ABSTRACT

In the world of statistical learning, there exist mainly two ways of working with data: supervised and unsupervised learning. The aim of this work is to analyze the last one, with respect to a particular set of techniques, the “cluster analysis”, and more precisely to go deep with two particular algorithms, the “K-Means”, a simple yet powerful algorithm, and the “DBSCAN”, a proposed solution to some problems that can arise from the K-Means analysis.

After an introduction to the work, it is reported a description of the two clustering methods: the main idea and the precise algorithm, the pros and cons of each one. Then, it is carried out a brief exploratory data analysis using the two methods with the RStudio IDE and specific libraries, to show an applications of cluster analysis.

INTRODUCTION

Data are everywhere today, and learning from them is of critical importance if there is a purpose of data driven and quantitative choices, based on objective information and not on intuition without anything supporting it.

Statistical learning is the set of techniques with the aim of understanding and interpreting data, inferring from them and, finally, trying to make predictions.

There exists two big families in the statistical learning framework: supervised and unsupervised learning.

With the first, from the measures of the covariates, or predictors, and the variable of interest, the “response”, the so-called fitting of the model is performed: it relates the response to the predictors, with the aim of understanding the relationship between the response and the predictors (inference) or predicting the response for future observations (prediction) (Hastie et al., 2013).

Unsupervised methods rely on a different strategy: for every measure we observe a vector of covariates but no associated response. It is not possible to fit a supervised model, like linear regression, since there is no dependent variable to predict. It is called “unsupervised” because there is no response that can supervise the analysis (Hastie et al., 2013).

It is useful for understanding the possible relationships between the observations. One unsupervised technique widely used is clustering, that is very useful when there is no need for human analysis, and in applications such as extracting generative features, identifying meaningful trends and structures, groupings in results, and exploratory purposes (Sarker, 2021), like the so-called EDA (Explanatory Data Analysis), or for data mining, known also as Knowledge discovery in databases (KDD): an area where the purposes are pattern recognition, statistics, databases analysis, knowledge acquisition, data visualization, spatial data analysis (Ankerst et al., 1999), economic sciences like marketing or quantitative finance, etc.

Clustering is the process of creation of smaller data groups, called clusters, starting from the original data set, so that the elements belonging to one group are "similar" to each other and different from the elements in the other groups.

Thus, each cluster should represent a class, a group. The concept of similarity is complex and not easy to define. One way to think it is in terms of the distance between random variables (Durojaye et al., 2022). But it is not the only one. Another type of similarity is the so-called “density”: spatially, how much dense are some variables with respect to others.

In terms of distance, a simple yet powerful clustering algorithm, is the “k-Means”, while an example of cluster analysis that uses density is the “DBSCAN” (Density-Based Spatial Clustering of Applications with Noise), a method widely used in particular applications, such as noise or shape recognition and spatial applications (Ester et al., 1996).

K-MEANS

The K-Means clustering algorithm allows to group similar elements in a pre-specified number of clusters (that is “K”), and to discover underlying patterns.

More in detail, it partitions the observations into K clusters, (K, the number of clusters, is specified before by the user), and does it trying to minimize the total within-cluster variation, summed over all K clusters. This variation is a distance, and is typically represented by the sum over all K of the pairwise squared Euclidean distances between the observations in the kth cluster divided by the total number of observations in the same kth cluster (Hastie et al., 2013). This is the function minimized by the K-Means.

The algorithm that solves this problem works in this way:

1: Randomly assign a number, from 1 to K, to each of the observations: those will be the initial clusters.

2a: Find the centroids (the mean) for all points in each cluster.

2b: Measure the distance between each observation and the centroids, re-assigning each one to the nearest centroid, so form new clusters.

3: Iterate from 2, until the centroids don’t stop changing.

A graphical representation of the steps is given in figure 1

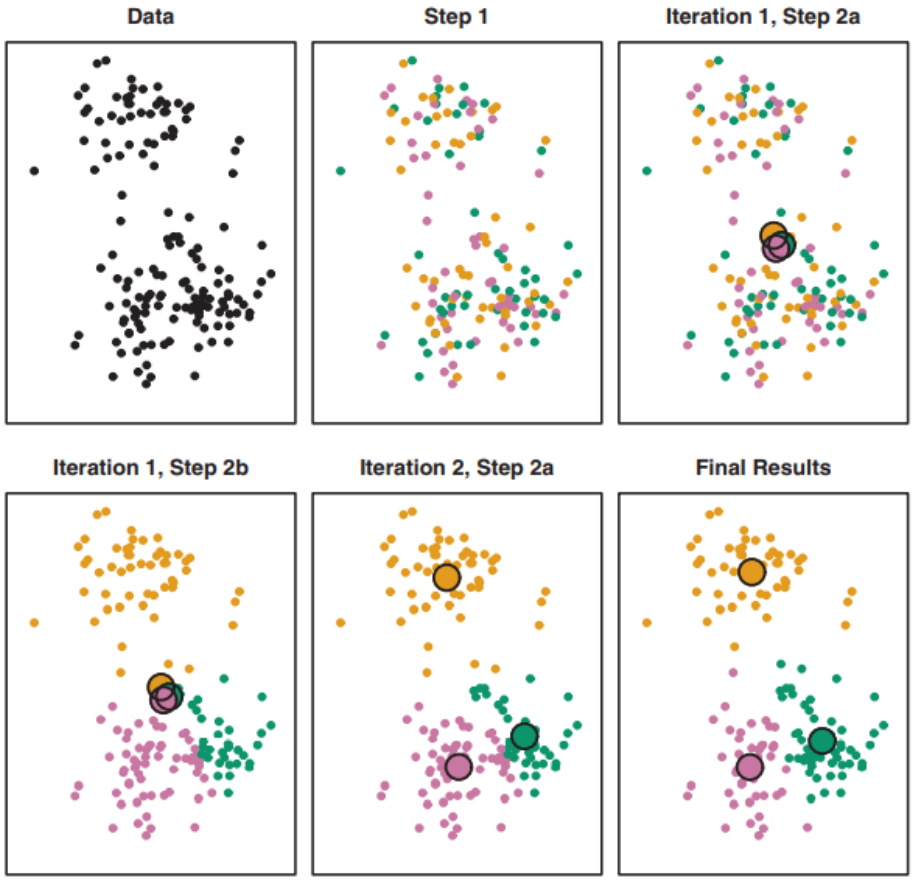
When the final step is reached, a local optimum is found. It is local because, given that the start was random, the initial assignment will inevitably influence the final result.

To choose among local optimums, it is common to repeat all the process several times, and compare the total variation between all the optimums, to find which is the least one, and then choosing it.

A problem is to decide the K: too much or too less number of clusters can drastically change the final result and interpretation.

There is no single answer: any solution that exposes some interesting aspects of the data should be considered (Hastie et al., 2013).

Figure 1: steps of the K-Means algorithm on an artificial database.



Source: An Introduction to Statistical Learning, 2013

One way is to rely on data knowledge, but it is rarely possible and not easy in any case.

Another solution is to use the Elbow Method.

It runs the K-Means for a range of values for K (usually from 1 to 10) and, for each one, it calculates the "Within Cluster Sum of Square" (WCSS), that is the sum of squared distance between each point and the centroid in a cluster.

The relation between WCSS (on the y-axis) against the number of K,is inverse: for an increase of K WCSS decreases, but not linearly. Starting from K=1, WCSS is at his max, and it initially decreases a lot but, at a certain K, the descending starts to be less strong: the best K is the one that is observed when the decrease starts lowering (Durojaye et al., 2022). This is because it should indicate that the observations starts to be near one another in each cluster and, adding more clusters, does not improve a lot the within variation.

Pros and cons:

- K-Means is simple, easily understandable, pretty fast and works better than other in particular data distributions, like well separated and convex shaped data.

- The number of clusters must be priorly selected; outliers can influence a lot the final result, hence, it does not work well if there is a lot of noise or if data follow a particular shape.

DBSCAN

The DBSCAN, Density-Based Spatial Clustering of Applications with Noise (Ester et al., 1996), is a density-based clustering algorithm. This means that it connects different regions of observations (or points) in the space spanned by observations, using as criterion the "density" that the points form in that space. The intuition is that, for each observation belonging to a cluster, the neighborhood of a given radius of the point has to contain at least a minimum number of observations (Hankerst et al., 1999; Ester et al., 2017; Ester et al. 1996); those points are said to be connected, while the isolated points are identified and classified as "noise", so that they don't belong to any cluster.

More precisely, the neighborhood is determined by the choice of a distance function D for two points p and q. It is possible to use any distance function but, generally, the Euclidean distance is the chosen one.

DBSCAN requires the user to choose two parameters before starting (Ester et al.,1996):

• Eps: the radius (talking in terms of Euclidean distance) that defines the Eps-neighborhood of a point p, NEps(p), that is the set of all q points belonging to D such that the distance between p and q is less than Eps;

• MinPts: a minimum number of points in an Eps-neighborhood of that point.

Secondly, letting p and q two points, the relation between those points can be defined as follow (Ester et al.,1996):

• Directly density-reachable: given Eps and MinPts, p is directly density-reachable from q if p belongs to NEps(q) and if NEps(q) is greater or equal to MinPts; this is the so-called "core object condition";

• Density-reachable: given Eps and MinPts, p is density-reachable from q if there is a chain of points $p_1, \dots, p_i, \dots, p_n$, $p_1 = q, p_n = p$, such that p_{i+1} is directly density-reachable from p_i;

• Density-connected: given Eps and MinPts, p is density-connected to q if there is a point o such that both p and q are density-reachable from o.

Thirdly, there exist three types of points:

• Core point: a point that satisfies the "core object condition", so a point where its Eps-Neighborhood, determined by Eps, contains at least MinPts number of points;

• Border point: a point that is not a core point but is directly density-reachable by a core point, so the border of the cluster that cannot be used to reach other points;

• Noise point: an outlier, not reachable by any point.

Finally, also the cluster must be properly defined.

The cluster C is a non-empty subset of a database of points D with respect to Eps and MinPts if, for all p, q, satisfies those two conditions (Ester et al.,1996):

1. If p belongs C and q is density-reachable from p, then q belongs C;

2. If p and q belongs to C, p is density-connected with q.

It should be noticed that directly density-reachability and density-reachability are symmetric for pairs of core points but not between a core and a border point and, in addition, density-reachability is transitive between core and border points (Ester et al.,1996).

There is more. Two border points of the same cluster C are not density reachable from each other because the core object condition does not hold for both of them. Hence, it is here that comes the density-connectivity property, because there must be a core point in C from which both border points of the cluster are density-reachable. For density reachable points, the relation of density-connectivity is symmetric and reflexive (Ester et al.,1996).

Two words must be spent also on Eps and MinPts.

They depend on each other, thus changing one requires changing the other one as well and, together, define the density in all the space: so, the concept of density depends on those parameters, and the clustering result is often quite sensitive to small changes on Eps and MinPts changes (Doran et al., 2019).

Eps is a distance, a surface or a volume, that must be considered for every single point, and MinPts, inside the area spanned by Eps, assign a point the label "core" or "border": so choosing them properly is of vital importance.

Giving Eps a high value could force the algorithm to insert more points than needed in a cluster, (only if the MinPts requirement is satisfied). On the other hand, if Eps is too small, the algorithm may become restrictive and could lead to more cluster than the real ones. MinPts works jointly with Eps but in the opposite direction: an excessive value assigned to it asks a point, for being "core", a quantity of other points in its Eps-neighborhood that can be unreasonably high, but decreasing the parameter can label as noise points than instead belongs to a cluster.

A rule of thumb is to start by setting MinPts to the dimensionality of the data plus one or higher (Ester et al.,1996). Then, for Eps, plotting the points' kNN distances (the distance of each point to its k-th nearest neighbor) against the points themselves, in decreasing order, and look for a knee in the plot (Doran et al., 2019): points inside of clusters should have small k-nearest neighbor distances, because they are density-connected, while a sudden increase of the kNN distance (the knee) indicates that the points are most likely noise, thus with large kNN distance. The knee is the Eps value.

This method can be implemented in the R environment with kNNdistplot() function, choosing k according to MinPts value (Doran et al., 2019).

Now, the algorithm (Ester et al., 2017):

1: Random choice of a point p.

2: Find, if p is not already processed, NEps(p);

> 2a: if the number in NEps(p) is less than MinPts then label p as noise and go to another p;

2b: if the number in NEps(p) is greater or equal to MinPts, start a new cluster C with p inside;

3: for all points $q_i, i = 1, \dots, n$, in NEps(p) without p:

3a: if q_i is labelled as noise or is unclassified, then it belongs now to C;

3b: if q_i is already classified (e.g. as a border point) then go to another q_i;

4: Find, if q_i belongs to C, NEps(q_i);

4a: if the number in NEps(q_i) is less than MinPts then q_i is a border point, so continue;

4b: if the number in NEps(q_i) is greater or equal to MinPts then q is another core point, hence add NEps(q_i) to NEps(p) and continue processing every point: C expands itself.

Pros and cons:

Pros: the number of clusters does not need to be pre-specified; it performs well with arbitrary shapes and it is able to detect outliers and noise points.

Cons: it can be quite inefficient with high numbers of data (Ester et al., 2017); determining appropriate Eps and MinPts could not be easy and domain knowledge could be essential; moreover, given that Eps and MinPts are fixed, it can have difficulties in identifying well clusters with very different densities.

DATA ANALYSIS

On a dataset called Mall_Customers.csv, created by Stefanie Grewenig and deposited on GitHub (machineLearningAZ/Machine Learning A-Z Template Folder/Part 4 - Clustering/Section 25 - Hierarchical Clustering/Mall_Customers.csv), are recorded 200 observations representing clients of a shopping mall. From this dataset, it is conducted an analysis using K-Means and DBSCAN: here are presented only the final results commented, while all the R-Code is not attached but it is saved in a private environment, and at disposal of the professor. The following R libraries have been used: "dplyr", "GGally", "factoextra", "dbscan", "fpc", "cluster".

In the dataset there are five variables for each observation that report their ID (not of interest for the analysis, so dropped), their gender (0=female, 1=male), their age, annual income (hundreds of dollars), and spending score, a point between 1 and 100, calculated discretionally by the Mall center basing on how much a client shopped in it.

Before starting with K-Means and DBSCAN analysis, a rapid look to the variables can give a first insight into the data, useful later for interpreting the clusters.

For the qualitative variable Gender, 56% are females, and 44% are males.

For the quantitative ones, the means and standard deviations (σ) are:

- Age: mean of 38.85 years and σ of 13.97 years;
- Annual Income: mean of 60.56 hundreds $ and σ of 26.26 hundreds $;
- Spending score: mean of 50.2, and σ of 25.82.

Once looked at the means, comparing their coefficients of variation (CV) is useful to see how variable they are: the spending score is the much more varying feature (CV = 0.51), while the age is the less varying (CV = 0.36); annual income lies in between (CV = 0.43).

A rapid visual inspection of the existing joint relations is given by the graph in figure 2:

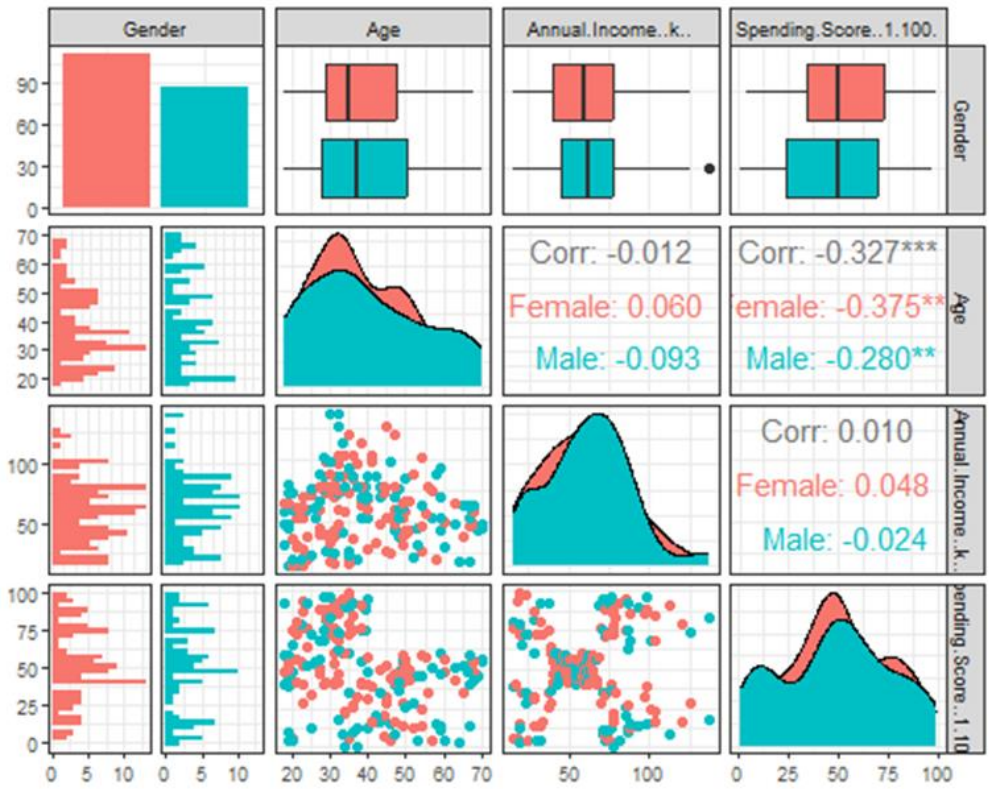
• Diagonal elements: the distribution for each variable is reported, grouped by sex;

• Off-diagonal elements:

• Left side: scatter plot (when only quantitative variables) and joint distribution between Gender (qualitative) and all others that are quantitative (first column);

• Right side: total and grouped (by Gender) correlation for the quantitative variables, and a box plot representing the grouped distribution (always by Gender) of all the quantitative variables on the first row.

Figure 2: Matrix-graph representing some joint characteristics of the features analyzed

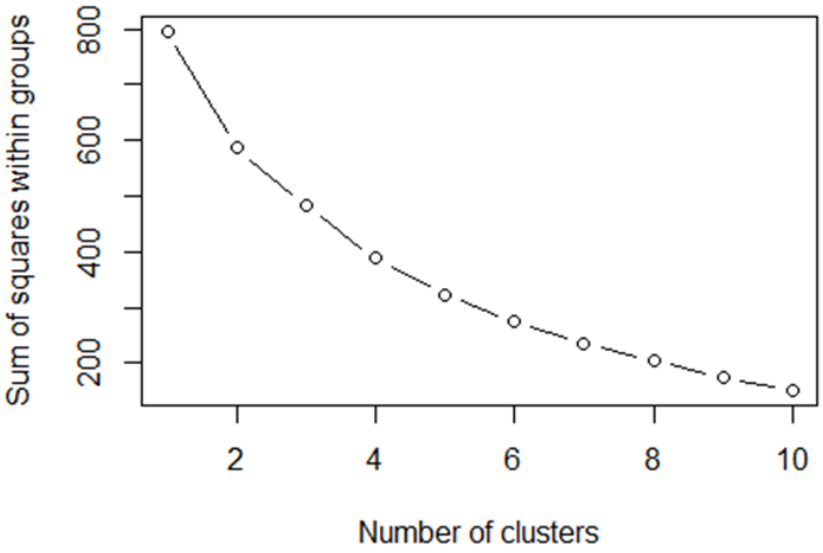


Source: personal elaboration using GGally library on RStudio

K-MEANS ANALYSIS

First, an implementation of the elbow method to choose k is useful: it consists in a creation of a vector containing all the sum of squares within a cluster, using the K-Means with, as input, the scaled data, giving as possible K a number between 1 and 10, letting it start with random centroids for 10 times, with a maximum number of iterations of 50. Then, plotting what is found, with as y axis the within sum of squares and as x axis the K (see figure 3).

Figure 3: plot of the elbow method



Source: personal elaboration using "factoextra" library on RStudio

According to the graph, K = 6 seems a good choice.

Once chosen k, implementing the K-Means in R gives the output reported in figure 4:

Figure 4: R output

```
##   kmeans Age_mean AnIncome_mean SpenScore_mean Gender Count
##    <int>    <dbl>         <dbl>          <dbl>  <dbl> <int>
## 1      1     32.7          86.5           82.1  0.462    39
## 2      2     39.9          90.5           16.1  0.567    30
## 3      3     50.6          49.7           40.1  0        41
## 4      4     56.6          49.6           38.9  1        30
## 5      5     24.6          40.7           61.5  1        23
## 6      6     25.9          42.2           57.5  0        37
```

Source: personal elaboration using "fpc" library on RStudio

The resulting clusters are:

1. A group of men/women, mean age of 33, with high both income and score.

2. A group of men/women, mean age of 40, with the higher income and lower score.

3. A group of women, mean age of 51, with mid-low both income and score.

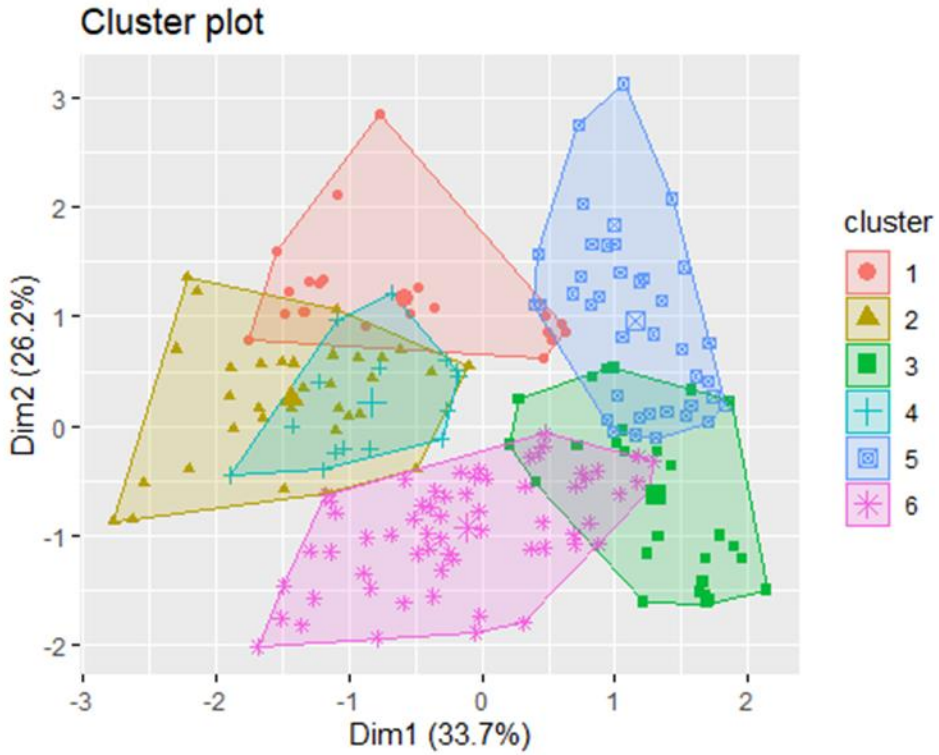
4. A group of men, mean age of 57, with mid-low income and high score.

5. A group of men, mean age of 25, with mid-low income and mid-high score.

6. A group of women, mean age of 26, with mid-low income and mid-high score.

A graphical representation of clusters (figure 5) spanned on two dimensions follows:

Figure 5: PCA cluster plot of K-Means



Source: personal elaboration using "factoextra" library on RStudio

The resulting clusters are of similar dimensions (ranging from 23 units to 41), and their convexity can be appreciated in figure 5; it is typical for K-Means.

It is also evident that Gender has determined a lot the clustering: only the first two contains both males and females, and if we look at clusters 5 and 6 they differ significantly only with respect to Gender, but the other variables are almost the same: this reflects the issue of choosing a fixed number of clusters before.

Anyway, quite a large variety exists within data and K-Means seems to capture it pretty well.

To see if the clusters are trustable, silhouette analysis is a useful tool.

First it is calculated a_i, the mean distance between the i^{th} observation and all others in the cluster C_i.

Then, define the mean of the distance from i to all points in any other cluster C_J and find b_i, the minimum of all the mean distances from i to any other cluster C_J.

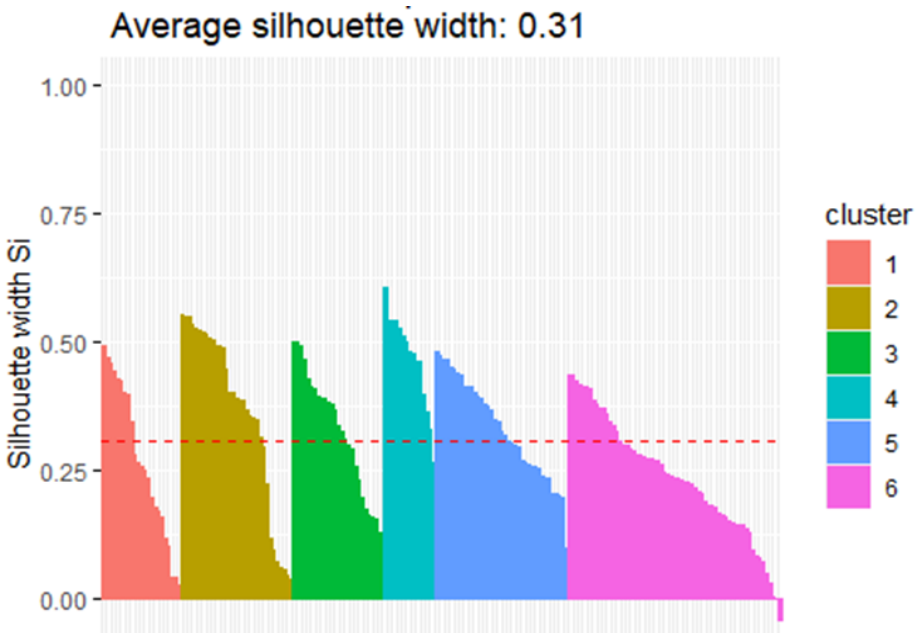
Finally, the silhouette $s_i = (b_i - a_i)/\, max\{a_i, b_i\}$ if in the cluster there are more than one observations, otherwise $s_i = 0$.

It goes from -1 to 1: closer to 1, better the clustering of i; around 0 and i is close to the nearest cluster; a negative value and maybe i is not in its true cluster (Rousseeuw, 1987).

The silhouette graph (figure 6) suggests that K-Means worked well in grouping the observations.

Other runs of the algorithm with slightly different K are not reported because changing them of one unity does not influence so much the final result, and changing them more, like k=3 or k=7, results in a very poor clustering according to silhouette analysis.

Figure 6: silhouette analysis of K-Means clustering



Source: personal elaboration using "cluster" library on RStudio

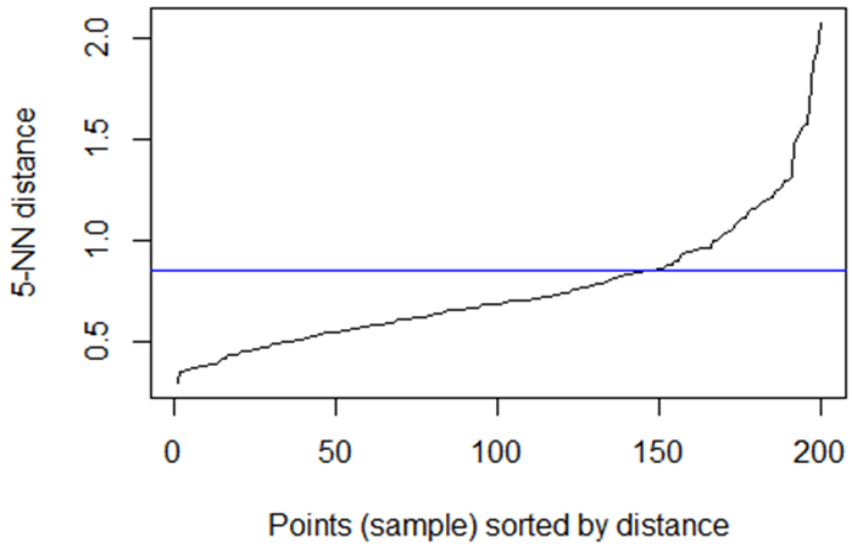
DBSCAN ANALYSIS

Moving on, now not one but three different DBSCAN analysis are reported: this is done to show how varying outputs can be by just inserting slightly different parameters.

FIRST RUN

Firstly, a graphical method for choosing the parameters (figure 7).

Figure 7: kNN distance plot



Source: personal elaboration using "dbscan" library on RStudio

According to the “dimension + 1 rule of thumb” [5], for the KNN distance plot MinPts=5 (that is k) should be set.

By viewing the plot (figure 7) choosing Eps = 0.85 seems a good idea because is where the distance starts to increase more.

DBSCAN can be performed.

Figure 8: R output

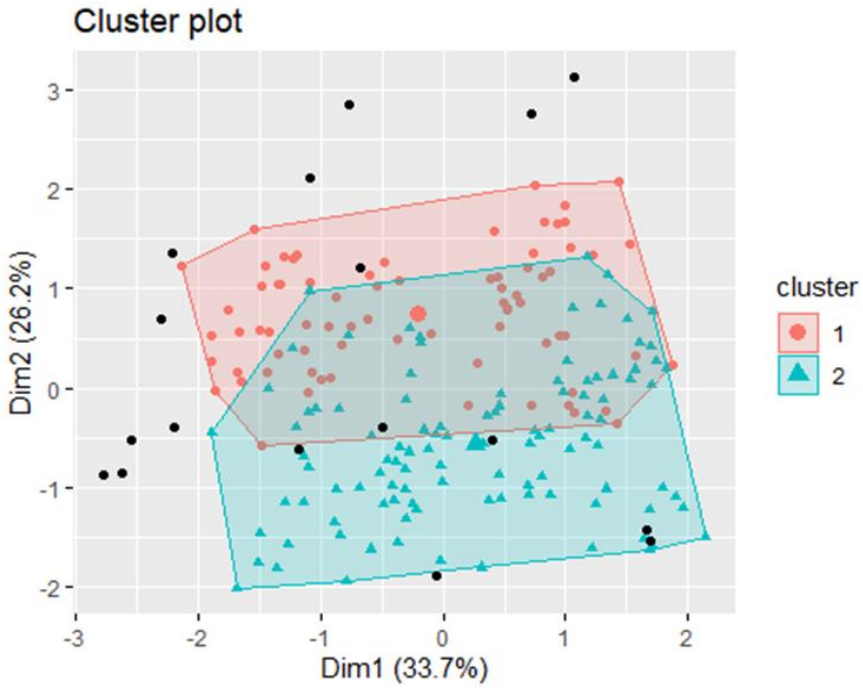
	dbscan	Age_mean	AnIncome_mean	SpenScore_mean	Gender	Count
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	0	42.1	60.1	32.4	0.765	17
2	1	39.0	61.9	52.7	1	75
3	2	38.3	59.7	51.3	0	108

Source: personal elaboration using "fpc" library on RStudio

A rapid look to the output (figure 8) shows that those clusters are not good: the three rows indicate the noise (first row) and two clusters. Starting from the third row, clearly DBSCAN puts together almost all the females, while the first cluster has the majority of men. The noise shown in the first row is not interesting since they are 17 units with nothing special.

A graphical inspection, reporting a PCA (figure 9) and a silhouette graph (figure 10), can show how those clusters are big and too general, hence not interesting.

Figure 9: PCA cluster plot of DBSCAN

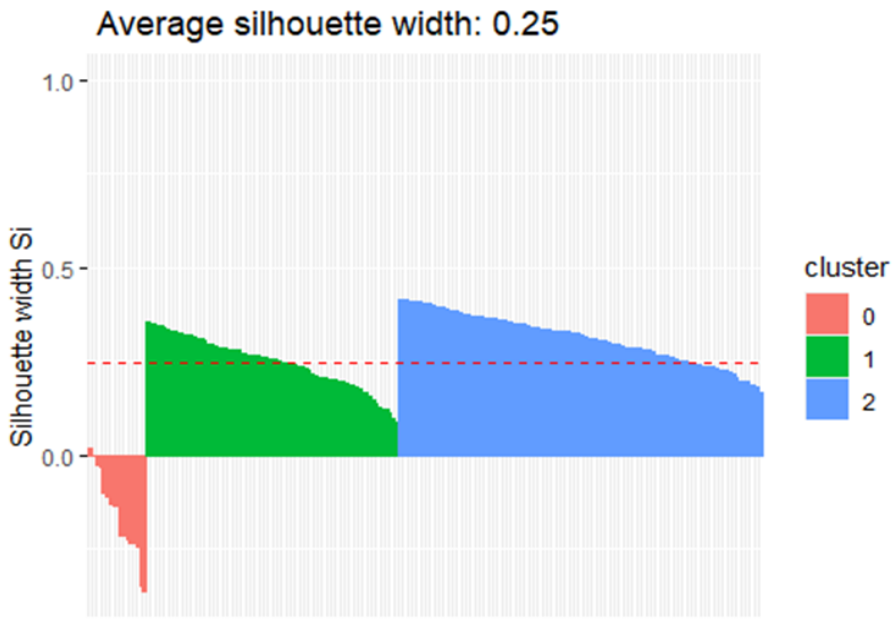


Source: personal elaboration using "factoextra" library on RStudio

Silhouette (figure 10) seems pretty good but clearly this tells nothing: DBSCAN clustered almost only by gender.

In addition, silhouette may not be the best indicator for the goodness of DBSCAN, since it is built using mean distances and, with an algorithm that follows density, so shapes, and no centroids, maybe such a measure can't tell how much well clustering is working. Anyway it is kept during the other analysis in this work because it is interesting to see how it changes and how it looks for every single cluster.

Figure 10: silhouette analysis of DBSCAN



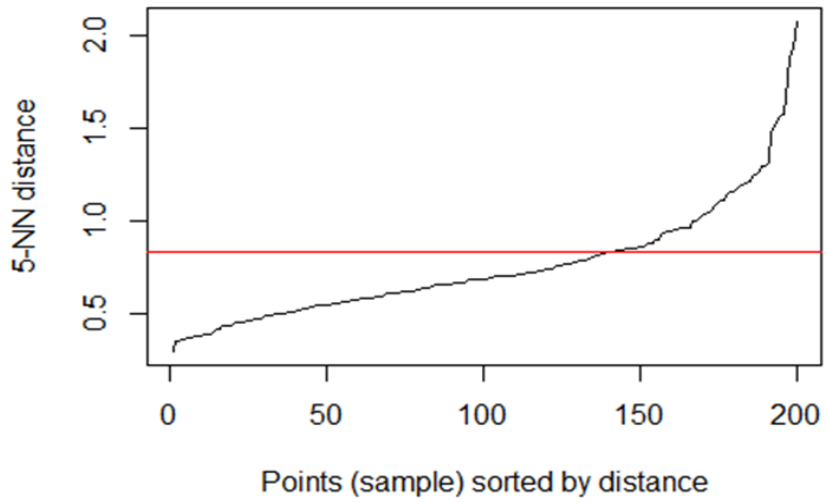
Source: personal elaboration using "cluster" library on RStudio

SECOND RUN

A second try, by just lowering a bit the Eps, is sufficient to discover new things.

In fact, putting it to 0.83, as the red line indicates in figure 11, and running again DBSCAN, new clusters are discovered; they are reported in the new output (figure 12).

Figure 11: kNN distance plot



Source: personal elaboration using "dbscan" library on RStudio

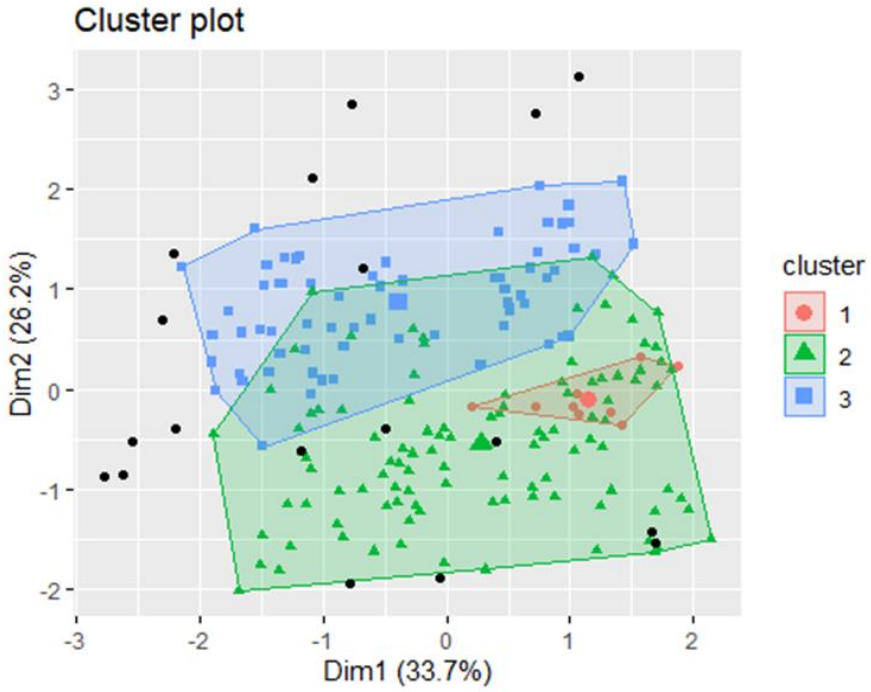
It can be seen how a little cluster is found in between the other two (see PCA plot in figure 13), and how different those clusters are with respect to K-means.

Figure 12: R output

	dbscan	Age_mean	AnIncome_mean	SpenScore_mean	Gender	Count
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	0	41.7	57.7	30.9	0.722	18
2	1	25	25.8	77.7	1	9
3	2	38.3	60.1	51.7	0	107
4	3	40.9	66.8	49.2	1	66

Source: personal elaboration using "fpc" library on RStudio

Figure 13: PCA cluster plot of DBSCAN



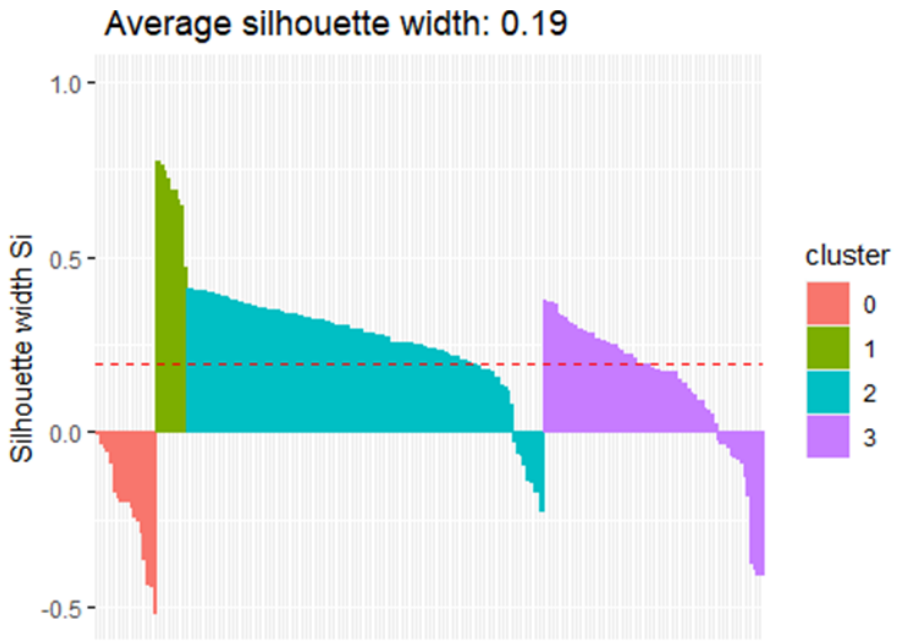
Source: personal elaboration using "factoextra" library on RStudio

Looking at silhouette (figure 14), the clusters could seem bad, especially the third one with a lot of negative values. Anyway, going deeper, DBSCAN gives a very interesting insight: 9 young males with a very low income but a very high spending score, differs a lot from others, and clusters 2 and 3, that groups again by Gender, seem to suggest that there are not high differences.

Nevertheless, this is not true: K-Means found that relevant differences in age, income and spending score exists. Maybe the gender is the more influencing variable in creating

clusters and in the density of data, but given that it is not so correlated to others as it can be observed in Figure 2, maybe removing sex something interesting happens, that is what is going to be performed.

Figure 14: silhouette analysis of DBSCAN

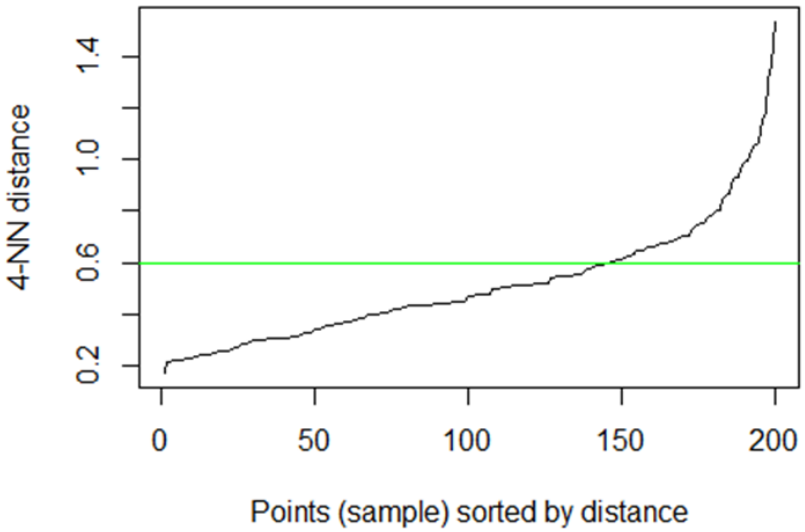


Source: personal elaboration using "cluster" library on RStudio

THIRD RUN

The third run of DBSCAN is made without the variable Gender. The first step is to set MinPts = 3, and then Eps is found using the kNN distance plot (figure 15).

Figure 15: kNN distance plot



Source: personal elaboration using "dbscan" library on RStudio

Implementing the algorithm with no gender and Eps = 0.6, MinPts = 3, gives the clusters shown in the following output (figure 16):

Figure 16: R output

	dbscan	Age_mean	AnIncome_mean	SpenScore_mean	Count
	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	0	39.6	73.6	37.7	18
2	1	40.8	52.7	44.8	142
3	2	32.7	83.1	82.4	36
4	3	20.8	76.2	8	4

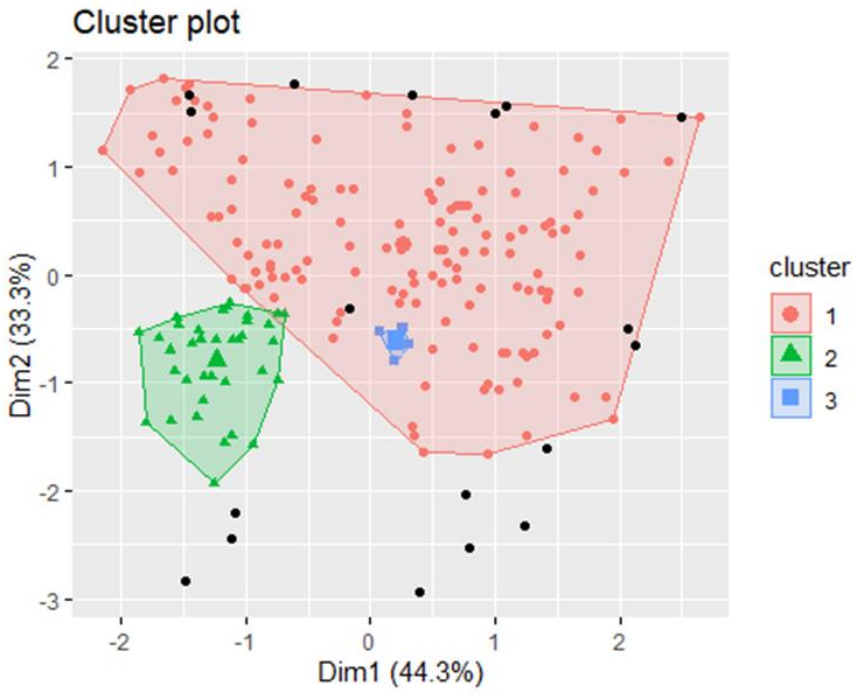
Source: personal elaboration using "fpc" library on RStudio

The majority of observations (the one not too far from income, spending score and age means) fall in the first cluster. Instead, the other two clusters are interesting: the second highlights 36 pretty young individuals with high both income and spending score, and the third 4 very young units with really high income but extremely low spending score.

Also the noise is curious: pretty high income and low score, with mean age around the total Age mean.

A graphical representation of how DBSCAN now clustered is given in figure 17.

Figure 17: PCA cluster plot of DBSCAN

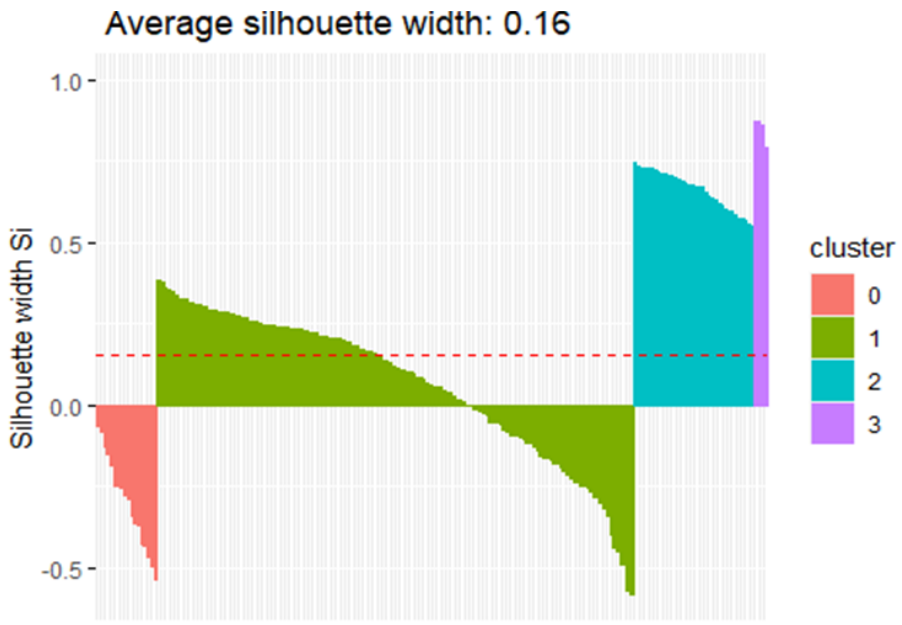


Source: personal elaboration using "factoextra" library on RStudio

Looking for the last time to silhouette (figure 18), it is not good but only regarding the first cluster; the other two clusters show instead high silhouette coefficients.

This is easily explainable: with respect to the first one, it has so much observations that, given the density approach, it is normal that some observations are closer to other clusters if measuring mean distance; while, for the other two clusters, they are so small (and, looking at the PCA plot in figure 17, convex) that a short distance from their clusters' means is almost obvious.

Figure 18: silhouette analysis of DBSCAN



Source: personal elaboration using "cluster" library on RStudio

CONCLUSIONS

After understanding how a partitioning (K-Means) and a density-based (DBSCAN) algorithms work, and having used them with a simple but various dataset, it is clear that the two methods offer totally different outputs: K-Means is good for convex, well separated (between) and homogeneous (within) distributions of data, can find groups just by looking at the distance from the means and running it several times ensures a local optimum that, given the K chosen is right, can describe very well the data.

Nevertheless, it is not always the case that data are grouped in such a simple and convenient way.

A density-based method like DBSCAN can, contrary to K-Means, find any particular shape, no matter how data are distributed, and it is not bounded by any prior choice of number of clusters.

Furthermore, while K-means forces all data to belong to one cluster, DBSCAN allows for noise, hence no outliers will be in the clusters and an analysis of also the noise can be done.

It is true that, for DBSCAN, two parameters must be chosen, but going and try several times can lead to interesting discoveries and a deeper knowledge about how data tends to be shaped.

If a prior knowledge of how observations are distributed exists, a choice between K-Means and DBSCAN is easily done. But if the context is of no knowledge, maybe implementing both, like it has be done in this work, is useful to see what the different algorithms can tell, with all their pros and cons, and discover different structures in the data.

Two is better than one.

REFERENCES

[1] Ankerst M., Breunig M. M., Kriegel H. P., Sander J. (1999). Optics: ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2), 49–60.

[2] Doran D., Hahsler M., Piekenbrock M. (2019). Dbscan: Fast Density-Based Clustering with R. *Journal of Statistical Software,* 91(1), 1–30.

doi:10.18637/jss.v091.i01

[3] Durojaye, D. I., Obunadike, G.N.. (2022) Analysis and visualization of market segmentation in banking sector using Kmeans machine learning algorithm, *FUDMA Journal of Sciences (FJS)*, Vol. 6 No. 1, 387-393.

doi:10.33003/fjs-2022-0601-910

[4] Ester M., Kriegel, H. P., Sander, J., Schubert, E., Xu, X. (2017). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN, *ACM Trans. Database Syst.* 42, 3, Article 19, 21 pages.

doi:10.1145/3068335

[5] Ester M., Kriegel, H. P., Sander, J., Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, Portland, 1996.. AAAI Press. 226–231.

[6] Hastie, T., James, G., Tibshirani, R., Witten, D. (2013). *An Introduction to Statistical Learning*, Springer New York Heidelberg Dordrecht London.

doi:10.1007/978-1-4614-7138-7

[7] Rousseeuw, P., J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics*, 20, 53-65

doi:10.1016/0377-0427(87)90125-7

[8] Sarker , I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions, *SN Computer Science,* (160), 21 pages.

doi:10.1007/s42979-021-00592-x