Analysis of
Customer Supermarket
Dataset

Elia Piccoli 621332
Nicola Gugole 619625
Alex Pasquali 626378

Index

1. Introduction

The aim of this report is exploring the various phases of a Market Basket Analysis conducted on a given dataset. In particular the objective is to extract the customer behaviour by working on a derived dataset which is constructed from the original one.

The initial dataset is composed of a set of records, each record represents part of a basket transaction and is composed of 8 attributes, containing basket id, product id, customer id and temporal information.
The derived dataset is also composed of a set of records, each record represents a customer and is composed of the customer's statistics derived from his/her interactions in the original dataset.

The first part of the analysis is composed of:
1. *Data Understanding and Preparation*: original dataset has to be cleaned. Looking at data distribution, missing values, correlation between attributes is fundamental to the semantical understanding of the data and the following cleaning. It is also extremely important for preparing and correctly computing the customer's statistics needed for the derived dataset construction.
2. *Customer Dataset Construction*: once the original dataset is cleaned it is possible to create the various indexes which compose the derived dataset.
3. *Clustering*: the use of 5 different algorithms (K-Means, DBSCAN, Hierarchical Clustering, Fuzzy C-Means, Genetic Algorithm) leads to finding groups of customers with similar properties and therefore defining some behavioural description of customer groups.

2. Data Understanding and Preparation

The original dataset is composed of **466678** entries representing part of a basket transaction.

2.1 Data semantics

Taking a closer look at the 8 attributes:

BasketID:
Basket identifier composed of a total of **24627** unique values.
Object attribute since not all basketIDs are numerical, more specifically they are either:
- Fully numerical: positive Qta
- Starting with C: negativa Qta
- Starting with A: only 2 entries, both having negative Qta and ProdDescr *Adjust bad debt*

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 466678 entries, 0 to 541909
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   BasketID        466678 non-null  object
 1   BasketDate      466678 non-null  object
 2   Sale            466678 non-null  object
 3   CustomerID      401605 non-null  float64
 4   CustomerCountry 466678 non-null  object
 5   ProdID          466678 non-null  object
 6   ProdDescr       465925 non-null  object
 7   Qta             466678 non-null  int64
dtypes: float64(1), int64(1), object(6)
memory usage: 32.0+ MB
```
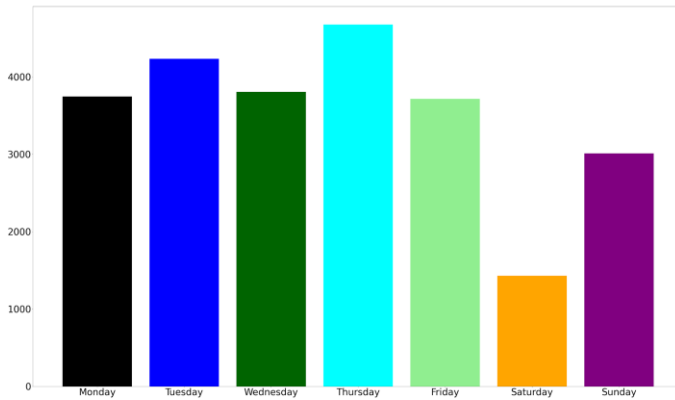
Timestamp of the basket transaction entry. Object attribute defining the period of observation (01/12/10 08:26 - 09/12/11 12:50).

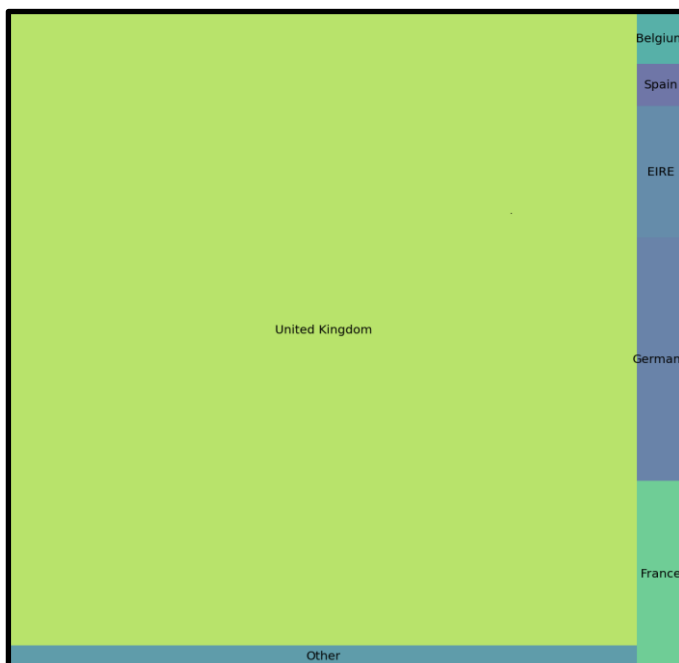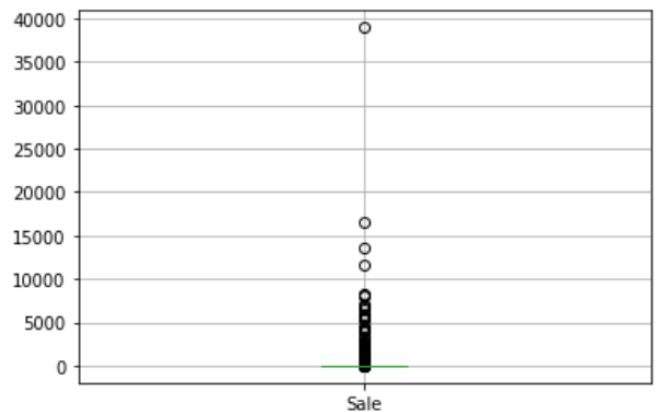*On the left*: number of orders per weekday.

Sale:
Cost per single product.
Object attribute (not float because decimal separator in the dataset is **comma** instead of **dot**).

CustomerID:
Customer identifier, composed of **4372** unique values, will be the index of the derived dataset.
Float64 attribute since for some arcane reason each customer terminates with the suffix '.0'

CustomerCountry:
Customer country composed of **37** possible countries.
Object attribute, a string which is the name of the country.

On the left a tree map representing the number of orders per country, showing the predominance of the UK in the market.

ProdID:
Product identifier composed of **3880** unique values.
Object attribute since these identifiers are alphanumeric strings.

ProdDescr:

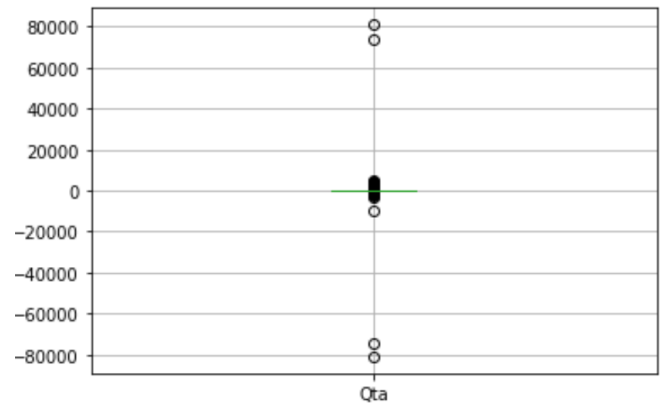Brief product description composed of **4007** unique values.

Object attribute, together with *CustomerID* is the only attribute having missing values.

Qta:

Product quantity, ideally it can be either:

- positive when buying the product
- negative when returning the product

Object attribute, boxplot shows it needs some analysis.

2.2 Data Cleaning and Data Transformation

Data semantics already showed that there are errors and impurities inside the dataset: missing values, data with wrong types, data which needs transformation. Before proceeding to the customer dataset creation, the data quality needs to be assured.

In order to do so, various filters and transformations were applied:

- *Sale* attribute has been transformed to floating point by converting the decimal separator from **comma** to **dot**.
- Entries with *Sale* values too low to be relevant (e.g. Sale < 0.01) were removed. As a side effect the resulting dataset had no more missing values regarding *ProdDescr*.
- Missing *CustomerID*s were removed from the dataset since without this attribute there is not an index for the customer dataset records.

	BasketID	Sale	CustomerID	ProdID	ProdDescr	Qta
201553	554301	1241.98	12757	M	Manual	1
201554	C554302	1241.98	12757	M	Manual	-1
297438	562946	2500.00	15581	M	Manual	1
342997	566927	2033.10	17846	M	Manual	1
406405	C571750	2118.74	12744	M	Manual	-1
406407	571751	2118.74	12744	M	Manual	1
467437	576339	1500.36	14096	DOT	DOTCOM POSTAGE	1
494727	578270	1270.06	14096	DOT	DOTCOM POSTAGE	1
508459	579196	1526.76	14096	DOT	DOTCOM POSTAGE	1
528083	580727	1599.26	14096	DOT	DOTCOM POSTAGE	1

- Arcane '.0' suffix was also removed from *CustomerID* after making sure that all customers had the same strange suffix.
- *Qta* attribute is the most delicate attribute since it can be negative. An important decision was made to decide when the attribute negativity was to consider legal, either:
    - *Qta* value is referring to a discounted product
    - *Qta* negative value is matched (for that same product and that same customer!) by a positive *Qta* value which is at least of the same magnitude. If the magnitude of negative *Qta* is higher than the corresponding positive there is no plausible meaning to give to that product: the returned quantity would be higher than the bought one, impossible in the dataset period of observation!
  Therefore, each record not satisfying the above conditions was removed from the dataset.
- Further outliers in *Qta* not removed by the previous filtering were dropped.
- Outlier analysis in *Sale* highlighted some particular *ProdID* values which required closer studies: ['POST' 'D' 'M' 'BANK CHARGES' 'DOT']

- ○ *BANK CHARGES:* by looking at these entries statistics it is plausible they represent some kind of bank payment that may not be strictly correlated to the customer sale analysis. Hence they were removed.
- ○ *D:* by looking at these entries *ProdDescr* ('Discount'), it is clear they regard some shopping-related discount on a particular customer, surely relevant for the analysis. Therefore they were left in the dataset.
- ○ *DOT/POST:* treated together because of their similar *ProdDescr* (both containing 'POSTAGE'), it is plausible these entries have something to do with postage payments, not necessarily correlated to the customer sale analysis. Therefore they were removed.
- ○ *M:* by looking at these entries *ProdDescr* ('Manual'), it is possible these records are referring to payments due/received (some of these are positive, some are negative) about manual works of some kind. Without any further information it is impossible to understand the true nature of these records and even more impossible to assure the relevancy for the analysis. Therefore they were removed.

By the end of this process the resulting dataset is cleaner and more meaningful, ready to be used for creating the customer dataset.

3. Customer Dataset

The derived customer dataset is composed of **4333** entries, one per customer. As a result of data cleaning some customers were deleted (at the beginning there were **4372** unique customers).

3.1 Indexes creation

Several indexes were computed and added to the customer dataset, in particular the aim was to create a diversified set of dimensions, including statistical indexes such as mean, max, min of various original attributes, but also temporal indexes and even a couple exploiting Shannon entropy.

- *TProd*: total number of items purchased by a customer during the period of observation.
- *DProd*: total number of distinct items purchased by a customer during the period of observation.
- *TRProd*: total number of products returned by a customer in the period of observation.
- *TSale*: total amount spent during the period of observation.
- *TSaleWRet*: total amount spent (without counting returned products) during the period of observation.
- *TOrder*: total number of orders made by a customer during the period of observation.
- *MaxPO*: maximum number of items purchased by a customer during a single shopping session.
- *MinPO*: minimum number of items purchased by a customer during a single shopping session.
- *MinPSale*: minimum amount spent for a single product during the period of observation. Calculated by finding minimum *Qta*Sale*.

- *MaxPSale*: maximum amount spent for a single product during the period of observation. Calculated by finding maximum *Qta*Sale*.
- *MeanPSale*: mean amount spent for a single generic product during the period of observation. Calculated by finding mean *Qta*Sale*.
- *MeanProdOrder*: mean number of items purchased by a customer during a generic shopping session.
- *MeanSaleOrder*: mean amount spent for a generic order during the period of observation.
- *OrderMonth:* mean monthly amount of orders made by a customer during the period of observation.
- *ProdMonth:* mean monthly amount of products purchased by a customer during the period of observation.
- *SaleMonth:* mean monthly amount spent by a customer during the period of observation.
- *SETSaleQta*: Shannon Entropy for the variability of *Qta*Sale* for a customer during the period of observation.
- *SESaleQtaOrder*: Shannon Entropy for the variability of *Qta*Sale* (accumulated per order) for a customer during the period of observation.
- *MaxOrderMonth:* the only categorical attribute, containing the month (as string of 3 characters) where the customer made the maximum amount of orders in the period of observation.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4333 entries, 0 to 4332
Data columns (total 20 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   CustomerID      4333 non-null    int64
 1   TProd           4333 non-null    int64
 2   DProd           4333 non-null    int64
 3   MaxPO           4333 non-null    int64
 4   MinPO           4333 non-null    int64
 5   MeanProdOrder   4333 non-null    float64
 6   TSale           4333 non-null    float64
 7   TSaleWRet       4333 non-null    float64
 8   MinPSale        4333 non-null    float64
 9   MaxPSale        4333 non-null    float64
 10  MeanSaleOrder   4333 non-null    float64
 11  TRProd          4333 non-null    int64
 12  MeanPSale       4333 non-null    float64
 13  TOrder          4333 non-null    int64
 14  OrderMonth      4333 non-null    float64
 15  ProdMonth       4333 non-null    float64
 16  SaleMonth       4333 non-null    float64
 17  SETSaleQta      4333 non-null    float64
 18  SESaleQtaOrder  4333 non-null    float64
 19  MaxOrderMonth   4333 non-null    object
dtypes: float64(12), int64(7), object(1)
memory usage: 677.2+ KB
```

3.2 Dataset Correlation and Redundancy Analysis

Analyzing the correlation matrix (excluding *MaxOrderMonth,* the only categorical attribute) it is clear there are some attributes too correlated one to the other. These attributes need to be dropped to avoid redundancy and bias in clustering.
Running a little script, the attributes over the correlation threshold (decided to be **0.75**) are **10** over the total of **20.** In particular, the attributes are: ['*TProd', 'MaxPO', 'MinPO', 'MeanProdOrder', 'TSale', 'TSaleWRet', 'TOrder', 'OrderMonth', 'ProdMonth', 'SaleMonth'*]
The final decision was to drop all these attributes apart from *TProd, TSale* and *TOrder*. Motivations:

- *TProd* and *TSale* are indeed highly correlated but they have a different meaning which is interesting for both for the clustering and the post-clustering analysis phase.

Correlation matrix

- *TOrder* was highly correlated to the dropped attributes. Once the chosen ones were dropped its correlation array was below the selected correlation threshold.
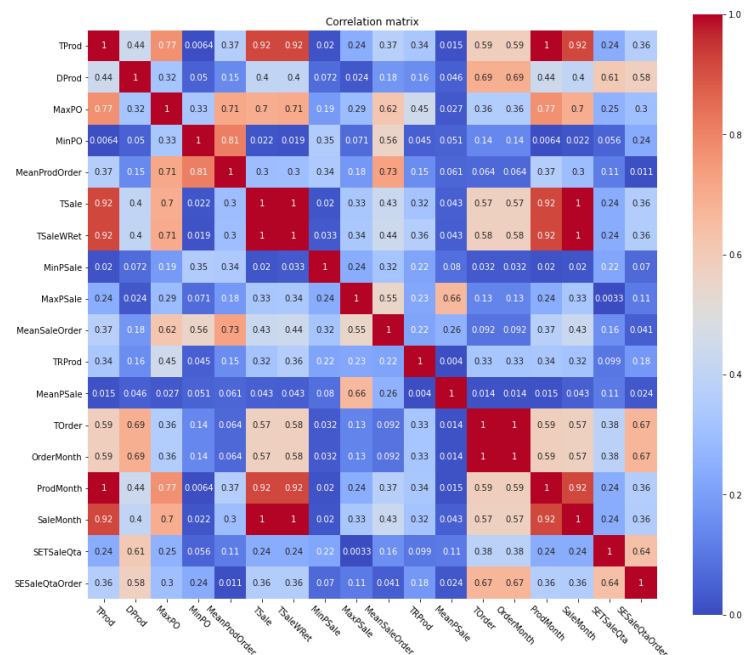
4. Clustering

4.1 K-Means

The clustering phase for the K-Means algorithm could be divided into 3 possible subparts:
- Choosing the attributes on which the algorithm will run
- Choosing the best value of K
- Analyze the results: does the algorithm individuate customer behaviours?

4.1.1 Choosing Attributes for Algorithm Run

Only a few of the **13** remaining attributes had to be kept for the clustering (to avoid the curse of dimensionality and to keep out non characterizing attributes), and so a subset had to be decided. The decision was driven using analysis tools, logic and also a lot of trial and error.
After using the **PCA** (taking into consideration the most predominant subset of attributes in the PCA generation) as an analysis tool, the results were in, and they were not so good.
Following this first phase, a systematic trial and error produced a subset selection which was also a logic choice: the chosen subset contains the most diversifying (with also a bias to choosing the most robust) attributes in the customer dataset:

['TProd', 'MeanSaleOrder', 'MeanProdOrder', 'MeanPSale']

4.1.2 Choosing K Value

K is a fundamental parameter in K-Means clustering, it defines how many clusters will be the result of the algorithm. It is important to try many different runs to see the resulting metrics over a plethora of different K values. After analyzing the metrics resulting from producing a number of clusters between **2** and **15**, the plots were strongly hinting to choose K = **4.**
This is due to the crossing of 3 parameters optimization: the informal elbow rule for the SSE metric (the lower this value is, the denser are the clusters), the need to keep as high as possible the Silhouette metric (the higher this value is, the farther are the clusters from each other), the need to keep as low as possible the Davies-Bouldin index (metric composed of a particular ratio between SSE and Silhouette).
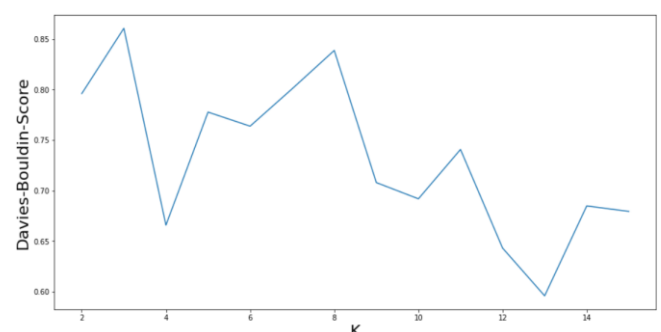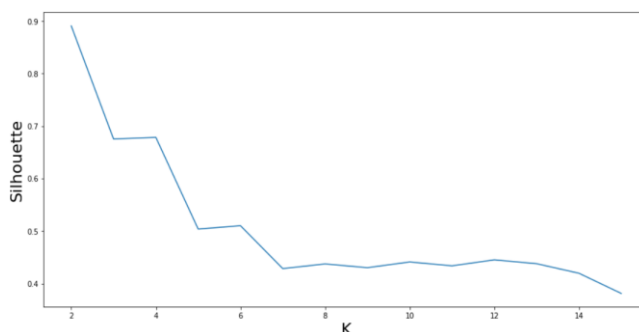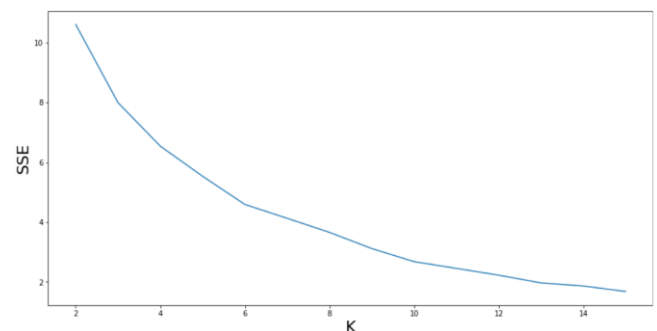

And in fact, the final decision was to consider K = 4, *SSE* = 6.54 and *Silhouette* = 0.68

4.1.3 Analyzing Results

A scatter plot of the clusters over the attributes *Tprod* and *MeanSaleOrder* can quickly give an overview of the results, showing how **2** clusters are located near the origin and seem well defined, a **third** cluster seems to be capturing all the entries farther from the origin apart from one single value (positioned exactly below the rightmost red star) which by itself composes a **fourth** cluster.

By taking a look at the number of customers per cluster we can see that the four clusters (numerated from left to right from **0** to **3**) include {0: **3857**, 1: **25**, 2: **450**, 3: **1**} for a total of **4333** customers.

The presence of a cluster composed of only 1 customer is strange to say the least. More insight on the situation is
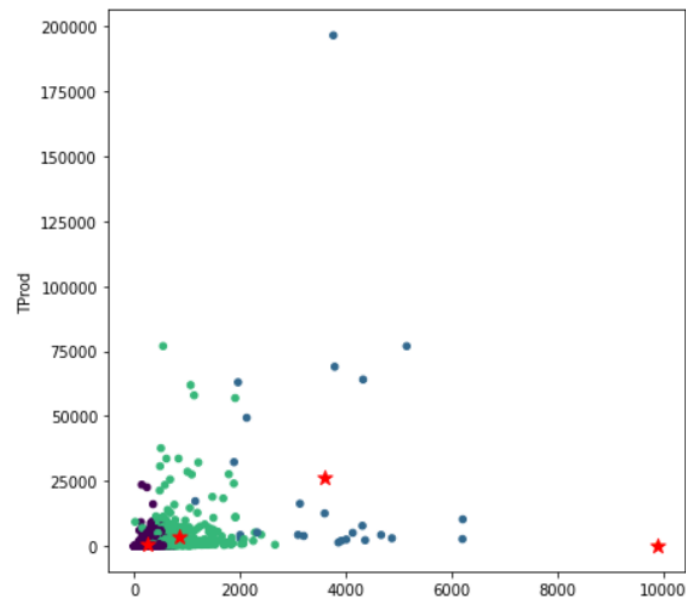

`{0: 3857, 1: 25, 2: 450, 3: 1}`
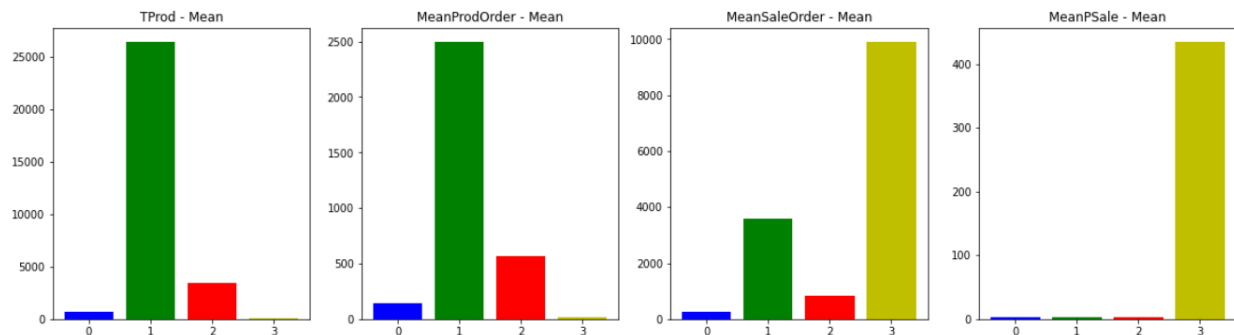
given by the barplot below, which shows how the anomalous cluster has a peculiar behaviour, distinct from the other clusters which seem to follow a certain pattern in every attribute plot.

The suspect of cluster 3 being an anomaly is reinforced by looking at the centroids with and without the presence of said cluster. The trend of the non-anomalous clusters clearly follows a trapezoid shape while the anomalous one follows the trend dictated by its only customer, which is obviously not as robust and as meaningful as the other clusters trend.
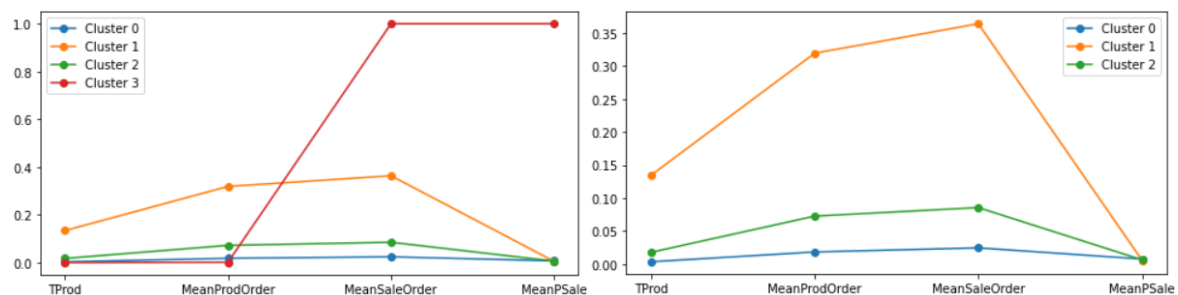
Some **postprocessing** is needed, there are 2 possible alternatives:
- *Merging* the anomalous cluster with the one most similar to it (in this case would be **cluster 1**). This leads to a deep variation in **cluster 1** statistics (e.g in **mean, std** and **max value**), as can be seen in the **before** and **after** below, which is about the *MeanPSale* attribute.

```
count      25.00000          count      26.000000
mean        2.60540          mean       19.222500
std         1.50657          std        84.743774
min         0.40000          min         0.400000
25%         1.04800          25%         1.234000
50%         2.54400          50%         2.647000
75%         3.34900          75%         3.553000
max         5.73100          max       434.650000
```

- *Discarding* the anomalous cluster, which was the definitive decision after the negative results brought by attempting merging.

After discarding the anomalous cluster, the result is composed of **3** clusters, each of them describing a customer behaviour:

- **Cluster 0:** containing the highest number of customers (**3857**, **89%** of the whole population), this cluster can easily define the group of **low-spending** customers, having lowest average total sale (**1188**) and lowest average number of products per order (**143**).
- **Cluster 1:** containing the lowest number of customers (**25**, **0.58%** of the whole population), this cluster can very well define the group of **high-spending** customers, having by far the highest average total sale (**49823**) and highest average number of products per order (**2497**). Interestingly, it is also the cluster with the highest average number of returned products (**490** per customer).
- **Cluster 2:** containing the value in the middle regarding number of customers (**450**, **10%** of the whole population), this cluster can define the group of **medium-spending** customers, having average total sale (**5385**) and average number of products per order (**568**).

```
---------------- Cluster 0 ----------------
          TSale        MinPSale         TRProd
count  3857.000000  3857.000000  3857.000000
mean   1188.337402     9.731426    11.662432
std    1956.851005    31.831756    70.553040
min       0.000000     0.060000     0.000000
25%     268.050000     0.950000     0.000000
50%     569.130000     5.040000     0.000000
75%    1340.150000     9.950000     3.000000
max   39888.160000   613.200000  2022.000000
---------------- Cluster 1 ----------------
          TSale        MinPSale         TRProd
count    25.000000     25.00000    25.000000
mean  49823.296800    590.19840   489.840000
std   80039.024866   1199.56193  1552.729658
min    2002.400000      0.39000     0.000000
25%    4314.720000      3.26000     0.000000
50%    7829.890000     25.44000     0.000000
75%   53258.430000    138.24000   288.000000
max  278571.620000   3861.00000  7714.000000
---------------- Cluster 2 ----------------
          TSale        MinPSale         TRProd
count   450.000000   450.000000   450.000000
mean   5384.749600    19.783978    78.093333
std   11974.328237    70.853207   474.129177
min     164.320000     0.140000     0.000000
25%     926.325000     2.100000     0.000000
50%    2000.925000     5.040000     0.000000
75%    4235.545000     9.900000     2.000000
max  132649.700000   931.500000  6420.000000
```

{0: 3857, 1: 25, 2: 450}
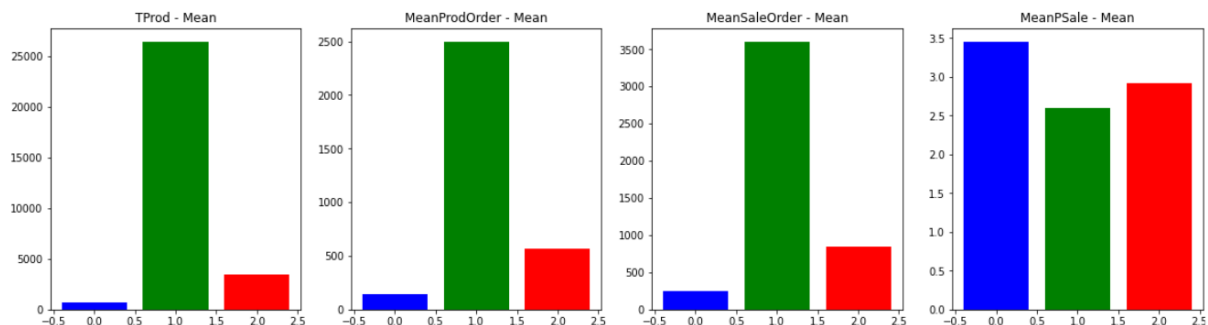
4.2 DBSCAN

The dataframe used for DBSCAN is the same used for K-Means, with the same selected subset of attributes. The distance between data points is computed using the *Euclidean* metric, resulting in a square matrix containing the pairwise distance between all pairs of points.

kth_distances is a dictionary having as keys all the integers between *Kmin* and *Kmax* (3 and 30) and as values it has lists containing the distance of the k-th nearest neighbour of each point.
For example:
- k = 3 → list containing the distance of each point from its 3rd nearest neighbor
- k = 4 → list containing the distance of each point from its 4th nearest neighbor
- and so on...

Each of these lists is sorted and plotted, making possible to notice that the curve of the distance behaves almost in the same way regardless of the value of K.
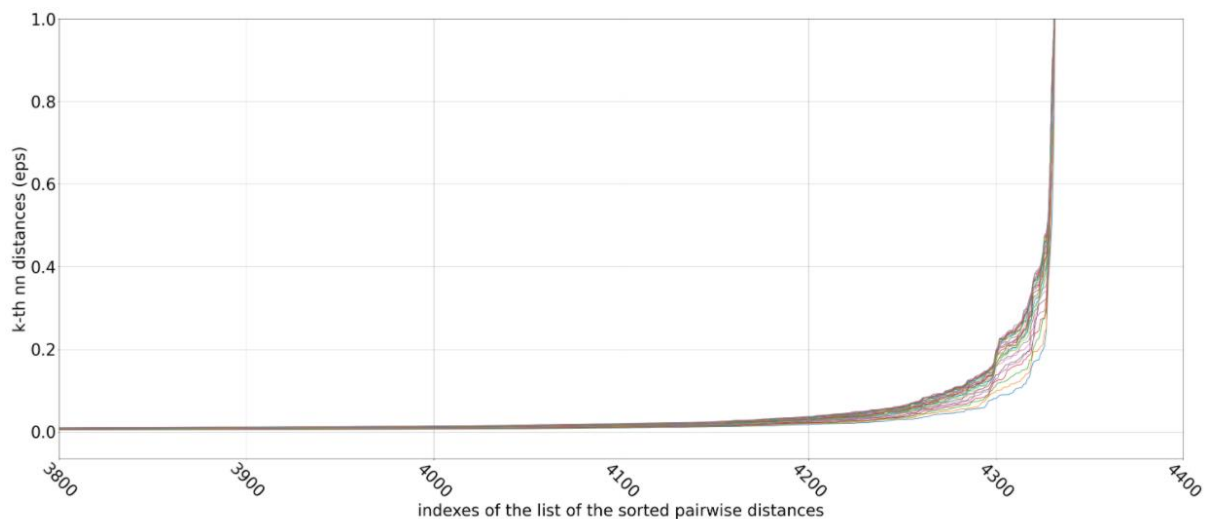
In order to visualize it as a single curve, there is a plot where, to each value in the x axis, is associated the median of the values in x of the *Kmin-Kmax* curves. The elbow is located between x=**4000** and x=**4200** with the corresponding values of eps between **0.0118** and **0.0313**.
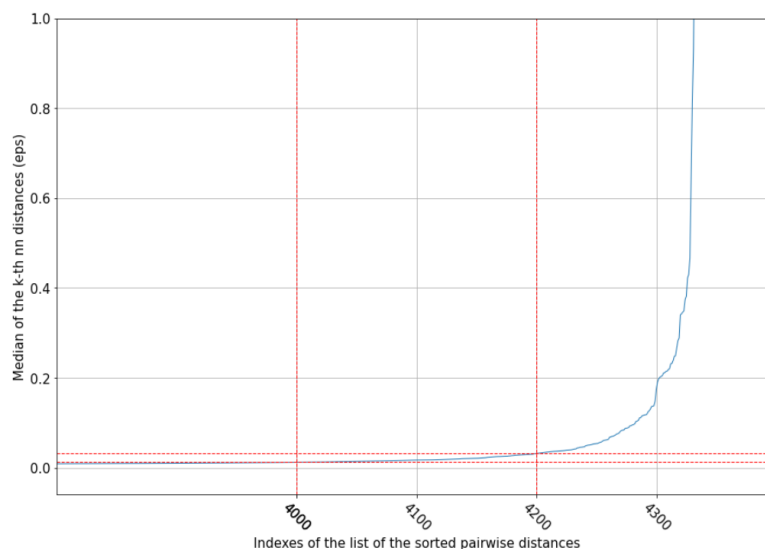
At this point the grid search method is exploited to find the best values for the two parameters of the DBSCAN algorithm: *epsilon* and *min-samples*.
To optimize the search:

- it is applied only on those values of eps that are within the *elbow window*
- Since for *min-samples* greater than **8**, only a single cluster is detected, the grid search is restricted on the values between 3 and 9

As result, the following heatmaps are produced:
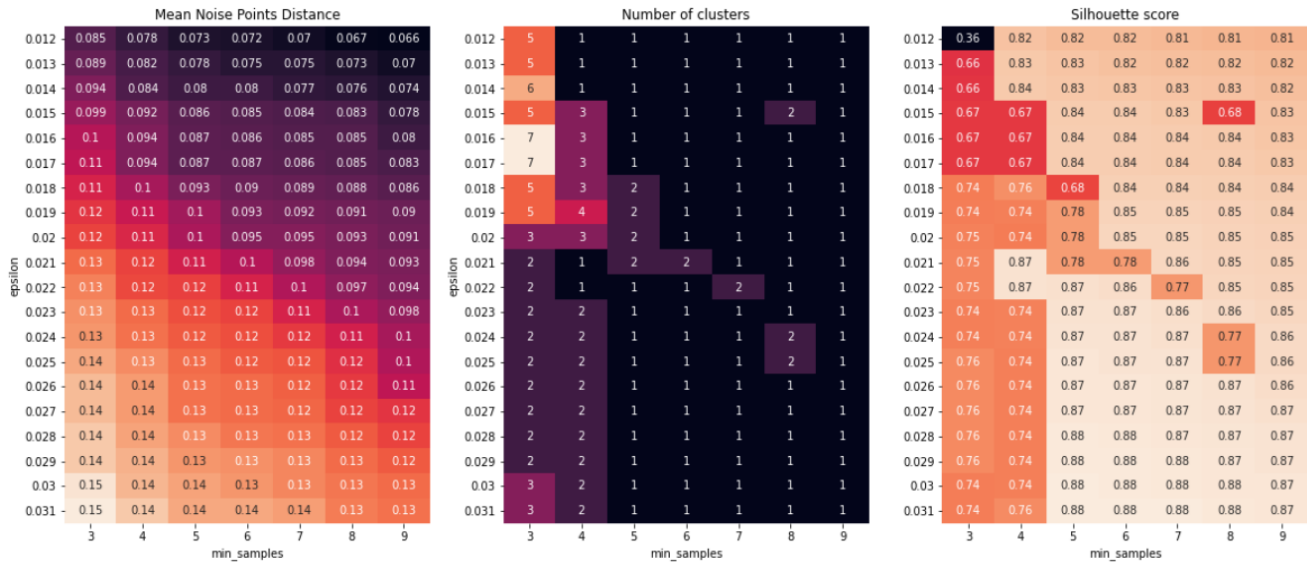
The best combination of parameters seems to be: *eps* = **0.02** and *min-samples* = **4**.
This will produce 3 clusters distributed as follows:

- Cluster 0: **4223** points
- Cluster 1: **4** points
- Cluster 2: **9** points
- There remain **97** noise points

Silhouette score: **0.74**
The result of this clustering algorithm applied to this dataset is very unbalanced, and different choices of the parameters do not improve the situation.

4.3 Hierarchical Clustering

4.3.1 Attribute choice and distance methods

For the clustering via Hierarchical Clustering the same set of attributes of the previous algorithms were used, in order to get results comparable in terms of indicators and properties among the clusters. For all the different runs of the algorithm the metric was the euclidean one since it better fits our data. The analysis goes through different methods like *complete, single, average* and *ward*, each of them changes the way in which the distance between two clusters is calculated. The first three are based on mathematical functions while the last one has a more statistical approach.
Through the study of the dendrograms the best value for the threshold was chosen with the objective of getting a balanced clustering result. In order to evaluate the goodness of the clustering we calculated the silhouette coefficient.

Method	Clusters	Silhouette
Complete	4311, 20, 1, 1	0.922
Single	4328, 1, 1, 1, 1, 1	0.935
Average	4312, 17, 1, 1, 1, 1	0.92
Ward	3853, 458, 22	0.672
Ward	3858, 458, 21, 1	0.672

Even if the silhouette is very high, the first three methods' results are pretty bad since the clusters are unbalanced, collapsing all the elements in one single cluster. With *ward method* we get the best results both from the point of view of the number of elements in the different clusters and coherence of values among clusters too.

4.3.2 Analysis of dendrograms using different methods
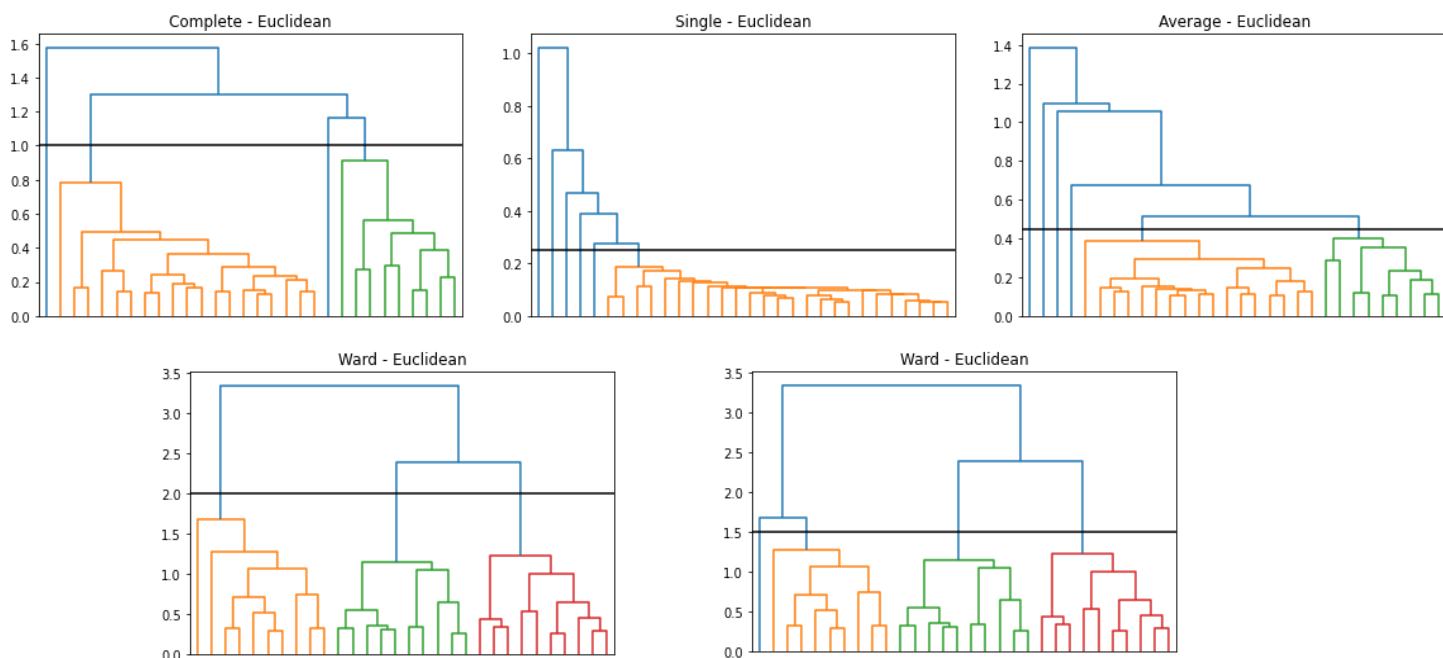
As expected from the theoretical formulation of the different methods, *complete* joins clusters at a higher value than *single* and *average* since it evaluates the distance - between clusters - as the maximum distance between points of two different clusters. From the dendrograms and the results previously shown, the first three methods are not able to create a good and balanced clustering, all the items are collapsed in a singular big cluster. With Ward's method, instead, the result is more balanced and it is possible to study a characterization of the elements inside the three main clusters. The results are similar to the ones obtained from K-Means, the three clusters seem to represent the three different types of customers: low, medium and high spending (keep in mind the last one can include outlier elements, which are customers who don't carry enough information for a good understanding).

4.4 Fuzzy C-Means

4.4.1 Introduction to the algorithms

This alternative clustering approach is part of the family of so-called fuzzy clustering algorithms, which are based on the idea that an element can belong to more than one cluster (soft clustering). This idea is exploited using various coefficients, where each of them represents the probability of the element to be part of a certain cluster. The set of coefficients (one per cluster) is also known as membership matrix.

The algorithm in its implementation is very similar to K-Means:

- Choose C (K) number of clusters
- Initialize centroids, either randomly or by some criteria (**K-Means++** in this case)
- Compute membership function for each element and calculate new centroids
- Repeat the previous step until stopping criteria is reached

Once the algorithm terminates each element will be assigned to a cluster by looking at its highest membership values.

4.4.2 Analysis of results

It is true that the algorithm is different with respect to K-Means (soft vs hard clustering), but the underlying idea is pretty much the same. With this bias the decision regarding the number of clusters value (K) was straight forward: either 3 or 4.

Choosing the number of clusters to be equal to 4 brings a new clustering result. The resulting clusters are composed respectively of **3086, 24, 1086** and **137** customers**,** giving a substantially different result from the ones reported by K-Means and Hierarchical Clustering. It is a more smoothed out result, as to be expected by such a *soft* approach. Furthermore, by taking a look at the bar plots, it is plausible to semantically differentiate the clusters by individuating four possible behaviours: **low-spending (cluster 0), medium-spending (cluster 2), high-spending (cluster 3)** and **extremely high-spending (cluster 4).** Talking about metrics this clustering has a Silhouette of roughly **0.5** and a Davies-Bouldin Index of **0.98.**
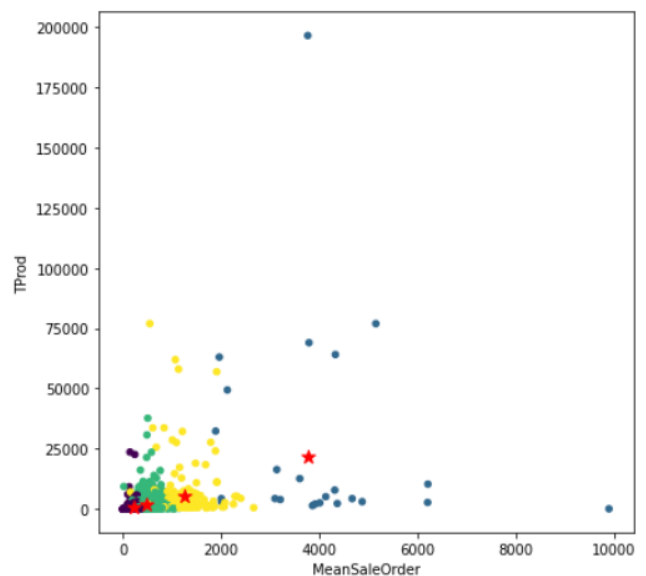
[0:3086] [1:24] [2:1086] [3:137]
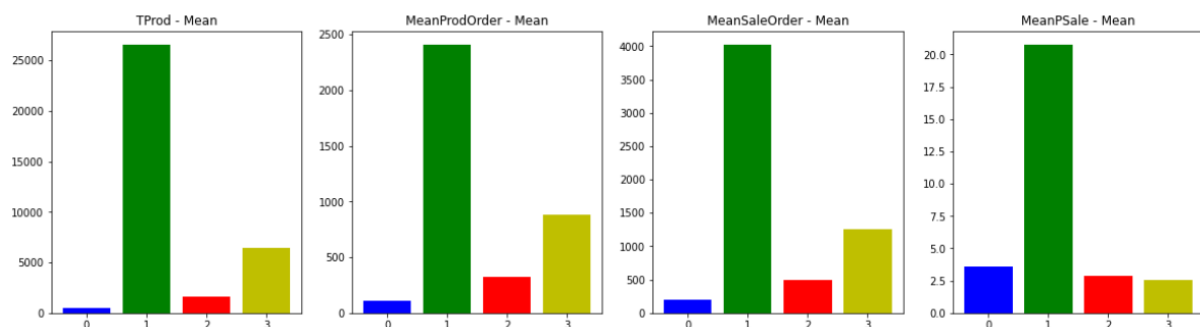
Choosing the number of clusters to be equal to 3 (and deleting the **anomalous value** already found using K-Means) brings a clustering result very similar to the ones already portrayed by K-Means and Hierarchical Clustering. This means that the results suggest a distinction between the different clusters due to the kind of customers they contain (once again **low, medium** or **high spending**). One difference that can be easily highlighted between the Fuzzy and the K-Means result stays in the number of elements per cluster, which are different due to the algorithm being *soft*, tending to include elements which would not be included in the same cluster if the clustering algorithm was a *hard* kind.

Talking about metrics this clustering has a Silhouette of roughly **0.62** and a Davies-Bouldin Index of **0.86.** Confronting the two clustering (K = 3 and K = 4) the better metrics lies in the 3-cluster result.

[0:3681] [1:41] [2:610]
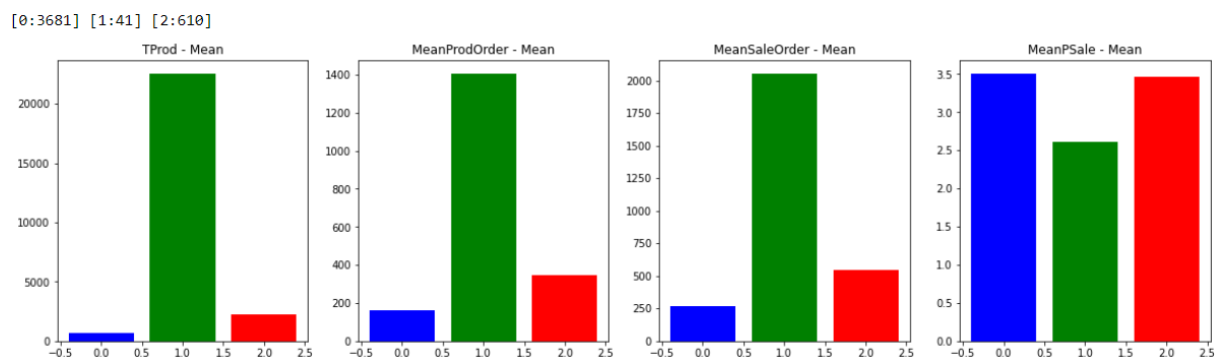
4.5 Clustering genetic algorithms

4.5.1 Introduction to the algorithms

To go even further beyond in the analysis of the data a completely different approach was considered. This alternative method tries to combine GA (genetic algorithms) to clustering in order to create a hybrid technique that exploits the ability of exploration of GA to find the best possible cluster that fits the data. The problem of clustering is easily reinterpreted in a genetic point of view through the following mapping of the elements:

- *Population*: represents a set of L different subdivision of the elements in K clusters
- *Mutation rate*: probability of an element to be assign to a random cluster
- *Fitting function*: metric to evaluate and find the best representation, in this case will be based on statistics, such as variance, among the elements in the cluster
- *Evolution step*: choose the best 2 models from the population, crossover them to get the new model for the new generation

Given this formalization the algorithm will start with a random population and after several steps it will have reached the best possible clustering of the data.

4.5.2 Analysis of results

This type of approach suffers from the fact that it is not possible to estimate if it will converge to a solution, and, if yes, after how many iterations.

In our case we tried with **500** iterations, **50** elements in the population and **3** clusters and **5** elements mutated in each child. The plot represents the best model that was obtained from the algorithm. It is clear that the classification of the elements is not balanced and it is not possible to distinguish different elements of different clusters, so it seems like it did not reach any good results.

This may be due to various reasons:

1. The number of iterations is too low, given this type of approach we can expect that we need an high number of generations in order to reach a good result (implementation needed too much time and space)
2. The data is a bit noisy so this may cause bad evaluation of best model
3. Better initialization of the population since it is generated randomly but the result should follow a specific pattern. An initialization that meets the objective would be better in terms of results and convergence

Given the analysis results a more guided approach to this method might give better outcomes.
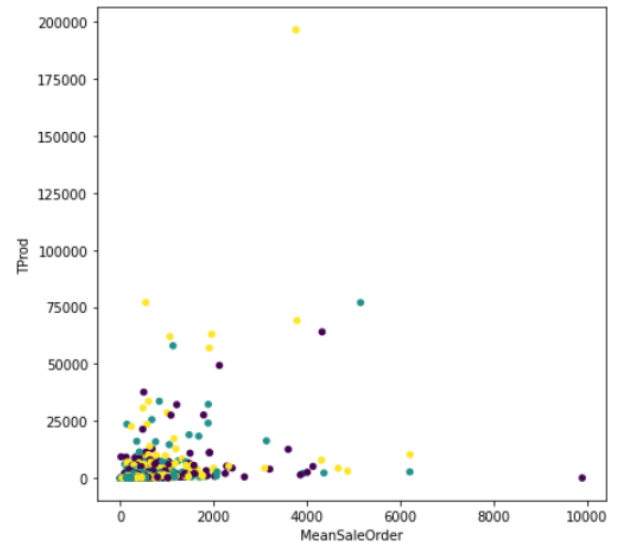
4.6 Final consideration over clustering

In conclusion many clustering algorithms have been applied to the constructed customer dataset, some brought results, some brought interesting results, some unfortunately were not useful in this case. More specifically:

- *K-Means* needed some postprocessing to delete a singular anomalous cluster but apart from that it produced a concrete and meaningful clustering, differentiating three possible customer behaviours.
- *Hierarchical Clustering* provided a plethora of possible solutions, proving itself not useful in this case when using *complete, single* and *average* methods. Nevertheless, the solutions provided using *Ward* method were incredibly similar to K-Means, pointing and hinting to the same possible customer behaviours.
- *Fuzzy C-Means* looked at the problem in a similar way compared to K-Means, but at the same time it turned the table using a *soft clustering* approach which resulted in smoother results. Concrete and meaningful clustering were its outcomes, outputting both a 3-cluster solution (highly similar to the K-Means/Hierarchical Clustering) and a 4-cluster solution. By comparing the metrics, the 3-cluster result is better than the 4-cluster one, leading once again to the approximate same solution as K-Means.
- *DBSCAN* unfortunately gives vastly unbalanced results, hinting that the dataset at hand may not be perfect for this kind of algorithm.
- *Clustering Genetic Algorithms* as highlighted before fail to get an appreciable result due to both the dataset values topology and to the tuning of the algorithm parameters