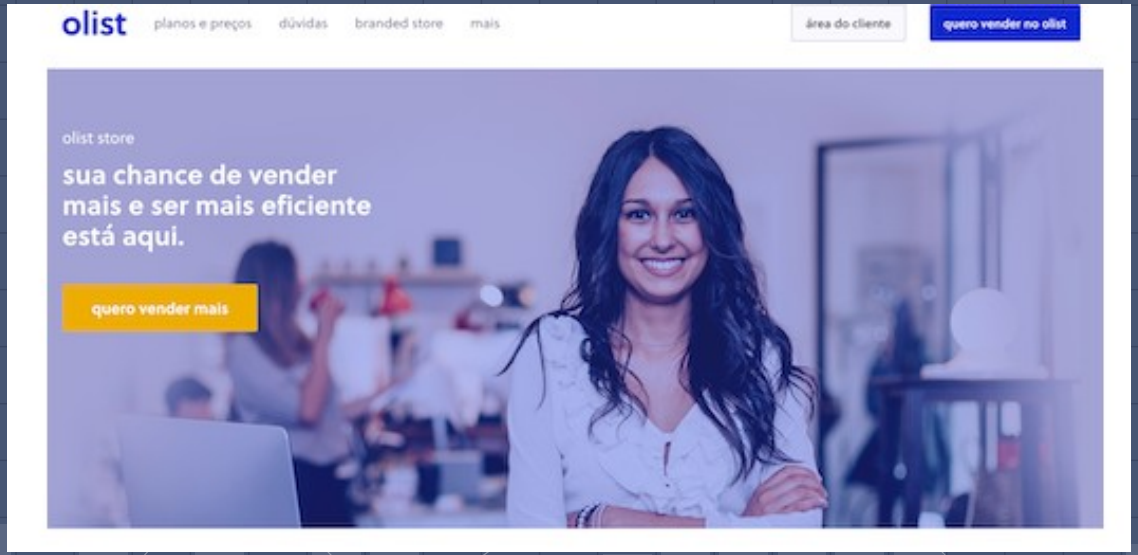
Segmentation des clients d'un site e-commerce

OpenClassrooms
Data Scientist.
Projet 5

Serge Davister
Mai 2023

En tant que consultant pour Olist, une entreprise brésilienne qui propose une solution de vente sur les marketplaces en ligne, je dois fournir à ses équipes d'e-commerce une description actionnable de la segmentation des clients et de sa logique sous-jacente pour une utilisation optimale, qu'elles pourront utiliser au quotidien pour leurs campagnes de communication, ainsi qu'une proposition de contrat de maintenance basée sur une analyse de la stabilité des segments au cours du temps.



Pour cette mission, Olist a fourni une base de données anonymisée comportant des informations sur l'historique de commandes, les produits achetés, les commentaires de satisfaction, et la localisation des clients depuis janvier 2017

Chargement et nettoyage des jeux de données

Il y a 9 tables .

1. Clients:

```
customer_id                99441
customer_unique_id         96096
customer_zip_code_prefix   14994
customer_city               4119
customer_state                27
```

Les clients :
Il n'y a pas de valeurs nulles dans la table.
Le customer_id sera utilisé lors des jointures. C'est un identifiant unique attribué à chaque commande
customer_unique_id: identifiant client

2. Géolocations :

```
geolocation_zip_code_prefix  1000163 non-null  int64
geolocation_lat              1000163 non-null  float64
geolocation_lng              1000163 non-null  float64
geolocation_city             1000163 non-null  object
geolocation_state            1000163 non-null  object
```

la géolocalisation:
Pas de valeurs nulles mais il y avait des doublons qui ont été supprimés 261.831 doublons
On peut utiliser zip_code_prefix pour jointure avec la table customers.
Je vais prendre une moyenne des variables latitude et longitude en groupant par geolocation_zip_code_prefix

```
geolocations.groupby(['geolocation_zip_code_prefix']).agg({'geolocation_lat':'mean','geolocation_lng':'mean',
                                                           'geolocation_city':'max','geolocation_state':'first'}).
```

```
geolocation_zip_code_prefix  19015 non-null  int64
geolocation_lat              19015 non-null  float64
geolocation_lng              19015 non-null  float64
geolocation_city             19015 non-null  object
geolocation_state            19015 non-null  object
```

3. Articles:

```
order_id                98666
order_item_id              21
product_id              32951
seller_id                3095
shipping_limit_date     93318
price                    5968
freight_value            6999
```

order_id : Identifiant unique de la commande
order_item_id : Identifiant des produits d'une même commande
product_id : Identifiant unique du produit
seller_id : Identifiant unique du vendeur
price : Prix de la ligne de commande
freight_value : Coût du fret

Chargement et nettoyage des jeux de données (suite)

4. Articles:

```
order_id              103886 non-null  object
payment_sequential    103886 non-null  int64
payment_type          103886 non-null  object
payment_installments  103886 non-null  int64
payment_value         103886 non-null  float64
```

order_id : identifiant de la commande
payment_type : moyen de paiement utilisé (voucher =coupon prépayé et boleto = moyen de paiement en espèces très utilisé au Brésil)
payment_sequential : séquence de paiement (une ligne par type de paiement)
payment_installments : Nombre de versements
payment_value : valeur du paiement

5. Avis clients:

```
review_id                99224 non-null  object
order_id                 99224 non-null  object
review_score             99224 non-null  int64
review_comment_title     11568 non-null  object
review_comment_message   40977 non-null  object
review_creation_date     99224 non-null  object
review_answer_timestamp  99224 non-null  object
```

```
1  reviews.nunique()
```

```
review_id                98410
order_id                 98673
review_score                 5
review_comment_title      4528
review_comment_message   36160
review_creation_date       636
review_answer_timestamp  98248
```

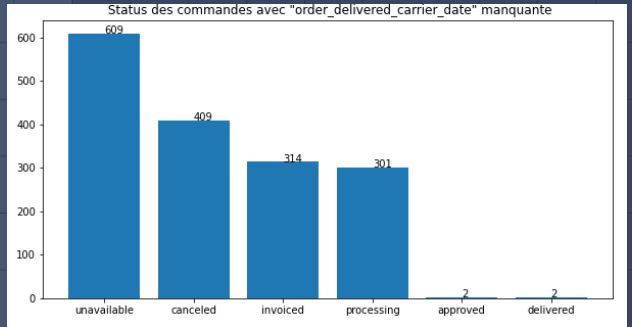
6. Commandes:

```
order_id                       99441 non-null  object
customer_id                    99441 non-null  object
order_status                   99441 non-null  object
order_purchase_timestamp       99441 non-null  object
order_approved_at              99281 non-null  object
order_delivered_carrier_date   97658 non-null  object
order_delivered_customer_date  96476 non-null  object
order_estimated_delivery_date  99441 non-null  object
```

```
order_id                       99441
customer_id                    99441
order_status                       8
order_purchase_timestamp       98875
order_approved_at              90733
order_delivered_carrier_date   81018
order_delivered_customer_date  95664
order_estimated_delivery_date    459
```

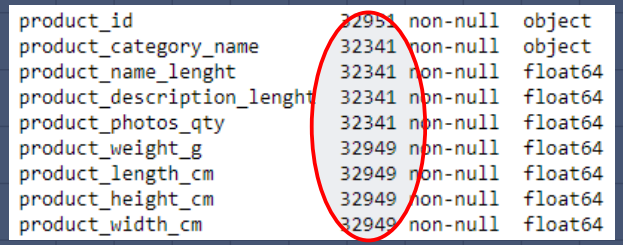
order_id: identifiant commande
customer_id : identifiant client unique attribué à chaque commande
order_status : status de la commande
order_purchase_timestamp : date et heure de commande
order_approved_at : date et heure d'approbation
order_delivered_carrier_date : date et heure prise en charge par le transporteur
order_delivered_customer_date :date et heure commande livrée
order_estimated_delivery_date :date estimée de livraison chez le client

```
Il y a 1637  valeurs manquantes dans order_delivered_carrier_date
```



Chargement et nettoyage des jeux de données (suite)

7. Produits:



product_id	32951	non-null	object
product_category_name	32341	non-null	object
product_name_lenght	32341	non-null	float64
product_description_lenght	32341	non-null	float64
product_photos_qty	32341	non-null	float64
product_weight_g	32949	non-null	float64
product_length_cm	32949	non-null	float64
product_height_cm	32949	non-null	float64
product_width_cm	32949	non-null	float64

il y a des produits qui n'ont ni catégorie, ni nom , ni description , ni photo mais qui ont des dimensions , des poids. Il y a également 2 produits sans dimensions ni poids. Nous les conservons et complétons les valeurs manquantes par "n.d" (non disponible)

8. Vendeurs: ce jeu de données ne sera pas pris en compte car l'étude porte sur les clients

9. Traduction des catégories: Cette table ne sera pas utilisée car j'ai conservé les catégories dans la langue du client.

Jointure des jeux de données customers et orders


```
customers_orders = customers.merge(orders, on='customer_id',how='inner')
```

Je conserve uniquement les commandes livrées (status == 'delivered' & 'order_delivered_customer_date'. NotNull)

```
customers_orders=customers_orders.loc[(customers_orders['order_status']=="delivered")&\
                                      (customers_orders['order_delivered_customer_date'].notnull())]
```

```
# supprimer colonnes non conservées pour la suite
list_to_del=['order_status','order_approved_at','order_delivered_carrier_date','order_estimated_delivery_date']
customers_orders.drop(columns=list_to_del,inplace=True)
```

```
les variables contenant des dates
order_purchase_timestamp']=pd.to_datetime(customers_orders['order_purchase_timestamp'].astype('datetime64[ns]'))
order_delivered_customer_date']=pd.to_datetime(customers_orders['order_delivered_customer_date'].astype('datetime64[ns]'))
```

Jointure du dataset obtenu avec la table items

```
items_=customers_orders.merge(items, on='order_id',how='inner')
```

```
# colonnes non conservées
list_to_del=['seller_id','shipping_limit_date']
items_.drop(columns=list_to_del,inplace=True)
```

Jointure du dataset obtenu avec la table payments

Nous allons ajouter une colonne qui comptera le nombre de moyens de paiement différents utilisés pour le réglement de la commande et une colonne qui totalisera le nombre d'échéances de la commande

```
payments_grby=payments.groupby(by="order_id").agg(
    {"payment_sequential": 'count',
     "payment_installments": 'sum'})
```

```
# On ne prend en compte que les commandes qui ont fait l'objet de payement (inner)
items_ = pd.merge(items_, payments_grby,
                  how="inner",
                  on="order_id")
items_ = items_.rename(columns={
    "payment_sequential": "payment_sequential_nbre",
    "payment_installments": "payment_installments_sum"})
```

Jointure du dataset obtenu avec la table reviews

Je vais ajouter une variable qui indiquera si la commande a été commentée(oui =True et non =False) et une variable qui donnera le score moyen de la commande.

```
reviews_grby = reviews.groupby("order_id").agg({
    "review_id": "count",
    "review_score": "last"})
```

```
items_ = pd.merge(items_, reviews_grby,
                  how="left",
                  on="order_id")
items_ = items_.rename(columns={
    "review_id": "note","review_score":"score"})
items_["note"] = np.where(items_["note"] == 1,
                                  True, False)
```

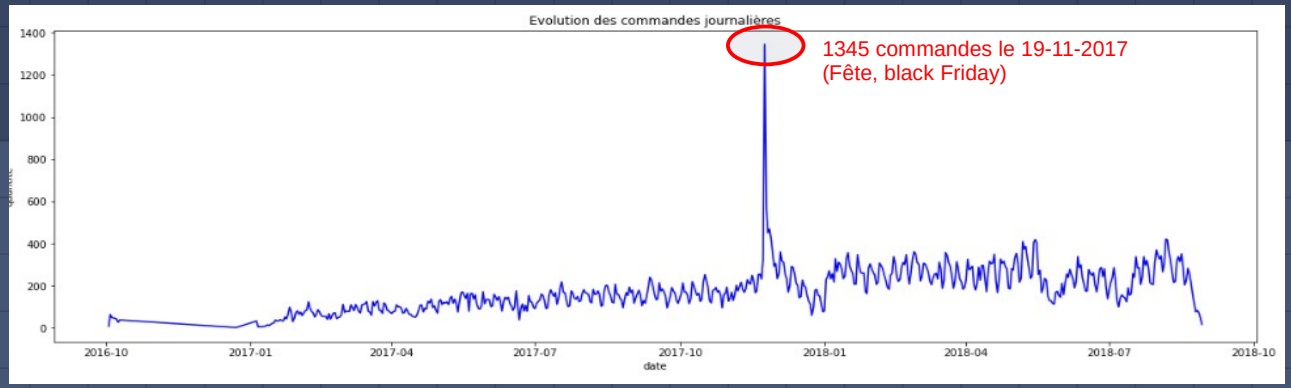
Jointure du dataset obtenu avec la table products

```
df = pd.merge(items_, products,
              how="left",
              on="product_id")
df.info()
```

```
customer_id                    110171 non-null  object
customer_unique_id             110171 non-null  object
customer_zip_code_prefix       110171 non-null  int64
customer_city                  110171 non-null  object
customer_state                 110171 non-null  object
order_id                       110171 non-null  object
order_purchase_timestamp       110171 non-null  datetime64[ns]
order_delivered_customer_date  110171 non-null  datetime64[ns]
order_item_id                  110171 non-null  int64
product_id                     110171 non-null  object
price                          110171 non-null  float64
freight_value                  110171 non-null  float64
payment_sequential_nbre        110171 non-null  int64
payment_installments_sum       110171 non-null  int64
note                           110171 non-null  bool
score                          109344 non-null  float64
product_category_name          110171 non-null  object
product_name_lenght            110171 non-null  float64
product_description_lenght     110171 non-null  float64
product_photos_qty             110171 non-null  float64
product_weight_g               110171 non-null  float64
product_length_cm              110171 non-null  float64
product_height_cm              110171 non-null  float64
product_width_cm               110171 non-null  float64
```

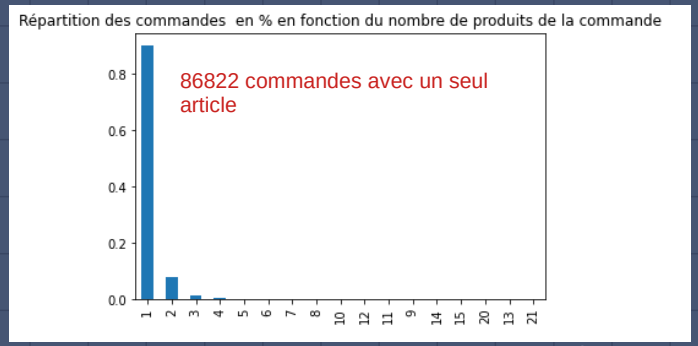
EDA

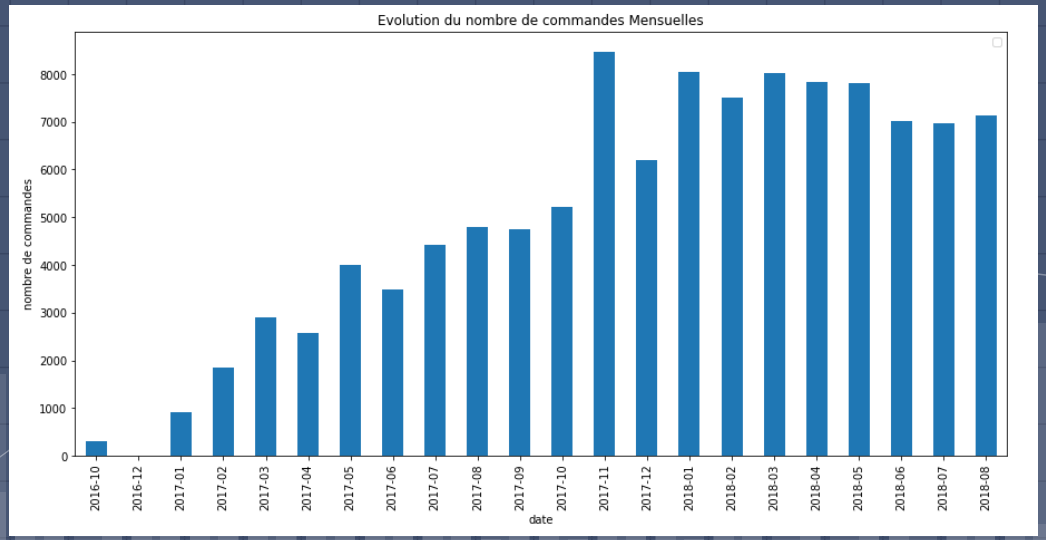
Les commandes



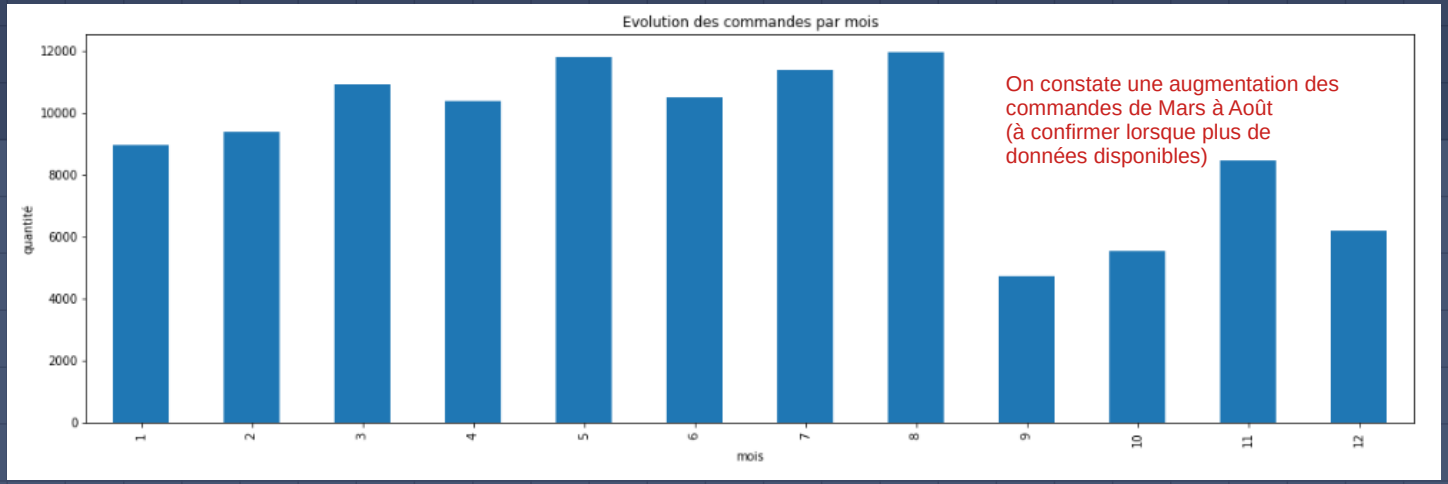

```
1  orders_day.describe()
```

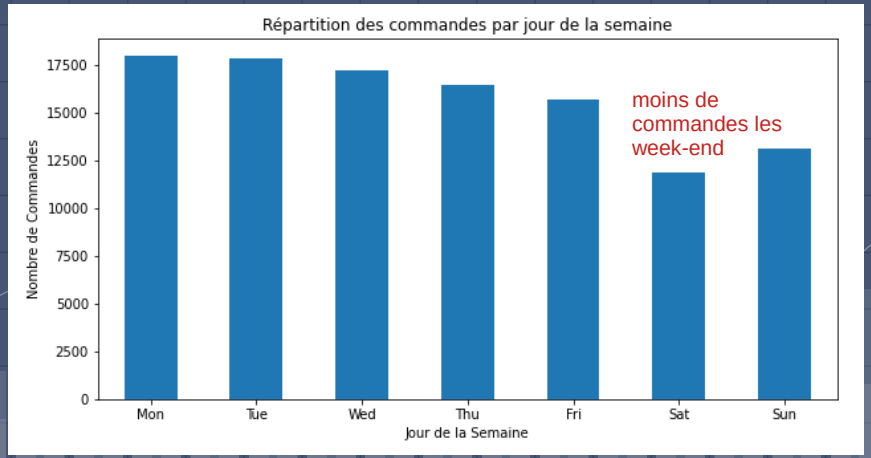
count	611.000000
mean	180.312602
std	103.692019
min	1.000000
25%	110.000000
50%	167.000000
75%	243.500000
max	1345.000000



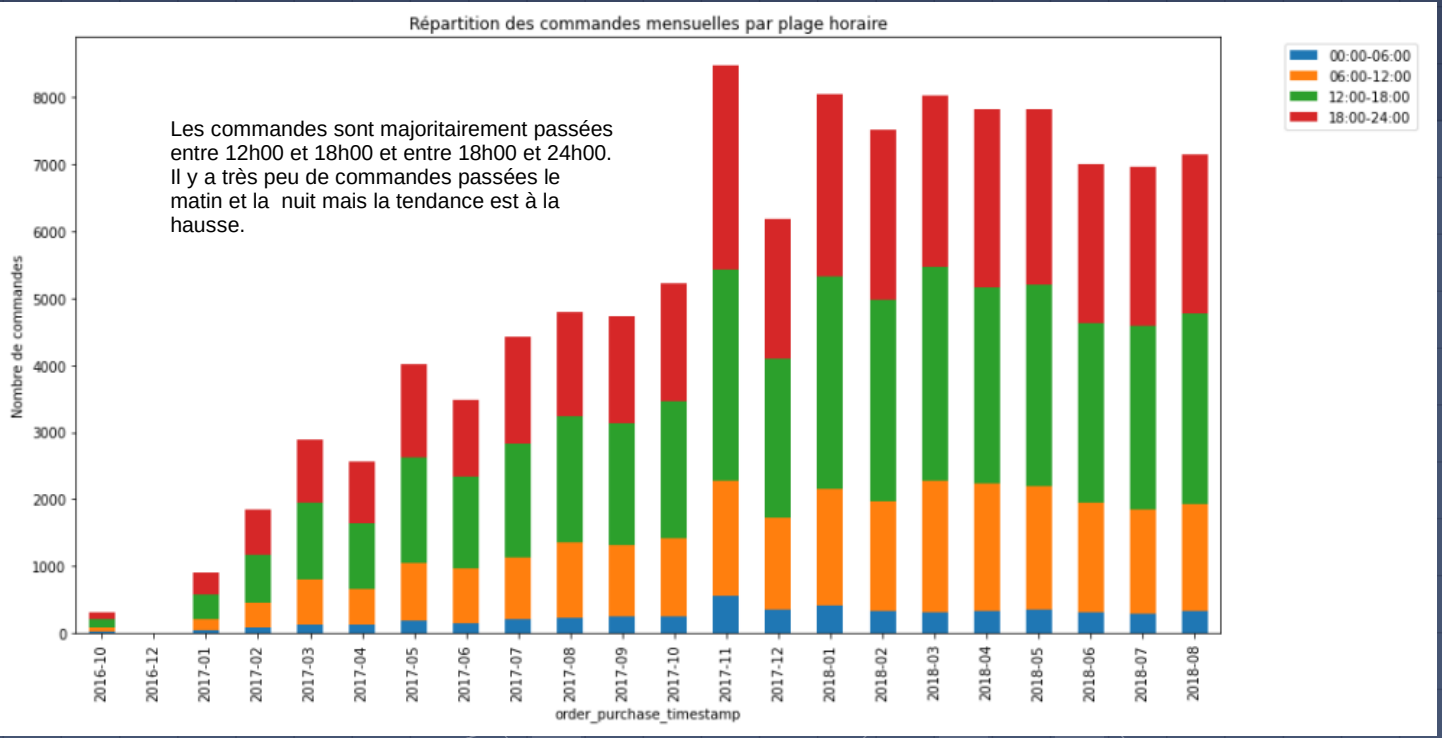


Les commandes





Les commandes



La segmentation des clients

2.1 calcul de la récence

le temps écoulé depuis le dernier achat

```
r_ = df.groupby(by='customer_unique_id',
                        as_index=False)['order_purchase_timestamp'].max()
r_.columns = ['customer_unique_id', 'last_purchase_date']
recent_date = r_['last_purchase_date'].max()
r_['recency'] = r_['last_purchase_date'].apply(
    lambda x: (recent_date - x).days)
r_.sort_values(by='recency',ascending=True).head()
```

	customer_unique_id	last_purchase_date	recency
66809	b701bebbdf478f5500348f03aff62121	2018-08-29 14:52:00	0
71658	c45221bb4573f66bdd4daf43fe2d4b3b	2018-08-28 19:27:43	0
20005	36a5c01d940c382346247b3e6c485c2d	2018-08-28 22:51:54	0
61029	a712a430955027da5bc257a10073a390	2018-08-28 21:56:30	0
91700	fb7e29c65321441231990afc201c1b14	2018-08-28 19:32:05	0

2.2 Calcul de la fréquence

Nous pouvons calculer la fréquence des achats des clients

```
f_ = df.groupby(
    by=['customer_unique_id'], as_index=False)['order_id'].count()
f_.columns = ['customer_unique_id', 'frequency']
f_.sort_values(by='frequency',ascending=False).head()
```

	customer_unique_id	frequency
73112	c8460e4251689ba205045f3ea17884a1	24
25304	4546caea018ad8c692964e3382debd19	21
71552	c402f431464c72e27330a67f7b94d4fb	20
38573	698e1cf81d01a3d389d96145f7fa6df8	20
5622	0f5ac8d5c31de21d2f25e24be15bbffb	18

2.3 Calcul du montant

nous pouvons calculer le montant des achats des clients chez Olist

```
m_ = df.groupby(by='customer_unique_id', as_index=False)['price'].sum()
m_.columns = ['customer_unique_id', 'monetary_value']
m_.sort_values(by='monetary_value',ascending=False).head()
```

	customer_unique_id	monetary_value
3724	0a0a92112bd4c708ca5fde585afaa872	13440.0
79617	da122df9eeddfedc1dc1f5349a1a690c	7388.0
43161	763c8b1c9c68a0229c42c9fc6f662b93	7160.0
80444	dc4802a71eae9be1dd28f5d788ceb526	6735.0
25431	459bef486812aa25204be022145caa62	6729.0

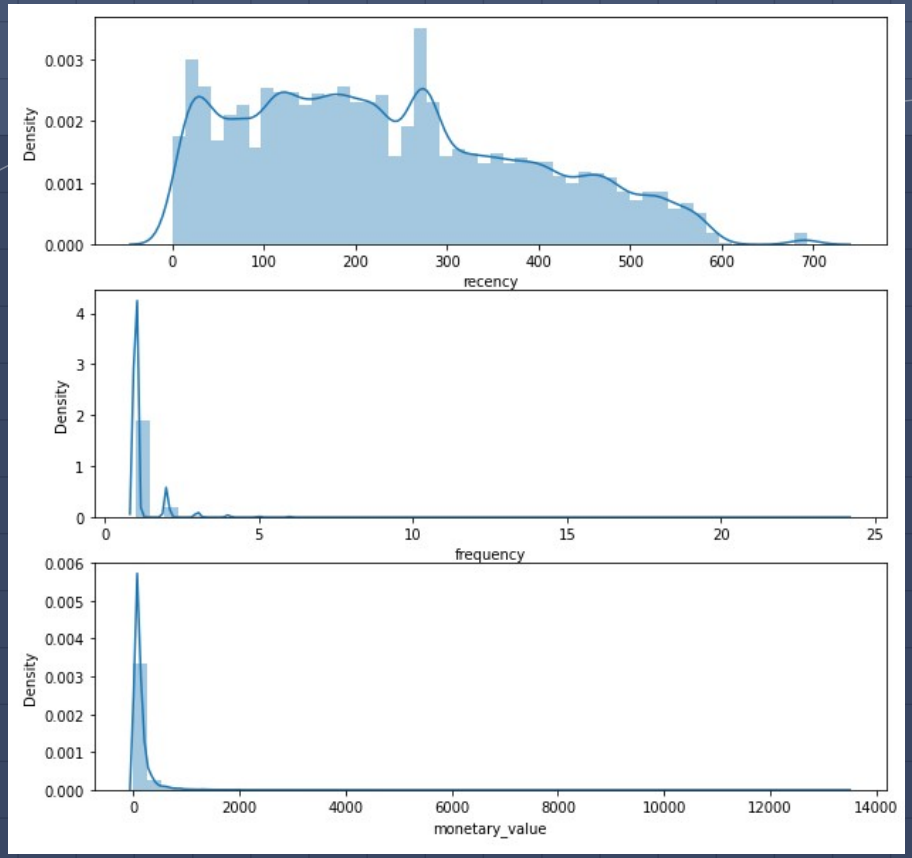
2.4 dataframe RFM

je vais fusionner les dataframes

```
rfm_ = r_.merge(f_, on='customer_unique_id').merge(m_,on='customer_unique_id')\
.drop(columns='last_purchase_date')
rfm_.head()
```

	customer_unique_id	recency	frequency	monetary_value
0	0000366f3b9a7992bf8c76cfdf3221e2	111	1	129.90
1	0000b849f77a49e4a4ce2b2a4ca5be3f	114	1	18.90
2	0000f46a3911fa3c0805444483337064	536	1	69.00
3	0000f6ccb0745a6a4b88665a16c9f078	320	1	25.99
4	0004aac84e0df4da2b147fca70cf8255	287	1	180.00

Les distributions RFM



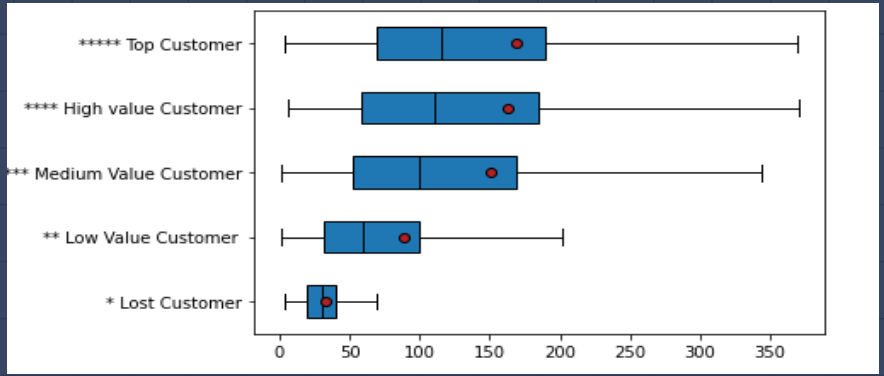
	recency	frequency	monetary_value
count	93336.000000	93336.000000	93336.000000
mean	236.899792	1.180370	141.622134
std	152.545336	0.620885	215.714177
min	0.000000	1.000000	0.850000
25%	113.000000	1.000000	47.650000
50%	218.000000	1.000000	89.700000
75%	345.000000	1.000000	154.712500
max	694.000000	24.000000	13440.000000

Le nombre de clients avec une commande unique est de 87.56 %. La majorité des clients du dataset a un achat unique , cela va influencer la segmentation.

Les segments clients

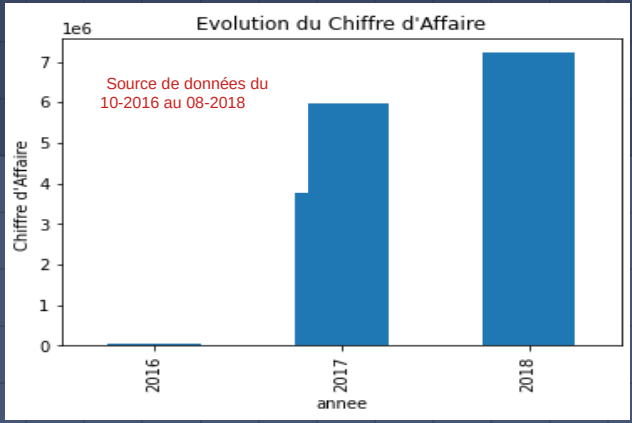
	customer_unique_id	recency	frequency	monetary_value	r_rank_norm	f_rank_norm	m_rank_norm	tot_rank	R	F	M	RFM	RFM_score
20532	381917c1951b8e044dd51544f465db6d	526	1	31.00	4.338205	43.782678	14.843683	21.0	1	1	1	111	1.0
5923	1019ed18792a025e05472c55766c34a0	428	1	24.90	13.962946	43.782678	8.742072	22.0	1	1	1	111	1.0
19731	35e32af235030874a6b6ccf5f328e8ac	420	1	35.99	14.820728	43.782678	17.867704	25.0	1	1	1	111	1.0
13581	25126926be8dbdb3264285caa7d0c40c	438	1	39.90	13.066051	43.782678	20.240850	26.0	1	1	1	111	1.0
19788	36023a7f667578e554e3a308c9d17598	449	1	39.90	11.800540	43.782678	20.240850	25.0	1	1	1	111	1.0

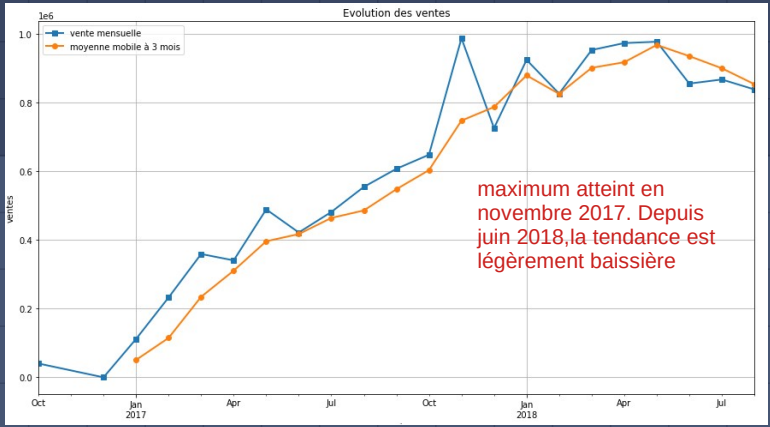
```
rfm_['client_segment']= np.where(rfm_['RFM_score'] >
                                 4.5, "***** Top Customer",
                                  (np.where(
                                    rfm_['RFM_score'] > 4,
                                    "**** High value Customer",
                                    (np.where(
                                        rfm_['RFM_score'] > 3,
                                 "*** Medium Value Customer",
                                 np.where(rfm_['RFM_score'] > 1.5,
                                '** Low Value Customer ', '* Lost Customer'))))))
```

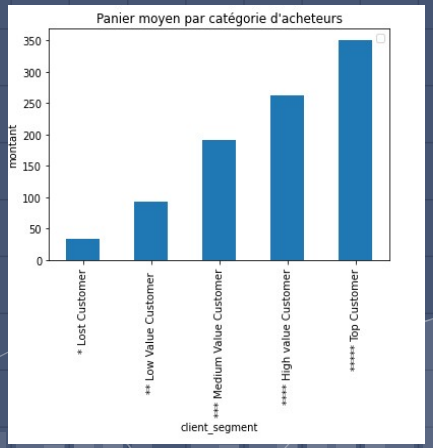


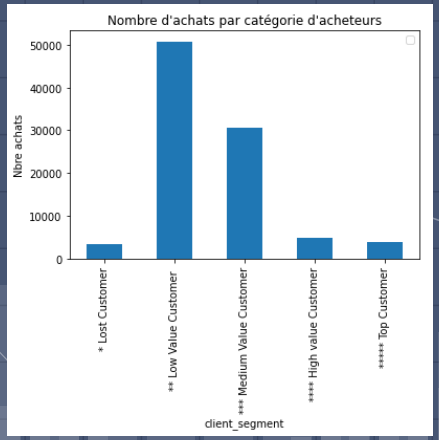
	recency	frequency			monetary_value		
	mean	min	mean	max	mean	count	sum
client_segment							
* Lost Customer	436.0	1	1.0	1	33.1	3440	113984.4
** Low Value Customer	281.4	1	1.0	21	92.1	50810	4681185.7
*** Medium Value Customer	179.8	1	1.3	20	191.2	30502	5832548.8
**** High value Customer	115.6	1	1.6	20	261.8	4755	1244900.9
***** Top Customer	73.3	1	2.1	24	351.5	3829	1345823.8

Les ventes , chiffre d'affaires et tendance

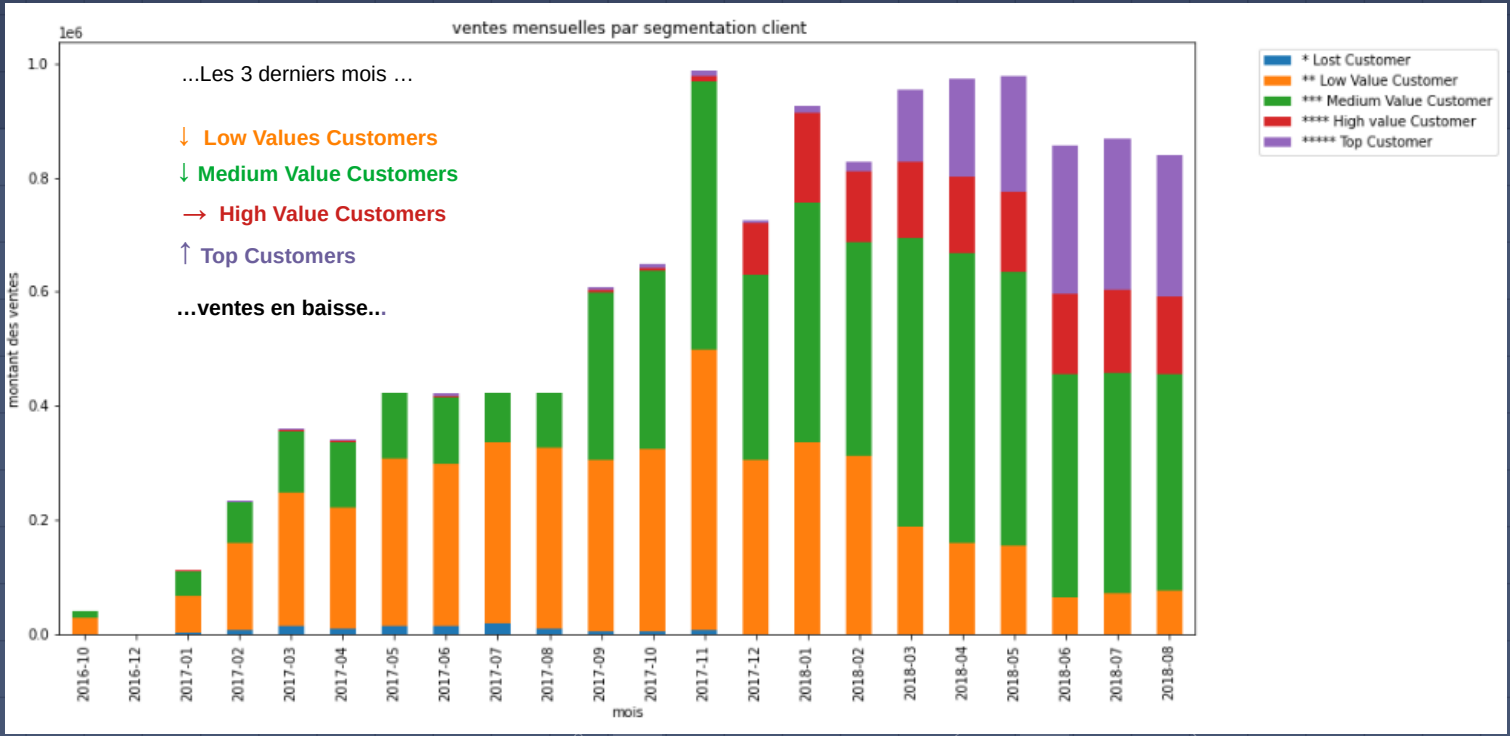




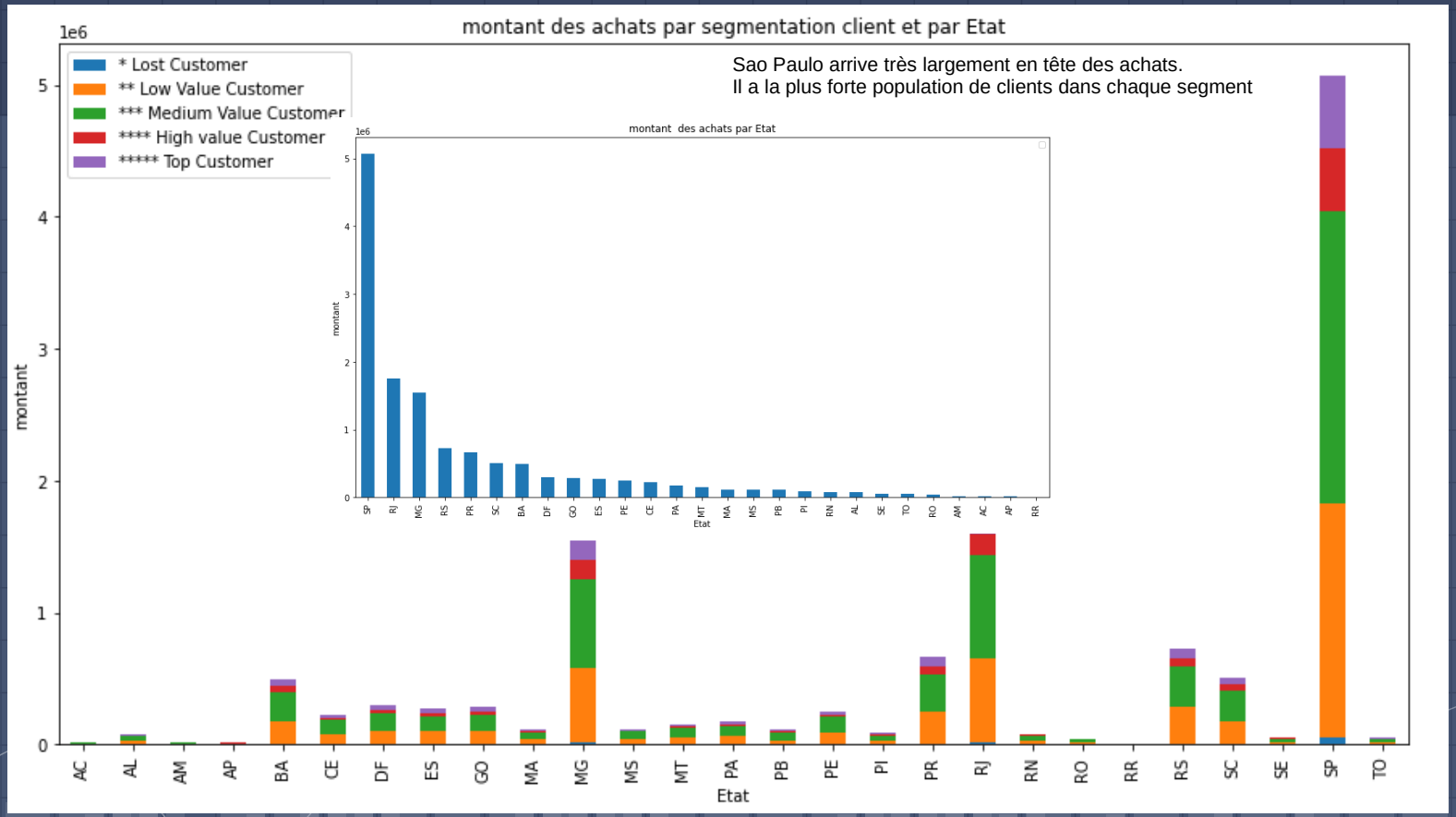




Evolution des ventes par segmentation client



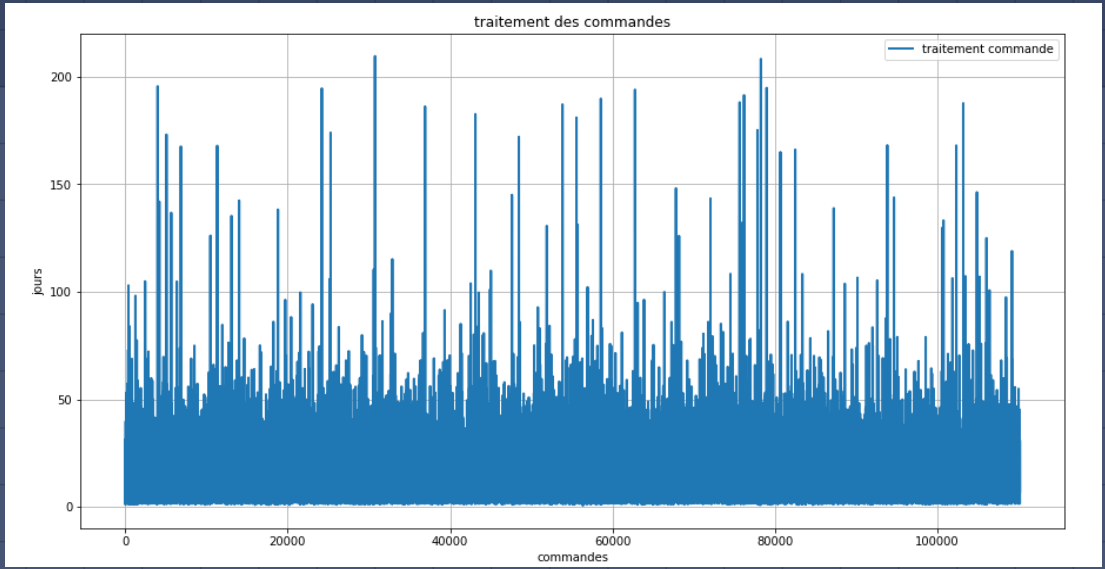
Répartition des ventes par segmentation client et par état

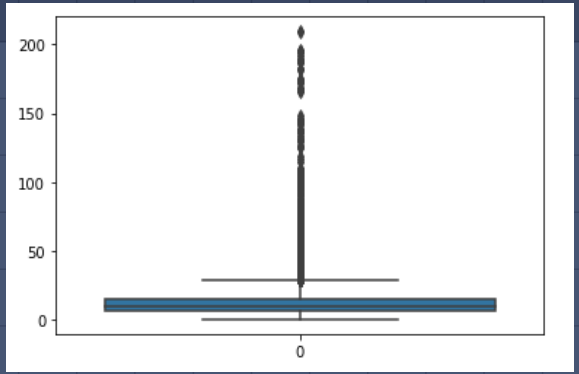


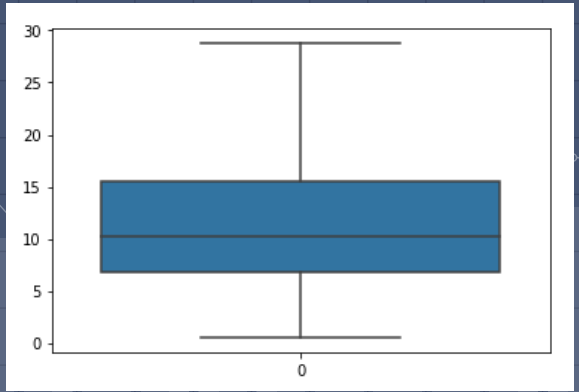
Sao Paulo arrive très largement en tête des achats.
Il a la plus forte population de clients dans chaque segment

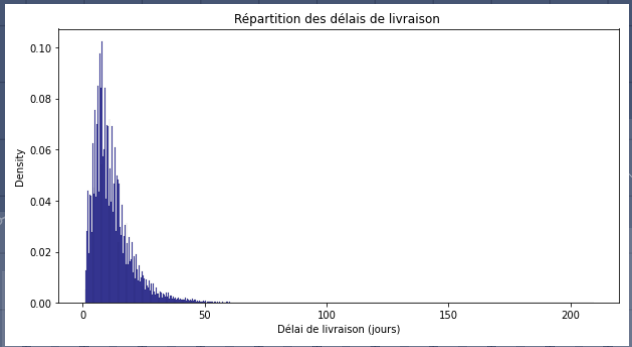
Le traitement des commandes


```
df['traitement']=(df['order_delivered_customer_date']-df['order_purchase_timestamp'])/timedelta(days=1)
```



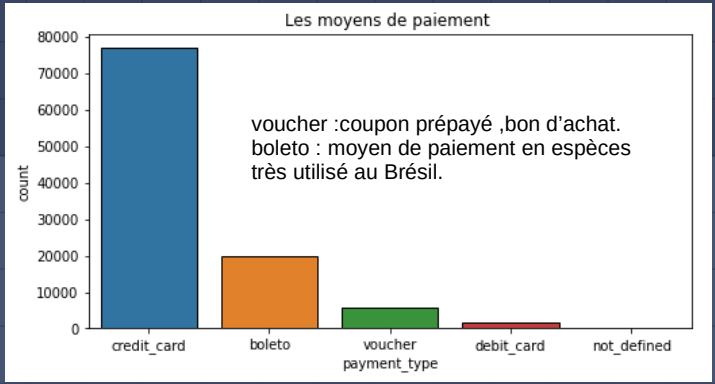


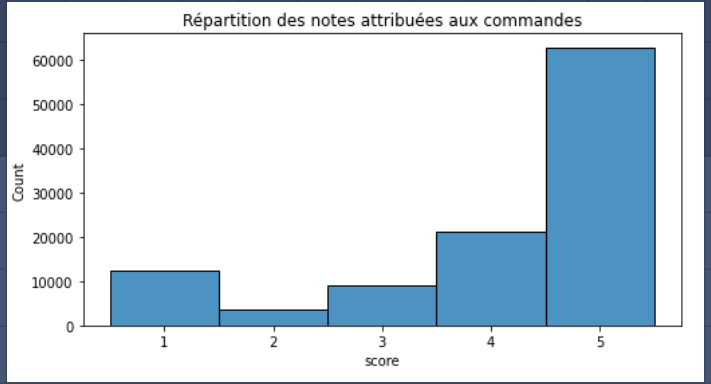




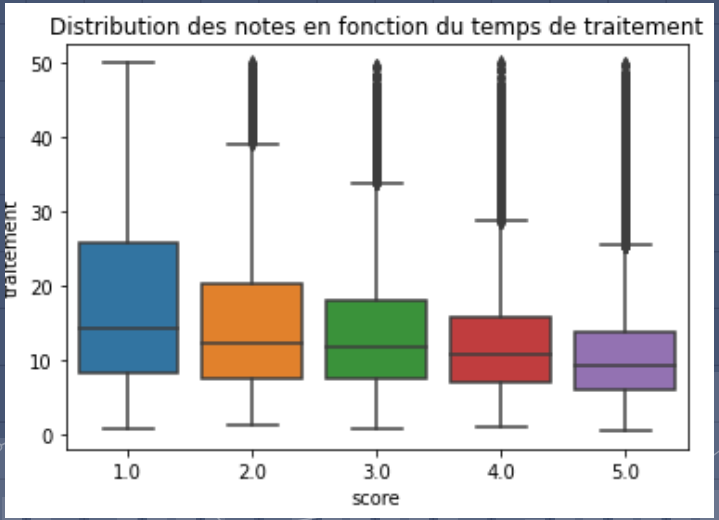
count	110171.0
mean	12.0
std	9.0
min	1.0
25%	7.0
50%	10.0
75%	16.0
max	210.0

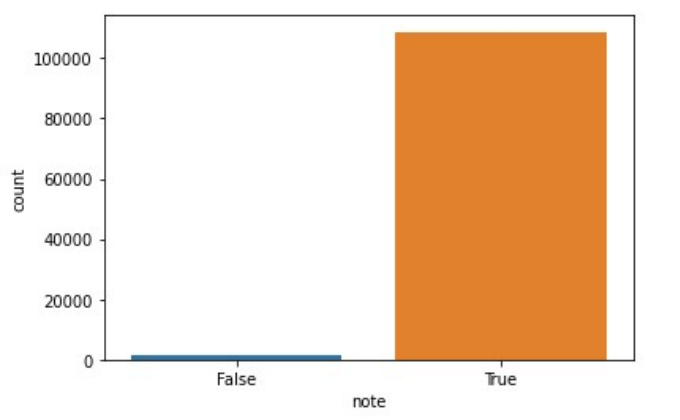
Les moyens de paiement et avis client





count	109344.000000
mean	4.081614
std	1.347328
min	1.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000





En moyenne plus le temps de traitement est court plus la note est élevée

Création du dataset de travail

0	customer_id	110171 non-null	object
1	customer_unique_id	110171 non-null	object
2	customer_zip_code_prefix	110171 non-null	int64
3	customer_city	110171 non-null	object
4	customer_state	110171 non-null	object
5	order_id	110171 non-null	object
6	order_purchase_timestamp	110171 non-null	datetime64[ns]
7	order_delivered_customer_date	110171 non-null	datetime64[ns]
8	order_item_id	110171 non-null	int64
9	product_id	110171 non-null	object
10	price	110171 non-null	float64
11	freight_value	110171 non-null	float64
12	payment_sequential_nbre	110171 non-null	int64
13	payment_installments_sum	110171 non-null	int64
14	note	110171 non-null	bool
15	score	109344 non-null	float64
16	product_category_name	110171 non-null	object
17	product_name_lenght	110171 non-null	float64
18	product_description_lenght	110171 non-null	float64
19	product_photos_qty	110171 non-null	float64
20	product_weight_g	110171 non-null	float64
21	product_length_cm	110171 non-null	float64
22	product_height_cm	110171 non-null	float64
23	product_width_cm	110171 non-null	float64
24	date	110171 non-null	object
25	mois	110171 non-null	int64
26	annee	110171 non-null	int64
27	mois_an	110171 non-null	period[M]
28	plage_horaire	110171 non-null	object
29	traitement	110171 non-null	float64
30	recency	110171 non-null	int64
31	frequency	110171 non-null	int64
32	monetary_value	110171 non-null	float64
33	RFM	110171 non-null	int32
34	RFM_score	110171 non-null	float64
35	client_segment	110171 non-null	object

imputer les valeurs manquantes de score avec la valeur médian

```
1  df.fillna(df['score'].median(),inplace=True)
```

```
list_to_del=['customer_city','customer_state',"order_purchase_timestamp",
             'order_delivered_customer_date',"order_item_id","product_id","product_name_lenght",
              "product_description_lenght","product_photos_qty","product_weight_g",
              'product_length_cm','product_height_cm','product_width_cm','date','mois','annee',
              'mois_an','plage_horaire',"client_segment"
             ]
```

```
data.drop(columns=list_to_del,inplace=True)
```

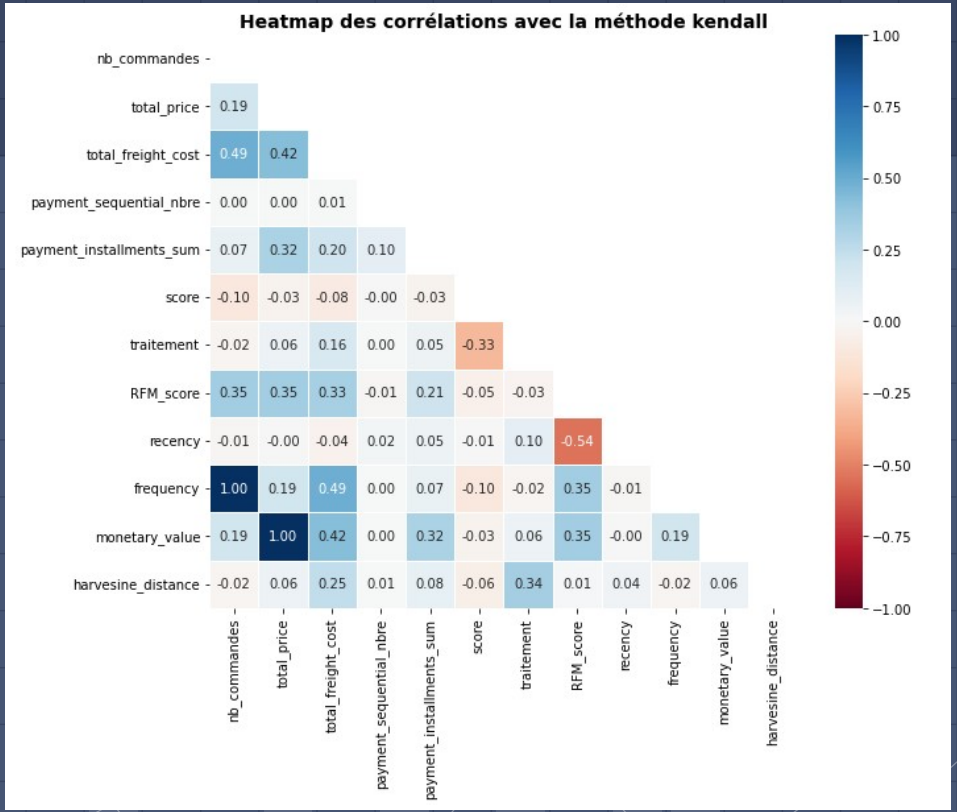
jointure pour récupérer les coordonnées moyennes des villes

```
data=pd.merge(data, geos.iloc[:,0:3],how="left",
              left_on="customer_zip_code_prefix",right_on="geolocation_zip_code_prefix")
```

Pour exploiter au mieux les données je vais calculer la distance Haversine entre le siège de Olist et l état où réside le client (latitude et longitude moyenne)

# Olist geolocalisation
olist_lat = -25.430182
olist_lon = -49.292507

Création du dataset de travail




```
1 df_cust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 93082 entries, 0 to 93335
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   customer_unique_id      93082 non-null  object
 1   nb_commandes            93082 non-null  int64
 2   total_price             93082 non-null  float64
 3   total_freight_cost      93082 non-null  float64
 4   payment_sequential_nbre 93082 non-null  float64
 5   payment_installments_sum 93082 non-null  float64
 6   score                   93082 non-null  float64
 7   traitement              93082 non-null  float64
 8   RFM_score               93082 non-null  float64
 9   recency                 93082 non-null  int64
 10  frequency               93082 non-null  int64
 11  monetary_value          93082 non-null  float64
 12  harvesine_distance      93082 non-null  float64
dtypes: float64(9), int64(3), object(1)
memory usage: 9.9+ MB
```

Il y a deux corrélations parfaites entre frequency et nb_commandes et entre monetary_value et total_price. Je vais enlever nb_commandes et total_price

Application de l'algorithme du K-Means

```
#Transformation en array Numpy
df.set_index('customer_unique_id', inplace=True)
X = df.values
X.shape
```

```
(93082, 10)
```

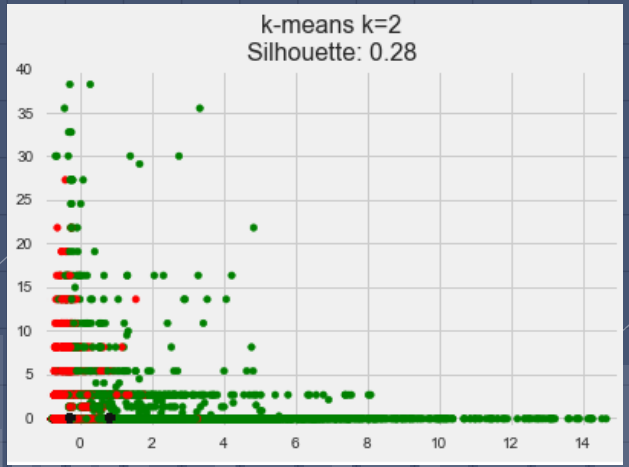
1.standardisation des données de grandeurs très différentes.

```
std_scale = StandardScaler().fit(X)
X_scaled = std_scale.transform(X)
```

2. initier algorithme kmeans

```
#  initialiser kmeans avec k=2
kmeans = KMeans( n_clusters=2,n_init=10,random_state=42)
kmeans.fit(X_scaled)
kmeans.predict(X_scaled)  #   ou kmeans.labels_
# calcule  silhouette score
kmeans_silhouette = silhouette_score(
    X_scaled, kmeans.labels_
).round(2)
```

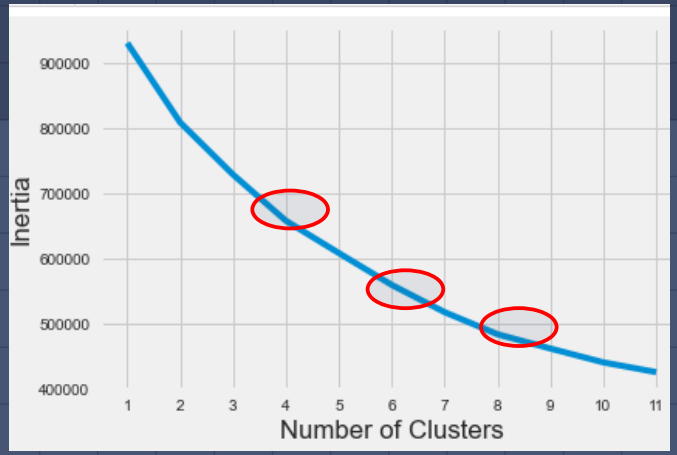
3. définir valeur de k : il faut un k idéalement entre 3 et 8 (segmentation clients) et avec un silhouette score important



Application de l'algorithme du K-Means

3.1 méthode du coude

```
# liste Inertia pour stocker les valeurs pour chaque k
inertia = []
dict_kmeans = {}
for k in range(1, 12):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs,)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
    dict_kmeans[k] = kmeans
```



Il y a un point idéal où la courbe commence à se plier, connue sous le nom de point de coude . La valeur x de ce point est considérée comme un compromis raisonnable entre l'erreur et le nombre de grappes. Ici il y a un point légèrement marqué à k=4 ,k=6 et k=8

3.2 coéfficient de silhouette

C'est une mesure de la cohésion et de la séparation des clusters. Il quantifie à quel point un point de données s'intègre dans son cluster attribué en fonction de deux facteurs : la proximité du point de données avec d'autres points du cluster et la distance entre le point de données et les points d'autres clusters. Les valeurs du coefficient de silhouette sont comprises entre -1et 1. Des nombres plus grands indiquent que les échantillons sont plus proches de leurs grappes qu'ils ne le sont des autres grappes.

```
# A list holds the silhouette coefficients for each k
silhouette_coefficients = []

#  Start at 2 clusters for silhouette coefficient
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(X_scaled)
    score = silhouette_score(X_scaled, kmeans.labels_)
    silhouette_coefficients.append(score)
```

SSE est maximum pour k =9.

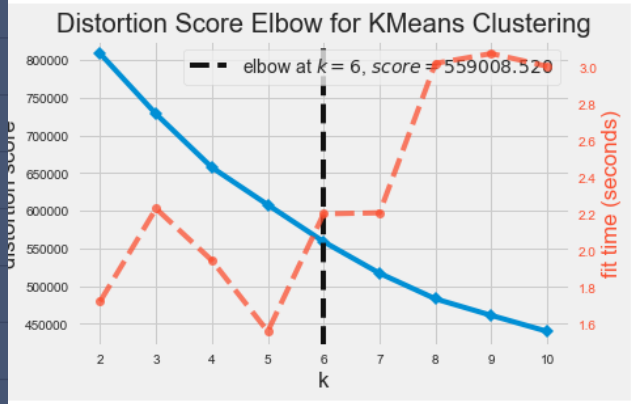
Application de l'algorithme du K-Means

3.3 ElbowVisualizer et SilhouetteElbowVisualizer : 2 outils de la librairie sklearn

```
numerical_features = list(df.select_dtypes(include=['int64','float64', 'uint8']).columns)
numerical_features
```

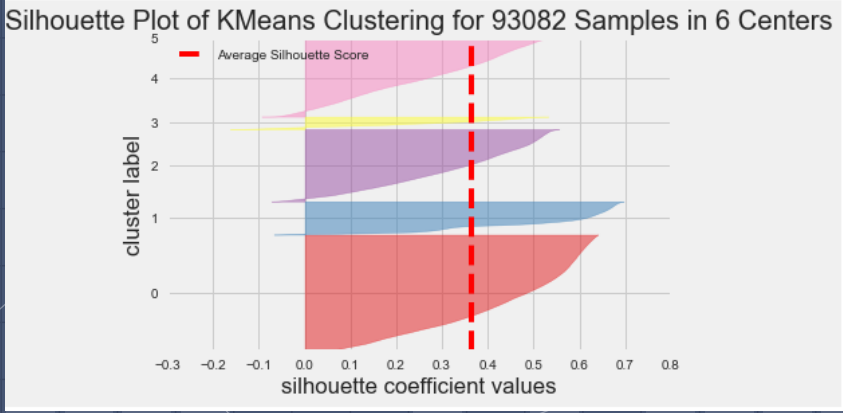
```
scaler = StandardScaler()
preprocessor = ColumnTransformer([
    ('scaler', scaler, numerical_features)])
```

```
kmeans_visualizer = Pipeline([
    ("preprocessor", preprocessor),
    ("kelbowvisualizer", KElbowVisualizer(KMeans(random_state=42,n_init=10),K=(4,12)))])
kmeans_visualizer.fit(df)
kmeans_visualizer.named_steps['kelbowvisualizer'].show()
```



Le nombre de clusters est ***k* = 6**

```
model_6clust = KMeans(n_clusters = 6, random_state=42,n_init=10)

sil_visualizer = SilhouetteVisualizer(model_6clust)
sil_visualizer.fit(df)
sil_visualizer.show()
```



Les 6 clusters ont un score de silhouette supérieur au score global.
Les scores sont compris entre 0.5 et 0.7.

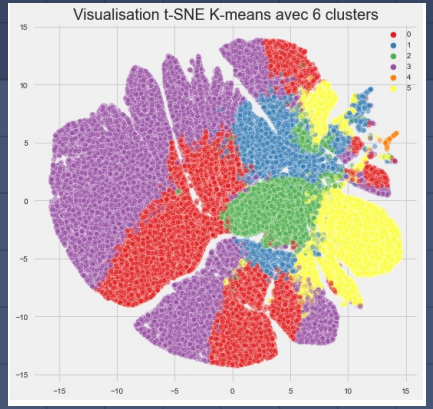
Application de l'algorithme du K-Means

K-means avec k = 6

Groupe	total_freight_cost	payment_sequential_nbre	payment_installments_sum	score	traitement	RFM_score	recency	frequency	monetary_value
0	19.863947	1.025272	2.804719	4.654801	9.321732	3.456931	125.186623	1.122142	127.759660
1	78.048836	1.033975	5.644925	3.873239	13.081874	4.075254	222.942618	2.619645	612.534864
2	16.626502	1.032147	.552763	4.582886	10.762490	2.260906	366.976970	1.046060	87.943920
3	33.778078	1.037513	.453472	4.125798	20.981891	2.963899	250.802133	1.073683	142.291460
4	21.828050	1.031438	2.959590	1.547904	20.792706	3.064269	230.861387	1.185882	124.580226
5	24.816142	6.126312	6.424869	4.192913	12.946152	2.824803	274.925197	1.177165	127.768031

k=6

Groupe	
0	36212
1	5019
2	31133
3	9188
4	11276
5	254

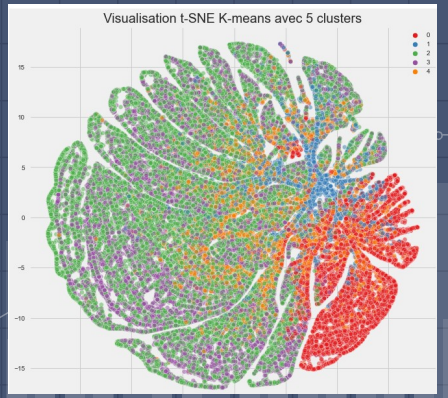


K-means avec k = 5

Groupe	total_freight_cost	payment_sequential_nbre	payment_installments_sum	score	traitement	RFM_score	recency	frequency	monetary_value
0	21.759697	1.040533	2.957060	1.548433	20.785350	3.061407	230.939577	1.184283	124.016818
1	76.880371	1.070311	11439	3.890207	13.044719	4.073935	222.815784	2.583986	602.685104
2	19.746957	1.030130	78009	4.655561	9.320704	3.449675	125.178603	1.119298	126.375058
3	16.638372	1.054978	2.584621	4.582475	10.767613	2.259249	368.012040	1.046066	88.111161
4	33.695677	1.050105	3.447955	4.122834	21.000908	2.962378	250.780400	1.073200	141.903717

k=5

Groupe	
0	11287
1	5233
2	36304
3	31064
4	9194



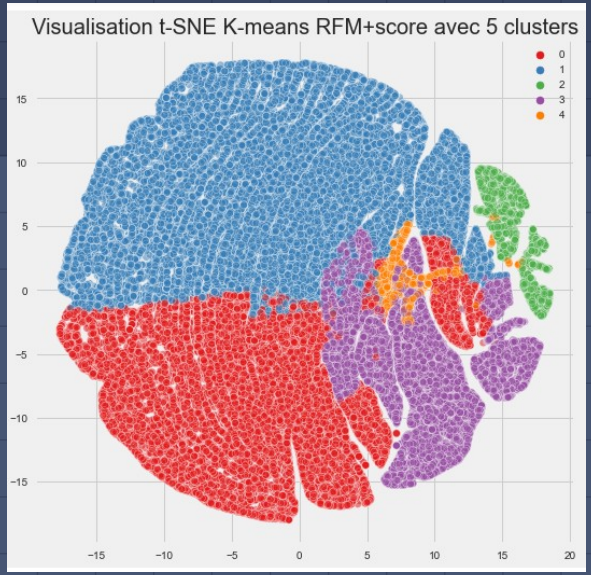
Comparaison des groupes K-Means

K-means avec dataset réduit (RFM+score) et k=5

	Groupe	recency	frequency	monetary_value	score
0	0	392.335445	1.085957	113.660441	4.643583
1	1	122.248347	1.083143	112.323805	4.707761
2	2	229.275373	3.864678	280.577235	3.613600
3	3	229.709236	1.158146	122.734143	1.758624
4	4	234.913066	1.194942	1199.895748	4.173165

V=4

Groupe	
0	31567
1	41591
2	2749
3	15277
4	1898



groupe 0 : "Low customers" client occasionnels.Très satisfaits mais n'ont plus achetés depuis 1 an? Prévoir une relance avec offre promotionnelle -10% à durée limitée. L'achat pour un montant minimum au cours du trimestre déclenche la ristourne

groupe 1 : " Medium Value customer" envoyer folder offre promotionnelle jusqu'à 20% sur série articles de saison

groupe 2 : "High Value Customers" Clients réguliers Prévoir offre promotionnelle avec frais livraison réduite ou gratuite sur toute commande d'un montant minimum (à définir) passée dans les 2 mois

groupe 3 : "Lost ou nearly-lost customers" Prévoir action pour comprendre leur insatisfaction et relancer en proposant produit équivalent avec réduction et/ou meilleurs délais de livraison

groupe 4 : "Top customers" clients réguliers avec panier important. Prévoir une relance avec offre promotionnelle sur articles/catégories les plus vendus à ces clients

Application de l'algorithme DBSCAN (Density-based spatial clustering of applications with noise

Contrairement au clustering k-means, DBSCAN ne nécessite pas de spécifier initialement le nombre de clusters.
Cependant, DBSCAN nécessite deux paramètres, à savoir. le rayon des voisinages pour un point de données donné p (ε) et le nombre minimum de points de données dans un ε-voisinage donné pour former des grappes Pour réduire le temps de calcul je travaille sur un sampling de 10000 échantillons..

On veut déterminer l' epsilon qui donne un nombre de clusters acceptable(k compris entre 4 et 8 pour une segmentation clients) , un nombre réduit de points "outliers" et un silhouette score moyen élevé.

```
range_eps=np.arange(1.5,3.0, 0.10)
for n in range_eps:
    print('eps value is '+str(round(n,1)))
    db=DBSCAN(eps=n,min_samples=3).fit(X_scaled)
    labels=db.labels_
    no_clusters = len(np.unique(labels) )
    no_noise = np.sum(np.array(labels) == -1, axis=0)
    core_samples_mask=np.zeros_like(db.labels_,dtype=bool)
    core_samples_mask[db.core_sample_indices_]=True
    print(set(labels))
    silhouette_avg=silhouette_score(X_scaled,labels)
    print("For eps value ="+str(round(n,1)),
          ' the average silhouette score is :',round(silhouette_avg,2))
    print('Estimated no. of clusters: %d' % no_clusters,'  ','Estimated no. of noise points: %d' % no_noise)
    print('  ')
```

```
eps value is 2.6
{0, 1, 2, 3, 4, -1}
For eps value =2.6  the average silhouette score is : 0.56
Estimated no. of clusters: 6    Estimated no. of noise points: 108

eps value is 2.7
{0, 1, 2, 3, 4, -1}
For eps value =2.7  the average silhouette score is : 0.56
Estimated no. of clusters: 6    Estimated no. of noise points: 99
```

Un eps de 2.7 et un min_samples = 3 nous donne un silhouette score moyen de 0.56 avec 6 clusters et 99 outliers.
Nous allons faire varier le nombre de min_samples

Application de l'algorithme DBSCAN (Density-based spatial clustering of applications with noise

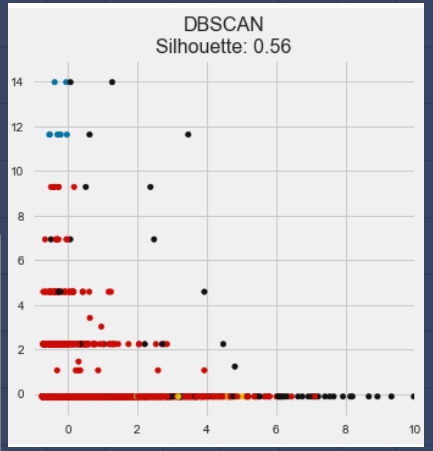

```
range_min_samples=np.arange(1,11,1)
for n in range_min_samples:
    print('min_samples value is '+str(round(n,1)))
    db=DBSCAN(eps=2.7,min_samples=n).fit(X_scaled)
    labels=db.labels_
    silhouette_avg=silhouette_score(X_scaled,labels)
    no_clusters = len(np.unique(labels) )
    no_noise = np.sum(np.array(labels) == -1, axis=0)
    core_samples_mask=np.zeros_like(db.labels_,dtype=bool)
    core_samples_mask[db.core_sample_indices_]=True
    #print(set(labels))

    print("For min_samples value ="+str(round(n,1)),"  "
          'estimated no. of clusters: %d' % no_clusters,'  ',
          'estimated no. of noise points: %d' % no_noise)
    print('The average silhouette score is :',round(silhouette_avg,2))
    print('  ')
```

```
min_samples value is 2
For min_samples value =2   estimated no. of clusters: 11    estimated no. of noise points: 89
The average silhouette score is : 0.55

min_samples value is 3
For min_samples value =3   estimated no. of clusters: 6    estimated no. of noise points: 99
The average silhouette score is : 0.56

min_samples value is 4
For min_samples value =4   estimated no. of clusters: 4    estimated no. of noise points: 108
The average silhouette score is : 0.59
```

Pour un eps = 2.7:
la valeur de min_samples = 3 donne un silhouette score = 0.56 avec 5 clusters et 99 outliers. Je conserve min_samples = 3 pour la suite

```
#  on lance le dbscan avec les valeurs eps et min_samples définies
db = DBSCAN(eps=2.7, min_samples=3)
predictions = db.fit_predict(X_scaled)
#labels_cls =db.predict(X_scaled)
dbscan_silhouette = silhouette_score(
    X_scaled, db.labels_
).round(2)
plt.figure(figsize=(12,5))
fte_colors = {0:'r',1: 'b',2:"g",3:"y",4:"orange",5:"c",6:'grey',-1:'k'  }
km_colors = [fte_colors[label] for label in db.labels_]
# afficher les points des différents clusters
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=km_colors,s=20,label=km_colors)
# ajuster échelle des axes
#plt.xlim(-1,2.5)
#plt.ylim(-1,15)
plt.title(f"DBSCAN\nSilhouette: {dbscan_silhouette}", fontdict={"fontsize": 16})

plt.show()
```

Groupe	
-1	99
0	9880
1	9
2	6
3	3
4	3

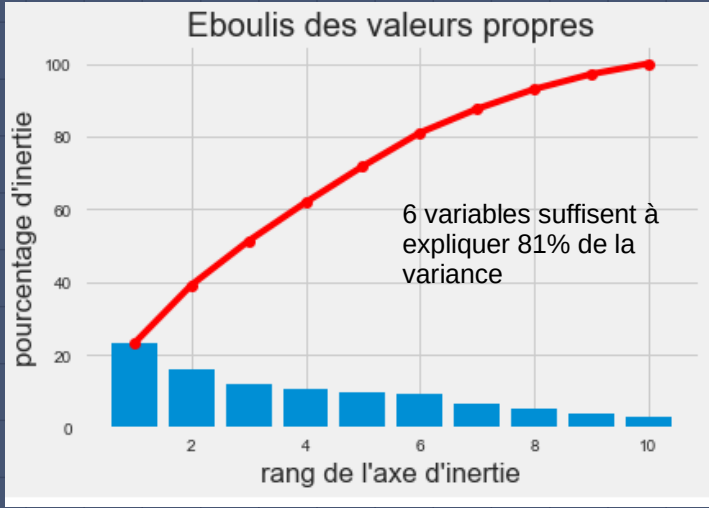


Un cluster est de taille très différente et compte 98,8 % des échantillons.La clustérisation avec DBSCAN donne des groupes très disproportionnés en terme de population qui s'imbriquent les uns dans les autres et ne sont pas exploitables dans un cadre professionnel de segmentation clients) .

Analyse en composantes principales ACP

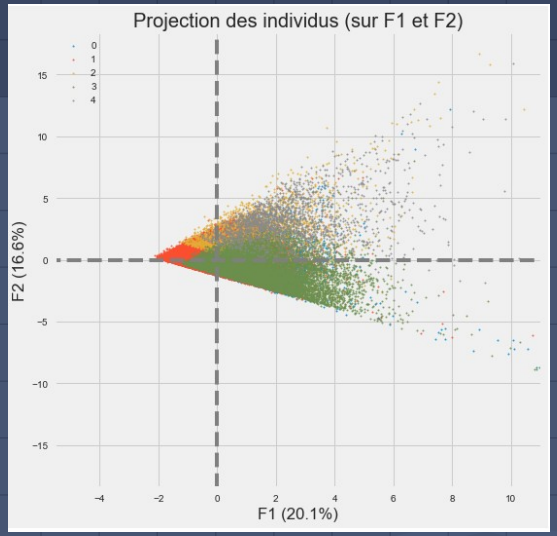

```
acp_in=data.copy()
# suppression des colonnes non numériques
X=acp_in.select_dtypes(np.number)
#Je vais standardiser() centrer et réduire) mes données
std_scale = preprocessing.StandardScaler().fit(X)
X_scaled = std_scale.transform(X)
pca = PCA()
pca.fit(X_scaled)
```

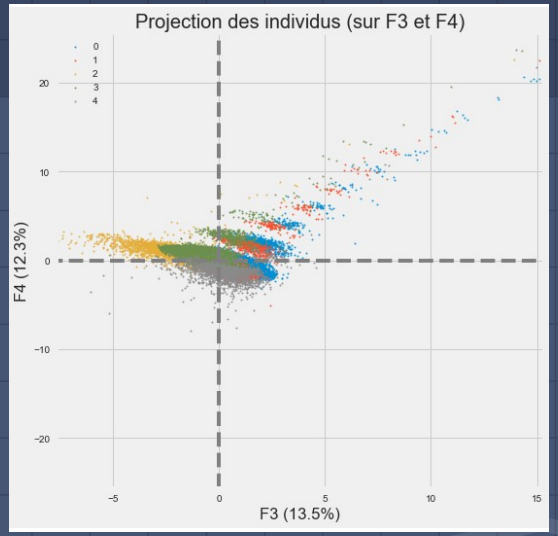
```
resume_var = pd.DataFrame(
    {
        "Dimension" : ["dim" + str(x + 1) for x in range(len(pca.explained_variance_))],
        "Variance expliquée" : pca.explained_variance_,
        "% variance expliquée" : np.round(pca.explained_variance_ratio_ * 100,1),
        "% cum. var. expliquée" : np.round(np.cumsum(pca.explained_variance_ratio_) * 100,1)
    }
)
resume_var
```

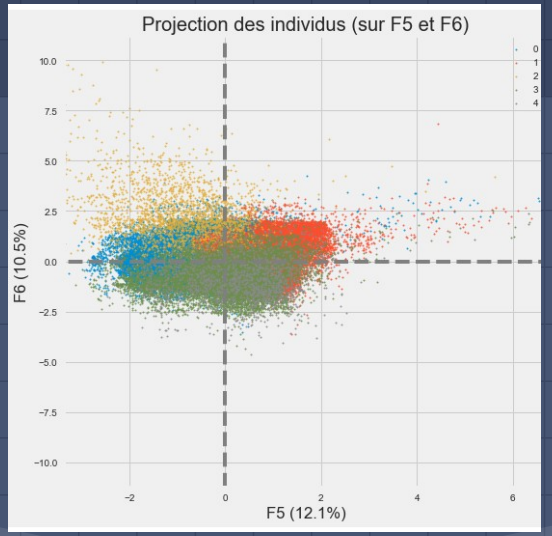


	Dimension	Variance expliquée	% variance expliquée	% cum. var. expliquée
0	dim1	2.321389	23.2	23.2
1	dim2	1.599629	16.0	39.2
2	dim3	1.209306	12.1	51.3
3	dim4	1.057975	10.6	61.9
4	dim5	0.992654	9.9	71.8
5	dim6	0.919358	9.2	81.0
6	dim7	0.654409	6.5	87.5
7	dim8	0.541722	5.4	93.0
8	dim9	0.413283	4.1	97.1
9	dim10	0.290382	2.9	100.0

Analyse en composantes principales ACP

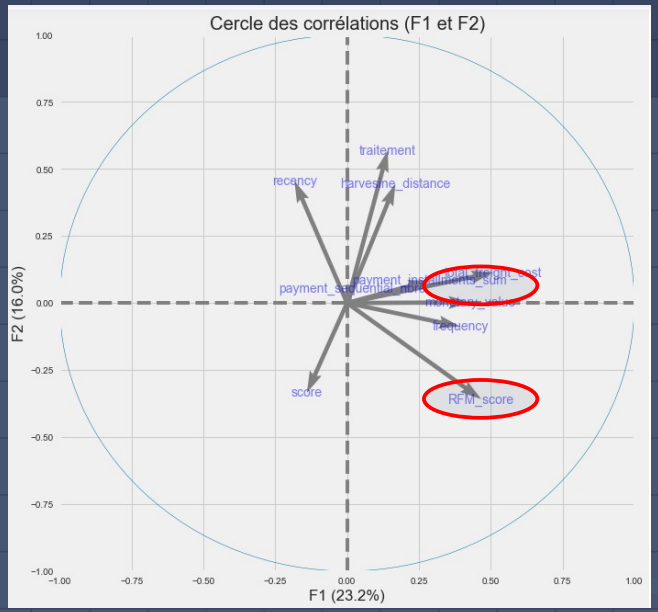


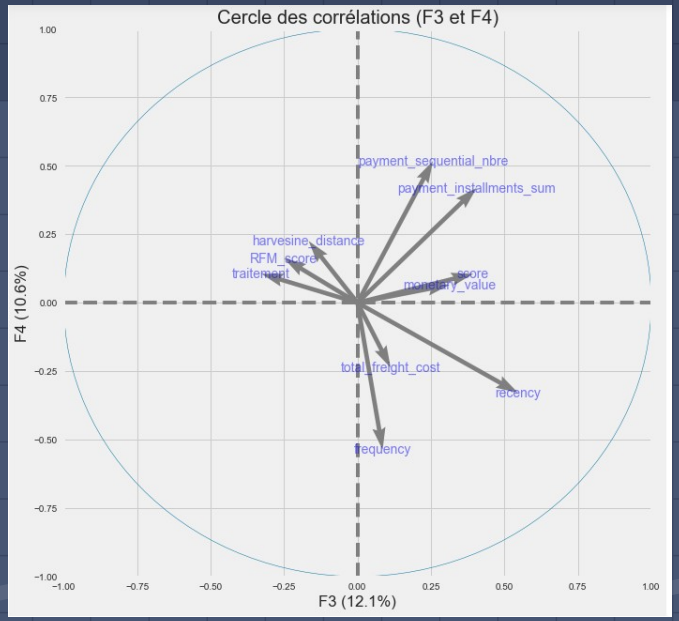




posante rincipale	total_freight_cost	payment_sequential_nbre	payment_installments_sum	score	traitement	RFM_score	recency	frequency	monetary_value	harves
PC1	0.507966	0.017265	0.287752	-0.139876	0.141777	0.467448	-0.183205	0.394086	0.428416	
PC2	0.113495	0.053120	0.083576	-0.335021	0.569166	-0.359031	0.455483	-0.087654	0.002285	
PC3	0.112739	0.256790	0.405579	0.394023	-0.327480	-0.250166	0.546283	0.085595	0.313674	
PC4	-0.237136	0.517518	0.419266	0.105515	0.106120	0.162324	-0.328703	-0.536834	0.065629	
PC5	-0.074173	0.633524	-0.039234	-0.535580	0.009327	0.000251	0.041928	0.275424	-0.146924	
PC6	0.258440	0.484430	-0.420749	0.419049	-0.050716	-0.035233	-0.062347	0.267555	-0.291425	
PC7	0.081267	0.162312	-0.597682	0.041335	0.152332	-0.030337	0.007372	-0.320153	0.667185	
PC8	-0.047931	-0.013480	0.093367	0.484467	0.711447	0.101978	0.001569	0.143239	-0.152093	
PC9	0.727274	-0.021220	0.112894	-0.042586	-0.033465	-0.356372	-0.270188	-0.354359	-0.233423	
PC10	-0.223661	-0.010566	0.112607	0.033604	0.065602	-0.651226	-0.523256	0.381400	0.290667	

Analyse en composantes principales ACP





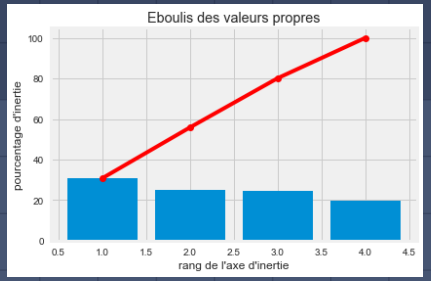
Composante Principale	total_freight_cost	payment_sequential_nbre	payment_installments_sum	score	traitement	RFM_score	recency	frequency	monetary_value
PC1	0.507966	0.017265	0.287752	-0.139876	0.141777	0.467448	-0.183205	0.394086	0.428416
PC2	0.113495	0.053120	0.083576	-0.335021	0.569166	-0.359031	0.455483	-0.087654	0.002285
PC3	0.112739	0.256790	0.405579	0.394023	-0.327480	-0.250166	0.546283	0.085595	0.313674
PC4	-0.237136	0.517518	0.419266	0.105515	0.106120	0.162324	-0.328703	-0.536834	0.065629
PC5	-0.074173	0.633524	-0.039234	-0.535580	0.009327	0.000251	0.041928	0.275424	-0.146924

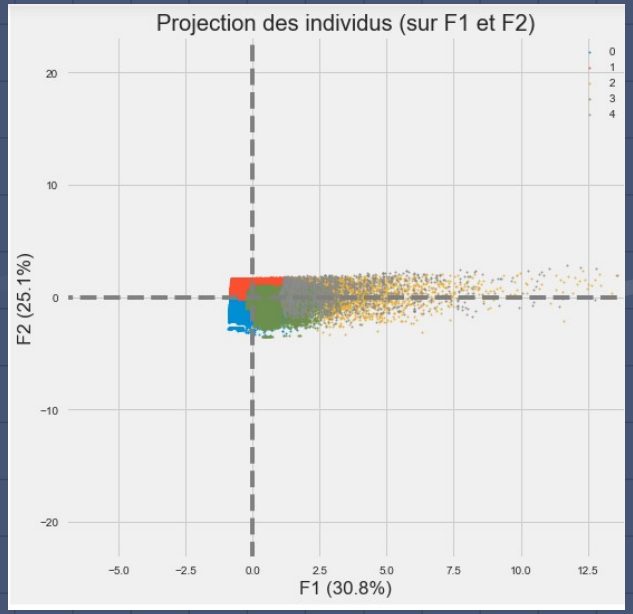
On constate que la composante principale 1 est fortement influencée par le coût du transport et le RFM_score.

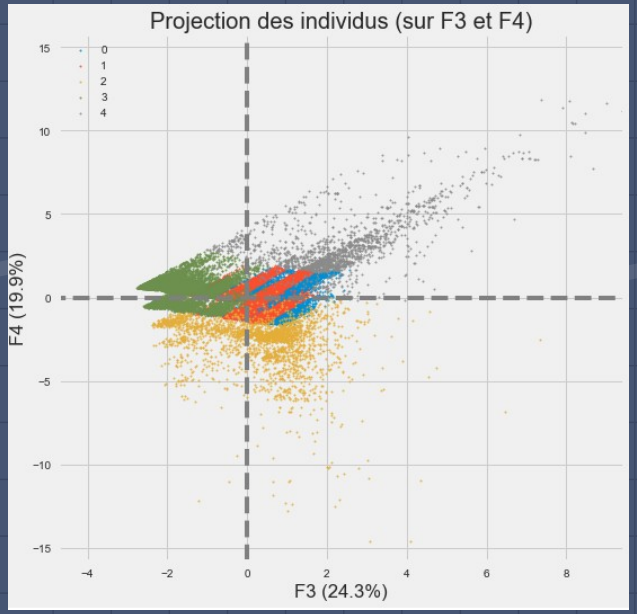
Analyse en composantes principales ACP avec variables RFM et score


```
recency
frequency
monetary_value
score
```

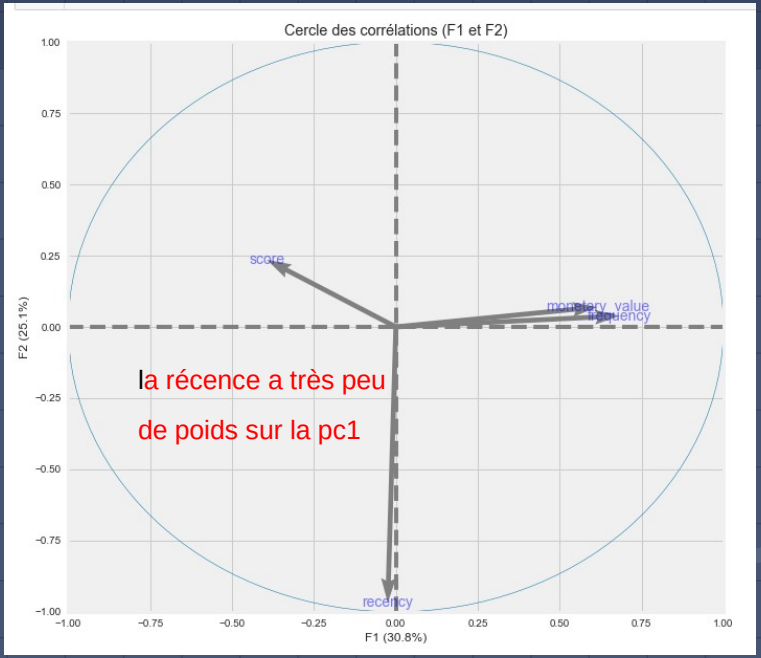
	Dimension	Variance expliquée	% variance expliquée	% cum. var. expliquée
0	PC1	1.231665	30.8	30.8
1	PC2	1.002243	25.1	55.8
2	PC3	0.970743	24.3	80.1
3	PC4	0.795393	19.9	100.0

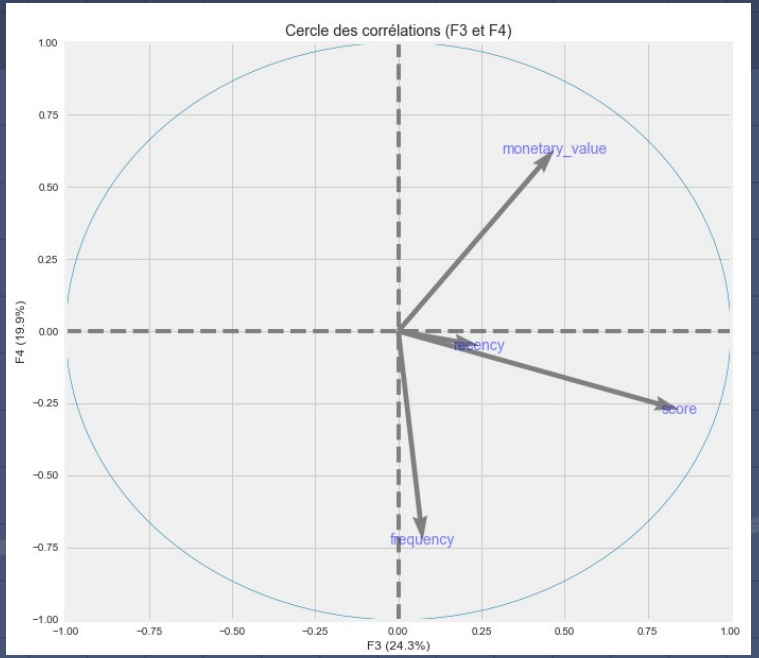






Analyse en composantes principales ACP avec variables RFM et score



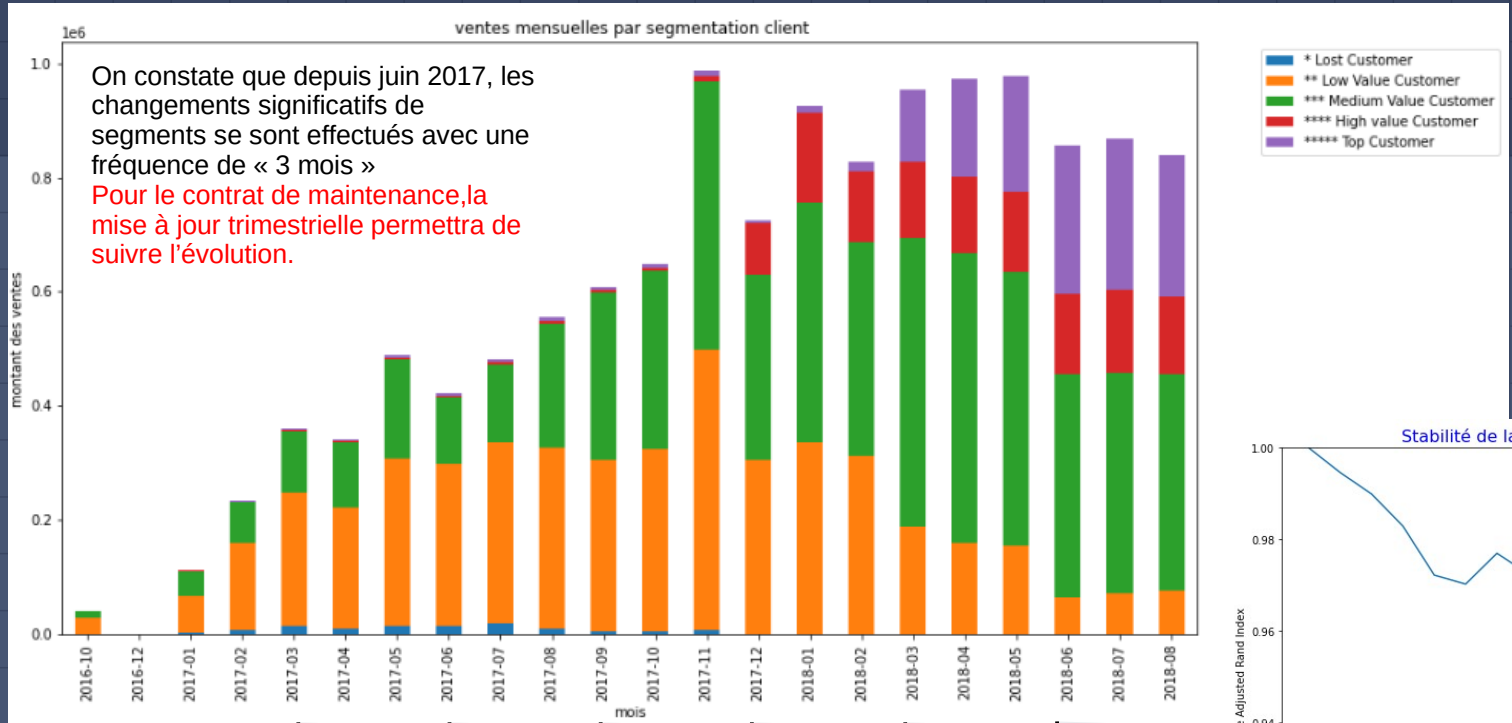


Composante Principale	recency	frequency	monetary_value	score
PC1	-0.024583	0.682739	0.614777	-0.394098
PC2	-0.968510	0.038530	0.069701	0.235895
PC3	0.242679	0.073625	0.469934	0.845487
PC4	-0.049914	-0.725922	0.629566	-0.272382

Période de mise à jour pour la segmentation client (contrat de maintenance)

On constate que depuis juin 2017, les changements significatifs de segments se sont effectués avec une fréquence de « 3 mois »

Pour le contrat de maintenance,la mise à jour trimestrielle permettra de suivre l'évolution.



↑ Low
↑ Med.
↑ Lost
↑ Top

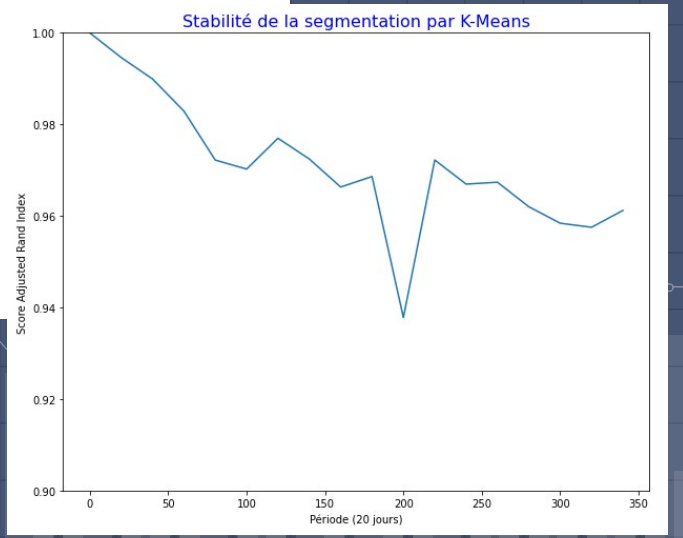
↓ Lost
↑ Med.

↑ Low
↑ Med.
↓ Lost
↑ Top

↓ Low
↓ Med.
↑ High

↓ Low
↓ Med.
↑ Top

↓ Low
↑ Top



Actions Marketing ciblées client

- fidéliser les nouveaux clients en proposant des bons de réduction valables 3 mois après le 1er achat.

- proposer une réduction sur les produits/catégories les plus achetés par les clients fidèles.

- réduire ou offrir les frais de transport dès la 3eme commande passée endéans les 6 mois ou dès que le panier total a dépassé un seuil fixé endéans les 6 mois pour l'ensemble de la clientèle.

- proposer de meilleurs services de livraison pour réduire le temps de traitement de la commande qui a un impact très important sur la note de satisfaction client.

- proposer des produits de substitution aux clients mécontents

- lancer enquête de satisfaction afin de collecter des données pertinentes

Actions Marketing en interne

- actualisation trimestrielle de la segmentation clients (algorithme k-means avec variables RFM + score)

- recherche d'autres transporteurs, création de centres de distribution locaux,.. pour réduire le temps de traitement

- étude ciblée « produits » pour en mesure de réagir en proposant des produits équivalents aux clients non satisfaits.

- Intégrer dans l'analyse (ou collecter plus de données) telles que l'âge, le sexe , la composition de famille , les hobbies ou centres d'intérêts , le type de résidence principale,la présence d'animal de compagnie ...des clients afin de cibler davantage leurs besoins .