Machine Learning II Project

Clustering and Customer Segmentation

Data Science Bachelors

2023/24

Professors: Fernando Bação & Ivo Bernardo

Group 18 members:

Suelen Faruk - 20221997

Vidhi Rajanikante - 20221982

Index:

Executive Summary

The purpose of this project is to make customer segmentation and identify different groups of customers with characteristics that they have in common on data of a supermarket using different machine learning unsupervised techniques.

This consists in dividing a large customer base into smaller groups based on shared characteristics that they may have.

Initially, we loaded the customer info analysis to take important information to do the customer segmentation. We did a map to visualize where our customers were located and to see the majority or minority where they are located. After that, we changed the name of the variables for better interpretation and took care of outliers and missing values. We also make charts of some variables to see their distribution and check for inconsistencies.

After all this initial preprocess part, we scaled the data with a robust scaler and implemented some clustering algorithms such as: DBSCAN, K-Means, SOM and Hierarchical Clustering. To interpret our final clusters, visualize and to see the quality of the solution we used UMAP. and we used dendrograms to better interpret hierarchical clusterings.

At the end we obtained a solution of final clusters such as: *Fake Vegetarians, Youngsters, Karens, Petters, Misers* and much more that we created to then do the association rules and be able to make promotions(Targeted Promotions) to all clusters.

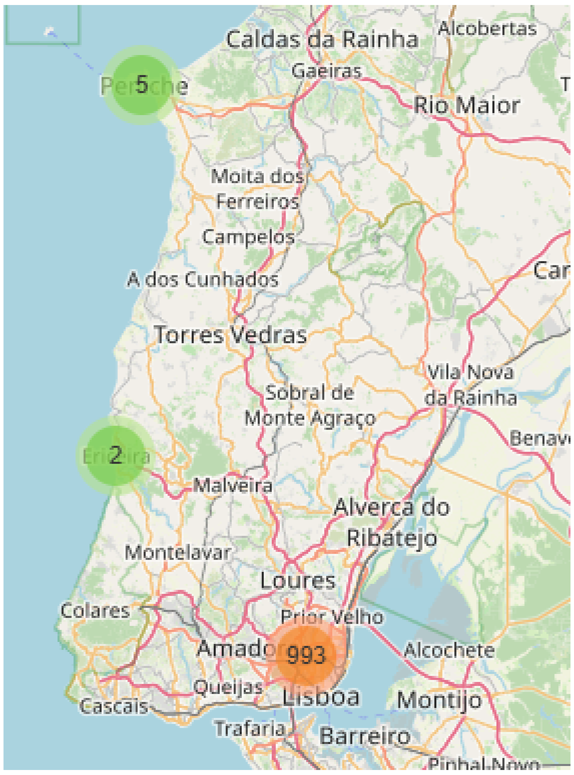
With the implementation of the unsupervised machine learning techniques referenced above the customer segmentation will be important to increase the profit of the supermarket and to see the best clients(the ones that buy more) and to take much more valuable insights that will make the supermarket understand the client desires.

Exploratory Data Analysis and Preprocessing

The exploratory data analysis that we did was divided in 6 steps: Data Preprocessing, Data Visualization, Variable Transformations, Feature Selection, Correlation Matrix Analysis and Inconsistencies.

Data Preprocessing:

In the initial phase of pre-processing the data, we created a map to visualize and locate our observations. Once this was done, we removed the word lifetime from all the variables that began with this word because there was no point in having it because it didn't say much about the variables.



We then checked for missing values and outliers. The variables with missing values were: loyalty_card_number with 18977, typical_hour with 1745 missing values, distinct_stores_visited and spend_fish with 1309 missing values, teens_home with 1021 missing values, spend_vegetables with 873 missing values, number_complaints with 654 missing values and lastly kids_home with 524 missing values. After checking the variables with missing values, we imputed them with Knn to fill in all the columns with missing values.

We used KNN Imputer because it is a method that does not risk reducing the variability of our dataset or wasting valuable data, which could happen if we treated missing values with the mean, mode or median method.

The customer_gender, customer_name and customer_birthdate variables are the only ones that are objects. We dropped the loyalty_card_number variable because there was no need to know which customers do or don't have this card.

Having done this, we identified the variables with outliers, which are kids_home, number_complaints, teens_home, distinct_stores_visited, spend_groceries, spend_electronics, spend_vegetables, spend_alcohol_drinks, spend_meat,spend_fish, spend_hygiene, spend_videogames, spend_petfood, total_distinct_products, percentage_of_products_bought_promotione year_first_transaction, in other words, almost all of them have outliers.

For the treatment of outliers, we only treated those outliers whose sum gave a value close to 7333 outliers. We didn't treat all the outliers because they could have a huge impact on our observations and change them. To get the sum of the outliers that we wanted to treat we created a column called outlier and give value 1 to the rows that had outliers and we sum those rows.

Variable Transformations

Once this was done, we filtered the customer_name column to create the customer_graduate column to see the customers who had an academic level and filled it in with the word Nal which stands for 'No academic Level' for those who don't have a master's degree, bachelor's degree or doctorate.

Next, we transformed the variable customer_birthdate to datetime and filtered out the year from that variable because the date and time of birth are not relevant information for us.

Then we transformed all the numeric variables to integers because it doesn't make sense to have decimal numbers for these variables and then we removed the latitude and longitude variables because the necessary information had already been taken from that column.

Feature Selection

After removing the variables from the matrix correlation that showed a high correlation between them, we didn't get good clustering results and so we decided not to remove any variables because of the clustering results.

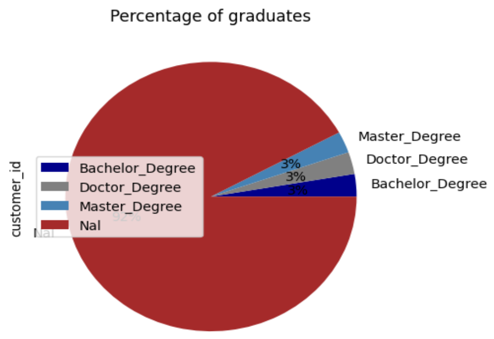
Correlation Matrix Analysis

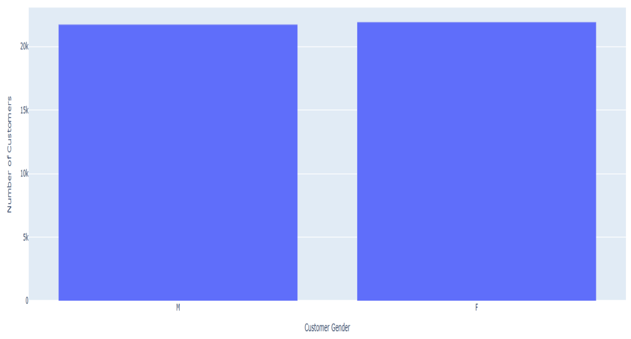
Initially, we ran the correlation matrix with the aim of making a feature selection of the variables with the correlation matrix. After seeing that the selection of variables gave poor clustering results, we decided to keep the correlation matrix to see the variables that have a high correlation with each other.

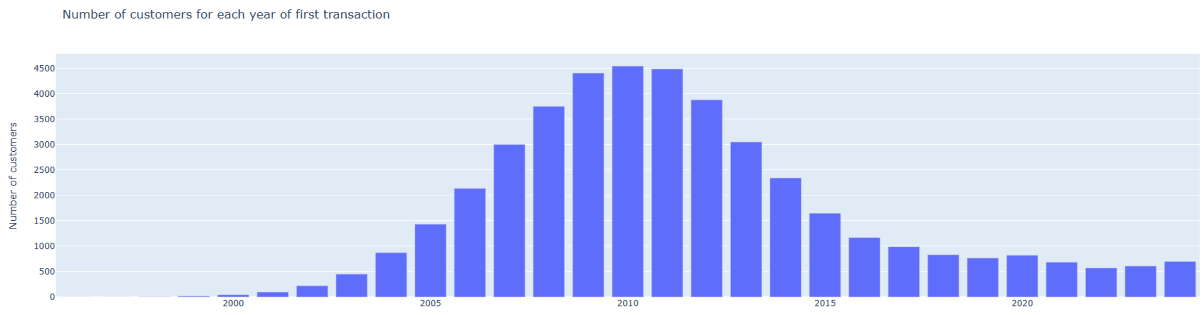
Data Visualization

To draw conclusions from our observations, we made some visualizations:

For the case of the customer_graduate variable through a pieplot we were able to observe that the percentage of students with an academic level is equal, and customers who do not have an academic level represent the majority of the population.







With a bar plot we can also see that the number of female and male clients shows a very small difference. With a histogram we can see that most of our clients became our clients in 2010.

And with bar graphs we saw that most of our clients have a child or a teenager at home or have neither a child nor a teenager.

And with another bar plot we saw that most of our customers have only one complaint.

Scaling

The scaler that was used to perform the clustering algorithms was the robust scaler because some variables were not treated due to the fact that the number of outliers in all the variables corresponded to more than 7% of all the observations and for these types of variables the best scaler is the robust scaler, which is why we had better results with this scaler compared to the others.

Initially, all the scalers were used to check the clustering results and see which one fruited the best one. From the results obtained, the results of MinMaxScaler were very poor, whereas the results of both StandardScaler and RobustScaler were similar yet RobustScaler's results seemed to be better. Thus the RobustScaler was chosen to be our main scaler also because it deals better with the presence of outliers.

After the scaling process, two pickle files were then exported, the preprocessed original dataset namely *file1* and the scaled dataset namely *f1_rb*. These were then imported in the *ClusteringAlgorithms* notebook for customer segmentation and clustering.

Customer Segmentation and Clustering

Before implementing the algorithms in the notebook, after all the importations, two functions are included to aid with the dendrograms and UMAPs plotting. These were not defined by us, *visualize_dimensionality_reduction* was provided to us and *plot_dendrogram* was done by Yehor which he provided us with.

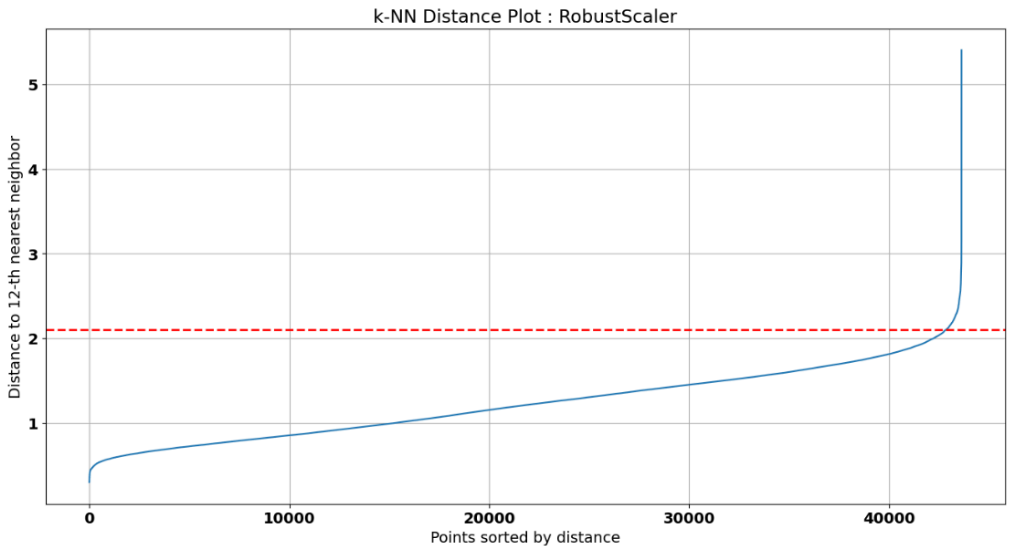
Clustering Algorithms

By implementing clustering algorithms, namely DBSCAN, KMeans, Hierarchical Clustering and SOM, on our *customer_info* dataset, we were able to achieve customer segmentation. Two 'ensemble' clustering methods were then utilized throughout the project - Tandem Approach 1: KMeans + Hierarchical Clustering and Tandem Approach 2: SOM + Hierarchical Clustering.

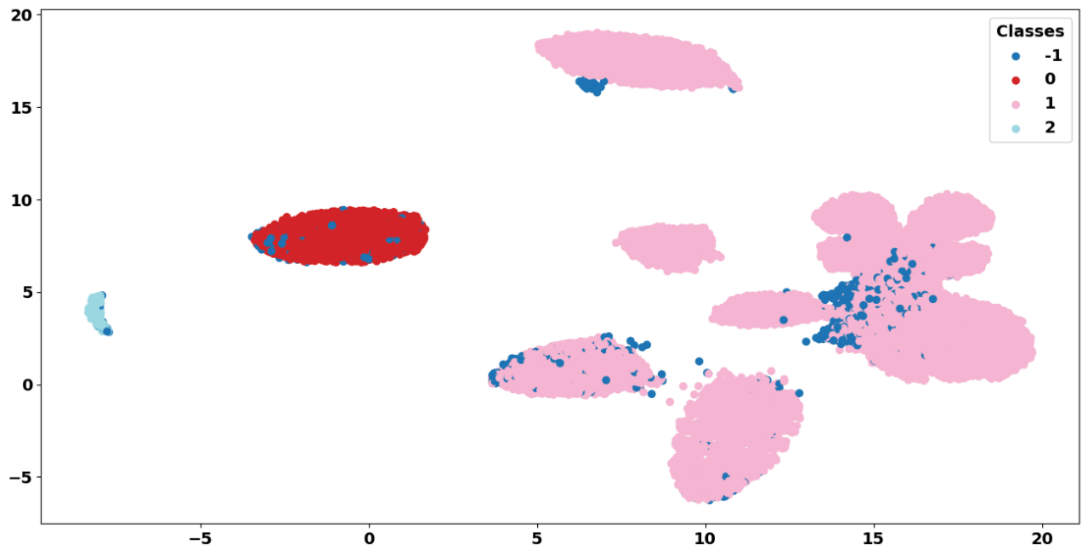
DBSCAN

Mainly chosen for the identification of multidimensional outliers and clustering purposes, the main hurdle to getting very good results was the suitable values for its hyperparameters.

The value for the *min_samples* was obtained through trial and error where the square root of the number of observations in our dataset, around 209, was first used but was not giving a good result. With this, we started with our trial and error with smaller numbers such as 25 and lower. 12 was then chosen as the value for the minimum samples as increasing it was not giving much difference whereas decreasing it was providing poorer results. A kNN-distance plot was created to check the knee point which provided us with the value for the *eps* hyperparameter.



After jotting these values into the algorithm, we obtained results which may not be satisfactory but it was the best one after a lot of trial and error. The silhouette score was then calculated which also showed that our DBSCAN clusters were of low quality as the score was below 0.5 (around 0.29).



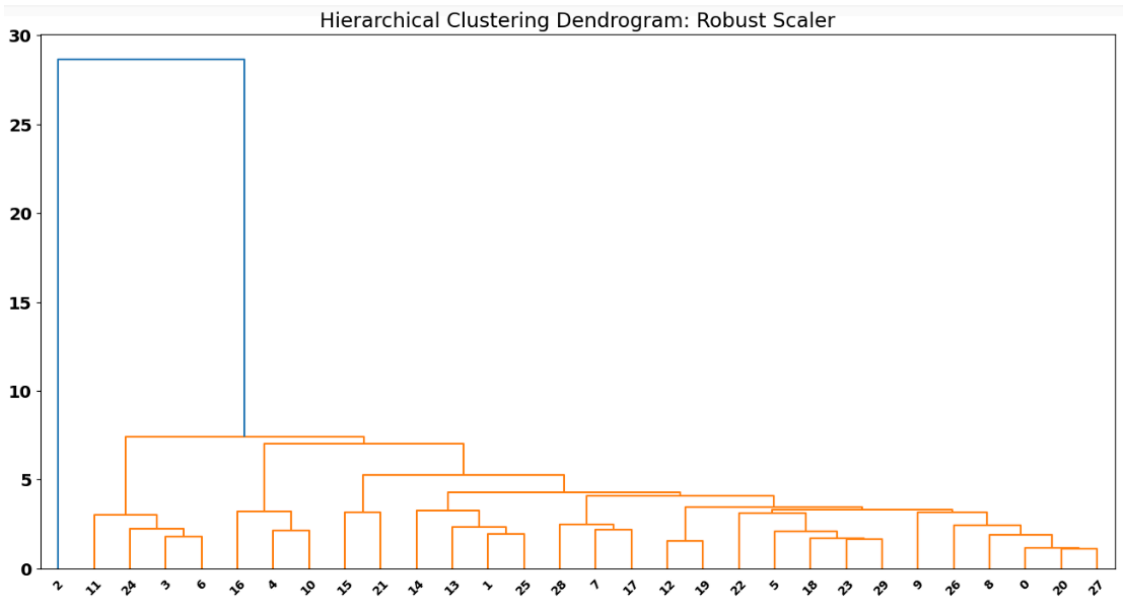
1352 points or observations were counted as noise points i.e. outliers though they were all spread out with the other points. Thus we decided not to remove or treat these outliers as they may contain important information and with this decision, we implemented this algorithm before the tandem approaches as we realized that it would only help us only with identifying the multidimensional outliers and not with clustering. Though the results, cluster numbers, were stored in the original dataset, *f1*, under *cl_db*.

Tandem Approach 1: KMeans & Hierarchical Clustering

With our first tandem approach, KMeans was first implemented to apply Hierarchical Clustering based on these results to get the final number of clusters, which were then stored in our original dataset, *f1*, under *cl_km*.

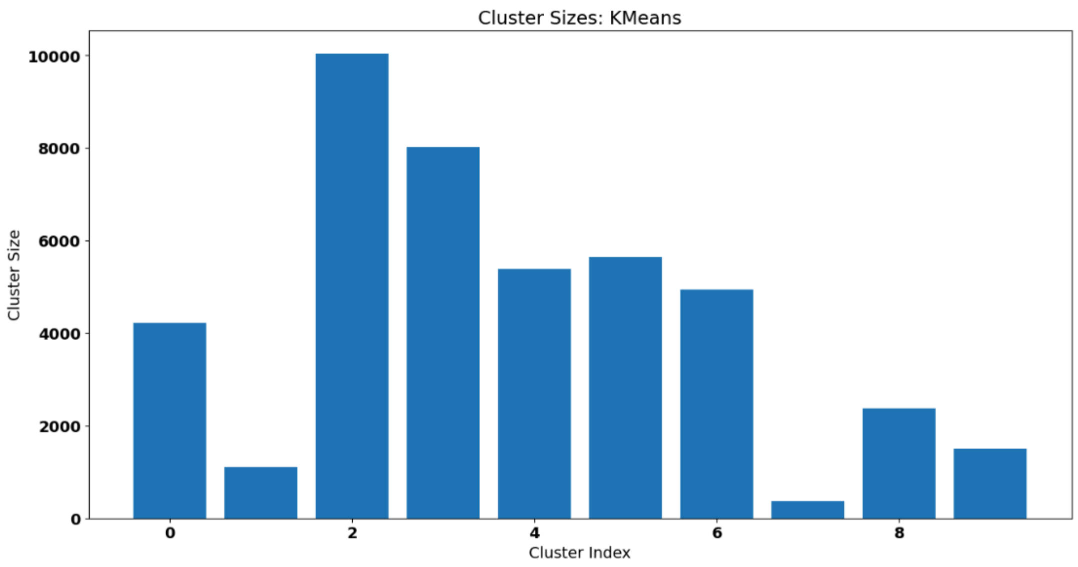
A large value for *n_clusters* was set for the KMeans with the thought that they will be filtered out when applying these results when implementing the Hierarchical Clustering. Obtaining the results from the KMeans, they were first stored in a copy of the scaled dataset and then used to fit into the *AgglomerativeClustering* and plot a dendrogram to choose the final number of clusters, which was 10 clusters - to also be able to capture the smaller clusters.

As mentioned above that the results of KMeans were stored in a copy of the scaled data, the same was done after obtaining the results from the *AgglomerativeClustering* simply for the purpose of storing the final number of clusters to the original dataset, *f1*.

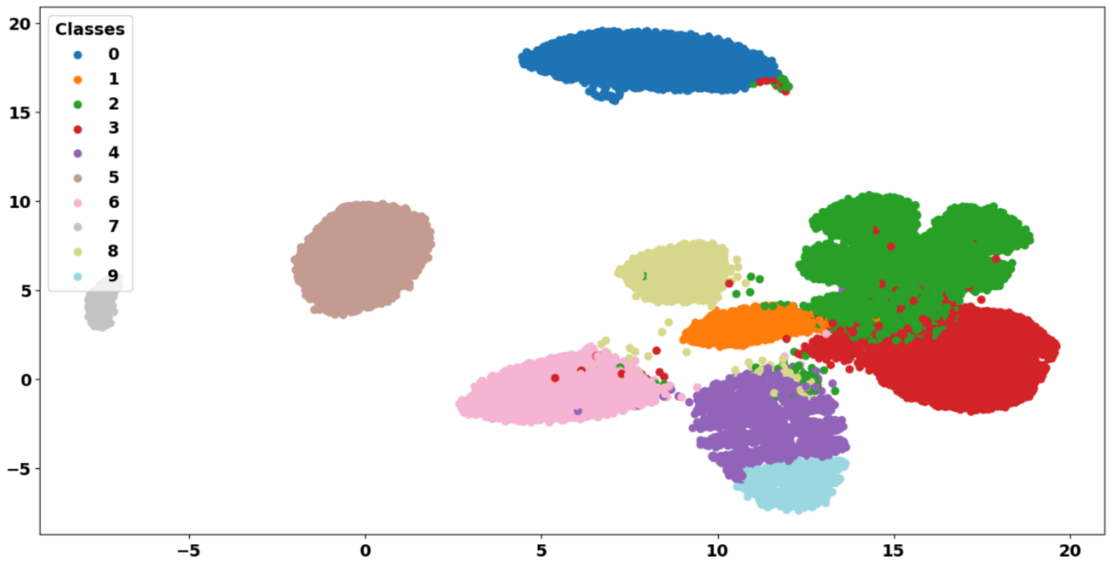


```
umber of clusters chosen:10
```

After the final number of clusters were chosen and stored appropriately, a plot representing the cluster sizes, the number of observations in each cluster, was created. This was mainly to see which clusters were the bigger ones compared to the rest.



In order to visualize the clusters, the UMAP visualization was utilized instead of the t-SNE map as it was quicker to run.

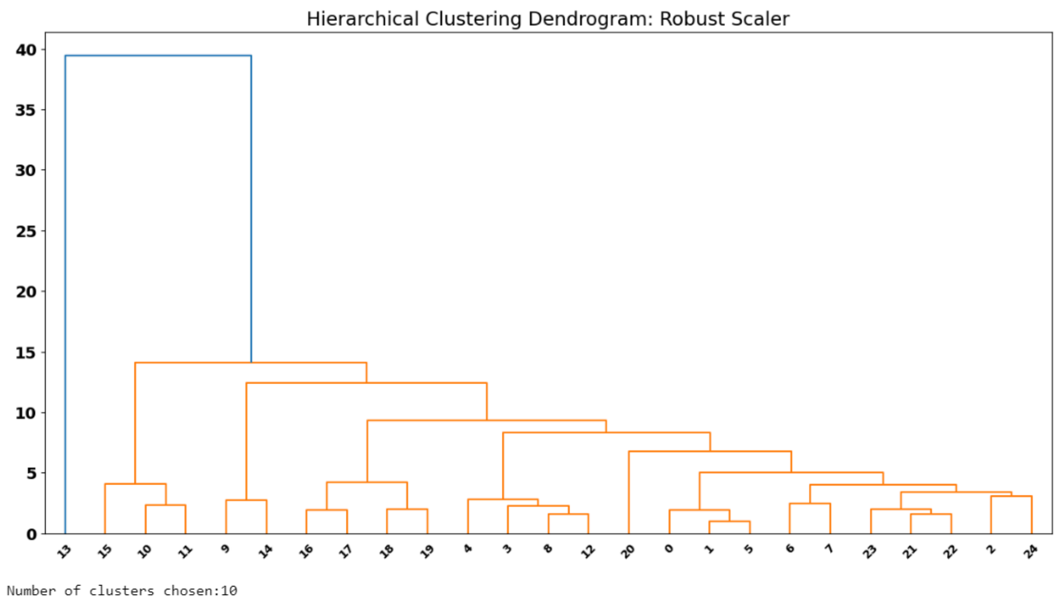


Though the visualization may not be appealing, it was the best one obtained after trial and error with the hyperparameter values. Some clusters are clearly shown such as clusters 1, 5, 7 and 9 which we can also see that clusters 1, 7, 8 and 9 are also the smallest. The rest may not be as clear but all the clusters are shown to be separated by color. We can also point out that the smaller clusters in size are well separated by color.

Tandem Approach 2: SOM & Hierarchical Clustering

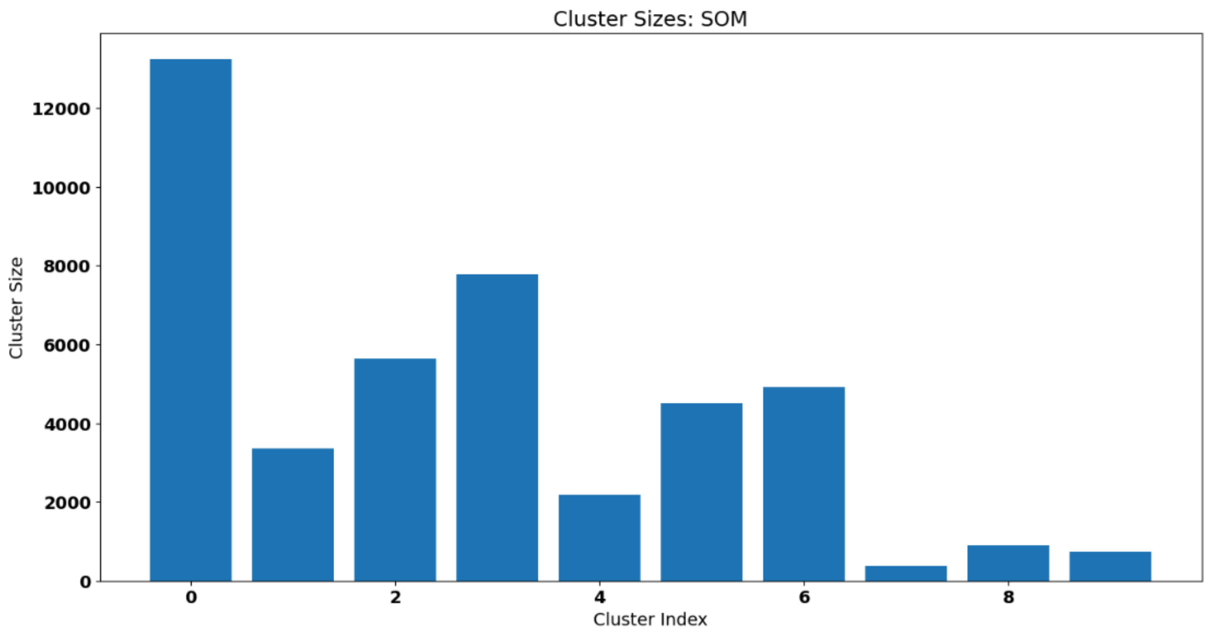
With our second tandem approach, similarly to the first tandem approach, SOM was implemented first to then apply Hierarchical Clustering based on the results we obtained from SOM to then get the final number of clusters, which were then also stored in the original dataset, *f1*, under *cl_som*.

The values of the hyperparameters *sigma* and *learning_rate* were again chosen through trial and error. SOM was trained on the *f1_rb* array with 2000 observations. After obtaining the results which was an array of the winner nodes, these were stored in a copy of *f1_rb* to then fit these into the *AgglomerativeClustering* and plot a dendrogram to choose the final number of clusters, which was also 10 clusters - with the same intention of being able to capture the smaller clusters too.

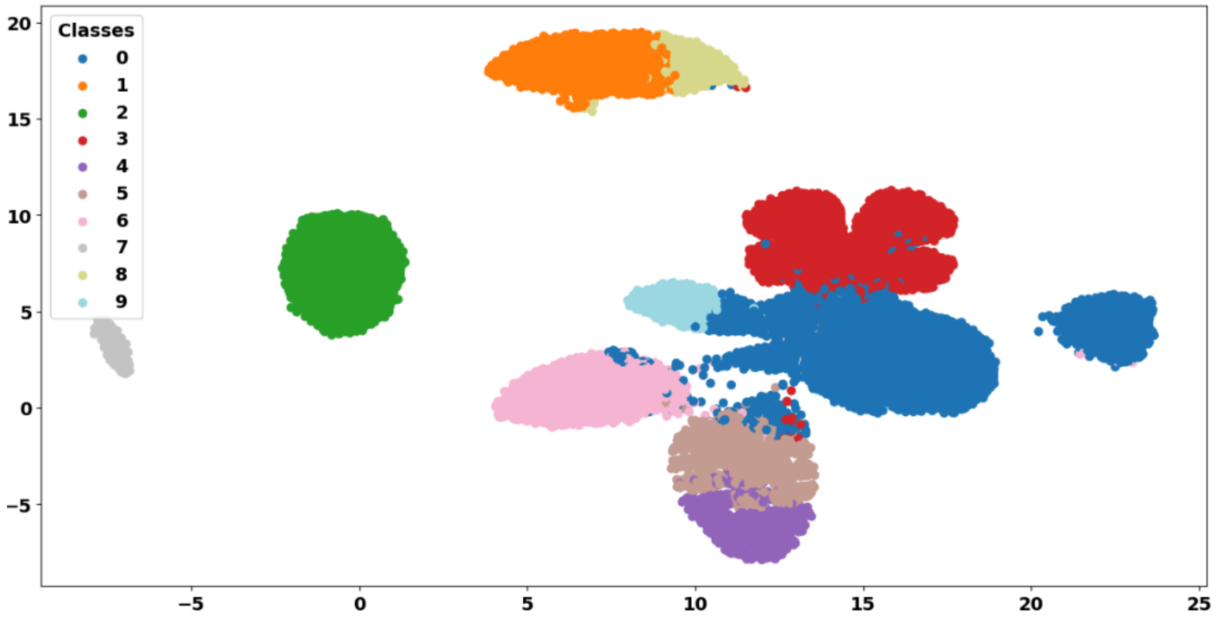


Similarly to the first tandem approach, all the results, number of clusters, were stored in copies of the scaled data, *f1_rb*, solely for the storing of the final number of clusters to the original dataset, *f1*.

Having the final number of clusters chosen and stored appropriately, another plot representing the cluster sizes, number of observations in each cluster, was created mainly to check how big or small the clusters were when compared to the rest.



A UMAP visualization was also made on the results of our second tandem approach.

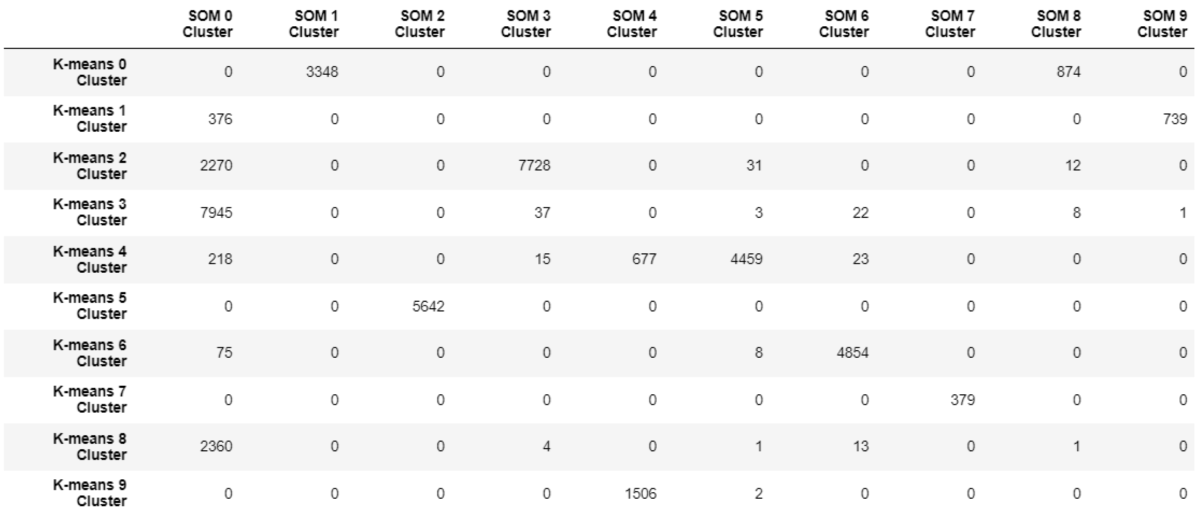


From the visualization above, we can see that we get similar results to the first tandem approach where cluster 7, being the smallest cluster, is also well defined on the visualization.

The rest of the clusters are not visualized as well as cluster 2 and 7 but they are color separated.

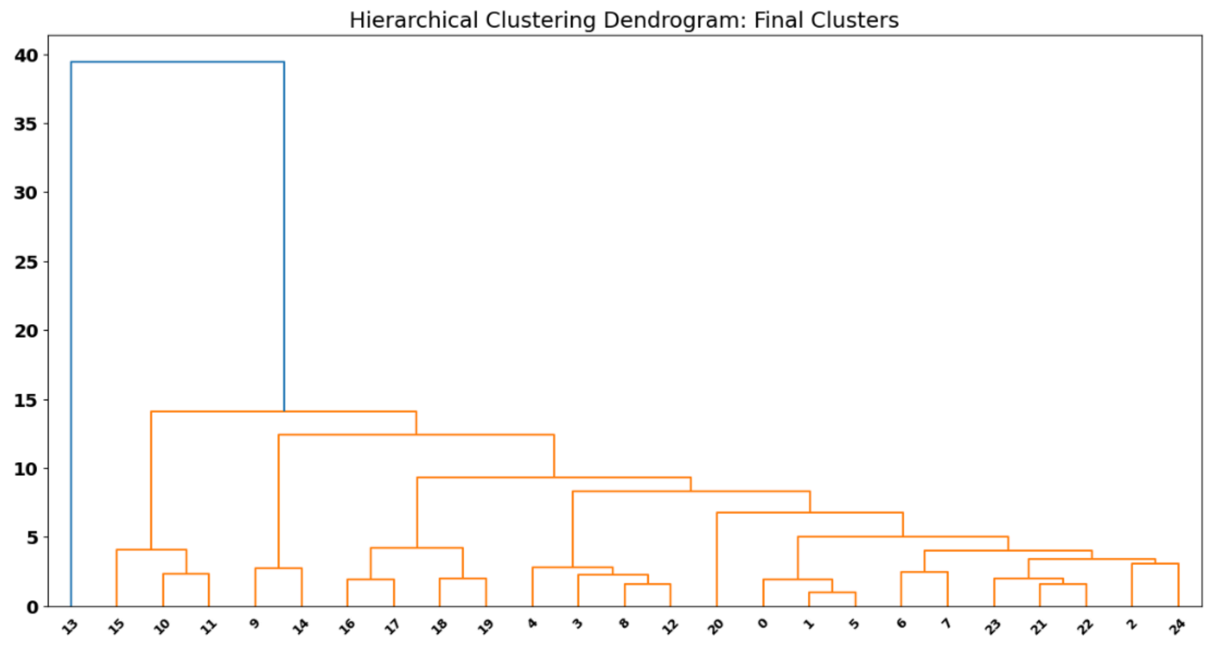
Final Clusters using Confusion Matrix

A confusion matrix was made to compare results from both tandem approaches undertaken. This shows us how much the KMeans and SOM agree with one another in the way they made the clusters i.e. how many observations are similar in each KMeans and SOM clusters.



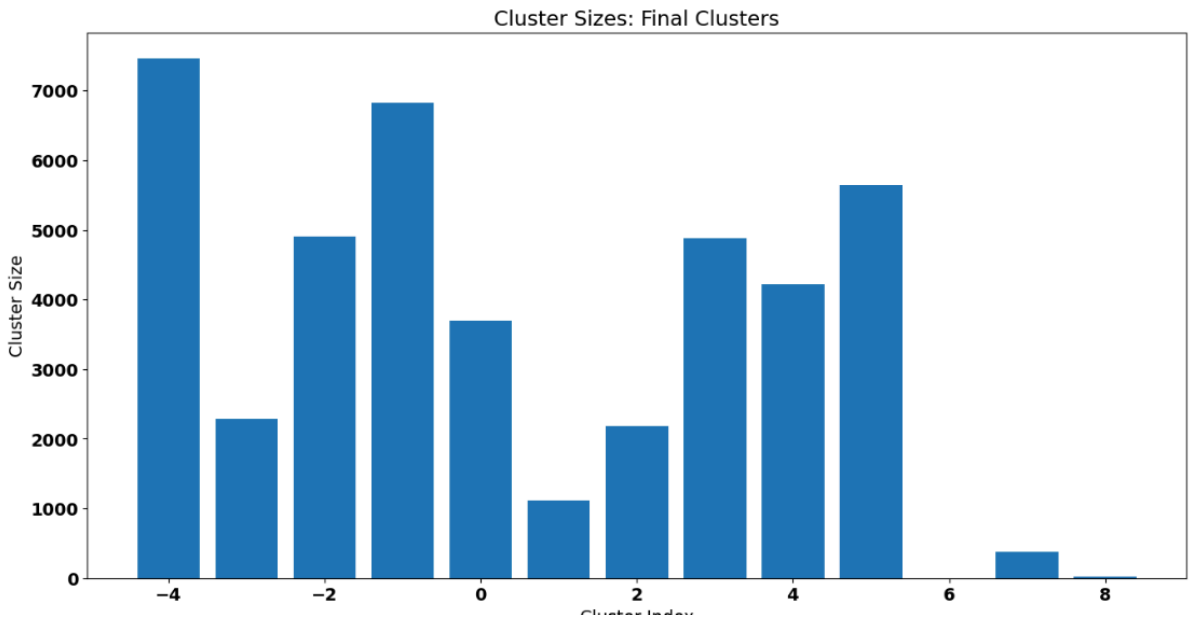
	SOM 0 Cluster	SOM 1 Cluster	SOM 2 Cluster	SOM 3 Cluster	SOM 4 Cluster	SOM 5 Cluster	SOM 6 Cluster	SOM 7 Cluster	SOM 8 Cluster	SOM 9 Cluster
K-means 0 Cluster	0	3348	0	0	0	0	0	0	874	0
K-means 1 Cluster	376	0	0	0	0	0	0	0	0	739
K-means 2 Cluster	2270	0	0	7728	0	31	0	0	12	0
K-means 3 Cluster	7945	0	0	37	0	3	22	0	8	1
K-means 4 Cluster	218	0	0	15	677	4459	23	0	0	0
K-means 5 Cluster	0	0	5642	0	0	0	0	0	0	0
K-means 6 Cluster	75	0	0	0	0	8	4854	0	0	0
K-means 7 Cluster	0	0	0	0	0	0	0	379	0	0
K-means 8 Cluster	2360	0	0	4	0	1	13	0	1	0
K-means 9 Cluster	0	0	0	0	1506	2	0	0	0	0

In order to join the two results to get a final number of clusters, another Hierarchical Clustering was done. But before that, we got the mean of the other variables based on the clusters of both KMeans and SOM and then fit these results into the *AgglomerativeClustering* to obtain the dendrogram.

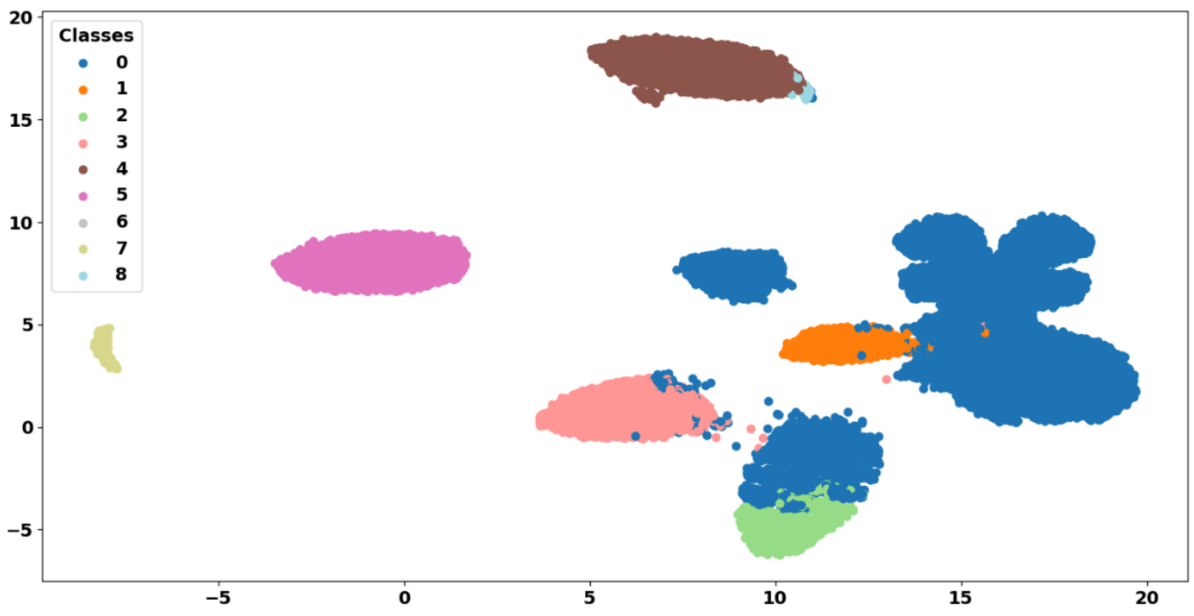


```
Number of clusters chosen:9
```

From the dendrogram, we chose 9 clusters and then stored these results to the original dataset, *f1*, under *final_cluster*. Similar to the tandem approaches we also plotted the cluster sizes to check the distribution of the observations.

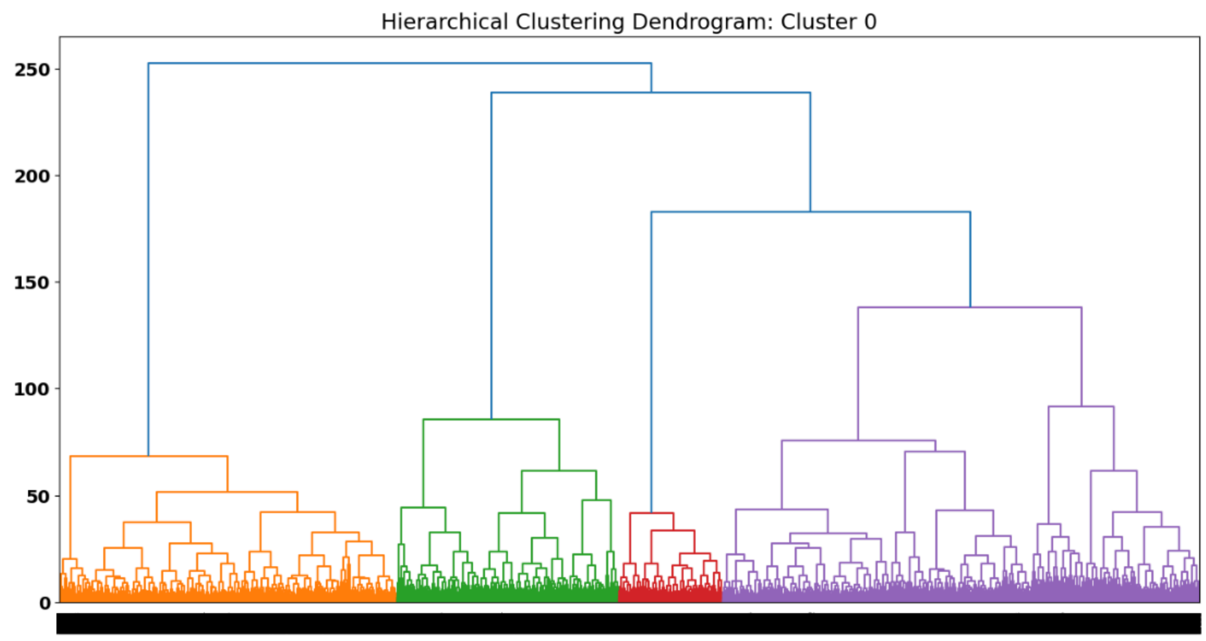


To get a better insight, a UMAP visualization was also made to see the separation of colors within the clusters.



Seeing the UMAP visualization above, clusters 1-5 and 7 are clearly separated yet clusters 6 and 8 have disappeared inside cluster 0, since cluster 0 is already a large cluster. Thus cluster 0 was broken into more clusters so that we can get hold of the hidden clusters.

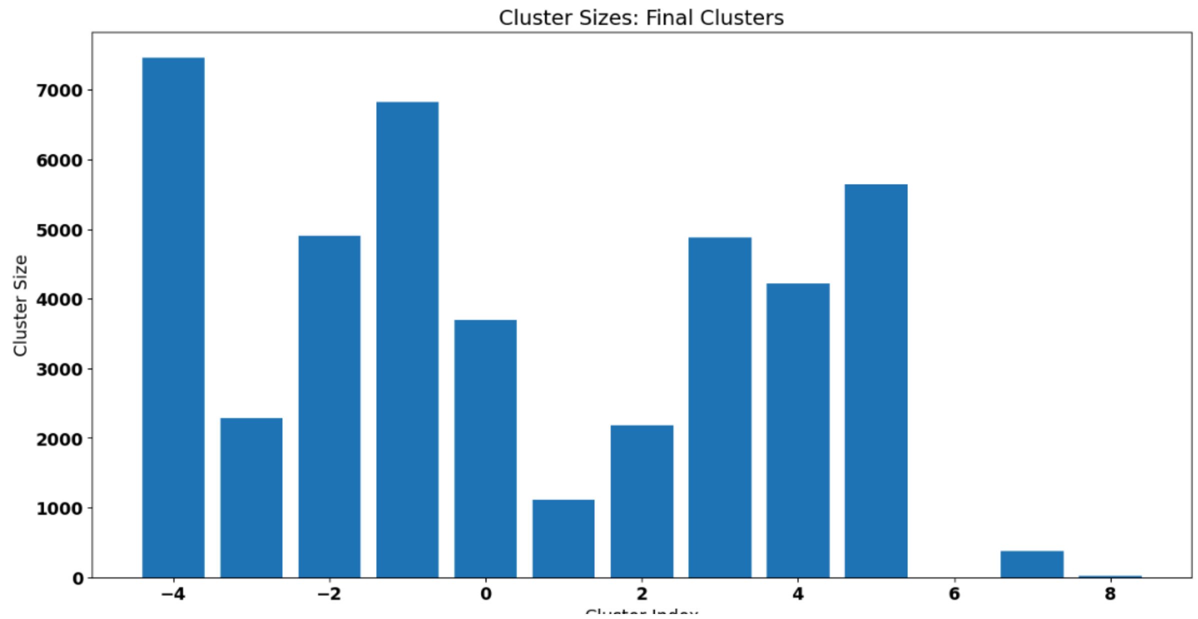
A filtered DataFrame called *cluster0* was created which only consisted of observations that were assigned to cluster 0. These were then fitted to *AgglomerativeClustering* to then plot the dendrogram to choose the number of clusters that cluster 0 was to be broken into.



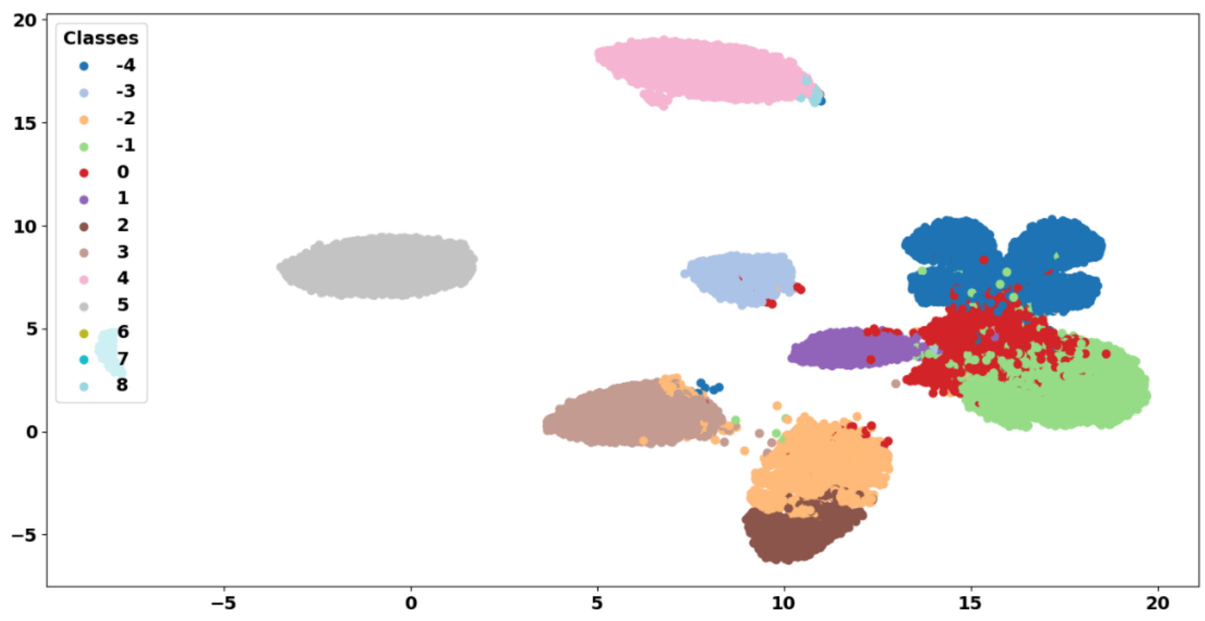
```
Number of clusters chosen:5
```

Clearly cluster 0 was decided to be broken into 5 clusters. The distributions of the observations in *cluster0* were then checked and they seemed fairly distributed. Negative values were given to the clusters i.e. 0 to -4 for each cluster to clearly identify the broken clusters of cluster 0. These results were then left joined with a copy of the original dataset, *cluster0_*.

The cluster sizes were then plotted to check the distribution of the observations of all the clusters - 13 final clusters in total.



Though we still have very small clusters, the plot looks way better than it did before breaking cluster 0. A UMAP visualization was also made to visualize the final 13 clusters.



Though the visualization may not be as pleasant to see, we can see more color separated clusters 2, 3 and 4. As well as Cluster 0 still giving us the issue of not being well separated by color. Yet overall, it seems to be a better solution than before creaking cluster 0.

Cluster Names

The original dataset, *f1*, was then set to check the means of the final clusters in order to give them customized names.

- Cluster -4: Fake Vegetarians
- Cluster -3: Millenials
- Cluster -2: Gamer Parents
- Cluster -1: Misers
- Cluster 0: Local Folks
- Cluster 1: Youngsters
- Cluster 2: Balanced Parents
- Cluster 3: Karens
- Cluster 4: Petters
- Cluster 5: Settled Techies
- Cluster 6: Dirty Oldies
- Cluster 7: Stingy Pescatarians
- Cluster 8: Retired Elders

A point to be noted is that apparently clusters -2 and 2 are quite similar as they both have the largest means of kids and teenagers and that their typical hour is in the evening - most probably after their work. Cluster -4 was named as *Fake Vegetarians* because they had the most spend on vegetables yet they also bought meat and fish. The *Youngsters*, cluster 1, was as expected with a considerable spend on alcohol and their typical hour being in the evening. *Karens* were identified with having the most complaints and distinct products and *Petters* were identified by the most spend on pet food and least electronics.

The cluster with the most promotions was named *Misers* and *Local Folks* were considered to be the ones with the basic stable spend on most items. *Settled Techies* were named due to the most spend on electronics and videogames but also because there was a considerable spend on other items too such as alcohol, groceries and meat as well as having the least promotions. Our *Stingy Pescatarians* had the most spend on fish, at most 20 times more than the other clusters.

These cluster names were then substituted in the original dataset, *f1*. A new dataset named *cl_finaldf* was created to store only the columns *customer_id* and *final_cluster*. This dataset was then exported in the form of a csv file to then be imported in the *AssociationRules* Notebook.

Targeted Promotions

To begin obtaining the association rules for our clusters, we first imported the *final_clusters* file as well as the provided *cust_basket* file containing the transactions of each customer in our dataset. These two were then joined into a new DataFrame which contained the *customer_id*, *list_of_goods* and *final_cl*.

Because we were to obtain customized promotions for the clusters, these *list_of_goods* were then filtered according to their corresponding cluster. With this, the *list_of_goods* was extracted from each filtered DataFrame and put into a list to apply the *apriori* algorithm. From this we pointed out the fact that *Karens*, *Petters* and *Dirty Antiques* had no transactions/invoices.

Each list was then divided into a train and test set where the train set was then encoded using the *TransactionEnconder* to then create a DataFrame of encoded information on each transaction done by the customer in their corresponding cluster. The *apriori* and *association_rules* were then implemented to mainly check the *support*, *confidence* and *lift* values. The appropriate values for *support* and *confidence* were obtained through trial and error and were different for each cluster. With this trial and error, the *lift* was checked every time to see if it improved or not as the *lift* is what indicated a good association rule.

The following paragraphs are the identified association rules with its promotion for each cluster. Similar approaches were taken to suggest promotions yet they somehow differ with the different clusters

Fake Vegetarians

With obvious results, vegetables such as tomatoes, carrots, asparagus, green beans and frozen vegetables were bought the most. Though the *lift* has a value of 1 and a *support* of 0.08, the association rules for this cluster were not as strong but they made sense. Thus the following promotions are suggested:

- ***Soup Lovers***: 20% off on the following: carrots, tomatoes, frozen veggies and asparagus!
- ***Tomato Peers:*** Get 30% off on tomatoes!

Millennials

Normal kitchen-related items were bought frequently by the customers in this cluster and with a small number of kids, the promotions were inspired by that fact. With the highest *lift* value of 2.04 and a *support* value of 0.01, we can say that napkins, cooking oil, spaghetti and candy bars were the most frequent items in their basket. Thus the following promotions are suggested:

- ***Overenergize***: Get a free candy bar only if you buy napkins, spaghetti and cooking oil!
- ***WIpe it off!***: 15% off on napkins if you buy cooking oil!

Gamer Parents

Having the same issue with a *lift* value of 1, the same strategy was applied here. The most frequent items were seen to be candy bars, babies food and cake. These had a *support* value of 0.01. Thus the following promotions are suggested:

- ***Sweet Delight***: 20% off on cake for a fromage blanc and 3 candy bars!
- ***Fry it out***: Get 15% off on napkins if you buy cooking oil, spaghetti and candy bars!

Misers

Though the customers in this cluster claimed to buy items mainly on promotion, the most frequent items in their basket was wine, champagne, beer and cider. With quite a high *lift* value of 8 and 9, the promotions inspired by these are:

- ***Grape Expectations***: 30% off on a bottle of white wine if you buy it with dessert wine and 2 packs of beer!

Local Folks

Having cooking oil, yogurt and cake as frequent items, the *lift* values were majorly 3 for these items. As they have kids and teens the following promotions were suggested:

- ***Cooking duo***: 30% off on a bottle of cooking oil if bought with cake and french fries!
- ***Snack Attack***: Free pack of gum if you buy candy bars and cake!

Youngsters

An obvious insight from this cluster than can be considered is the amount they spend on alcoholic drinks and cheaper foods, thus less grocery. With *lift* values of 2, the following promotions were suggested:

- ***Chic Fizz***: 50 % off on a bottle of champagne if bought with pasta and french fries!
- ***Sweet Tooth***: Free dessert wine for cookies and two packs each of beer and cider!

Balanced Parents

Similar to the *Gamer Parents* the promotions suggested may be similar as they have stable values for all the items and the frequent items are similar to those of *Gamer Parents.* With that in mind, the following promotion is suggested:

- ***Breakie munchies***: For milk and babies food bought, get a pack of muffins on a 50% sale!!

Settled Techies

The amount spent by the customers in this cluster were quite high and checking the association rules, the most frequent items were samsung galaxy 10, grated cheese, cottage cheese, etc. These items were not quite frequent with those of other clusters thus no promotions were suggested as they are shown to be well settled.

Stingy Pescatarians

Another cluster with obvious insights such as different fish being the most frequent items as well as them having bought most items on promotion. With a high *lift* value of 6 and 7, the following promotions are suggested:

- ***Prawn in***: Get 15% off on shrimp if you bought it with oil, milk and sandwich!
- ***Sea's the taste***: 20% off on all seafood if bought with \`fromage blanc\`!

Retired Elders

As expected, since the customers in this cluster are older people, fish or seafood must be more frequent as they have more health benefits as compared to those in meat. With a constant yet high *lift* value for this cluster, 4.4, and a *support* of 0.2, the following promotion is suggested:

- ***Reel in the Fun***: 10% off on \`ratchet & clank\` for any shrimp, olive oil and cooking oil bought!

Conclusion

To achieve the goal of the project that is to make customer segmentation and identify different groups of customers with characteristics that they have in common on data.

We did exploratory data analysis and preprocessing and later on the customer segmentation, clustering, association rules and interpretation to have the targeted promotions.

Though obtaining the dendrogram took a couple of minutes, in comparison to how long it took us to obtain other results in the *Clustering Algorithms* notebook (excluding the UMAPS), we thought it was better to do so instead of having hidden clusters within a large cluster.

The scaling method that gave us better results, in comparison to the other scalers, was the robust scaler.

After visualizing the number of observations of each cluster we obtained a solution of 10 clusters such as: Fake Vegetarians, Millennials, Gamer Parents, Misers, Local Folks, Youngsters, Balanced Parents, Settled Techies, Stingy Pescatarians and Retired Elders.

At the end of this project, we aim to provide the supermarket with results that we can take important information in order to improve the supermarket profit because with the information taken from the customer segmentation we can understand what can be improved and also which promotions can be given to the clients.

References

- Built In. (n.d.). *Pandas DataFrame: Show All Columns/Rows*. https://builtin.com/data-science/pandas-show-all-columns
- Python Software Foundation. (n.d.). *AST — Abstract Syntax Trees*. https://docs.python.org/3.8/library/ast.html
- Pandey, V. (n.d.). *Using Apriori Algorithm for Association Rule Mining*. https://www.linkedin.com/pulse/using-apriori-algorithm-association-rule-mining-dr-vivek-pandey/
- GeeksforGeeks. (n.d.). *Python | Convert a string representation of list into list*. https://www.geeksforgeeks.org/python-convert-a-string-representation-of-list-into-list/
- Mullin, T. (n.d.). *DBSCAN Parameter Estimation*. https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd
- Analytics Vidhya. (n.d.). *Lambda, Map, and Filter Functions in Python*. https://medium.com/analytics-vidhya/lambda-map-filter-functions-in-python-4c03679dd747