Università di Pisa

Data Mining Project - Report

Analysis of a Supermarket's Customers

Valerio Mariani (560014)
Antonio Strippoli (625044)

A.Y. 2020/2021

Contents

1 Data Understanding

The dataset is given as a data matrix containing **471.910 rows**, where each record corresponds to a product in a basket. Excluding row number, the dataset has a total of **8 attributes**, where all the numerical attributes are *discrete*.

Attribute	Dtype	Non-Null Count
BasketID	*object*	471.910
BasketDate	*object*	471.910
Sale	*object*	471.910
CustomerID	*float64*	406.830
CustomerCountry	*object*	471.910
ProdID	*object*	471.910
ProdDescr	*object*	471.157
Qta	*int64*	471.910

Table 1.1: *First overview of the dataset obtained through the Pandas' method* `DataFrame.info()`*. We can immediately note non-null values and some unexpected "object" data type for some attributes.*

1.1 Data Semantics

In this section we will briefly analyze the semantics of each attribute. Below, we provide their **descriptions**, as well as their **Level of Measurement**.

- **BasketID** (*Nominal*): basket identification code.
  We identified and interpreted three different categories based on the code:
  - *Numerical only*: the customer pays the items in the basket;
  - *Starting with 'C'*: the customer receives some form of money, either given by discounts or for returned items;
  - *Starting with 'A'*: service information reporting 'Adjust bad debt', not a real purchase;
- **BasketDate** (*Ordinal*): date-time of the purchase;
- **Sale** (*Ratio-Scaled*): unit price of the product. It is always positive, except some entries for service information contain it as a negative value;
- **CustomerID** (*Nominal*): numerical identifier of the entry's customer;
- **CustomerCountry** (*Nominal*): country where the customer probably lives;
- **ProdID** (*Nominal*): identifier of the purchased product. Generally it is a numerical code with optional trailing characters, but some particular products use strings without numbers (like discounts or bank charges);
- **ProdDescr** (*Nominal*): description of the product (mainly in uppercase);
- **Qta** (*Ratio-Scaled*): amount of purchased products. It is positive for numerical only BasketIDs, negative for those starting with 'C';

1.2 Data Quality

Table 1.1 already gave us some previews about the data quality, which we further analyzed and improved.

We noticed that there were only 2 ***BasketIDs*** starting with 'A', which we decided to drop, as they were service information. Instead, those starting with 'C' always had negative quantities, so we had two ways to identify the same basket's category. We decided to remove the starting 'C' character, since this way we were able to treat this attribute as integer and lighten the dataset.

We performed a consistency check on ***BasketDate*** but we found no issue with the data, so we simply told Pandas to parse *BasketDate* column as a *DateTime*. We also noticed some entries having the same *BasketID* but *BasketDate*'s values differing of 1 minute at most, which led us to rename the attribute as *PurchaseDate*, since we think it is more appropriate.

We checked the ***Sale*** attribute, which was not recognized by Pandas as a *float* due to the floating numbers being written using a comma separator instead of point. To accommodate the *.csv format*, we performed a **replace**. In addition, we considered invalid and dropped every entry with absolute value of *Sale* less than 0.01.

Pandas recognized ***CustomerID*** values as *floats*, but we found no reason for them to be floats apart from having a trailing '.0', so we **converted** them to *int* in order to lighten the dataset. Moreover, we dropped entries with missing *CustomerID*, since they would not be helpful for customers' profiling.

Interestingly, we found that some entries had the same *CustomerID* but different ***CustomerCountry***. Based on our previous guess of CustomerCountry meaning, we decided to not change these attribute's values.

We noticed that some ***ProdIDs*** containing alphabetical characters were inconsistent (the same product showed sometime uppercase letters, others lowercase letters). We found out that this attribute is case insensitive, so we put everything in uppercase.

Pandas pointed out that ***ProdDescr*** had some *missing values*, which we could not fill since other entries having the same *ProdID* did not have a *ProdDescr* as well. Anyway, we decided not to drop them as they could still contain useful information. Some non-empty descriptions were really useful to determine strange entries, like "test" or "check?", which we decided to drop. Moreover, we noticed that entries had *unnecessary trailing whitespaces* or were not in uppercase, so we standardised them.

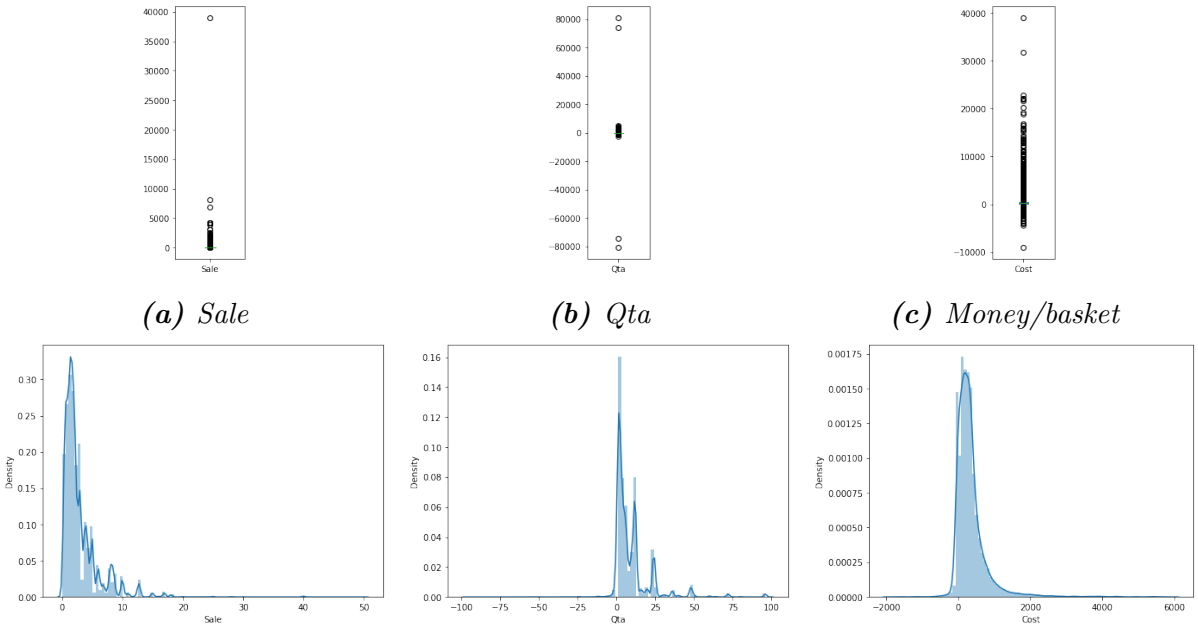
By going further with our analysis, we observed that some baskets contained a **repeated product**, which made no sense due to the existence of the *Qta* attribute. However, while some of those products had the same sale, others had not. So, we aggregated those with the same sale, while keeping the others.

We also found out that some entries sharing the same ***ProdID*** did not always share the same ***ProdDescr***. We decided to uniform those descriptions with the longest one, since we thought it was more informative (e.g. *FLOWER FAIRY 5 DRAWER LINERS* → *FLOWER FAIRY,5 SUMMER B'DRAW LINERS*).

Finally, we searched for tuples corresponding to **returned objects that have never been bought** according to our data, dropping these records.

After these preliminary cleanups and fixes, we searched for **outliers** over *Sale* and *Qta* attributes, as well as over the *total number of articles and sale for each basket*.

Since we did not find any outlier in the number of articles per basket, in Figure 1.1 we only depicted three **box plots**, that highlight the presence of the outliers together with their respective **distribution histograms**. Note that the plots have been produced consecutively (from left to right), after the removal of the outliers of the previous category.



(a) *Sale* ***(b)*** *Qta* ***(c)*** *Money/basket*

(d) *Sale* ***(e)*** *Qta* ***(f)*** *Money/basket*

Figure 1.1: *Boxplots (a, b, c) to identify outliers in Qta, Sale and BasketID attributes, along with their distributions (d, e, f).*

For each category we studied the applicability of **IQR** as method to remove outliers, as well as the distribution of the outliers **among the customers**. We have shown that IQR, while cutting off the outliers, also removed entries which we considered to be non-outliers, so we decided to search and define *thresholds* for each category. After defining the thresholds, we noticed that the outliers were distributed among a lot of users, so we only removed the respective entries.

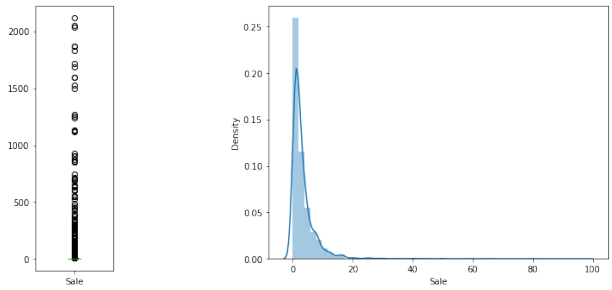
At the end, we checked for duplicated entries in the dataset, with no results. The final dataset has a total of **394.117 entries**, which is **16.48%** smaller than the original one.

1.3 Data Distribution, Statistics & Correlation

After several fixes and cleanups to the dataset, we are ready to perform a statistical analysis to assess the distribution of each attribute and find hidden information in the dataset. First, we computed the **Pearson correlation** matrix, finding no explicit correlation between any pair of attributes. Then, we started analyzing the cleaned-up ***Qta*** and ***Sale*** values.

We got the supermarket's **product catalog with the respective price**, which are the unique pairs (*ProductID*, *Sale*). Those *Sale* values now vary between 0.01 and 2118.74, with a mean value of 10.72 and a standard deviation of 85.67. We think we got appropriate values for the articles of a supermarket, since most of the articles are not pricey.

We represent its distribution histogram and box plot in Figure 1.2.



(a) Box Plot *(b) Distribution*

Figure 1.2: *Statistics for sales in the unique catalog of products.*

The ***Qta*** attribute now has a total of 387.823 positive values against 6.294 negative, which is a really unbalanced distribution between the number of buys and returns. The positive values now range between 1 and 3.186, with a mean value of 12 and a standard deviation of 41. Instead, the negative values range between -1.350 and -1, with a mean value of -11 and a standard deviation of 48.

In Figure 1.3, we represent a scatter plot that highlights the **relationship between the *Sale* attribute and the *Qta* Attribute**. We can see that the priciest products are bought/returned in small quantities, while cheaper products are usually bought/returned in large quantities.

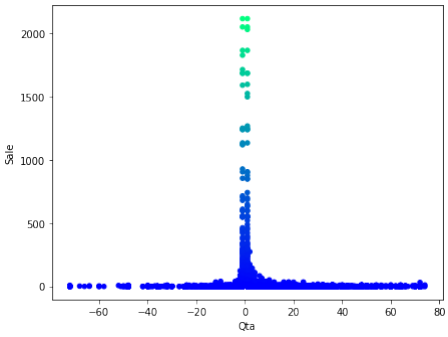


Figure 1.3: *Relationship between Sale and Qta (colors are not relevant).*

Next, we analyzed the **total profit per month** of the supermarket (considering both buys and returns), as well as the **monthly number of baskets**. The bar plot comparing the values (in percentage) is reported in Figure 1.4.

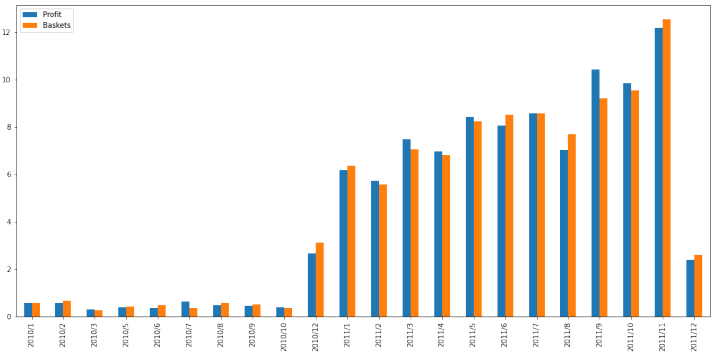
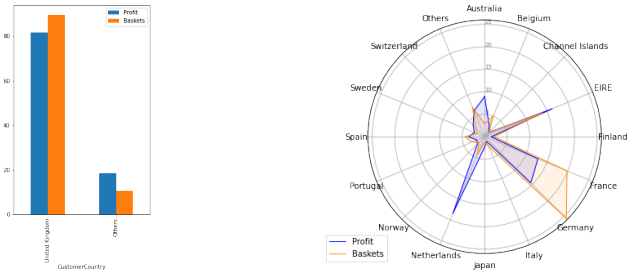


Figure 1.4: *Monthly statistics of the supermarket chain.*

As we can see, the two plots have a very similar pattern. In fact, we found a **correlation of 0.995** between these pairs of values, which is a further confirmation of the quality of the dataset. Moreover, from these plots we can actually see that the supermarket had *little activity* (and so *little profit*) in the year 2010, while it had a sales boom in the year 2011. Regard the year 2010, it could also be that the supermarket chain *did not provide* all the data.

Since the supermarket has **customers from different countries**, we analyzed again the profit and number of baskets, this time with respect to each country. We discovered that the majority of the profit comes from the *United Kingdom* (>80%), which made hard to read a unique plot containing every country. So, we firstly produced a plot comparing *United Kingdom* against all the other countries, then we did a second plot to study the *Others* category of the first one. Plus, in the second plot we aggregated in *Others* the countries with *less than 25 baskets*, to improve readability. The two plots can be seen in Figure 1.5.



(a) *UK - Others* ***(b)*** *Less popular Countries*

Figure 1.5: *Per country statistics of the supermarket.*

As yet another proof of the data quality, we found again an high correlation between *Number of baskets* and *Profit*, with a value of **0.999**. The three most profitable countries are *United Kingdom*, *Germany* and *France*:

- *United Kingdom* had a profit of **3.590.307,81** and a total of **15.601** baskets;
- *Germany* had a profit of 131.454,01 and a total of 466 baskets;
- *France* had a profit of 102.453,09 and a total of 343 baskets;

We think it is unlikely to have such huge gaps between the most profitable country and its successors, so we decided to investigate on the **countries' activity** during the timeline available, which we can see in Figure 1.6.

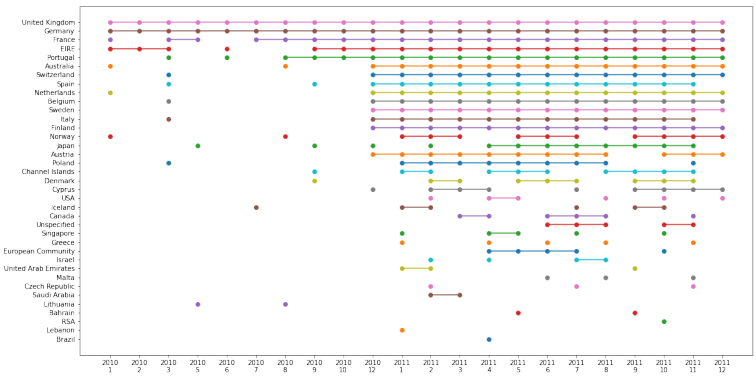
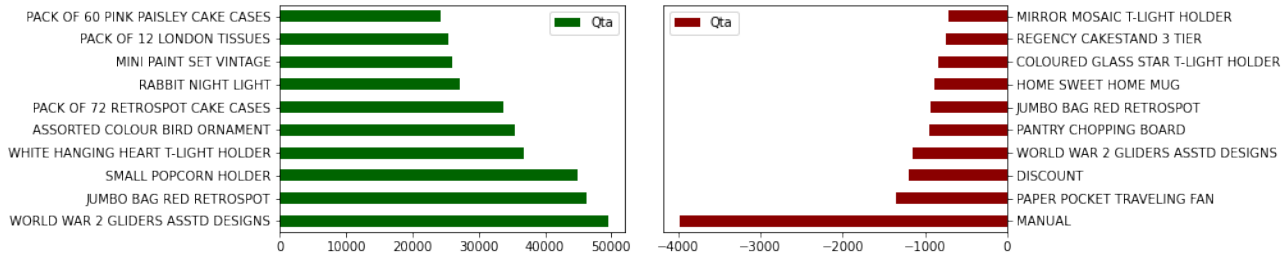


Figure 1.6: *Compared view of each country monthly activity.*

We found out that customers from several countries have been less active in a variety of months, particularly in *2010*. This analysis helps explaining such a poor profit for most of the countries: it is possible that customers from those countries were just passing through, as it is also possible that our dataset is not complete and we miss some data.

As our final statistical analysis, we searched for the 10 **most bought/returned products** over the 3617 in the dataset, which we resumed in Figure 1.7.



(a) *Popular products of the supermarket.* ***(b)*** *Most returned products of the supermarket.*

Figure 1.7: *Per country statistics of the supermarket.*

It is interesting to note that this supermarket sells mostly fortuitous items, which we decided to analyze in details: in Figure 1.8, we computed the occurrences of the top 10 most frequent words in the products' description. This is a further confirmation that **the dataset is not about an every-day supermarket**, which is something we will take into consideration in our next analysis.

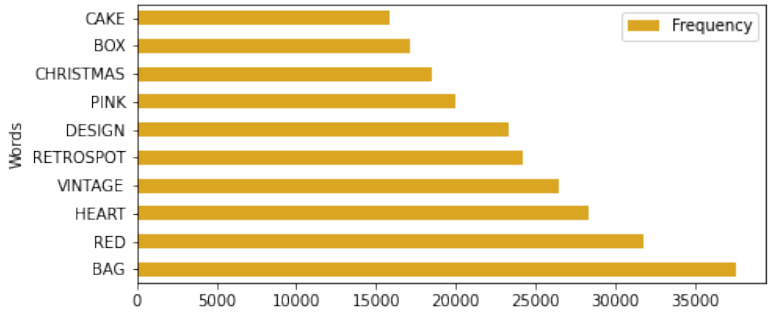


Figure 1.8: *Top 10 most frequent words in the products' description.*

2 Data Preparation

After the analysis and improvements made to the given dataset, we built a new dataset aimed at profiling the customers. This process yielded a dataset containing **17 columns** and a total of **4.336 entries**, where each row corresponds to a *registered customer* (indeed, the dataset is indexed by the *CustomerID* attribute).

2.1 Data Semantics

We computed the *4 requested attributes*, as well as *16 more attributes*. Below, we will briefly analyze the semantics of each attribute, providing their **descriptions** and their **Level of Measurement**.

Requested attributes:

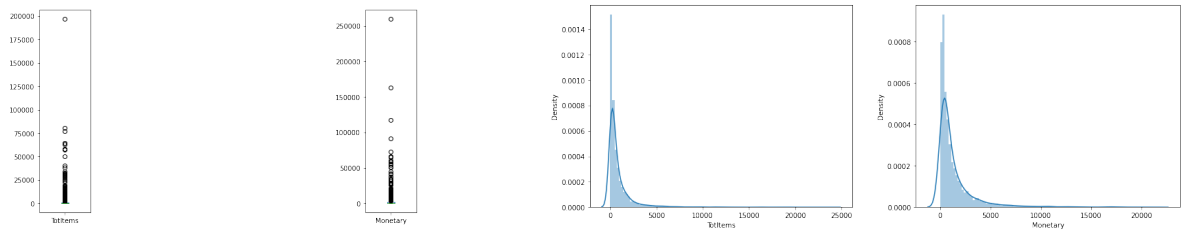
- **TotItems** (l) (*Ratio-Scaled*): the total number of items purchased by a customer during the period of observation;
- **UniqueItems** (lu) (*Ratio-Scaled*): the number of distinct items bought by a customer in the period of observation;
- **MaxItems** (lmax) (*Ratio-Scaled*): the maximum number of items purchased by a customer during a shopping session;
- **E-Prods** (E) (*Ratio-Scaled*): the Shannon entropies on the purchasing behaviour of the customer, based on diversity of the purchased items;

Additional attributes:

- **Recency**, **Frequency**, **Monetary** (*Ratio-Scaled*): quantitative factors often used as marketing analysis tool. They are respectively the number of days since last purchase, number of days the user went to the shop and total amount of money spent;
- **MeanBasketItems and PReturn** *(Ratio-Scaled)*: more attributes concerning the number of items, respectively medium number of objects in basket and percentage of the objects returned;
- **MaxSale, MeanBasketSale, MeanItemSale** *(Ratio-Scaled)*: attributes concerning amount of money spent, respectively max of baskets' amount, medium baskets' amount and mean unitary price of items bought;
- **E-Sale, E-Baskets, E-Intervals** *(Ratio-Scaled)*: more entropies capturing customer's behaviour. They are calculated respectively on the sales' diversity of purchased items, on the baskets' amount diversity and on the diversity of the inactivity intervals. Note that in order to compute meaningful entropies, we performed some binning based on quartiles;
- **PurchasingFreq, WeekDayPref, WeekMonthPref** *(Ratio-Scaled)*: attributes concerning time, respectively the mean purchasing frequency of the user (in days) and day of the week/week of the month in which a customer buys most of his items;
- **ItemPref** (*Nominal*): ID of the item that is most frequently bought by the customer;
- **MainCountry** (*Nominal*): main country where the customer probably lives;

2.2 Data Quality

We searched once more for outliers, since we computed new attributes. We found out a minor amount of outliers in **TotItems** and **Monetary**, which we decided to remove setting thresholds. In Figure 2.1, we visualized the box plots of these attributes to highlight the presence of outliers. After the removal, we have a total of **4.329 entries**.



(a) TotItems *(b) Monetary* *(c) TotItems* *(d) Monetary*

Figure 2.1: *Boxplots (a, b) to identify outliers in TotItems and Monetary attributes, along with their distributions (c, d).*

2.3 Data Distribution, Statistics & Correlation

After the outliers removal, we performed a statistical analysis to find hidden information in the new dataset. First, we computed the **Pearson correlation** matrix, depicted in Figure 2.2.

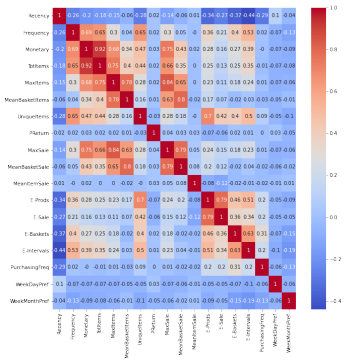
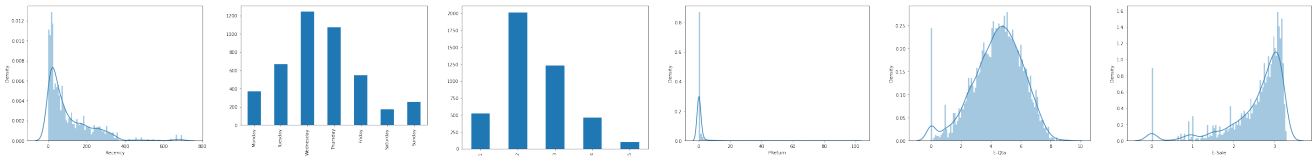


Figure 2.2: *Heatmap of Pairwise correlations between numerical attribute in the customer dataset.*

We only found a single high correlation which is noteworthy, and it is between *E-Prods* and *E-Sale* with a value of **0.79**: as users buy different items, it is more likely that those items have different sales. Other high correlations like **0.92** between *Monetary* and *TotItems* reported us the absence of relevant anomalies: as the users buy more items, the price will be higher.

Then, we have investigated on some interesting **distributions**, which we depicted in Figure 2.3.



(a) Recency *(b) WeekDay* *(c) WeekMonth* *(d) PReturn* *(e) E-Qta* *(f) E-Sale*

Figure 2.3: *Histograms of the most interesting distributions of the CustomerProfilation dataset.*

First, we looked at the distribution of **Recency** attribute (2.3a), that shows us how the vast majority of the customers have returned to the shop in the last year. Moreover, we have a drop around 400, which corresponds to the low number of baskets in the year 2010 (see Figure 1.4).

Then we studied the most popular **periods of time** (2.3b and 2.3c). We can see that the most popular day is *Wednesday*, and the most popular week of the month is the *second*. We found unlikely for the customer not to purchase much from the shop during the week-ends: this increases our suspicion about this supermarket not being a regular one.

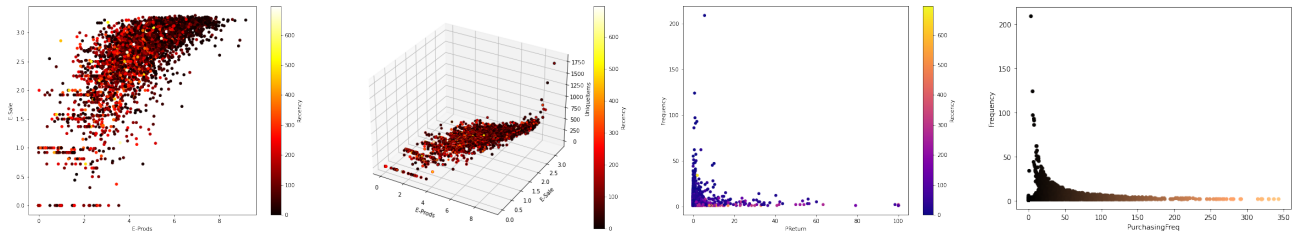
Next we looked at the **PReturn** attribute (2.3d): most of the customers return a little part of their purchases, which we think is a nice statistic for a supermarket.

Finally, we studied the distribution of the **two entropies** (2.3e and 2.3f), interestingly noting how the first one resembled a normal distribution. We have already seen that this couple of attributes is highly correlated, but we decided to further investigate their correlation.

In Figure 2.4a, we can see how the **two entropies** are in some kind of logarithmic relation, and how **Recency** values are lower as the entropies grow: this could be related to the fact that the supermarket has possibly expanded the catalog, allowing the customers to buy a wider range of products[1]. To deepen the study, we produced a 3D scatter plot in Figure 2.4b by adding the **UniqueItems** attribute, to show that the two entropies are in logarithmic relation with the *UniqueItems* attribute, and so that newer customers tend to buy higher numbers of unique items.

We also found interesting the relationship between **Frequency, PReturn** and **Recency**, which we visualized in Figure 2.4c. We can highlight that only customers who shopped less (low *Frequency* value) tend to return a lot of the items they buy. Moreover, a large number of them are old users, which could mean that the supermarket might had and solved issues related to some products in the past.

At the end, we analyzed the relationship between **Frequency** and **PurchasingFreq**, visualized in Figure 2.4d: customers who came back many times to the supermarket have returned in few days; instead, those who returned few times took several days to return.



(a) *Entropies, Recency.* ***(b)*** *Entropies, Recency, UniqueItems.* ***(c)*** *Frequency, PReturn, Recency.* ***(d)*** *Frequency, PurchasingFreq.*

Figure 2.4: *Interesting relationships between the customer profiles' attributes.*

[1]This fact could also be noted from the heatmap in Figure 2.2, that reported a negative value of correlation between the two entropies and the Recency attribute

3 Clustering

In this chapter we discuss **3 different clustering algorithms** (K-means, DBScan and Hierarchical), fine-tuning their hyperparameters and evaluating results from both the statistical and semantical point of view.

We evaluated results based on a total of **5 set** of attributes:

- {*(Recency, TotItems, Monetary), (Recency, TotItems, MaxSale)*}: selected as the ones we thought were the best based on the **semantical meaning**;
- {*(Recency, TotItems, MeanItemSale), (Frequency, TotItems, MeanBasketItems, MeanBasketSale)*}: selected as the results of an **optimization algorithm** we developed, that aims to maximize the following function using Sum of Squared Error (SSE) and Silhouette (SIL) scores: $SSE * a + SIL * b - \left(\frac{k}{c}\right)^2$, where a, b, c are hyperparameters which we defined based on the domain of each score and k is the number of clusters. The reported two sets of attributes are respectively the best among the sets of **three** and **four** attributes;
- {*(Recency, Frequency, Monetary)*}: selected as quantitative factors often used as **marketing analysis tool**;

We employed these sets of attributes in every clustering algorithm, but the most interpretable results were given by the **last set** of attributes we described, so we will later report only those of that specific set. Moreover, by using only 3 attributes we were able to produce **3D plots** where the clusterization is more evident, which we think is something that should not be underestimated.

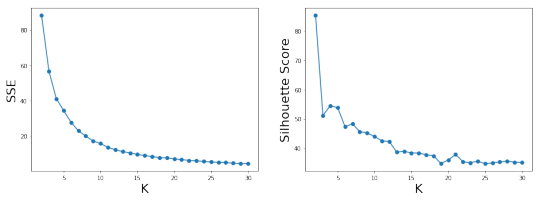
We decided to normalize the values using the **MinMax scaler**, since no attributes we used follows a Gaussian distribution (so *Standard scaler* is not suitable) nor there are any outliers (so *Robust scaler* is not suitable as well).

3.1 K-Means

First, we selected the best value of k through a search in the space of the possible k-values, then we clusterized using the selected hyperparameter and evaluated the results both in terms of centroid-analysis and distribution of the variables within the clusters.

3.1.1 Identification of the best value of k

First, we fine-tuned the **hyperparameter k** by calculating both **Sum of Squared Errors** (SSE) and **Silhouette Score** for k values ranging in *[2,30]*, visualizing the trends in Figure 3.1.

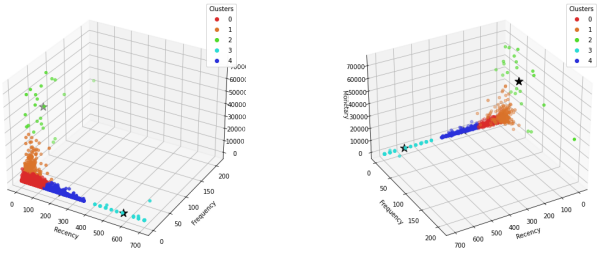


(a) *Distribution* ***(b)*** *Distribution*

Figure 3.1: *SSE and Silhouette Score for each value of k.*

We chose $k=5$ because it is a good compromise between the values of SSE (that we want to **minimize**) and Silhouette Score (that we want to **maximize**).

In Figure 3.2, we present a scatter plot of the clusters together with their centroids.

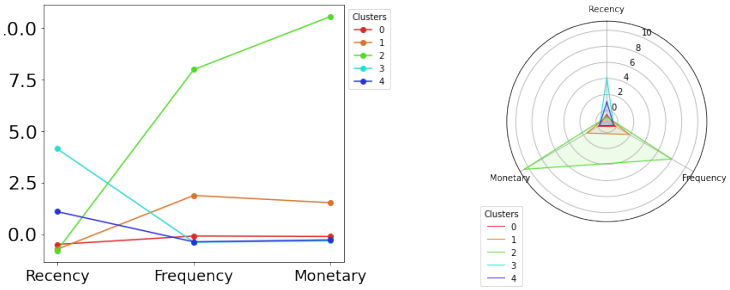


(a) *Distribution* ***(b)*** *Distribution*

Figure 3.2: *Scatter plot of clusterization attributes.*

3.1.2 Analysis of the centroids

To study the computed centroids, we employed parallel-coordinate (Figure 3.3a) and a radar plot (Figure 3.3b).



(a) *Parallel Coordinates* ***(b)*** *Radar Plot*

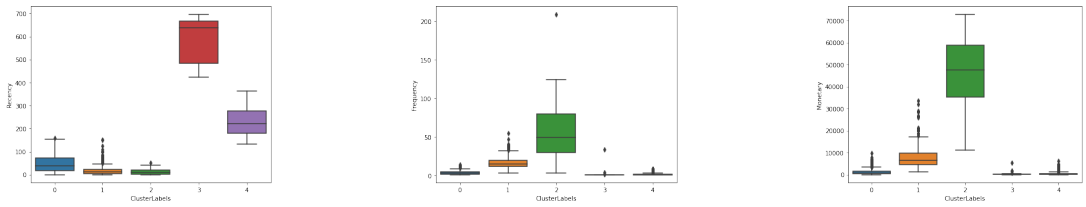
Figure 3.3: *Analysis of the centroids.*

From both the two plots in Figure 3.3 and Figure 3.2, we were able to characterize the clusters from a semantical point of view:

- **Cluster 0**: *new customers*, with every attribute having low values. The supermarket should implement strategies to retain these customers;
- **Cluster 1**: *new wealthy customers*. They're like Cluster 0, but they spend more money;
- **Cluster 2**: *best customers*, they come to the supermarket regularly and they spend large amount of money.
- **Cluster 3**: *non loyal customers*, probably with no interest in the supermarket. We think that the supermarket should ignore them, as they are also a very little amount;
- **Cluster 4**: *averagely inactive customers*. They went to the supermarket few times spending little money, but they're customers from the 2011 year, when the supermarket was most active. The supermarket could further investigate on this cluster to understand why they loose their loyalty;

3.1.3 Distribution of attributes within the clusters

We box-plotted the distribution of the attributes used for the clusterization and grouped them in Figure 3.4. These plots agree with our previous analysis of the centroids, but we thought it was a more interesting way to study those attributes.



(a) Recency *(b) Frequency* *(c) Monetary*

Figure 3.4: *Box-plot of clusterization's attributes.*

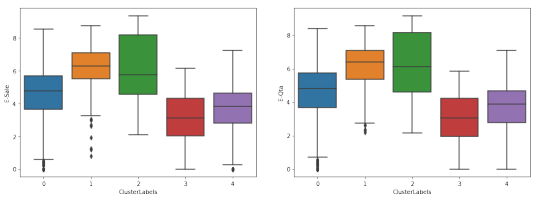
Then, we looked at the **number of entries per cluster** (Figure 3.5a). Cluster 0 constitutes the majority of the customers, which means that a large number of customers is new, while the less populated clusters are 2 and 3, which are respectively rich and inactive users, two class of customers which, indeed, we expected to find in a very small percentage.

In more, we computed the mean value of **MeanItemSale** for each cluster in Figure 3.5b, resulting in Cluster 2 having the highest value (which is to be expected from customers who spent a lot of money).

(a) Clusters' distribution. *(b) MeanItemSale*

Figure 3.5: *Distribution of each cluster along with the averages of MeanItemSale.*

After, we did the box-plots of **E-Qta** and **E-Sale** attributes for each cluster, obtaining the plots in Figure 3.6. We noted that clusters 3 and 4 have low entropies, which is consistent with the information provided in Figure 2.4a, that shows how high values of Recency are associated to low values of entropy in the purchasing behaviour.



(a) E-Sale *(b) E-Qta*

Figure 3.6: *Parallel Coordinates*

Finally, we looked at the items that are **most bought** from the customers in each cluster, the **most frequent country** and the **periods of activity** of each cluster, reporting those information in Table 3.1.

ClusterLabel	Preferred Item	MainCountry	PrefWeek	PrefDay
0	World war 2 gliders asstd designs	United Kingdom	2	2
1	Small popcorn holder	United Kingdom	2	2
2	Jumbo bag retrospot	United Kingdom	2	2
3	Mini paint set vintage	United Kingdom	2	1
4	World war 2 gliders asstd designs	United Kingdom	2	3

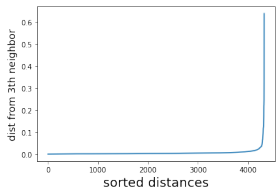
Table 3.1: *Preferred Item of each Cluster*

3.2 DBScan

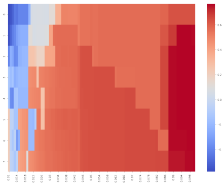
3.2.1 Study of the clustering parameters

For the DBScan algorithm, we firstly used the knee method to identify a range of possible values of the *eps* parameter, and plotted the results in Figure 3.7a. The result showed us that the optimal value of eps approximately lies in the (0,0.1) range.

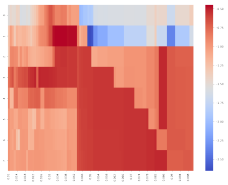
To look for the best eps value in the selected range we performed a **grid search in the hyperparameters-space**, using **silhouette score** and **Davies–Bouldin's score** to evaluate the results of the clustering. We used both the two measures individually and the difference between the two of them, since we have to maximize silhouette and minimize Davies–Bouldin. In Figures 3.7b and 3.7c we represented our results with two heatmaps.



(a) *Distances between points.*



(b) *Silhouette.*



(c) *Davies–Bouldin (Opposite).*

Figure 3.7: *Searches in hyperparameters' space.*

Initially we took the highest values from the heatmap in Figure 3.7c, but we obtained lots of little clusters so we decided to increase min_pts and after some trials we selected 0.046 as epsilon value and 4 as minimum points in a cluster, and using these values for the hyperparameters, we obtained the scatter plots in Figure 3.8.

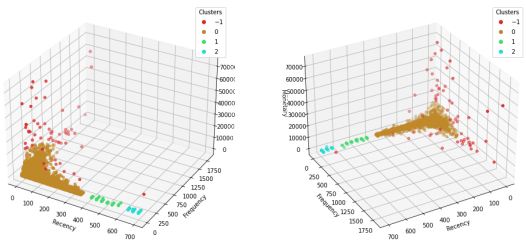


Figure 3.8: *Scatter plot of clusterization attributes.*

The plots in Figure 3.8 show that DBScan algorithm was able to agglomerate the points that KMeans put in clusters 0,1,2,4, exploiting its ability to use density instead of distance to decide whether two points should belong to the same cluster or not.

3.2.2 Characterization and interpretation of the obtained clusters

We represented the distribution of the attributes employed for the clusterization through box-plots in Figure 3.8. It clearly shows that DB-Scan assigned labels 1 and 2 to the inactive customers, label -1 to the best customers, and label 0 to regular customers (the great majority of the customers, as shown in Figure 3.11). Also, we plotted the number of customers assigned to each label in Figure 3.11, and wasn't shocked when we found out that the great majority of the customers belong to cluster 0.

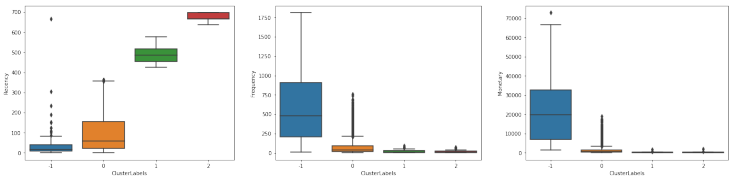


Figure 3.10: *Box plot of clusterization attributes.*

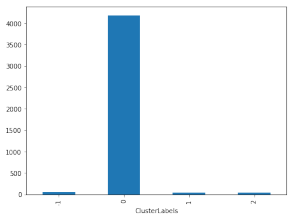
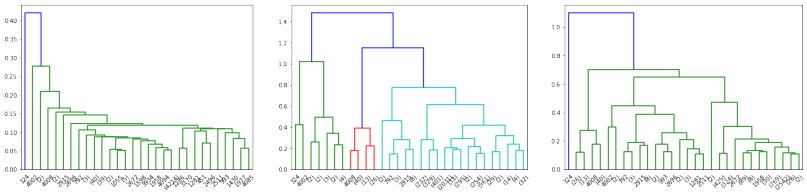


Figure 3.11: *Clusters' distribution.*

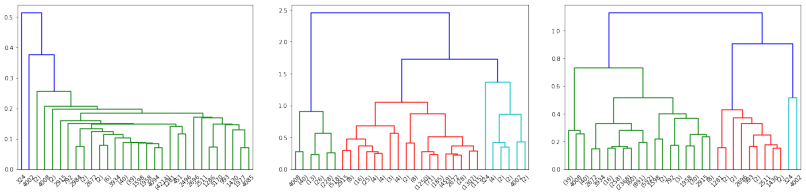
3.3 Hierarchical

3.3.1 Study of the clustering parameters

As for Hierarchical agglomerative clustering, we ran the algorithm with both **Euclidean** and **Manhattan distance metrics**, with **min**, **max**, and **average** methods, and plotted the relative dendograms in Figure 3.12.



(a) *MIN* ***(b)*** *MAX* ***(c)*** *AVERAGE*



(d) *MIN* ***(e)*** *MAX* ***(f)*** *AVERAGE*

Figure 3.12: *Dendograms with Euclidean distance employed in the top row (a,b,c) and Manhattan distance for the bottom one (d,e,f).*

As we could expect, dedondograms produced with *min method* show **lower distances** between each bifurcation, while those produced with *max method* have **higher distances**. Looking at the results of *min* and *average* methods we noted an high **number of singletons** (which have been kept isolated until the latest iterations) and few numerous clusters; instead, *max method* better distributed the records among the clusters, so we decided to further investigate this technique, choosing the *euclidean* distance due to its higher interpretability.

3.3.2 Characterization and interpretation of the obtained clusters

We performed several **tree cuts** to determine the final clusters and extract the corresponding labels, plotting the results in Figure 3.13.

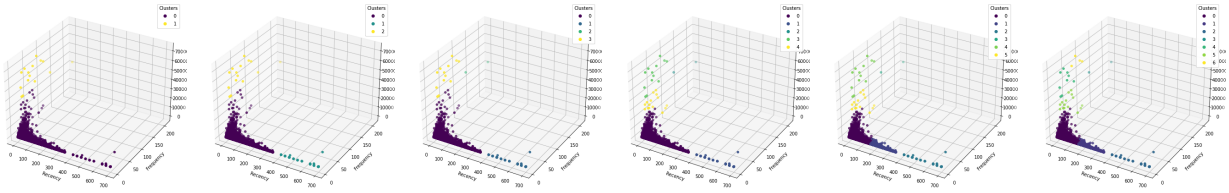


Figure 3.13: *Last iterations of hierarchical clustering algorithm using max method and Euclidean distance as metric.*

We kept the result having **6 clusters** as the best one, since it was the most semantically meaningful (also *comparable* to the results of the other clusterization algorithms).

3.4 Comparison of the clustering techniques

To compare the results obtained by the clustering algorithms, we computed **Silhouette** score and **Davies-Bouldin** score, reporting the results in Table 3.2.

	Silhouette	***Davies Bouldin***
K-means	0.54	0.66
DBScan	**0.64**	0.59
Hierarchical	**0.64**	**0.57**

Table 3.2: *Scores obtained by different clustering algorithms.*

All the algorithms obtained good scores, however the **Hierarchical** has the maximum Silhouette score and the minimum Davies-Bouldin score.

Concerning semantical meaning:

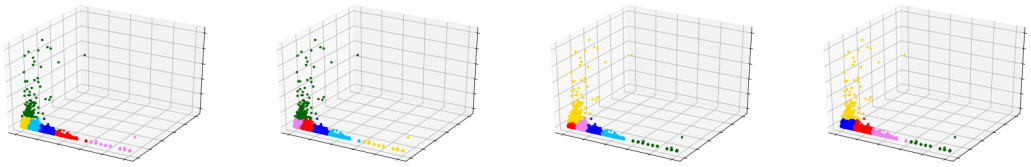
- **K-means**: we were able to derive a meaningful customers' segmentation with 5 different classes of customers;
- **DBScan**: did not provide a satisfying clustering even if it performed very well, since it was able to identify particular customers as noise, creating a big cluster with the remaining ones;
- **Hierarchical**: results were really close to those produced by K-means, but customers' classes were not as clear as the ones of k-means.

Against this background, we think that **K-means** is the algorithm that produced the most meaningful clustering, since it allowed us to produce some considerations and suggestions for the supermarket about each cluster.

3.5 Optional task: Exploring Pyclustering

In addition to the discussed three algorithms, we performed clusterization by employing other 5 algorithms implemented in the PyClustering library.

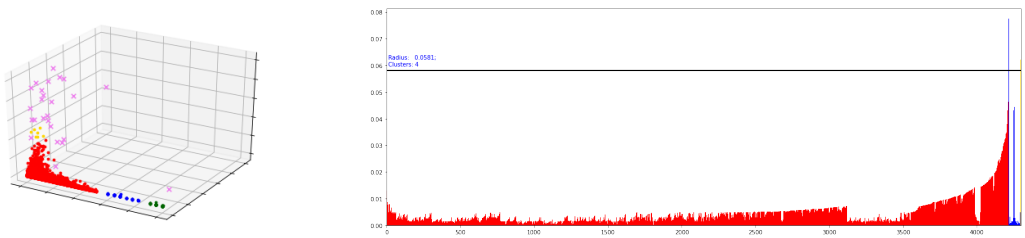
First, we tried **Fuzzy C-means** as an alternative to *K-means*. We used K-Means++ to initialize centroids, trying $\{4, 5, 6\}$ as initial number of clusters (since they gave best results for K-means). For the fuzzyness parameter, we tried $\{2, 2.1, 2.2, 2.5, 3, 4, 5\}$. In general, we obtained results really close to the one that K-means gave, with the difference that it was possible to *better control clusters' size* using the fuzzyness parameter. In Figure 3.14 we reported some of the results we selected that shows the potential of this technique. In more, Figure 3.14d shows an interesting segmentation we obtained, which we believe could be a valid alternative to the K-means' result (Figure 3.2) since non-loyal customers are *better grouped* and a *different segmentation of the loyal ones* was obtained.



(a) *6 - 2.1* ***(b)*** *6 - 2.2* ***(c)*** *6 - 2.5* ***(d)*** *5 - 2*

Figure 3.14: *Interesting results of Fuzzy C-means (left is n. of centroids, right is fuzzyness).*

Next, we focused on the **Optic** algorithm to execute another *density-based algorithm*. Due to the nature of Optic, we passed an high value of ϵ (0.08) to make it work correctly (also, we were not interested too much in computational cost in our analysis), while keeping the same value of min_pts we passed at *DBScan*. We reported the outcome in Figure 3.15a, obtaining a result close to the DBScan's one (Figure 3.8), but Optic produced a *new cluster* for very wealthy customers (yellow points) and captured *more points as noise*. In more, we reported the *reachability plot* in Figure 3.15b, where it is possible to see that clusters were well-defined basing on the reachability-distance and that Optic fine-tuned the ϵ value to one really close to the one we selected for DBScan (*DBScan*: **0.046**; *Optic*: **0.0581**).



(a) *Clusters' distribution.* ***(b)*** *Reachability plot.*

Figure 3.15: *Optic results.*

We also tested other algorithms, without obtaining interesting results:

1. We tried with **X-means**. To initialize centroids, we used once more K-Means++, trying 5 different initial number of centroids: $\{1, 2, 3, 4, 10\}$. We provided a really high value for the maximum number of centroids (100), so that we could be sure that the algorithm would stop bisecting clusters only when BIC value inverted the trend. All the executions terminated with a large number of final clusters (about 50), resulting in a low semantical meaning. This result is probably due to the non-globularity of our data, that brought the algorithm to clusterize data in almost-singleton structures.
2. We tried **Expectation Maximization** with 5 as the number of clusters to find. This algorithm yielded a clusterization that is similar to the one made by density-based algorithms, but this time without isolating noise/outliers.
3. Finaly, we tried the **Genetic approach** (based on centroids), but it failed to yield meaningful clusterization even after 1000 epochs with a population of 100 individuals. We decided to abandon the computation after several hours as this approach is not suitable for large datasets of real data due to slow convergence. However, we think that this Genetic Approach could be an effective centroids-based method for smaller datasets, as it does not require a long search in hyper-parameters space.

4 Predictive Analysis

To predict the *spending-class* of each customer between *high-spender*, *medium-spender* and *low-spender*, we decided the **normalization method for the customer's data** and discussed about different approaches we tried to **compute the labels**. Then, we defined a **new customer profilation** dataset and compared the **results of different predictive models**, **selecting the best one** for the task.

4.1 Normalization's method choice

We evaluated several metrics allowing the classification task, considering the supermarket's typology and the results of some statistical analysis we performed on the customers. First, we thought about using attributes normalized w.r.t. **segments of the customer's period of activity** (weeks, months...).

From Figures 1.7 and 1.8 we inferred the supermarket's typology, which we recall is one that sells mainly **fortuitous items**. We think that these types of supermarket should not be interested about shopping's frequency of the customers, as they don't sell essential goods. They might be more interested in the spending's magnitude of each customer.

So, we computed statistics of the *MeanBasketItems* attribute (mean number of items per basket), obtaining the 1st quartile equal to 93 items. This result, combined with the nature of the items sold and their prices, led us to the conclusion that **many customers are wholesalers**. From common knowledge, wholesalers tend to shop with a significantly lesser frequency and regularity with respect to normal customers, but we still wanted to verify that.

Considering only customers who returned to the shop at least once in the period of observation (which are only the 65%), they come back with a mean of 56 ± 44 days, so **customers do not come regularly nor within brief periods of time**. Even considering the hypothesis of some customers being represented with multiples IDs, these are still quite high values. So, if we were to consider months as a segmentation parameter, we would not differentiate enough customers, while choosing a briefer period of time might end up in several wrong classifications (like wholesalers classified as low-spenders just because they came few times, even if they spent a lot).

Given all these considerations, we decided to compute attributes normalized w.r.t. the **number of baskets**, ignoring the periods of inactivity of each customers.

4.2 Definition of new customer profiles

Considering the task we have been given, we immediately noticed the unsuitability of our previous customer profilation defined in Chapter 2: many attributes are not related to spending (like *MainCountry*, *PReturn* or *WeekDayPref*) or are not normalized. So, we created a new dataset, recycling some ideas from our previous customer profiles and creating new attributes:

- ***Recency, Frequency, MeanBasketSale, MeanBasketItems, E-Prods, E-Sale, E-Baskets, E-Intervals*** and ***PurchasingFreq*** are the attributes we already defined in Section 2.1;
- ***MeanItemSaleBasket*** is the normalized *MeanItemSale* attribute;
- ***Cat0, Cat1*** and ***Cat2*** are 3 new attributes we defined. We performed a binning over the price of the products exploiting 1st and 2nd quartiles, obtaining highly expensive, average expensive and inexpensive items. Hence, we defined these three new attributes as the **mean number per basket of items purchased** by the user for each **item's category**.

In order to be more sure about users' classification, we dropped every customer having *less than 2 baskets*, resulting in 1.494 removed entries. Then, we used these data to compute the labels (as we will describe in Section 4.3). Finally, we have regenerated the dataset, this time only by taking into account the *first 2 baskets* of each user.

4.3 Computation of the labels

We discussed and tried several ideas in order to assign meaningful labels to the customers.

First, we tried to classify users based on the attributes (**Cat0, Cat1, Cat2**), but these quantities alone were not very indicative (we also tried to compute differently the attributes, with no good results) and we obtained several wrong classifications.

Then, we thought about a more articulated method:

- First, we performed a skimming of users having low *TotItems* and *E-Sale*. This way, we would only consider users who bought a **sufficient number of items** having different sales;
- After that, we thought that the **MeanItemSaleBasket, Cat0, Cat1 and Cat2** would now be more representative of the user's spending capacity, so we made some tries to assign the labels based on that. We tried to define thresholds and perform clustering, without obtaining satisfying results: *MeanItemSaleBasket* was still ambiguous, since the number of low-cost items was really high compared to the other categories, so we abandoned this idea as well.

Finally, we thought that **MeanBasketSale** (average of the money spent for basket) could be a good indicator of the user's spending capacity, so we decided to compute the three labels based on this attribute only. By analyzing the distribution of this attribute, we found and dropped *8 outliers*. Then, we decided to compute the labels by performing clustering using **k-means** with *k=3*, but first, we temporarily removed values >= 1000 (which we have then re-integrated in the dataset as high-spending customers) in order to obtain a denser distribution. After clustering was performed, we obtained 1.329 low-spender, 1.091 medium-spender and 407 high-spender, with thresholds calculated at 287.21 (low-medium) and 551.85 (medium-high).

4.4 Models' selection, configuration and results

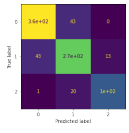
We have trained and tested **6 classification algorithms**, fine-tuning the hyper-parameters through **grid searches** and **3-fold cross-validations** (when possible). We normalized the values using the **MinMax scaler**. Since the classes distribution was unbalanced, for each algorithm we also tried to pre-process the training set using the **SMOTE over-sampling** technique, generating new *legal* synthetic data. We decided to not employ under-sampling due to the low number of records we have been given.

First, we tried **Decision Tree**. We thought that the final model needed no post-pruning, since it was tiny enough. But the results were not satisfactory, so we tried the **Random Forest**, which not surprisingly obtained better results, even if it made interpretability harder. However, for this task we decided to not focus on that, so we went on and tried other models. We tried with **Naive Bayes**, but it expected completely independent attributes, so it performed poorly with respect to the others. Then, we moved on with **k-Nearest Neighbors**, which obtained an accuracy of 1 even if the values chosen for k were not 1, and performed enough well considering the high-dimensional data on which we are working. We also tried the **Support Vector Machine**, even if it is not easy to be sure about the grid search being performed in an optimal way, due to the wide range of possible values for regularization parameter. Finally, we built a really compact **Artificial Neural Network** composed of two layers and 3 output neurons, which surprisingly performed really well. In Table 4.1, it is possible to see the performances of every algorithm, while we decided to leave the parameters of each classifier in the Python notebook.

	TRAINING-SET								**TEST-SET**							
	No-Sampling				*Over-Sampling*				*No-Sampling*				*Over-Sampling*			
	A	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**
Decision Tree	0.88	0.88	0.88	0.88	0.91	0.91	0.91	0.91	0.83	0.83	0.83	0.83	0.79	0.79	0.79	0.79
Random Forest	0.99	0.99	0.99	0.99	**1.0**	**1.0**	**1.0**	**1.0**	**0.84**	**0.84**	**0.84**	**0.84**	**0.86**	**0.86**	**0.86**	**0.86**
Naive Bayes	0.71	0.71	0.71	0.71	0.63	0.67	0.63	0.62	0.72	0.71	0.72	0.70	0.69	0.70	0.69	0.68
k-Nearest Neighbors	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.74	0.75	0.74	0.73	0.74	0.74	0.74	0.74
Support Vector Machine	0.80	0.81	0.80	0.79	0.83	0.83	0.83	0.83	0.78	0.80	0.78	0.78	0.80	0.80	0.80	0.80
Artificial Neural Network	0.84	0.84	0.84	0.84	0.86	0.87	0.86	0.86	0.83	0.83	0.83	0.83	0.84	0.85	0.84	0.84

Table 4.1: *Scores obtained by different classification algorithms. A='Accuracy', P='Precision', R='Recall', F1='F1 Score'.*

In general, the over-sampling technique did not always improve the scores, but the most satisfying results were obtained by employing it. Indeed, the best numerical results on the test-set were obtained by the **Random Forest** when using over-sampling, followed by the **Artificial Neural Network** and the **Support Vector Machine**. Moreover, the Random Forest has the advantage of being completely interpretable, so we decided to report further information on the model we obtained. By looking at the features that Random Forest showed the most interest, we obtained **MeanBasketItems** with *38% importance*, followed by **Cat1** with *16%*, **Cat0** with *8%* and **MeanItemSaleBasket** with *7% importance*: we think this is a good result, since it showed a good correlation between the quality/quantity of items bought and the final labels. Finally, we report the confusion Matrix and the per-class scores in Figure 4.1, where we found no anomalies.



	Low-spending	**Medium-spending**	**High-spending**
Precision	0.91	0.78	0.86
Recall	0.87	0.85	0.80
F1-score	0.89	0.81	0.83
Support	399.00	328.00	122.00

Figure 4.1: *Per-class scores of the Random Forest.*

5 Pattern Mining

To mine frequent sequential patterns, we used the **GSP algorithm based on Apriori**.

We defined two hyper-parameters: ***min_baskets*** and ***min_support***, where *min_baskets* is the minimum number of baskets that a user must have to be included in the analysis, while *min_support* is the minimum support (in percentage) that makes a sequence frequent. We tried the following values: $min\_baskets = \{20, 10, 5, 3, 2\}$; $min\_support = \{0.4, 0.35, 0.3, 0.25, 0.2, 0.15\}$.

To execute the algorithm, we **reshaped customer's data as a sequence of baskets** and kept only the *ProdID* from each transaction, since the other attributes were not suitable for the task.

5.1 Hyper-parameters selection

To choose the final hyper-parameters, we based our decision on 3 factors: the **n. of resulting patterns**, n. of patterns **longer than 2**, n. of patterns containing a **duplicated product**.

As the *min_baskets* value decreased we noted a substantial decrease in the *n. of resulting patterns*, eventually reaching 0 for high values of *min_support*. From this we can derive that casual customers do not follow evident patterns, which led to the decision of using ***min_baskets* = 10**. To choose the best value of *min_support*, we evaluated both output's quantity and quality (we were interested in patterns longer than 2 not containing repeated products). At the end, the best values were obtained by employing ***min_support* = 0.25**.

5.2 Results

By employing the selected hyper-parameters, we obtained **73 frequent patterns**, which we think is high enough for patterns having support greater than 25%. 56 of them had length 1, 14 had length 2 and 3 had length 3. Moreover, 11 patterns contained repeated products and 1 pattern had an event with two items. Table 5.1 contains some of the most interesting patterns we found.

Basket 1		Basket 2		Basket 3		Support
REGENCY CAKESTAND 3 TIER	10					42 %
JUMBO BAG RED RETROSPOT	26					42 %
ASSORTED COLOUR BIRD ORNAMENT	51					34 %
REGENCY CAKESTAND 3 TIER	11	*REGENCY CAKESTAND 3 TIER*	10			33 %
JUMBO BAG RED RETROSPOT	29	*JUMBO BAG RED RETROSPOT*	26			33 %
JUMBO BAG RED RETROSPOT	24	*JUMBO BAG VINTAGE DOILY*	19			27 %
JUMBO BAG RED RETROSPOT	32	*JUMBO BAG RED RETROSPOT*	25	*JUMBO BAG RED RETROSPOT*	30	27 %
WHITE HANGING HEART T-LIGHT HOLDER	30	*WHITE HANGING HEART T-LIGHT HOLDER*	24	*WHITE HANGING HEART T-LIGHT HOLDER*	15	26 %
JUMBO BAG RED RETROSPOT	21	*LUNCH BAG RED SPOTTY*	15			26 %
LUNCH BAG RED SPOTTY	17	*JUMBO BAG RED RETROSPOT*	22			25 %
REGENCY CAKESTAND 3 TIER	13	*REGENCY CAKESTAND 3 TIER*	11	*REGENCY CAKESTAND 3 TIER*	13	25 %

Table 5.1: *Most relevant patterns found and their relative support. Numbers next to each item represent the mean Qta value over the transactions composing the pattern.*

We represented each item with a color to have a better look at how the same item was captured by different patterns. Interestingly, the items contained in the sequences share a **semantical meaning**: most of the times the *same item* was bought (like for cakestands, probably because wholesalers found out those items sold well); in other cases instead, the user bought the *same item's typology* (like for the red and vintage bag, probably because wholesalers wanted to re-sell more variants of the item).

We investigated about whether some items in the patterns were as well in the **most bought items** of the store (Figure 1.7a) and, as was foreseeable, they were.

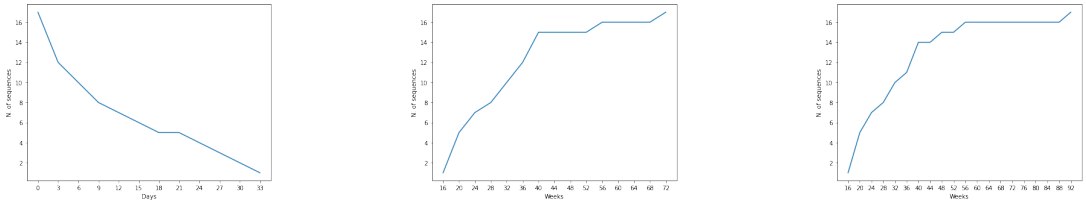
To further analyze the results, we also computed the **mean quantity** over the transactions composing the pattern. We can see a lot of high values, which is a further confirmation about the selected customers being wholesalers. Moreover, it is interesting to see that the quantities tends to **decrease** once a wholesaler has once bought an item, probably due to the fact that their stock was not finished yet.

5.3 Optional task: Time constraints

To impose time constraints over the patterns, we had to **modify** the GSP implementation. We decided not to change the original use of the library, so the time stamps must be provided as an additional parameter. In order to implement time constraints, we have done two main changes:

1. The ***isSubsequence*** recursive function has been re-written as iterative and modified to check for time constraints between events.
2. An additional function ***generateContiguousSubsequences*** has been written to generate (k-1)-contiguous-subsequences instead of direct ones when time constraints are passed.

We executed GSP with same values of min_baskets and min_sup as before (10; 0.25), introducing and testing individually the 3 time constraints. To define the values to test, we employed ***datetime.timedelta*** objects. The ranges of values to test were chosen accordingly to the minimum/maximum days required to start/stop to see any changes in results' size. We tested **min_gap** for values in *0, ..., 33 days* with *step 3*, **max_gap** for values in *16, ..., 72 weeks* with *step 4*, **max_span** for values in *16, ..., 92 weeks* with *step 4*. We plotted the trends in 5.1.



(a) *Min Gap.* ***(b)*** *Max Gap.* ***(c)*** *Max Span.*

Figure 5.1: *Trends of the Time Constraints.*

At a glance, the plots may appear **contradictory**, since within 33 days passed to *min_gap* we do not get sequences longer than 2 anymore, while using *max_gap* and *max_span* we start to obtain sequences longer than 2 only with values greater than 16 weeks (112 days). Actually, the plots are telling us that **users do not follow patterns within similar time periods**. To verify this assumption, we have:

1. *Retrieved the sequences longer than 2 without using any time constraint;*
2. *Retrieved the transactions from the dataset which followed the sequences;*
3. *Computed the time gap between each event retrieved;*
4. *Computed the distribution of the time gaps for each sequence.*

The obtained distributions are all similar to each other. Figure 5.2 is one of them.

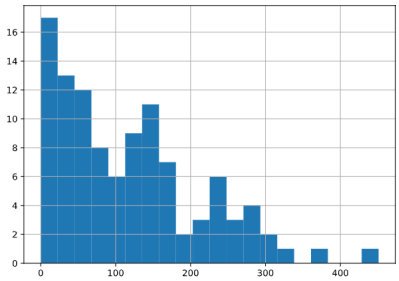


Figure 5.2: *Time gaps' distribution of [['JUMBO BAG RED RETROSPOT'], ['JUMBO BAG VINTAGE DOILY']].*

Analyzing the distributions, we observed that a **good number of time gaps are within 30 days**, which explains why we stopped obtaining sequences for min_gap values larger than 33. Moreover, the **remaining time gaps are approximately decreasingly distributed between 30 days and 300 days**, which do explain the trends of max_gap and max_span. To further remark the correctness of the data, we can refer to Table 5.1: as we can see, a lot of support values are really close to 25% (min_support provided), which means that loosing the support of even few customers in a pattern by using the time constraints could make the pattern non-frequent.

So, even if we limited our study to the most loyal customers (min_baskets=10), we discovered that a lot of the **customers follow the retrieved patterns in short intervals of time** (as we previously discovered in Figure 2.4d). In that case, the shop should expect customers to come back soon to purchase items similar to the ones they bought before.

6 Conclusions

At first, we started with little to no information about the supermarket. With our analysis, we were able to define the typology of the supermarket as one that only sells **fortuitous items**. By studying the dataset, we repeatedly reported that the **quality** of the dataset can be improved, starting from the data acquisition and storage: for example, the dataset contained many **anomalies** (as we have seen during our outliers' removal) and **products' description** could be written more explicitly (see MANUAL, DISCOUNT).

Moreover, we think that **acquisition of personal data** and **products' tags** would have helped the cluster analysis, even if we have already obtained **meaningful segmentation** of the customers through the usage of K-means and some of the algorithms of Pyclustering.

Through the predictive analysis, we were able to define **models that effectively predicted the spending-class** of the user, finding an interesting correlation between the spending-class we defined through clustering and the quality/quantity of items bought.

Finally, the pattern mining task revealed **few (and short) interesting patterns**: once more, we think data quality affected the number and length of the results (as probably some transactions are referring to multiple customer instead of a single one). Still, we found **meaningful semantical correlations** between items in the patterns (like in the case of bags) which we think is not an obvious result. Moreover, by applying time constraints, we were able to discover **time information** about how the majority of customers follow the patterns.