Customer Segmentation: A Key to Unlocking Business Growth and Success

Miguel Marques

20221839

9/06/2024

Machine Learning II

Executive Summary

Human behavior is deeply rooted in habits and routines, a truth that major supermarket chains understand well. They recognize that most people tend to do their shopping at the same place week after week and often opt for the same products. For these companies, attracting and retaining customers is crucial to ensuring the profitability of the business. In this highly competitive context, it is essential for companies to understand and effectively engage with their customer base. By utilizing advanced techniques of customer segmentation and behavior analysis, companies can develop tailored marketing strategies, adapted to meet the diverse needs and preferences of consumers. Targeted advertising to each type of customer makes a significant difference in the company's profits, which is why it is so important to correctly collect data for analysis and conduct competent analysis.

Our methodology comprised multiple phases:

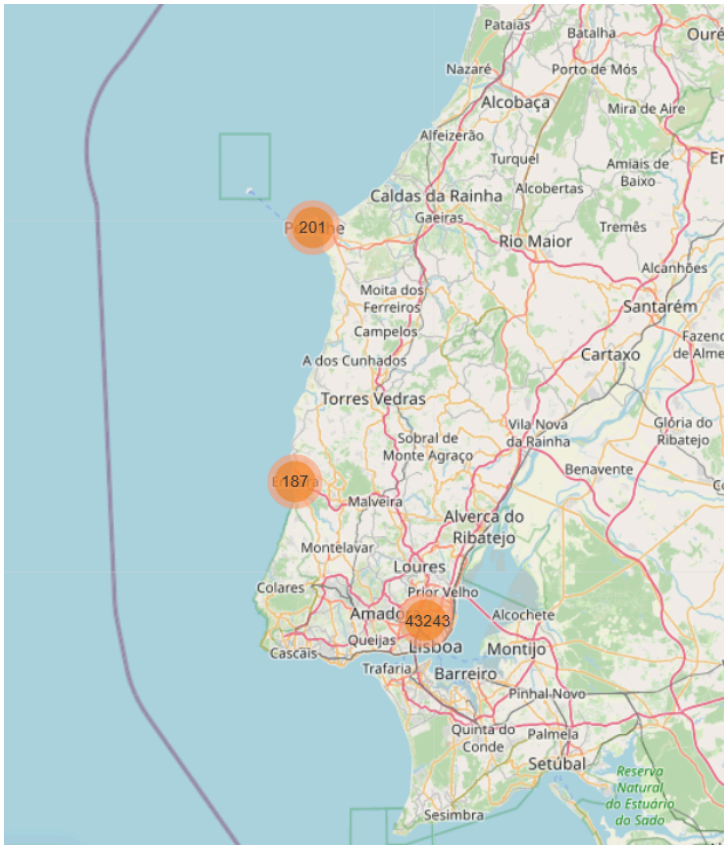
- Exploratory Data Analysis: I sought to better understand the data through visualizations
- Data Preprocessing: This phase encompassed data transformation, cleansing and missing values input
- Customer Segmentation and Clustering: I used autoencoders as a technique to reduce the dimensionality of the data and tested several clustering algorithms (K-Means, DBSCANMean Shift, Hierarchical single Cluster)

Exploratory Data Analysis

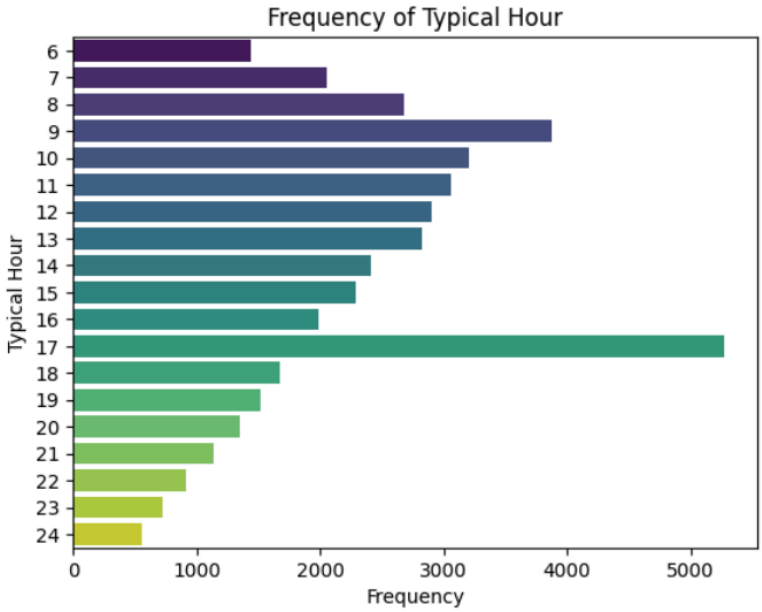
After importing the necessary libraries and datasets, I started exploring customer_info, the dataset that provides the information needed for customer segmentation. I noticed that there were missing values, but I decided to first gain some insights into the data in order to choose the best process for the imputation method.

```
kids_home                   1.200981
teens_home                  2.340079
number_complaints           1.498934
distinct_stores_visited     3.000160
typical_hour                3.999450
lifetime_spend_vegetables   2.000871
lifetime_spend_fish         3.000160
loyalty_card_number        43.494305
dtype: float64
```

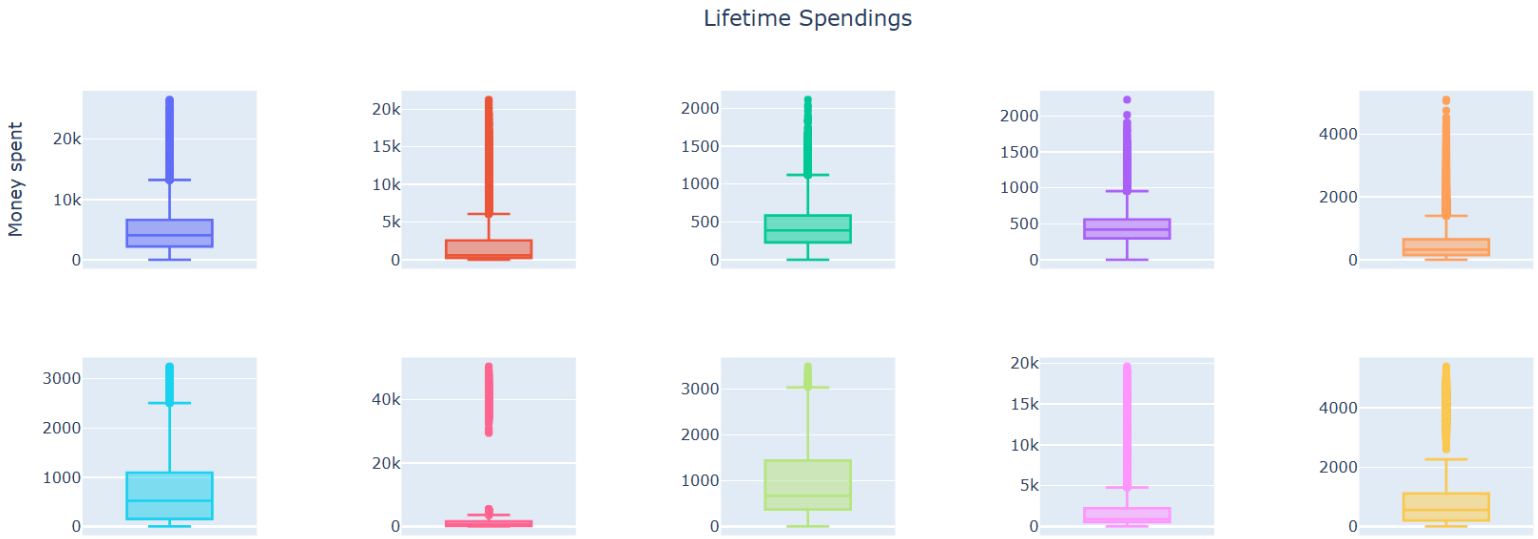
We found that the vast majority of customers live in the Lisbon metropolitan area and the rest are distributed across Peniche and Ericeira.



We check the distribution of typical hours that customers shop



I found that all lifetime_spent categories have many extreme outliers



Data cleaning and transformation

I started by creating new variables and transforming others. The search for better understanding of the data and consequently, the visualizations, will continue to be part of the process.

New variables were created:

- **Age**: Calculated using each customer's date of birth.
- **Loyalty card**: I noticed that almost half of this column contained missing values, not due to data collection failures, but because it refers to the identification number of each customer's card. Since this number does not provide any additional information, I decided to remove it and instead use whether the customer is registered or not.
- **Years since first purchase**: This provides a more intuitive measure of potential customer loyalty.

When dealing with outliers, it is crucial to consider the nature of the data and its potential impact on the analysis. In many cases, removing outliers can result in the loss of valuable information and distort the results. For example, by imposing an upper limit on the lifetime_spent variables, we would be ignoring the fact that certain types of customers may spend significantly more than the median. It is precisely these insights that are most crucial to the analysis. We need this variability and distinction to be able to better segment customers.

Therefore, I decided to just limit the ceiling of the values of the following variables:

kids_home		teens_home		distinct_stores_visited		number_complaints	
0.0	12647	0.0	15587	1.0	25338	0.0	15591
1.0	23779	1.0	20585	2.0	10093	1.0	24908
2.0	2543	2.0	4255	3.0	4000	2.0	2277
3.0	1393	3.0	1748	4.0	1761	3.0	75
4.0	1186	4.0	389	5.0	752	4.0	51
5.0	867	5.0	45	6.0	304	5.0	40
6.0	439	6.0	1	7.0	68	6.0	18
7.0	185			8.0	6	7.0	7
8.0	54					8.0	8
9.0	14					9.0	2

All were limited to 4 except number_complaints which was limited to 3. In other words, 3 can be interpreted as "3+"

This approach was only performed on a single dataset, and I reserved another one to test its performance on other types of clustering algorithms. For example, in density-based clustering algorithms, such as DBSCAN, outliers do not necessarily need to be removed before running the algorithm because these types of algorithms focus on the local density of data points, rather than relying exclusively on the Euclidean distance between them.

In other words, points that are isolated and have low local density can be identified as outliers or noise by the algorithm, without the need to remove them beforehand.

In summary, we are left with 2 datasets.

Two variables were removed from the model:

- **customer_name:** a person's name has no relevance to this analysis
- **lifetime_spend_videogames**: high correlation was verified with other variables in the model(image below)

I decided to remove the variable 'lifetime_spent_videogames' after imputing the missing values. I conducted the step of checking the correlations only with the data already imputed, as those are the ones that feed into the model. Despite the small percentage of missing values and their minor impact on the model, I chose to check the correlations before imputation. The expected outcome was confirmed, and thus the variable was removed both before and after imputation, maintaining consistency

Missing values

```
percentage_missing_values.sort_values()
```

```
kids_home                    1.200981
number_complaints            1.498934
lifetime_spend_vegetables    2.000871
teens_home                   2.340079
distinct_stores_visited      3.000160
lifetime_spend_fish          3.000160
typical_hour                 3.999450
loyalty_card_number         43.494305
dtype: float64
```

percentage of missing values in each column

In the previous step, I had defined 2 datasets:

- data: the one to which the limitations were applied to the variables (clip upper)
- data2 :No outlier removal

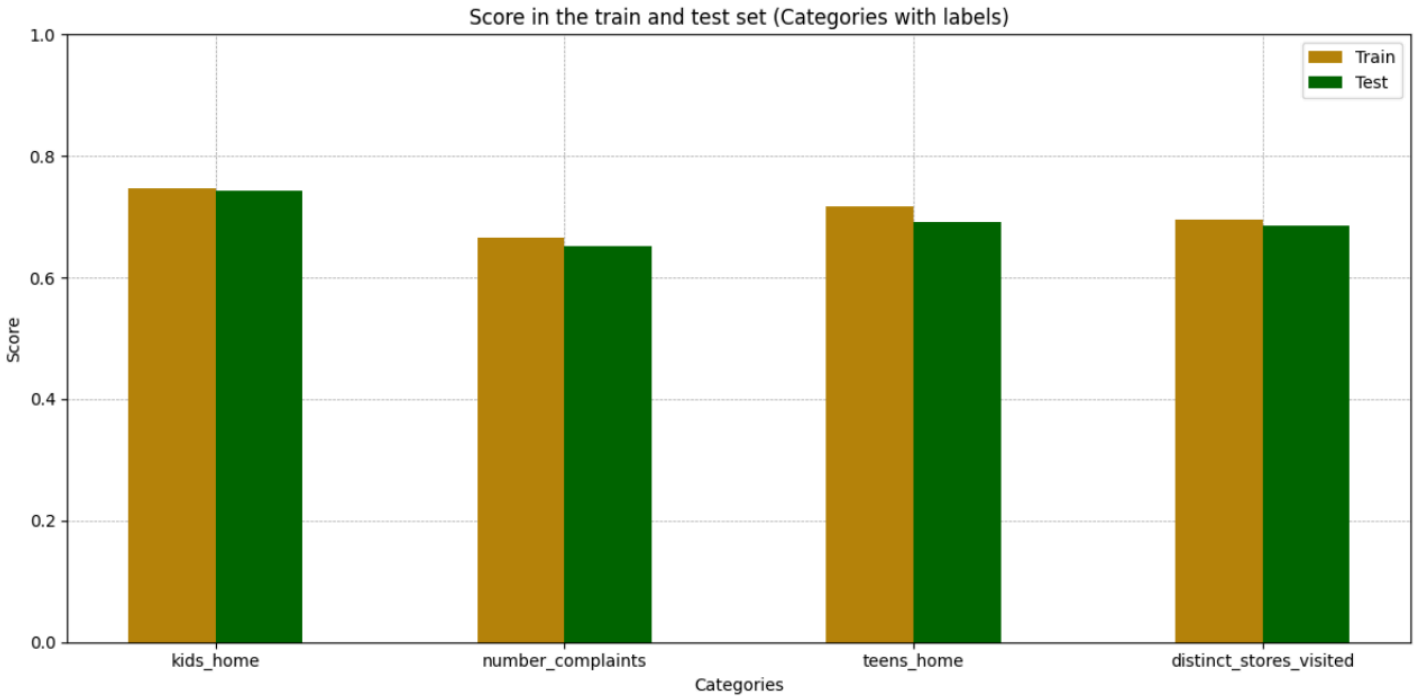
Each column with missing values will be imputed sequentially based on the percentage of missing values it has. In other words, columns with fewer missing values will be imputed first.

In the first iteration, the algorithm will be trained on all data that does not contain missing values and will fill in the missing values for kids_home. In the next iteration, the data that contained missing values only in the previously imputed column (kids_home) will be used to train the algorithm for the next variable.

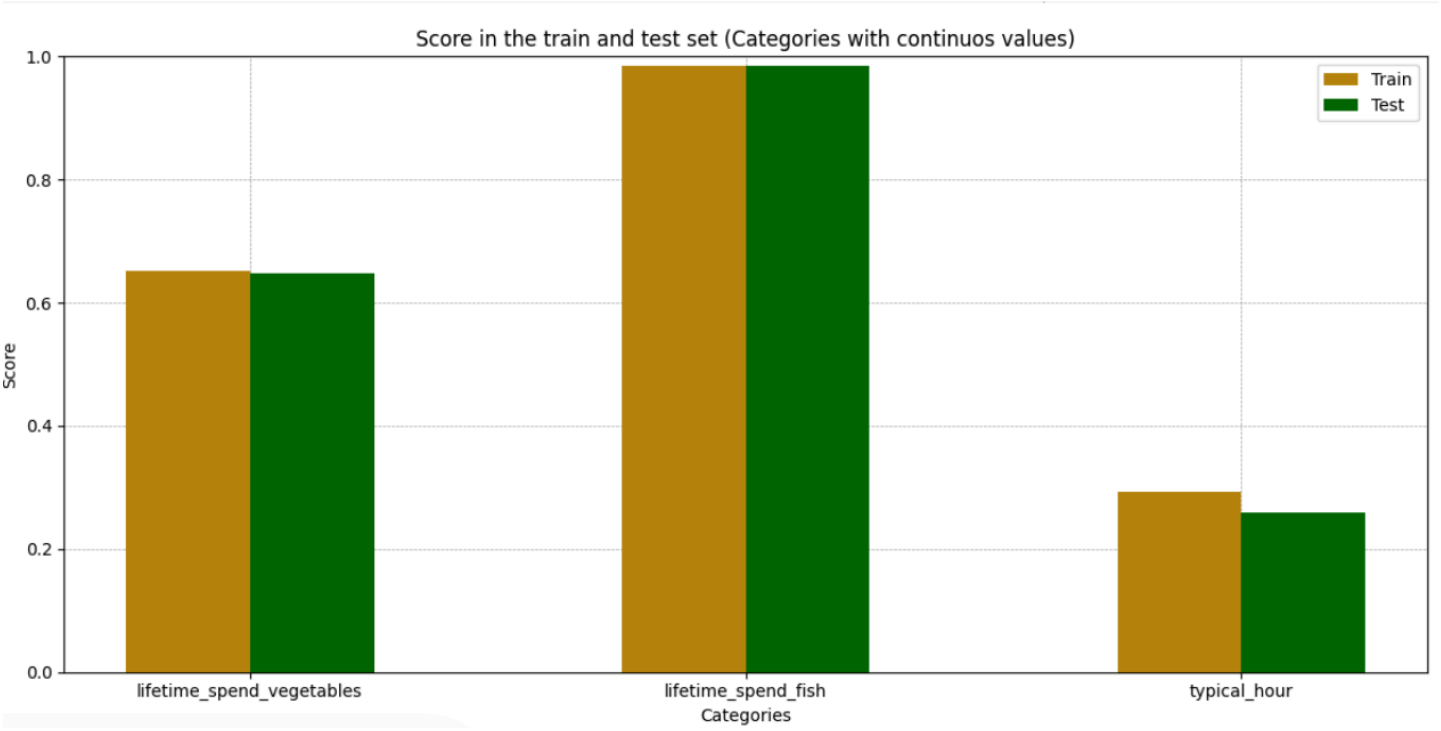
This strategy aims to optimize imputation in the sense that the algorithm is trained with the maximum amount of information available at each step. In addition, following an order from the smallest to the largest number of missing values is preferable, as it allows the imputed values to be trained based on a smaller amount of missing data, which can lead to more accurate imputation. If we did the opposite, with columns with more missing values imputed first, we would have more imputed values (whose veracity we do not know) being used to train the imputation model, which could lead to less reliable results.

The variables were divided into class data ('kids_home'', number_complaints', 'teens_home', 'distinct_stores_visited') and continuous data ('lifetime_spend_vegetables', 'lifetime_spend_fish', typical_hour'). The algorithm used to predict the data is different. In the first group, a neural network was used, trained in 50 epochs, which presented very good accuracy results that will be presented later. I researched about the best loss function to use and I came to the conclusion that 'sparse_categorical_crossentropy' was the most suitable for this type of work because it handles multiclass data quite well. Since 'distinct_stores_visited' did not have any labels as input equal to zero (values start from 1), I decided to take the approach of subtracting 1 from all values in the column. In the end, after imputation, I added 1 to all values again to compensate. The second group was imputed with the KNeighborsRegressor algorithm that was optimized for each column with a GridSearch to find the best number of neighbors.

For the classed data, the accuracy was recorded and for the continuous data I recorded the r2_score. The data is represented below:



Accuracy as a scoring measure



These results left me with more doubts in the sense that I found such a perfect input in lifetime_spend fish strange and the model's performance in typical hour was not satisfactory.

So, I thought about deleting all the rows with missing values in typical hour, but it would be an unnecessary loss of information since they represent 4% of the data (the imputation with the

worst performance was precisely in the variable with the most missing values). An r2 score of 0.3, although low, I think that leaving the data like this will not impact the work as much as a complete removal of the rows would.

Feature Selection

I check the correlation between the variables and decide to remove lifetime_spent_videogames because have a high correlation with a lot of variables

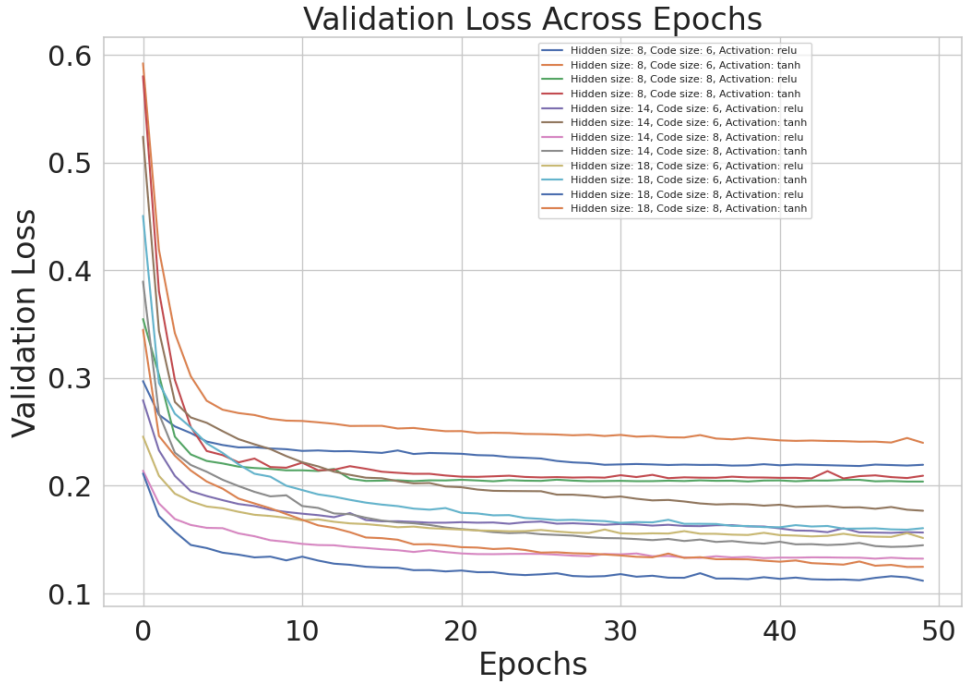
Customer Segmentation

I started by scaling both datasets with Robust Scaler. This was chosen because it is the most robust to outliers and because it preserves the shape and distribution of the original data.

Autoencoders to Dimensionality Reduction

To dimensionality reduction, I decide to use autoencoders.

In order to find the best parameters for the layers, I implemented a kind of GridSearch, testing all the parameters and trying to minimize the loss.



The best parameters found were:

- hidden_sizes = 18
- code_size = 8
- activations = ‘relu’

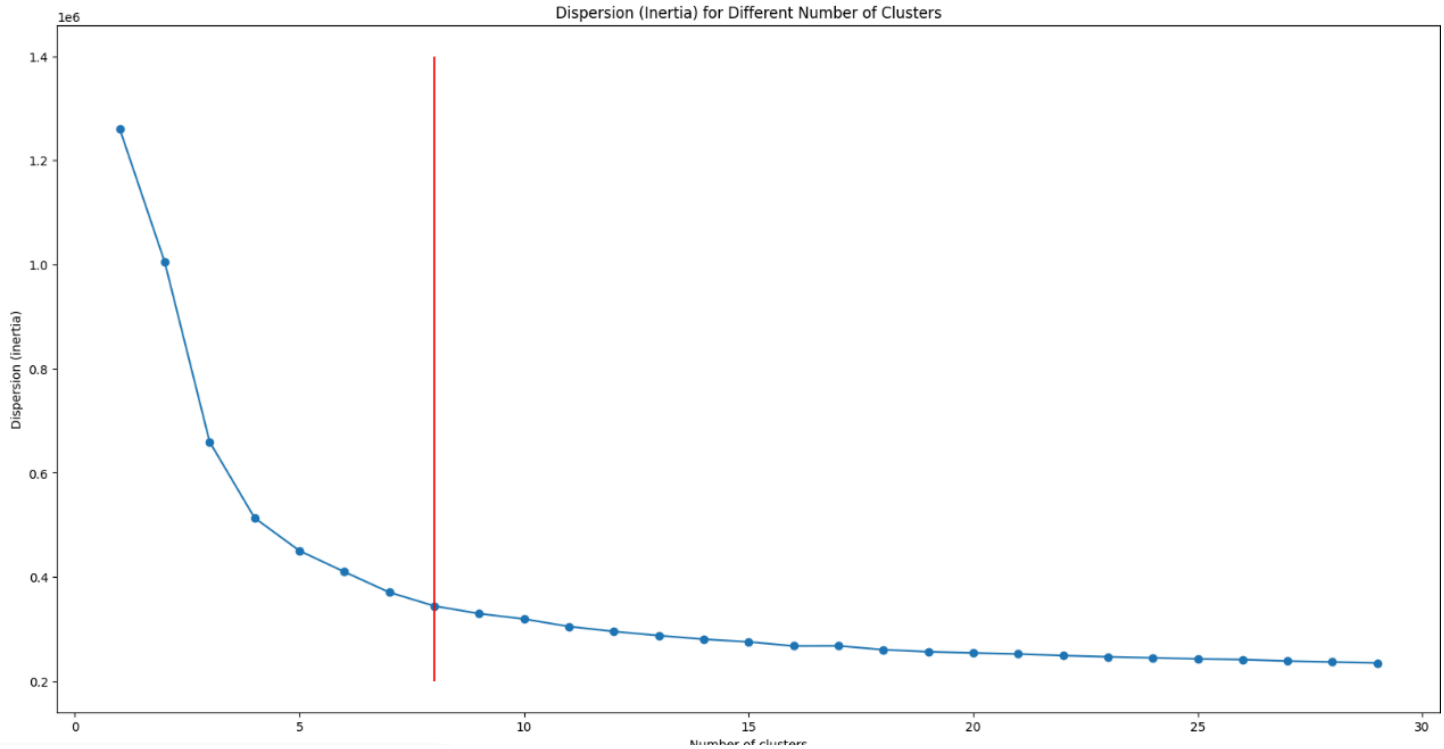
I autoencoder my data and saved my dataset to use in clustering algorithms.

Clustering Algorithms

I have tried different clustering algorithms. At the beginning of each different algorithm, I copy data and data2 so as not to touch the original dataset and to be able to record the clusters in a column of the dataset.

K-means

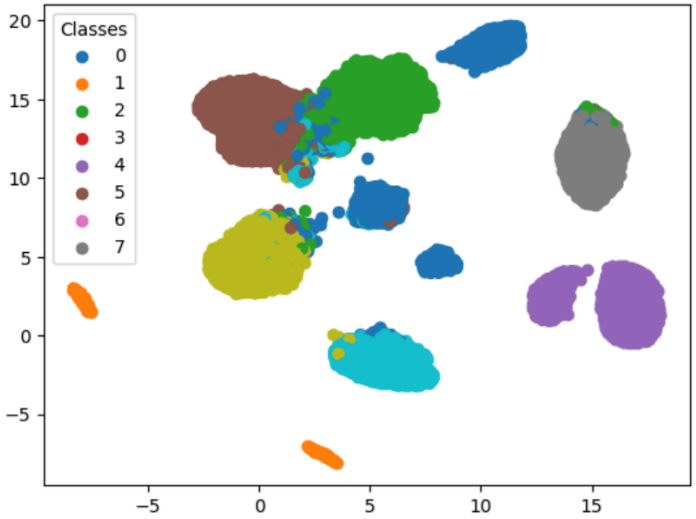
I started by experimenting different values for the number of clusters and saving the inertia for each one of them in a dictionary. Then I plotted the graph below:



In this graphic I looked for the elbow, that is, the point at which the inertia no longer decreases so much that it justifies increasing a cluster. Especially in this work where I am prioritizing interpretability, I think that an adequate number of clusters would be between 7 and 9. I ran the code several times with the options mentioned above and obtained the best results with clusters = 8.

I also tested the second dataset and the results were worse (visualization of the U-MAP and the silhouette score)

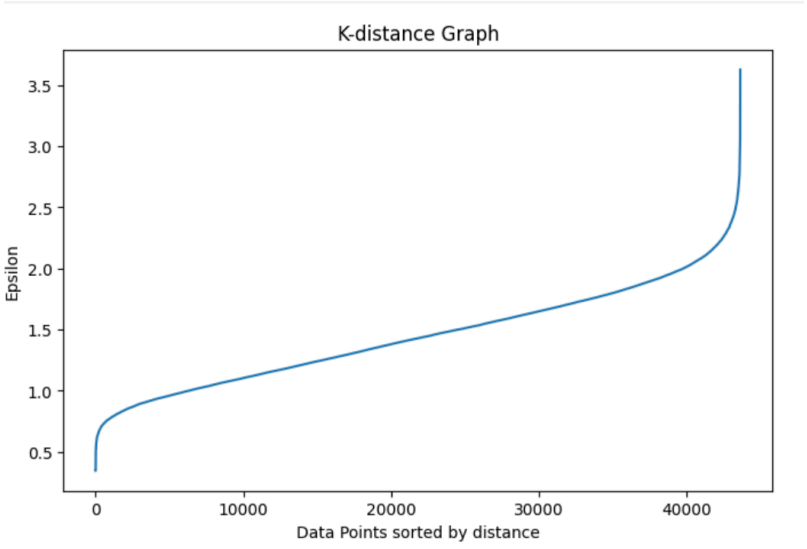
I visualized the clusters with UMAP, a technique that reduced the data to 2 dimensions and with the help of a scatter plot I colored the points according to the associated clusters. The parameters n_neighbors and min_dist were tested and ended up being configured with the values 40 and 0.5, respectively, in order to balance the preservation of the local and global structure of the data and the density of the clusters in the projection.



Overall, the clusters are well defined and separated, especially clusters 1, 4 and 7. There is some uncertainty in some regions between clusters 0, 2 and 5 but I considered it normal due to the nature of the data.

DBSCAN

DBSCAN and Mean Shift are algorithms that will be tested on both datasets. They handle outliers better than K-means and can bring more value to our model



The elbow indicates the transition between dense regions and sparse regions of the data set, where the cluster division should be made, starting from a certain epilson value (change in the curve)

Clustering did not yield good results. The datasets are associated with a single cluster or as noise. I tested with different parameters for epilson.

The points being associated with a single cluster could be related to the epilson and the min_samples being a very high value. I tested new algorithms with lower values for each of the parameters and the problem persisted. That is, instead of the data being divided into more

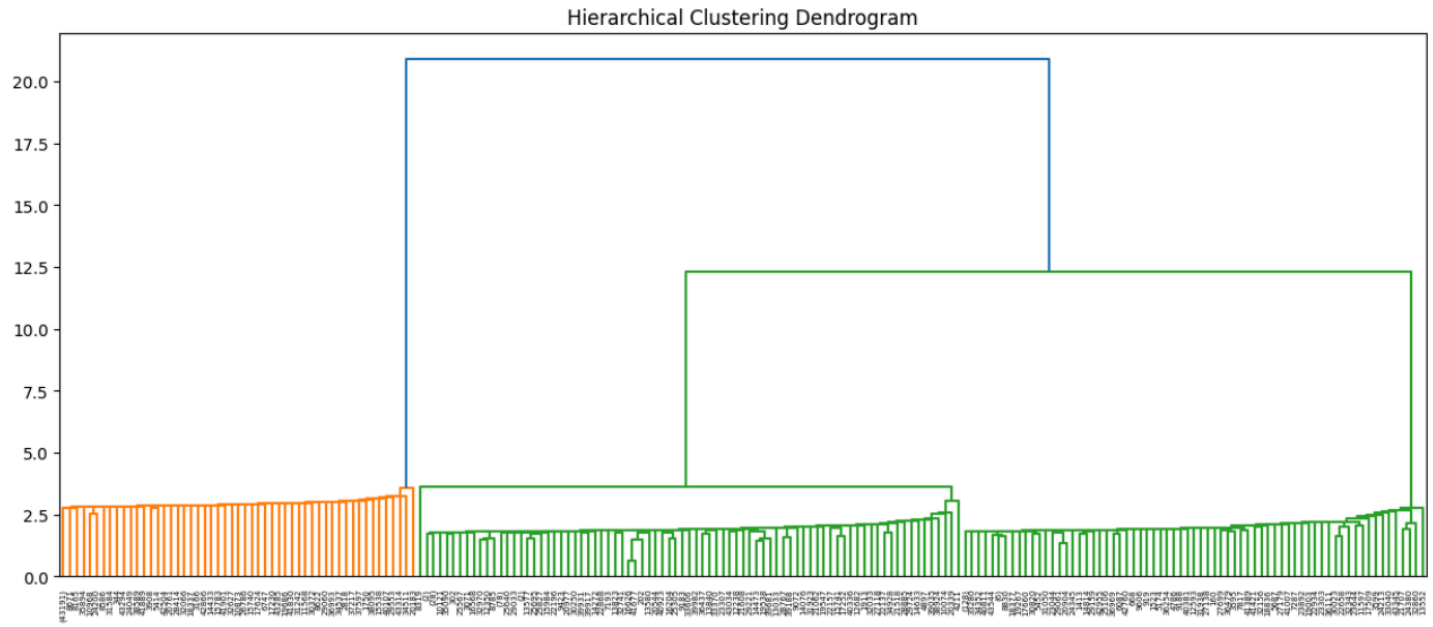
clusters, they continued to be associated with a single cluster or noise (when I obtained several clusters as a result it was because these clusters were composed of 1 or 2 observations).

I tested with the second model, where the clip upper was not made but the results were still not satisfactory.

Mean Shift

I started by estimating the bandwidth size and used this value in a first clustering. Few clusters were created and almost all the data belonged to a single one. So, I decreased the radius of the neighbor search window to obtain more clusters. Since the points continued to be associated with a single cluster, the hypothesis of using this algorithm was discarded.

Hierarchical single Cluster



This algorithm was also unable to create well-defined clusters with acceptable distributions like K-Means.

```
cluster
0    43239
1      186
2      201
3        1
4        1
5        1
6        1
7        1
dtype: int64
```

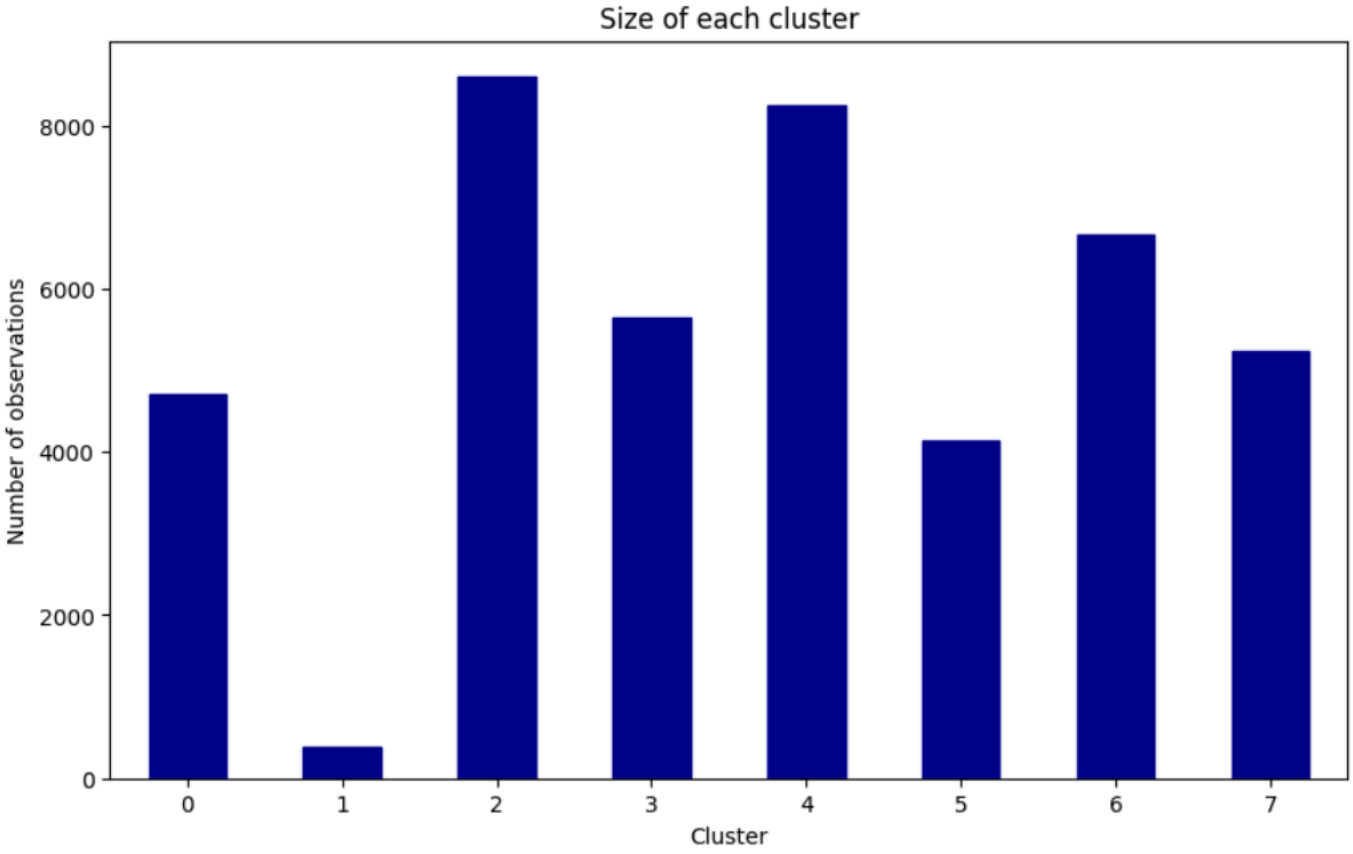
nr_clusters = 8

I try to use 8 clusters to compare the solution with the k-means solution. This was not possible, and my final solution fell back on the division into 8 k-means clusters.

Final Solution

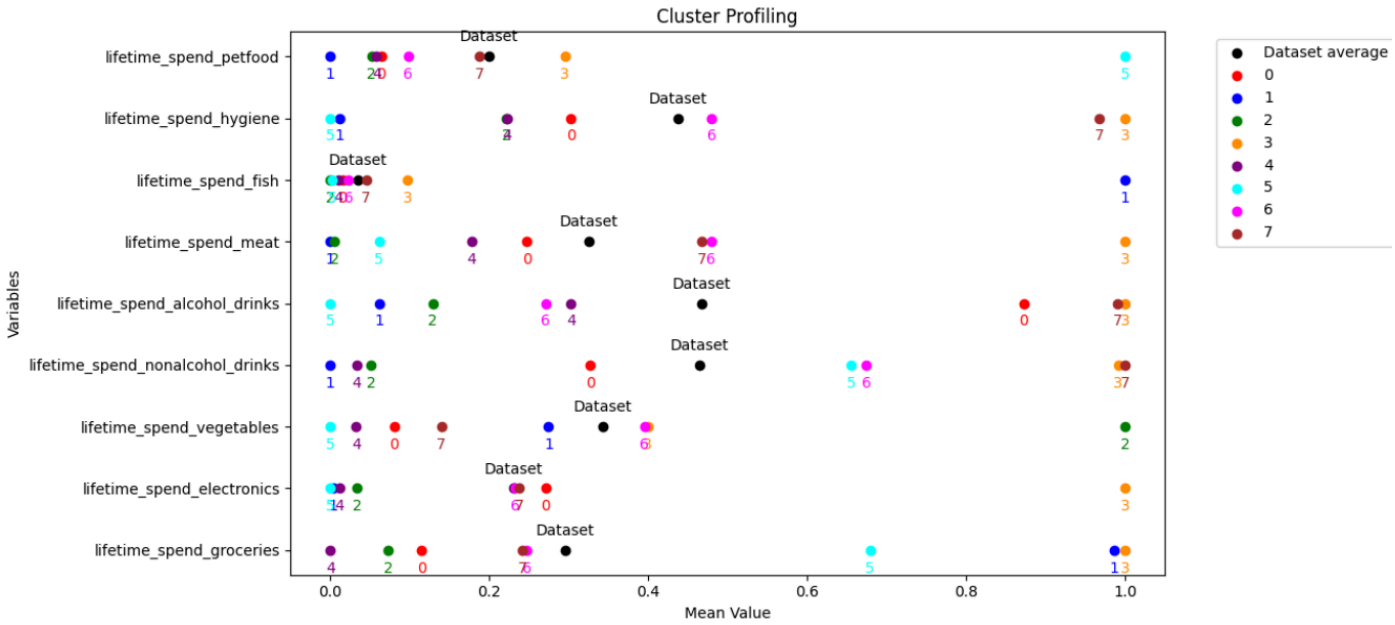
I tested the encoded data and got worse results than expected. The reduction in dimensionality made me lose several important features of the dataset even though it presented such a loss = 0.12

The final solution was found in the first algorithm, which presented a much better clustering than the other algorithms.

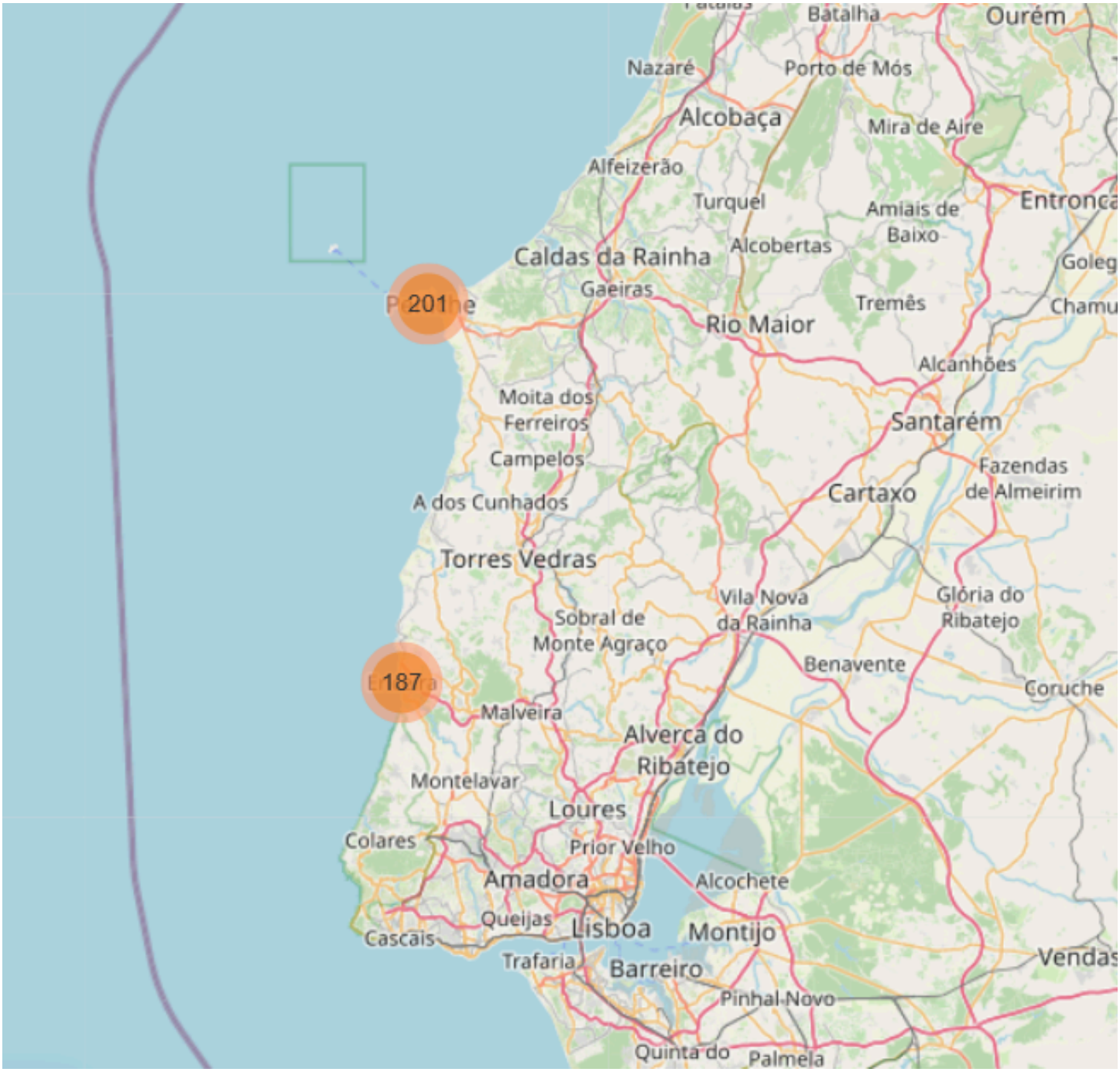


The clusters are similar in size except for the first one.

I proceeded to analyze each cluster, how the mean values in each variable were distributed in relation to the mean of each dataset.



I also check the location of each cluster and I found that the cluster 1 is exactly the customers that doesn´t belong to the Lisboa region.



Summary of the identified customer segments:

Cluster0:Low number of kids and teens. Low average age (young).Higher spending on electronics. **young people**

Cluster1: Higher spending on fish and groceries. Located in Peniche and Ericeira. Low number of kids and teens. Little variety of products purchased (possibly due to the stores being smaller). **Ericeira and Peniche customers**

Cluster2: maiores gastos em vegetables.They present values similar to the date average in the other variables. **Median customers**

Cluster 3: Higher spending relative to other groups and not interested in promotions. **Big spenders**

Cluster4:low spenders(spend less than other groups in all categories), more distinct stores visited and take advantage of promotions. **Advantage-takers**

Cluster5:They spend large amounts of money on pet food and grocers. No children. More complaints. No loyalty card. Little variety of products purchased. **Pet owners**

Cluster6:Spending relatively close to average in all categories. High number of children. More complaints. Typical hour 17. **Family parents**

Cluster7:higher spending on hygiene and in the 2 drinks categories. Distinct stores visited high. Typical hour 9. Greater number of different products purchased. **Early risers**

Basket analysis

I started the analysis of the dataset by checking if there were repeated values in invoice_id, supposedly a column with unique values corresponding to each transaction. There are repeated values in the dataset, with the same invoice but with different customer_ids. I don't know which customer made each purchase. It would be a mistake to remove one of the two rows. I could remove both, but I would be putting the possibility of losing important information. I'm going to keep both.

After separating each customer's purchases, I used TransactionEncoder to input the customer purchase data into the apriori algorithm. I used a support value of 8% (typically a value between 3% to 8% depending on the dataset) and a minimum confidence value of 0.7. For most clusters, I retained several purchase patterns. I ordered the values so that values with the highest confidence appeared at the top of the list, making it easier to analyze the most important patterns.

Targeted Promotions and and suggested campaigns:

Cluster0: Promotions: Buy cooking oil and you have 20% discount on oil
If you buy gums, you have 10 % discount on cooking oil
Buy 2 oils and get 1 one free
5 euros discount coupon on the first purchase with the card (we need to to retain these young people)

Cluster1: Promotions: 10% discount on this products:cider, beer and withe wine

Buy 2 oils and get 1 one free

Try to implement a greater variety of fishing products in stores

Cluster2:10% on vegetables and groceries for purchases over €50, 20% for purchases over €100.Loyalty card with offers (customers already know the stores and a wide variety of products, they just have to spend more by making all the purchases they need in the store)

Cluster3:Champagne offer with purchases over €200(are not attracted by promotions and are

already big spenders;eventually, a points system to maintain customer interest in the store)

Cluster4:By buying 2+ units of cake, cooking oil and candy bars you get a 30% discount in Oil

Loyalty card and welcome coupon offer. Promotions on different products and points system

Cluster5:There are no purchases for this cluster.

10% discount in any of the categories other than petfood and groceries

Cluster6: 10% discount in the whole basket

Coupon offering system as an attempt to mitigate each complaint (only receive coupons if have a loyalty card)

Cluster7:If you buy 3 packs of babies food you get one free

Points system that applies to all stores

Conclusion

I believe I achieved good results in this study, demonstrating the effectiveness of clustering in differentiating customers in multiple aspects and in personalizing marketing strategies for each segment. I believe I did a good job in the imputation stage of missing values in the way I maximized the use of available information for better predictions. On the other hand, despite the expectation of more expressive results with encoded data, it was a good lesson to understand that, even with a low loss value, the dataset was unable to capture all the variability

of the data. This insight highlights the importance of a flexible and adaptable approach, especially in complex data analysis projects.

I enjoyed doing this work, it allowed me to better understand the concepts and I consider it a very practical work that resembles countless real-life situations.