UNIVERSITÀ DI PISA

Data Mining Project

Samuel Fabrizi
s.fabrizi1@studenti.unipi.it

Gerardo Zinno
g.zinno@studenti.unipi.it

Data Mining (309AA), 2020/2021

Contents

1 Introduction

The paper describes our work in the project of Data Mining course. The project requires data mining tools to analyse a dataset in the commerce field. Kotler and Armstrong [1] pointed out that attracting customers is an important task, but retaining customers is more important since losing a customer means losing the entire stream of purchase that the customer would make over a lifetime. For this reason, we based our analysis on quantitative, qualitative and temporal information. We are interested in both recent and old customers. The purpose of the company is probably to keep old customers and gain new ones into its business.
We start with an exploratory analysis of the dataset. In this phase, we want to understand the details of each attribute. This work is explained in section 2. In section 3, we clean the dataset according to the results obtained in the previous section. Then we produce a set of indicators useful for both clustering and predictive analysis. Section 4 and 5 show a comparison among several clustering and classification models respectively. Finally, section 6 contains a sequential pattern mining analysis on customer shopping sequences.

2 Data Understanding

In the data understanding phase our main goals are the analysis of the accuracy of the data, the visualization of the data in order to find outliers and the analysis of missing value. The handling of the data is postponed to the next phase, here we constrain ourselves to only explore the dataset.

We start by loading the dataset with `pandas`, remove the duplicate rows and look at the dataset infos and its head (tail).

	BasketID	BasketDate	Sale	CustomerID	CustomerCountry	ProdID	ProdDescr	Qta
Attribute Types	object	object	object	float64	object	object	object	int64
Missing Values	0	0	0	65073	0	0	753	0
Missing Values (%)	0	0	0	13.9439	0	0	0.161353	0

(a) info

	BasketID	BasketDate	Sale	CustomerID	CustomerCountry	ProdID	ProdDescr	Qta
0	536365	2010-12-01 08:26:00	2,55	17850.0	United Kingdom	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
1	536365	2010-12-01 08:26:00	3,39	17850.0	United Kingdom	71053	WHITE METAL LANTERN	6
2	536365	2010-12-01 08:26:00	2,75	17850.0	United Kingdom	84406B	CREAM CUPID HEARTS COAT HANGER	8
3	536365	2010-12-01 08:26:00	3,39	17850.0	United Kingdom	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6
4	536365	2010-12-01 08:26:00	3,39	17850.0	United Kingdom	84029E	RED WOOLLY HOTTIE WHITE HEART.	6

(b) head

Figure 1: First look at the dataset

From the informations in the images 1a and 1b we can see that the type of some columns is not recognised by `pandas`. As an example, we would expect the `Sale` to be an `int64` but it is inferred to be the generic type `object`. We can also already see that there are missing values. An in-depth analysis of each attribute is required and it is performed in the relative notebook. We list here only the most interesting discoveries.

2.1 BasketID

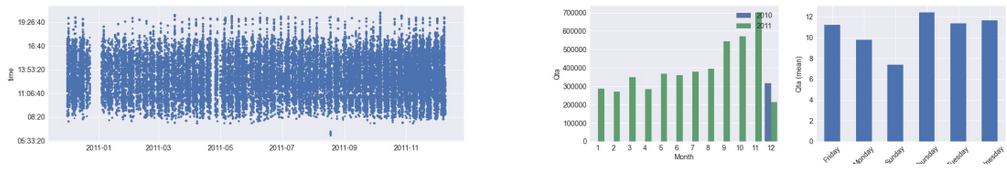
This attribute does not present missing values. All the values we can see from the `head` are numerical but the inferred type is `object` instead so we split the numerical values from the non numerical ones.
After seeing that the non numeric part is consisting of more than 9000 entries we have excluded the possibility of errors. After a more exhaustive analysis, we find out that in addition to purely numerical IDs there are two other kind of IDs.

- the ones starting with a `C`
- two ids starting witn an `A`

The identifiers starting with the letter `C` have a positive sale but a negative quantity. Our hypothesis is that the `C` stands for *Cancel* and indicates that something has been removed from the basket.
The two basket starting with the `A` are outliers as we can see from the *Adjust bad debt* in the description and from the fact that they have no customer field and a negative sale. It's probably an insertion done by an employee.

2.2 BasketDate

This attribute does noot present missing values and it has been casted to a `datetime64` format without errors. We can see that the period of observation goes from 1/12/2010 to the 9/12/2011 so it spans one year.



(a) Period of observation

(b) Days

Figure 2: Temporal distribution at first glance

In figure 2a we can see that the purchases are concentrated between 8:00 and 19:30, possibly the opening and closing hours of a physical shop since if it was an online seller we would have found orders outside the usual working hours. We can also see that there is a gap of multiple days where no orders were registered. We can only make hypotheses about it, like that the shop was closed or that they were unable to log the purchases.
In figure 2b we can see that there are no orders on Saturday, the closing day on the shop and the distribution of orders in the week and in the whole period of observation.

2.3 Sale

Initially the `Sale` attribute is not recognized as `float64`. This is due to the fact that the numbers are saved with a comma as a decimal separator instead of the dot. We inserted the dot and converted the values to numeric without errors.

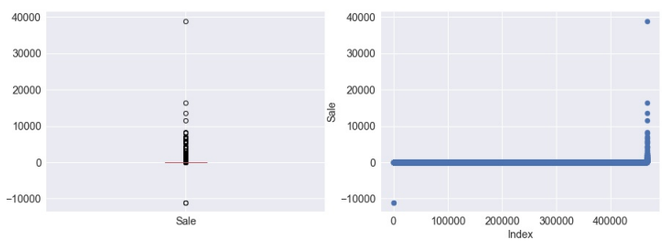


Figure 3: Sale distribution

From the boxplot in figure 3 we can see that the Sale distribution highlights the presence different outliers. We already know that the negative value(s) corresponds to the Adjust Bad Debt entries found in the section 2.1.
Analyzing the distribution of the values we see that there are entries with huge Sale value and a high standard deviation. We will handle those value in the data preparation phase.

2.4 CustomerID

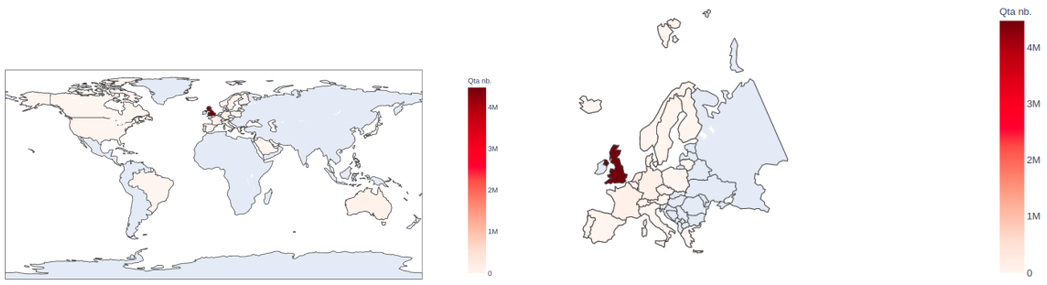
This attribute presents missing values, we substitute those missing values with the value 0 as a placeholder, already knowing that we'll drop the entries in the next phase considering that our final goal is the customer segmentation and it is fundamental to be able to distinguish between different customers.
There are 4372 unique `CustomerID`s. All the entries have an integer values but they are recognised as `float64` by `pandas` because of the way they are saved in the dataset.

2.5 CustomerCountry

From the figures 4a, 4b we see that the great majority of customers comes from the United Kingdom (83%). We also notice that the 0.0007% of entries has the value `Unspecified` in the CustomerCountry field.
Observing the plot that reports the World's distribution of quantity, it is possible to see that there are not purchases made by people from oriental and african countries.

2.6 ProdID

We led the analysis of the `ProdID` attribute in the same way as the `BasketID` one. We notice that there are several patterns. A visualization of the records shows that there are both identifiers consisting of only numbers and identifiers consisting of letters and numbers. Observing



(a) Distribution of customers around the world (b) Distribution of customers around Europe

Figure 4: Worldwide customers distribution

these values it seems that there isn't a specific correlation between product identifiers and descriptions (see Fig. 5a).

```
85123A          -> WHITE HANGING HEART T-LIGHT HOLDER
84406B          -> CREAM CUPID HEARTS COAT HANGER
84029G          -> KNITTED UNION FLAG HOT WATER BOTTLE
84029E          -> RED WOOLLY HOTTIE WHITE HEART.
82494L          -> WOODEN FRAME ANTIQUE WHITE
85099C          -> JUMBO  BAG BAROQUE BLACK WHITE
84997B          -> RED 3 PIECE RETROSPOT CUTLERY SET
84997C          -> BLUE 3 PIECE POLKADOT CUTLERY SET
84519A          -> TOMATO CHARLIE+LOLA COASTER SET
85183B          -> CHARLIE & LOLA WASTEPAPER BIN FLORA
```

(a) ProductIDs and Product Descriptions

```
POST            -> POSTAGE
D               -> Discount
C2              -> CARRIAGE
M               -> Manual
DOT             -> DOTCOM POSTAGE
BANK CHARGES    -> Bank Charges
S               -> SAMPLES
AMAZONFEE       -> AMAZON FEE
DCGS0076        -> SUNJAR LED NIGHT NIGHT LIGHT
gift_0001_40    -> Dotcomgiftshop Gift Voucher £40.00
```

(b) Special ProductIDs

Figure 5: Description of products

We find it is more interesting to analyse the description of special identifiers, and so we computed the mapping in figure 5b. There are some interesting entries. For example the identifier `D` is described as *Discount* and indeed there are negative quantities for those products with identifier `D`. The same goes for `S` identifiers.

2.7 ProdDescr

This attribute has 753 missing values that we replace with the `GENERIC PRODUCT` value.
We find out that the mapping between `ProdID` and `ProdDescr` would be a one to one mapping if it wasn't for some spelling errors, some missing values, or some employee annotation but the true description for a given `ProdID` can be easily inferred if needed.
As an example from the notebook, all the descriptions for a specific product are ‘‘`PACK 3 BOXES CHRISTMAS PANNETONE`’’, ‘‘`?`’’, ‘‘`check`’’, ‘‘`PACK 3 BOXES CHRISTMAS PANETTONE`’’, ‘‘`GENERIC PRODUCT`’’. It is clear that we are dealing with a pack of three boxes for *panettone* but due to errors and other annotations, five different descriptions are associated to the same identifier. This is the least interesting of the attributes for our purposes but it could be useful for other purposes like inferring how the color of a given object influences its sales through an NLP model. For a lack of time we decided to not proceed with this kind of analysis. From a visual analysis of the descriptions we find that we are dealing with a supermarket selling gift material.

2.8 Qta

As already noticed in a previous section, *Qta* contains many outliers.

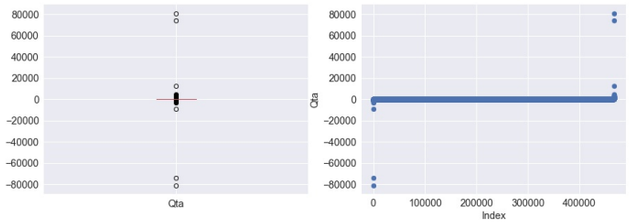
0.02% of the entries contain a negative quantity. As already said, there are some cases in which it makes sense, such as in discount entries. We have also to take into considerations that the mean has a value around 10. Indeed it's easy to identify these outliers in the plots in figure 6b, especially in the second one.

3 Data Preparation

This is the phase of the data preparation. The goal of this phase is to produce a customer profile that will later be fed to the clustering algorithms. In order to do so we have to select the attributes, treat the missing data and the outliers and improve the data quality starting from the knowledge we obtained from the data understanding phase.

count	466678.000000
mean	10.806528
std	232.627771
min	-80995.000000
25%	2.000000
50%	4.000000
75%	12.000000
max	80995.000000

(a) Qta description



(b) Qta boxplot

Figure 6: *Qta* analysis

3.1 Data cleaning

The attributes analysis explained in section 2 give us the tools to understand the meaning of the different attributes. Keeping in mind that that our aim is the analysis of customer's profile, we decide to remove all entries without a **CustomerID** since we can't discern the various customers behind that single id of value 0. After the deletion of the entries with missing **CustomerID** the size of the dataset goes from 466678 to 401605. The original dataset contains a lot of outliers. We carry out a uni-variate outlier analysis using simple visualization techniques. To remove some of the outliers we remove all the points that do not fall between the first and third quartile (Q_1 and Q_3). In order not to be too restrictive, we add a parametric tolerance and we discard all the elements e s.t.

$$e < \epsilon * Q_1 \vee e > \epsilon * Q_3$$

.

3.2 Feature creation

We add a new attribute, the **Revenue**, computed as the product of quantity and sale and it was useful for computing the monetary value indicator (see Sec. 3.3.1).

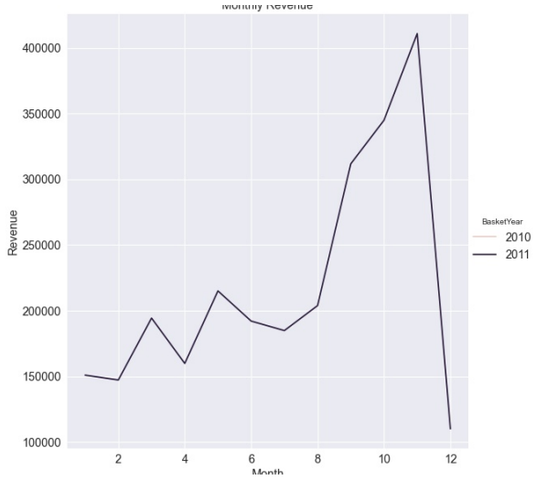


Figure 7: Monthly revenue

The plot in figure 7 shows monthly revenue. In December the revenue value plummets due to the fact that the observations end at the beginning of the month. It's also interesting to see that the month in which there are the maximum amount of revenue is November. This could be a really useful consideration to take into account to choose the best marketing strategy.

3.3 Our indicators

In order to build a customer profile we have to extract for each customer some useful indicators from the dataset.
We choose to compute the RFM indicators and the days-since-first-purchase indicators in order to have indicators that are able to capture a temporal aspect.

3.3.1 RFM indicators

RFM (Recency, Frequency and Monetary value) indicators are widely used in many practical areas of marketing and customer segmentation [2]. Adopting RFM model, decision makers can effectively identify valuable customers and then develop effective marketing strategy.

Recency is an indication about how recently customers made a purchase. In figure 8 we reported the distribution of the recency. It is possible to see that most of the customers have small values of recency. They are probably frequent customers.

Figure 8: Recency distribution

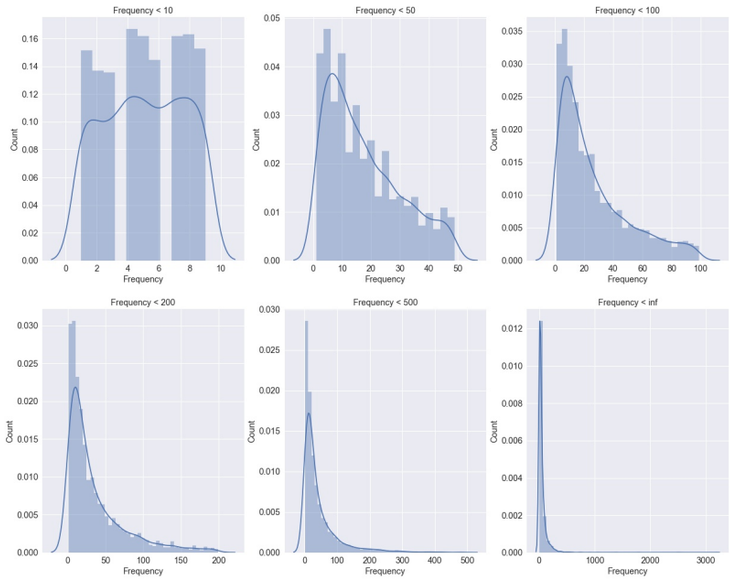


Figure 9: Frequency distribution

Frequency is the measure of how often the customer makes a purchase. After noticing that there are a lot of one-time buyers with a frequency value of 1, or casual buyers, we decided to look at the distribution of this indicator by fixing various thresholds. The result can be seen in figure 9. As the threshold increases, the plot squeezes towards the value 0. This behaviour underlines that there are only few customers with an high frequency value.

Monetary value is a measure of how much money a single customer has spent during the whole period of observation. The same considerations of the Frequency apply here. Since there are a lot of customer that spends only a few money, we analyse the distribution of the M.V. by fixing various thresholds. The result can be seen in figure 10. It is interesting to see that up to a threshold of 100 the number of customers with that monetary value increases. But when we increase this threshold up to 1000, there is a dramatic decrease in the amount of customers.

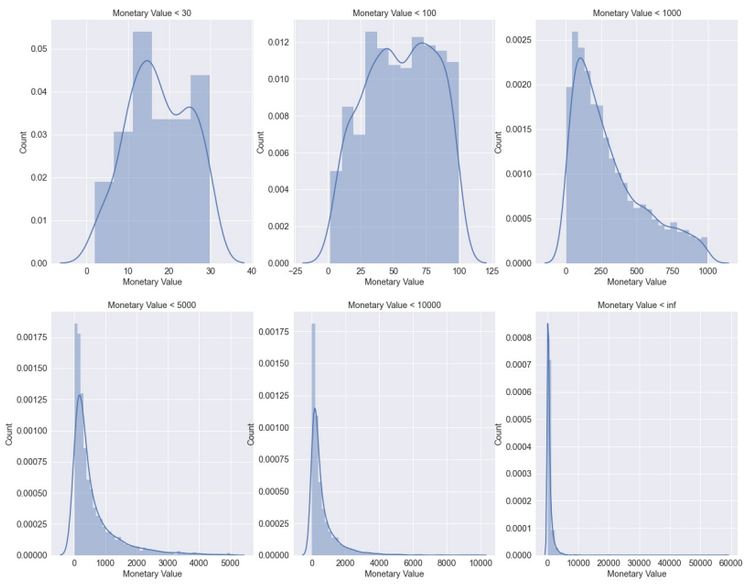


Figure 10: Monetary Value distribution

3.3.2 Days from first purchase

This indicator measures how many days it's been since the customer made the first purchase. Distribution plot in figure 11 shows that most of the customers has a *FirstPurchase* value greater then 350. Considering that data covers one year, it probably means that most of them are loyal customers.

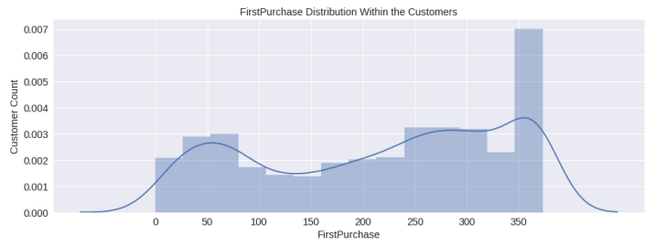


Figure 11: Days from first purchase distribution

3.4 Other indicators

In the following there is a list of the mandatory indicators required by the task that we computed and some additional ones.
There is not much to say about the computation of these indicators, they were all computed by simple manipulations of the dataframe.

- **I** Total number of items purchased by a customer during the period of observation;
- **Iu** Number of distinct items purchased by a customer during the period of observation;
- **Imax** Maximum number of items purchased by a customer during a shopping session;
- **Imin** Minimum number of items purchased by a customer during a shopping session;
- **E** Shannon entropy on the purchasing behavior of the customer, computed on the products the customer bought during the period of observation;

After all the indicators are computed, we merge them into a new dataframe representing the customer profile. This dataset contains quantitative, qualitative and temporal information about the customer behaviour. For this reason we believe that those indicators can be really useful for both descriptive and predictive analysis.

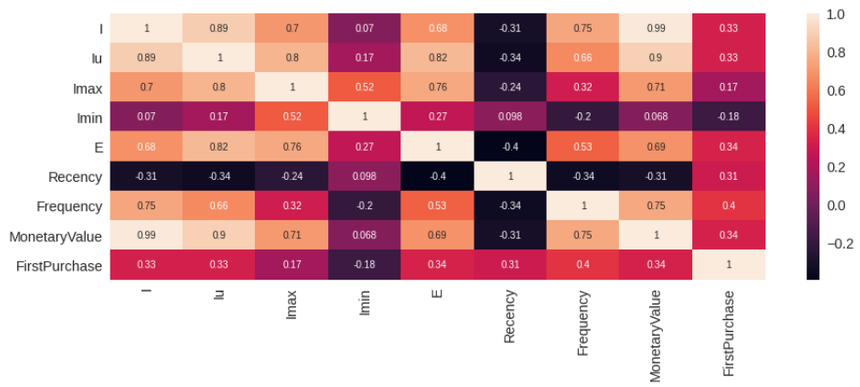


Figure 12: Correlation matrix of indicators

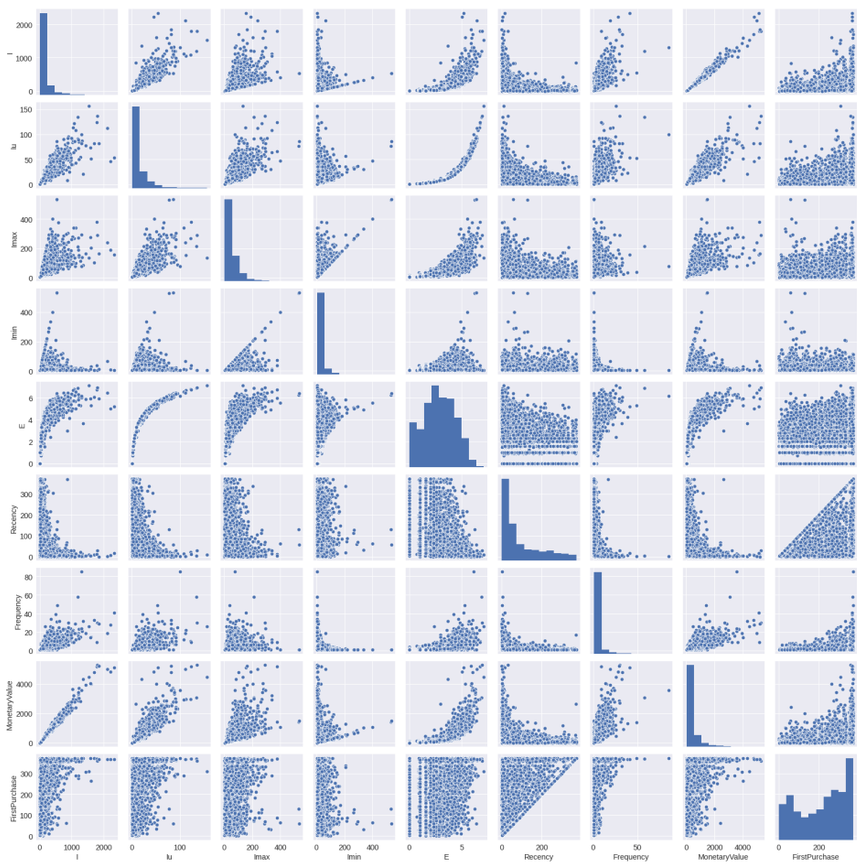


Figure 13: Pairwise relationships between the indicators

It is interesting to see how the computed indicators are correlated, in the figure 12 we can see through an heatmap the correlation matrix computed on the customer profile dataframe. Black-ish colors indicate a negative correlation while pink-ish ones a positive correlation.
From the figure 13 the pairwise relationships between the indicators can be seen. The diagonal of the matrix shows the marginal distribution of the data in each column.

4 Clustering analysis

In this section, we apply different clustering algorithms to the customer's profile. First of all, we extract only the numerical attributes. Because of the curse of dimensionality problem, we decide to delete the correlated attributes. At the end, we have the following attributes: `Imax`, `Imin`, `E`, `Recency`, `Frequency`, `MonetaryValue`, `FirstPurchase` (see Fig. 14).
In the following sections, we describe the method used to approach the clustering. In section 4.1, we describe how we use the PCA [3] to improve the algorithms performances. Then we proceed with the clustering methods definition. Finally, in section 4.6 we choose our "best" clustering algorithm and validate our choice.

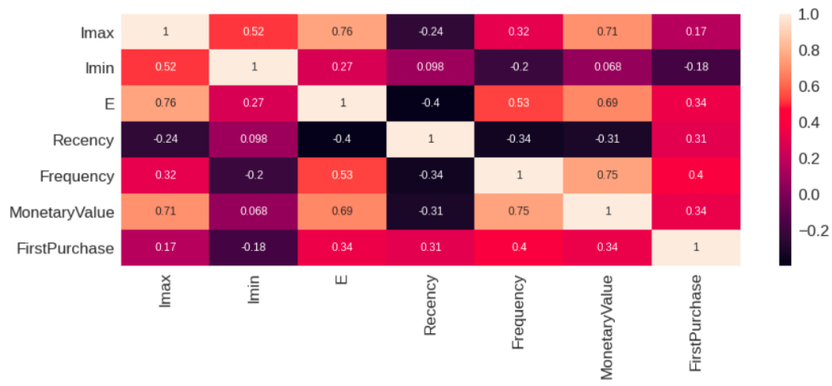


Figure 14: Correlation matrix of indicators used for clustering

4.1 Principal Component Analysis

According to some theoretical studies [4, 5], unsupervised dimension reduction is closely related to unsupervised learning. Furthermore, it can be shown that the global solution to clustering algorithms lies in the PCA subspace. Based on this insight, we decide to perform clustering in the PCA subspace. Figure 15 shows that the first feature explains roughly 50% of the variance within our data set while the first two explain 85% and so on. We chose to use three principal components. In this way, we capture more than 95% of the explained variance.
We create a simple pipeline in which there is the PCA in the first step and the *MinMaxScaler* in the second one. Then we pass the dataset to this pipeline. Now, we are ready to proceed with the clustering algorithms.

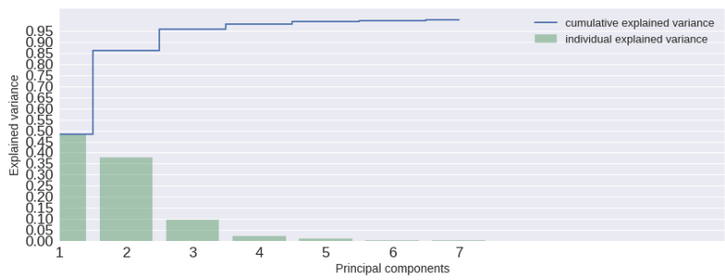


Figure 15: Principal components explained variance

4.2 Density-based Clustering

With a simple 3D plot of the PCA results, it is evident that there are some outliers. In DBSCAN, the clustering approach is different from typical clustering approaches. DBSCAN can define outlier (anomalous) points that do not fit any clusters. Furthermore in our dataset, the shape of the scatter plot is not suitable for density-based algorithms. For these reasons, we decide to use the DBSCAN only as outliers detectors. In subsection 4.6 we validate this approach.
The inputs of the algorithm are dataset and user-defined *eps* and $MinPts$ parameter values. We fix the $MinPts$ as $2 * \#attributes$ as suggested in [6]. To evaluate the effect of the parameters on discovering anomalies, we conducted experiments varying the *eps* within the range $[0.14, 0.008]$. As can be seen in table 1, as the *eps* parameter increases, the number of anomalies detected by the algorithm decreases. But if we take a value less or equal than 0.008, we obtain many small clusters. Furthermore, most of the points are indicated as outliers. According to these considerations we use the following parameters configuration: $eps = 0.1, MinPts = 12$.
DBSCAN clustering result is reported in figure 16. It shows the scatter plot in the first two principal components subspace. As expected, the DBSCAN identify the first horizontal line as a separated class. The black points are the detected outliers (noise points). A density-based clustering is not the best option for clustering analysis. Therefore we use it as outliers detector and delete the outliers found.

eps	# clusters	# outliers
0.13	1	15
0.11	1	38
0.1	2	50
0.008	29	3508

Table 1: DBSCAN results with different *eps* values

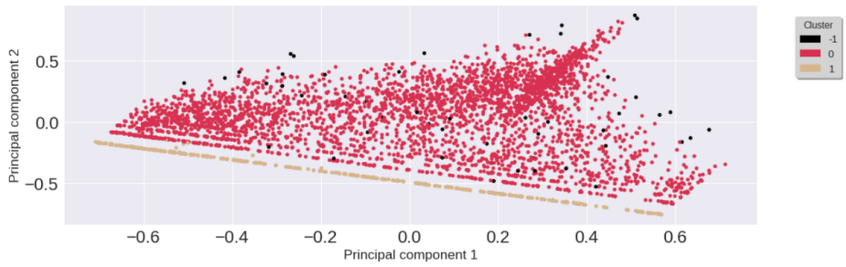
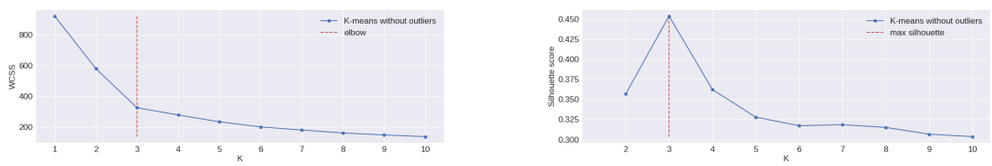


Figure 16: DBSCAN results

4.3 K-Means

In this section, we describe the K-Means clustering results. We apply the K-means algorithm in the first three principal components as explained in Sec. 4.1.
In order to pick the best k value, we use two different approaches: elbow and silhouette-based methods. We report the results in figure 17. We use the *kneed* library to programmatically compute the elbow point. In the silhouette-based method, we take into consideration the k value with the maximum silhouette score. Under both approaches, we obtain the same results. Therefore we decide to use a 3-clusters solution.



(a) Elbow method

(b) Silhouette-based method

Figure 17: Elbow and silhouette method to identify best k value

Considering that we have only three principal components we plot the K-means results in all possible combinations of these dimensions (see Fig. 18). As you can see in figure 19, the clusters are well-separated. In section 4.6 we continue to discuss the K-means results and we validate our approach from a numerical point of view too.

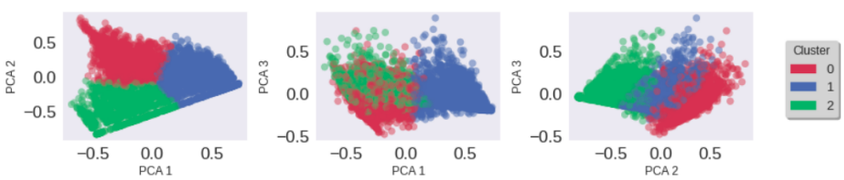


Figure 18: K-means results in 2D

4.4 Hierarchical Clustering

In this section, we describe the approach used to identify cluster with hierarchical algorithm. We use different linkage parameters:

- **Single** link technique is good at handling non-elliptical shapes, but is sensitive to noise and outliers;
- **Complete** linkage is less susceptible to noise and outliers, but it can break large clusters and it favours globular shapes;

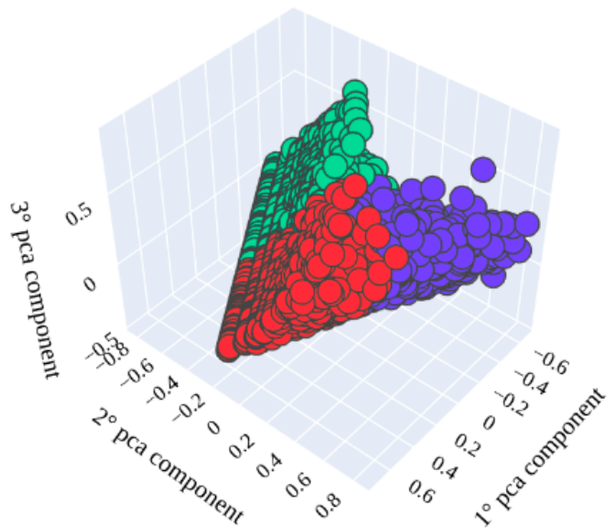


Figure 19: K-means results in 3D

- **Average** linkage is an intermediate approach between the single and complete link approaches;
- **Ward** hierarchical clustering uses the optimal value of the eucledian distance function to choose the pair of clusters to merge at each step.

The length of the two legs of the U-link represents the distance between the child clusters. This entails that more the distance of the vertical lines in the dendrogram, more the distance between those clusters.

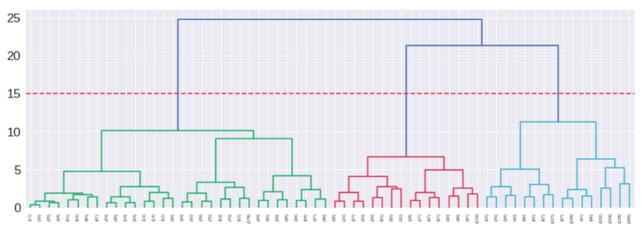


Figure 20: Best number of clusters in Ward hierarchical clustering dendogram

We choose the ward linkage as best hierarchical clustering. Observing the dendogram of Ward linkage, you can see that it has the longest vertical line (see the notebook to visualize the dendograms of other linkage approaches). This entails that it contains a subdivisions of points with the greatest cluster cohesion. We set a threshold distance in order to choose the best number of clusters (see Fig. 20). We try to set the threshold in such a way that it cuts the tallest vertical line. The number of vertical lines this newly created horizontal line passes is equal to the optimal number of clusters; in our case it is three. This approach can be seen as a first confirmation of the results obtained with K-Means clustering.
We re-apply the hierarchical algorithm with Ward linkage and plot the first two principal component (Fig. 21).

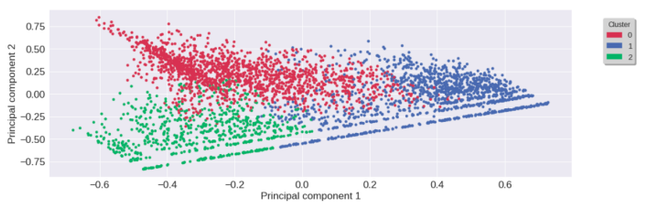


Figure 21: Ward hierarchical clustering results in two dimensions

4.5 X-Means

The last clustering algorithm used is X-Means (from the *pyclustering* library). It tries to automatically determine the number of clusters based on BIC scores. We set the initial centers to 2 and the maximum one to 20. X-means dynamically increases them using the BIC criterion to control the process of splitting clusters. We use the K-Means++ algorithm to choose the initial centers for algorithms. Considering that this algorithm is based on a random sample, we proceed with different runs. In figure 22 we report the number of clusters found in 100 executions of the algorithm. Most of the time, X-means identifies a 3-clusters solution. This behaviour confirms the results obtained in Sec. 4.3 with K-means.

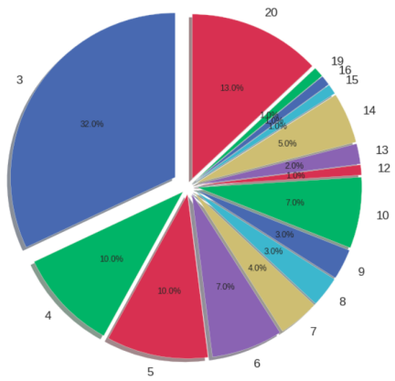
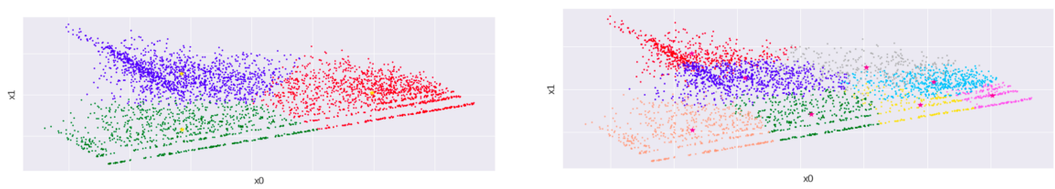


Figure 22: Occurrence probability of the number of clusters in X-means



(a) 3-clusters X-means solution

(b) 8-clusters X-means solution

Figure 23: Different executions of X-means algorithm

For the sake of clarity, we show two different X-means results. The plot in figure 23a shows the 3-clusters solution while the plot in figure 23b shows the 8-clusters one. When X-means identifies a high number of clusters, it only unpack the 3 based clusters found in most of the runs.

4.6 Cluster validation and interpretation of clusters

We choose K-means as the best clustering approach for this task. In this section, we motivate and validate our choice. Then, we explore the clusters to understand if there is a characterization in a real-world scenario.
For sure, the ideal number of clusters is three. K-means, X-means and Hierarchical approaches share this result. Observing the 3D visualization of the K-means results, we have a confirmation of that. Indeed, the three clusters seem to be well-separated. But we are also interested in a strong validation approach. So we decide to use a silhouette-based method for the validation phase.
As already explained, we choose the best K value taking into consideration the maximum silhouette score. This result does not entail that the clustering results are valid. For example, we have to see if there are some mistakes in clusters assignment.
Silhouette offers the advantage that it does not depend on the clustering algorithm used. Moreover, several authors [7, 8] consider a silhouette-based interpretation of clusters analysis extremely useful. For each object i we define the silhouette score $s(i)$. Essentially, $s(i)$ measures how well object i has been classified. Negative values indicate that those samples might have been assigned to the wrong cluster. Furthermore, the thickness of the silhouette plot underlines the clusters distribution. We also define the *overall average silhouette width* for the entire dataset. It is the average of the $s(i)$ for all objects i in the whole dataset.
For the sake of clarity, we report a comparison between two different clusterings (see Fig. 24). The left plot shows the silhouette score for the application of K-Means over the original cus-

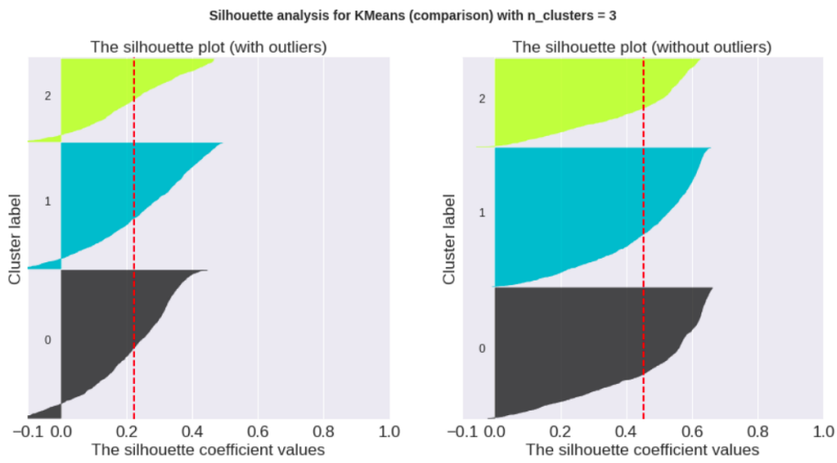


Figure 24: Comparison of silhouette score between k-means with and without outliers

tomer indicators (without DBSCAN outliers removal). On the contrary, the right one shows the chosen clustering approach. As you can see, the elimination of outliers entails a clear change in the thickness of the silhouettes. Moreover, the first plot shows negative silhouette scores for all clusters. It is also interesting to note that all clusters silhouettes are over the *overall average silhouette width*. These considerations validate our choice of the best clustering.

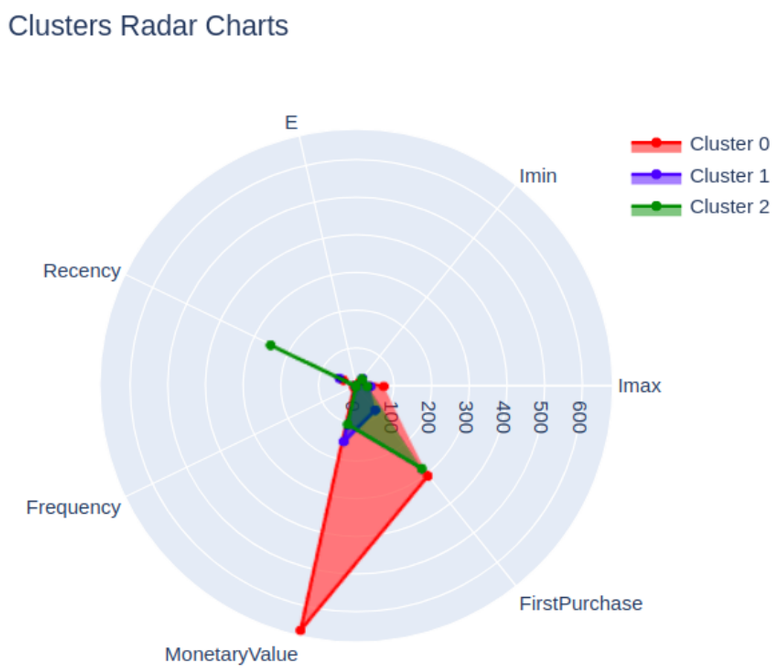


Figure 25: Visualization of clusters profiles

According to the radar charts, there are well-defined profiles for each cluster (see Fig. 25). Cluster 0 (red) represents the most profitable customers. They have a high values in `MonetaryValue` and `FirstPurchase` indicators. Furthermore, the low `Recency` underlines that they are loyal customers. Cluster 2 (green) have a high `Recency` and `FirstPurchase` values. These customers purchased for a short time period. One of the company purposes could be to understand the reason for this behaviour. Cluster 1 (blue) represents recent customers, as you can see from the `FirstPurchase` indicator. The company is interested in the fidelity of these customers. Mean statistics are shown in table 2. This information is useful considering that the scale of the radar chart does not permit to understand the others indicators distribution (such as `Frequency` and `Entropy`).

Cluster	Frequency	E	Imin	Imax
0	6.25	4.02	17.47	72.83
1	1.71	2.20	26.71	37.88
2	1.43	1.81	24.78	29.78

Table 2: Statistics of clusters

5 Predictive analysis

In this section, we describe the method used to address the predictive analysis task. We want to predict the spending behaviour of the customers. For this purpose, we create three different classes: `high`, `med` and `low`.
We label the customers by relying on the RFM score because it captures both quantitative and temporal information. For each customer, we assign a score (from 1 to 4) at each RFM attribute (`Recency`, `Frequency` and `MonetaryValue`) based on a quartiles-based approach[1] (see Tab. 3). We compute the RFM score taking the arithmetic mean of these three scores. Finally, we compute the labels using the $qcut$[2] function with $q = 3$.

Recency	Frequency	MonetaryValue	R_Quartile	F_Quartile	M_Quartile
18	1	345.92	4	1	3

Table 3: Example of RFM quartile score

Figure 26 shows the features of each labelled customer. They meet our expectations. Then, we analyse the `FirstPurchase` indicator to understand if recent customers may be classified as high-spender. Indeed, the company could be interested in both keep old customers and bring new ones. Although the indicators are computed on an annual basis, temporal information can be used to discriminate potential new customers. Results show how recent customer with low R-values and high F-values are classified as high-spender (independently from M-score).
After removing the indicators adopted for the labelling, we defines the attributes to use for the prediction. We choose the following attributes: `I`, `Iu`, `Imax`, `Imin`, `E` and `FirstPurchase` (see Sec. 3 for a detailed description). These indicators capture both quantitative, qualitative and temporal information. Therefore, they permit to satisfy company requirements about customer classification.
In the following sections, we describe the models used to approach the classification problem. We start with the description of the model selection approach (Sec. 5.1). In Section 5.2, we describe the strengths and weaknesses observed in each model. Finally, in Section 5.3 we compare both training and validation results to choose the best model for the prediction task.

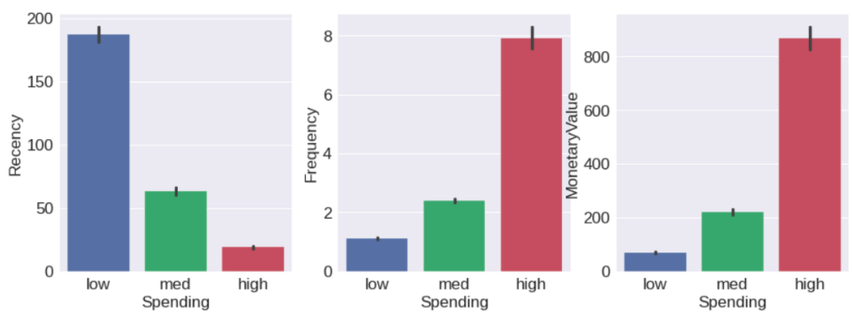


Figure 26: Labels distribution w.r.t RFM indicators

5.1 Model Selection

First of all, we split the whole dataset into train and test subsets with 70% and 30% sizes respectively. Data is split in a stratified fashion because labels are not entirely balanced (see Fig. 27. In this way we have the same labels distribution on both training and test. Furthermore, stratified cross-validation seems to be a less biased estimation model [9]. Then, we continue with the definition of the models and we start the model selection using only the train set.

[1] Example of quartiles-based split in a real-world scenario `https://bit.ly/3o5F2zF`
[2] Quantile-based discretization function

We decide to use a stratified K-fold technique for the model selection phase. We set the K parameter to be 5 because it seems to be the most reasonable value considering the number of examples per class. We use the following approach for each model:

- define a dictionary with the range values of the most interesting hyperparameters that we want to explore for that model;
- run the stratified 5-fold algorithm using the *f1_ weighted* score as evaluation metric;
- compute useful statistics on both training and validation results;
- choose the best set of hyperparameters for that model.

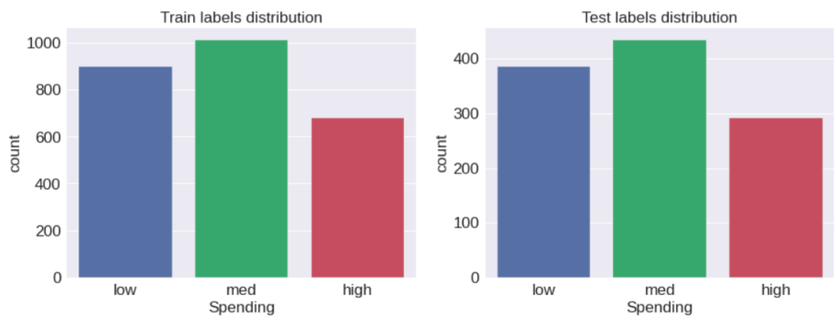


Figure 27: Labels distribution in train and test sets

5.2 Results

In the following, there is a brief description of the models used. If you want to see the hyperparameters range of each model, you can see the related notebook. For simplicity, we report only the results of the models with the best parameters configuration.

- **Gaussian Naive Bayes**
  This model is not really interesting because its approach is too much trivial. It obtains bad performances in this prediction task. This is probably due to the independence assumption among features. This assumption is not valid in this type of tasks. Indeed, a good predictive customer-spending model has to take into consideration the dependencies among variables;
- **K-Nearest Neighbors**
  One of the most important KNN hyperparameters is the number of neighbors K. Random search algorithm shows that the best K value is ≈ 10. As we decrease the value of K, our predictions become less stable. Inversely, as we increase the value of K, our predictions become more stable due to majority voting, but there is an increasing number of errors.
- **Support Vector Machine**
  Figure 28 shows the roc curve of the SVM model. ROC curve reports true positive rate on the Y axis and false positive rate on the X axis. This means that the top left corner of the plot is the "ideal" point - a false positive rate of zero, and a true positive rate of one. As you can see SVM performs very well in this task. It is able to capture more than 90% of the whole area in all three output classes.
- **Decision Tree**, **Random Forest** and **AdaBoost**
  Figure 29 show the the importance of features. The bars are the impurity-based feature importances of the model. Black lines represent inter-estimators variability (see figures 29b, 29c). Feature importances of all three models reflect the key attributes: `I` and `FirstPurchase`. They represent what we are interesting in, namely quantitative and temporal information.

We apply a standardization preprocess only in two models. Tree-based algorithms do not require inputs to be normalized, since they are invariant to monotonic transformations. On the contrary, we apply data standardization for **KNN** (it is based on the notion of distance) and **SVM** (the optimal hyperplane is influenced by the scale of the input features).

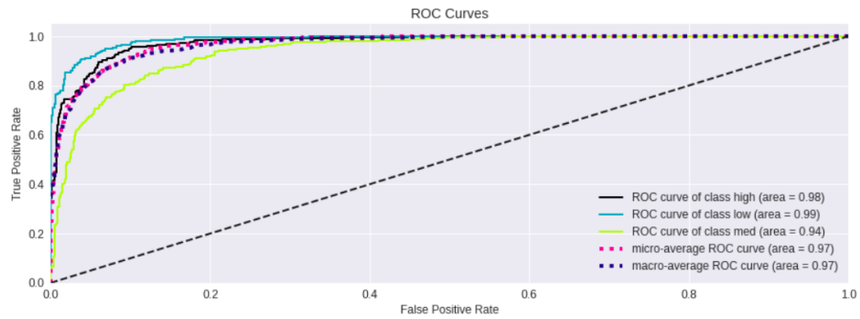
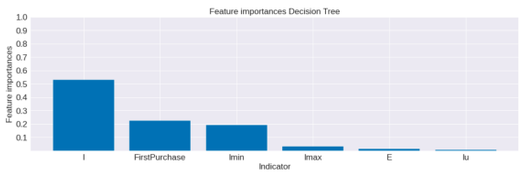
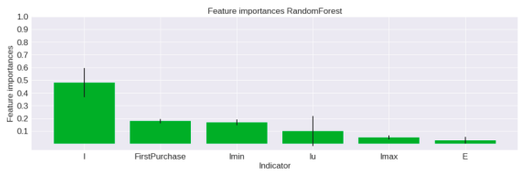


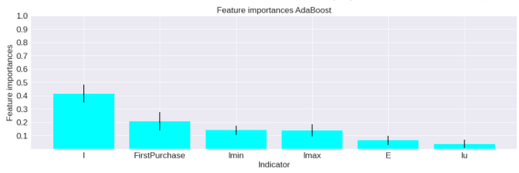
Figure 28: ROC curve with best SVM model



(a) Feature importances DecisionTree



(b) Feature importances RandomForest



(c) Feature importances AdaBoost

Figure 29: Feature importances

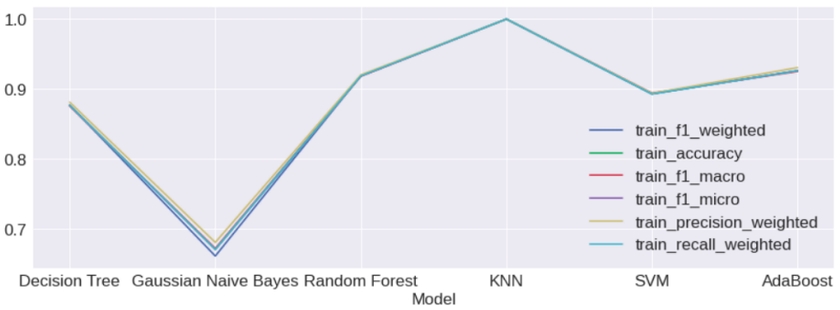


Figure 30: Training results

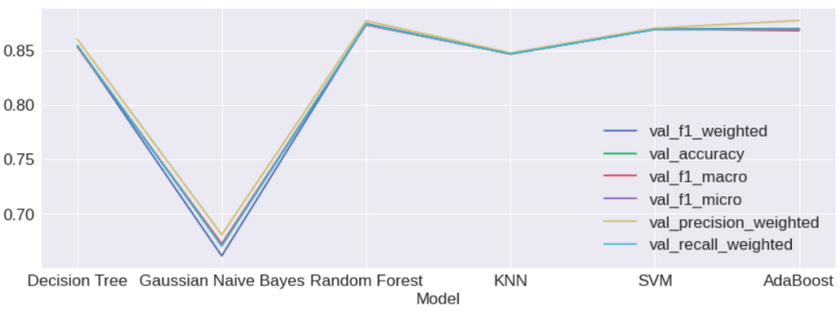
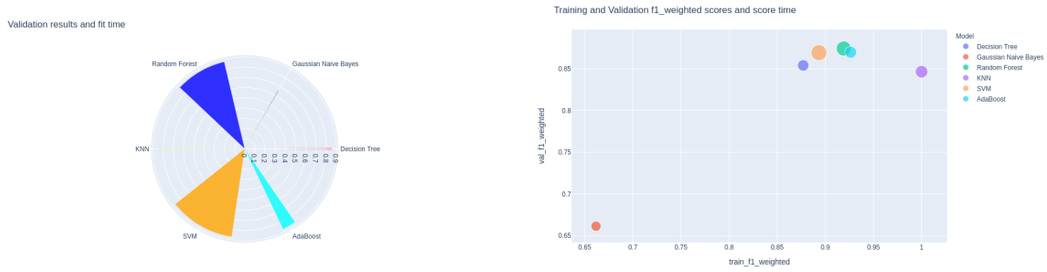


Figure 31: Validation results

5.3 Models comparison

We choose **AdaBoost**, **SVM** and **RandomForest** as best models for the prediction tasks. We decide to discard **Gaussian Naive Bayes**, **DecisionTree** and **KNN** according to *f1_ weighted* score and variance between training and validation scores. Although **KNN** has good performances in classes prediction and fit time, it has a large gap between training ($\simeq 100\%$) and validation ($\simeq 82\%$). Figures 32b and 33 show this observation. We report in figure 32 both functional and non-functional scores of each model. In the left plot the bars width represent the fit time. In the right one, the dimension of the circles is proportional to



(a) Validation metric results and fit time

(b) Validation score and score time

Figure 32: *f1_weighted* and time performance

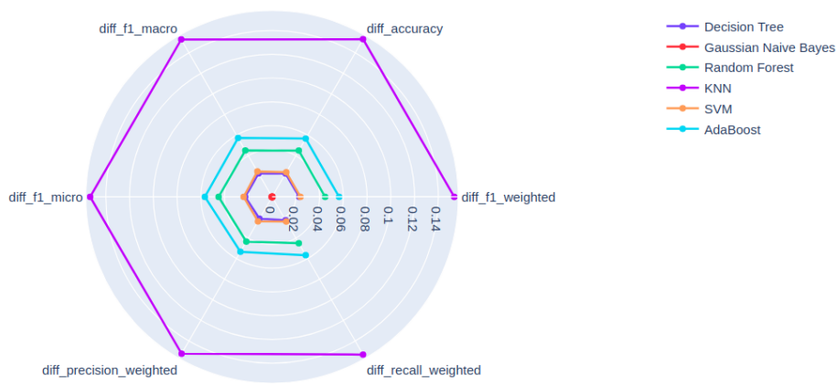
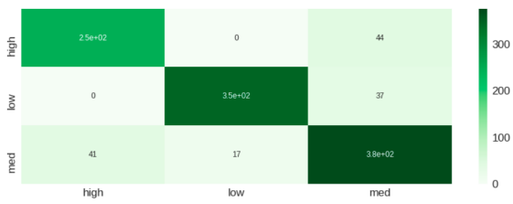


Figure 33: Variance between training and validation results

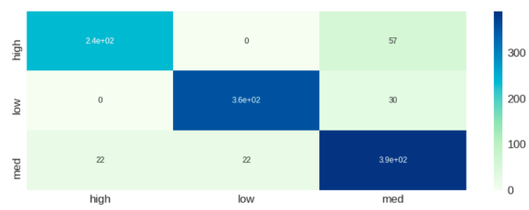
the score time. As expected, models with the best performances are those who require more time to fit.

As you can see in figure 31, **RandomForest**, **AdaBoost** and **SVM** obtain the best performances among other models in all metrics. We use only the validation performances to choose the best models. Test set is used in model assessment phase to understand the predictive ability over examples never seen before. Figure 34 shows the confusion matrices produced by the best models in these examples. They reach a score grater than 86% in both *f1* and *accuracy* metrics.

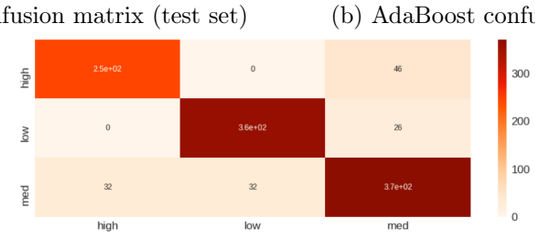
Finally, we carry out an error analysis to understand the hardest customers to predict. Test set is composed of 1111 examples. Table 4 shows the percentage of incorrectly classified test instances. All three models share 78 test examples in which they predict the wrong class. Apparently, there are no strong connections among those.



(a) Random Forest confusion matrix (test set)



(b) AdaBoost confusion matrix (test set)



(c) SVM confusion matrix (test set)

Figure 34: Confusion matrices of the best models

Model	Misclassification rate
Random Forest	0.112
AdaBoost	0.118
SVM	0.122

Table 4: Misclassification rate on test set

6 Sequential Pattern Mining

6.1 Sequential Pattern Mining without time constraints

We test several algorithms for performing the SPM. We find that with a high value 25% for the *minsup* parameter (minimum support threshold computed on the percentage of customers), we only find few sequential patterns. Each of those consists of only a single itemset with one item in the great majority of cases (singleton sets). There are few isolated exceptions in which this itemset contained two items. With a `minsup` strictly greater then 25% , no sequential patterns were found. Instead, with a `minsup` of 25%, all the algorithms find the same two singletons.
We find these patterns not to be interesting since a single frequently bought item can be computed in other simpler ways. So, we lower the *minsup* threshold to a value between 6% and 10% since at this level less frequent but more interesting patterns come out. We try the following algorithms:

- **SPADE** the popular "**S**equential **PA**ttern **D**iscovery using **E**quivalence classes" algorithm;
- **SPADE Parallel** a parallelized version of SPADE;
- **CM-SPADE** an algorithm based on SPADE with few optimizations that should make it faster.[3];
- **GSP** the classical Generalized Sequential Patterns algorithm;
- **CLOFAST** CloFAST is a fast pattern-growth algorithm for discovering closed[4] sequential patterns in sequence databases;
- **Bide+** an algorithm for discovering closed[4] sequential patterns in sequence databases.

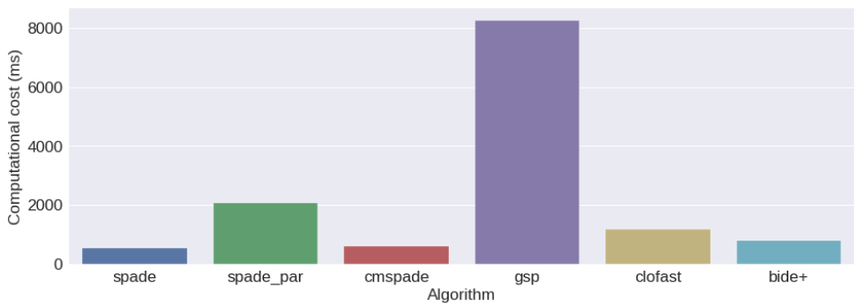


Figure 35: Comparison of computational costs among several sequence pattern mining algorithms

As you can see from the plots in figures 35 and 36, the worst algorithm in terms of time and memory performances is **GSP**. Fixing a support value threshold, all those algorithms obtain the same value of *frequent sequences count*. Therefore, the "best" sequential pattern mining algorithm for this task is **SPADE** and its variation (CM-SPADE). It seems a bit curious that the parallelized version of **SPADE** performs worse then the original version.

6.1.1 Interesting Patterns

The singleton found by the algorithms with a minsup of 25% has a support of 390 (26.72% of the customers). It is a *candleholder*.

[3]https://www.philippe-fournier-viger.com/spmf/CM-SPADE.php

[4]A frequent closed sequential pattern is a frequent sequential pattern such that it is not included in another sequential pattern having exactly the same support

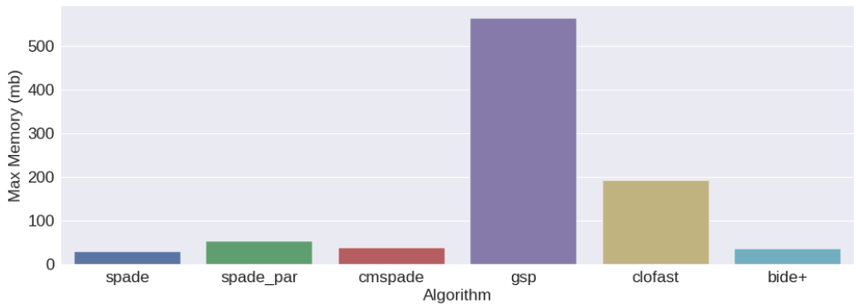


Figure 36: Comparison of memory usage among several sequence pattern mining algorithms

Once we replace the `ProdID` with the description we find that the non-singleton sequences found by the algorithms are almost entirely made by the same elements but with different colors or different patterns printed on them (e.g. *<"JUMBO BAG RED RETROSPOT", "JUMBO BAG VINTAGE LEAF">*). Therefore, we decide to try to find patterns among customers belonging to the same spending category to understand if there are different results. We take the customer profile defined in section 3 and we annotate it as explained in section 5 to split the customers into three spending categories. Then, we apply the **CM-SPADE** algorithm with several *minsup* values to see if interesting patterns emerged but we do not obtain better results. This result seems to be reasonable because of the kinds of product sold by the company. Indeed as you can see in figure 37, most of the customers have a shopping session with less then 3 items. These considerations validate our hypothesis about the nature of the shop (see Sec. 2.7).

6.2 Sequential Pattern Mining with time constraints

A time-extended sequence database is a set of time-extended sequences. In order to create this database we first compute a basket timeseries in which for each customer we save for each month all the shopping sessions he or she did in that month like lists of `ProdID`. Then, we use the month of the list of baskets to annotate the baskets with temporal information. We numerate the months from 0 to 12 to not have two columns with the same identifier. After having computed the database, we save it on a file in the format required by the library we are using and pass it to the following algorithms:

- **HirateYamana** is an algorithm for discovering frequent sequential patterns respecting some time-constraints to filter uninteresting patterns.
- **Fournier** combines features from several other sequential pattern mining algorithms. It can be used to discover closed[4] sequential patterns with time-constraints.

They take as input time-extended sequence database and in addition to `minsup`, the following constraints [10]:

- minimum/maximum time interval allowed between two succesive itemsets of a sequential pattern.
- minimum/maximum time interval allowed between the first itemset and the last itemset of a sequential pattern

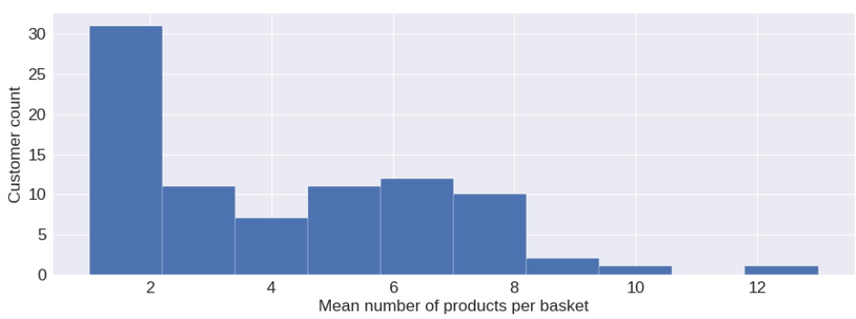


Figure 37: Mean number of items purchased by a customer per shopping session

6.2.1 Interesting Patterns

Fixing the `minsup` at 0.6 like for the non constrained algorithms, we tried all the combination for the constraints of value from an interval that goes from 0 to 15.
In our case these algorithms have not found sequences with more than one itemset. The more frequent pattern found is the *candleholder* also found by the unconstrained algorithms described in section 6.1.1. The algorithms assign a relative timestamp to the first pattern and then the number of timeunits of distance to the following itemsets in the frequent sequence. All the frequent patterns found have timestamp 0. This result seems interesting from the company point of view. Indeed, it probably means that customers buy specific products in the same months. Based on these considerations, the store could change its marketing strategy to offer different variants of these products.

References

[1] Philip Kotler and Gary Armstrong. *Principles of marketing*. Pearson education, 2010.

[2] Jo-Ting Wei, Shih-Yen Lin, and Hsin-Hung Wu. A review of the application of rfm model. *African Journal of Business Management December Special Review*, 4:4199–4206, 01 2010.

[3] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[4] Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems*, pages 1057–1064, 2002.

[5] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004.

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[7] Peter Rousseeuw. Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65, 11 1987.

[8] Tippaya Thinsungnoen, Nuntawut Kaoungku, Pongsakorn Durongdumronchai, Kittisak Kerdprasop, and Nittaya Kerdprasop. The clustering validity with silhouette and sum of squared errors. pages 44–51, 01 2015.

[9] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. 14, 03 2001.

[10] Yu Hirate and H. Yamana. Generalized sequential pattern mining with item intervals. *J. Comput.*, 1:51–60, 2006.