Authors: Diletta Goglia and Marco Petix

Data Mining Project 20/21

Analysis of the “customer_supermarket” dataset

Task 1: Data Understanding and Preparation

The “customer_supermarket” dataset

The customer_supermarket dataset contains information about the transactions between customers around the world and an unspecified business entity. This data is classified among eight different attributes for a total of 471910 records.

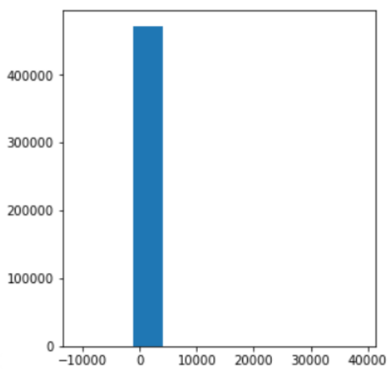
The Semantics and Statistics of the “customer_supermarket” dataset

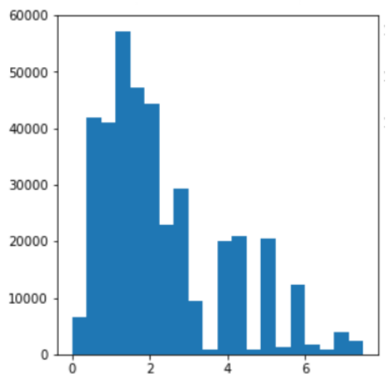
We proceed by describing the aforementioned attributes and their distributions:

BasketID (String)
- A predominantly numeric attribute, recognized by pandas as an object due to the presence of a few alphanumeric strings (see **Dealing with the negative values of Qta** below).
- It identifies the purchase sessions initiated by a single customer.
- The attribute, presenting 24627 distinct values, accounts to the same amount of individual purchasing sessions. Said amount decreases to 18867 after dropping the records containing outliers.

BasketDate (String at loading, then Datetime)
- A datetime attribute, initially recognized as containing objects prior to conversion through the pandas casting function.
- It contains the date and time of each purchase session initiated by the customers.
- The observation period identified by BasketDate starts on the 1st of January 2010 and ends on the 10th of December 2011.

Sale (String at loading, then Float)
- A numerical attribute, initially recognized by pandas as object due the use of commas as delimiter for the decimals, then cast to Float.
- It represents the amount spent by the customer to purchase a single unit of the products bought during each transaction.
- Basic checks regarding the attribute’s statistics revealed the presence of two records with negative values associated with the adjustment of debts.
- The general shape of the distribution of the attribute values was evaluated using the parameters of skew and kurtosis thus proving to be extremely positively asymmetric and leptokurtic. These conditions resulted greatly diminished after applying a process of outliers elimination within the dataset.
- Below, the distribution in question is shown as it appeared before, and after, the aforementioned process.





Customer ID (Float at loading, then String)

- A numerical attribute later casted to object due to its role as identifier for the customers taking part in the transactions.
- The attribute accounts to 4373 individual customers. Said amount decreases to 3477 after dropping the records containing outliers.
- It also contains 65080 missing values whose replacement is addressed within the following section.

CustomerCountry (String)

- A categorical attribute representing a country associated with each customer.
- The attribute accounts to 38 distinct countries.
- The vast majority of the dataset population (90%) consists in records with ‘United Kingdom’ as value for CustomerCountry.

ProdID (String)

- An alphanumeric attribute acting as identifiers for the products purchased within a transaction.
- The attribute accounts to the 3953 individual products. Said amount decreases to 3672 after dropping the records containing outliers.

ProdDescr (String)

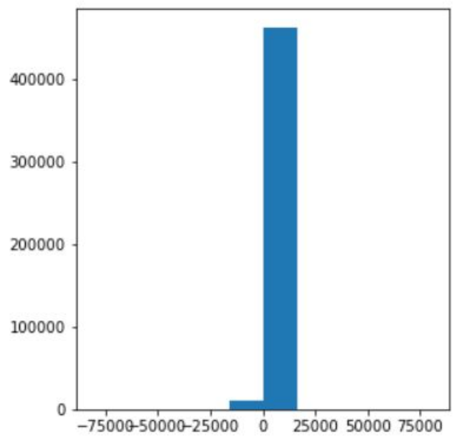
- An textual attribute representing a brief description of the products purchased within a transaction.
- The attribute accounts to 4098 individual descriptions, meaning that some of the products accounted for by ProdID descripted in multiple different ways.
- It also contains 753 missing values whose replacement, however, is not addressed due to the low influence of said attribute with respect to our analysis.

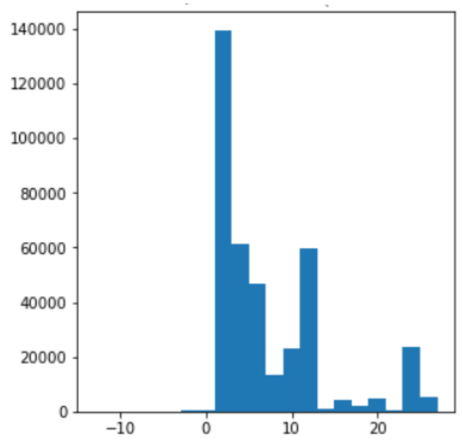
Qta (Integer)

- A numerical attribute representing the amount of unity purchased within a transaction with respect to a single type of product.
- Basic checks regarding the attribute’s statistics revealed the presence of several records with negative values, some of which further characterized by a BasketID containing C. According to our hypothesis, such records, while representing the legitimate cancellation of purchases, could mislead the calculation of some of the indicators developed in the second part of this task. To prevent this eventuality, a

process aimed at eliminating, or at least balancing, the pairs of transactions representing the completed and canceled purchase was applied.

- As for the Sale attribute, the general shape of the distribution of the attribute values was evaluated using the parameters of skew and kurtosis thus proving to be mildly positively asymmetric while still extremely leptokurtic. Once again, these conditions resulted greatly diminished by the process of elimination of outliers within the dataset.
- Below, the distribution in question before and after the aforementioned process.





Assessing and improving the Data Quality of “customer_supermarket”

In order to improve the quality of the data contained in the dataset, and therefore also of the analysis to be carried out, we examined the dataset in order to identify duplicate records or records with missing values and outliers.
Here follows the results of our analysis in this regard and the measures we have taken to correct them.

Duplicates in “customer_supermarket”

Assuming, by personal interpretation, the presence of duplicate records as the result of errors during data entry, and not as the intentional addition of multiple transactions, we decided to search for them within the dataset and, eventually, drop them.

Extra: Dealing with the negative values of Qta

As already mentioned, studying the dataset statistics led to the discovery of 9084 records representing canceled purchases.

By realizing the misleading influence that these records, and their positive counterparts (representing the same purchases before cancellation), could have on the computation of the shopping behavior of the customers we implemented a process aimed at identifying, cancelling, or balancing, said couples.

Not all the negative records showed a counterpart of the opposite sign but, by applying the aforementioned process, we proceeded to eliminate 8009 of the former and an equivalent number of the latter.

Missing values in “customer_supermarket”

As for the identification of missing values, having already confirmed the presence of the same in the CustomerID and ProdDescr attributes, we have considered the option of analyzing the records with value 0 (often used as NaN for numeric attributes) for the Sale and Qta attributes.

After actually identifying several of these records, we decided to consider the value 0 as legitimate in order to represent aspects of particular transactions. No replacement process was therefore applied for the latter.

The only missing values subjected to a replacement process were those belonging to the CustomerID attribute. This is how said process was implemented:

1. We segmented the dataset based on the value for CustomerCountry in each record;
2. Within each of the 38 segments of the dataset, one for each country represented by the CustomerCountry attribute, we identified the most active customer by computing the mode on the CustomerID attribute;
    a. In this regard, it was necessary to create a custom CustomerID in order to make up for the lack of a non-null one by the population segment associated with the CustomerCountry 'Hong Kong'.
3. We used these CustomerID values as a replacement for the missing ones.

The process therefore worked by associating the transactions without a customer, but with a CustomerCountry value relating to a particular region, to the most active customer in the region itself.

Outliers in “customer_supermarket”

The process of identifying and deleting outliers resulted in the deletion of 69967 records from the dataset. Their presence, as well as being strongly suggested by the Kurtosis values for the Sale and Qta attributes, was ascertained through both the use of boxplots and the computation of the interquartile range (IQR).

Correlation and visualization with “customer_supermarket”

Explicitly analyzing the correlation between the dataset numerical attributes did not produce remarkable results.

The Sale and Qta attributes proved to be barely correlated and the visualization via scatter plot, also augmented by the introduction of categorical attributes such as CustomerCountry, did not lead to any interesting conclusions.

In order to build plots displaying the behaviour of Sale and Qta over period of times, we created an *ad hoc dataframe* by duplicating the original one and setting the BaskeDate attribute as index.

This led to the discovery of interesting correlations such as:

- the increase in transactions near the middle of the week, Thursday seems to be the busiest day from the point of view of purchases by customers;

- the presence of few purchases with very high prices during the first few hours of operation of the supermarket in question;
- the increase in the number of low-medium price purchases close to the central hours of the day, possibly due to the purchase of food items for lunch.

The “customers” dataset

A second dataset was then created, this time focused on the representation of the purchasing behavior of each individual customer, and then indexed through the values of the Customer ID.

The Semantics of the “customers” dataset

We proceed by describing the attributes computed for the aforementioned dataset:

I (Integer)
- The total amount of products purchased by the customer.

Iu (Integer)
- The total amount of distinct products purchased by the customer.

Imax (Integer)
- The maximum amount of products purchased by the customer within a single shopping session.

E (Float)
- The Shannon's Entropy computed on the ProdID values associated with the customer.
- It’s a measure of variety, and disorder, in the products purchased by the customer.

Eb (Float)
- The Shannon's Entropy computed on the BasketID values associated with the customer.
- It’s a measure of variety, and disorder, in the transactions initiated by the customer.
- The aim is to distinguish the regular customers from one-time spenders.

Ew (Float)
- Shannon's Entropy calculated on all days of the week associated with the user's transactions.
- It's a measure of the variety and clutter in the customer's purchase timeline.
- The goal is to distinguish the customers who visit the store regularly.

Em (Float)
- Similarly to Ew, this is the Shannon’s Entropy computed on the months associated with the user’s transactions.

- Having a period of observation of slightly longer than a year, it's aim is to distinguish those who initiated transactions with the supermarket for a short period of time, like a single month.

Stot (Float)
- The total amount spent by the customer within the whole period of observation.

Smax (Float)
- The maximum amount spent by the customer within a single shopping session.

NSess (Integer)
- The total number of shopping sessions initiated by the customer.

Additional Attributes designed for cluster characterization

Country (String)
- The attribute Country associated with the Customer within the "customer_supermarket" dataset

Weekly Behaviour (**Monday**, **…**, **Sunday**) (7 Integers)
- Seven attributes containing the total number of products purchased by the customer during each specific day of the week
- The aim is to understand if the population of particular cluster tends to prefer a particular day for its purchases

Weekday (String)
- The day of the week during which the customer made more purchases overall
- The aim is similar to the one stated for weekly behaviour but while experimenting with an attribute simpler to compute

Monthly Behaviour (**January**, **…**, **December**) (12 Integers)
- Twelve attributes containing the total number of products purchased by the customer during each specific month
- The aim is to understand if the population of particular cluster tends to prefer a particular month for its purchases

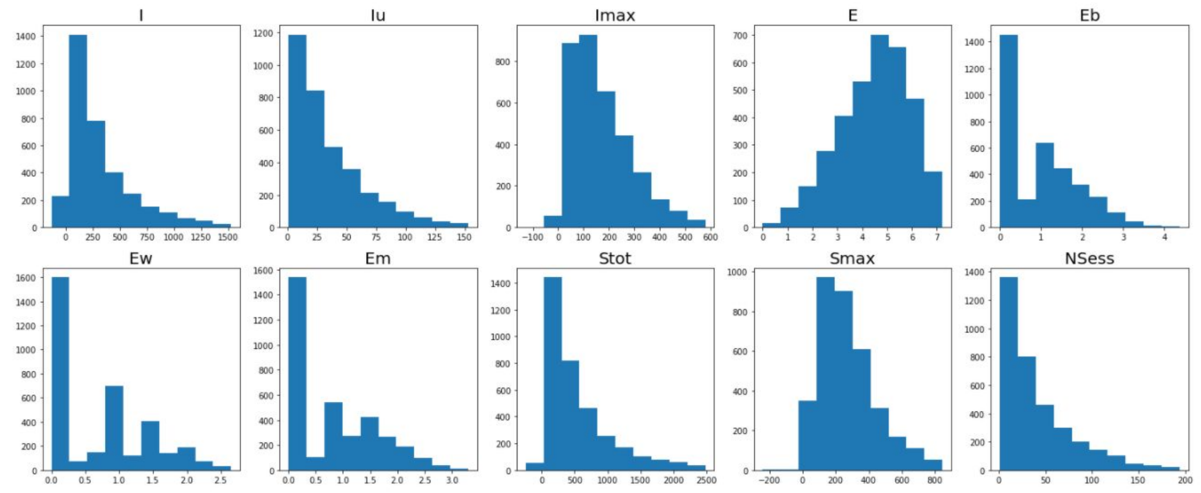
Month (String)
- The month during which the customer made more purchases overall
- The aim is similar to the one stated for monthly behaviour but while experimenting with an attribute simpler to compute

Data Statistics of the "customers" dataset

Once again we computed the skew and kurtosis parameters in order to obtain general information regarding the distributions of the new attributes.

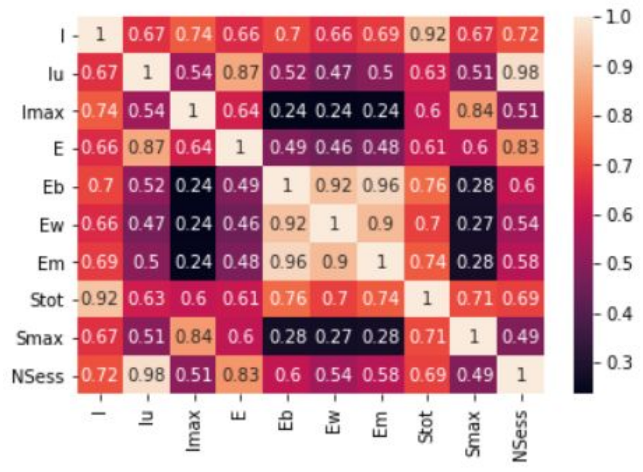
Attributes like I, Iu, Stot and Nsess resulted deeply skewed. Also, most of the attributes from Customers presented values of Kurtosis greatly over 3.

The presence of so many leptokurtik suggested an heavy presence of outliers. In order to deal with said records we again identified them, by computing the interquartile range (IQR), and subsequently dropped them.
These are the distributions of the attributes from Customers as they appeared after the aforementioned process.



Correlations within the new Dataset

We computed the correlation matrix between the main attributes of the new dataset in order to spot couples of highly correlated attributes potentially describing redundant informations.



Here follows some of the conclusions obtained by analysing the matrix above:

- The high correlations between **I** and **Stot**, and **Imax** and **Smax**, both refer to the increase in cost being obviously proportional to the number of products bought.
- The correlation between **Iu** and **E** is expected due to both attributes being an indicator of variety (and disorder) within the customer's choice of products.

- The very high correlation between **Iu** and **NSess**, as the one between **E** and **NSess**, highlight a tendency to prioritise the purchase of new products, new sessions often leading to the purchase of products that are not familiar.

Task 2: Data Clustering

Dataset introduction

The following clustering analysis is performed on the *customers* dataset: it contains 3494 observations and describes the **purchase behaviour of each single customer.**

The "customers" dataset contains both numerical attributes directly involved in the clustering process, and some additional attributes, both categorical and numeric, used later for the clusters characterization and interpretation.

For the exhaustive list of indicators we refer to the previous dataset explanation section of task1.

Pre-processing

Before starting the clustering process, some operations must be done. The two main activities for this step are:

1. ### High-correlated features elimination

   Clustering analysis and results can be affected by higher correlations: this because the semantic meaning of strongly correlated features is that they are basically capturing the same information.

   Dropping redundant attributes also benefits the analysis by reducing the dimensionality of the dataset and raising the influence that more useful features could have on the whole clustering process.

   For this purpose, we first examined the correlations between attributes to be clustered in order to identify the highly correlated couples. Then we fixed a maximum **threshold** value at 0.9 in order to identify highly correlated features and, subsequently, we selected which of these must be deleted.

   As a result of this analysis, the indicators that we decided to drop because of too high correlation are: *Eb, Em, Ew, Nsess*

   In particular, the high value of correlation for the three entropy-based indicators *Eb, Em, Ew* is due to the fact that they had very similar nature and so they were describing almost the same kind of information. Moreover, we had no problem in deleting them since they do not affect the clustering process.

On the contrary, even if the attribute *Stot* is highlighted as highly correlated with *I*, we decide not to drop them because they will further reveal themselves to be the most influential indicators that we have regarding the clustering process (see below "Identifying the most influential attributes" section).

2. Normalization

We have chosen the Min-Max normalization approach in order to have always-interpretable data so that we can avoid applying inverse transformation (needed instead in the Z-Score approach).

For this purpose we used the MinMaxScaler library from Scikit-learn.

Clustering by K-means

The K-means algorithm allows to divide a set of objects into *k* groups on the basis of their attributes.

For this project we used the KMeans library from Scikit-learn.

The value of *k* is the input of the algorithm, and it represents exactly the number of clusters that you want to find.

Identification of the best value of k

Clearly, the definition *a priori* (i.e. not knowing the data) of this number is one of the most delicate phases. Our aim in this context is to find the best value of *k* by running multiple executions. We started from an initial value of 2 and we went on computing the clusterization until we reached the value we chose for the maximum value of *k,* that is 20.

Evaluation

In order to evaluate the outcomes of this *k* value selection process we used the following three techniques, based on two different metrics:

1. **Elbow Method on the clusters Inertia:** plotting the SSE as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.
2. **Average Silhouette Method** as an indicator of both separation and cohesion among clusters.
3. **Insights from Hierarchical Clustering** performed through the Ward method, which aims at the local optimization of the SSE function.

Comparing conclusions

According to the first two methods, the values 5 and 8 could be plausible candidates for the role of optimal *k*. The elbow method also suggests the value 3 and, while the value 2 stands as the most promising candidate according to Silhouette score. The 2 value is also supported by the dendrogram previously displayed.

Ultimately, **we decide to compute four different clustering runs**, each associated with a potential *k* value candidate, and to leave the choice for the best *k* to post-processing evaluation on the obtained results.

Characterization of the obtained clusters

After having executed K-Mean algorithm for all the four *k* values candidates, we now present an analysis aimed toward the study of the **clusters composition and population**, so that we can fully understand and interpret the results obtained by the k-means algorithm.

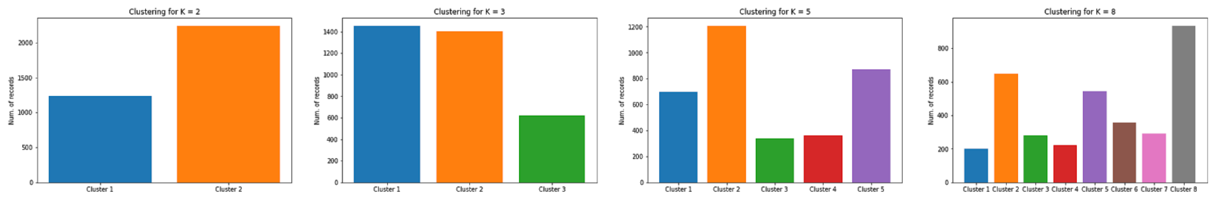


Fig. #: distribution of the original population of the customers dataset within the various clusters.

Identifying the most influential attributes

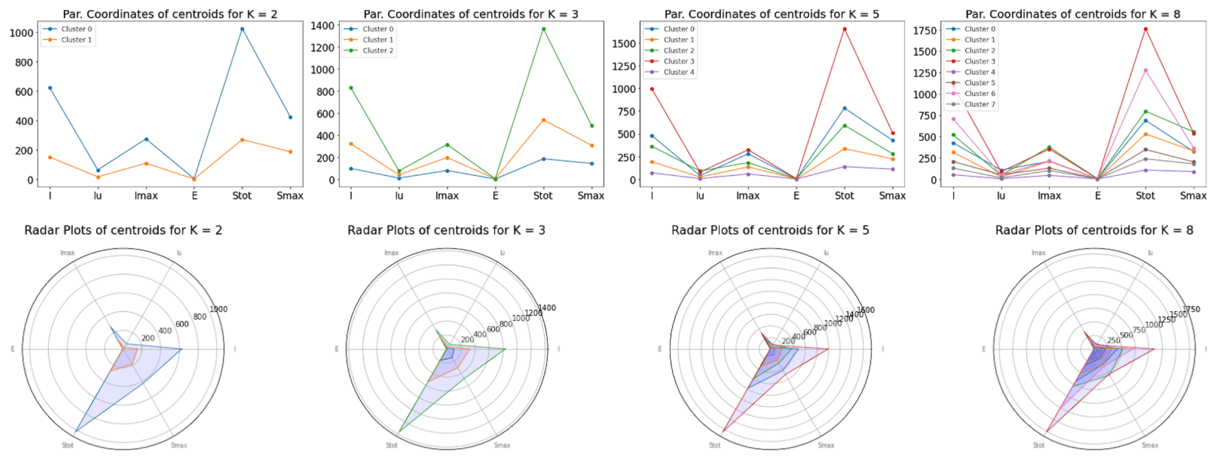


Fig. #. The coordinate plot and the radar plot show the differences for each attribute among the centroid of each cluster.

The same results seem to be shared among the four clustering: the centroids, and thus the population of each cluster, mostly differs in their values for I and Stot. Therefore, our decision to not drop them because of high-correlation value is justified.

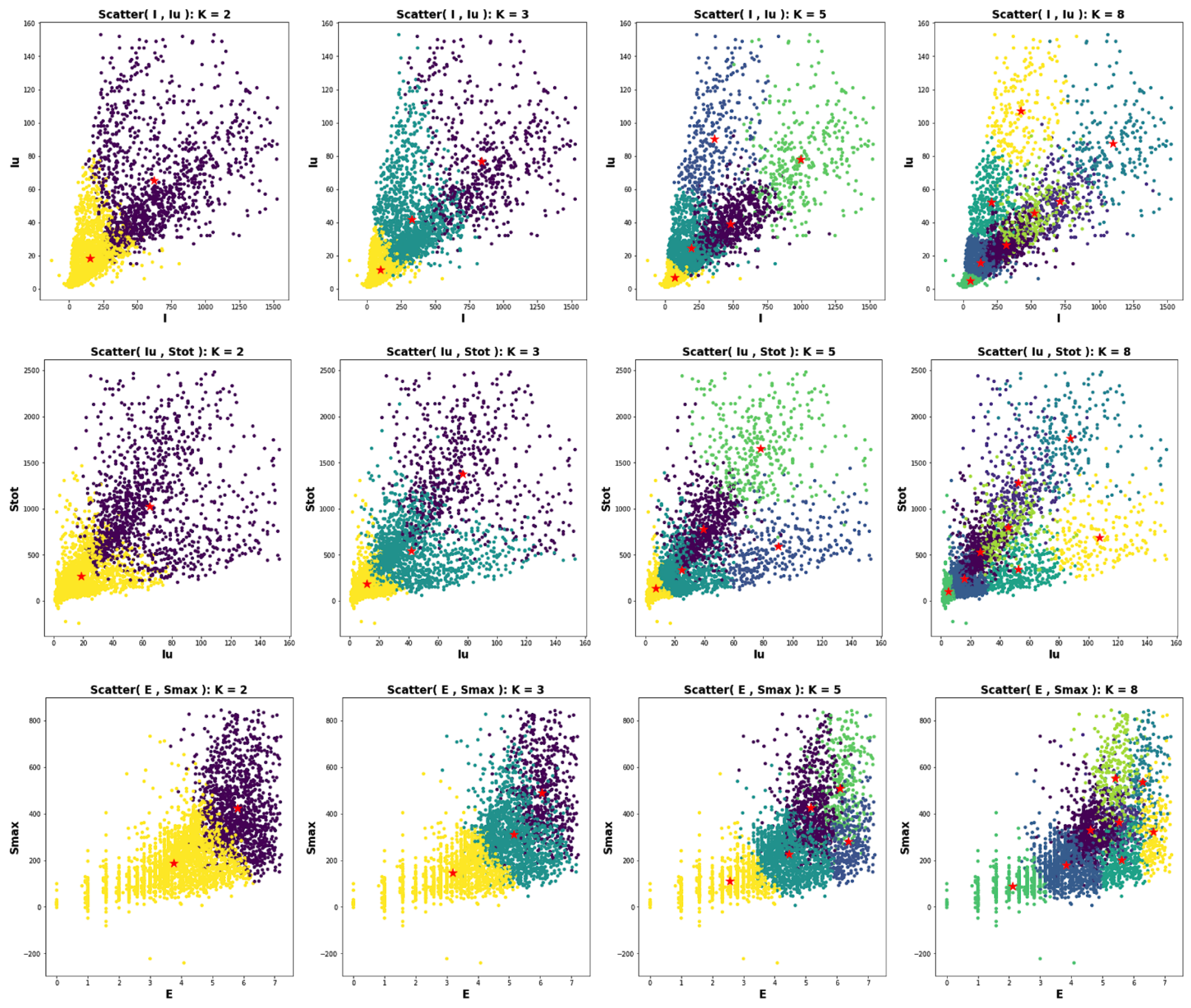
Characterizing the clusters

The interpretation of clusters with respect to categorical labels led to several barplots and comparison. In this report we do not show them since they are not meaningful for any conclusion or interesting observation. As a matter of fact, as we will confirm later in this report, the process of building clusters is not following one of these categorical features. Instead, the algorithm performs divisions with respect to a completely different indicator, that we will discover and explain later.

Visualizing the clusters and the centroids

We now present a global view on the clusters and their population thanks to bidimensional visualization, i.e. scatter plots for specific correlations between pairs of features.

Here you can clearly see the cluster separation and assignment in different colors, and also the centroids in red, so that it is immediate to understand and visualize where the center (i.e. the average value) is located for any cluster.



The graphs above show a sparse region of points at the top of the visualization. Clustering such as that for k equal to 2, 3 or 5 group this region into single clusters and this will certainly have a negative impact on the cohesion index of the latter.

Clustering like that for k equal to 8 instead covers this area with different clusters, thus gaining cohesion but showing a low separation index due to the overlaps in the central area.

Evaluation of the clustering results

At this point, we are going to use already-known metrics in order to evaluate the clustering obtained with our four versions of k-means. Doing this will grant us a numerical expression of the overall "goodness" of the arrangements of our clusters and, again, it will guide our choice toward which one is the best number of clusters for the optimal partition of this dataset.

The (internal) evaluation metrics that we are going to compute are:

- **SSE** to measure cohesion
- **Davies Bouldin Score** to measure separation
- **Silhouette Score** to measure them both

Clusters	Cohesion (SSE)	Separation (Davies Bouldin Score)	Cohesion and Separation (Silhouette Score)
K = 2	339.51	0.91	0.45
K = 3	247.53	1.05	0.36
K = 5	167.96	1.04	0.33
K = 8	120.46	1.07	0.31

As previously observed and expected, the SSE of our clusterings tends to decrease with the increase of k. A low value of SSE for k equals to 8 was therefore expected.
The SSE value for k equals to 5, however, appears relatively low if compared with the one from the other candidates. This could be an indicator of optimal cohesion within the clusters.
As already observed during our preliminary observations, the clustering for k equals to 2 seems the most promising in terms of Silhouette.

The latest results certainly make the choice of the optimal k rather conflicting. On the basis of what we have observed during the characterization of the various clusters, however, the choice of this value could be relatively irrelevant if not for reasons related to the field of application of the clustering process itself.

Final interpretation

By this we mean that, having assessed that the differences between the populations of the clusters for different values of k are mostly related to the **spending power of a customer**, the choice of one or the other value of k would simply represent the addition of intermediate classification intervals between the extremes linked to customers with small or large propensity / spending capacity.

This conclusion will be supported and reinforced by the DBSCAN clustering results.

K-means conclusion

What emerges from the evaluation phase with numerical metrics is that our hypothesis (coming from data visualization) that *k*=3 was the optimal choice is definitely confirmed by these numerical results.

Pros

As strongly sustained by literature, the k-means algorithm converges quickly. Despite our data not having a globular shape, the result of the partitions was pretty good.

Cons

The main disadvantage of k-means is that it requires choosing *a priori* the number of clusters to be identified; if the data is not naturally partitioned you get strange results. This human-driven definition for *k* values definitely required a lot of analysis and adjustments: in case of autonomous establishment from the algorithm itself of the correct number of clusters (as we will see for DBSCAN), this entire process would not have been necessary.

In terms of the quality of the solutions, the algorithm does not guarantee the achievement of the global optimum: the quality of the final solution largely depends on the initial set of groups and can, in practice, obtain a solution much worse than the overall optimum. Since the algorithm is usually extremely fast, it is possible to apply it several times and choose the most satisfactory solution among those produced.

Density-based clustering

The main aim of density-based algorithms is to find **dense regions of space**, i.e. areas in which there is an agglomeration of points. The notion of density, as well as the way in which we can quantify this "agglomerations", are determined by the choice of algorithmic parameters.

Moreover, unlike other clustering methods, they incorporate a **notion of noise** and they are able to “filter” this out.

DBSCAN algorithm

For the aim of this project we performed the density-based clustering using DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) algorithm.

It has only two input parameters: the radius ***eps*** and the minimum number of neighborhoods ***k***. Its output consists of cluster **labels** for each point in the dataset.

Parameters tuning

We succeeded in understanding which is the optimal assignment for both the parameters using the **knee method:** we selected the candidates as best values of *eps* for several neighbor chosen, i.e. for *k* parameter.

With this tool we can estimate where the value of *eps* parameter gets lower and lower, since it becomes stable.

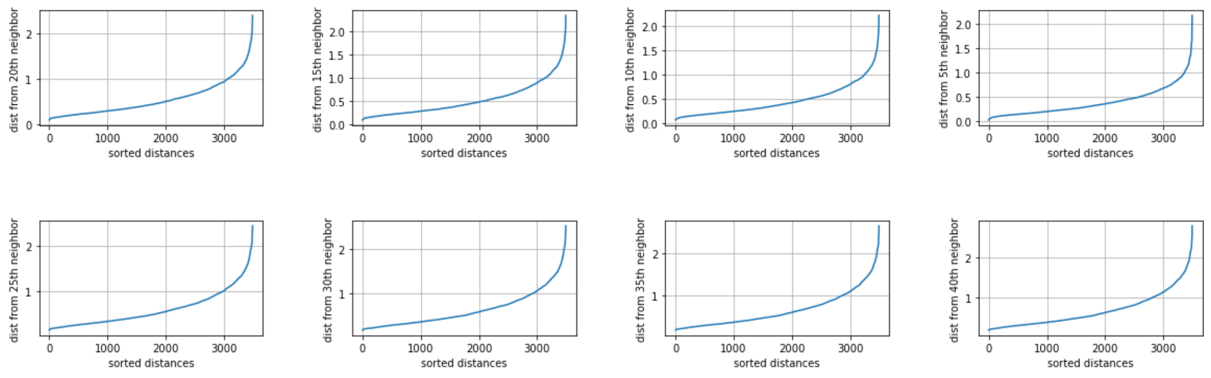


Fig. #: visualization of the knee-method to estimate *eps* for different possible values for *k*. It seems clear that the **candidates values for *eps* parameter are between 0.5 and 1**.

In the plots above we tried several values for *k* in order to know how many neighbors were appropriate to consider in a region with radius *eps* (second parameter).

Then, to confirm the best *eps* assignment, for each respective *k* quantity of neighbors, we had to do a lot of executions and trials (summarized in the attached file “param_tuning”), exploring different combinations of values within the range [0.5, 1].

Best configuration of parameters

What emerges from the previous analysis on parameters tuning (further supported by some tests on data visualization that we did) is that the best combination of parameters to use with this dataset to perform the DBSCAN algorithm are:

- **case1**: *min_sample* = 5, *eps* = 0.86
- **case2**: *min_sample* = 20, *eps* = 0.75

Both the configurations returned a number of clusters equal to 2.

As a matter of fact, this combo returns the best balance between noise points and number of clusters detected.

Two possible noise interpretations

These two cases are different in the way in which noise can be interpreted. Depending on these interpretations, both correct, we can recognize two possible outcomes:

- *Interpretation of case1*.

  Even if there are both **more-dense and less-dense regions**, since **DBSCAN is not able to recognize them** as two separate clusters (as explained before), the algorithm creates a **unique cluster which includes both**, labeled with 0. The remaining points (assigned to -1 label) represent what we properly call *noise*, i.e. data points that are really far from the dense area.

- *Interpretation of case2.*

  Since DBSCAN is not able to recognize these clusters with different densities, it assigns the most dense region to label 0 and the remaining region (less dense) to the label -1 : in this case, the meaning of the label -1 is *not* representing the usual definition of *noise*, but just the **inability of DBSCAN to recognize two different clusters with different densities**.

We will support these hypotheses using data visualization tools.

As a result of this analysis, we decided to run DBSCAN with different parameters, following both the configurations of the two cases presented.

Visualization and statistics

The following table reports some statistics about both the DBSCAN runs, like the respective parameters, **how many clusters** the algorithm has found and also the **size** of each of them (i.e. how many points each cluster contains).

Case	K (min neighbors)	Eps	Number of clusters	Noise label	Noise population	Cluster label	Cluster population
1	5	0.86	2	-1	99	0	3395
2	20	0.75	2	-1	483	0	3011

Tab. #: DBSCAN statistics of both the executions. It is evident from the difference of the noise population of the two cases that they can be interpreted in two different ways, both valid, as explained before..

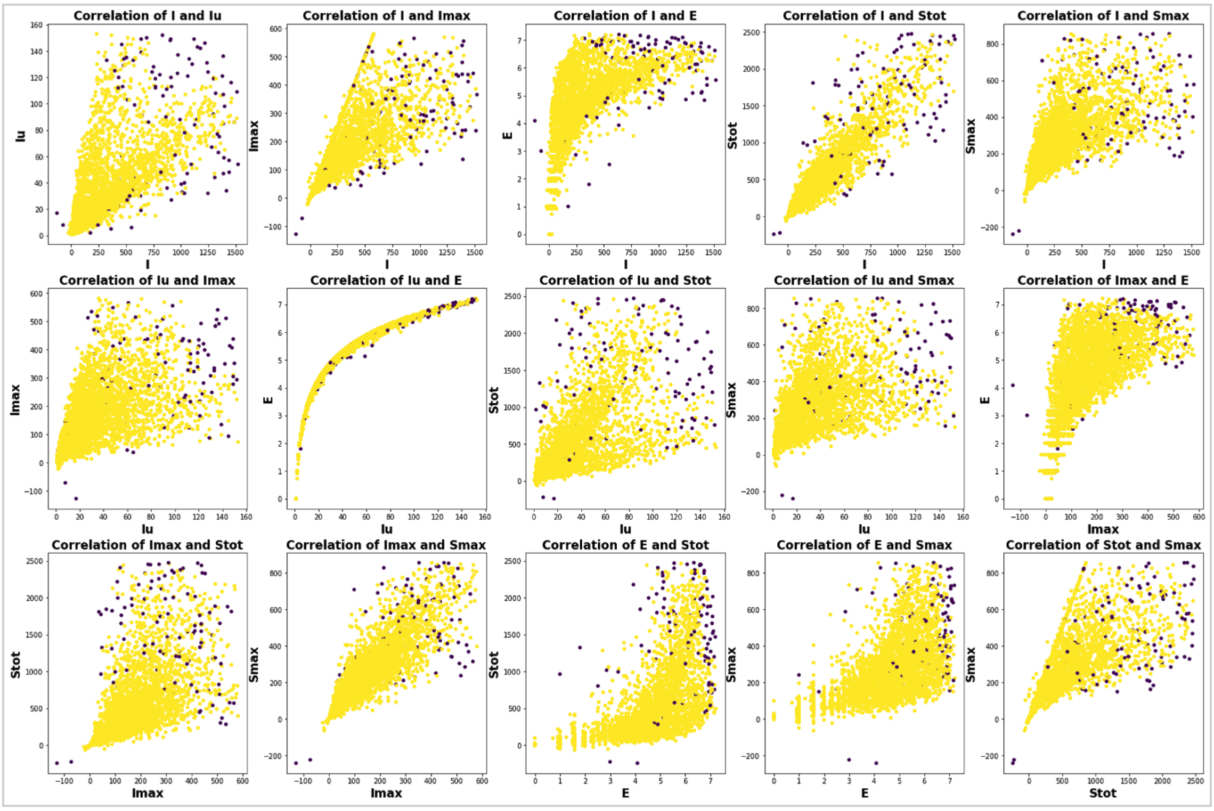


Fig. #: **Case1.** It is easy to see that DBSCAN is not able to distinguish more-dense and less-dense regions as different clusters (as opposed to K-Means that did it properly): so it groups them into a unique cluster (the yellow one). In this situation, data points labeled with -1 are *literally* noise points.

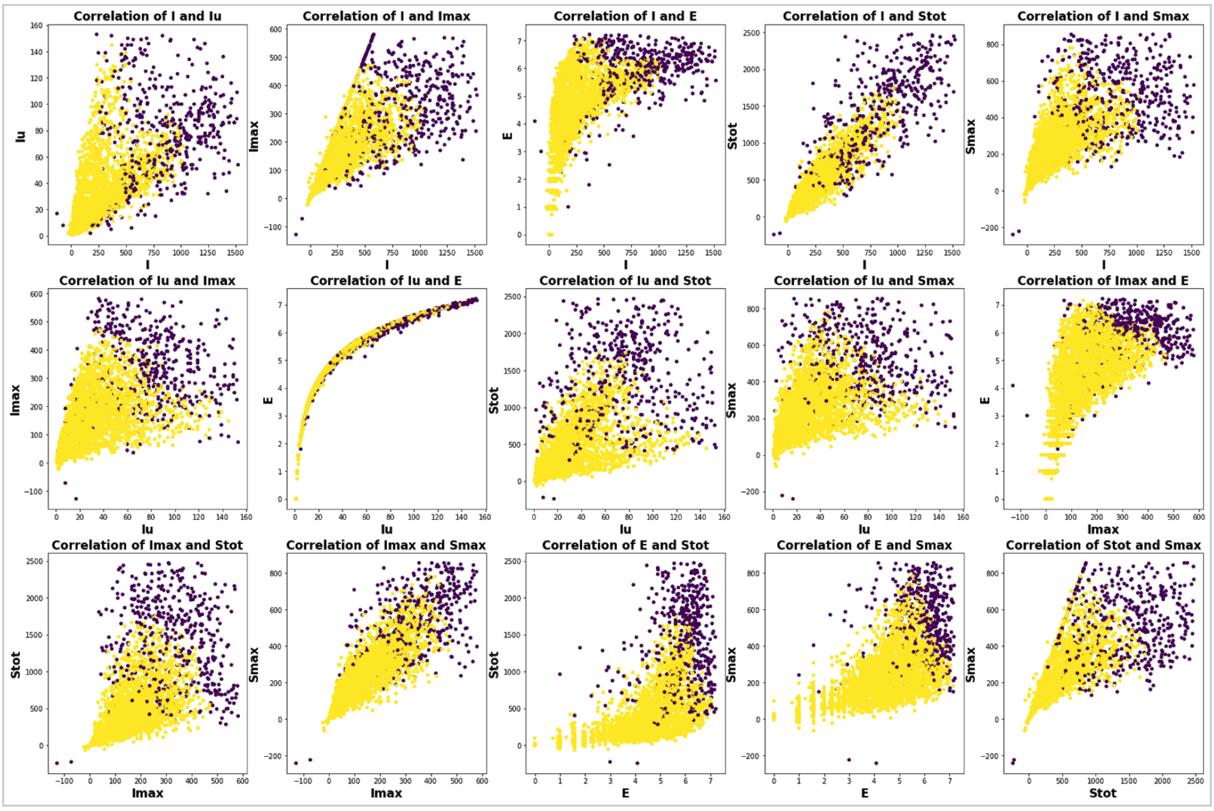


Fig.#: **Case2.** Here the inability of DBSCAN to clusterize together the two regions with different densities is interpreted in another way: since it cannot assign a separate cluster to the less-dense region, it automatically assigns to it the label -1. But, coherently with this interpretation, -1 points are not properly *noise*, but the representation of the algorithm's limitation.

Post-processing analysis

From the data visualization process above we can infer com first results and observations about DBSCAN algorithm functioning:

- **Robustness to noise**. It is clearly evident one of the major advantages of the density-based clustering: it is able to detect whatever it recognizes as noise and to label it as -1.
- **Ability to capture clusters with different shapes**. The yellow cluster is clearly a non-globular cluster but DBSCAN can easily detect it.
- **Unable to detect variable densities**. This is a well known limitation of DBSCAN algorithm, and here we have the opportunity to clearly see it: in the second interpretation we proposed, the 'agglomerate' in yellow is the most dense part and so it is correctly recognized as a cluster, but the purple one is classified as noise just because it is less dense that the other. To make a comparison, k-means algorithm instead classified that region as a separate cluster.

Evaluation with internal metrics

We now reinforce these observations based on data visualization with numerical information by computing internal indexes.

Then we will further proceed in the next paragraph in computing also external ones.

For this purpose we use again **silhouette** and **separation** scores in order to have a full overview of not only the cohesion but also separation between clusters.

Case	Silhouette	Separation
1	0.40	1.32
2	0.44	1.04

Tab. #: Silhouette and separation metrics computed for both the DBSCAN runs.

The result shows that the two obtained clusters are **well-separated**, especially in case1, **but not so cohesive.**

Evaluation with external metrics

In order to better interpret the DBSCAN clustering results, we now bring into play categorical features to discover clusters' semantic meanings and to understand how they are distributed.

The external indexes that we used in this phase are:

- **Similarity** between original features and labels found by the algorithm.
- **Homogeneity** that measures if clusters are homogeneous wrt class labels.
- **Completeness** measures how much original labels are concentrated in specific clusters.
- **Mutual Information** is a function that measures the agreement of the two assignments (i.e. original classes and labels assigned by the algorithm), ignoring permutations.

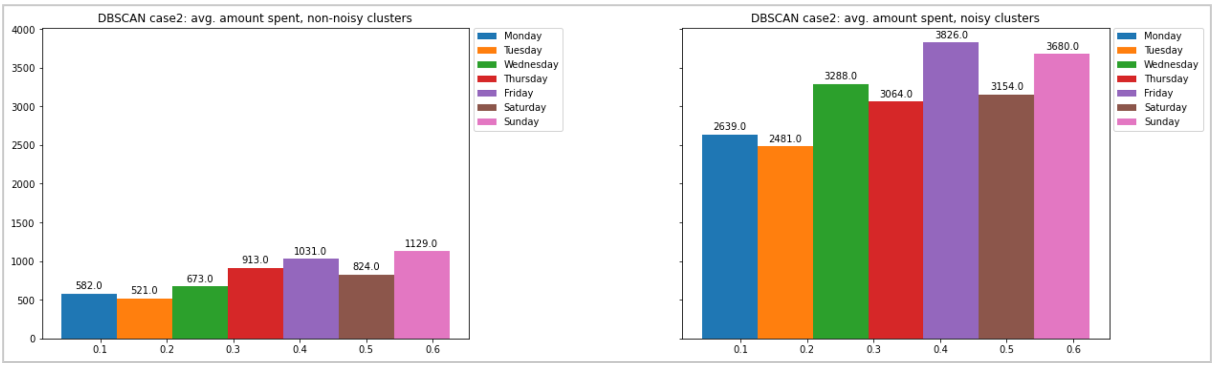
What emerged from that analysis is that no one of these metrics returns an interesting result: this is because none of the attributes for which we computed the external metrics is actually significant for clustering interpretation. As a matter of fact, as we already discovered during k-means results interpretation, the clusters are representing the **amount spent.** This means that customers are grouped according to how much they spent. This will be further confirmed at the end of DBSCAN evaluation results.
For all these reasons, it is perfectly reasonable that evaluation with respect to weekdays, months and country does not return any useful information.

Data visualization for clustering interpretation

In order to understand even better this latter conclusion about how categorical features are distributed with respect to DBSCAN results, we now present the most meaningful plot.

Coherently with what has been already said, the plot shows that the distinction between clusters is actually made following different "spending groups", i.e **groups of customers that have different spending behaviour.**

During this analysis we follow again the two interpretations of the DBSCAN algorithm.



Monthly distribution of clusters (for DBSCAN case2), plotted by **average amount spent:** noisy cluster is the one on the right side.

What we did here was: first computing the total spent with respect to weekly and then monthly distribution; then, dividing our population into four groups (representing noisy and non-noisy clusters of both the DBSCAN runs) and computing for that group the average value of the expense.

In this way, since the mean value is computed proportionally to the cluster's population, we can compare the plots and, as a consequence, we can compare noise with non-noisy clusters.

What emerges is that, while both cases of DBSCAN classifies as noisy the customers with lower average amount spent, an unexpected behaviour is displayed by monthly distribution of case 2, where there is the strongest difference between clusters: here, customers labeled with -1 (i.e. noisy group) are clearly those who spent more. For a better explanation, we presented above only and exactly the plot under consideration.

According to the interpretation of this plot, there actually exist two clusters, i.e. **two groups of people with different spending behavior.** However, since DBSCAN is not able to recognize clusters with different densities, this second group of people who spend a lot more is simply labeled as noise.

Thanks to the utility of data visualization tools within the interpretation, we are now able to discover this **separation made by following different customers purchasing behavior**, and so between clusters. This is not only a proof of the DBSCAN limitations but also the confirmation that **the second case is, in this context, the more appropriate interpretation to give.**

This whole result is a confirmation of the previously mentioned conclusion related to k-means clustering: the groups (i.e. clusters) ar ebuilt by the algorithms following the criterion **"spending power of a customer".**

Density-based clustering conclusions

Pros

- Density-based clustering methods differ from K-Means in the fact that they **do not specify the number of clusters beforehand**: for this reason, deciding *a priori* the k parameter is not needed. The algorithm is able to discover them by itself.
- Robust wrt noise: it detects noisy points and assigns them a -1 label (instead in K-Means they're assigned to the closest centroid)
- Is able to find clusters with any shape (not only globular / spherical shape as K-Means does), because there is no notion of *centroids* here.

Cons

- Density-based clustering methods are somewhat **harder to tune** compared to parametric clustering methods like K-Means. Things like the epsilon parameter for DBSCAN are less intuitive to reason about compared to the number of clusters parameter for K-Means, so it's more difficult to pick good initial parameter values for these algorithms.

  As a matter of fact, we had to do a lot of executions in order to tune parameters in a sufficiently good way (see above *Parameter tuning*).

- Another limit of the DBSCAN algorithm is that it is not **able to not recognize different densities**. From the scatter plots above it is easy to see that DBSCAN does not work very well in a varying-density environment.

  For this reason we had to distinguish between two different noise-assignment interpretations.

Hierarchical Clustering

Comparison

For all the DBSCAN's limitations and difficulties of interpretation, we state that the best algorithm to apply in this context and for this data, in order to have a clean, meaningful and clearly readable clustering result, is k-means algorithm.