Master Degree in Computer Science

Data mining project: Basket customers analysis

Antonio Pace - Francesco Botrugno - Francesco Caprari

Academic Year: 2022-2023

Contents

1 Task 1: Data Understanding & Data Preparation

1.1 Dataset overview

The data for this project consists of purchases made in various stores in Russia, it also contains information about the items stored and their categories.
There are three different datasets:

- **item_categories**
  contains the association between category identifier and its corresponding category name, the dataset consists of 84 record.
  - **item_category_name**: category's name.
  - **item_category_id**: category's identifier.
- **items**
  contains the association between product identifier and its corresponding category and name, the dataset consists of 22170 record.
  - **item_name**: item's name.
  - **item_id**: item's identifier.
  - **item_category_id**: category's id of the item.
- **basket_supermarket**
  contains information about purchases by customers in a certain period of time, the dataset consists of 504087 record, **Figure 1** shows a first distribution of the attributes and shows that correlation is low.
  - **date**: date of purchase.
  - **shop_id**: shop's identifier from which the purchase was made.
  - **item_id**: identifier of the item bought.
  - **item_price**: individual item's cost in Rubles (RUB).
  - **item_cnt_day**: number of items purchased in a single shopping session.
  - **user_id**: identifier of the user who purchased the item.
  - **basket_id**: identifier of the shopping session, the basket_id uniquely identifies the shop_id, the date and the user.

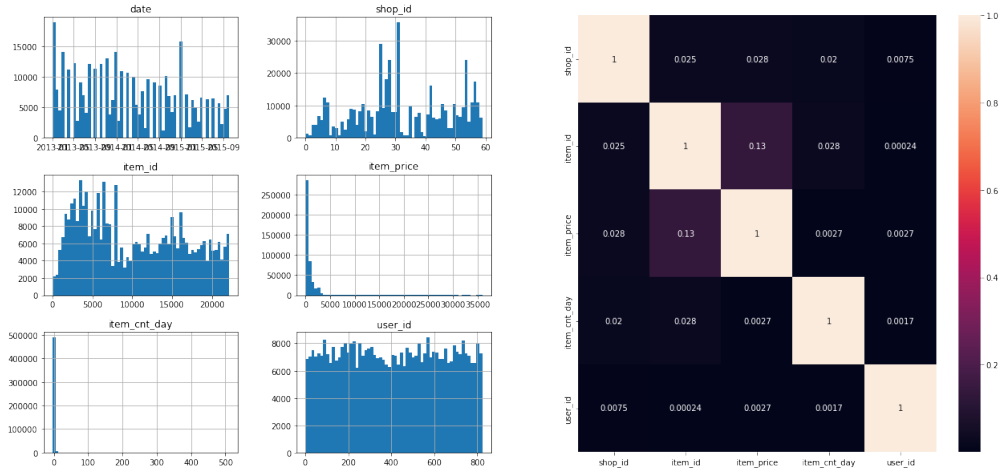

Figure 1: Distribution and correlation of basket_supermarket

1.2 Data Quality

This phase consists of improving quality of data, i.e. filling missing values, correcting errors and managing outliers. We notice that the types of the attributes are correct, the datasets don't contain any duplicated row, dates are reasonable (between 2013 and 2015) and basket_id is correctly associated to a single user, shop and date. We discover few negative values for item_cnt_day later converted to their absolute value.

In this report we will focus only on the most important phases: we will show in detail in Paragraph 1.2.2 some features we found while looking for errors, in Paragraph 1.2.1 how we managed missing values and in the last Paragraph 1.2.3 how we identified and managed outliers.

1.2.1 Errors and features

We suppose that items with very different prices among purchases could be errors, therefore we calculate a price coefficient as:

$$\frac{(maxPrice - minPrice)}{meanPrice} \times 100$$

Then we select items using $mean+3\sigma$ and exploring the results brings us discovering that probably the dataset is related to online shopping, because ones of the items having the highest difference in prices are *Доставка (по Москве)* (Delivery to Moscow) and *Доставка товара* (Ware delivery). We also find out the category *Подарки - Мягкие игрушки* (Gifts - soft toys) discovering that some shops used to gift snake toy plushies. We discover that there are many categories name including the word *Подарки* (Gifts), since their price distribution is similar to the global one we decide to keep these rows.

1.2.2 Missing values

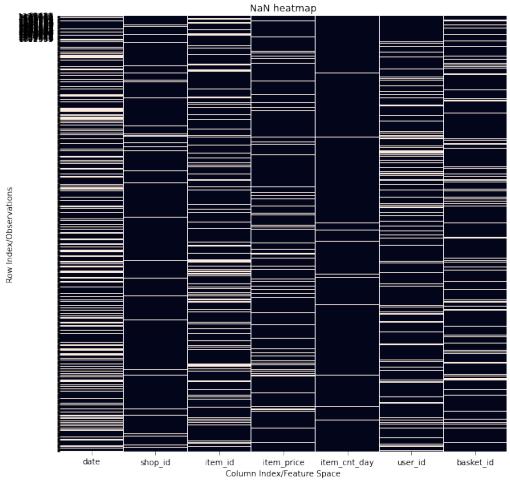

Figure 2: NaN values distribution of basket_supermarket

The only dataset containing NaN values is basket_supermarket. Since the number of global missing values is high (see Figure 2) we check whether it is good at least for individual users. We want to see if inferring data from user's behaviour is a good choice despite the high number of global NaN values and therefore if defining a profile for the users makes sense.

We analyse completeness of user's purchases calculating, for every attribute, the number of missing values over the total values available, and multiplying by 100 to get a percentage measure. Then we calculate a global completeness variable as the mean among all the attributes' completeness, discovering that every user has at least 80% of valid data and therefore we can infer missing values from the data available. A bad scenario would have been having many users on the top-left side of the graph, meaning that most of the users had very low completeness.

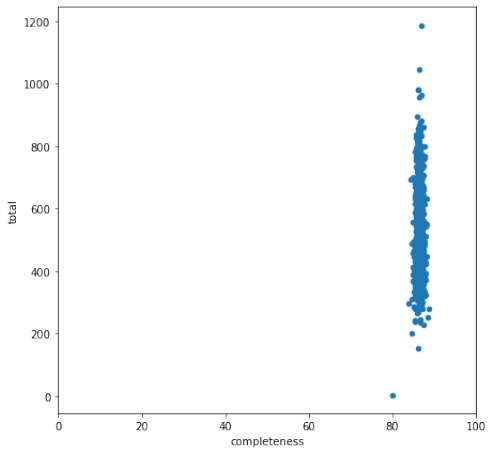

Figure 3: Scatterplot of global completeness

To fill NaN values in basket_supermarket first we eliminate the rows where there are more than six NaN values, or which have all id fields equal to NaN, since there is

no possibility of retrieving informations.

We handle replacing missing values differently depending on the type of attribute we're replacing.

- **date**: we replace the missing values exploiting basket_id, knowing that basket_id can only be associated with a single date. We do a *groupby* operation on the basket_id attribute, then we take the first value. Dates that still have NaN values are replaced with the median date for each user.

- **shop_id**: we replace the missing values exploiting basket_id, knowing that the basket_id can only be associated with a single shop_id. We do a *groupby* operation on the basket_id attribute, then we take the first value. Shop_id that still have NaN values are replaced with the most frequent shop_id for each user.

- **user_id**: we replace the missing values exploiting basket_id, knowing that the basket_id can only be associated with a single user. We do a *groupby* operation on the basket_id attribute, then we take the first value. For the user_id where the basket_id is NaN we delete the remaining rows since we need to profile users.

- **item_price**: We replace the missing values with the average price at which the product is sold. The remaining rows where item_id is still NaN are replaced with the global median of the price for all the items.

- **item_cnt_day**: we use the median to replace the NaN values.

- **item_id**: we set item_id to -1 where item_id is NaN, we also add the Unknown category and item to items and category dataframe.

- **basket_id**: we substitute the missing values exploiting the combination of the attributes date, shop_id, user_id, which together uniquely identify a basket_id. For the remaining rows that had one of the values equal to NaN, which therefore we cannot replace, we assign a dummy basket_ID (a progressive integer).

1.2.3 Outliers detection

The only fields that need to be adjusted are *item_cnt_day* and *item_price* in the basket_supermarket dataset. We use the mean and the standard deviation to identify the noisy values since the distribution is left skewed, in both cases we replace the noise points with the median. We use Box plots to validate the operation, since the data is skewed we can't expect a "clean" Box plot but we can still notice an improvement as we can see comparing Figure 4 with Figure 5. The percentage of outliers detected in this dataset is 0.75% for item_cnt_day and 0.96% for item_price and 0,96% - 1,71% overall.

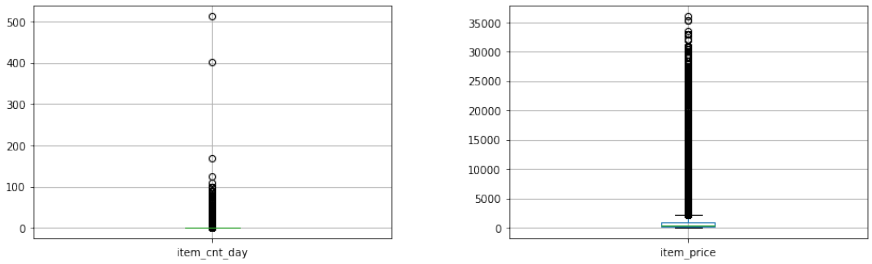

Figure 4: Boxplot before outliers detection

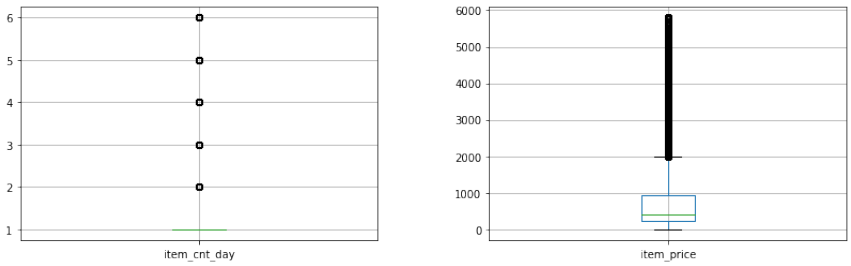

Figure 5: Boxplot after outliers detection

1.3 Indicators and users profiles

We create a second dataset that describes a user's profile, *UserProfile.csv*. The new dataset contains the following attributes:

- **I**: the total number of items purchased by a customer during the period of observation.
- **Iu**: the number of distinct items bought by a customer in the period of observation.
- **Imax**: the maximum number of items purchased by a customer during a shopping session.
- **IAvg**: the average number of items purchased by a customer during a shopping session.
- **entropy_itemID**: the Shannon's Entropy computed on the item_id values. It's a measure the diversity of products purchased by the customer.
- **entropy_CategoryID**: the Shannon's Entropy computed on the categoryId of the item, measures the diversity of the types of products the customer buys.
- **totalAmount**: is the total spent by each user, we calculate the amount spent multiplying the item_cnt exchange rate with the price field.
- **meanAmount_BSKT, maxAmount_BSKT, minAmount_BSKT**: for every user we calculate the min, max and average amount spent.

- **stdAmount_BSKT**: for every user we calculate the standard deviation of amount spent per basket.
- **unique_baskets**: how many shopping sessions are associated with the customer.
- **avg_month_spent**: it represents how much on average a customer spends each month.
- **stdMean_ratio**: is the Coefficient of Variation, the ratio between stdAmount_BSKT and the mean of the amount spent per basket.
- **totalAmount_factor**: is the product between totalAmount and stdAmount_BSKT.

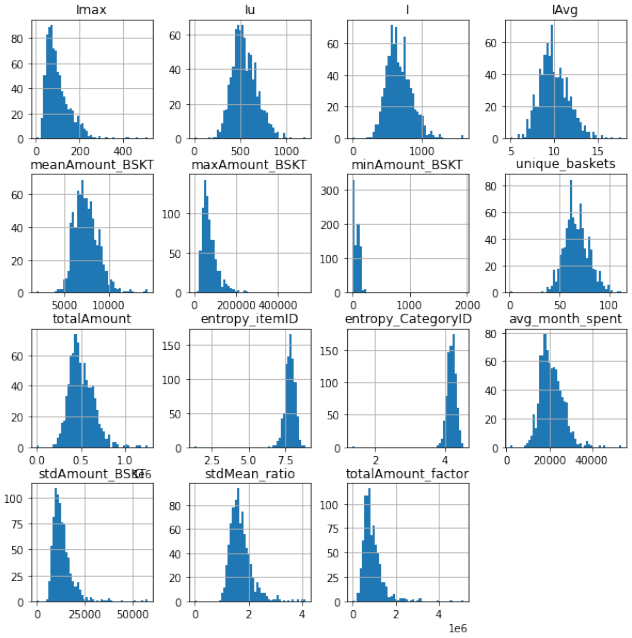

Figure 6: Features distribution before outliers detection

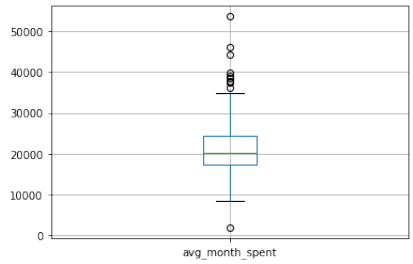

Figure 7: A box plot before outliers detection

1.3.1 Outliers detection

After defining indicators we observe their correlation, their distribution (Figure 6) and box plots (Figure 7). To improve the quality of our data we try to identify outliers. We

use two methods depending on the attribute distribution: if the distribution is clearly Gaussian we use quantiles method, if the distribution is mostly skewed we decide to cut at $mean \pm 3\sigma$. We always check that the number of outliers for every attribute isn't too high in proportion to the dataset length. After identifying noisy values we substitute them with the mean or the median, again depending on distribution. To check the effectiveness of our method we compare final distribution (Figure 8) and box plots (Figure 9) with the previous ones.

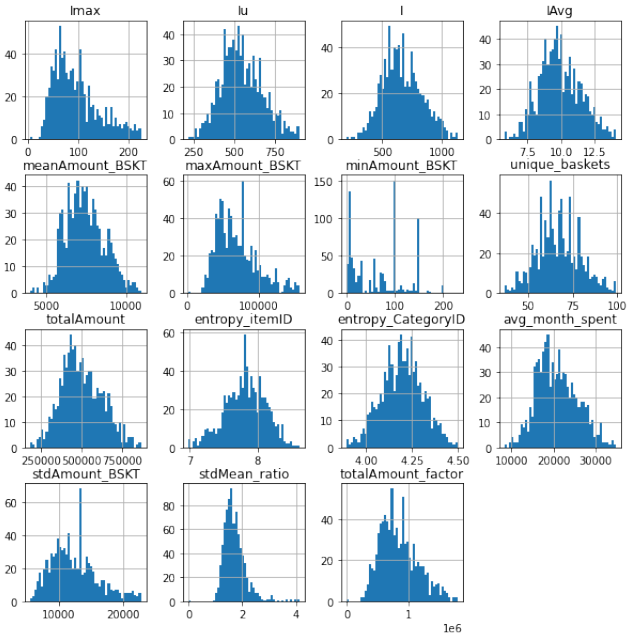

Figure 8: Features distribution after outliers detection

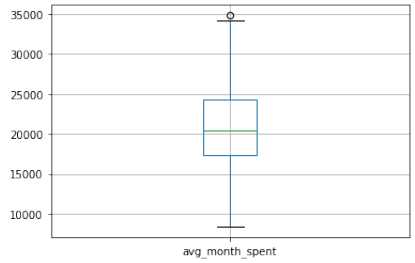

Figure 9: A box plot after outliers detection

2 Task 2: Clustering

We evaluate different types of clustering methods which are K-Means, hierarchical, and DBSCAN.

2.1 Clustering by K-means

First we eliminate highly correlated features identified by using the correlation matrix shown in Figure 10, we use 0.9 as the limit to eliminate indicators. Eliminating high

correlated features is not mandatory but can be useful, and since we have very similar indicators (like Iu and I) removing them can turn out to be a good choice. We tried different combinations between the indicators for choosing the best ones for the K-Means, founding: [*stdMean_ratio, maxAmount_BSK, stdAmount_BSKT*].

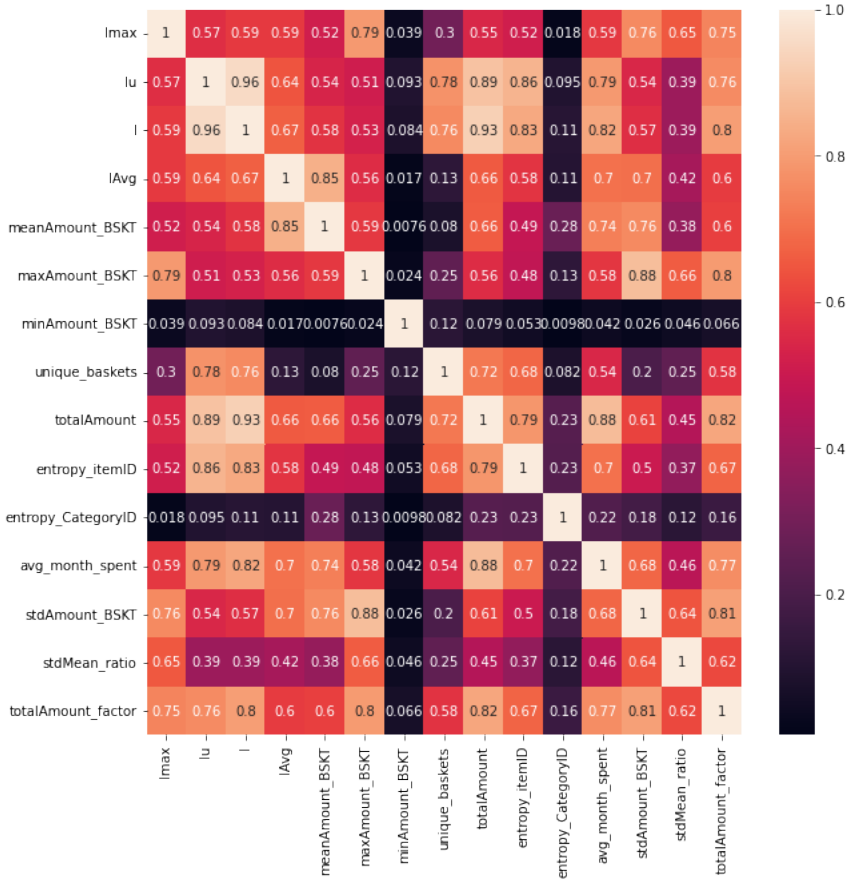

Figure 10: Distribution of feature after outliers substitution

To choose the best parameters for k-means we use the Elbow Method, so we estimate the value of the Silhouette and SSE for different values of K. Given the graphs in Figure 11, we pick K=3 which is a good trade-off returning an SSE of 18.31 and silhouette score equal to 0.46.

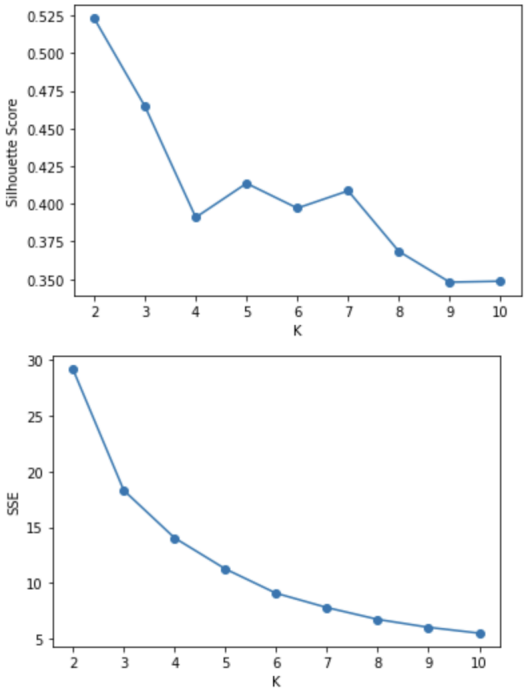

Figure 11: Silhouette and SSE for different values of K

In Figure 12 we can see an overview of the result for the K-means with the scatter plots, for every pair of attributes.

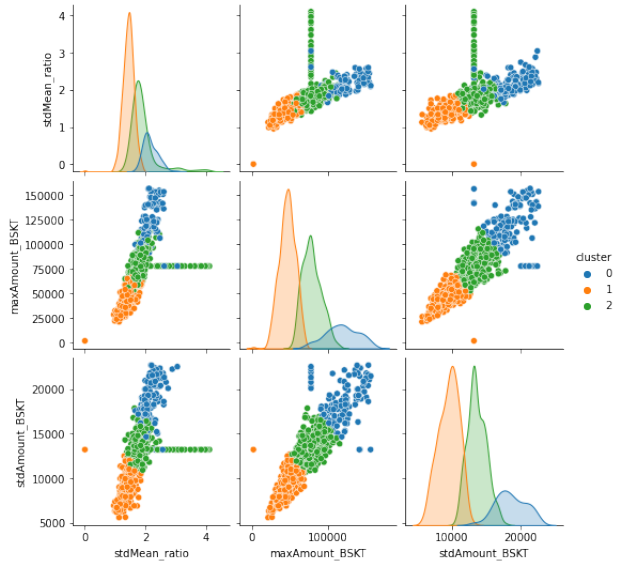

Figure 12: Clustering with K-Means

The similarity matrix in Figure 13 shows how points that belong to the same cluster are close to each others but it isn't very significant in our scenario given that we don't have well defined clusters.

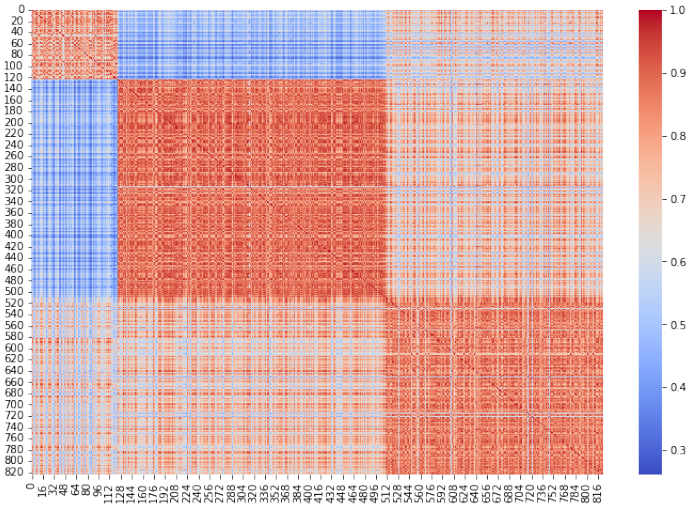

Figure 13: Similarity matrix after K-Means

Figure 14 gives a visualization of the clusters' centroids. We can see that *Cluster 0* contains the most spending customers and *Cluster 0* the least spending ones. We can suppose that K-Means creates three clusters which are the three different labels (low-spending, medium-spending and high-spending). The silhouette score obtained is not very high, this can be due to the fact that K-Means is not good for high correlated data and works better for spherical clusters.

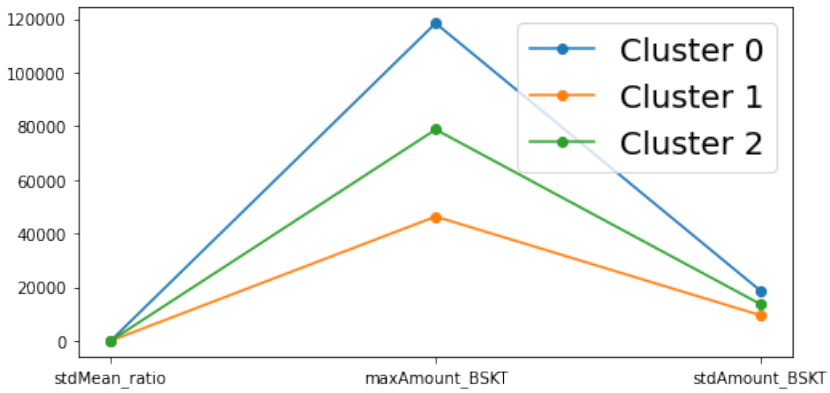

Figure 14: Centroid plotting after K-Means

2.2 Hierarchical Clustering

For the hierarchical clustering we use three different methods: average (Figure 15), single (Figure 16) and complete (Figure 17). Since there isn't a general criteria to establish the cut-off point of a dendrogram we consider the silhouette measure and the distance on the y axis cutting where branches are longer and clusters are most far away from each other.

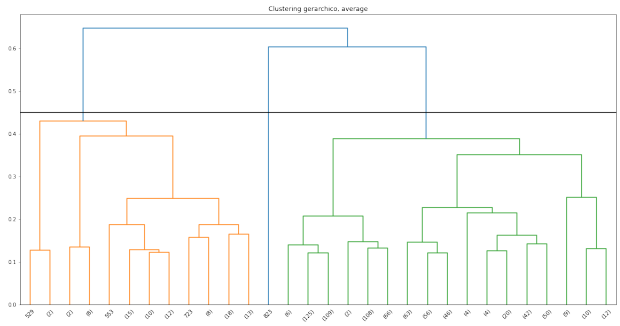

Figure 15: Average method

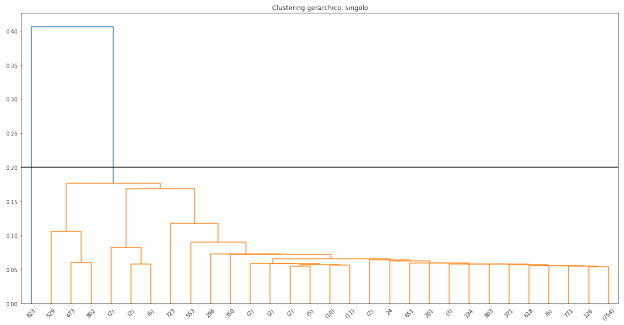

Figure 16: Single method

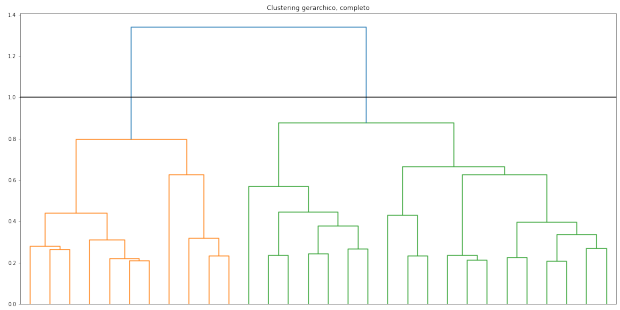

Figure 17: Complete method

Method	N°Cluster	Silhoutte
Complete	2	0.5069
Single	2	0.4509
Average	4	0.3937

2.3 DBSCAN

We try a density based approach combining different attributes together and analysing how it affects the clustering result. First we have to establish the best value for *eps* and *min_ sample*. We choose *eps*=0.0959 based on the distance from the 3rd neighbor focusing on the elbow point of Figure 18 and *min_ sample* around 20. We try different

combinations of the two parameters until we obtain a better value of silhouette coefficient, which in the end is found to be 0.47. Due to the fact that DBSCAN is density based, it treats our data as a whole unique cluster and few noise points, as shown in Figure 19 where noise points are the purple ones.

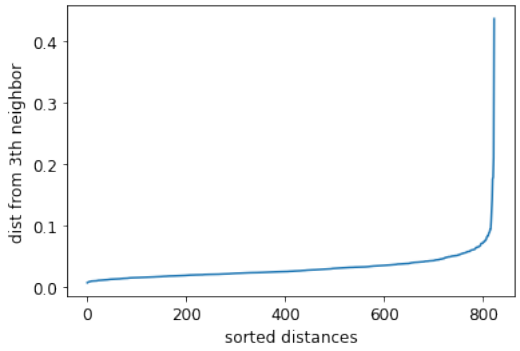

Figure 18: Graph for choosing eps parameter

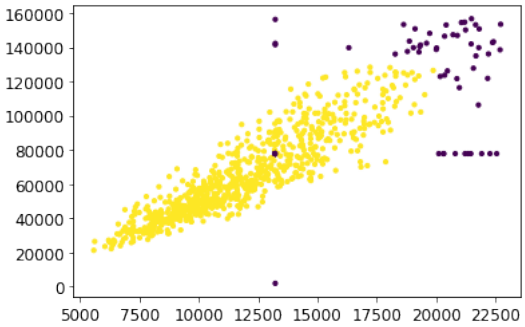

Figure 19: Clusters for DBSCAN

2.4 Clustering with pyclustering

We choose two algorithms for clustering from pyclustering: CURE and MBSAS. The first one can find also non spherical clusters and the second favors the creation of compact clusters. We use a subset of three attributes, we evaluate a proper value of K to use as parameter by using a function and by graphic visualization. The Figure 20 shows the final results of the two algorithms. MBSAS seems to define better clusters, taking into account also the third dimension.

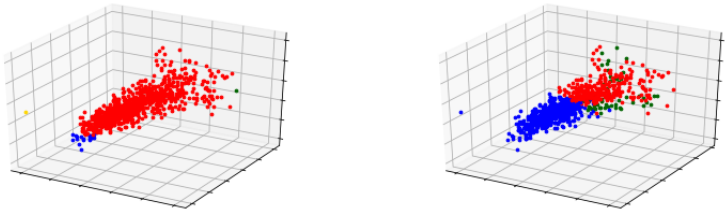

Figure 20: On the left the result of CURE with K=4 and on the right MBSAS with K=3

3 Task 3: Classification

The aim is to perform a predictive analysis in order to classify our customers in three categories: high-spending, low-spending and medium-spending. For this task, we decide to take a subset of indicators, avoiding high correlation between them:

[meanAmount_BSKT, unique_baskets, stdMean_ratio, totalAmount, totalAmount_factor, stdAmount_BSKT, maxAmount_BSKT]

We define the label using quantiles, the attribute we have chosen to build the labels on is *meanAmount_BSKT*, because it best describes the behavior of a customer since it is the average of the total amount spent per shopping session. We consider the minor elements of the first quartile as **low-spending customer**, the major elements of the second quartile are the **high-spending customer** and the rest of the elements are the **medium-spending customer**. The data distribution among the label:

- **low-spending**: 206.
- **medium-spending**: 412.
- **high-spending**: 206.

After defining the features and dropping *meanAmount_BSKT*, we split the dataset into train set and test set. To classify the data we use 6 different model: The decision tree, Random Forest, Naive Bayses, Svm, Knn and AdaBoost. For each of them we calculate the metric like the accuracy, precision, recall and F1 and we plot the **confusion matrix** and the **Roc curve**.

3.1 Decision Tree

We have a good accuracy in this model both for the training and for the test set, the decision tree is in fact one of the best models for our data. In fact, looking at the ROC curve (Figure 22) we have that the elbow is close to 1, we have some errors in the classification of the high-spending and low-spending class, as we can see in Figure 21.

Figure 21: Confusion matrix of Decision Tree

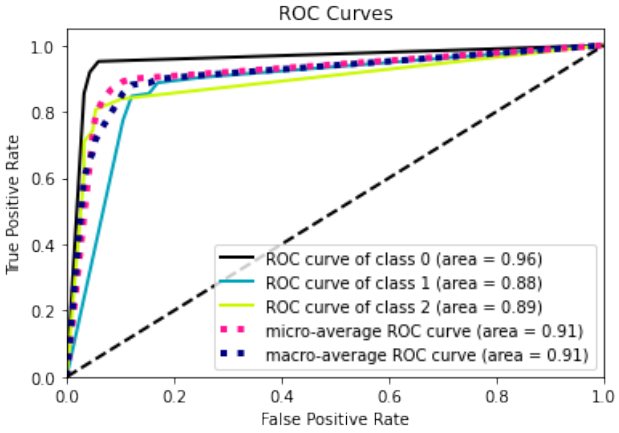

Figure 22: ROC curve of Decision Tree

3.2 Random Forest

Similarly to what happens for the decision tree, also in this case we have good results both on the test set and on the training set, in the ROC curve (Figure 24) the elbow is near 1 for the True Positive Rate and this mean that the False Negatives are very few, looking at the confusion matrix (Figure 23), we see that we have more missclassification errors for the High-spending class.

Figure 23: Confusion Matrix of Random Forest

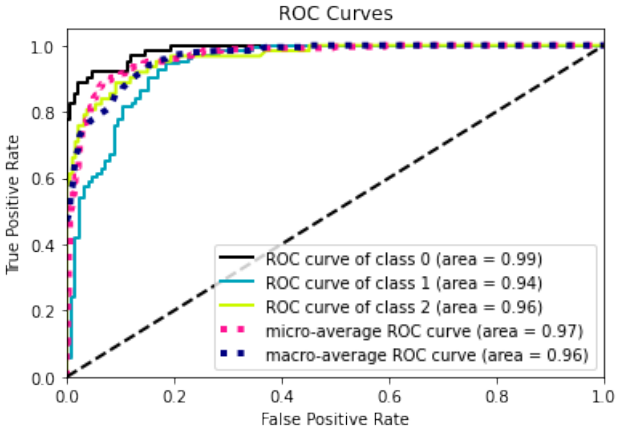

Figure 24: ROC Curve of Random Forest

3.3 SVM

SVM does not fit well with our data in fact we have a very low accuracy both in the test set and in the training set. In fact the curve is close to the center diagonal line as we can see in the ROC curve (Figure 26), above all we have the precision and the recall near to zero for both the high-spending and the low-spending class. Also the confusion matrix in Figure 25 shows very bad missclassification. This is due to SVM needing linear decision bounds that we don't have in our data, therefore it could have made some preprocessing and optimization to our data to try reaching a better classification.

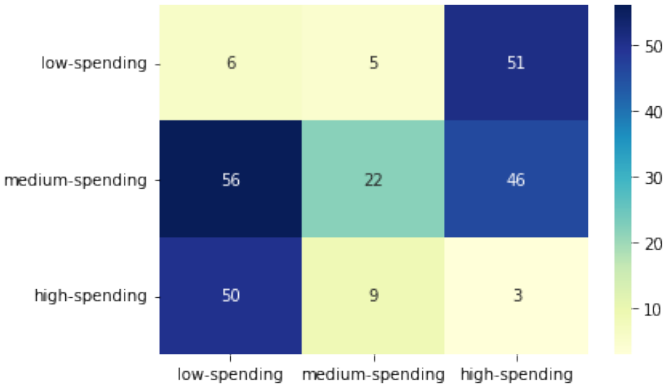

Figure 25: Confusion Matrix of SVM

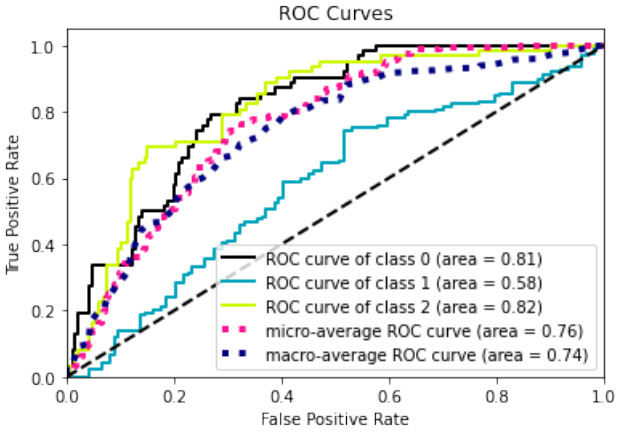

Figure 26: ROC Curve of SVM

3.4 Gaussian Naive Bayes

This model does not fit well with our data, especially for the medium-spending class, as we can see from the Confusion matrix, which shows a lot of missclassified data (Figure 27) and from the ROC curves being too low (Figure 28. This can happen because Naive Bayes needs independent probabilities that we don't have in our case.

Figure 27: Confusion matrix of Gaussian Naive Bayses

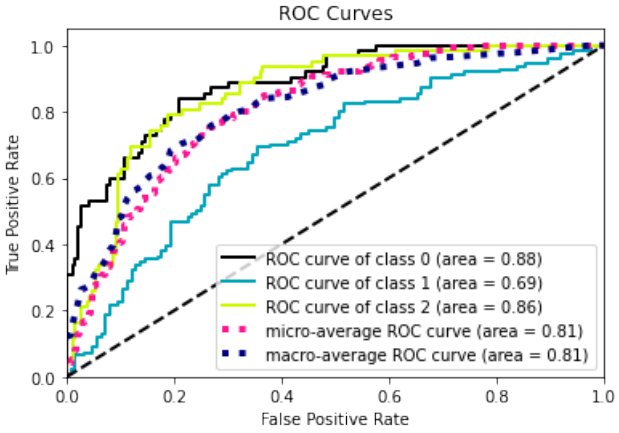

Figure 28: ROC curve of Gaussian Naive Bayses

3.5 Ada Boost

With Ada we don't have an excellent result, almost all users are classified as medium-spending (see Figure 29) and the Roc curve is anomalous (Figure 30).

Figure 29: Confusion Matrix of Ada Boost

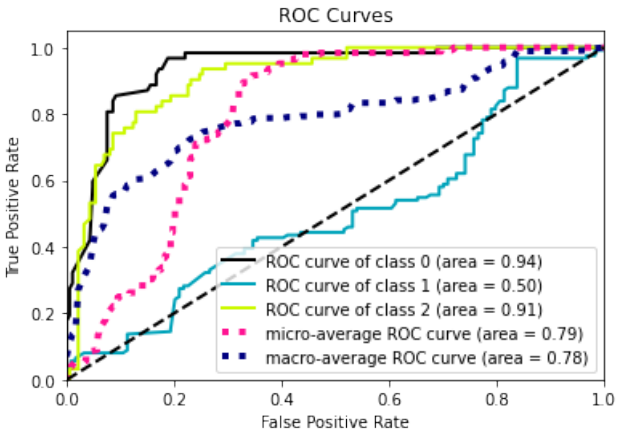

Figure 30: ROC Curve of Ada Boost

3.6 K-Nearest-Neighbors

This model does not classify our data well, we have more classification errors especially in the medium-spending class than the other two classes (see Figure 31, indeed observing the Roc curve (Figure 32) we see how the elbow referring to the medium-spending class is very close to the central diagonal.

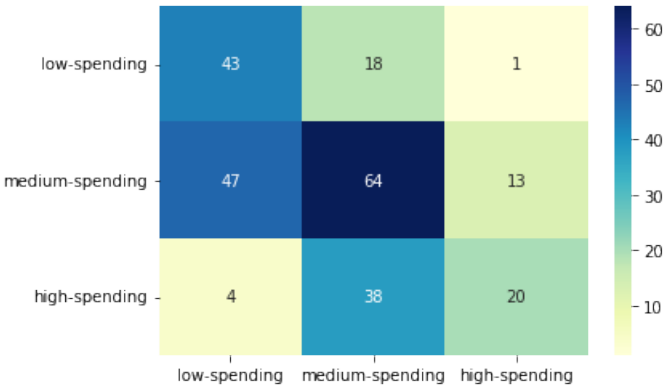

Figure 31: Confusion matrix of K-Nearest-Neighbors

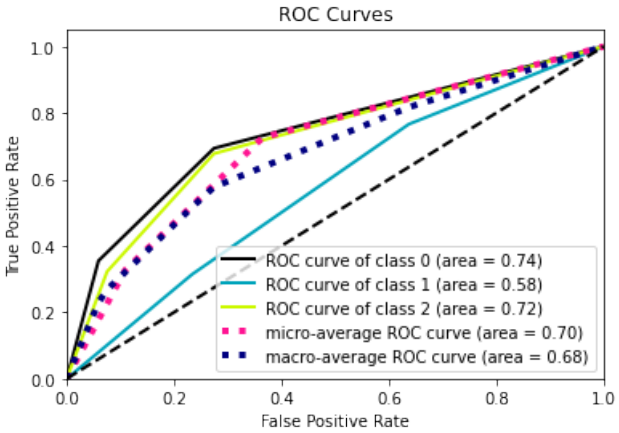

Figure 32: ROC Curve of K-Nearest-Neighbors

3.7 Result of different models

Model	Accuracy	Precision	Recall	F1
Decision Tree	0.85	0.86	0.84	0.85
Random Forest	0.88	0.89	0.86	0.87
Gaussian Naive Bayes	0.62	0.62	0.67	0.63
SVM	0.12	0.23	0.11	0.13
K-Nearest-Neighbors	0.51	0.53	0.51	0.50
Ada Boost	0.57	0.76	0.45	0.43