A.A. 2020-2021

Data Mining: Project Report

Group 1
Gabriele Pisciotta
Antonio Di Mauro
Fabio Murgese

Contents

1 Data Understanding

The dataset contains informations about the puchases of products across the world. In this section we'll explore it analyzing the meaning of the values contained in it and their quality.

1.1 Data semantics

Our dataset is composed of 8 attributes:

1. `BasketID`: the unique ID of each buying session. It's mainly a sequential number, but there are cases in which it starts with 'C'.
2. `BasketDate`: it's a date related to each purchase.
3. `Sale`: a continuous numerical attribute describing the cost of the product.
4. `CustomerID`: a discrete numerical attribute that describes each unique customer.
5. `CustomerCountry`: a categorical attribute representing the country in which the purchase has been done.
6. `ProdID`: a categorical attribute that is the ID of the bought product.
7. `ProdDescr`: a categorical attribute that describes the product or the purchase.

Each row of our dataset represents the detail of a purchase receipt. In total, we have 471910 rows.

1.2 Distribution of the variables and statistics

Here we analyze in details the numerical attributes of our dataset. Even if CustomerID is a numerical attribute, it's not meaningful for us, so we'll describe only mean and variance of `Qta` and `Sale`. For both of them, we have 471910 non-null values:

- `Qta`: 4.03 ± 83.76
- `Sale`: 10.71 ± 231.35

Here we analyze the purchases for each dates during different periods of observation.

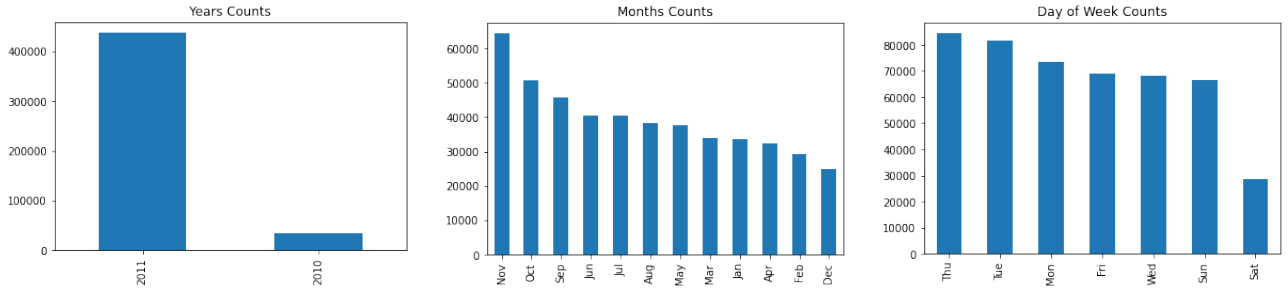

Figure 1: Dates analysis

As we can see from Figure 1.1, we can see that almost all the purchases were done in 2011. From Figure 1.2 we can see that we have a peak in November and the least amount in December, otherwise we can notice an almost constant behavior. From Figure 1.3 we can state that on Saturday we have half the amount of the standard. In Figure 2 we can observe that the vast majority of the customers are citizens of the United Kingdom, so probabily these records were taken in the UK, while recording also tourists purchases. We can see the unique values of Table 1.

BasketID	CustomerID	CustomerCountry	ProdID
22190	4372	37	3684

Table 1: Unique values.

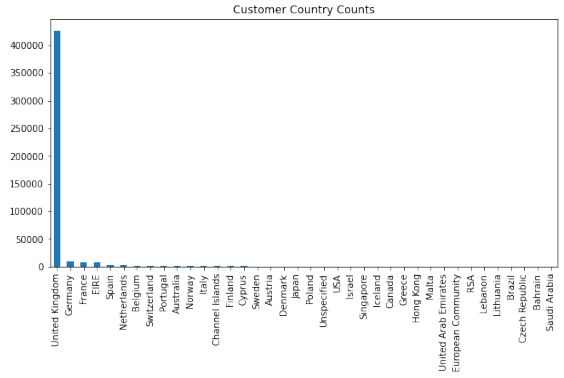

Figure 2: Customer country distribution.

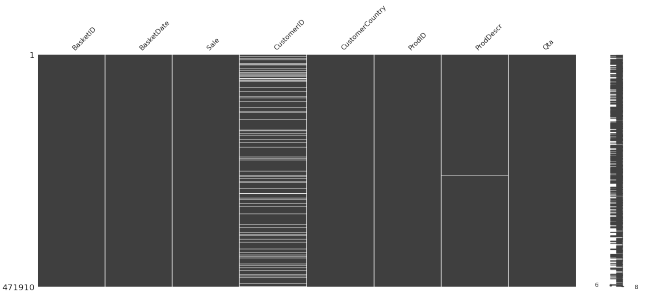

Figure 3: Missing values.

We created a column `Amount` representing the total amount of money spent for each product in a basket and also analyzed this quantity with respect to the different quarters (Figure 6). The Amount's values lies in 2247.01 ± 5624.24.
We then analyzed the total quantities bought for each product. The average quantity of sold items is 1328.05 ± 2936.95 for the entire period, despite the median is 375.50. We can see that the mean expense for product is 3.84 ± 17.18 and that the median is 1.95. The first quartile of the products costs is 0.98 and the fourth quartile is 3.81. Also min (0.000750) and max (744.14) values have disproportionate total selling values (probably outliers).

1.3 Assessing data quality

Here we analyze the data quality of our dataset. By looking at Figure 3 it is possible to recognize the missing values of the dataset (identified by the white lines). The attribute `CustomerID` is the field with most missing values.

With the boxplot we can easily see how strange it is that very few products are so far from all the others, in terms of pricing. These are probably outliers and we'll trat them properly in the next stages.

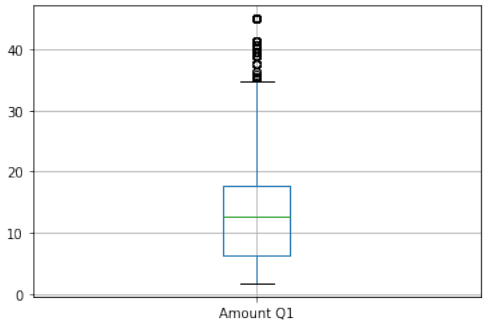

Figure 4: Money Spent Q1 box plot.

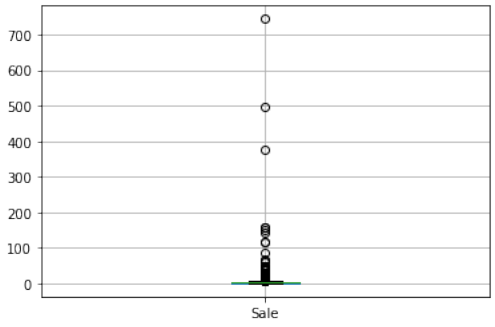

Figure 5: Products box plot.

Let's now analyze the negative quantities in the dataset. In Table 2 we can see the products with negative quantities.
From a first view it may seem that all the `BasketID` starts with a letter 'C', but we've discovered that all the rows having negative quantities have `BasketIDs` that starts with 'C', or the vast majority of them, and also with '5'.

BasketID	BasketDate	Sale	CustomerID	ProdID	ProdDescr	Qta
C536379	2010-01-12 09:41	27.50	14527	D	Discount	-1
C536383	2010-01-12 09:49	4.65	15311	35004C	SET OF 3 COLOU....	-1
C536391	2010-01-12 10:24	1.65	17548	22556	PLASTERS IN TIN....	-12
C536391	2010-01-12 10:24	0.29	17548	21984	PACK OF 12 PIN....	-24
C536391	2010-01-12 10:24	0.29	17548	21983	PACK OF 12 BLU....	-24

Table 2: Negative values.

We noticed that `BasketIDs` that starts with '5' often don't present the `CustomerID`, while for the `BasketIDs` starting with 'C' we have only 821/9084 unspecified `CustomerIDs`. For each `BasketID` we can have multiple rows, so in order to get the right amount of `CustomerIDs` missing, we have to drop all the duplicates for each `BasketID`. After the drop, we get the proper amount of `CustomerIDs` missing: 100 / 3754. What if we remove the 'C'? Does exist any corrispondence in the main dataframe? According to the previous analysis there's no corrispondence between the `BasketIDs` having positive quantities and `BasketIDs` having negative quantities. So we can say that those are disjoint sets.
Also, we noticed that the description for the same `ProdID` may vary, e.g: product *47566B* has 2 different descriptions that contains the reasons of that negative transaction: *"reverse previous adjustment"* and *"incorrectly credited C550456 see 47"*; so, we dropped some product description keeping only one. Finally we've got the main products having negative quantities with a single description, as we can see in Table 3.

From the computed statistics the majority of the negative quantities, that lies in between the first and the third quartile, are included into the interval -5/-52 but we have also negative quantities up to 80995 (almost surely an outlier).

We solved all this quality issue by removing duplicates and dropping rows having specific columns to NaN. This was enough to not having anymore inconsistencies such as BasketID

	ProdID	Qta	ProdDescr
0	23630	-1	SET 10 CARDS HANGING BAUBLES 17080
1	85039A	-1	SET/4 RED MINI ROSE CANDLE IN BOWL
2	85036A	-1	GARDENIA 1 WICK MORRIS BOXED CANDLE
3	22833	-1	HALL CABINET WITH 3 DRAWERS
5	22869	-1	NUMBER TILE COTTAGE GARDEN 1

Table 3: Single description table.

with negative quantities that doesn't start with 'C'.

2 Data Preparation

In this section we extract new interesting features for describing the customer profile and his purchasing behavior. We computed:

- l: the total number of items purchased by a customer during the period of observation (`Total products bought`)
- lu: the number of distinct items bought by a customer in the period of observation (`Distinct Products`)
- $lmax$: the maximum number of distinct products purchased by a customer during a shopping session in the period of observation (`Max Products In Basket`)
- E: the *Shannon entropy* of the money spent (`Entropy of money spent`)
- the maximum, minimum and mean amount of money spent for basket
- the total money spent during the period of observation
- the maximum, minimum and mean product price in customer's basket. This value refers just to the product's value, not to the price that the customer has paid (e.g.: just the `Sale`, not `Sale` * `Qta`)
- the quarters analysis concerning the quantities of products bought and the amount of money spent by customers.

2.1 Variables transformations & generation

We modified the original dataset identifying and eliminating the outliers (using the box plot method, e.g.: Figure 6). We also analyzed the amount of money spent on products by the customers, splitting the records by quarters of the year simulating the information needed for a business intelligence task. In Figure 7 we can see the comparison between just 20 customers (for the sake of presentation) representing the total quantities bought in the 3 different quarters of the year. Also, we made the same analysis for semesters and trimesters, having similar results.

Figure 6: Pre-post transformation of Amount Q1 - removal of outliers.

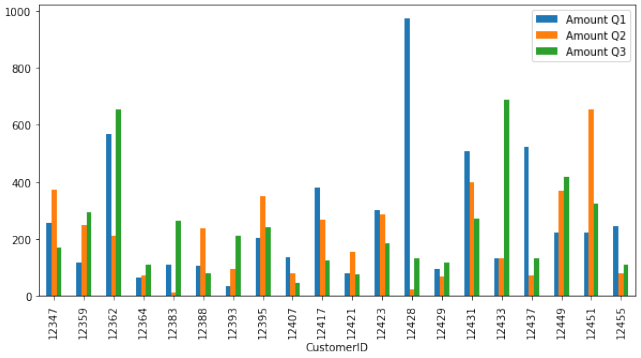

Figure 7: Quarters expense for customers.

2.2 Correlation analysis

As we can see from Figure 8 the quantities and the amounts registered in the same quarter are very highly correlated, so more the products bought, more the expense. While in different quarters we notice slightly different habits in purchases by customers, still having a quite high correlation. We have groups of features that are correlated (e.g.: correlation ≥ 0.80), for example the quantity of products bought in a quarter and the money spent. We take this information into account for the next steps, in order to avoid to add redundant information in the clustering and the predictive analysis, that may lead to wrong deduction.

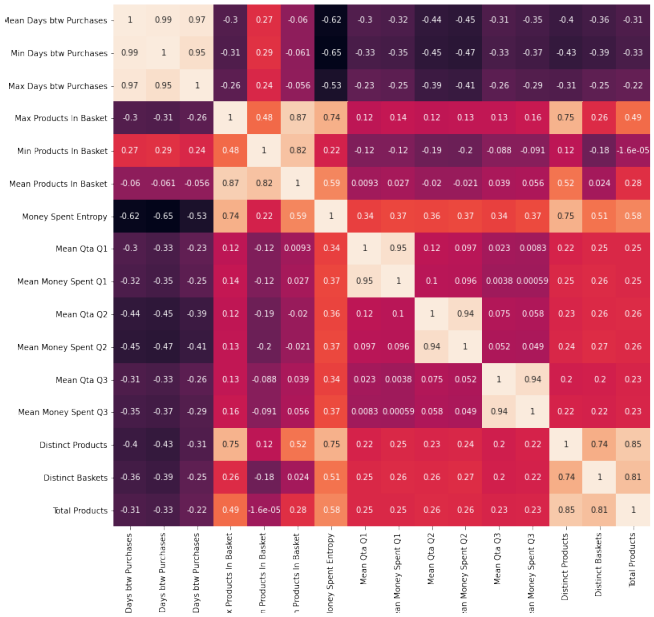

Figure 8: Correlation matrix.

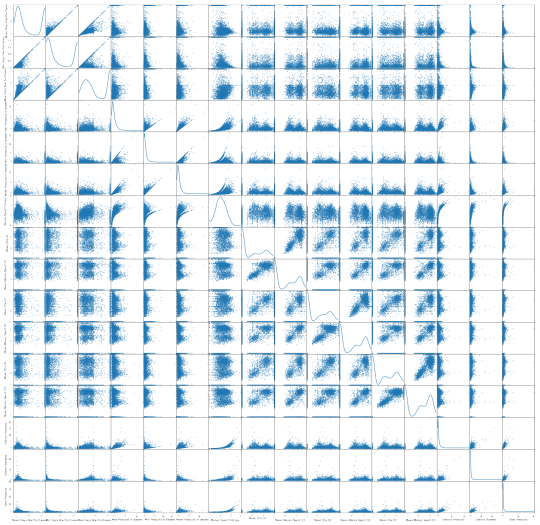

Figure 9: Distribution of the variables.

2.3 Statistics of the final dataset

We renamed the following features with an incremental number: `Mean Days btw Purchases`, `Min Days btw Purchases`, `Max Days btw Purchases`, `Max Products In Basket`, `Min Products In Basket`, `Mean Products In Basket`, `Money Spent Entropy`, `Mean Qta Q1`, `Mean Money Spent Q1`, `Mean Qta Q2`, `Mean Money Spent Q2`, `Mean Qta Q3`, `Mean Money Spent Q3`, `Distinct Products`, `Distinct Baskets`, `Total Products`.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
count	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035	4035
mean	340.06	312.02	380.38	15.58	7.13	10.84	2.84	3.92	6.10	4.11	6.98	4.81	7.65	30.42	3.75	295.86
std	288.71	311.62	264.58	14.39	8.25	9.49	1.27	4.35	6.41	4.14	6.72	4.29	6.36	41.12	6.27	591.50
min	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	2
25%	87.76	28	143.50	6	2	5	2.01	0	0	0	0	0	0	8	1	56
50%	167.83	125	278	12	5	8.40	2.84	2	5.63	4	8.11	4.85	9	17	2	138
75%	697	697	697	20	9	14	3.73	8.33	13.04	8.06	14.04	8.95	13.80	39	4	342
max	697	697	697	196	145	145	7.8	12	17.4	12	17.7	12	17.40	976	183	23901

Table 4: Variable statistics

3 Clustering Analysis

In this section we analyze four different Clustering techniques applied to our data:

- *KMeans*,
- *DBSCAN*,
- *Hierarchical Clustering*,

- *MBSAS* (from PYCLUSTERING library).

3.1 Outliers Removal with DBSCAN

Before proceeding to the effective exploratory analysis with the different clustering techniques, we exploited *DBSCAN* to detect and remove outliers from our dataset, thanks to its inherent ability to do so: it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions. We chose to consider this "noisy" point as proper outliers of our dataset, and so deleting them before proceeding, and it's possible to see the difference in Figure 10.

Figure 10: Before and after outliers removal.

3.2 KMeans

In order to find the best clustering, we considered four types of analysis:

1. on the entire dataset,
2. on a manually selected subset of the dataset based on parallel coordinates,
3. on a manually selected subset of the dataset,
4. on a subset of the dataset including just the most uncorrelated features.

For each analysis we run a grid search to find the best hyper-parameters (**numberOfClusters** $\in [2, 20]$), evaluating the results with respect to the Knee curve (*SSE*). The best clustering is obtained with 2), taking into account a subset of the features selected using the parallel coordinates analysis generated on the entire set of features: `Distinct Products`, `Mean Days btw Purchases`, `Min Days btw Purchases`, `Max Days btw Purchases`, `Total Products`.

From the parallel coordinates plot in Figure 11 we noticed that cluster 0 and 2 follow the same trend, apart from the last attribute: this means that the centroids have similar values for these attributes and so could exists some sort of correlation among this two clusters. Instead, cluster 1 follows a different characterization with respect to the other two, with features assuming different values.

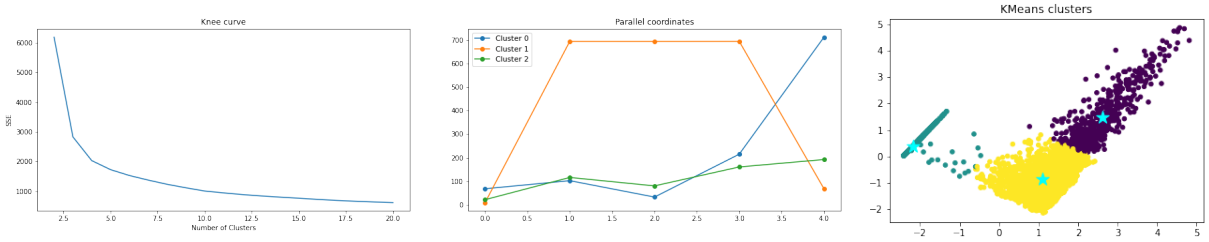

Figure 11: Knee-curve, parallel coordinates and clusters of KMeans analysis 2)

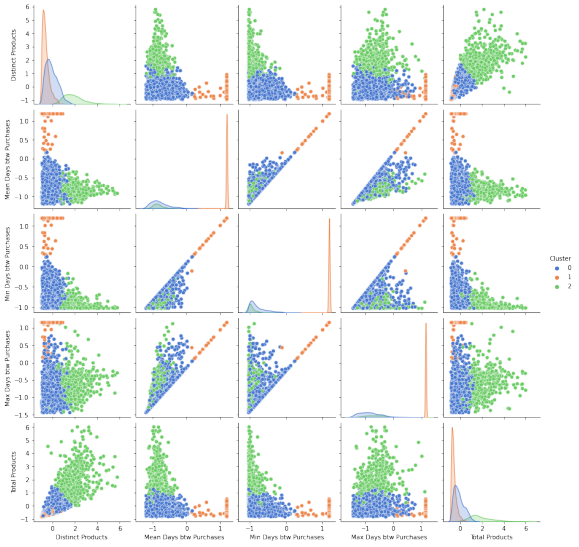

Table 5: Pairplot of features - 2).

Number of Clusters	3
Num. of init	20
Max iters.	200
SSE score	2833.905
Silhouette score	0.666
Clusters	*0*: 570, *1*: 1518, *2*: 1636

Table 6: Best clustering obtained.

3.2.1 Characterization

We took into account only the feature `Mean Days btw Purchases`, instead of considering also `Min Days btw Purchases` and `Max Days btw Purchases`, because have similar distributions.
Cluster 0's customers buy a number of distinct products that are in the mean (`Distinct Products`), they buy frequently with values around -1 (`Mean Days btw Purchases`) and they buy a lot of products on average with values around +2 (`Total Products`).
Cluster 1's customers buy less distinct products with respect to the other clusters with values around -1 (`Distinct Products`), they buy rarely with values around +1 (`Mean Days btw Purchases`) and they buy a lot of products on average with values around +2 (`Total Products`).
Cluster 2's customers buy more distinct products with values around +2 (`Distinct Products`), they buy rarely with values around +1 (`Mean Days btw Purchases`) and they buy less amount of total products with values around -1 (`Total Products`).

3.3 DBSCAN

We analyze three different approaches using DBSCAN picking the best hyper-parameters according to the Silhouette score of the clusters including the noise and without the noise:

1. on the entire dataset,
2. on a manually selected subset of the dataset (`Mean Days between Purchases`, `Money Spent Entropy`, `Mean Products In Basket`),
3. on a reduced dataset (PCA=2).

We chose to reduce the dataset to 2 dimension with the PCA because of the analysis of the variance explained in figure 7. According to this analysis, with two dimension we can describe almost 80% of the variance of the initial dataset, so we can state that it's the best trade-off between the dimensionality and the amount of information expressed in those dimensions.

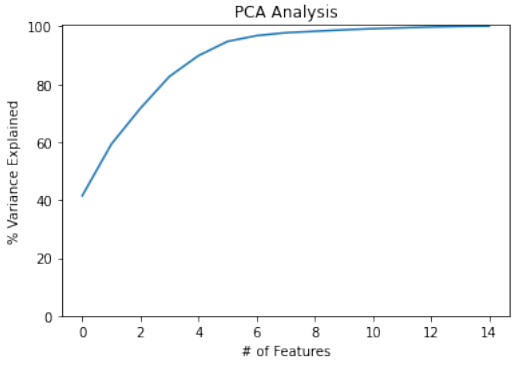

Table 7: Variance explained with PCA

Number of Clusters	**2**
Samples	40
Eps	0.4
Silhouette	0.475
Silhouette (w/o noise)	0.536

Table 8: Best clustering obtained.

We also optimize the hyperparameters of the DBSCAN algorithm, formerly **eps** and **n_samples**, in the following ranges:

- eps: 0.1, 0.2, 0.3, 0.4, 0.5
- n_samples: 1, 2, 5, 10, 15, 20, 25, 30, 35, 40

The resulting best approach is 2), according to the silhouette score on the clusters without noise.

It's possible to see that DBSCAN can cluster exploit the density of the points that in this case it's easy to interpret as two different clusters, thanks to the fact that in the previous step we already removed some outliers that lied in the middle of the space between the two blobs.

3.3.1 Characterization

We compute the pairplot (Figure 12) related to that dataset, picking all the customers that aren't noise.
Cluster 0's customers have the Money Entropy higher than the ones in Cluster 1. They also have (on average) more products in a baskets. For the frequency of buying (Mean Days between

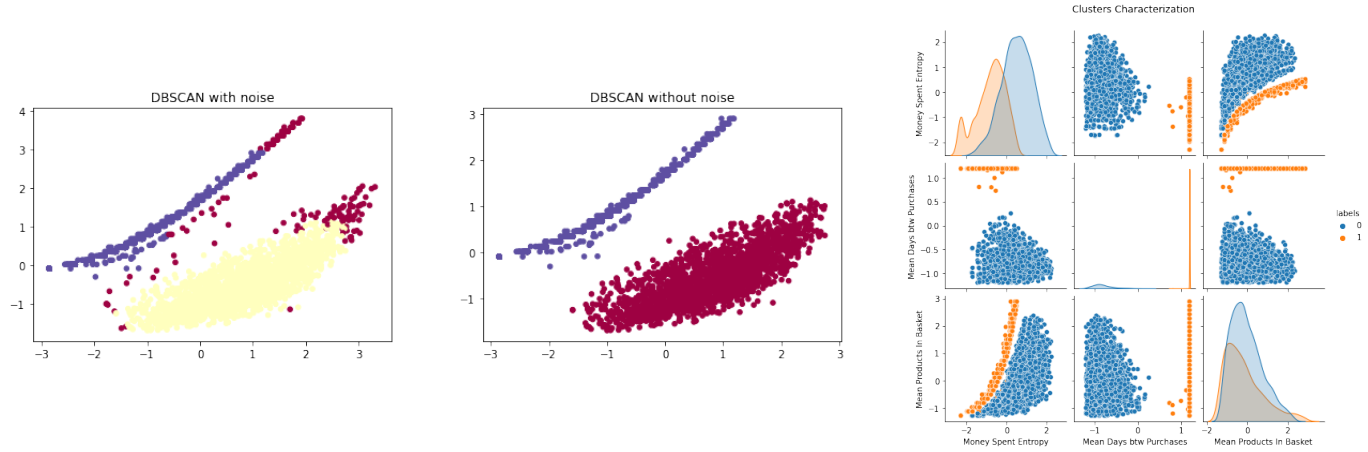

Figure 12: Resulting clustering applying DBSCAN

Purchases), we can note that the customers are divided between those who buy frequently (feature with value near -1) and those who buy rarely (feature with value near +1).
Cluster 1's customers, instead, have the Money Entropy lower than the ones in Cluster 0, the frequency is very high (with values distributed around -1) and they have on average a bit less products in baskets.

3.4 Hierarchical Clustering

The generation of clusters with the Hierarchical approach has followed an **Agglomerative** strategy, with the usage of *Euclidean* and *Manhattan* distance to compute distance between points in the different clusters, and *Complete*, *Average*, *Single* and *Ward* type of linkage. We decided to operate with **numberOfClusters** $\in [2, 10]$ and plotted the Silhouette score for each clustering result. The constant evidence was that after 4 or 5 clusters generated, the Silhouette scores plot dropped and the composition of the clusters was totally unbalanced, having in results one or two clusters with almost each data point in the dataset, and very few points in the remaining clusters. We analyzed this clustering technique:

1. on the entire dataset,
2. on a manually selected subset of the dataset (`Mean Days between Purchases`, `Money Spent Entropy`, `Mean Products In Basket`),
3. on a reduced dataset (PCA=2).

The best results were obtained on 2); in Table 9 we can see the composition of the best clustering obtained.

In Figure 15 we can notice how clusters are formed using different inter-cluster distances with *dendrograms*, tree-like representations of the arrangement of the clusters produced. We can notice that with Euclidean Average we have points with lower distance and 4 clusters pretty much well defined. With Manhattan Average, instead, we have higher distance between data points and 2 clusters better discriminated (lower distance between higher linkages). With Single

Number of Clusters	4	2
Linkage	*Average*	*Average*
Metric	*Euclidean*	*Manhattan*
Silhouette score	0.457	0.481
Clusters	*0*: 419, *1*: 1834, *2*: 1090, *3*: 381	*0*: 1514, *1*: 2210

Table 9: Clusters composition of the best clustering obtained.

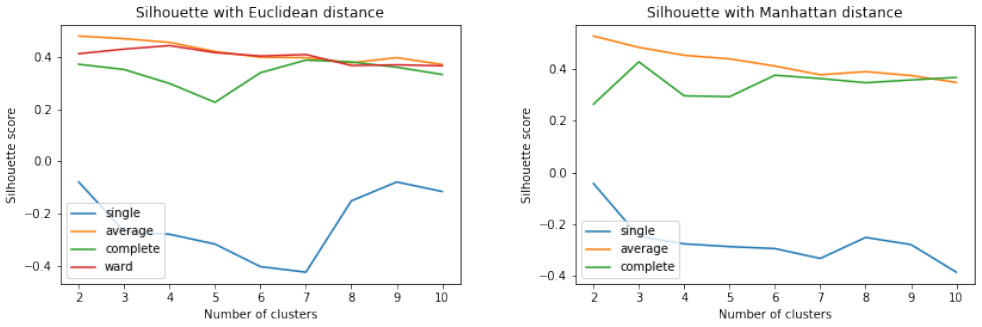

Figure 13: Silhouette score plot.

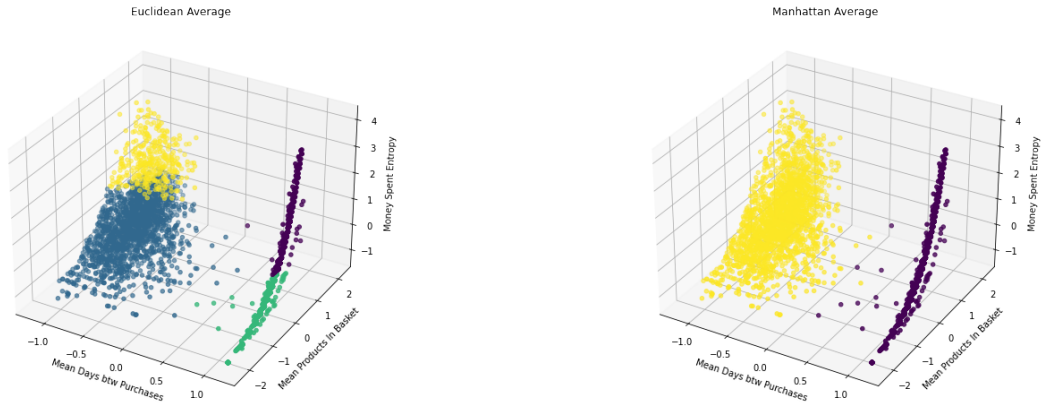

Figure 14: Customers visualization.

linkage, instead, we had the worst results, so we decided not to show these dendrograms. Furthermore, in Figure 16 we can see the results of hierarchical clustering on the dataset reduced with *PCA* (2 dimensions).

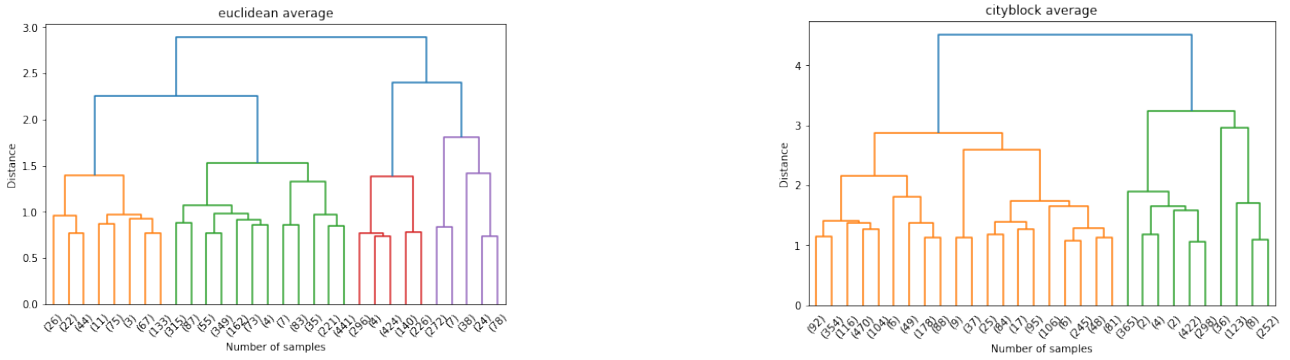

Figure 15: Dendrograms of the best clustering obtained.

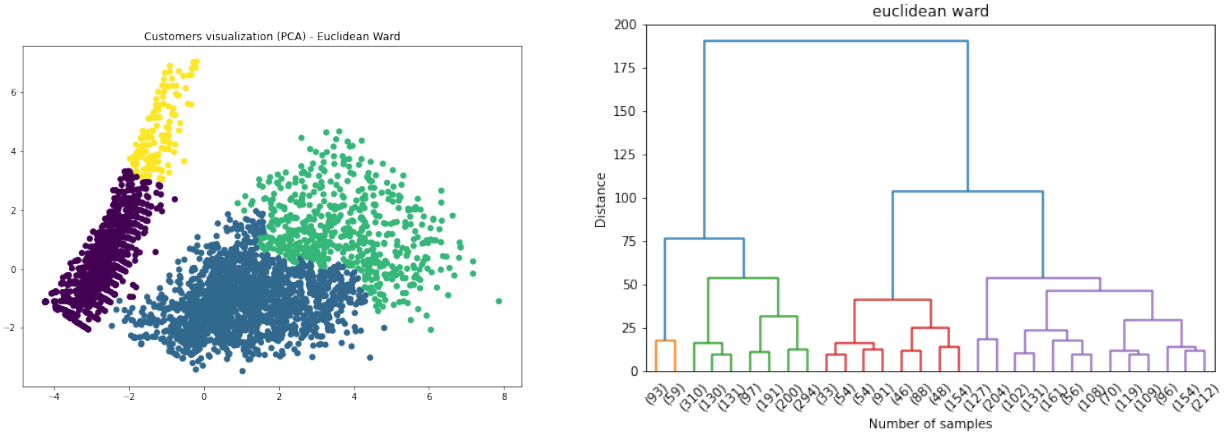

Figure 16: Hierarchical clustering on *PCA* reduced dataset.

3.5 MBSAS (optional)

To use this clustering algorithm, we exploited the PYCLUSTERING library. The hyperparameters we tune are the **threshold** and **n_clusters**, according to the following grid search:

- threshold: [0.1, ..., 1.5, 2.0, 3.0, 5.0, 10.0, 20, 50, 100]
- n_clusters: 2, 3

The threshold specified here is the threshold of euclidean distance between the points. We run three kind of experiment, picking the best hyperparamethers according to the silhouette

1. on the entire dataset,
2. on a manually selected subset of the dataset,
3. on a reduced dataset (PCA=2).

The resulting best approach is 3), according to the silhouette score that is possible to see in Table 11.

It's possible to see from 10 that the clusters that emerges are somehow 'noisy', this is due to

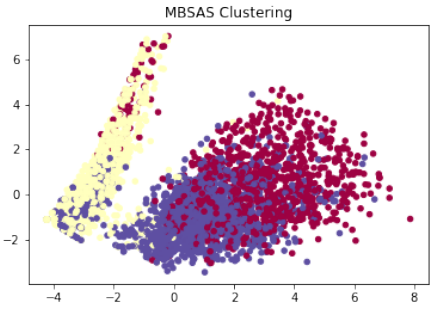

Table 10: Results applying *MBSAS*.

Number of Clusters	**3**
Threshold	5.0
Eps	0.4
Silhouette	0.396

Table 11: MBSAS clusters composition

the nature of the *MBSAS* algorithm which is highly influenced by the order of presentation of the data points and the distance from the representative according to the whole sequence, considering also a distance threshold. It's possible to see from the notebook that the result may vary a lot changing the hyperparameters of the clustering algorithm (upper bound of the number of clusters and distance threshold).

3.6 Best Clustering results

We can see that the best results obtained with a clustering techniques are obtained with KMeans. It's the method that presents the best results according to the silhouette as we can see in Table 6. For what concerns DBSCAN, it is very good at identifying cluster based on their density, and in the Figure 12 emerges quickly its ability to consider the two manifolds. We noticed, from the results in Table 9 that also the Hierachical achieves a good silhouette score. In general, the advantage of agglomerative hierarchical clustering is that it tends to produce more accurate results. The downside is that hierarchical clustering is more time/resource consuming than the others. We in the end consider DBSCAN a really good approach because it have an high silhouette score and its also useful because of its ability to detect noisy points and let us remove them, instead of trying to assign them to clusters.

4 Predictive Analysis

In this section we analyze different classification models we used in order to predict the customer's spending behavior:

- *K-Nearest Neighbors*
- *Support Vector Machine* for Classification
- *Decision Tree*
- *Random Forest*
- *MultiLayer Perceptron*

In Table 12 we can see the algorithms used and the hyper-parameters tried for the experiments. The hyperparameter search has been done with a 5-Fold Grid Search cross validation strategy(2-Fold for MLP), selecting the best according to the mean F1 validation score.

Algorithm	Hyper-parameters
KNN	*n_neighbors* : 5, 20, 50 *weights* : 'uniform', 'distance' *algorithm*: 'ball_tree', 'kd_tree', 'brute'
SVC	*C*: 1, 0.5, 0.1, 0.01, 0.001 *gamma*: 'auto' *kernel*: 'linear', 'rbf' *class_weight*: 'balanced', None
Decision Tree	*criterion* : 'gini' *max_depth* : None *min_samples_split* : 2
Random Forest	*max_depth* : 10, 20, 50, None *max_features*: 'auto' *min_samples_split* : 2, 5, 10 *min_samples_leaf* : 1, 2, 4 *bootstrap* : True, False *n_estimators*: 100, 200 *criterion* : 'entropy', 'gini' *class_weight* : 'balanced', None
MultiLayer Perceptron	*lr* : 0.0001, 0.0002, 0.0003, 0.01, 0.02, 0.03, 0.1, 0.2, 0.3 *lr_decay* : 0.9, 0.8 *decay_steps* : 10000, 100000 *epochs* : 200, 300, 500, 750, 1000 *batch_size* : 5, 10, 25, 50 *lambda* : 1e-5, 1e-6, 1e-7 *hidden_layers* : 2

Table 12: Setup environment of the tested hyper-parameters.

4.1 Label Generation

First of all, we needed to assign different labels to our customers: **high spending, medium spending, low spending**. These labels have been assigned based on a combination of selected features that allowed us to give each record (customer) the proper label. We decided to follow a *Supervised Discretization* approach based on the percentiles of the following features:

- `Money Spent Entropy`
- `Mean Days between Purchases`
- `Mean Products In Basket`
- `Mean Qta` (sum of the mean quantity for each quarter of the year)
- `Mean Money Spent` (idem)

This let us have robust features according to the time of observation and also according to generalization purposes, in order to classify correctly also relatively new users. We can see from Table 13 that we have a sort of Gaussian distribution about the number of examples centered in the medium-spending category. This is reasonable due to the fact that we can assume that this

Customer type	# of examples
high-spending	772
low-spending	681
medium-spending	2271

Table 13: Distribution of examples for each label.

phenomena, with increasing number of customers, we'll have more and more medium-spending ones, while high-spending and low-spending will increase at a slower rate. We can suppose that a good enough explanation for this is the law of large numbers.

4.2 K-Nearest Neighbors

At the end of the cross validation, we picked the best hyper-parameters:

preprocessing	n_neighbors	weights	algorithm	mean_valid_f1
No scaler	5	distance	ball_tree	0.896
Standard scaler	5	distance	ball_tree	0.919

We used these parameters to re-train the classifier on the whole training set and then computed the metrics for both training set and test set.

	F1	P	R
Train	1.0	1.0	1.0
Test	0.915	0.915	0.897

Table 14: Metrics (macro)

	F1	P	R
Train	1.0	1.0	1.0
Test	0.929	0.945	0.916

Table 15: Metrics (macro) (standardized)

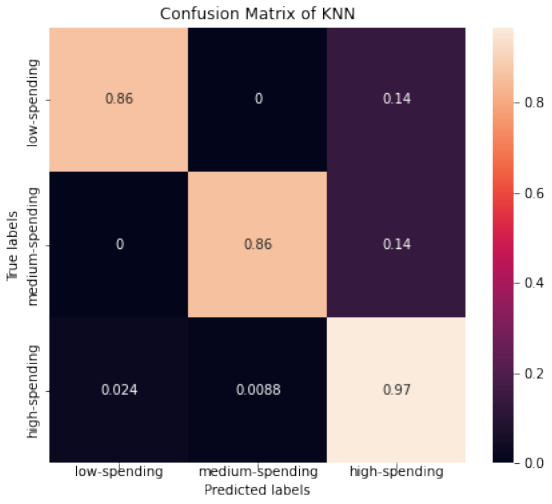

Table 16: Confusion Matrix

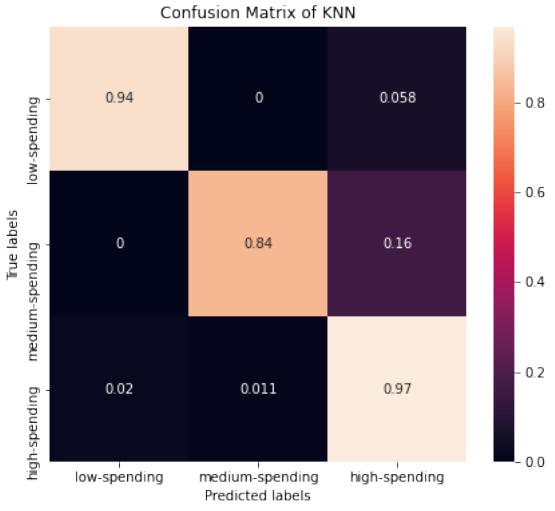

Table 17: Confusion Matrix (standardized)

4.3 SVM

At the end of the cross validation, we picked the best hyper-parameters:

preprocessing	C	class_weights	gamma	kernel	mean_valid_f1
No scaler	0.01	balanced	auto	linear	0.859
Standard scaler	1	None	auto	rbf	0.920

We used these parameters to re-train the classifier on the whole training set and then computed the metrics for both training set and test set.

	F1	P	R
Train	0.860	0.834	0.914
Test	0.839	0.815	0.893

Table 18: Metrics (macro)

	F1	P	R
Train	0.93	0.955	0.909
Test	0.916	0.949	0.892

Table 19: Metrics (macro) (standardized)

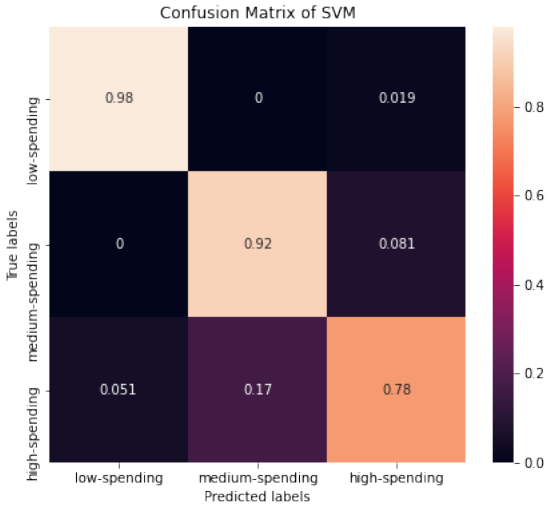

Table 20: Confusion Matrix

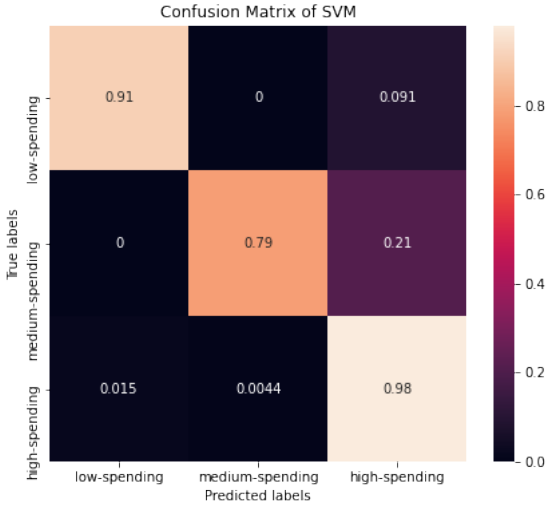

Table 21: Confusion Matrix (standardized)

4.4 Decision Tree

For the Decision Tree we leaved the hyper-parameters listed in Table 12. We run it on the non standardized dataset, obtaining the following results:

	F1	P	R
Train	1.0	1.0	1.0
Test	1.0	1.0	1.0

Table 22: Metrics (macro)

Considering the perfect results, we avoided to insert in this report the confusion matrix with its obvious content. A more interpretable way to look at the Decision Tree results is with the Figure 17:

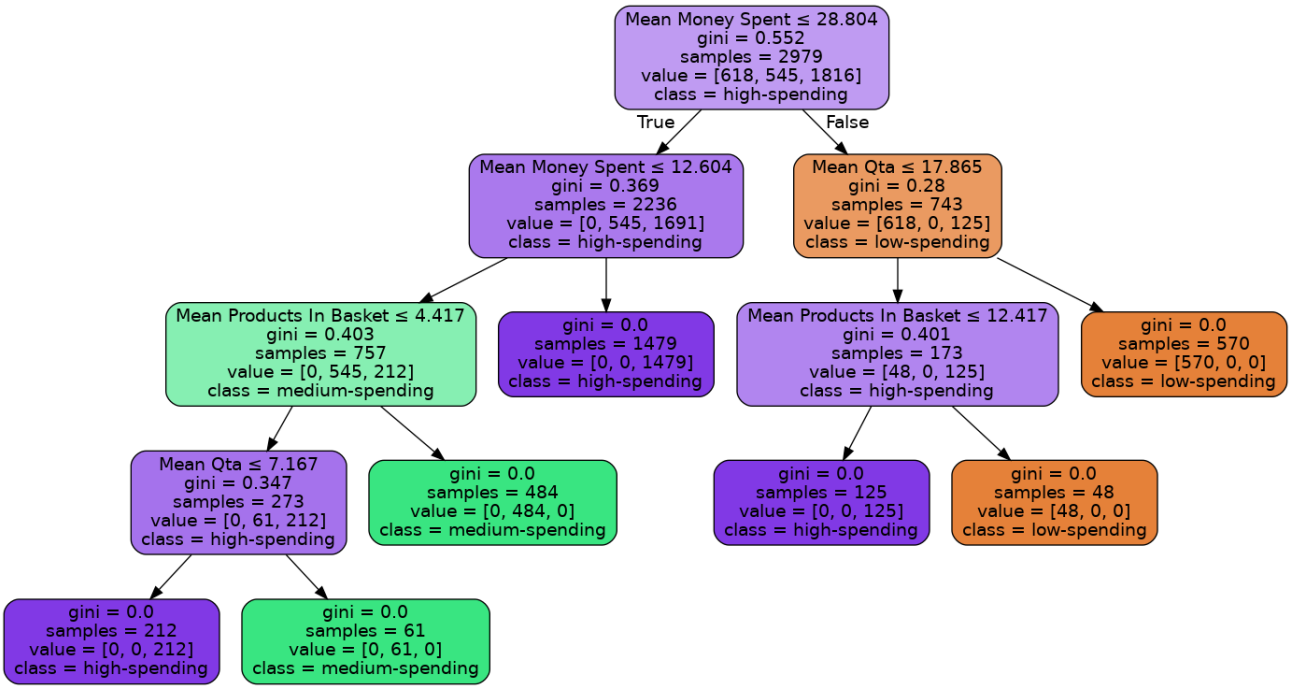

Figure 17: Decision Tree explained

4.5 Random Forest

For the Random Forest we picked the best hyper-parameters listed in Table 23. We run it on the non standardized dataset, obtaining the results listed in Table 24.

max_depth	min_samples_split	min_samples_leaf	bootstrap	n_estimators	criterion
10	2	2	False	200	'gini'

Table 23: Random Forest best hyper-parameters

	F1	P	R
Train	1.0	1.0	1.0
Test	1.0	1.0	1.0

Table 24: Metrics (macro)

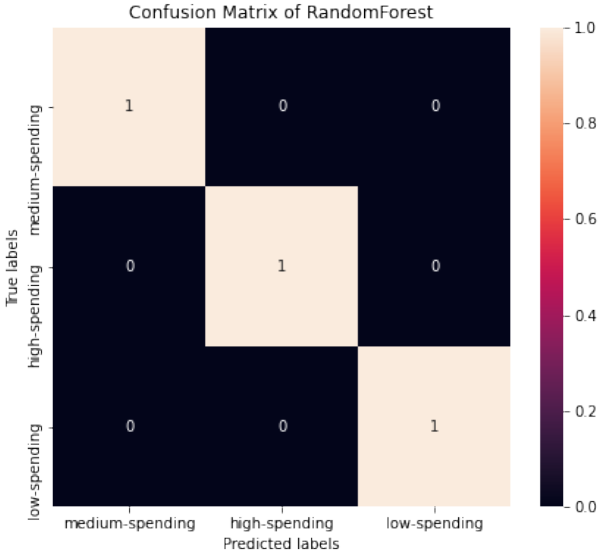

Figure 18: Random Forest confusion matrix

4.6 MultiLayer Perceptron

After the model selection phase, we picked the best hyper-parameters:

lr	lr_decay	epochs	batch_size	lambda	n_hidden	decay_steps	optimizer
0.0001	0.9	1000	25	1e-7	15	10000	'Adam'

Table 25: MLP best hyper-parameters for both the experiments

After the model assessment phase using the parameters in Table 25, we executed the predictions of the model on the test set. The results obtained are shown in the following tables:

	F1	P	R
Train	0.909	0.929	0.892
Test	0.913	0.933	0.897

Table 26: Metrics (macro)

	F1	P	R
Train	0.940	0.968	0.961
Test	0.967	0.967	0.967

Table 27: Metrics (macro) (standardized)

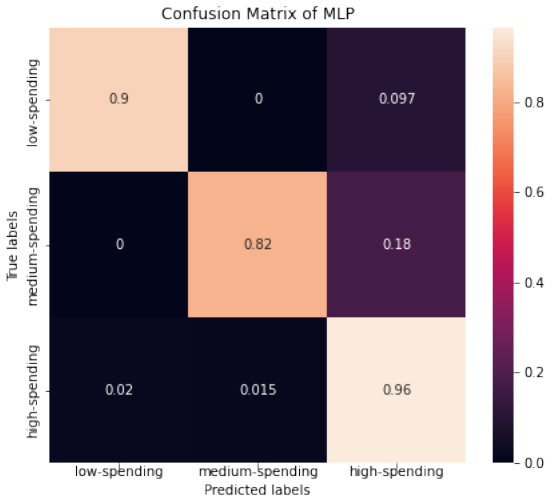

Table 28: Confusion Matrix

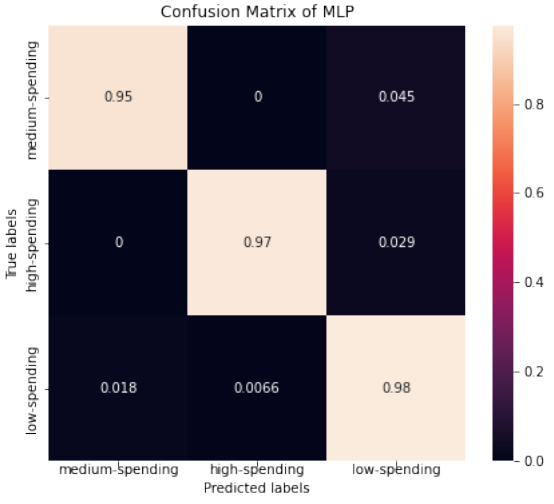

Table 29: Confusion Matrix (standardized)

4.7 Best Classification model

Guided by the metrics computed on the test set for estimating the generalization error and by the "Occam's Razor rule", we consider as the best classification model the Decision Tree, because of its simplicity and its perfect score. We can observe in Figure 17 how it can reproduce in a perfect way the label generation method we used, exploiting the quartiles of the features involved. Also the Random Forest, being an ensemble of Decision Tree, achieves perfect results,

but being more complex in terms of architecture, being an ensemble of Decision Tree. We chose the simple Decision Tree also for its explainability by default, without involving external tools. For the other models experimented, we noted that there are some misclassified examples, probably due to class imbalance. We exploited also the standardization as preprocessing, which let us achieve better results when involved. As future works to improve the classification, we could consider methods of oversampling / undersampling to try to re-balance the dataset, but the label generation rule up to now is quite simple and the use of a Decision Tree model fit perfectly. For more complex rules and definition of "spendent", more complex classifiers (and also set of features) should be tested: for example, from the results in table 27 we can see that the MLP model is quite flexible and is the second best model.

5 Sequential Pattern Mining

5.1 Generalized Sequential Pattern

To use this algorithm, we exploited the SPMF[1] library, thanks to the opportunity to also use it for timing constraints. We experimented also the suggested `gsp.py` script, but we noticed that for our dataset it was very slow (order of hours), while with the Java implementation we're able to complete the analysis in few minutes.
We run experiments with different value of support: 3.5%, 5% and 10%, noting that the higher the support, the less the sequences obtained. With 5% support we got `177` sequences and we reported some of them in Table 30.

Sequence	Support
CREAM HANGING HEART T-LIGHT HOLDER, CREAM HANGING HEART T-LIGHT HOLDE	416
REGENCY CAKESTAND 3 TIER, REGENCY CAKESTAND 3 TIER	363
⋮	
LUNCH BAG RED RETROSPOT, LUNCH BAG SPACEBOY DESIGN	201
LUNCH BAG RED RETROSPOT, JUMBO BAG RED RETROSPOT	201
⋮	
PARTY BUNTING, CREAM HANGING HEART T-LIGHT HOLDER	169
LUNCH BAG SUKI DESIGN, LUNCH BAG PINK POLKADOT	169
LUNCH BAG CARS BLUE, LUNCH BAG ALPHABET DESIGN	169

Table 30: Sequences ordered by support.

From this analysis we can note that the customers are habitual with respect to some objects, e.g.: from the first two rows of the Table 30, it's possible to see that the sequence is composed by the same type of object (light holder, regency cakestand), and going down to the third and forth rows it's possible to see that the second object in the sequence is related to the first (e.g.: two lunch bags in different versions).

[1] `http://www.philippe-fournier-viger.com/spmf/`

5.1.1 GSP with time constraints (optional)

For this experiment we chose to order first by the number of elements of the sequence and then by support, in order to show on top the sequences that have an higher number of products. With 5% support and time constraints (*min_gap*: `1`, *max_gap*: `7`, *maximum span*: `30`) we found `35` patterns. In Table 31 we show some examples obtained.

Sequence	Support
GREEN REGENCY TEACUP AND SAUCER, PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY TEACUP AND SAUCER	240
GREEN REGENCY TEACUP AND SAUCER, ROSES REGENCY TEACUP AND SAUCER	313
PAPER CHAIN KIT 50'S CHRISTMAS, PAPER CHAIN KIT VINTAGE CHRISTMAS	313
⋮	
GARDENERS KNEELING PAD CUP OF TEA, GARDENERS KNEELING PAD KEEP CALM	254
LUNCH BAG RED RETROSPOT, LUNCH BAG BLACK SKULL	253
⋮	
PACK OF 72 RETROSPOT CAKE CASES, 60 TEATIME FAIRY CAKE CASES	231
JAM MAKING SET WITH JARS, JAM MAKING SET PRINTED	231
⋮	
SET OF 3 CAKE TINS PANTRY DESIGN, SET OF 6 SPICE TINS PANTRY DESIGN	223
JUMBO BAG VINTAGE LEAF, JUMBO BAG VINTAGE DOILY	221

Table 31: Sequences with time constraint ordered by number of elements and support.

Time constraints reduced considerably the number of sequences retrieved and, also, the execution time of the two experiments varied substantially: from 1 minute and 39 seconds of the standard implementation to 1.67 seconds of the constrained one.

6 Conclusions

In summary:

- the data understanding and pre-processing phase has been very challenging, because of the lack of significant categorical values that didn't allow us to use exact values to make simple categorization of the customers; also, the presence of just two continuos significative values (`Sale` and `Qta`) was quite limiting.
- The clustering analysis was completed efficiently by almost all the algorithms; in particular, we exploited *DBSCAN* to detect outliers before the other clustering analysis.
- The classification task was quite straightforward: we were able to achieve high precision with almost all the models tried; we found out that Decision Tree is perfectly able to learn our simple rules used to generate the labels, and that the standardization is crucial to obtain better performance on most of the models.
- The sequential pattern mining task, finally, was really time consuming for the Python implementation, so we also tried a most efficient one in Java; time constraints turned out to be very efficient to compute sequences.

As future works, we could consider to involve a better usage of the `ProdDescr` field, exploiting NLP techniques to extract more info about the products, for example expanding their information by linking them to popular Knowledge Graphs (e.g.: `DBPedia`), and use products info to have better knowledge about customers behavior (e.g.: customers that buys more some categories instead of others), and then use this to have a better predictability of customers' behavior and their clustering.