Projet 05 :

Segmentez des clients d'un site e-commerce

Mohamed A.

N.B :

We were asked to provide 3 notebooks , I put all code in one notebook with a clear separation for each step.

- *Nom_Prénom_1_notebook_exploration_mmaaaa*
- *Nom_Prénom_2_notebook_essais_mmaaa*
- *Nom_Prénom_3_notebook_simulation_mmaaaa*

Un **notebook de l'analyse exploratoire** (non cleané, pour comprendre votre démarche).

Un **notebook** (ou code commenté au choix) **d'essais** des différentes approches de modélisation (non cleané, pour comprendre votre démarche).

Un **notebook de simulation pour déterminer la fréquence nécessaire de mise à jour** du modèle de segmentation.

Summary of my Notebook :

I -Loading all necessary pyhton Modules

II-Loading all csv files on notebook

III- Exploratory Data analysis :

III-1 Check size and type of data :

- Data size :
- Data types :

III- 2 Merging datafrmes to get one useful set of data :

III- 3 Correcting dtypes aspect for some columns :

III- 4 Dropping duplicated rows :

III- 5 Identifying and handling missing values :

III- 6 Understand data and perform some analysis :

IV - CUSTOMER SEGMENTATION

IV-1 RFM clustering and Analysis :

IV-2 Calculating RFM score

IV- 3 Rating Customer based upon the RFM score :

V- FEATURE ENGINEERING :

VI- CLUSTERING USING UNSUPERVISED ALGORITHMS :

VI-1 K_MEANS CLUSTERING :

VI-2 K_PROTOTYPES CLUSTERING :

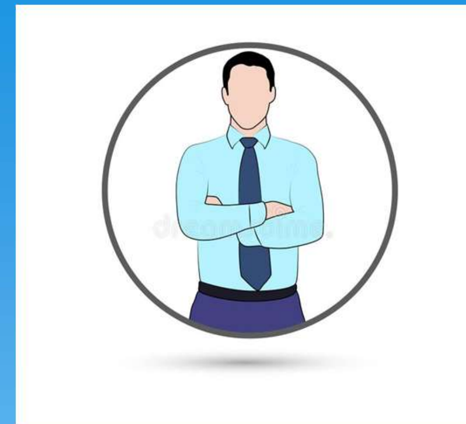
VI-3 DBSCAN CLUSTERING :

VII- ARI CALCULATION AMD MODEL STABILITY :

VII-1 Identify clients distribution over each interval of time from 30 days to 210 days.

VIII- Feature importance and clustering explicability

Contexte du projet :



Je suis consultant pour Olist, une entreprise brésilienne qui propose une solution de vente sur les marketplaces en ligne

Olist souhaite que vous fournissiez à ses équipes d'e-commerce une **segmentation des clients** qu’elles pourront utiliser au quotidien pour leurs campagnes de communication.

Votre objectif est de **comprendre les différents types d’utilisateurs** grâce à leur comportement et à leurs données personnelles.

Vous devrez **fournir à l’équipe marketing une description actionable** de votre segmentation et de sa logique sous-jacente pour une utilisation optimale, ainsi qu’une **proposition de contrat de maintenance** basée sur une analyse de la stabilité des segments au cours du temps.

Etapes du projet et livrables :

1 – Presentation of Olist compagnie et country from where data has been collected

2- Data Description & EDA outputs .

3- Clients Segmentation using RFM technique

4- Clients clusterisation by unsupervised techniques :

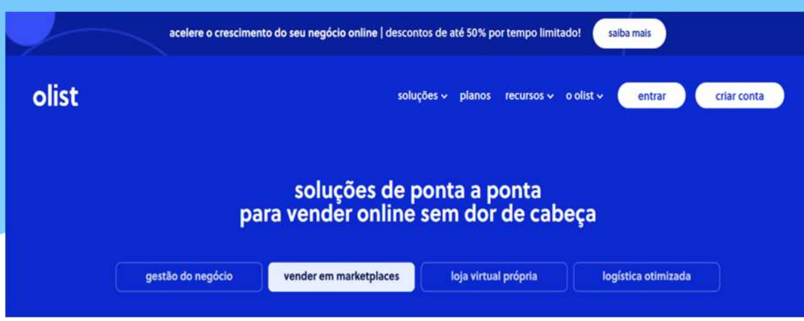
4-1 Feature engineering
4-2 K-Means
4-3 K-protoypes
4-4 DBSCAN

5- Supervised ML clustering for model explicability :

6- ARI score for stability assesement and maintenance's contract

1 – Who, how and from where ? :

Olist is **an e-commerce solution whose purpose is to stimulate a company's sales in the digital environment.** Using this platform, it is possible to advertise on the main marketplaces active in the country and maintain centralized management of operations with Olist's logistical and customer service support.



2- Data description and EDA outputs :

Liste des fichiers

```
['olist_customers_dataset.csv',
 'olist_geolocation_dataset.csv',
 'olist_orders_dataset.csv',
 'olist_order_items_dataset.csv',
 'olist_order_payments_dataset.csv',
 'olist_order_reviews_dataset.csv',
 'olist_products_dataset.csv',
 'olist_sellers_dataset.csv',
 'product_category_name_translation.csv']
```

9 fichiers csv fournis

Shapes & columns

	dataset	no_of_columns	columns_name	no_of_rows
0	customers	5	customer_id, customer_unique_id, customer_zip_code_prefix, customer_city, customer_state	99441
1	sellers	4	seller_id, seller_zip_code_prefix, seller_city, seller_state	3095
2	geolocalisations	5	geolocation_zip_code_prefix, geolocation_lat, geolocation_lng, geolocation_city, geolocation_state	1000163
3	items	7	order_id, order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value	112650
4	payments	5	order_id, payment_sequential, payment_type, payment_installments, payment_value	103886
5	orders	8	order_id, customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date	99441
6	reviews	7	review_id, order_id, review_score, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp	99224
7	products	9	product_id, product_category_name, product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm	32951

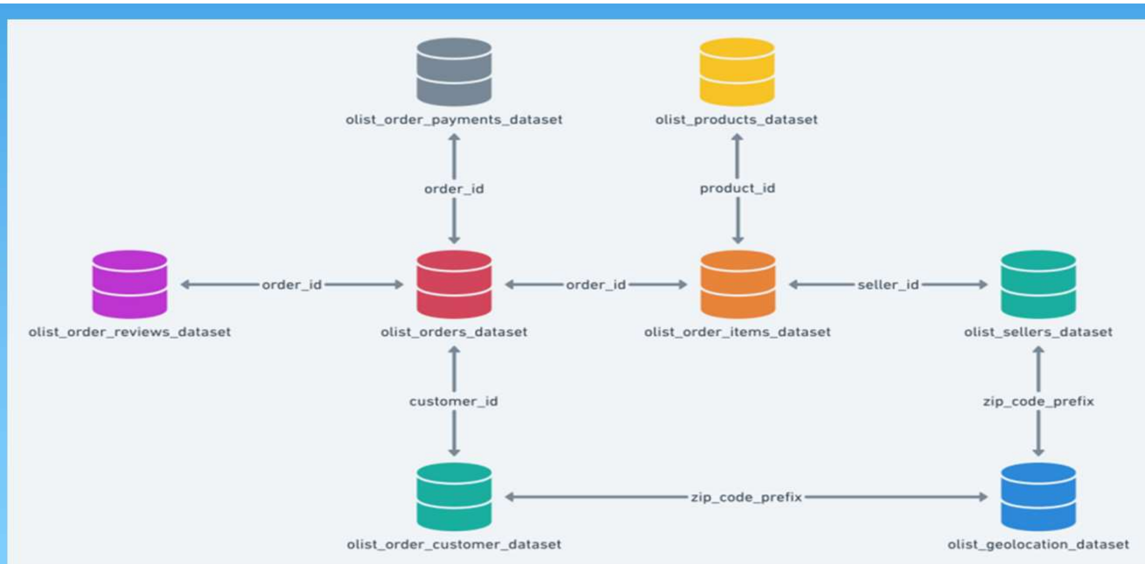
Dtypes

	dataset	numeric_features	num_features_name	object_features	objt_features_name	bool_features
0	customers	1	customer_zip_code_prefix	4	customer_id, customer_unique_id, customer_city, customer_state	0
1	sellers	1	seller_zip_code_prefix	3	seller_id, seller_city, seller_state	0
2	geolocalisations	3	geolocation_zip_code_prefix, geolocation_lat, geolocation_lng	2	geolocation_city, geolocation_state	0
3	items	3	order_item_id, price, freight_value	4	order_id, product_id, seller_id, shipping_limit_date	0
4	payments	3	payment_sequential, payment_installments, payment_value	2	order_id, payment_type	0
5	orders	0		8	order_id, customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date	0
6	reviews	1	review_score	6	review_id, order_id, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp	0
7	products	7	product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm	2	product_id, product_category_name	0

2- Data description and EDA outputs :

2- 1 Merge des fichiers csv :

Le schema ci dessous indique la connexion qui existe entre les differents csv source, et c’est sur la base de ces colonnes commune qu’un merge est realise pour obtenir un dataframe unique et representatif pour l’analyse qui va suivre dans ce projet



Le script du code utilise pour le merge

```
df_merge_1 = pd.merge(left=df_cust, right=df_or, left_on='customer_id', right_on='customer_id')
df_merge_2 = pd.merge(left=df_merge_1, right=df_or_items, left_on='order_id', right_on='order_id')
df_merge_3 = pd.merge(left=df_merge_2, right=df_or_rev, left_on='order_id', right_on='order_id')
df_merge_4 = pd.merge(left=df_merge_3, right=df_prod, left_on='product_id', right_on='product_id')
df_merge_5 = pd.merge(left=df_merge_4, right=df_seller, left_on='seller_id', right_on='seller_id')
df_global = pd.merge(left=df_merge_5, right=df_or_pay, left_on='order_id', right_on='order_id')
print('Merged dataframe shape  :',df_global.shape)
print('*****************************************')
display(df_global.head(3))
```

```
Merged dataframe shape  : (117329, 39)
*****************************************
```

2- Data description and EDA outputs :

2- 2 Traitement des missing values, dtypes , outliers et duplicated :

Date dtype correction :

```
#   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
0   order_delivered_carrier_date   116094 non-null  object
1   order_delivered_customer_date  114858 non-null  object
2   order_estimated_delivery_date  117329 non-null  object
3   shipping_limit_date            117329 non-null  object
4   review_creation_date           117329 non-null  object
5   order_approved_at              117314 non-null  object
6   review_answer_timestamp        117329 non-null  object
7   order_purchase_timestamp       117329 non-null  object
dtypes: object(8)
```

```
---  ------                         --------------   -----
0   order_delivered_carrier_date   116094 non-null  datetime64[ns]
1   order_delivered_customer_date  114858 non-null  datetime64[ns]
2   order_estimated_delivery_date  117329 non-null  datetime64[ns]
3   shipping_limit_date            117329 non-null  datetime64[ns]
4   review_creation_date           117329 non-null  datetime64[ns]
5   order_approved_at              117314 non-null  datetime64[ns]
6   review_answer_timestamp        117329 non-null  datetime64[ns]
7   order_purchase_timestamp       117329 non-null  datetime64[ns]
```

Duplicated removal :

```
df_global.drop_duplicates(subset=['customer_id','customer_unique_id','order_purchase_timestamp','order_id'],keep='first', inplace=True)
```

Handling missing values :

```
# Replacement of missing dates
df_global["order_approved_at"].fillna(df_global["order_purchase_timestamp"], inplace=True)
df_global["order_delivered_customer_date"].fillna(df_global["order_estimated_delivery_date"], inplace=True)
# Handling missing values of numerical features
df_global['product_weight_g'].fillna(df_global['product_weight_g'].median(),inplace=True)
df_global['product_length_cm'].fillna(df_global['product_length_cm'].median(),inplace=True)
df_global['product_height_cm'].fillna(df_global['product_height_cm'].median(),inplace=True)
df_global['product_width_cm'].fillna(df_global['product_width_cm'].median(),inplace=True)
# replacing product_cateory_name missing values by 'no_category' to avoid losing data about customer and eventually geo_ratios
df_global['product_category_name'].fillna('no_category',inplace=True)
# replacing product_cateory_name missing values by 'review_comment_message' by no_review
df_global['review_comment_message'].fillna('no_review',inplace=True)
```

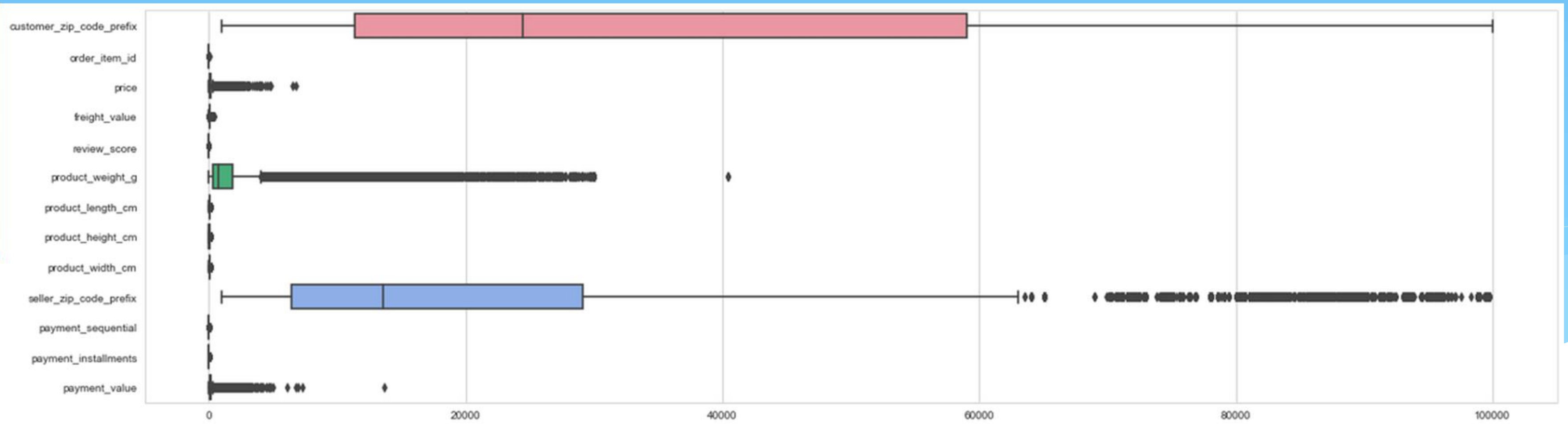
2- Data description and EDA outputs :

2- 2 Traitement des missing values, dtypes , outliers et duplicated :

Dropping irrelevant columns :

```
df_global.drop('review_id',axis=1,inplace=True)
df_global.drop('review_comment_title',axis=1,inplace=True)
#dropping order delivery carrier date
df_global.drop(labels='order_delivered_carrier_date',axis=1,inplace=True)
#dropping 'product_name_lenght','product_description_lenght','product_photos_qty'
df_global.drop(labels='product_name_lenght',axis=1,inplace=True)
df_global.drop(labels='product_description_lenght',axis=1,inplace=True)
df_global.drop(labels='product_photos_qty',axis=1,inplace=True)
df_global.drop('order_id', axis=1, inplace=True)
```

Outliers handling :

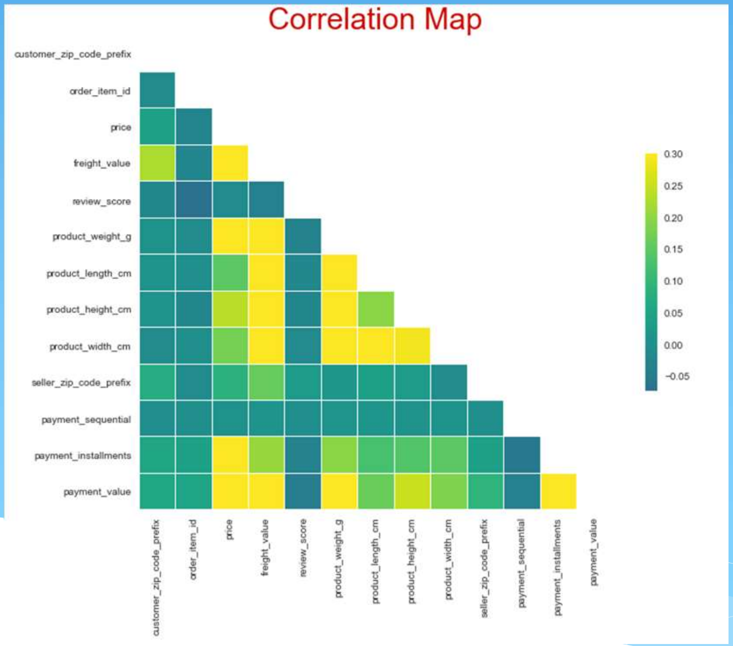



```
No remarkable outlier within data
```

2- Data description and EDA outputs :

2- 3 Data analysis :

	customer_zip_code_prefix	order_item_id	price	freight_value	review_score	product_weight_g	product_length_cm	product_height_cm	product_width_cm	seller_zip_code_prefix	payment_sequential	payment_installments	payment_value
count	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000	97916.000000
mean	35174.983772	1.016494	125.748446	20.174040	4.105162	2100.469903	30.085961	16.473161	23.017096	24593.422801	1.022805	2.914835	157.742223
std	29822.392569	0.148882	189.949099	15.891107	1.331291	3763.819239	16.111278	13.315982	11.731499	27677.490470	0.250824	2.707027	216.861653
min	1003.000000	1.000000	0.850000	0.000000	1.000000	0.000000	7.000000	2.000000	6.000000	1001.000000	1.000000	0.000000	0.010000
25%	11353.750000	1.000000	41.500000	13.250000	4.000000	300.000000	18.000000	8.000000	15.000000	6429.000000	1.000000	1.000000	60.000000
50%	24422.000000	1.000000	79.000000	16.350000	5.000000	700.000000	25.000000	13.000000	20.000000	13560.000000	1.000000	2.000000	103.260000
75%	59032.750000	1.000000	139.900000	21.210000	5.000000	1800.000000	38.000000	20.000000	30.000000	29156.000000	1.000000	4.000000	174.990000
max	99990.000000	9.000000	6735.000000	409.680000	5.000000	40425.000000	105.000000	105.000000	118.000000	99730.000000	27.000000	24.000000	13664.080000



2- Description des donnees fournies & explications de l'EDA :

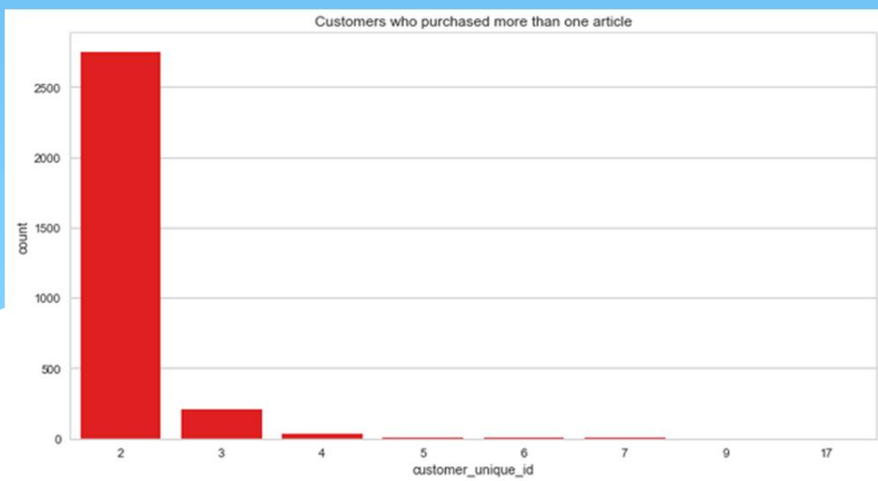
2- 3 Data analysis :

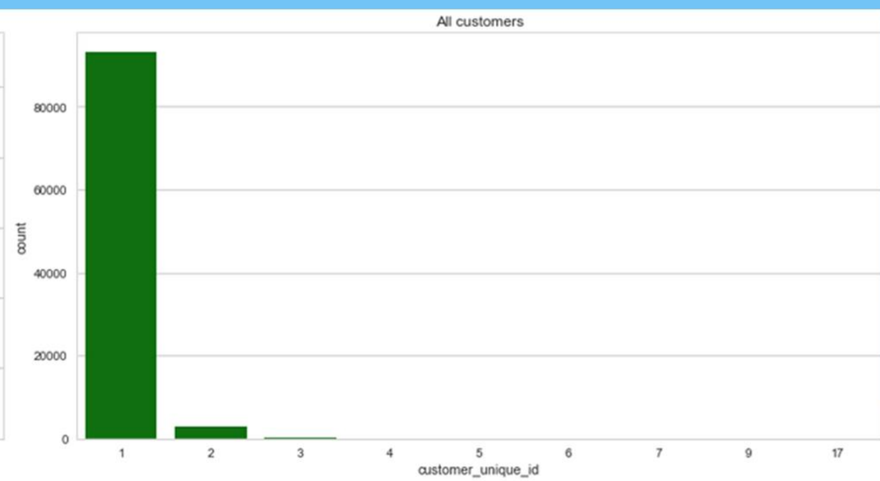
```
************   Count of    : *****************
1- cities    : 4108
2- states    : 27
3- zip_codes    : 14955
*******************************************
```

```python
# Exemple of zip_codes distribution within a city
# Check in the net if those zip_codes are really distributed as provided by dataset
# Url with all information : https://www.postcodesdb.com/AlphabeticSearch.aspx?country=Brazil&city=Santar%C3%A9m
santarem = df_global.loc[df_global['customer_city']=='santarem']
santarem['customer_zip_code_prefix'].value_counts()
```

```
68005    19
68040     7
68030     6
68010     5
68020     3
68025     2
68022     1
68035     1
```

```
*************************************************************
clients who have purchased more than once   2997
clients who have purchased more than once ratio  :3.01 %
*************************************************************
```



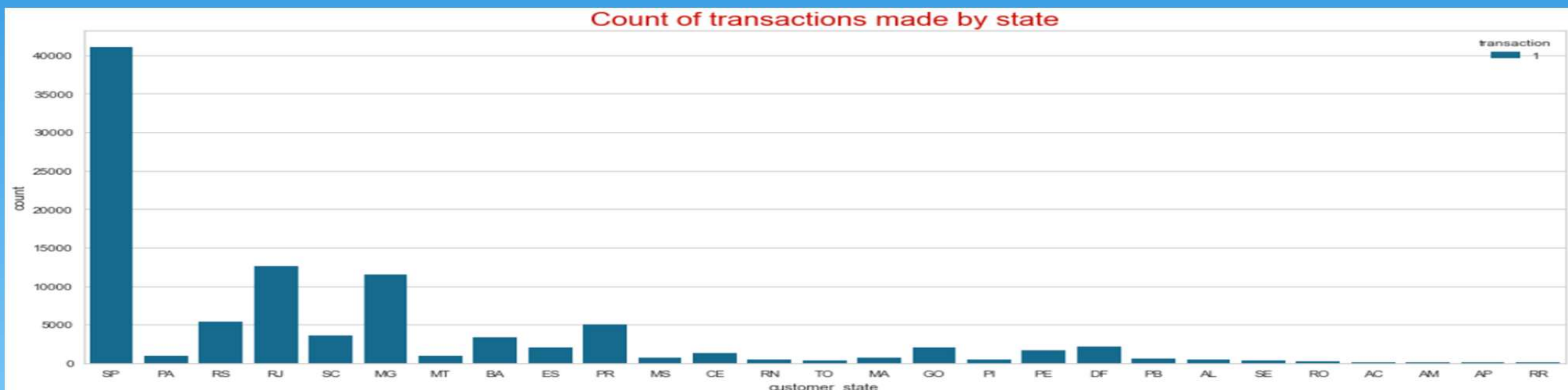


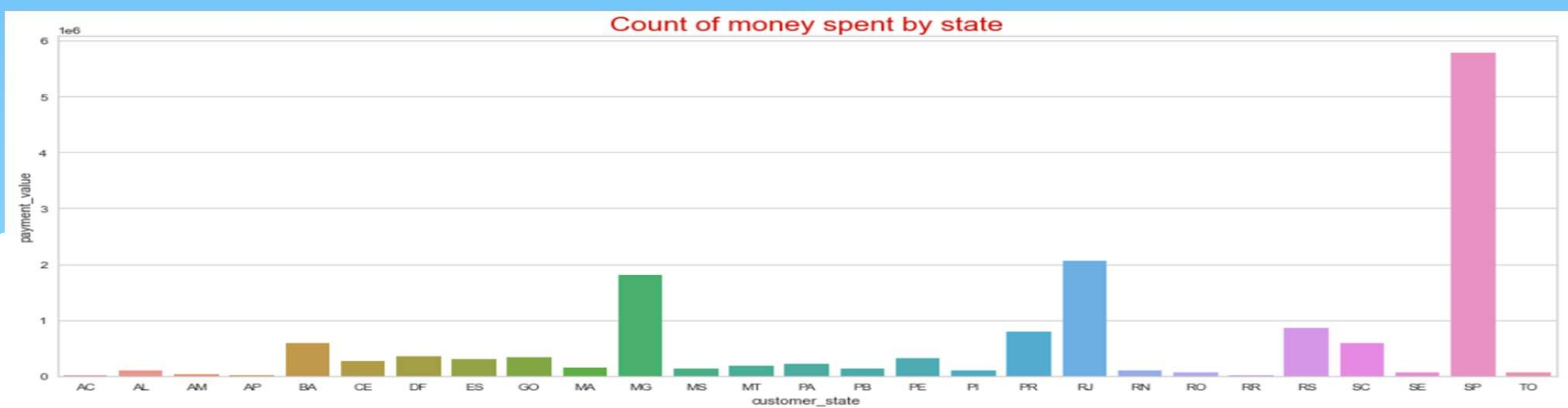
2- Data description and EDA outputs :

2- 3 Data analysis :

```
df['payment_value'].value_counts(bins=5)

(-13.655, 2732.824]      97854
(2732.824, 5465.638]        57
(5465.638, 8198.452]         4
(10931.266, 13664.08]        1
(8198.452, 10931.266]        0
```

```
****************************************************
Number of products types  : 74
****************************************************
```





3- Clients Segmentation using RFM technique :

RFM stands for recency, frequency, monetary value. In business analytics, we often use this concept to divide customers into different segments, like high-value customers, medium value customers or low-value customers, and similarly many others.

Recency: How recently has the customer made a transaction

Frequency: How frequent is the customer in ordering/buying some product

Monetary: How much does the customer spend on purchasing products.RFM stands for recency, frequency, monetary value. In business analytics, we often use this concept to divide customers into different segments, like high-value customers, medium value customers or low-value customers, and similarly many others.

	CustomerCode	Recency	Frequency	Monetary
0	0000366f3b9a7992bf8c76cfdf3221e2	115	1	141.90
1	0000b849f77a49e4a4ce2b2a4ca5be3f	118	1	27.19
2	0000f46a3911fa3c0805444483337064	541	1	86.22
3	0000f6ccb0745a6a4b88665a16c9f078	325	1	43.62
4	0004aac84e0df4da2b147fca70cf8255	292	1	196.89

	CustomerCode	Recency	Frequency	Monetary	R_rank_norm	F_rank_norm	M_rank_norm
0	0000366f3b9a7992bf8c76cfdf3221e2	115	1	141.90	76.153927	48.486064	48.486064
1	0000b849f77a49e4a4ce2b2a4ca5be3f	118	1	27.19	75.121938	48.486064	48.486064
2	0000f46a3911fa3c0805444483337064	541	1	86.22	3.592166	48.486064	48.486064
3	0000f6ccb0745a6a4b88665a16c9f078	325	1	43.62	28.791174	48.486064	48.486064
4	0004aac84e0df4da2b147fca70cf8255	292	1	196.89	33.800148	48.486064	48.486064

	CustomerCode	RFM_Score
0	0000366f3b9a7992bf8c76cfdf3221e2	2.63
1	0000b849f77a49e4a4ce2b2a4ca5be3f	2.62
2	0000f46a3911fa3c0805444483337064	2.09
3	0000f6ccb0745a6a4b88665a16c9f078	2.28
4	0004aac84e0df4da2b147fca70cf8255	2.31
5	0004bd2a26a76fe21f786e4fbd80607f	2.57
6	00050ab1314c0e55a6ca13cf7181fecf	2.59

3- Clients Segmentation using RFM technique :

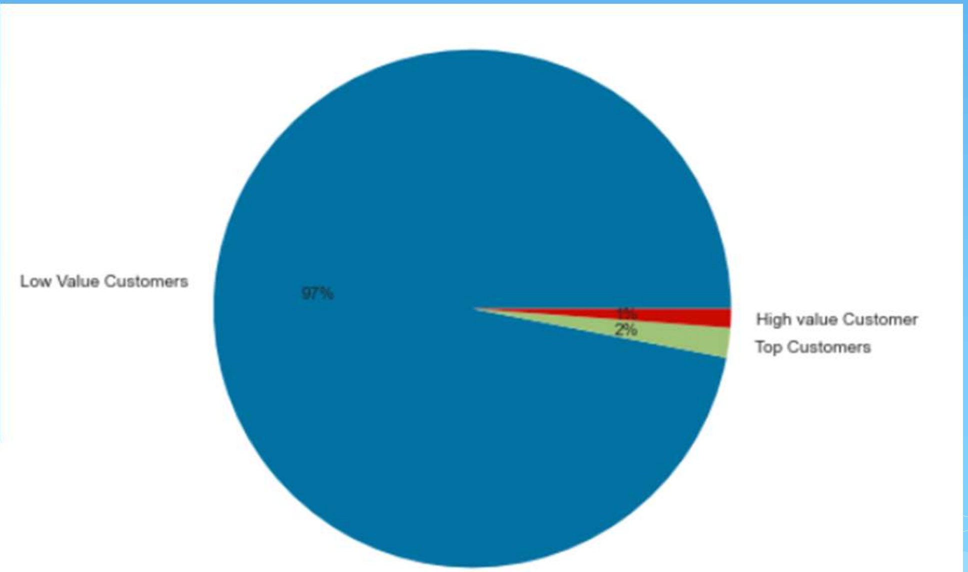
rfm score >4.5 : Top Customer

4.5 > rfm score > 4 : High Value Customer

4>rfm score >3 : Medium value customer

3>rfm score>1.6 : Low-value customer

rfm score<1.6 :Lost Customer



4- Clients clusterisation by unsupervised techniques :

4-1 Feature engineering :

- **Time and delay variables :**

```
# Define 'delivery time'
df['delivery_time_d'] = (df['order_delivered_customer_date'] - df['order_purchase_timestamp']).dt.days
# Define 'proposed delivery time'
df['proposed_delivery_time_d'] = (df['order_estimated_delivery_date'] - df['order_purchase_timestamp']).dt.days
# Define 'delay between proposed and effective delivery time'
df['Diff_delivery_time_d'] = (df['order_estimated_delivery_date'] - df['order_delivered_customer_date']).dt.days
# Define 'delay between review_answer and review_creation time'
df['Diff_review_time_d'] = (df['review_answer_timestamp'] - df['review_creation_date']).dt.days
# Define 'shipping time limit from order_approved_time'
df['Diff_shipping_limit_ord_app'] = (df['shipping_limit_date'] - df['order_approved_at']).dt.days
```

- **Product volume variable :**

```
# Define prodcut_volume to replace product shape specifications
df['prodcut_volume'] = df['product_length_cm']*df['product_height_cm']*df['product_width_cm']
```

- **Distance between buyer and seller variable :**

```
# calculate distance between seller and customer to be able to include positions within analysis
df['distance'] = np.sqrt((df['seller_geolocation_lat'] - df['customer_geolocation_lat'])**2 + (df['seller_geolocation_lng'] - df['customer_geolocation_lng'])**2)
```

4- Clients clusterisation by unsupervised techniques :

4-2 K-Means clustering :

- **Steps of clustering : We have to drop all categorical columns as K-Means deals only with numerical data**

```
# Scaling the data
scaler = MinMaxScaler()
X = scaler.fit_transform(df_km)
```

```
kmeans = KMeans(n_clusters=10,init='k-means++' )
kmeans.fit(X)
```

Results :

```
# Estimate kmeans inertia
kmeans.inertia_
```

```
2089.3797805491185
```

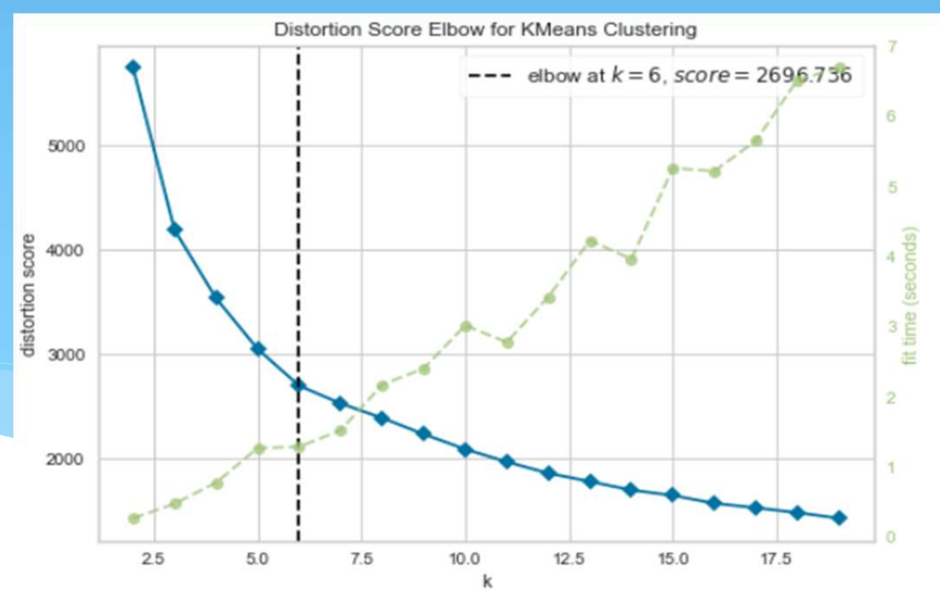
```
kmeans.score(X)
```

```
-2089.379780549119
```

- **Elbow method for K optimisation best choice :**

Elbow method is one of the most popular method used to select the optimal number of clusters by fitting the model with a range of values for K in K-means algorithm. Elbow method requires drawing a line plot between SSE (Sum of Squared errors) vs number of clusters and finding the point representing the “elbow point” (the point after which the SSE or inertia starts decreasing in a linear fashion). Here is the sample elbow point. In the later sections, it is illustrated as to how to draw the line plot and find elbow point.

The elbow point represents the point in the SSE / Inertia plot where SSE or inertia starts decreasing in a linear manner.

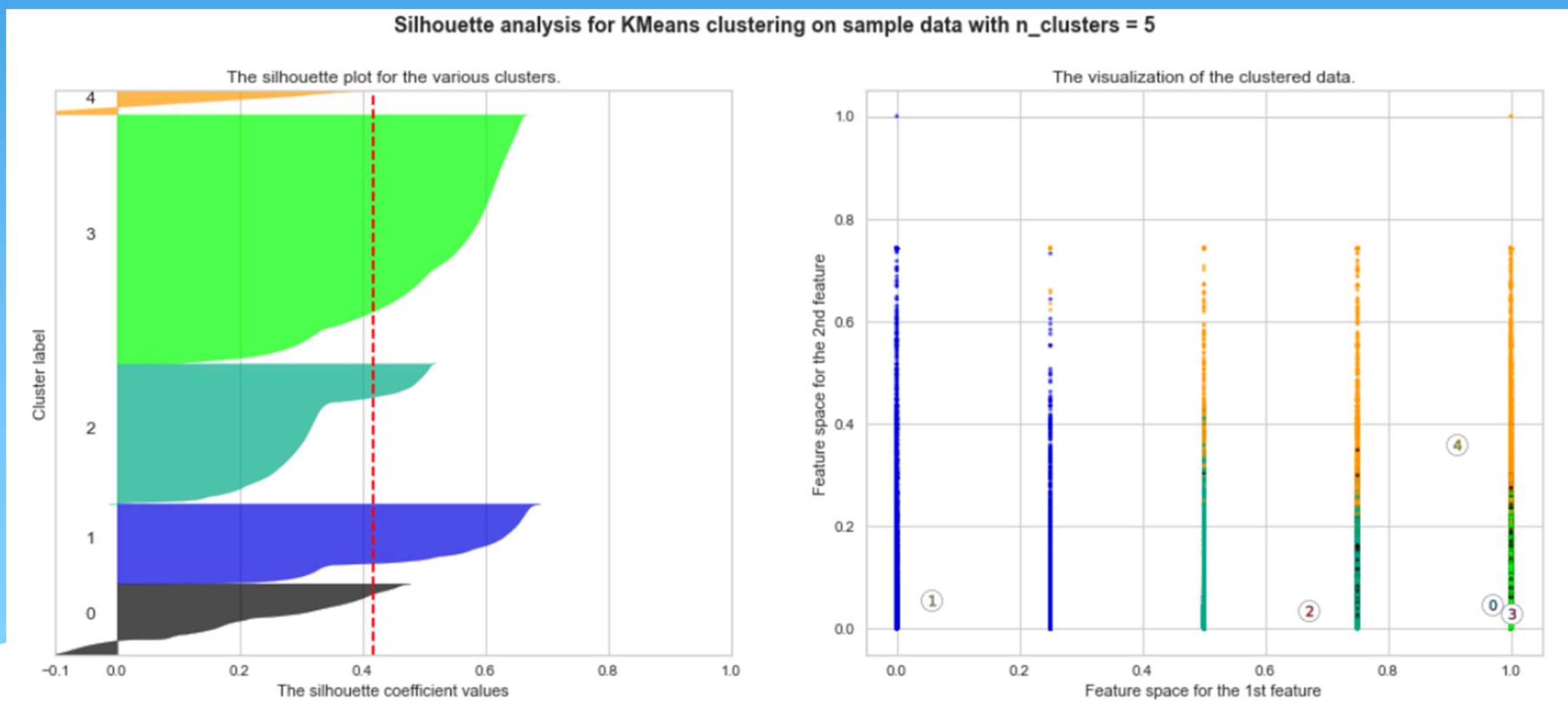


4- Clients clusterisation by unsupervised techniques :

4-2 K-Means clustering : (suite)

- **Silouhette method : We have to drop all categorical columns as K-Means deals only with numerical data**

```
For n_clusters = 4 The average silhouette_score is : 0.4353769021201024
For n_clusters = 5 The average silhouette_score is : 0.41689544501254094
For n_clusters = 6 The average silhouette_score is : 0.3990040026055856
For n_clusters = 7 The average silhouette_score is : 0.40340691487215785
```



Best choice for K = 5 (by combining Elbow and silhouette scores)

4- Clients clusterisation by unsupervised techniques :

4-3 K-Prototypes clustering :

We define relevant columns (both categorical and numerical) to be used for clustering.
This is based to get some balance between real and academic projects and avoid time consuming analysis which can go beyond the allowed time for this project.

#	Column	Non-Null Count	Dtype
0	customer_city	97916 non-null	object
1	customer_state	97916 non-null	object
2	review_score	97916 non-null	int64
3	product_category_name	97916 non-null	object
4	product_weight_g	97916 non-null	float64
5	seller_city	97916 non-null	object
6	seller_state	97916 non-null	object
7	payment_type	97916 non-null	object
8	payment_installments	97916 non-null	int64
9	payment_value	97916 non-null	float64
10	distance	97916 non-null	float64
11	prodcut_volume	97916 non-null	float64
12	delivery_time_d	97916 non-null	int64
13	proposed_delivery_time_d	97916 non-null	int64

Steps of clustering :

```
Categorical columns          : ['customer_city', 'customer_state', 'product_category_name', 'seller_city', 'seller_state', 'payment_type']
Categorical columns position  : [0, 1, 3, 5, 6, 7]
```

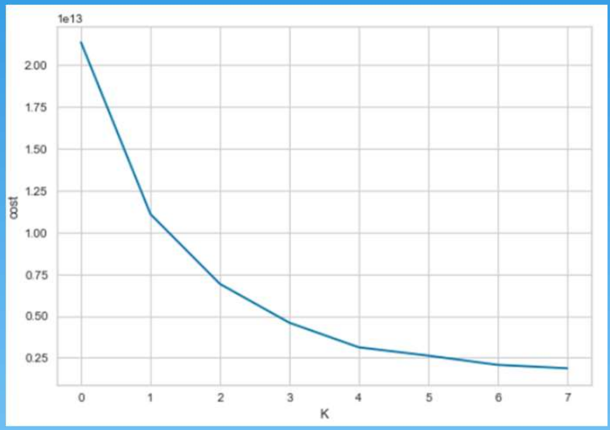
```
array([['franca', 'SP', 4, ..., 107136.0, 8, 19],
       ['santarem', 'PA', 1, ..., 107136.0, 18, 39],
       ['nova santa rita', 'RS', 3, ..., 107136.0, 18, 35],
       ...,
       ['guarulhos', 'SP', 3, ..., 74307.0, 6, 6],
       ['uruacu', 'GO', 5, ..., 41976.0, 9, 22],
       ['bom repouso', 'MG', 1, ..., 13860.0, 35, 27]], dtype=object)
```

4- Clients clusterisation by unsupervised techniques :

4-3 K-Prototypes clustering :

We define a loop to get an elbow curve based on function cost for each choosen 'K'
The plot at the right shows how costs vary according to number of clusters.

```
# No scaling of numerical columns of dataframe
#Choosing optimal K value
k_clusters = [2,3,4,5,6,7,8,9]
cost = []
X = df_pro
for num_clusters in k_clusters:
    kproto = KPrototypes(n_clusters=num_clusters, random_state=42,verbose=2,max_iter=15)
    kproto.fit_predict(X, categorical=[0, 1, 3, 5, 6, 7])
    cost.append(kproto.cost_)

plt.plot(cost)
plt.xlabel('K')
plt.ylabel('cost')
plt.show
```



Best choice for K = 5 (according to costs curve)

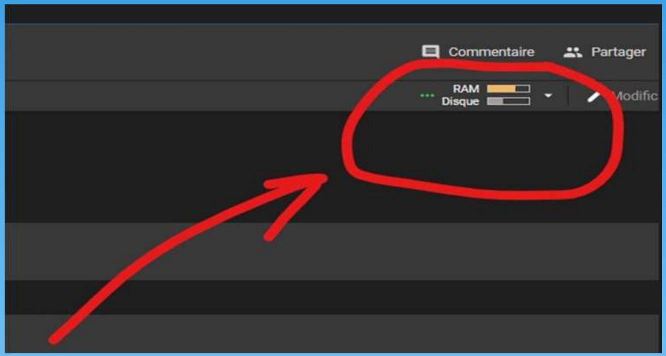
4- Clients clusterisation by unsupervised techniques :

4-4 DBSCAN:

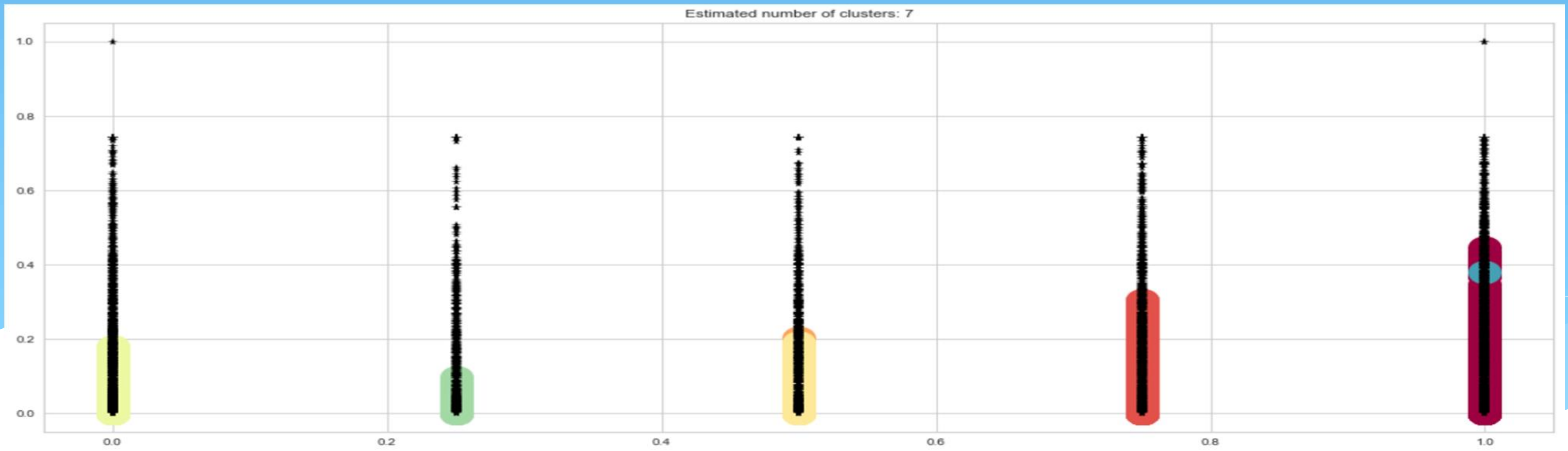
Coding steps :

```
scaler = MinMaxScaler()
X = scaler.fit_transform(df2)

print('matrix shape :',X.shape)
clustering = DBSCAN(eps=0.06, min_samples=20
                   ).fit(X)
cluster = clustering.labels_

matrix shape : (97916, 8)

# check how many clusters were found during processing
len(set(cluster))

8

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(cluster)) - (1 if -1 in cluster else 0)
n_noise_ = list(cluster).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(X, cluster))

Estimated number of clusters: 7
Estimated number of noise points: 10780
Silhouette Coefficient: 0.305
```

Unfortunately and according to hardware limitation (memory issue even with 32g RAM) , the optimization of parameter ‘eps’ wasn’t possible (see image below)



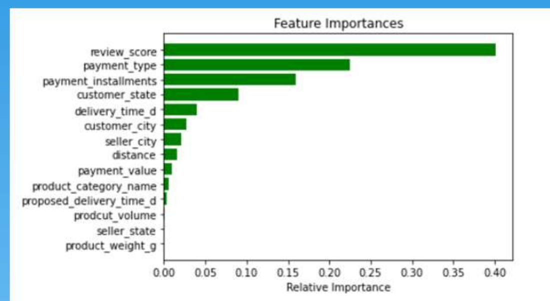
Graphical representation of clusters :



5- Supervised ML clustering for model explicability :

Coding steps :
1 – We perform clustering with optimal K (K=5)
2- We define cluster_labels as TARGET and use a RandomForestClassifier to fit_predict those labels using dataset
3- We extract feature importance and mean results from classifier used
3- We use Shap and Shapash to get features importances for each class

Feature importane for all dataset :



Means and 75% :

From class '0' to '4'

	review_score	product_weight_g	payment_installments	payment_value	distance	prodcut_volume	delivery_time_d	proposed_delivery_time_d	cluster_id
mean	4.552635	1911.665406	1.008911	137.717098	5.712086	13898.790935	11.812412	23.19573	0.0
75%	5.000000	1650.000000	1.000000	155.572500	7.708434	15750.000000	15.000000	28.00000	0.0

	review_score	product_weight_g	payment_installments	payment_value	distance	prodcut_volume	delivery_time_d	proposed_delivery_time_d	cluster_id
mean	4.432418	2195.824314	3.219245	150.037088	5.874325	15806.355003	13.042965	26.243136	1.0
75%	5.000000	1928.250000	4.000000	170.760000	7.247719	20539.000000	17.000000	31.000000	1.0

	review_score	product_weight_g	payment_installments	payment_value	distance	prodcut_volume	delivery_time_d	proposed_delivery_time_d	cluster_id
mean	1.659859	2327.61195	3.18843	176.602693	6.086634	16577.668604	18.567681	24.101086	2.0
75%	2.000000	1931.00000	4.00000	193.435000	7.845549	19800.000000	26.000000	29.000000	2.0

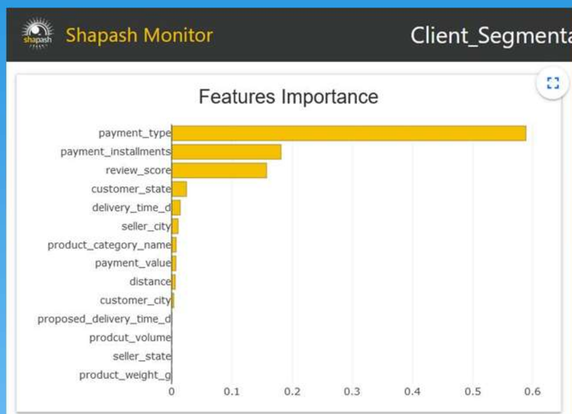
	review_score	product_weight_g	payment_installments	payment_value	distance	prodcut_volume	delivery_time_d	proposed_delivery_time_d	cluster_id
mean	4.815665	1397.321433	2.185003	118.609413	4.375147	11363.896509	9.07031	21.058925	3.0
75%	5.000000	1400.000000	3.000000	143.060000	5.853201	14400.000000	12.00000	26.000000	3.0

	review_score	product_weight_g	payment_installments	payment_value	distance	prodcut_volume	delivery_time_d	proposed_delivery_time_d	cluster_id
mean	4.716932	3395.37304	5.704413	244.421133	7.147640	22259.28566	11.667236	24.958713	4.0
75%	5.000000	3850.00000	8.000000	269.240000	9.247108	27237.00000	15.000000	30.000000	4.0

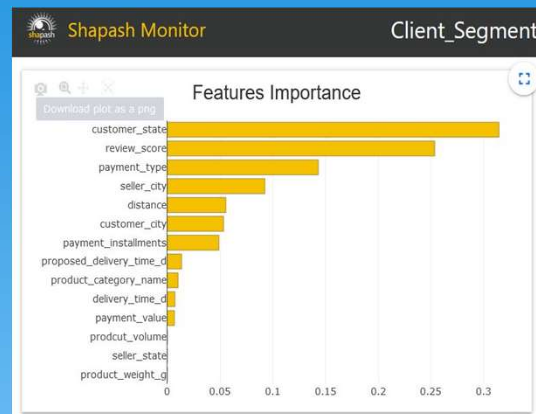
5- Supervised ML clustering for model explicability :

Feature importane for all dataset :

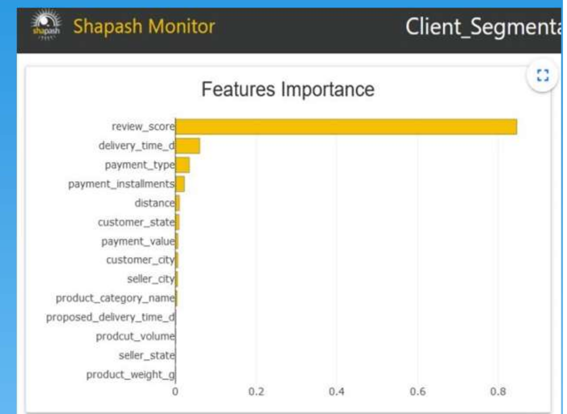
Class '0'



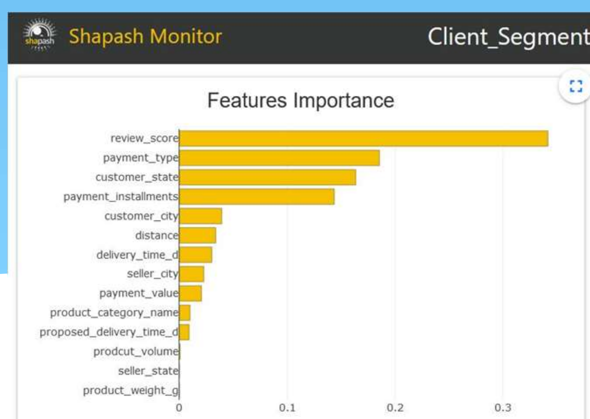
Class '1'



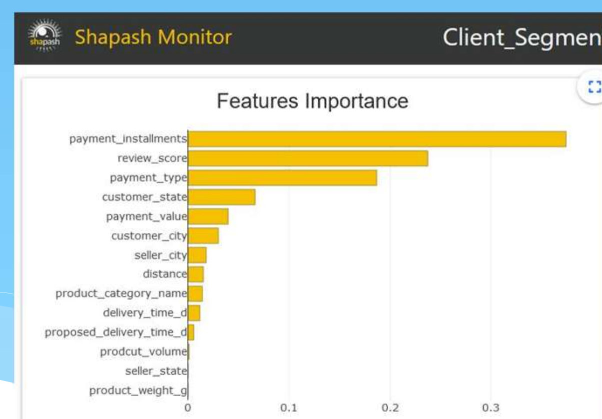
Class '2'



Class '3'



Class '4'



6- ARI score for stability assesement and maintenance's contract :

Visualisation of transactions distribution according to different time intervalls

30 days

	Interval	Count_clients	Date_limit
0	1	11	2016-10-04
1	2	294	2016-11-03
2	3	0	2016-12-03
3	4	1	2017-01-02
4	5	779	2017-02-01
5	6	1883	2017-03-03
6	7	2526	2017-04-02
7	8	2423	2017-05-02
8	9	3511	2017-06-01
9	10	3192	2017-07-01
10	11	3788	2017-07-31
11	12	4083	2017-08-30
12	13	4310	2017-09-29
13	14	4329	2017-10-29
14	15	6864	2017-11-28
15	16	6075	2017-12-28
16	17	6538	2018-01-27
17	18	6799	2018-02-26
18	19	7282	2018-03-28
19	20	6807	2018-04-27
20	21	6967	2018-05-27
21	22	5809	2018-06-26
22	23	5822	2018-07-26
23	24	7565	2018-08-25
24	25	257	2018-09-03

60 days

	Interval	Count_clients	Date_limit
0	1	305	2016-11-03
1	2	1	2017-01-02
2	3	2662	2017-03-03
3	4	4949	2017-05-02
4	5	6703	2017-07-01
5	6	7871	2017-08-30
6	7	8639	2017-10-29
7	8	12939	2017-12-28
8	9	13337	2018-02-26
9	10	14089	2018-04-27
10	11	12776	2018-06-26
11	12	13387	2018-08-25
12	13	257	2018-09-03

90 days

	Interval	Count_clients	Date_limit
0	1	305	2016-12-03
1	2	2663	2017-03-03
2	3	8460	2017-06-01
3	4	11063	2017-08-30
4	5	15503	2017-11-28
5	6	19412	2018-02-26
6	7	21056	2018-05-27
7	8	19196	2018-08-25
8	9	257	2018-09-03

120 days

	Interval	Count_clients	Date_limit
0	1	306	2017-01-02
1	2	7611	2017-05-02
2	3	14574	2017-08-30
3	4	21578	2017-12-28
4	5	27426	2018-04-27
5	6	26163	2018-08-25
6	7	257	2018-09-03

150 days

	Interval	Count_clients	Date_limit
0	1	1085	2017-02-01
1	2	13535	2017-07-01
2	3	23374	2017-11-28
3	4	33501	2018-04-27
4	5	26420	2018-09-03

180 days

	Interval	Count_clients	Date_limit
0	1	2968	2017-03-03
1	2	19523	2017-08-30
2	3	34915	2018-02-26
3	4	40252	2018-08-25
4	5	257	2018-09-03

210 days

	Interval	Count_clients	Date_limit
0	1	5494	2017-04-02
1	2	25636	2017-10-29
2	3	47332	2018-05-27
3	4	19453	2018-09-03

6- ARI score for stability assesement and maintenance's contract :

Visualisation of transactions distribution according to different time intervalls
11 runs have been made with a 5 ARI values for each
We got 55 ARI values for different time steps and different intervals

Shows all intervall's dates

Determine second date of first interval

```
2016-09-04
Provide delta time for first interval    :365
**********************************************
first interval : 2016-09-04      2017-09-04
Second interval: 2017-09-04      2017-11-03
Third interval : 2017-11-03      2018-01-02
4 th  interval : 2018-01-02      2018-03-03
5 th  interval : 2018-03-03      2018-05-02
6 th interval  : 2018-05-02      2018-07-01
**********************************************
**********************************************
first dataframe shape  : (23349, 14)
Second dataframe shape : (31922, 14)
Third dataframe shape  : (44785, 14)
4 th  dataframe shape  : (59048, 14)
5 th  dataframe shape  : (72855, 14)
6 th  dataframe shape  : (72855, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.27425739319733283
ARI 1 vs 3    : 0.34207605031628774
ARI 1 vs 4    : 0.523203607365514
ARI 1 vs 5    : 0.5291371441988411
ARI 1 vs 6    : 0.3153224875538858
```

```
2016-09-04
Provide delta time for first interval    :500
**********************************************
first interval : 2016-09-04      2018-01-17
Second interval: 2018-01-17      2018-03-18
Third interval : 2018-03-18      2018-05-17
4 th  interval : 2018-05-17      2018-07-16
5 th  interval : 2018-07-16      2018-09-14
6 th interval  : 2018-09-14      2018-11-13
**********************************************
**********************************************
first dataframe shape  : (48507, 14)
Second dataframe shape : (62473, 14)
Third dataframe shape  : (77083, 14)
4 th  dataframe shape  : (87838, 14)
5 th  dataframe shape  : (97916, 14)
6 th  dataframe shape  : (97916, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.298513582022155
ARI 1 vs 3    : 0.32487847487126853
ARI 1 vs 4    : 0.28011877549337927
ARI 1 vs 5    : 0.4201731227358364
ARI 1 vs 6    : 0.4201731227358364
```

```
2016-09-04
Provide delta time for first interval    :540
**********************************************
first interval : 2016-09-04      2018-02-26
Second interval: 2018-02-26      2018-04-12
Third interval : 2018-04-12      2018-05-27
4 th  interval : 2018-05-27      2018-07-11
5 th  interval : 2018-07-11      2018-08-25
6 th interval  : 2018-08-25      2018-10-09
**********************************************
**********************************************
first dataframe shape  : (57697, 14)
Second dataframe shape : (68179, 14)
Third dataframe shape  : (78562, 14)
4 th  dataframe shape  : (87011, 14)
5 th  dataframe shape  : (97727, 14)
6 th  dataframe shape  : (97727, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.4915708786399065
ARI 1 vs 3    : 0.2570469802522336
ARI 1 vs 4    : 0.4112206943718684
ARI 1 vs 5    : 0.4151658727666916
ARI 1 vs 6    : 0.3700473253509726
```

```
2016-09-04
Provide delta time for first interval    :365
**********************************************
first interval : 2016-09-04      2017-09-04
Second interval: 2017-09-04      2017-09-14
Third interval : 2017-09-14      2017-09-24
4 th  interval : 2017-09-24      2017-10-04
5 th  interval : 2017-10-04      2017-10-14
6 th interval  : 2017-10-14      2017-10-24
**********************************************
**********************************************
first dataframe shape  : (23349, 14)
Second dataframe shape : (24833, 14)
Third dataframe shape  : (26205, 14)
4 th  dataframe shape  : (27647, 14)
5 th  dataframe shape  : (29062, 14)
6 th  dataframe shape  : (29062, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.3414166169401056
ARI 1 vs 3    : 0.3510724599469801
ARI 1 vs 4    : 0.3124381294434958
ARI 1 vs 5    : 0.35521056809958973
ARI 1 vs 6    : 0.36651614816610406
```

```
2016-09-04
Provide delta time for first interval    :365
**********************************************
first interval : 2016-09-04      2017-09-04
Second interval: 2017-09-04      2017-09-05
Third interval : 2017-09-05      2017-09-06
4 th  interval : 2017-09-06      2017-09-07
5 th  interval : 2017-09-07      2017-09-08
6 th interval  : 2017-09-08      2017-09-09
**********************************************
**********************************************
first dataframe shape  : (23349, 14)
Second dataframe shape : (23501, 14)
Third dataframe shape  : (23639, 14)
4 th  dataframe shape  : (23735, 14)
5 th  dataframe shape  : (23850, 14)
6 th  dataframe shape  : (23850, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.38234835272739864
ARI 1 vs 3    : 0.4062058969870798
ARI 1 vs 4    : 0.37172033277459826
ARI 1 vs 5    : 0.2487881532321986
ARI 1 vs 6    : 0.4125392606299404
```

```
2016-09-04
Provide delta time for first interval    :30
**********************************************
first interval : 2016-09-04      2016-10-04
Second interval: 2016-10-04      2016-10-05
Third interval : 2016-10-05      2016-10-06
4 th  interval : 2016-10-06      2016-10-07
5 th  interval : 2016-10-07      2016-10-08
6 th interval  : 2016-10-08      2016-10-09
**********************************************
**********************************************
first dataframe shape  : (71, 14)
Second dataframe shape : (111, 14)
Third dataframe shape  : (160, 14)
4 th  dataframe shape  : (203, 14)
5 th  dataframe shape  : (243, 14)
6 th  dataframe shape  : (243, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.2989152074251135
ARI 1 vs 3    : 0.37945343545583704
ARI 1 vs 4    : 0.47563124287947434
ARI 1 vs 5    : 0.3970829518147888
ARI 1 vs 6    : 0.48876800333401615
```

```
2016-09-04
Provide delta time for first interval    :700
**********************************************
first interval : 2016-09-04      2018-08-05
Second interval: 2018-08-05      2018-08-06
Third interval : 2018-08-06      2018-08-07
4 th  interval : 2018-08-07      2018-08-08
5 th  interval : 2018-08-08      2018-08-09
6 th interval  : 2018-08-09      2018-08-10
**********************************************
**********************************************
first dataframe shape  : (92923, 14)
Second dataframe shape : (93295, 14)
Third dataframe shape  : (93659, 14)
4 th  dataframe shape  : (93973, 14)
5 th  dataframe shape  : (94256, 14)
6 th  dataframe shape  : (94256, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 0.4396645177481386
ARI 1 vs 3    : 0.8022930902032158
ARI 1 vs 4    : 0.5755140632425944
ARI 1 vs 5    : 0.5201550221222222
ARI 1 vs 6    : 0.3482121734904134
```

```
2016-09-04
Provide delta time for first interval    :120
**********************************************
first interval : 2016-09-04      2017-01-02
Second interval: 2017-01-02      2017-01-03
Third interval : 2017-01-03      2017-01-04
4 th  interval : 2017-01-04      2017-01-05
5 th  interval : 2017-01-05      2017-01-06
6 th interval  : 2017-01-06      2017-01-07
**********************************************
**********************************************
first dataframe shape  : (306, 14)
Second dataframe shape : (306, 14)
Third dataframe shape  : (306, 14)
4 th  dataframe shape  : (338, 14)
5 th  dataframe shape  : (342, 14)
6 th  dataframe shape  : (342, 14)
**********************************************
ARI results    :
ARI 1 vs 2    : 1.0
ARI 1 vs 3    : 1.0
ARI 1 vs 4    : 0.47591724818957826
ARI 1 vs 5    : 0.48062480598608465
ARI 1 vs 6    : 0.7223734414911048
```

6- ARI score for stability assesement and maintenance’s contract :

Visualisation of transactions distribution according to different time intervalls

```
2016-09-04
Provide delta time for first interval    :124
************************************************
first interval : 2016-09-04      2017-01-06
Second interval: 2017-01-06      2017-01-07
Third interval : 2017-01-07      2017-01-08
4 th  interval : 2017-01-08      2017-01-09
5 th  interval : 2017-01-09      2017-01-10
6 th interval  : 2017-01-10      2017-01-11
************************************************
************************************************
first dataframe shape  : (342, 14)
Second dataframe shape : (346, 14)
Third dataframe shape  : (352, 14)
4 th  dataframe shape  : (357, 14)
5 th  dataframe shape  : (363, 14)
6 th  dataframe shape  : (363, 14)
************************************************
ARI results    :
ARI 1 vs 2    : 0.5745246904540746
ARI 1 vs 3    : 0.7156490096712405
ARI 1 vs 4    : 0.581359017410691
ARI 1 vs 5    : 0.7307169133679626
ARI 1 vs 6    : 0.9407684074522004
```

```
2016-09-04
Provide delta time for first interval    :124
************************************************
first interval : 2016-09-04      2017-01-06
Second interval: 2017-01-06      2017-01-16
Third interval : 2017-01-16      2017-01-26
4 th  interval : 2017-01-26      2017-02-05
5 th  interval : 2017-02-05      2017-02-15
6 th interval  : 2017-02-15      2017-02-25
************************************************
************************************************
first dataframe shape  : (342, 14)
Second dataframe shape : (447, 14)
Third dataframe shape  : (846, 14)
4 th  dataframe shape  : (1417, 14)
5 th  dataframe shape  : (2152, 14)
6 th  dataframe shape  : (2152, 14)
************************************************
ARI results    :
ARI 1 vs 2    : 0.575009440881135
ARI 1 vs 3    : 0.5077397857096446
ARI 1 vs 4    : 0.5329412210015646
ARI 1 vs 5    : 0.5026841596200754
ARI 1 vs 6    : 0.39024013797149554
```

```
2016-09-04
Provide delta time for first interval    :128
************************************************
first interval : 2016-09-04      2017-01-10
Second interval: 2017-01-10      2017-01-11
Third interval : 2017-01-11      2017-01-12
4 th  interval : 2017-01-12      2017-01-13
5 th  interval : 2017-01-13      2017-01-14
6 th interval  : 2017-01-14      2017-01-15
************************************************
************************************************
first dataframe shape  : (363, 14)
Second dataframe shape : (375, 14)
Third dataframe shape  : (388, 14)
4 th  dataframe shape  : (398, 14)
5 th  dataframe shape  : (415, 14)
6 th  dataframe shape  : (415, 14)
************************************************
ARI results    :
ARI 1 vs 2    : 0.6807344820647852
ARI 1 vs 3    : 0.5007844718227913
ARI 1 vs 4    : 0.5345373008612018
ARI 1 vs 5    : 0.49312211387272575
ARI 1 vs 6    : 0.4942869655686988
```

Comments :

- Best ARI scores reached was 0.94 with stability even with small rate of time.
- According to all results found , Clustering has to be performed ‘daily’ for this dataset

Conclusions :

- Silouhette score for K means and K Prototypes are quite similar, we have choosen K prototye according to the importance of information provided by catgorical variables

- K Prototype offers a interesting client clustering which can be used for commercial and marketing actions for each cluster.

- RFM clustering is a good reflection of this dataset.

- Main majority of clients are in ‘touch and go’ purchasing mode.

- Reasons of lack of loyalty to this site have to be investigated deeply to take actions.

- It would have been far better if Dataset used for this project was different in terms of diversity and characterisation of clients patterns in order to let us measure how clustring approaches can provide more relevant informations