University of Pisa

Department of Computer Science

Data Mining
Group 18

Supermarket Analysis

Authors:

Donato Meoli
Enrico D'Arco
Luigi Quarantiello

January, 2020

Contents

1 Data Understanding

The dataset contains *471910* entries, each of them represent a purchase related to a supermarket made by a customer over a period of two years.

1.1 Data Semantics

The dataset contains 8 attributes that correspond to:

- **BasketID** (*24627*): a 6 digit integer number uniquely assigned to each purchase; it may start with *C* or *A*; if it starts with a C, it indicates a cancellation, if it starts with a with A it indicates a bad debt adjusted;
- **BasketDate**: the day (*from 2010/12/01 to 2011/12/09*) and time (*from 6am to 21pm*) when each purchase was placed;
- **Sale** (*∼4avg*): the unit product price, all in the same currency, probably in sterling;
- **CustomerID** (*4372 + 65073na*): a 5 digit integer number uniquely assigned to each customer;
- **CustomerCountry** (*37*): the name of the country where each customer resides;
- **ProdID** (*3953*): a 5 digit + (eventually) letters identifier uniquely assigned to each distinct product; identical codes with different letters identify the same products with different characteristics (*e.g.*, *84997D*: '*PINK PIECE POLKADOT CUTLERY SET*' vs. *84997C*: '*BLUE PIECE OLKADOT CUTLERY SET*');
- **ProdDescr** (*4097 + 753na*): the description of the product purchased;
- **Qta** (*∼11avg*): the purchased quantities of each product per order.

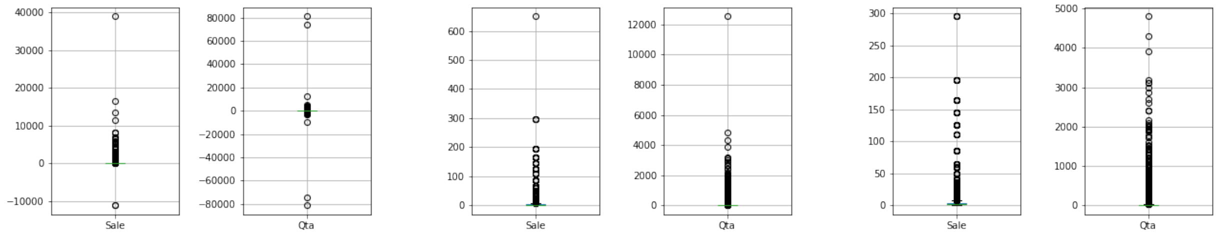
1.2 Assessing Data Quality

In order to assess the quality of data, we proceed by removing the *5232* duplicate entries which represents the 1.11% of the entire dataset, so we will work with the remaining *466678* rows.

From the plots in Figure 1a it's possible to see that the two numerical attributes have really high outliers, both positive and negative. The presence of negative values is in contrast with the semantic of the attributes since they should be positive.
From the fact that *Qta* seems to have a symmetric behaviour we can assume that a negative Qta represent a *refund.* This hypotesis is also supported by the fact that almost all the records with negative *Qta* have a *BasketID* starting with *C*, which may stand for *cancellation.* Note that there are records with negative *Qta* whose *BasketID*'s don't start with *C*; by analyzing the corresponding *ProdDescr* we can conclude that they refer to errors or damaged items.
For what concerns the negative *Sale*, there are just two records with that property, and from the *ProdDescr* ('*ADJUST BAD DEBT*') we can conclude that are due to errors. All the rows which can be identified as *errors* have a *Null CustomerID*

Then, we proceed by removing the entries corresponding to the *65073* null *CustomerID* values, the 13.94% of the dataset, since our main goal is to outline the customer behavior and there is no way to integrate them to trace the customer's orders. By doing so we also deleted the errors discussed before.

Afterwards, we removed the *ProdID* that does not respect the defined format and we found some that contains only letters, *e.g.*, '*POST*', '*D*', '*C2*', '*M*', '*BANK CHARGES*', etc. with the following respective *ProdDescr*: '*POSTAGE*', '*Discount*', '*CARRIAGE*', '*Manual*', '*Bank Charges*', etc.. As a result, we dropped *1273* entries.



(a) (b) (c)

Figure 1: *Qta* and *Sale* boxplots before and after data cleaning and outliers removal

1.3 Variables Transformations

In order to manage the presence of negative quantities as shown in the first boxplot in Figure 1a, we decide to keep track of the portion of each order that has been canceled. At this point we should face three different cases:

- a *cancelation exists with a counterpart*, i.e., exists an order with the same (but positive) quantity and a previous date, so both are canceled;
- a *cancelation exists without a counterpart*, this is probably due to the fact that the orders were performed before December 2010 (the entry point of the database), so we remove this;
- a *cancelation exists with multiple counterparts*, so we delete the most recent.

To also manage the presence of the remaining prices equal to zero, 34 entries, as shown in the second boxplot in Figure 1a, we filled these values with the average of the selling prices of the products with the same id.

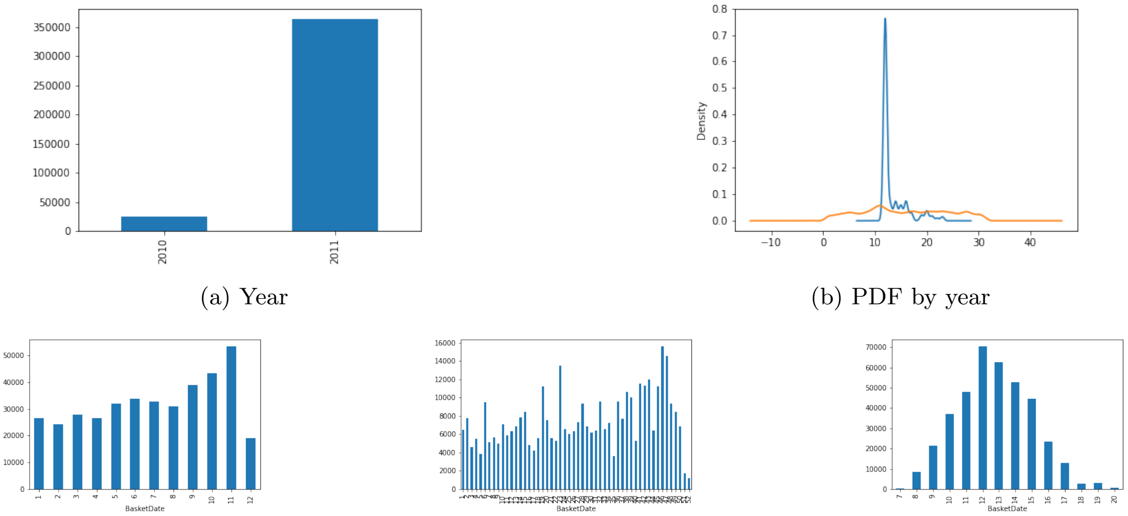
As we can see in Figure 1b, the previous operations already cleaned some of the outliers we had in the original dataset; now we manually check them, to determine if they are errors or not.

In the case of *Sale*, we have that the maximum value is 649.5, which is quite high but we discovered that this purchase is referred to a 60 pieces of an item, *i.e. picnic basket wicker 60 pieces*, so we proceed by dividing the *Sale* value by 60 and incrementing the *Qta* with the same value.
In the case of *Qta*, we see just one value are really away from the others, and it represent huge purchases, with quantity equal to 12540. We decided to drop the row.

We decided to create the attribute *TotSale*, i.e., the product between *Sale* and *Qta*, that represents the total amount spent by a customer for each type of product purchased. We made this choice mainly to have a clearer look to the dataset, emphasizing an important information that was not explicit in the original table, and also for the feature extraction later.

1.4 Variables Distribution

From the Figure 2a, we can see that the attribute is highly unbalanced; in fact, we have that almost all the records are related to transaction of the 2011, while the objects from 2010 are very few. Indeed, the rows of 2011 represent about the 93% of the whole dataset.
Furthermore, from the Figure 2b, that represents an estimation of the probability density function of *BasketDate* divided by year, we can appreciate the two different distributions.
In fact, for the 2010, we have a very uneven plot, which indicates that the records are not uniformly distributed with respect to the days in a month. This because, for the majority of the months in 2010, there were registered only transactions from a single day, the 12th; this is the value for which the plot shows the peak. On the other hand, the distribution for the 2011 is much more homogeneous, meaning that the transactions were registered for most days in the months of that year. For these reasons we decided to remove the entries of the 2010, since



(a) Year

(b) PDF by year

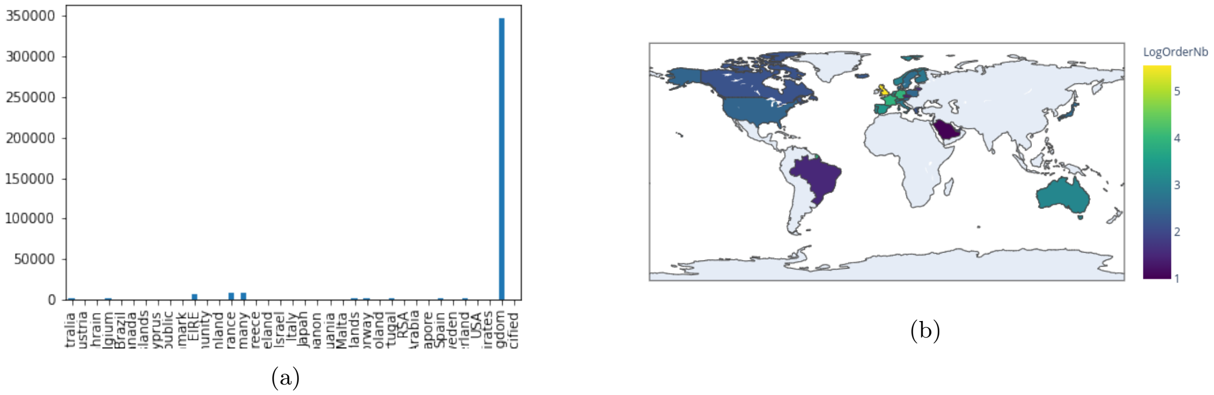
(c) PDF by month

(d) PDF by week

(e) PDF by hour

Figure 2: BasketDate distributions

the data were not taken regularly, and so they could alter our next study. We will focus on the 2011 entries, which were present much more uniformly and for this reason they could be more representative.
Other interesting distributions are plotted in Figure 2c, 2d and 2e. In the first one, we can see that the last weeks of the year are the one with more purchases; that is consistent with our expectations, since those are the weeks closest to Christmas time, that typically represents a great period of shopping. This thesis is supported also from the second one, in which we can see that December is the month with the most purchases. The third one is focused instead on the hours in a day; we found that, unsurprisingly, the most popular hourly is lunchtime.



(a)

(b)

Figure 3: CustomerCountry distributions

In Figure 3a and 3b, we can see the distribution of the *CustomerID* with respect to the country; from the plot, it is clear that the most frequent country is the *United Kingdom*, that is present in about the 90% of the rows.

Finally, we see some informations about the correlation of the attributes, to see if some of them are redundant.

From the Figure 4, we can see that almost all the attributes are uncorrelated, except for *TotSale*, that shows ah high correlation with *Qta*; that follows what we expected, since *TotSale* is, by construction, dependent on *Qta*. So, we conclude that all the original columns are independent, and so we don't need to perform any further manipulation.

We ended up with a cleaned dataset, consisting of *363577* entries.

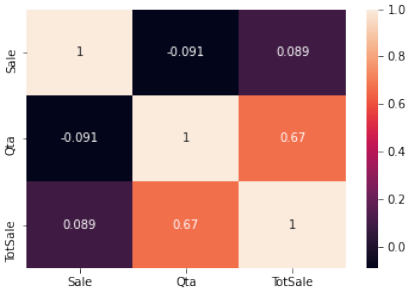


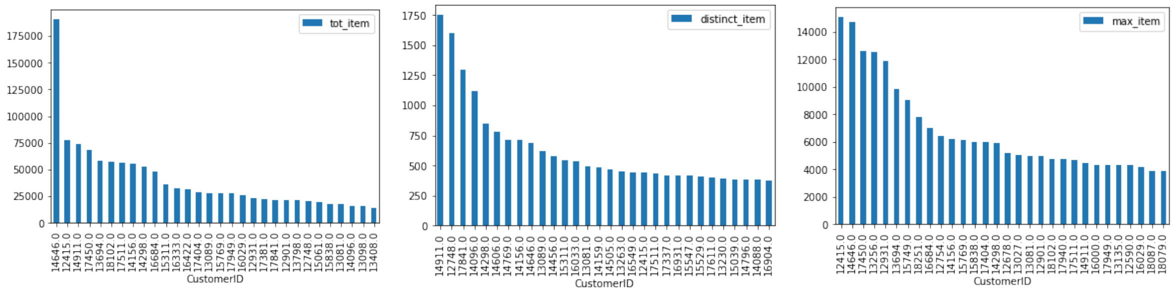
Figure 4: Correlation Matrix

2 Data Preparation

In order to describe the customers behavior, we extract the following new features from the dataset:

- the total number of items purchased by a customer;
- the number of distinct items bought by a customer;
- the maximum number of items purchased by a customer during a shopping session.

In Figure 5, we can see some visualization for these features; in particular, they represent the first 30 customers with the biggest values for each feature.
An interesting information is clear from the plot 5c, where we can see that the maximum quantities purchased in a single shopping session are very big; they are all above 3500, with the maximum equal to 15049. These are very high values, unlikely for a retail customer; this led us to think that the supermarket in question also sells wholesale.



(a) Total number of items per customer

(b) Number of distinct items per customer

(c) Maximum number of items per customer

Figure 5: Visualization for the extracted features

Now, with respect to the *TotSale* attribute we can take into account:

- the average price spent by a customer during a shopping session;
- the Shannon entropy on the purchasing behavior of the customer.

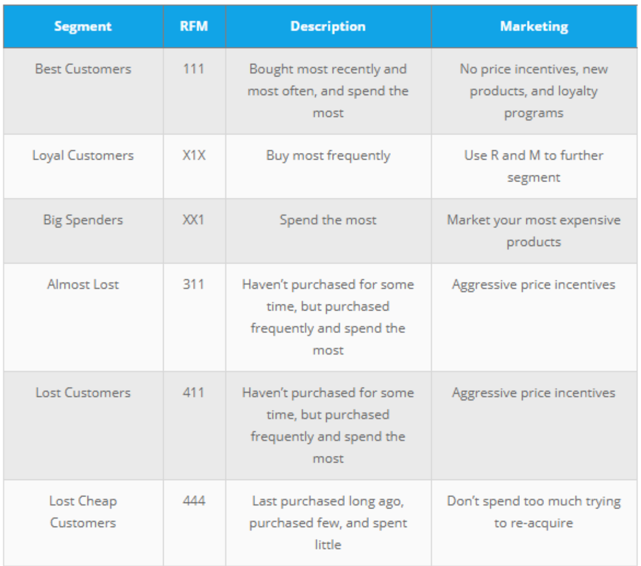
The entropy represents the variability of the customer's spending habits, *i.e.* a bigger value means that the customer did not have a regular behavior since he spent always a different amount of money, while lower values identify predictable spending behavior since the customer tends to spend always the same amount of money.
As shown in 8a, we have a small peak corresponding to 0, meaning that there are some customers with very specific habits, but the maximum is reached for ~ 4, which means that the majority of the clients have a quite unpredictable behavior.

2.1 RFM Model

At this point we decide to introduce a domain specific model, called *RFM*, to provide an interesting customer segmentation based on the purchasing behavior of the customers. In particular, the *RFM (Recency, Frequency, Monetary)* analysis refers to:

- *Recency* is the number of days between present date and date of last purchase each customer;
- *Frequency* is the number of orders for each customer;
- *Monetary* is the purchase price for each customer.

and it helps divide customers into various categories or clusters to identify customers who are more likely to respond to promotions and also for future personalization services as shown in 8a.



Segment	RFM	Description	Marketing
Best Customers	111	Bought most recently and most often, and spend the most	No price incentives, new products, and loyalty programs
Loyal Customers	X1X	Buy most frequently	Use R and M to further segment
Big Spenders	XX1	Spend the most	Market your most expensive products
Almost Lost	311	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Customers	411	Haven't purchased for some time, but purchased frequently and spend the most	Aggressive price incentives
Lost Cheap Customers	444	Last purchased long ago, purchased few, and spent little	Don't spend too much trying to re-acquire

Figure 6: RFM Segmentation

To calculate the individual RFM we will use the quartil statistical method, i.e. dividing score into four parts, by associating them a number from 1 to 4, so we will deal with the following:

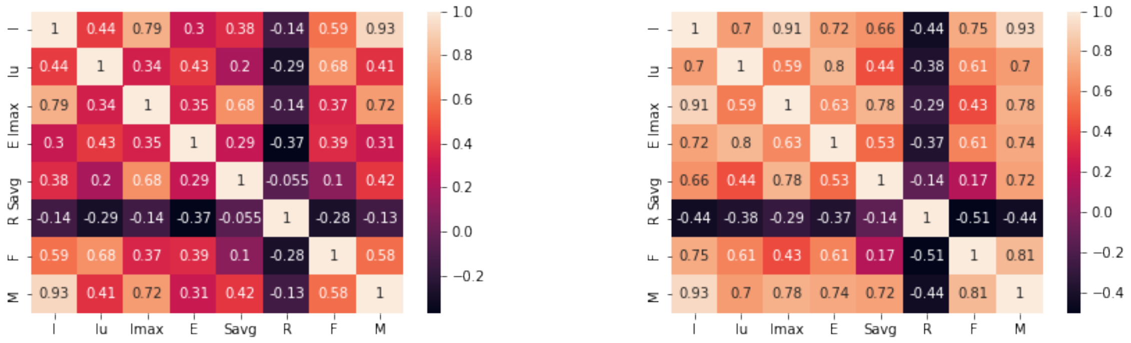
- best *Recency* score (= 1): most recently purchase
- best *Frequency* score (= 1): most quantity purchase
- best *Monetary* score (= 1): spent the most

Wrt our dataset the customer segmentation result as following:

- the number of *Best Customers* is 440
- the number of *Loyal Customers* is 1012
- the number of *Big Spenders* is 1051
- the number of *Almost Lost* is 105
- the number of *Lost Customers* is 11
- the number of *Lost Cheap Customers* is 431

Now, we can visualize the correlation between the attributes. In Figure 7a, as expected, we find that the average basket value is highly correlated with the total quantity purchased and the amount of money spent. Furthermore, we can see that also the year frequency is correlated with the amount spent and the total quantity. In the end, of course, the quantity purchased is very highly correlated with the total amount spent.
Since the dataset deal with both retail customers and wholesalers, we have that some attributes have really spread values. To weight less the difference between high value with respect to the difference between small values, we took the logarithm in base ten of those attributes.

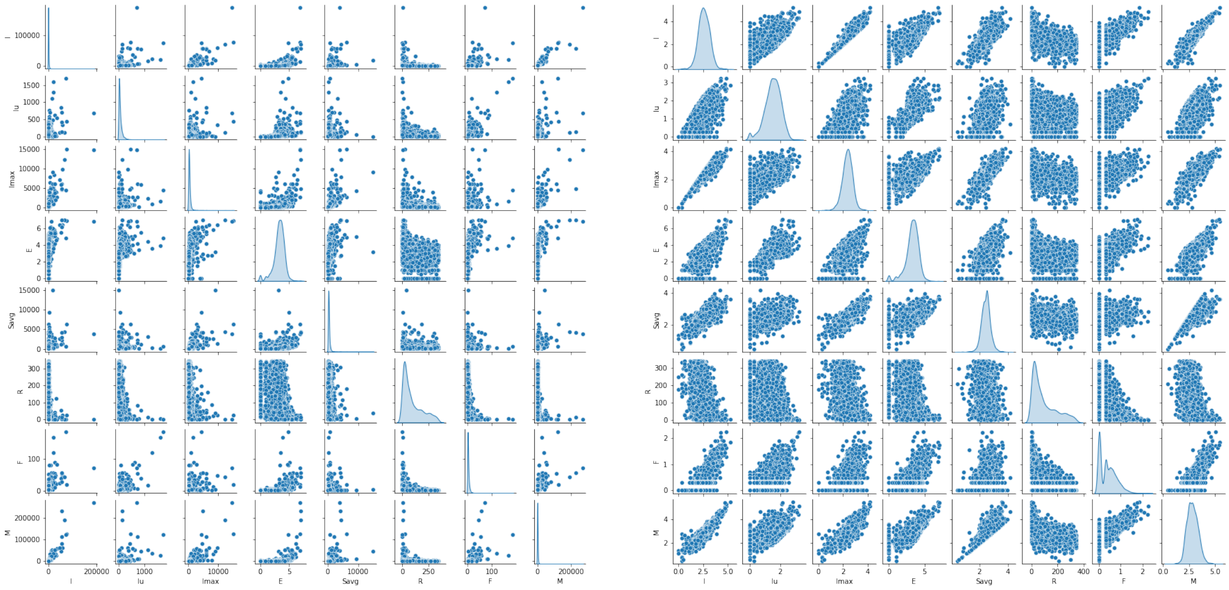
In Figure 7b and Figure 8b we can see the correlation matrix and the pairplot after computing the logarithm in base ten of I, Iu, $Imax$, $Savg$, R, F and M.



(a) (b)

Figure 7: Correlation Matrix before and after log-normalization

In Figure 8, we can visualize the pairplot for the attributes we have.
On the diagonal, we have the density plots, where we can appreciate the distribution of the features. Instead, the other scatter plots show the relationship between two variables. By analyzing those, we can have a confirmation of what we already found thanks to the previous plot.



(a) (b)

Figure 8: Pairplots before and after log-normalization

3 Clustering

We now run and compare some clustering algorithm in order to find some structures among the data. First we start with a basic *K-Means*, followed by *Hierarchical clustering techniques* and *DBSCAN*. In the end, we also analyze the behavior of other algorithms, like *Fuzzy C-Means* and *Birch*.

3.1 Preprocessing

The data matrix was first standardized and then the two principal components were extracted. The choice was between two and tree principal component since they retain respectively ∼0.75 and ∼0.89 of the variance of the data.

We selected the two principal components, in this way, we could also have a visual inspection.

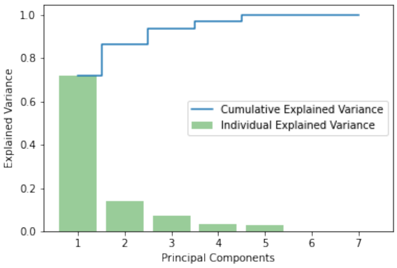


Figure 9: Explained variance ratio

3.2 K-Means

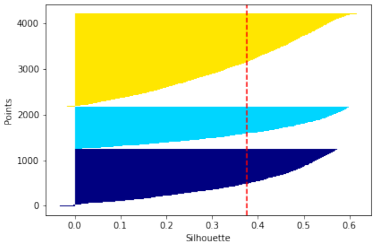
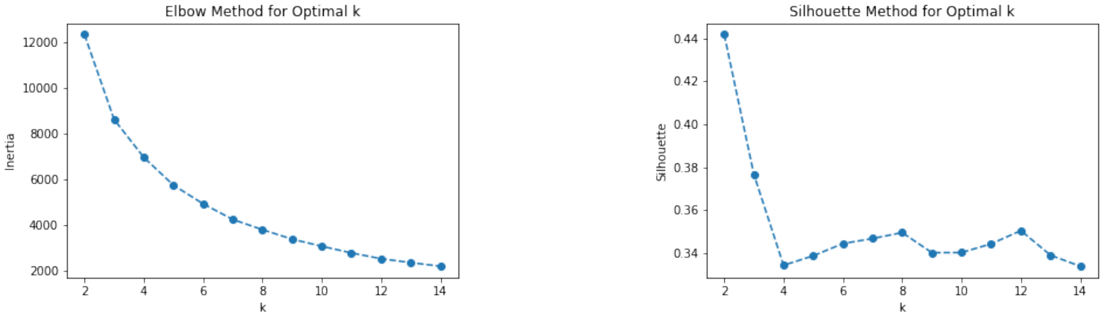


Figure 10: Silhouette score for each data point

The algorithm runs for K ranging from *2* to *15* clusters. For each iteration the SSE and the average silhouette value were computed.

By looking at the plots in Figure 11, we can see that there isn't a prominent *elbow shape*, so we needed the analysis of more metrics to get the optimal number of clusters. By inspecting the silhouette score, it is possible to see that it has its maximum for $K = 2$, followed by $K = 3$. In conclusion, we opted for 3 clusters, to get a trade off between high silhouette and low SSE.



(a) SSE

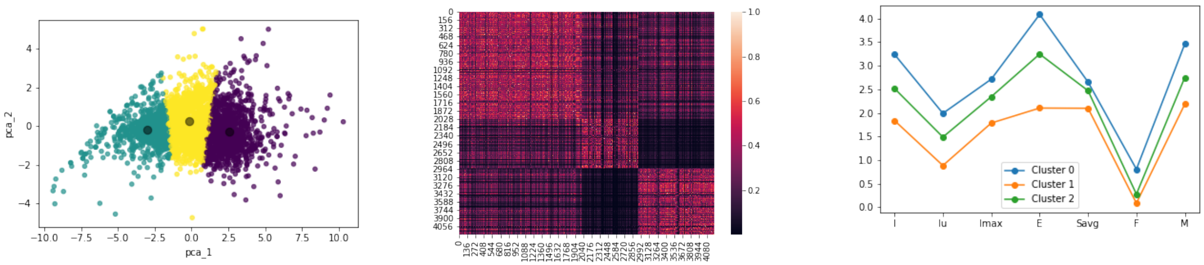
(b) Average silhouette

Figure 11: *K-Means* metrics

The resulting clusters have a comparable number of points. In particular, *Cluster 2* has **2006** points, *Cluster 1* **1440** and *Cluster 0* **760**.
In Figure 10, we can see the plot of the Silhouette score, where each different color represents a different cluster, and the dotted red line is the average silhouette score. In particular we have that *Cluster 0* has a silhouette of *0.39*, *0.40* for *Cluster 1* and *0.35* for *Cluster 2*, with an overall average silhouette of *0.39*. We see that a small fraction of points in *Cluster 0* have a negative value but overall the Silhouette suggests a good clustering.

In Figure 12a, we can appreciate the results of the algorithm, where we can see that the clusters are well separated from one another.
The similarity matrix (Figure 12b) shows the affinity of elements inside a cluster; we can see that the results

are pretty good. Customers within the same cluster are similar, where similarity between two customers is measured using e^{-d}, where d is the eucledian distance.
The plot in Figure 12c shows the average values for each attribute divided by the clusters. In particular, it is possible to see that *Cluster 0* contains the most frequent and spending customers, followed by *Cluster 1* and *Cluster 2*.



(a) Clustering result

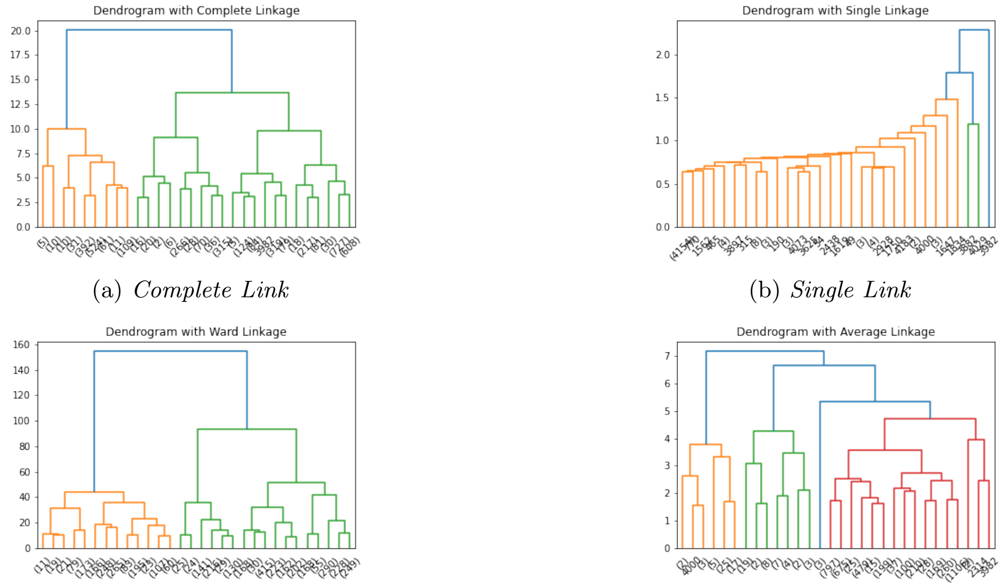
(b) Similarity heatmap

(c) Average values per cluster

Figure 12: K-Means

3.3 Hierarchical clustering

We used different kinds of algorithm: *Complete Link*, *Single Link*, *Ward Link* and *Average Link*. By looking at the dendograms in Figure 13, if we cut the tree by selecting two or three clusters, we can see that *Single Link* and *Average Link* generate an unbalanced clustering, leaving the last merges between a large cluster and a small one, in a case even a singleton. The most balanced results are obtained with *Ward Link* and *Complete Link*.



(a) *Complete Link*

(b) *Single Link*

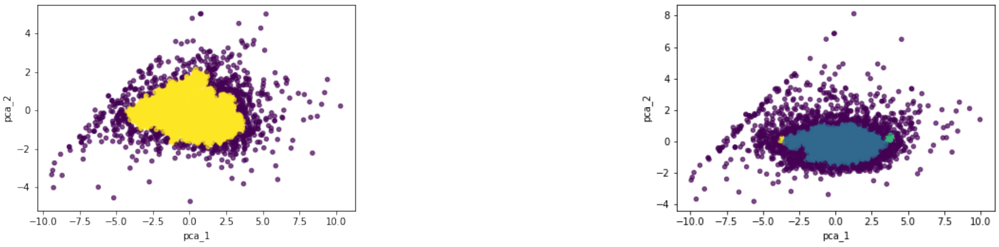
(c) *Ward Link*

(d) *Average Link*

Figure 13: Dendograms

3.4 DBSCAN

We explored different combinations of *eps* for a given *MinPts*; to choose *eps* we checked the *KNN* distance, with *K* equal to *MinPts*.
If *MinPts* is 20, the optimal *eps* is around 0.25; in this case, the algorithm found a large dense area surrounded by noisy points, as is showed in Figure 14a. By increasing *eps* to 0.3 the results are the same, if instead we use the value of 0.2, the algorithm finds more clusters of negligible dimension.



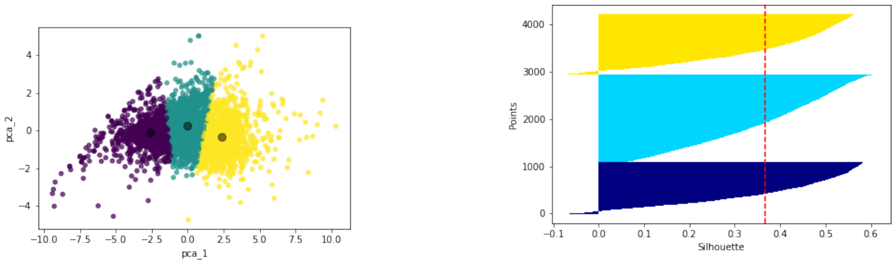
(a) *eps*= 0.25

(b) *eps*= 0.2

Figure 14: Results of DBSCAN

3.5 Other clustering

Together with the previous algorithms, we also tried to run *Fuzzy C-Means* and *Birch*. They both generates a good partitioning, in particular *Fuzzy C-Means* result's resemble the *K-Means* clustering. However the silhouette score is *0.38*, slightly lower than *K-Means*.



(a) SSE

(b) Average silhouette

Figure 15: *Fuzzy C-Means* results

For what concerns *Birch* result's, we have that, even in this case, the algorithm is able to partition the customers into three categories but it generates unbalanced classes. In fact the clusters' cardinalities are *188*, *1315* and *2703*.

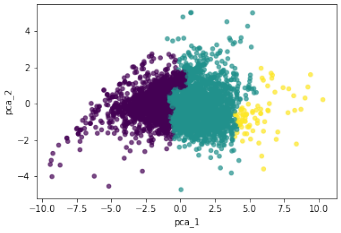


Figure 16: *Birch* results

3.6 Conclusions

In conclusion, we opted for the *K-Means* clustering to characterize the customers. As is shown in Figure 17, the algorithm is able to identify three class of customers corresponding to the low, medium and high spending customers.

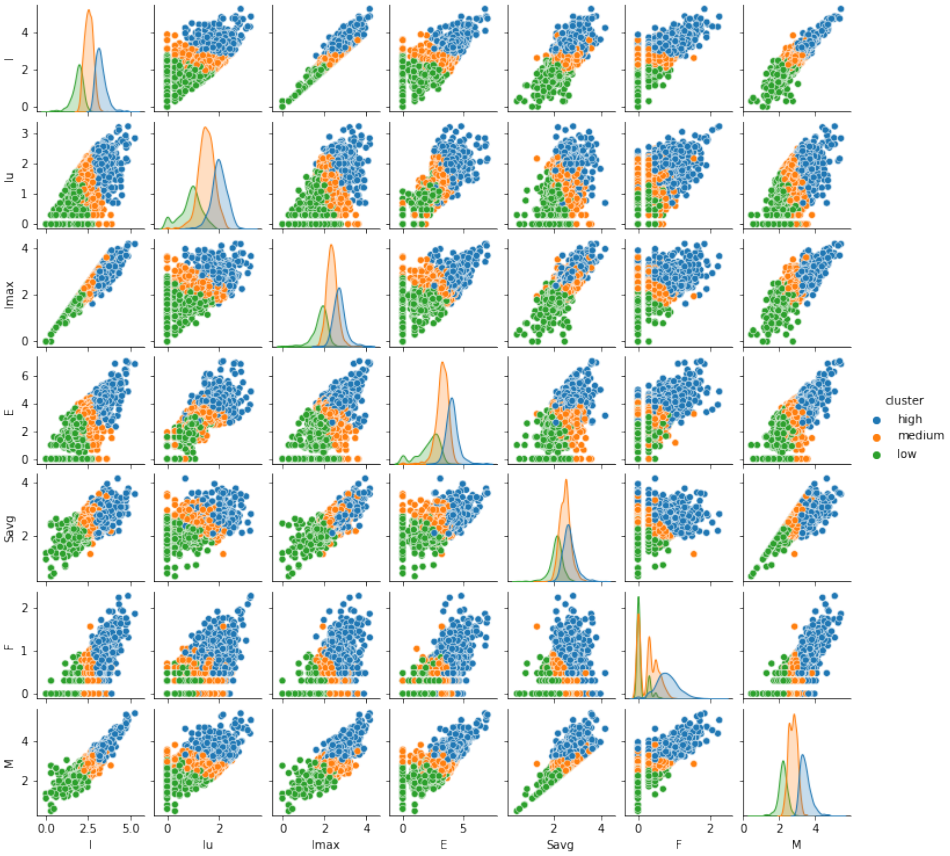


Figure 17: K-Means results

4 Predictive Analysis

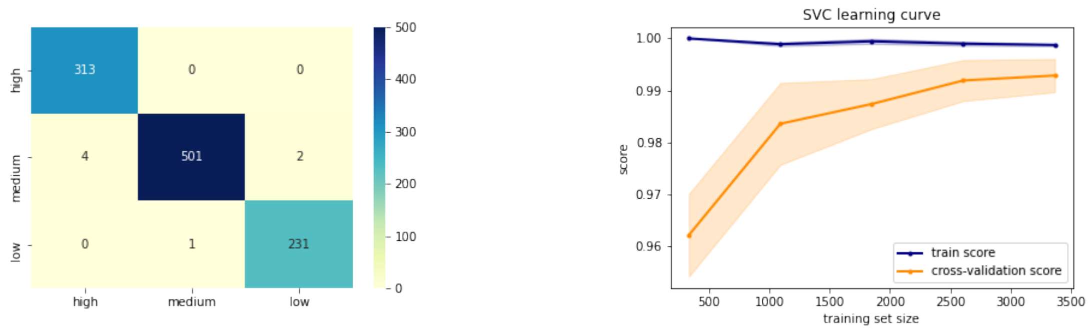
In this section, we consider the problem of predicting the spending behavior of each customer. We will use the main classifier models and evaluate their performances.

First, we will use the customer profile deriving from the K-means clustering, from the previous section. From the Figure 12a, we recall that we partitioned the dataset in 3 clusters, that represent the **high-spending** customers, the **medium-spending** customers and the **low-spending** customers. We can see that the clusters are well separated from each other, and so we can use them as customer classification.

To perform the task, which is a *supervised* one, we split the dataset into training and test set, equal to the 75% and 25% of the dataset respectively. Plus, during the splitting, we specify that each partition must have approximately the same relative class frequencies; that is to manage the unbalance we have in the original dataset, in which the **medium-spending** customers are much more frequent than the other classes.
We also standardize the data with a standard scaler, achieving a distribution with 0 mean and unit variance.

For each model, we perform a grid search, to find the best values for some hyperparameters; for that, we used a *5-fold cross validation.*

4.1 Support Vector Machine



(a) Confusion Matrix for SVM Classifier

(b) Learning curve for SVM Classifier

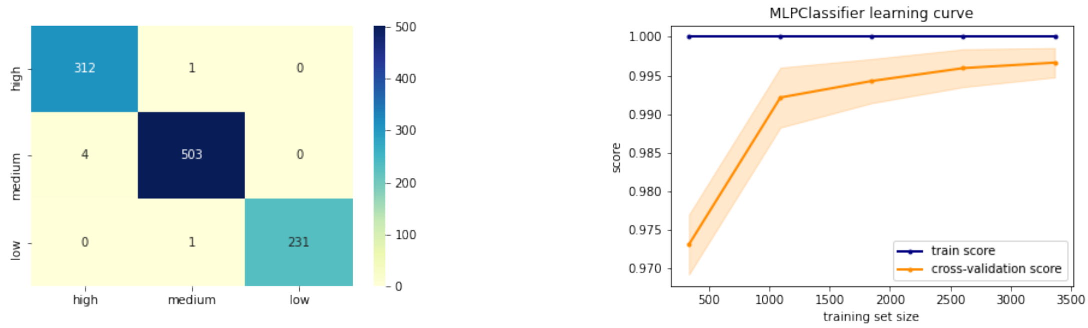
Figure 18: Support Vector Machine

The first model we analyze is the **SVM**, that is a classifier that searches for an hyperplane that can linearly separate the data points, possibly in a transformed space, in the non-separable case.

We use a Linear Support Vector Classifier and, after the grid search, we found that the best value for the regularization parameter is $C = 1000$.
With this configuration, we achieved a training accuracy of 0.9935 and a test accuracy of 0.9962.

From the Figure 18a, we can see the confusion matrix related to the results on the test set, and we notice that the model made only 4 mistakes on the whole dataset; for that, we have that also the precision and the recall for each class are all above 0.98.
In Figure 18b, we can see the learning curve for this model. We have that the training and validation scores are quite similar, even if, for small sizes of the training set, the train score is slightly greater; that means that more training examples helped increasing the generalization capability of the classifier. We can also appreciate, from the shadows around the curves, that the standard deviation of the scores decreases by increasing the size of the set.

4.2 Neural Network



(a) Confusion Matrix for Neural Network

(b) Learning curve for Neural Network

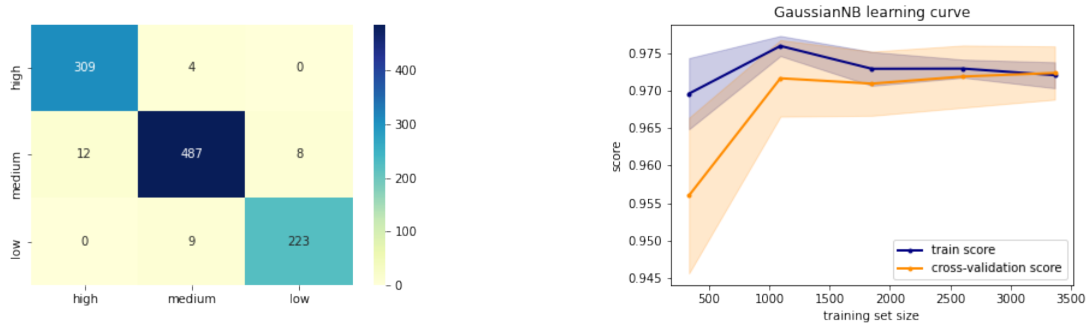
Figure 19: Neural Network

Now, we use a Neural Network, in particular a **Multi-Layer Perceptron**.
The best values for the hyperparameters that we found were:

- *1 hidden layer* with *50 units*
- *sigmoid* activation function
- *l-bfgs* optimizer, that can converge faster and with better results on small datasets like ours

With this model, we got a training accuracy of 1 and a test accuracy of 0.996; the reason is clear in Figure 19a, where we can see that very few data points were misclassified.
Furthermore, from Figure 19b, we notice an improvement in the validation score with the increasing of the training set size; in the end, the curves have more or less the same behavior.

4.3 Naive Bayes Classifier



(a) Confusion Matrix for Naive Bayes

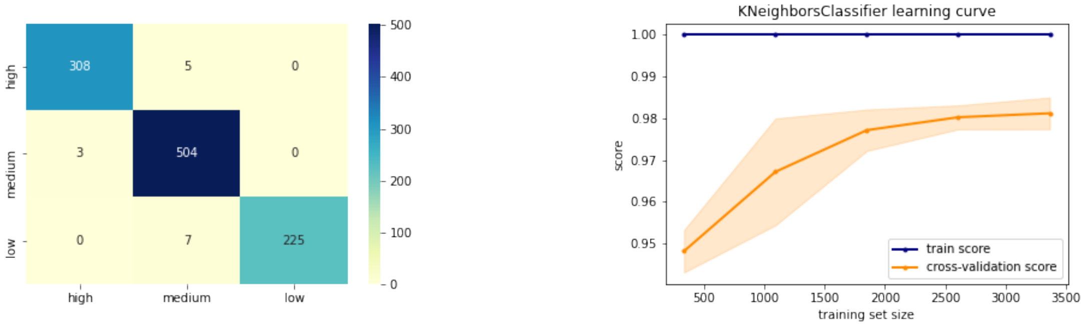
(b) Learning curve for Naive Bayes

Figure 20: Naive Bayes Classifier

The Naive Bayes Classifier tries to estimate the class conditional probability for each item, using the *Bayes* theorem, with the assumption that all the features are conditionally independent. In particular, we used a **GaussianNB**, where the likelihood of the features is assumed to be Gaussian.

In Figure 20a, we can see that the performances of this model are slightly worse than the previous ones; in fact, we achieve a training accuracy of 0.9635 and a test accuracy of 0.9743. The learning curve in Figure 20b shows that , with a small training set, the validation score is lower than the training one, with a big standard deviation; with a small increase in the size, the two values converge to a point, with no further improvements.

4.4 K-Nearest Neighbors



(a) Confusion Matrix
for K-Nearest Neighbors

(b) Learning curve
for K-Nearest Neighbors

Figure 21: K-Nearest Neighbors

The K-Nearest Neighbors model is an instance-based classifier, that, for each record, selects the class label based on the majority of its nearest neighbors.
The grid search showed that the best configuration is

- *n_neighbors* equal to 50
- *distance* as weight function; that makes the weight of a point to be inversely proportional to its distance

With these values, we have a training accuracy of 1 and a test accuracy of 0.9838. The Figure 21a shows that some points are misclassified, with some customers, belonging to the high and low profile, incorrectly labeled as medium.
Instead, from Figure 21b, we can see the training score is always equal to 1, while the validation score shows some improvements with bigger training sets.

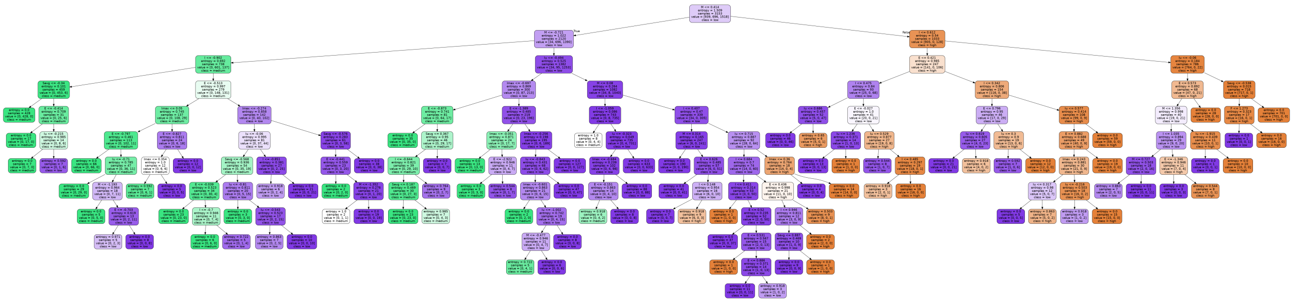
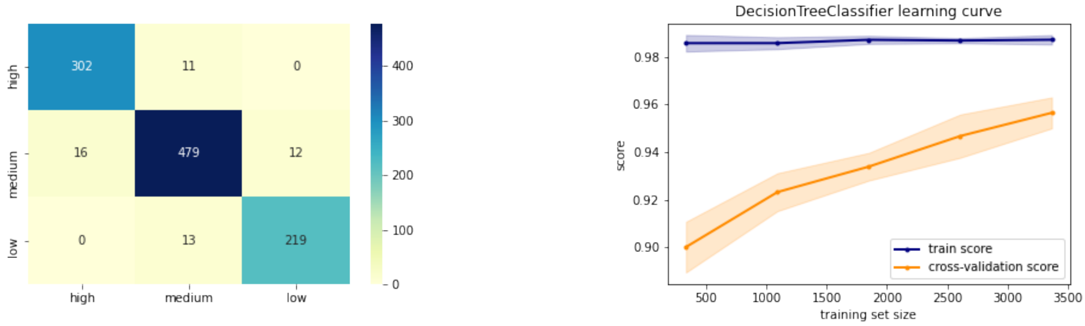


Figure 22: Decision Tree

4.5 Decision Tree



(a) Confusion Matrix
for Decision Tree

(b) Learning curve
for Decision Tree

Figure 23: Decision Tree

A Decision Tree model learns some decision rules from the training data, in order to grow a tree that is able to predict the class label for our test points. The best configuration found is:

- *criterion* equal to *entropy*
- *max_depth* equal to 500
- *max_features* equal to 5
- *min_samples_leaf* equal to 2
- *min_samples_split* equal to 10
- *splitter* equal to *best*

The best result is a training accuracy of 0.9830 and a test accuracy of 0.9468, but one of the advantages of this kind of model is that it is easy to interpret, since it can be visualized. In Figure 22, we can have a representation of the best estimator that we found.

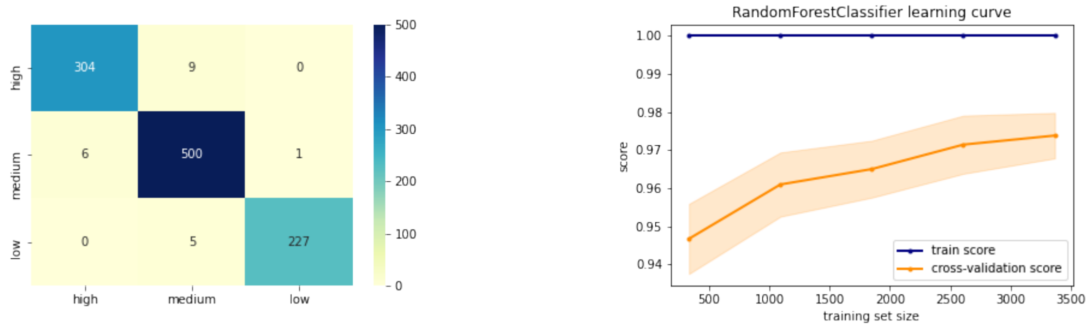
The Figure 23a shows the model makes several mistakes, with respect to the other classifiers, given that the decision tree is a quite simple model.
On the other hand, from the learning curve ,represented in Figure 23b, we can appreciate the distance between the training and validation score; as the size increases, the validation one increases, but it cannot reach the training value.

4.6 Random Forest

Lastly, we analyze the Random Forest. It is an *ensemble* model, composed by several decision trees, and the result is given by averaging the predictions of the single classifier. In this case, the best hyperparameters are:

- *max_depth* equal to 100
- *max_features* equal to 1
- *min_samples_leaf* equal to 2
- *min_samples_split* equal to 5
- *n_estimators* equal to 1500

As a results, we have a training accuracy of 0.9994 and a test accuracy of 0.9753; these results are confirmed by the Figure 24a, that shows that the model made some errors.



(a) Confusion Matrix for Random Forest

(b) Learning curve for Random Forest

Figure 24: Random Forest

The Figure 24b shows that, for small sizes of the training set, the validation score is quite low; with bigger sets, this score has some improvements but it is not able to approach the training score, that is constantly equal to 1.

4.7 Conclusions

In conclusion, the model that performs the best is the Neural Network, which is almost perfect both on training and test set; also the SVM, K-Nearest Neighbors and Random Forest have very good performances, having a very high accuracy.
Overall, even if the other models give slightly worse results, they all have an accuracy greater than 93%, which is pretty good. That is because the dataset is not very complex, since the clusters given by K-Means(Fig. 12a) are almost linearly separable.

5 Sequential Pattern Mining

The goal of this section is to describe the results obtained from the application of a *SPM (Sequential Pattern Mining)* algorithm in order to discover some frequent patterns in the products purchased by customers.

So, in order to achieve this goal, we need to model each customer as a sequence of baskets, sorted by *BasketDate*.

5.1 Generalized Sequential Pattern Mining

We now apply the *GSP (Generalized Sequential Pattern)* algorithm, an apriori-based Sequential Pattern Mining algorithm to discover the frequent patterns present in our dataset.
GSP is a breadth-first algorithm that starts by finding all the 1-sequence that are frequent, i.e. the first level candidate sequences, that are those patterns with support greater than the *min_sup*. We set *min_sup* equal to 5%, which is quite low.
Then, at each step, *i.e.* for each successive layer, it generates new longer candidates by merging pairs of the previous ones, and later it eliminates all the sequences with low support. It stops when no more frequent sequences are found.

We found 41 frequent patterns, all of them made up of just two products in the same sequence or in two different ones, except for one with three products in the same sequence.
By taking a first look at those sequences, we can notice that some of them are related to the same product, so we can imagine that these patterns are supported by regular customers or customers who are comfortable with that product. Some examples are:

- {*WHITE HANGING HEART TLIGHT HOLDER*}, {*WHITE HANGING HEART TLIGHT HOLDER*}
- {*JUMBO BAG RED RETROSPOT*}, {*JUMBO BAG RED RETROSPOT*}
- {*REGENCY CAKESTAND TIER*}, {*REGENCY CAKESTAND TIER*}
- {*ASSORTED COLOUR BIRD ORNAMENT*}, {*ASSORTED COLOUR BIRD ORNAMENT*}
- {*PARTY BUNTING*}, {*PARTY BUNTING*}
- {*LUNCH BAG RED RETROSPOT*}, {*LUNCH BAG RED RETROSPOT*}
- {*LUNCH BAG BLACK SKULL*}, {*LUNCH BAG BLACK SKULL*}
- {*LUNCH BAG SUKI DESIGN*}, {*LUNCH BAG SUKI DESIGN*}
- {*SET OF CAKE TINS PANTRY DESIGN*}, {*SET OF CAKE TINS PANTRY DESIGN*}

On the other hand, for the remaining sequences, we can notice that those are related to the same product with different characteristics in terms of shape, color, size, etc., so we can imagine that these patterns are supported by wholesale customers that purchase a stock of same products with different characteristics. Some of them are:

- {*GREEN REGENCY TEACUP AND SAUCER*, *PINK REGENCY TEACUP AND SAUCER*, *ROSES REGENCY TEACUP AND SAUCER*}
- {*GARDENERS KNEELING PAD CUP OF TEA*, *GARDENERS KNEELING PAD KEEP CALM*}
- {*HEART OF WICKER LARGE*, *HEART OF WICKER SMALL*}
- {*WOODEN HEART CHRISTMAS SCANDINAVIAN*, *WOODEN STAR CHRISTMAS SCANDINAVIAN*}

In conclusion, we can assert that there are not very interesting frequent patterns, *i.e.* patterns that involve different kinds of products, and so we can say that the customers' behavior is quite different from each other. This is probably due to the presence of wholesale and retail customers purchasing in the same dataset.

5.1.1 Time Constraints

To make the SPM phase more interesting we decided to involve some time constraints in the GSP algorithm known as:

- *max_span*: the maximum duration of the whole sequence;
- *min_gap*: the minimum time between two elements of the sequence;
- *max_gap*: the maximum time between two elements of the sequence.

The algorithm works as specified before, with the addiction that it prunes also all the candidates that violate these time constraints.

In order to perform the mining phase with these constraints by taking into account the time elapsed between two purchases, we need to expand our dataset definition by associating a timestamp to each basket. So we decided to use as timestamp the day of the year, since the dataset covers a period of about a year and the vast majority of the customers purchased at most one basket per day; hence, we defined *BasketDayOfYear*, a numerical attribute that goes from 2 to 344.

For our analysis, we choose *min_sup* equal to 5% and we decide to constraint the frequent patterns as follow:

- the overall duration of the pattern instance must be at most of 1 month, so we set *max_span* equals to 30 days;
- *min_gap*: each element of the pattern instance must be at least 1 day after the previous one, so we set *min_gap* equals to 1 day;
- *max_gap*: each element of the pattern instance must be at most 1 week after the previous one, so we set *max_gap* equals to 7 days.

As a result, we obtain that all the frequent patterns attributable to retail customers, i.e. those made up by two products in two different sequences, disappear, while all the frequent patterns attributable to wholesale customers remain unchanged. This helps us to support our thesis considering that often the wholesale customers buy products in stock, therefore all purchases are made at the same time.