Report of Data Mining Project
Customer Supermarket

Grieco Giuseppe
g.grieco6@studenti.unipi.it

Lepri Marco
m.lepri2@studenti.unipi.it

Sangermano Mattia
m.sangermano1@studenti.unipi.it

DM course (309AA), Academic Year: 2020/2021

Contents

1 Introduction

In this report we show our work on a dataset of transactions made by customers of a generic store. The document is divided in the following sections:

- **Data Understanding**: it contains the dataset analysis, both from a semantic and statistical point of view;
- **Data Preparation**: it describes how the dataset is partitioned and how is cleaned;
- **Clustering**: it describes the clustering task, including the customers dataset preparation and the cluster analysis;
- **Classification**: it describes the classification task, including the customers dataset preparation and the analysis of the classification models;
- **Pattern Mining**: it contains the details of the sequential pattern mining task, including the analysis of the found patterns;

2 Data Understanding

2.1 Data semantics

The dataset consists of 471910 entries, each of them with 8 features. A complete specification with type and a brief description of the features is given in Table 1. The `Sale`, `BasketDate` and `CustomerID` fields were converted from strings to the corresponding type after inspecting the data types.

Name	Type	Description
BasketID	Alphanumeric	Unique identifier of the basket/order
BasketDate	Date	Date of the purchase
CustomerID	Integer	Unique identifier of the Customer
CustomerCountry	Categorical	Country of purchase
ProdID	Alphanumeric	Unique identifier of the product
Sale	Float	Price of the product
ProdDescr	Text	Description of the product
Qta	Integer	Quantity purchased

Table 1: Features specification

The records in the dataset correspond to single transactions for a generic wholesale shop or supplier. An order is identified by an unique `BasketID` and its content is described with multiple rows all sharing the same `BasketID`, one for each item in the basket. Each basket is then associated with a single `CustomerID` identifying the customer that performed the order. Each transaction contains also the price and the quantity of the purchased product, uniquely identified by `ProdID`. The dataset covers approximately one year of transactions from 01/12/10 to 09/12/11.
In the following, we describe more in detail the dataset and its attributes, focusing on the correlation among them and on what particular kind of records represent.

2.1.1 BasketID and Negative Qta

There are two major types of `BasketID`s: the ones corresponding to standard orders, which are normal 6 digits IDs, and the ones corresponding to cancelled orders, which start with the letter C followed by a normal 6 digits ID. These last type of orders in particular were recognized to be cancelled or returned since the quantity (`Qta`) is always negative and they are (almost) always preceded, in time, by a standard order from the same customer for the same product in greater or equal quantity. There is also a third type of IDs, which appear only two times, where the ID starts with the letter A. These two records corresponds to the two records with B as `ProdID` which we discuss in Section 2.1.3.

The negative Qta appears also in other records which have normal BasketID. These records, however, always have empty CustomerID, Sale equal to 0 and the corresponding BasketID contains only that element. Moreover, ProdDescr does not seem to contain the description of the corresponding product, but just comments on the transaction itself. This suggests that these records correspond to internal management operations, for example concerning products that are damaged or lost, and do not correspond to actual transactions made by standard customers.

2.1.2 CustomerIDs

The attribute CustomerID identifies the customer who performed the transaction. There are, however, many transactions that do not have any value for the attribute CustomerID. Of these, some belong to the type of transactions discussed at the end of section 2.1.1, but the majority seem to correspond to actual transactions made by real customers. There are also some cancelled orders with no CustomerID associated. This suggests that there exist two type of customers: one that are somehow registered in the wholesale shop system, who can be uniquely recognized by their CustomerID, and customers which use the service without registering or giving information about who they are.

2.1.3 ProdID

ProdID uniquely identifies the object the transaction refers to. In most of the cases it is a 5 digits code with an optional alphabetical character at the end, probably specifying the version (e.g. color) of the corresponding product. Each ProdID also has its own ProdDescr although it is not unique, but some variants may exist.
However, there are some ProdIDs that do not follow the above pattern, but are general alphabetical or alphanumerical codes. Below, we briefly describe their possible meaning:

- POST, DOT, C2: These could correspond to shipping fees or costs, since they always have POSTAGE, DOTCOM POSTAGE or CARRIAGE, as product description. This suggests that the store can also deliver orders directly to the customers and that multiple ways or platform to buy from the store exist.
- D: This could correspond to discounts applied to baskets, although it does not directly appear in the basket itself. Indeed, it always appears in BasketIDs corresponding to cancelled orders, always with Qta $= -1$. Then, Sale could correspond to the applied discount value.
- M: This ProdID is always associated with the ProdDescr "Manual" and probably corresponds to transactions manually inserted into the system.
- BANK CHARGES: This ProdID appears in two different situations. The first, in "standard" basket associated with a non-NaN CustomerID with Sale $= 15.00$ and Qta $= 1$. These transactions probably are bank fees. Since the corresponding baskets contains only this transaction, the fees are related to the payment of some other basket. The other times it appears in "cancelled" baskets associated with no CustomerID, variable Sale and Qta $= -1$. These probably correspond to other banking movements not directly related to specific orders, considering also that Sale can be very high.
- S: This ProdID always has "SAMPLES" as description and appear in cancelled baskets with no CustomerID associated. These probably correspond to products given as free samples, without keeping track of which customer takes them.
- AMAZONFEE: The corresponding entries always have "AMAZON FEE" as description, which clearly refers to fees paid to Amazon, and appear in cancelled baskets (only once it appear in a standard basket, but it might be caused by an insertion mistake). Considering also that the corresponding Sales are very high, these probably refers to the fees to let the store operate also on the Amazon website.
- DCGS*: Different ProdIDs appear in this category, depending on what is after DCGS. They seem to correspond to actual products considering how the corresponding baskets often contains many other products. However, many times they appear with negative Qta (although having a "standard" BasketID), Sale $= 0$ and "ebay" or empty description. It is not clear what these entry refer to but are probably related to internal management operations as discussed in Section 2.1.1

- `gift_0001_`*value*: These correspond to gift voucher as also the description "`Dotcomgiftshop Gift Voucher £`*value*`.00`" indicates. This suggests that the entries could come also from an online shop where gift cards can be bought. Possible values are: 10, 20, 30, 40, and 50.
- `B`: This `ProdID` appears only two times, in `BasketID`s starting with an A and negative `Sale`. The description "`adjust bad debt`" seems to refer to internal management operations, but too little information is available to be sure what these entries refer to.
- `PADS`: This `ProdID` probably refers to filler pads for the package to be delivered with the products, considering that they appear in baskets with other products and have a very low `Sale` (0.001).
- `CRUK`: It is not clear what this `ProdID` refers to. The corresponding description is "`CRUCK Commission`" and appears always in a cancelled basket and with varying `Sale`. Interestingly, all the entries with this `ProdID` are associated with the same `CustomerID`: 14096.

2.2 Data duplication

Overall, the dataset does not seem to contain major errors or duplication issues. The only case of duplicate data are some transactions whose `BasketID` is associated with other transactions for the same `ProdID` with the same `Sale`. However, since the quantity is most of the times different, these are probably cases in which the same product was inserted in the basket multiple times, rather than errors or duplicate records. A related issue is the case of `BasketID`s corresponding to the same `CustomerID` that have the same `BasketDate`. These might correspond to single orders which are, however, delivered to different addresses or paid with different methods and, therefore, treated as different baskets.

2.3 Data analysis

In order to perform statistical analysis, the dataset has been partitioned in the two following sets:

- Purchases: All the entries with `BasketID` without trailing letters (e.g. standard orders), `Qta` > 0 and normal `ProdID`. This set is further divided in entries with a non-null `CustomerID` and entries with no `CustomerID` associated. These corresponds to the actual orders performed by the customers;
- Returns: All the entries with a trailing C in the `BasketID` (thus, with `Qta` < 0) and normal `ProdID`. These corresponds to the returns or cancelled orders made by the customers.

The following plots show some statistics about the set containing the customers' orders.

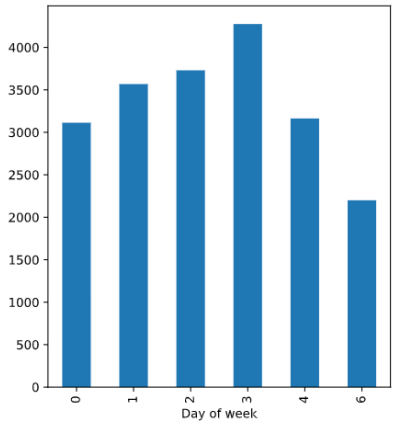

Figure 1: Number of baskets per day of week. The most active days seem to be the Thursdays (day 3), while on Sundays (day 6) the activity is much lower than in the other days. Interestingly, no orders appear on Saturdays (day 5), which might suggest that the store related to the dataset is closed on those days.

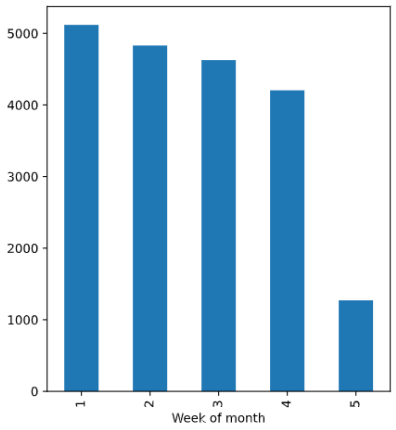

Figure 2: Number of baskets per week of month. There is a decreasing tendency as the month proceeds, although the number of baskets remain in all the cases above 4000. The only exception is the 5th week which is, however, usually shorter than the other weeks, so it is less probable that a customer performs an order in that period.

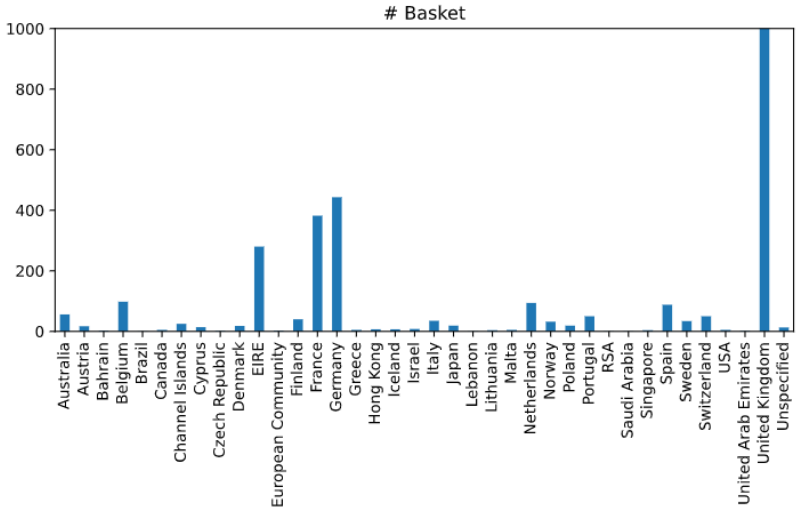

Figure 3: Number of baskets per country. The distribution of orders is highly unbalanced and sees United Kingdom as the country in which almost all orders are made. This leads us to think that the shop related to the dataset is mainly for United Kingdom.

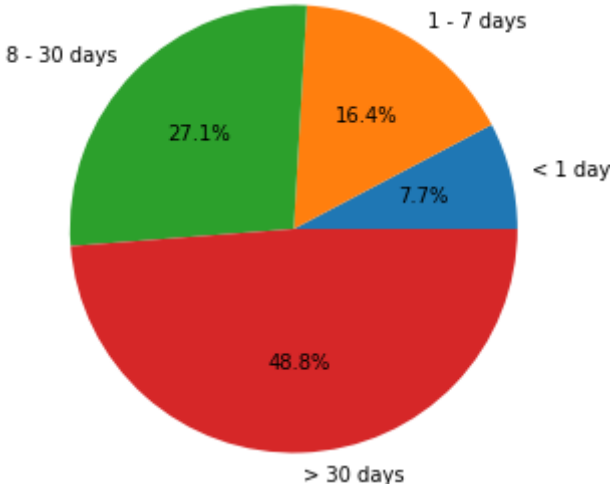

Figure 4: Return distance distribution. Many orders are returned before 7 days, in particular on the same day. These are probably purchases made by mistake

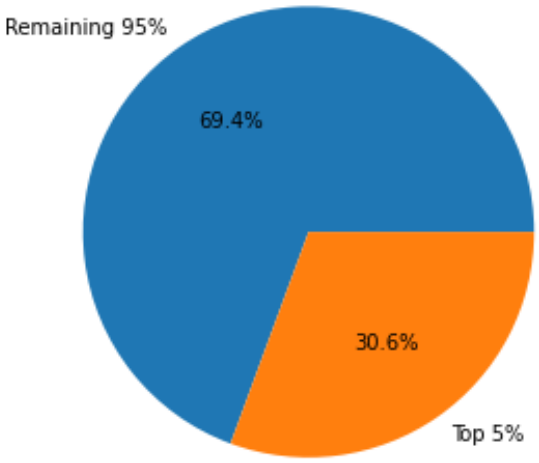

Figure 5: This figure shows that the 5% of the most purchased products covers 30.6% of the total transactions, thus identifies a dominant subset of the products.

3 Data Preparation

3.1 Additional Fields

Name	Type	Description
`BasketDay`	Integer [1,31]	Day of month of the transaction
`BasketMonth`	Integer [1,12]	Month of the year of the transaction
`BasketYear`	Integer	Year of the transaction
`BasketWeekOfMonth`	Integer [1,5]	Week of month of the transaction
`BasketWeekOfYear`	Integer [1,52]	Week of year of the transaction
`Tot`	Float	Total cost of the transaction. Computed as `Qta` · `Sale`

Table 2: Features added to the dataset

The dataset was enriched with additional features, computed starting from the preexisting ones. Table 2 shows the added features. Most of them were extracted from the `BasketDate` attribute to be used to compute the statistics, while `Tot` was used also to clean the dataset from the outliers, as described in the next section.

3.2 Data Cleaning

In order to improve the quality of the dataset, all erroneous records or entry with meaningless values have been deleted or transformed:

1. Entries with `ProdID` equal to `BANK CHARGES`, `S`, `AMAZONFEE`, `CRUK`, `PADS` and `B` have been deleted for the reasons described in Section 2.1.3. (90 records)
2. Product returns: All products and related returns made on the same day of purchase have been deleted if the returned quantity is equal to the number of products purchased otherwise it has been updated. We assumed that those products have been bought by error. (1601 records)
3. Internal management transactions: These entries, described in Section 2.1.1, have been deleted because they do not seem to be related to actual orders. (685 records)
4. Uppercasing: We uppercased all the alphabetical characters in the `ProdID`s since some of them had the tailing letter lowercased despite having the same `ProdDescr` as in the case with uppercased tailing letter.
5. Too little sale: All the entries with `Sale` less then 0.01 have been removed, since we do not want to keep such low granularity in the dataset. (610 records)
6. Product joining: All entries with the same `BasketID`, `ProdcutID` and `Sale` have been merged into one entry by updating the quantity accordingly. (9965 records)
7. Outliers: They were considered outliers all the entry beyond the maximum ($Q3 + 1.5 \cdot IQR$) and the minimum ($Q1 - 1.5 \cdot IQR$) of the two box-plot in Figures 6 and 7. The outliers have been removed and the final outcomes are shown in Figures 8 and 9. (43269 records)

In total 56220 entries were removed, for a final purchases dataset containing 405715 transactions.

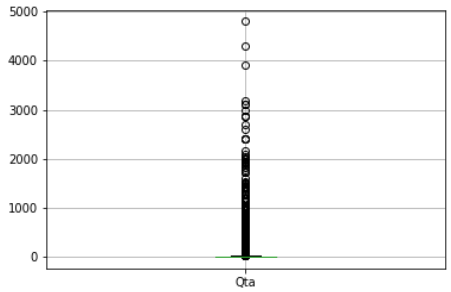

Figure 6: `Qta` initial box-plot

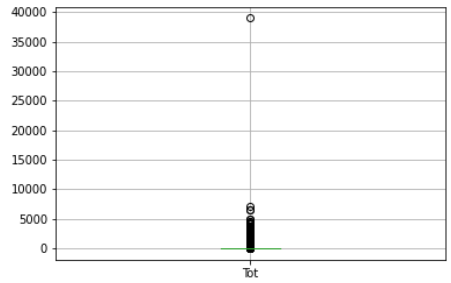

Figure 7: `Tot` initial box-plot

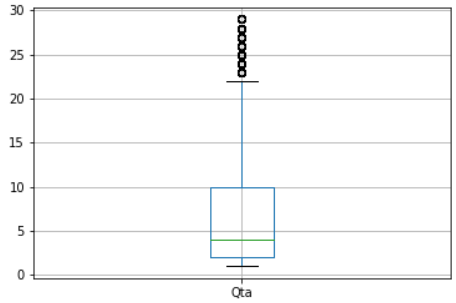

Figure 8: `Qta` final box-plot

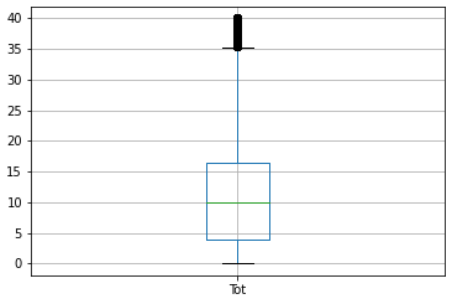

Figure 9: `Tot` final box-plot

4 Clustering Analysis

In this section, we describe the task of clustering the customers, basing on their purchasing behaviour. The customers are represented as a set of indicators, computed starting from the orders dataset. Section 4.1 describes the indicators and the final dataset used for the clustering, while Sections 4.2, 4.3, 4.4 and 4.5 contain the analysis of the clusters found with the different clustering algorithms.

4.1 Indicator variables

Table 3 shows the indicators representing each customer. Note that most of them were computed considering the weekly activity of the customer rather than the single baskets or the monthly activity. This grouping is large enough to contain sufficient information in order to represent the customer for the clustering task and it is not sensitive to single purchases; on the other hand it is small enough to take into consideration temporal changes.

Name	Description
`NumBaskets`	Total number of baskets
`NumProducts`	Total number of distinct products
`TotalProducts`	Total number of purchased products
`MaxProd`	Maximum number of items in a basket
`distance_in_time_purchase`	Average period of time (in weeks) between two purchases
`Qta_entropy`	Entropy (among weeks) of the sum of the `Qta` within a week
`Qta_mean`	Average of the sum of `Qta` within a week
`#ProdID_entropy`	Entropy (among weeks) of the number of distinct products bought in a week
`#ProdID_mean`	Average number of distinct products bought in a week
`Tot_entropy`	Entropy (among weeks) of the total spent within a week
`Tot_mean`	Average of the sum of `Tot` spent within a week
`Sale_entropy`	Entropy (among weeks) of the average `Sale` within a week
`Sale_mean`	Average of the sum of `Sale` within a week

Table 3: Indicators

We decided to add both entropy and mean because the entropy is useful to quantify the predictability of customer behaviours, while the mean serves to understand the magnitude of the value.

4.1.1 Entropy

In order to compute the entropies, the attributes they refer to have been discretized in three distinct values. Table 4 shows the thresholds used for the discretization of each attribute. They were selected by looking at the corresponding distribution among the week of years, so that the central part contains high frequency values (average user behavior) and the two lateral parts contain low frequency values. Then the entropy for a specific attribute has been computed considering the frequency of the discretized values.

Name	1st threshold	2nd threshold
`Qta_entropy`	59	212
`#ProdID_entropy`	8	29
`Tot_entropy`	120	350
`Sale_entropy`	2	3.6

Table 4: Threshold for the discretization

4.1.2 Outliers

In order to detect and decide which points to drop as outliers we looked at the box plots and the distribution histograms of the indicators. Finally we defined a list of threshold, one for each indicator, above which the

points were considered outliers. This operations resulted in removing 62 customers.

4.1.3 Standardization and Normalization

Each indicator variable has been first standardized using the z-score and than normalized using Min-Max normalization.

4.1.4 Correlation

In order to check if a relationship between two indicator variables exists, two correlation coefficient has been computed: Kendall (Figure 10) and Spearman (Figure 11).

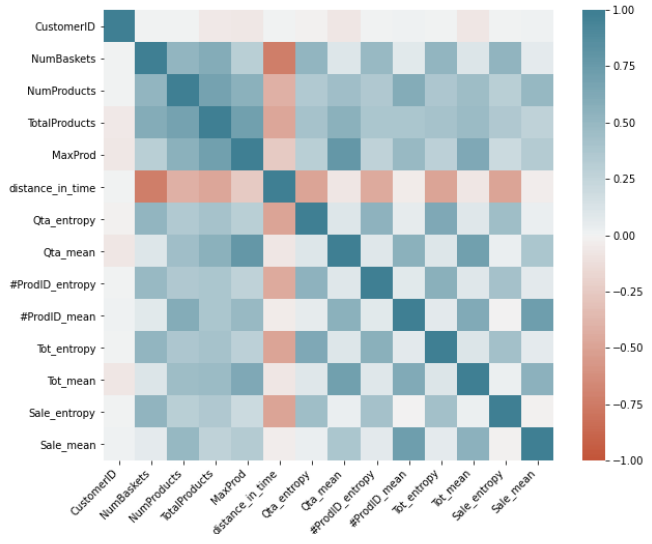

Figure 10: `Kendall` correlation

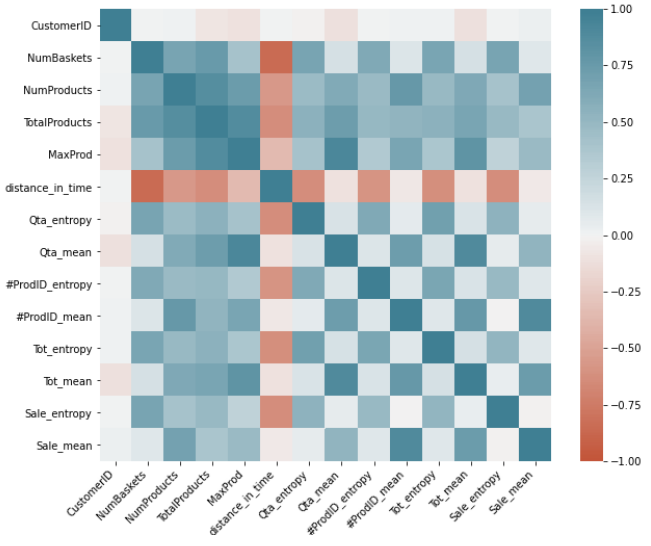

Figure 11: `Speraman` correlation

Both correlation coefficients highlights the same relationships, with Spearman coefficient appearing to be generally higher (in absolute value) compared to Kendall one. From the results obtained the following highly correlated features has been dropped: `ProdId_entropy`, `ProdId_mean`, `Tot_entropy`, `Tot_mean`, `NumBaskets`, `MaxProd`, `NumProducts`.

4.2 DBScan

The `DBScan` algorithm is controlled by two parameters: `eps`, which is the radius of the neighborhood of a point, and `min_samples`, which is the minimum number of points, in the point's neighborhood, to consider the point a core point. The values for the parameters have been decided by looking at the distance from the k-th nearest neighbor for each data point, whose plot is shown in Figure 12. The plot suggests values between 0.15 and 0.25, depending on the value of k. The final analysis takes into account the clusters found with `eps`=0.2 and `min_samples`=10.

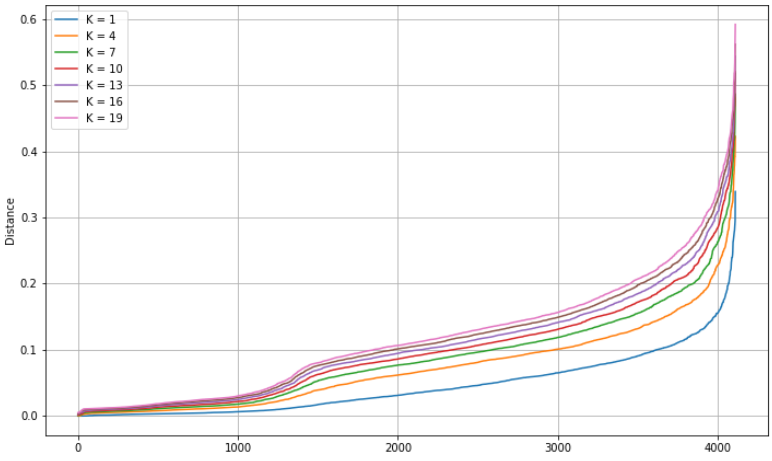

Figure 12: k-th nearest neighbour distance curve

Table 5 shows statistics about the 5 found clusters while Figure 24 shows a 3-dimensional visualization of the data points, colored according to the cluster. Similar results were obtained with other parameters combinations, e.g. `eps`=0.18 and `min_samples`=4.

cluster id	Size	Core points	SSE (MSE)
0	1317	1213	582 (0.442)
1	337	306	118 (0.351)
2	1886	1849	781 (0.414)
3	406	369	159 (0.391)
4	19	13	3.5 (0.183)
Noise points			145
Silhouette score			0.425

Table 5: DBScan clusters statistics

The most significant attributes for the clustering are the two entropies. Indeed, the 5 clusters can be easily identified by the combination of the two entropies's values:

- **Cluster 0**: contains those customers that have both an high `Qta_entropy` and an high `Sale_entropy`, meaning that their purchasing behaviour cannot be easily identified. They also seem to have an higher number of baskets on average, which could also effect the unpredictability of their behaviour.

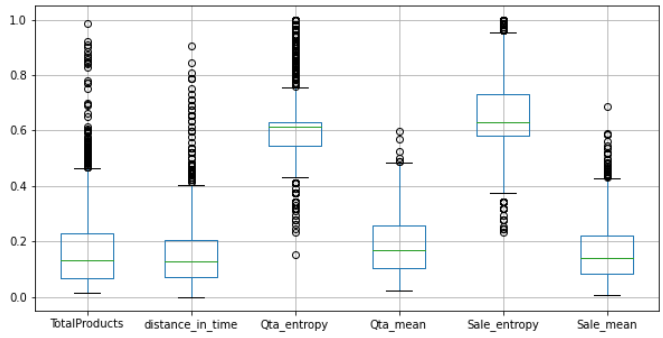

Figure 13: Box plot related to cluster 0

- **Cluster 1**: contains those customers with a medium `Qta_entropy` and a low (always 0) `Sale_entropy`, meaning that they tend to buy products in the same range of price but in varying quantity.

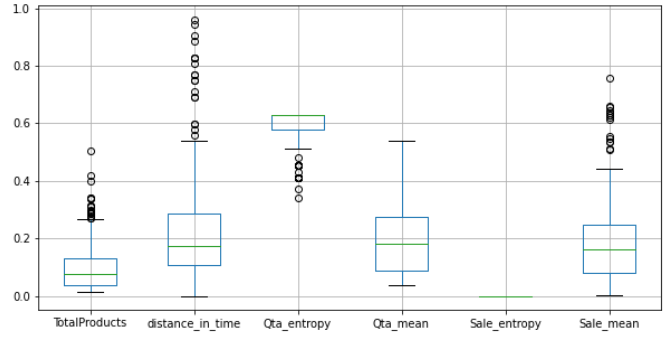

Figure 14: Box plot related to cluster 1

- **Cluster 2**: most of the customers in this cluster have just one basket and, therefore, have 0 `Qta_entropy` and `Sale_entropy`. However, there is a subset of the customers in this cluster that have 2 or more baskets whose entropies are still 0, meaning that their purchasing behaviour is very predictable. For those customers with just one basket instead, the purchasing behaviour is unknown which is the biggest difference between the two kind of customers.

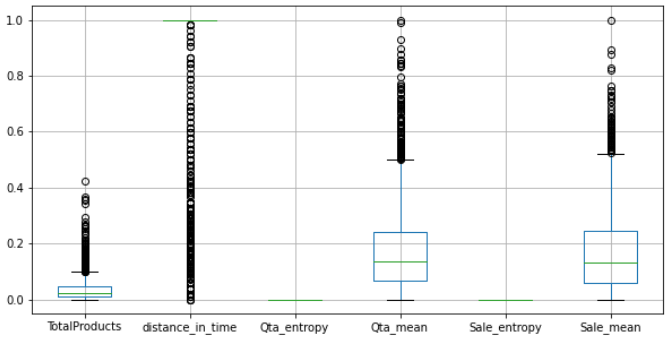

Figure 15: Box plot related to cluster 2

- **Cluster 3**: contains the customers with medium `Sale_entropy` and low (always 0) `Qta_entropy`, meaning that they tend to buy the same quantity of products but in different ranges of price.

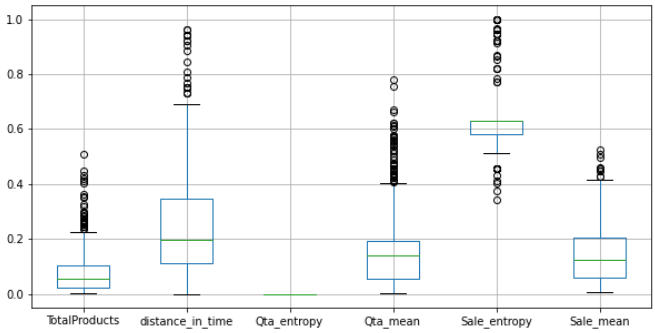

Figure 16: Box plot related to cluster 3

- **Cluster 4**: the customers in this cluster are actually very similar to the customers in Cluster 1. The main difference is that the customers in this cluster have an higher `Qta_entropy`. The same happen with cluster 3, which have a subset of customers with a very high `Sale_mean`, but in that case the algorithm was not able to distinguish them.

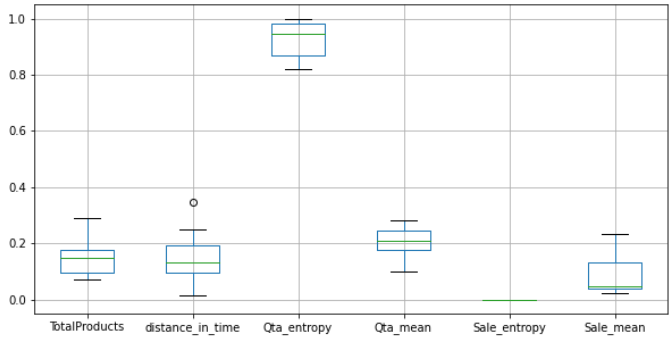

Figure 17: Box plot related to cluster 4

In order to improve the quality of the analysis with the other clustering algorithms, all the points identified as noise have been removed, as well as the points in Cluster 4, whose small size could mislead the algorithms.

4.3 K-Means

The value for K has been decided by looking at the Elbow and the Silhouette curves, shown in Figures 18 and 19. A good trade-off between a low SSE and a high value of the silhouette is obtained with K equal to 4 or 5.

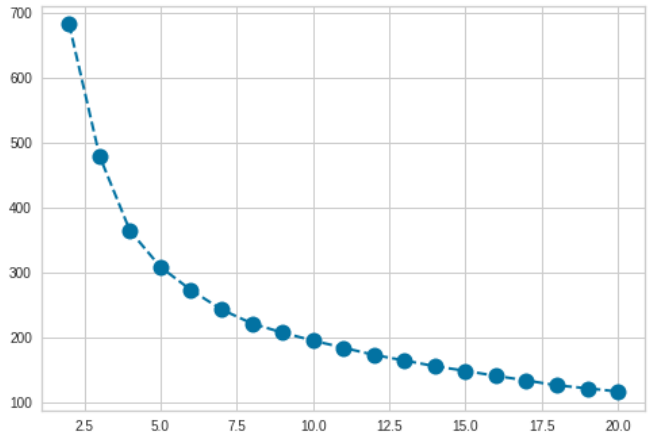

Figure 18: `SSE` with incremental k

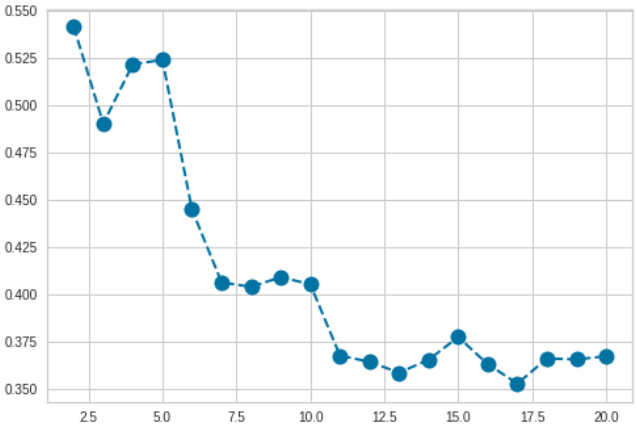

Figure 19: `Silhouette` score with incremental k

Considering K=5, Figure 20, shows the parallel plot of the centroids of the 5 clusters, while Figure 25 shows a 3-dimensional representation of the clusters. Again, the most meaningful attributes are `Qta_entropy` and `Sale_entropy`, while `Sale_mean` and `Qta_mean` do not give any kind of information. Interestingly, the clusters obtained have the same meaning of the clusters obtained using the `DBScan` algorithm, with the difference that `K-means` was able to distinguish between customers with just one basket (Cluster 1, Figure 20) and customers with 2 or more baskets (Cluster 3, Figure 20) but whose purchasing behaviour is predictable, both in terms of `Qta` and `Sale` (Cluster 2 in Section 4.2). The same thing did not happen setting K=4, which is the reason 5 was used as final value for the analysis.

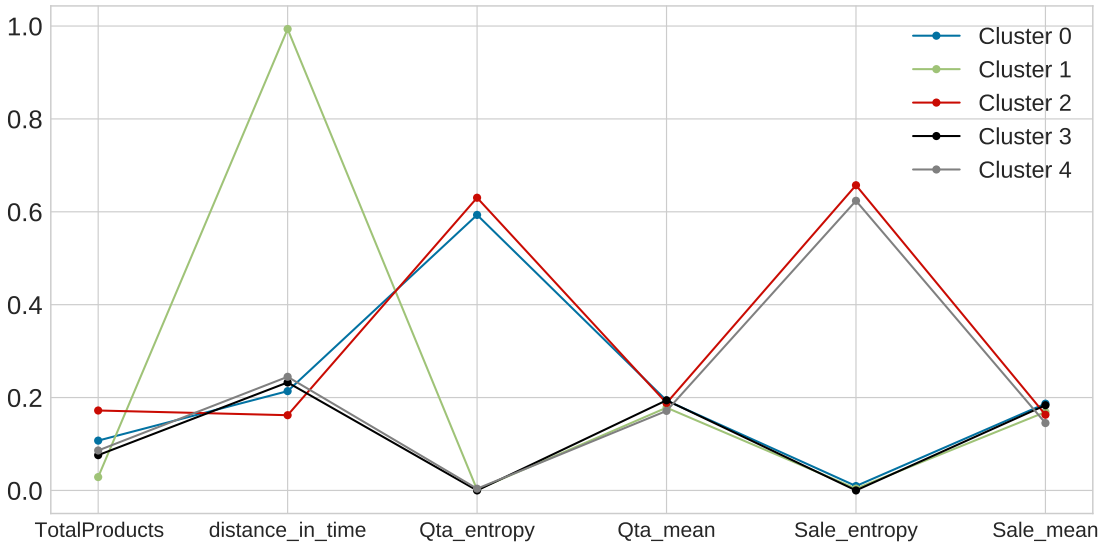

Figure 20: `Parallel plots` constructed using the centroid values of each cluster using k = 5

4.4 Hierarchical Clustering

The parameters controlling the algorithm are the metric used to compute the distance matrix and the linkage criteria used to determines the merge strategy. The values for the parameters that have been explored are reported in Table 6.

Parameter	Values
`Metric`	euclidean, sqeuclidean, mahalanobis chebyshev
`Linkage`	complete, single, avarage weighted, centroid, median, ward

Table 6: Hierarchical Clustering explroed parameters range

By looking at the results, Ward-Euclidean turns out to be the best combination obtained, looking at both the dendograms (Figure 21) and the clusters visualization (Figure 26). As the dendogram shows, 10 is a good cut value, this value was in fact used and the number of clusters found is 5. The final clusters obtained have the same meaning as those of k-means.

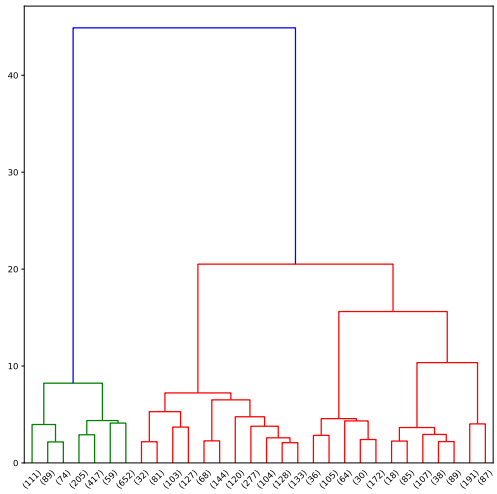

Figure 21: Dendogram using ward and euclidean distance

A further analysis of how the final solution is obtained shows the following:

- The main parameters that guided the algorithm are the entropies.
- The cluster containing customers with a single basket is obtained merging two clusters: both of them contain customers with a single basket but one contains higher spending customers, in terms of `Sale_mean`.
- The cluster containing customers with medium-high value for both the entropies is obtained merging two clusters: both of them maintain the property just described on the entropies but one contains customers who buy more products while spending the same on average.

4.5 Self-Organizing Map

The model is controlled by only one parameter: the size of the grid. Its value has been decided trying different sizes and the following `heuristic`: $5 \cdot \sqrt{N}$, where N is the number of data points in the dataset. The final model has 182 winning neurons (neurons with at least one associated point). In order to find

the final clusters, K-means was ran using as input the weights of the winning neurons. Figures 23 and 22 recommend to use, as in the other cases, 5 as the number of clusters.

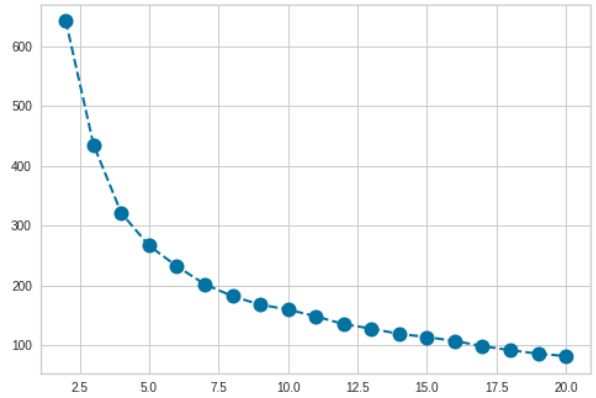

Figure 22: `SSE` with incremental k

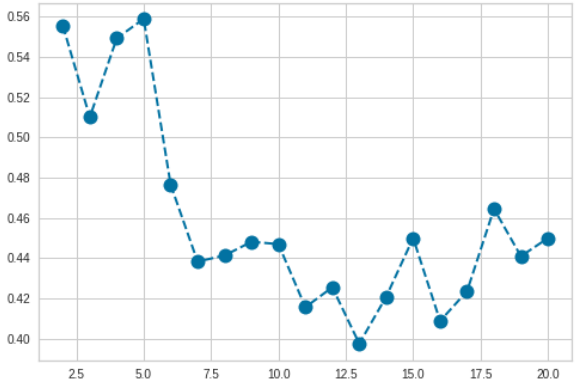

Figure 23: `Silhouette` score with incremental k

The final clusters obtained have the same meaning as those of k-means and hierarchical clustering, the 3D prospective is shown in Figure 27.

4.6 Clusters visualization

In Figures 25, 26, 24 and 27 are shown 3D representations using PCA for each method used.

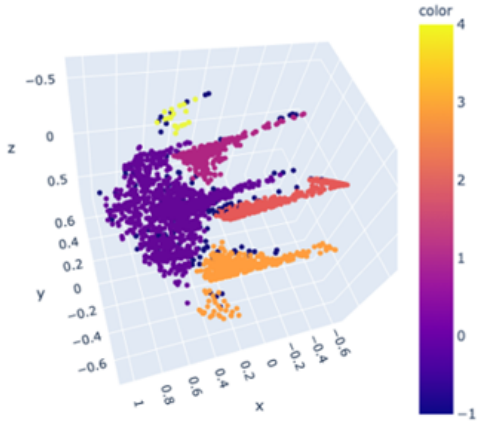

Figure 24: 3D PCA of the DBScan clusters (-1 is the Noise)

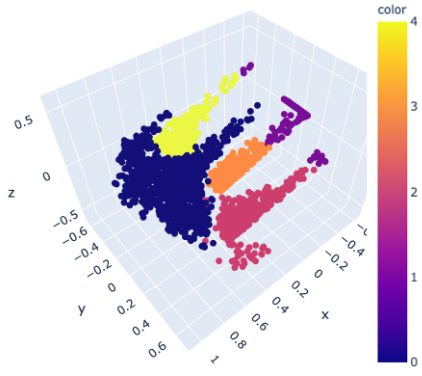

Figure 25: 3D PCA of the K-means clusters

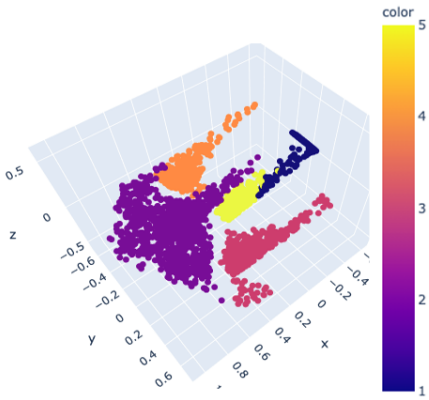

Figure 26: 3D PCA of the Hierarchical Clustering clusters

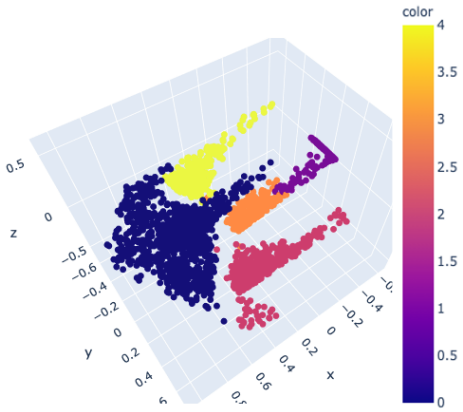

Figure 27: 3D PCA of the Self-Organizing Map clusters

5 Classification

In this section we discuss the task of classifying the customers according to their spending behaviour. The customers were divided in three classes: `Low Spending`, `Medium Spending` and `High Spending`.
The indicator variables are described in section 5.1 while the labeling of the data is discussed in section 5.2.

5.1 Indicator variables

For the classification task we used a subset of the indicator variables used for the clustering, described in section 4.1. Differently from the clustering, these indicators were computed considering the monthly activity of a customer rather than the weekly activity. This choice is motivated by the assumption one month is the minimum observation period to have enough information to classify the customer's profile.
For the classification indicator variables we applied the same pre-processing used for the clustering indicators, i.e.: outliers detection and deletion, standardization and normalization and finally correlation analysis. The resulting list of indicators used is: `NumBasket_mean`, `DistinticProducts_mean`, `Qta_mean`, `Qta_entropy`, `Sale_mean`, `Sale_entropy`.

5.2 Labeling

The customers have been labeled according to the total amount spent, averaged by month for the same reason described in Section 5.1:

$$V_c = \frac{\sum_{i=1}^{N_c} \texttt{Tot}_c^i}{M_c}$$

where V_c is the indicator value for the customer c, N_c is the total number of transactions made by the customer, $\texttt{Tot}_c^i$ is the total cost ($\texttt{Sale} \times \texttt{Qta}$) of transaction i of customer c and M_c is the number of months in which the customer have at least one transaction.
We used M_c instead of the total number of months in the period of observation in order to give more importance to the single purchases, rather than the frequency or regularity of purchases. Indeed, we consider a customer as `High Spending` if, when they come to buy something, they spend a lot, while customers that buy frequently but spend little are considered lower-spending.

The actual label is then computed discretizing V_c in three categories:

$$l_c = \begin{cases} \texttt{Low Spending} & \text{if } V_c \leq 140 \\ \texttt{Medium Spending} & \text{if } 140 < V_c \leq 350 \\ \texttt{High Spending} & \text{if } 350 > V_c \end{cases}$$

The thresholds were selected in correspondence of the 1st and 3rd quartile of V_c distribution among all customers, hence resulting in an unbalanced dataset with the majority of the customers labeled as `Medium Spending`.

5.3 Model evaluation

The dataset has been divided into working set and test set containing, respectively, 80% and 20% of the data.
For each model, a model selection phase was performed, using a k-Fold Cross-Validation approach with k=5. In all the cases, the same folds were used in order to make the comparison among models fair.
Since the dataset is unbalanced towards `Medium Spending` customers, the weighted multiclass F1 score have been used as score function for the model selection, in order to penalize models biased on the most frequent class. Moreover, two different strategies were used to make the training more balanced: dataset oversampling and data point weighting, according to their class frequency in the dataset.

5.4 Classification models

All the models described in the following have been trained according to the validation schema described in section 5.3. Each model has then been retrained, using the best hyperparameters, on the whole training set, therefore exploiting as mush as possible the available data.
For each model, after the final training, an analysis was performed to understand when the model fails and why. This analysis has been performed using the test set.

5.4.1 Decision Tree

Decision Trees are fairly simple models which however are extremely prone to overfitting if no constraint is set on the growth of the tree. Table 7 shows the value for the parameter in the grid search. As strategy to deal with the unbalanced dataset, the data points have been weighted according to their class frequency.

Parameter	Description	Values
`criterion`	Splitting criterion	`Gini`, `Entropy`
`min_samples_split`	Minimum number of samples in a node to perform a split	15%, 10%, 5%, 1%
`min_impurity_decrease`	Threshold on the minimum impurity decrease	0, 0.025, 0.05, 0.075, 0.1
`max_depth`	Maximum tree depth	5, 10, 20

Table 7: DecisionTree grid search parameters (120 parameter combinations)

The best model uses `Entropy` as splitting criterion, 10 as maximum depth, 1% of samples as `min_samples_split` and no constrain on the impurity decrease. It achieves 0.91 and 0.85 weighted F1 score on training and validation set respectively.

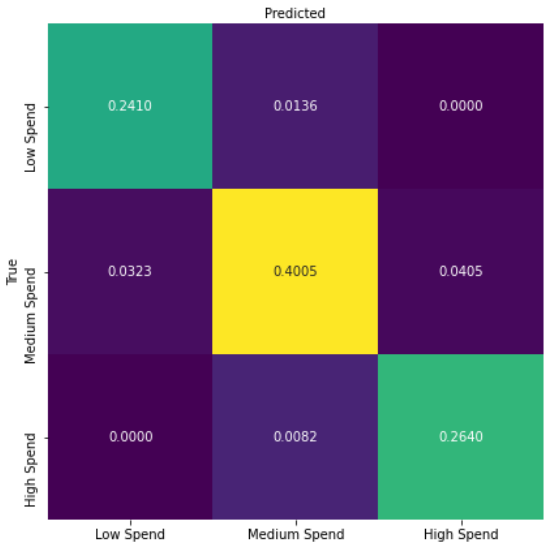

Figure 28: Decision Tree confusion matrix for training data

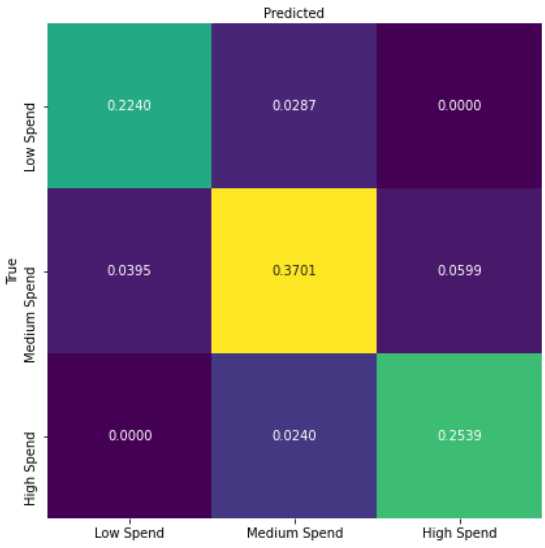

Figure 29: Decision Tree confusion matrix for test data

Visualizing the tree, the topmost nodes always perform a split on `Qta_mean`, `DistinctProduct_mean` or `Sale_mean` while, in the lower part of the tree, also the entropies are used. `NumBaskets_mean` seems to never appear in the tree which might imply the irrelevance of the attribute for the task. Indeed, the `importance` of `NumBaskets_mean` for the classifier is zero, while the most important feature is `Qta_mean`. Figures 28 and 29 show the confusion matrices for the training and test data respectively. The majority of the errors are `Medium Spending` customers being classified as `Low Spending` or `High Spending`, while just few customers from these two classes are confused with `Medium Spending` customers. Indeed, the

second class has an higher precision (few false positives) while the other two classes have an higher recall (few false negatives). This could be caused by the higher weight given to the 1st and 3rd class for which a missclassified example has an higher cost than a missclassified example for the 2nd class. Interestingly, no `Low Spending` customer is confused as `High Spending` and vice-versa, showing that the model is able to catch the ordering among the labels.
The test set results are similar to training set results but, of course, with an higher error rate, which could be caused also by the presence of outliers.

5.4.2 Random Forest

In order to improve the Decision Tree performance, we tested also Random Forest on the classification task. Table 8 shows the values for the parameters in the grid search. Since the overfitting is easily controlled by the randomness in the Decision Trees trainings, the only parameter limiting the trees' growth is `max_depth`.

Parameter	Description	Values
`criterion`	Splitting criterion	`Gini`, `Entropy`
`max_depth`	Maximum tree depth	10, 20, 30
`n_estimators`	Number of estimators	10, 50, 100

Table 8: RandomForest grid search parameters (18 parameter combinations)

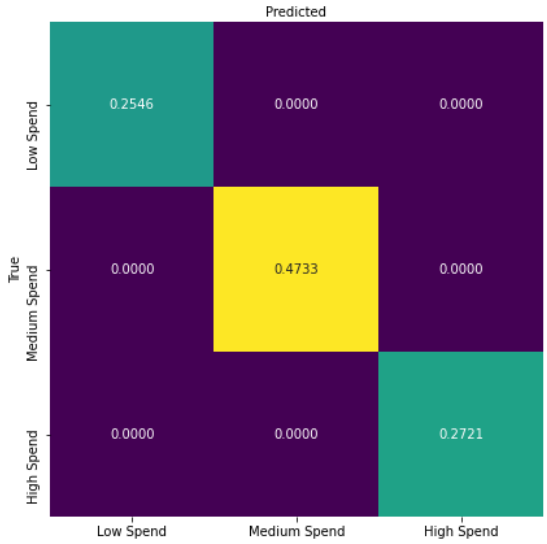

Figure 30: Random Forest confusion matrix for training data

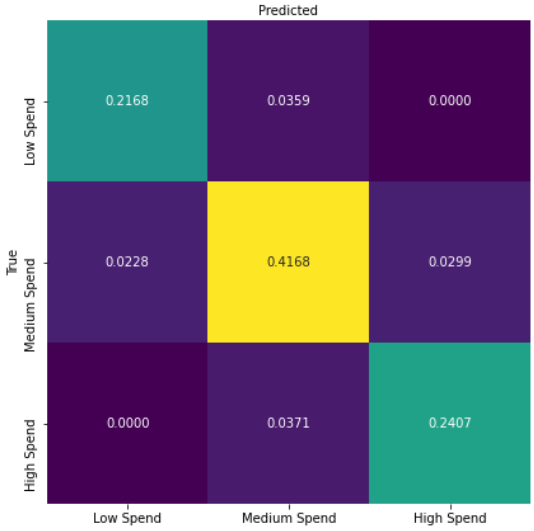

Figure 31: Random Forest confusion matrix for test data

The best model has 100 estimators, with `Entropy` as splitting criterion and maximum depth equal to 30. It achieves perfect score on the training set while 0.892 weighted F1 score on the validation set.
Again, the most important feature is `Qta_mean`, followed by `DistinctProduct_mean` and `Sale_mean` while `NumBaskets_mean` is still the least important feature.
Figures 30 and 31 show the confusion matrices for the training and test data respectively. Having a perfect score on the training set no errors are reported on the training set confusion matrix. However, on the test set the errors are still present, this time with an inverted tendency: the majority of the errors are now `Low Spending` or `High Spending` customers being confused as `Medium Spending` ones.

5.4.3 K-nearest neighbors

The parameters explored for KNN are reported in the table 9. As strategy to deal with the unbalanced dataset, oversampling has been used.

Parameter	Values
`Number of neighbors`	2, 4, 8, 16, 32, 64, 128, 256
`Distance metric`	Euclidean, Chebyshev, Minkowski (with $p \in \{3, 6, 12\}$)
`Weight function`	Distance, Uniform

Table 9: K-nearest neighbors grid search parameters

The experiments show that weighting the points by the inverse of their distance, so letting closer neighbors have a greater influence, leads to a general improvement of the performance, while the best metric distance seems to be the Euclidean one. Indeed, the best model is obtained with `K` equal to 16 and Euclidean as distance metric. For obvious reasons, the model achieves a perfect score on the training set, while it gets 0.901 weighted F1 score on the validation set.

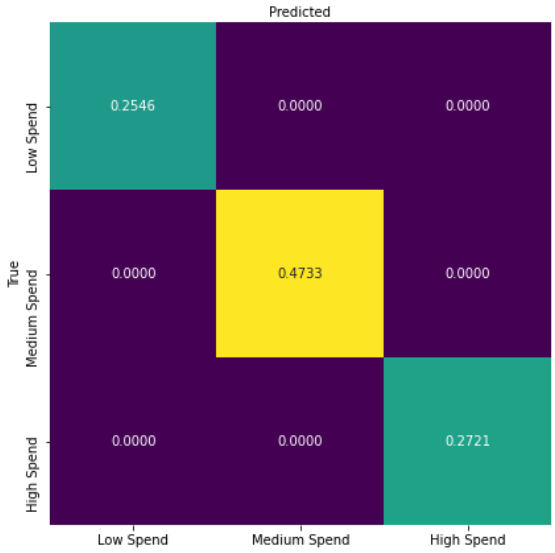

Figure 32: K-nearest neighbors confusion matrix for training data

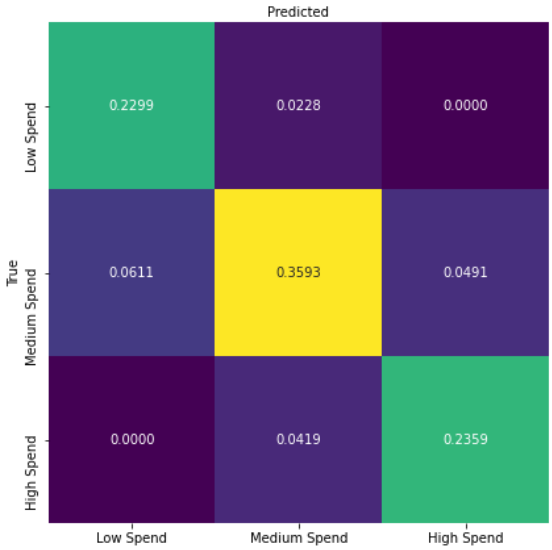

Figure 33: K-nearest neighbors confusion matrix for test data

Figures 32 and 33 show the performances on the training and test set respectively. Differently for Random Forests, the majority of the errors are `Medium Spending` customers being missclassified as `Low Spending` or `High Spending`. However, the overall error rate seems to be higher than the error rate for the Random Forest, meaning that the model still overfits the training data.

5.4.4 Rule Based

The algorithms used for the rule base model are RIPPER and IREP. Since both support only two output classes, one positive and one negative, we decided to create a one-vs-all model for each label, resulting in three models with two outputs labels. In order to combine the models, they have been given a priority so that no conflicting predictions are generated and only the first positive answer is considered valid. The priorities were assigned based on the performances obtained during the training of the individual models. As default class, in case all three models give a negative answer, the `Medium Spending` one is used, being the most frequent in the dataset.
Table 10 shows the hyperparameters values used during the grid search. As strategy to deal with the unbalanced dataset, oversampling has been used.
In general, `RIPPER` algorithm gave much better results than `IREP` algorithm. Indeed, `IREP` was not able to achieve a score greater than 0.80 on the validation set, while, with `RIPPER`, the score was almost always greater than 0.80.

Figures 34 and 35 shows the confusion matrices for the training and test set respectively. In both cases, most of the errors are `Low Spending` customers being classified as `Medium Spending` or `High Spending`,

	prune_size	RIPPER iterations
RIPPER	0.3 - 0.5 - 0.6	1 - 3 - 5 - 7
IREP	0.3 - 0.5 - 0.6	-

Table 10: Rule based model parameters

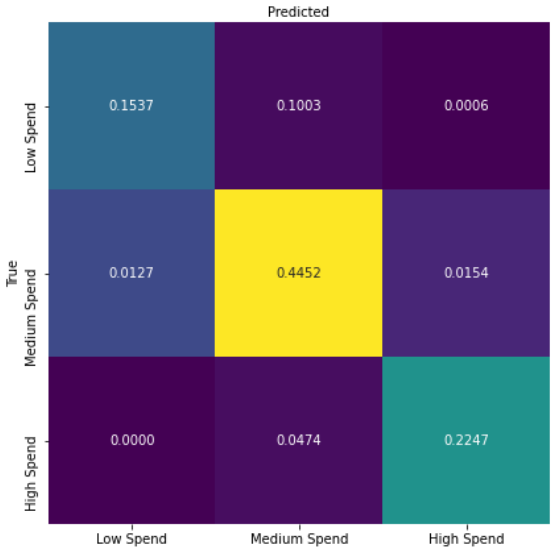

Figure 34: Rule Based system confusion matrix for training data

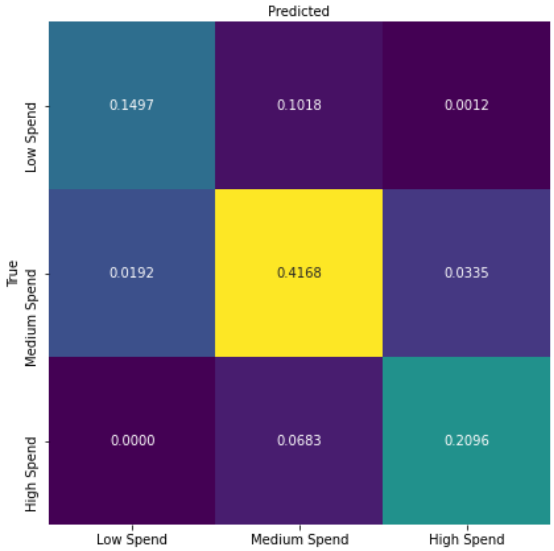

Figure 35: Rule Based system confusion matrix for test data

while the class higher recall is the `Medium Spending`. Indeed, the model for `Medium Spending` customers has the higher priority, while the model for `Low Spending` customers has lowest priority, which also explains why some of them are confused with `High Spending` customers.

5.4.5 Support vector machine

Another model analyzed for the classification task is SVM, Support vector Machine. The parameters are reported in the table 11. Note that Gamma refers to the coefficients for rbf, poly and sigmoid kernels. The value `scale` for the coefficient means that it is computed as $\frac{1}{\#features * \#samples}$. As strategy to deal with the unbalanced dataset, the data points have been weighted according to their class frequency.

Parameter	Values
`Kernel`	rbf, poly, sigmoid
`C`	0.01, 0.1, 1, 2, 4, 8
`Gamma`	scale, 0.01, 0.1, 16, 32

Table 11: Support vector machine grid search parameters

The experiments show that using the polynomial kernel function leads to a general improvement in the performance. Note that the degree of the polynomial kernel has been set to 3. The best model, obtained with kernel `poly`, `C`=1 and `Gamma`=16, has 171 support vectors for the `Low Spending` class, 528 for the `Medium Spending` one and 135 for the `High Spending` one.
Figures 36 and 37 show the performance on the training and test set. They highlight the same behaviour of KNN, with the difference that, on the training set, a perfect score was not achieved, while, on the test set, a slighter better results are obtained.

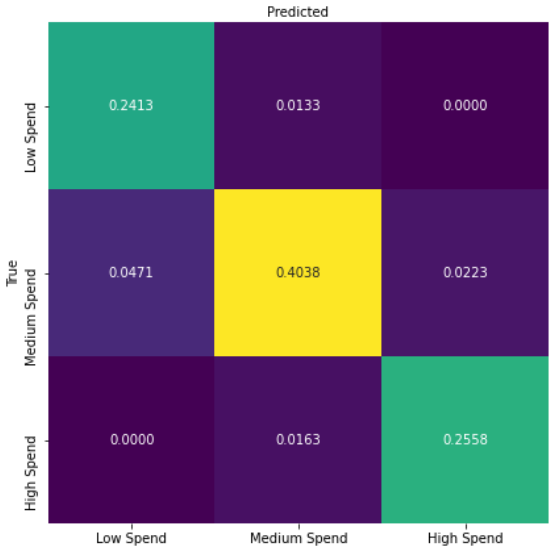

Figure 36: Support vector machine confusion matrix for training data

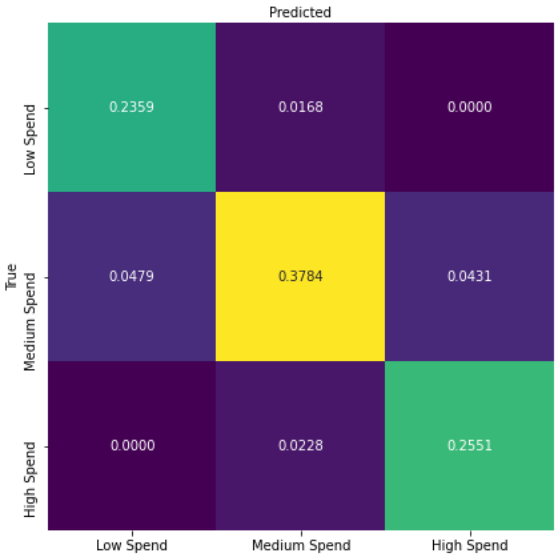

Figure 37: Support vector machine confusion matrix for test data

5.4.6 Naive Bayes

Unlike the other models, the only parameter controlling Naive Bayes is the likelihood distribution, which was set to a Gaussian distribution, being the most appropriate among those available. Since the Gaussian Naive Bayes requires no parameter, no gird search was performed and only one model was trained. As strategy to deal with the unbalanced dataset, oversampling has been used.

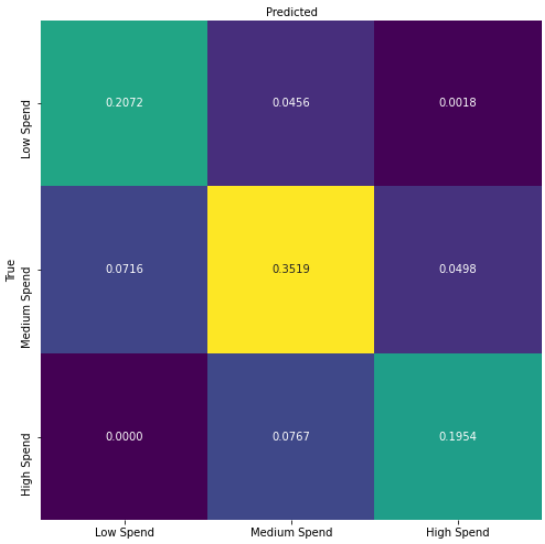

Figure 38: Naive Bayes confusion matrix for training data

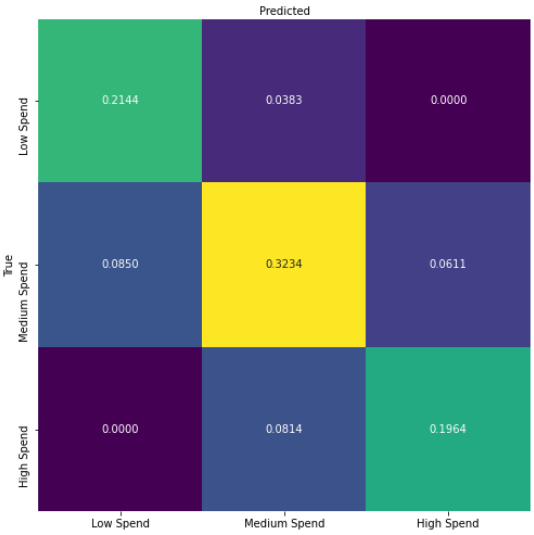

Figure 39: Naive Bayes confusion matrix for test data

Figures 38 and 39 show the performances on the training and test set. The performances are worse than those of most of the models, although it is not clear if it is better than Rule Based Classifier. Despite the usage of oversampling, `Low Spending` and `High Spending` customers are not recognized in many cases. Indeed the recall of the two classes is much lower than the recall of the second class. Interestingly, on the test set, `Low Spending` customers are classified better than on the training set, while the performance for the other two classes decreases.

5.5 Final models

Table 12 shows the best parameters combination for each model. Among them, the best one according to the validation schema is KNN, with Random Forest and SVM as second and third models, although, as we discussed previously, KNN seems to perform slightly worse on the test set than the other two models.

Algorithm	**parameters**	**Tr avg**	**Tr std**	**Vl avg**	**Vl std**
KNN	`K`: 16 `Distance metric`: `Euclidean`	1.0	0.0	**0.902**	0.008078
Random Forest	`splitting_criterion`: `Entropy` `max_depth`: 30 `n_estimators`: 100	1.0	0.0	0.892	0.0099
SVM	`kernel`: `poly` `C`: 1 `gamma`: 16	0.903	0.0017	0.890	0.0075
Decision Tree	`splitting_criterion`: `Entropy` `maximum_depth`: 10 `min_sample_split`: 0.01 `min_impurity_decrease`: 0	0.91	0.0053	0.85	0.0042
Rule Based Classifier	`algorithm`: `RIPPER` `prune_size`: 0.5 `RIPPER_it`: 5	0.832	0.007	0.816	0.0158
Naive Bayes Classifier		0.756	0.004	0.755	0.011

Table 12: Best hyperparameters combination

6 Sequential Pattern Mining

In this section we discuss the task of finding sequential patterns in the purchasing history of the customers. For this task, the customers are represented as the list of their baskets, each of which is represented as the list of the products it contains.
For each customer, the basket are sorted chronologically. However the information about the exact date and time is not used in the SPM algorithm and, therefore, it has been dropped.
The lists representing the baskets contain the `ProdID` of the corresponding products, sorted increasingly. According to the meaning of letters in the `ProdID` (Section 2), all the alphabetical characters were removed from the `ProdID`s obtaining only numerical IDs.

6.1 Experiments and Results

The algorithm used to find the sequential patterns is the *Apriori* algorithm, controlled only by the parameter `minSupport`, which is the minimum number of customers supporting a pattern in order to define it as frequent.
Table 13 shows the number of found patterns with different value for the support threshold. Among the 2-items patterns supported by at least 5% of the customers, 31 are made of a single element, while 8 are made of 2 elements containing one item each, whereas, for those supported by at least 150 customers, 124 are made of one element and 109 of two. The only two 3-items patterns found contain three elements, all sharing the same item (`ProdID`s 85099 and 85123), and have a support of 178 and 166 customers.

`minSupport`	**1-item patters**	**2-items patterns**	**3-items patterns**
10% (417)	30	-	-
5% (208)	251	39	-
∼3% (150)	492	233	2

Table 13: Number of patterns found with different `minSupport`

For the single items, the two most bought ones have `ProdID` equal to 85123 and 85099, with a support of 746 and 696 customers respectively (∼17% of the customers). The same products appear in some 2-items patterns and in the only two 3-items patterns found. In all the cases, the pattern contained single item elements, with the same item in all the elements. Looking at the frequencies, it seems that customers that buys one of these products are likely to buy it again later. Indeed, the *Lift*, in this case computed as the support count of a sequence divided by the support count of one of its subsequences, shows how the probability of buying the product increases from the 17-18% to the 41-44% if it has been bought already once.
The same pattern can be observed with other products, still among those bought by at least 10% of the customers. In particular, of the 8 2-items patterns with two elements, 7 contain the same product in both elements, while just one contain two different products. For those in the first case, the probability of being bought increases from 10-16% to 35-50% if they have been bought previously. For the products in the second case, the product descriptions reveals that the two products are actually of the same kind (`JUMBO BAG ...`), thus giving, again, credit to the idea that customers tend to buy the same product of kind of products repetitively.
A similar behaviour emerges from the 2-items patterns made of just one element. In most of the cases, the products in the pattern belong to the same kind of objects. Indeed, in 27 out of the 31 patterns, the two products differ only for the color or style of the object, as in the case of `ProdID`s with different tailing letters. In the other cases, the products are still related, as in the case of `ProdID` 22577 and 22578 corresponding to the descriptions `WOODEN HEART CHRISTMAS SCANDINAVIAN` and `WOODEN STAR CHRISTMAS SCANDINAVIAN`, both Christmas decorations.

A downside of this analysis is that most of the found patterns have a too low support count to be confident they are actual recurrent patterns, especially in the case of 3-items patterns supported by only 4% of the customers. This, however, shows how no major pattern is present in the dataset, apart from the fact that customers tend to buy again the same products they bought already and that most customers mainly buy a certain kind of product.
A possible enhancement could be to further reduce the number of items in the dataset, grouping them according to the associated `ProdDescr`. In this way, each customer would be characterized only by few items and more interesting patterns could emerge, although on the level of type of products rather than on products themselves.