seed points. The nearest samples to these seed point will create initial clusters.

2. Calculate samples distances to groups' central points (centroids) and assign the nearest samples to their cluster.

3. The third step is to calculate newly created (updated) cluster centroids.

Then repeat steps 2 and 3 until the algorithm converges.

Another method that we used is known as DBSCAN. The core idea of DBSCAN is around the concept of dense regions. The assumption is that natural clusters are composed of densely located points. This requires the definition of “dense region”. These two parameters are required to do DBSCAN algorithm.

- Eps, ε - distance
- MinPts – Minimum number of points within distance Eps

Optionally the distance metric can be specified by a user, but usually Euclidean distance is implemented (like in scikit learn).

A “dense region” is therefore created by a minimum number of points within distance between all of them, Eps. Points which are within this distance but not close to the minimum number of other points are treated as “border points”. Remaining ones are noise or outliers.

5. Experiments

Strengths and weaknesses analysis

KMeans	DBSCAN
Simpler to implement	
Scales well to larger datasets	Does not scale well to large datasets
Guarantees convergence	There is no guarantee of convergence
Number of clusters need to be determined	Domain knowledge may be required to determine the ideal Eps and MinPts, though cluster number is not needed
Results are very easily affected by noise	Robust to outlier and is able to filter out noise
Performs poorly on data with non-spherical shape	Performs well with arbitrary shape clusters
Is only able to operate based off the random centroids	Has a consistent statistical interpretation of density-connected components
Gets slow when number of dimensions increases	
If clusters are different in terms of varying densities or sizes, as only one K value is	If clusters are different in terms of varying densities or sizes, as only one Eps distance

passed in to the program, it will be unable to differentiate between these varied clusters and would be unable to properly generalize between these clusters.	and MinPts value is passed in to the program, it will be unable to differentiate between these varied clusters and would be unable to properly generalize between these clusters.

KMeans clustering:

Pros

-Simplicity of implementation
-Scales well to larger datasets
-Guarantees convergence

Cons

-Need to determine the number of clusters
-Sensitivity to outliers
-Does not work well on data with non spherical shape

DBSCAN:

Pros

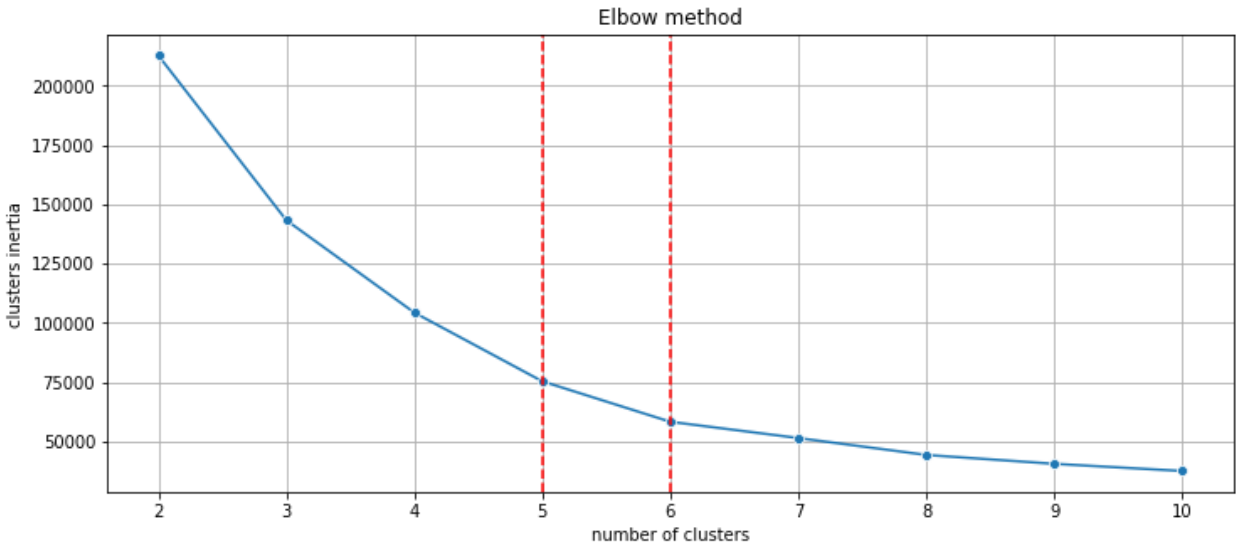
-Robust to outliers and able to detect the outliers
-Does not require a specified number of clusters beforehand
-Performs well with arbitrary shape clusters
-Has a consistent statistical interpretation of density-connected components

Cons

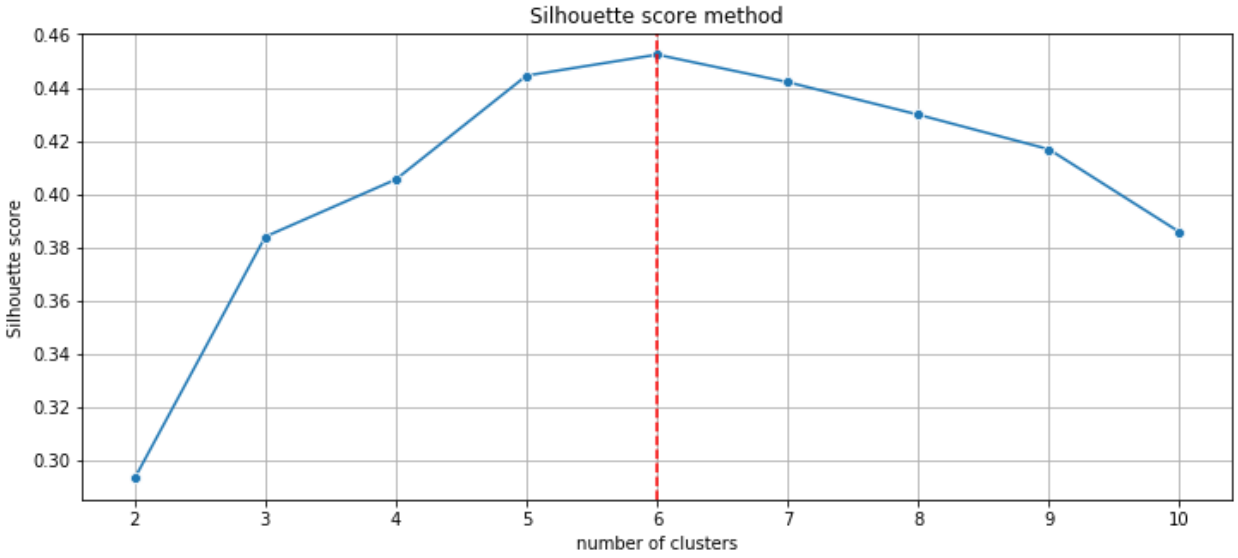
-Domain knowledge may be required to determine the ideal Eps
-If clusters are different in terms of varying densities or sizes, as only one Eps distance and MinPts value is passed in to the program, it will be unable to differentiate between these varied clusters and would be unable to properly generalize between these clusters.

Parameter settings

For our parameter settings, for K means, we made use of the elbow method to come up with the number of clusters needed, giving us the following results.



There is no clear “elbow” visible, which brings us to either a value of 5 or 6. Which leads us to seeing the silhouette score in order to evaluate the quality of the clusters, with the results as shown below.



This also continues telling us that either a K value of 5 or 6 would be the most ideal to use due to how close they are, which will be our main tests for the KMeans clustering.
We used a similar method to determine the Eps value and MinPts value, and the silhouette method ended up giving us values of Eps = 0.4 and MinPts = 7, as shown below.

```
eps 0.2
\min samples 8
clusters present: [-1  0]
clusters sizes: [156  11]
Silhouette Score: -0.2393615335452477

eps 0.3
\min samples 7
clusters present: [-1  0  1  2]
clusters sizes: [119  30  11   7]
Silhouette Score: -0.1695682589970686

eps 0.3
\min samples 8
clusters present: [-1  0  1]
clusters sizes: [130  29   8]
Silhouette Score: -0.15138644771886964

eps 0.4
\min samples 7
clusters present: [-1  0  1]
clusters sizes: [ 53 101  13]
Silhouette Score: 0.2581064152198133

eps 0.4
\min samples 8
clusters present: [-1  0  1  2]
clusters sizes: [70 78 11  8]
Silhouette Score: 0.11859148004718649
```

The best hyperparam are eps: 0.4 and min samples: 7, because it has the highest silhouette score, but samples is inlcuded with noise.

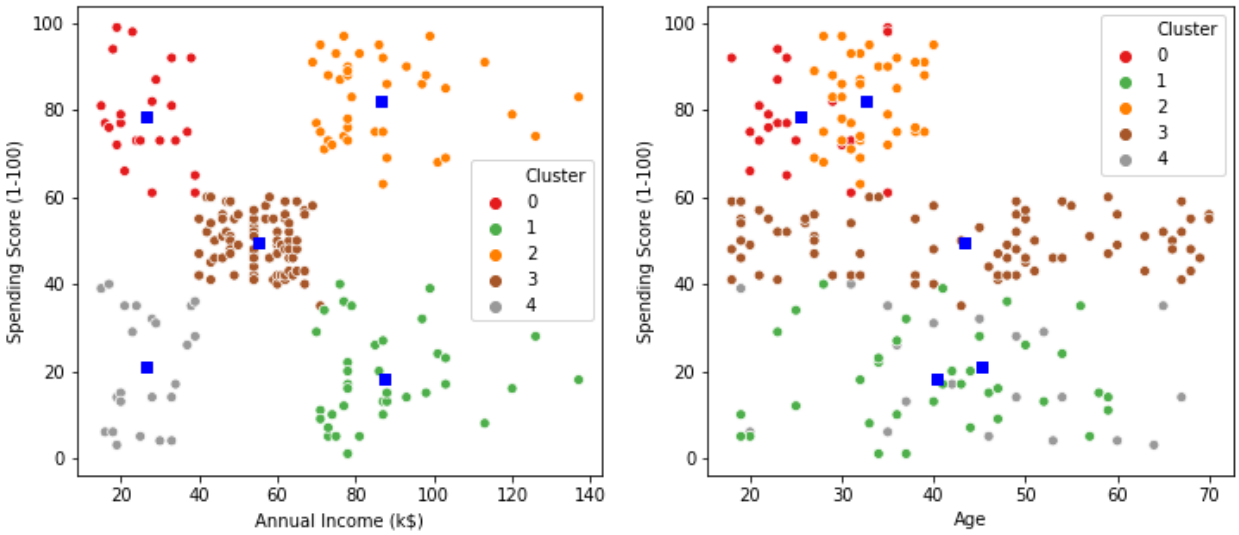
Performance

For KMeans clustering, the greatest factor which affects its performance involves the initialization of the centroids, the limitation of being unable to process natural shapes as well as its sensitivity to noise.

For DBSCAN, the above factors which affect KMeans clustering do not apply to it. However, it is faced with the issue of an appropriate Eps and MinPts value selection. In addition, it also faces the extra problem of if the clusters present are of varying densities or sizes, as only one Eps distance and MinPts value is passed in to the program, it will be unable to differentiate between these varied clusters and would be unable to properly generalize between these clusters. Note that a similar issue is faced by KMeans clustering as well.

As such, to further elaborate on these points, we will be referring to the experimental values which we have gained below, featuring KMeans clustering with 5 and 6 as their K values, as well as a DBSCAN with 5 clusters all on the same datasets.

K means clustering:
KMeans clustering with 5 clusters



K-Means algorithm generated the following 5 clusters:

-Clients with **low** annual income and **high** spending score

-Clients with **medium** annual income and **medium** spending score

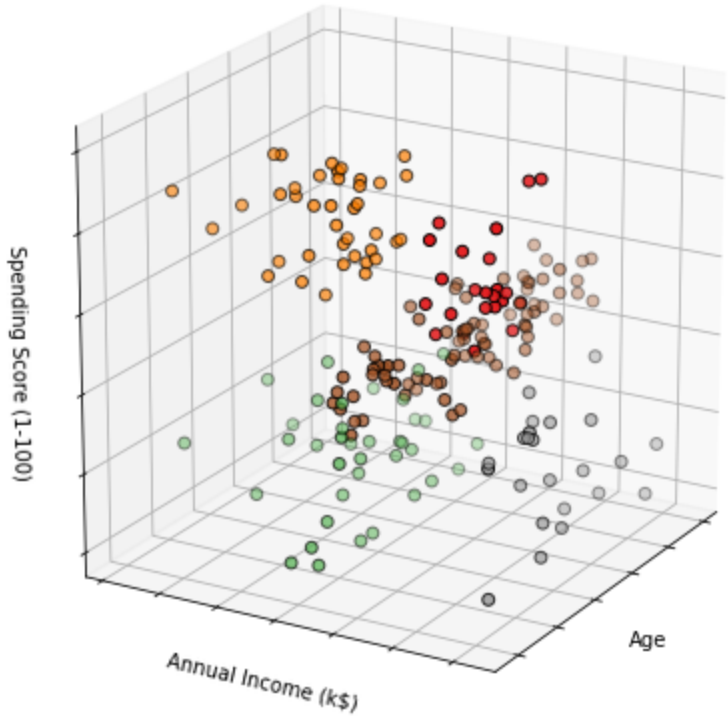
-Clients with **high** annual income and **low** spending score

-Clients with **high** annual income and **high** spending score

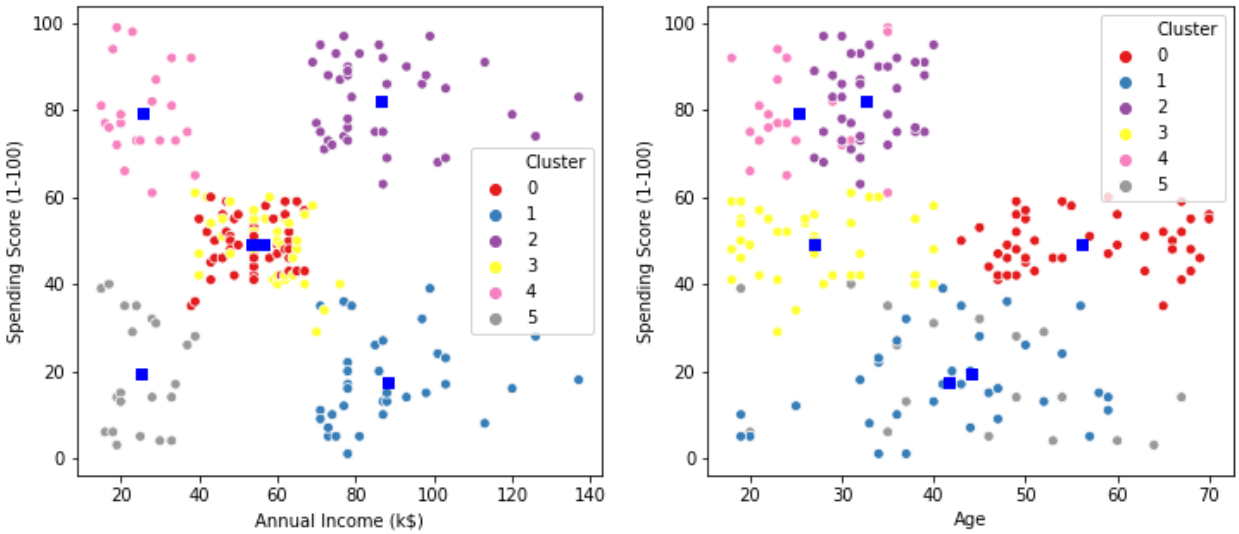
-Clients with **low** annual income and **low** spending score

There are no distinct groups in terms of the customers' age. The biggest cluster is cluster number 1 with 79 observations ("medium-medium" clients). There are two smallest ones, each containing 23 observations (cluster 3 "high-high" and cluster 0 "low-high" clients). Below, there is also a 3D projection of the 5 generated clusters. It is not very helpful in terms of visualisation in a static mode though, if the code is ran in an interactive environment (e.g. Spyder), it will be more useful in helping for visualizations.

3D view of K-Means 5 clusters



KMeans clustering with 6 clusters



K-Means algorithm generated the following 6 clusters:

-Younger clients with **medium** annual and **medium** spending score

-Clients with **high** annual income and **low** spending score

-Younger clients with **medium** annual and **medium** spending score

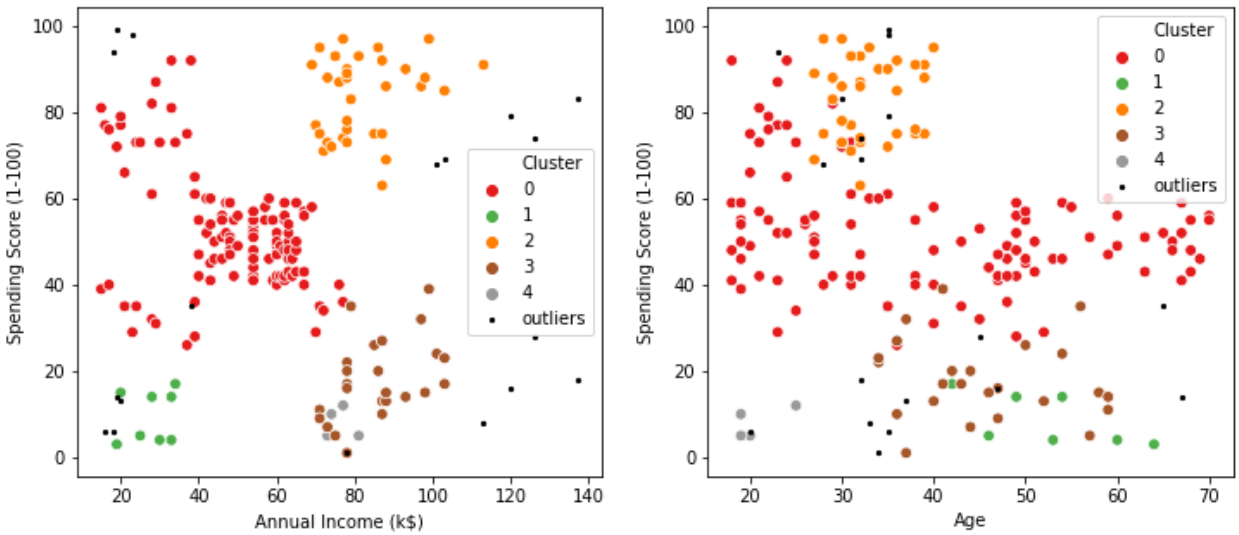
-Clients with **high** annual income and **high** spending score

-Clients with **low** annual income and **low** spending score

-Clients with **low** annual income and **high** spending score

There are no distinct groups in terms of the customers' age.

DBSCAN:



As we can see from the graphs displayed above, the KMeans clustering is able to provide decent results when it has 5 clusters on annual income. This is due to the initialized centroids being well spaced out. However, for the other three diagrams shown, we can see that certain initialized centroids are very close to one another which causes the clustering to be diminished, even when the elbow method was applied to find the ideal number of clusters to use. DBSCAN on the other hand does not face this problem. We are also able to see that the outliers may have dragged some of the clusters aside due to KMeans clustering being very sensitive to outliers, which worsens the results even further. DBSCAN on the other hand, is able to even show which points are determined as outliers and differentiate them from the main clusters, showing that it has a more consistent statistical interpretation of density-connected components due to its ability to filter out noise.

6. Conclusion

In essence, when considering clustering, there are many factors which need to be considered as well as preferably having some forms of domain knowledge. For instance, considering the pros and cons of KMeans and DBSCAN clustering methods, if we were more knowledgeable of the age/annual income spread in the clusterings, we would be able to select the more appropriate method of clustering. In the event of many outliers being present in the dataset, DBSCAN would outperform KMeans as DBSCAN is more robust in its ability to filter out noise. However, KMeans scales better to bigger datasets, due to its simplicity of implementation and the guarantee of convergence.