


Segmentez des clients d'un site e-commerce

Morgan May

Projet n°4 du parcours Ingénieur Machine Learning



Sommaire

1. Sujet & interprétation
 2. Préparation et exploration des données
 3. Premier modèle : Segmentation RFM
 4. Modélisation “RFM” alternative
 5. Segmentation par K-Means
 6. Segmentation par DBSCAN
 7. Fréquence de maintenance du modèle et conclusion
- 



1. Sujet & interprétation



1. Sujet & interprétation

Le sujet de ce projet est la réalisation d'une segmentation des clients d'un site d'e-commerce brésilien dénommé Olist.

Pour réaliser cette segmentation, il nous est 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 entre 2017 et 2018.

Le problème ainsi décrit est un cas classique d'**application de techniques d'apprentissage non-supervisé**.

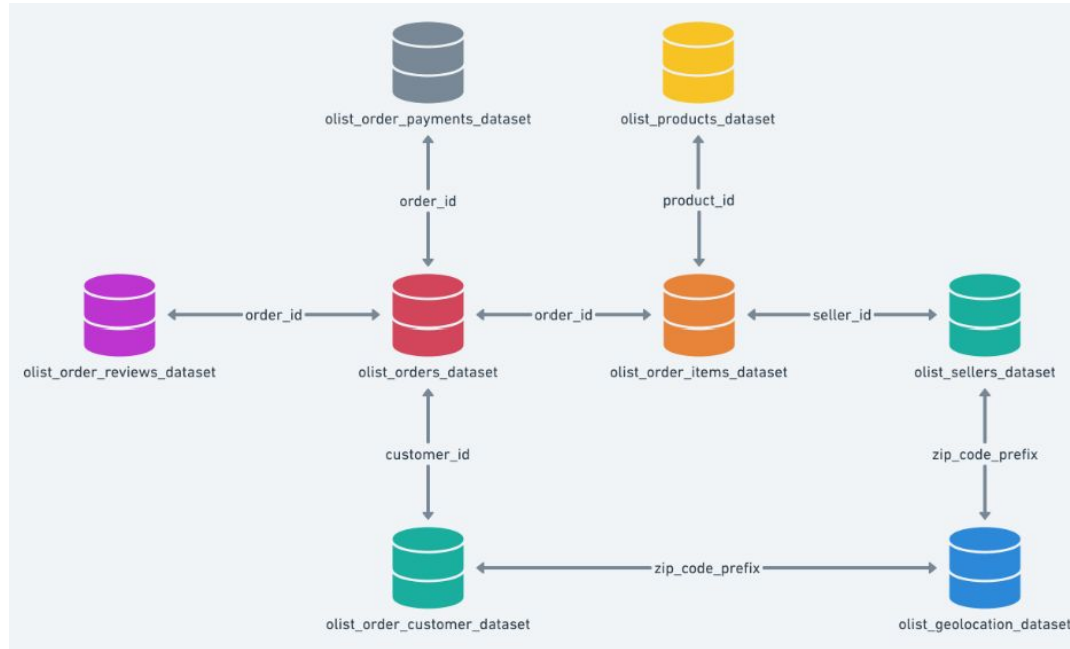


2. Préparation et exploration des données



2. Préparation et exploration des données

La base de données dont nous disposons est composée de plusieurs jeux de données répartis de la manière suivante :



2. Préparation et exploration des données

Afin de réaliser notre segmentation, nous avons besoin de réaliser une modélisation basée sur un seul jeu de données. La première étape de préparation des données consiste donc ici à regrouper ces jeux de données.

Pour cela, la première étape est de joindre l'ensemble de ces datasets en utilisant la méthode `merge()` de pandas, en utilisant une variable clé communes entre chaque dataset :

```
# Merging customers_data and orders_data :  
  
orders = pd.merge(customers_data, orders_data, on='customer_id', how='left')  
  
print(orders.shape)  
  
orders.head(10)
```

2. Préparation et exploration des données

Suite à l'ensemble de ces opérations de regroupement, nous obtenons un dataset décrivant l'ensemble des commandes et composé de 119143 lignes pour 40 colonnes, ces dernières étant les variables initialement présentes dans chaque dataset :

```
'customer_id',  
'customer_unique_id',  
'customer_zip_code_prefix',  
'customer_city',  
'customer_state',  
'order_id',  
'order_status',  
'order_purchase_timestamp',  
'order_approved_at',  
'order_delivered_carrier_date',  
'order_delivered_customer_date',  
'order_estimated_delivery_date',  
'order_item_id',  
'product_id',  
'seller_id',  
'shipping_limit_date',  
'price',  
'freight_value',
```

```
'product_category_name',  
'product_name_lenght',  
'product_description_lenght',  
'product_photos_qty',  
'product_weight_g',  
'product_length_cm',  
'product_height_cm',  
'product_width_cm',  
'product_category_name_english',  
'seller_zip_code_prefix',  
'seller_city',  
'seller_state',  
'payment_sequential',  
'payment_type',  
'payment_installments',  
'payment_value',
```

```
'review_id',  
'review_score',  
'review_comment_title',  
'review_comment_message',  
'review_creation_date',  
'review_answer_timestamp',  
'customer_longitude',  
'customer_latitude',  
'seller_longitude',  
'seller_latitude',  
'day_of_week_purchase',  
'hour_purchase']
```


2. Préparation et exploration des données

Dans cette planche et les 5 suivantes, quelques étapes de l'exploration des données (ainsi que des étapes de préparation correspondante) sont présentées afin de mieux cerner le jeu de données et ses particularités.

Exploration des statuts des commandes répertoriées :

L'affichage des valeurs prises par la variable "order_status" nous permet de voir que bien que la plupart des commandes aient le statut "delivered", certaines n'ont pas encore été livrées.

Comme nous souhaitons potentiellement utiliser certaines variables comme celles relatives aux avis déposés, nous gardons ici uniquement les commandes livrées.

```
data['order_status'].value_counts()
```

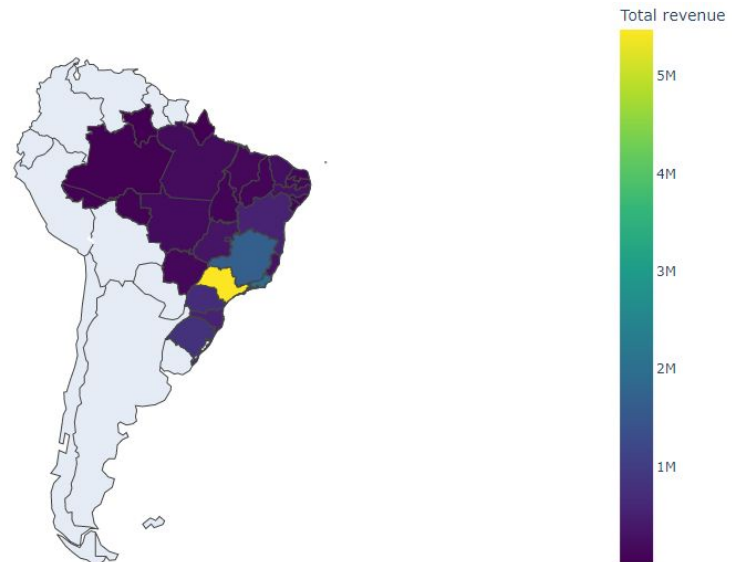
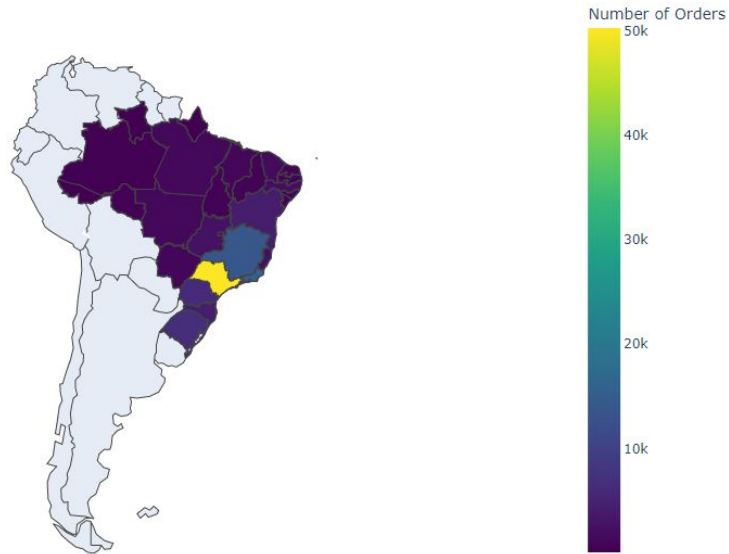
delivered	115723
shipped	1256
canceled	750
unavailable	652
invoiced	378
processing	376
created	5
approved	3

Name: order_status, dtype: int64

2. Préparation et exploration des données

Exploration de la répartition géographique des clients :

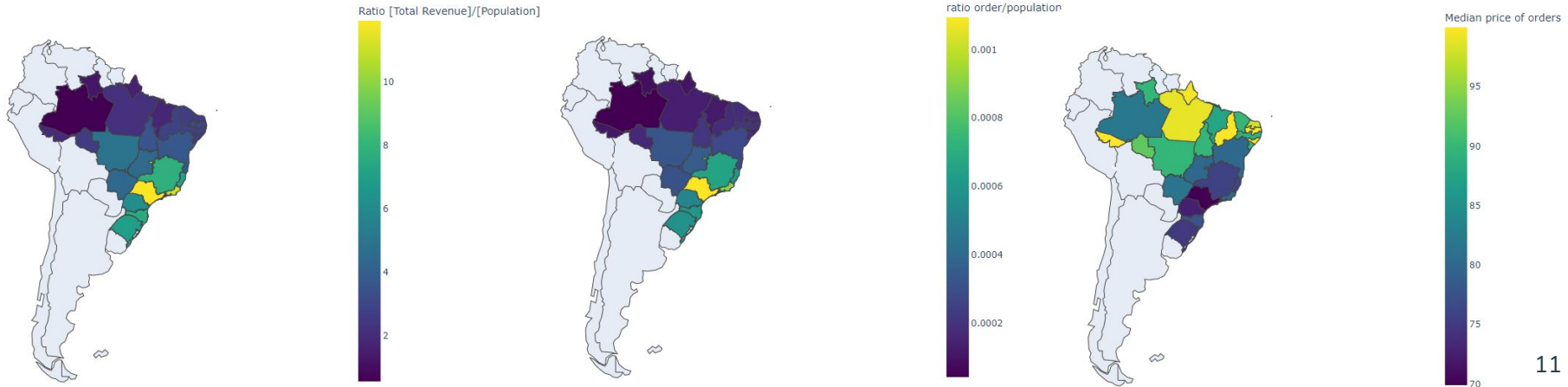
On peut voir sur le graphe ci-contre que le chiffre d'affaires généré et le nombre de commandes par Etat varient et semblent notamment se concentrer dans la région Sud-Est du Brésil.



2. Préparation et exploration des données

suite de l'exploration de la répartition géographique des clients :

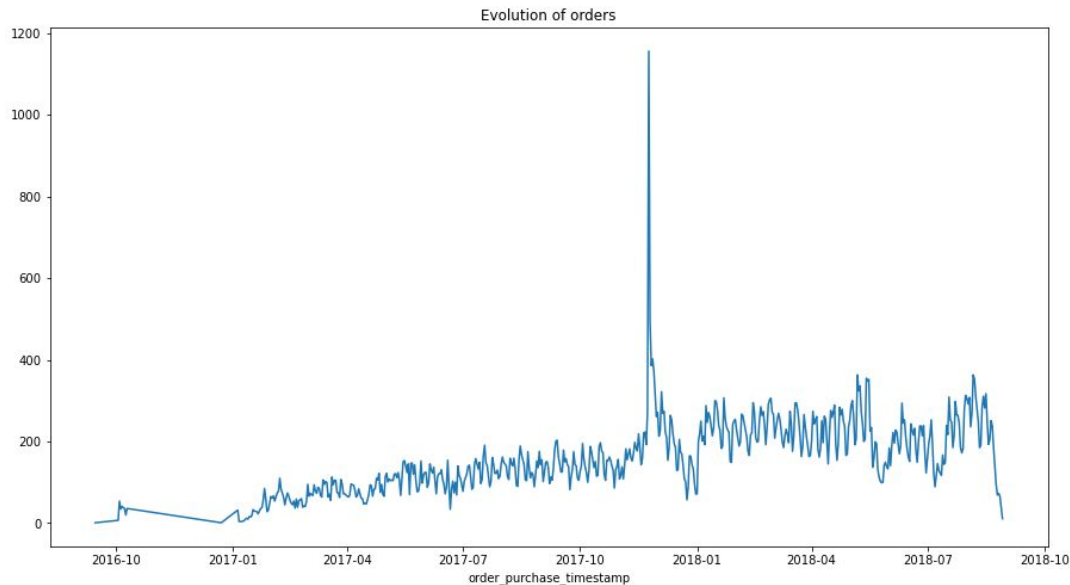
Afin d'éviter le biais de la densité de population, nous récoltons des données démographiques sur Wikipédia concernant le nombre d'habitants de chaque Etat : l'analyse des ratios [Chiffre d'affaires/Habitants], [Commandes]/[Habitants] et celle du prix médian **confirment une concentration en nombre de commandes et en chiffres d'affaires sur la région Sud-Est et met en avant des prix médians plus faibles que dans les autres états.**



2. Préparation et exploration des données

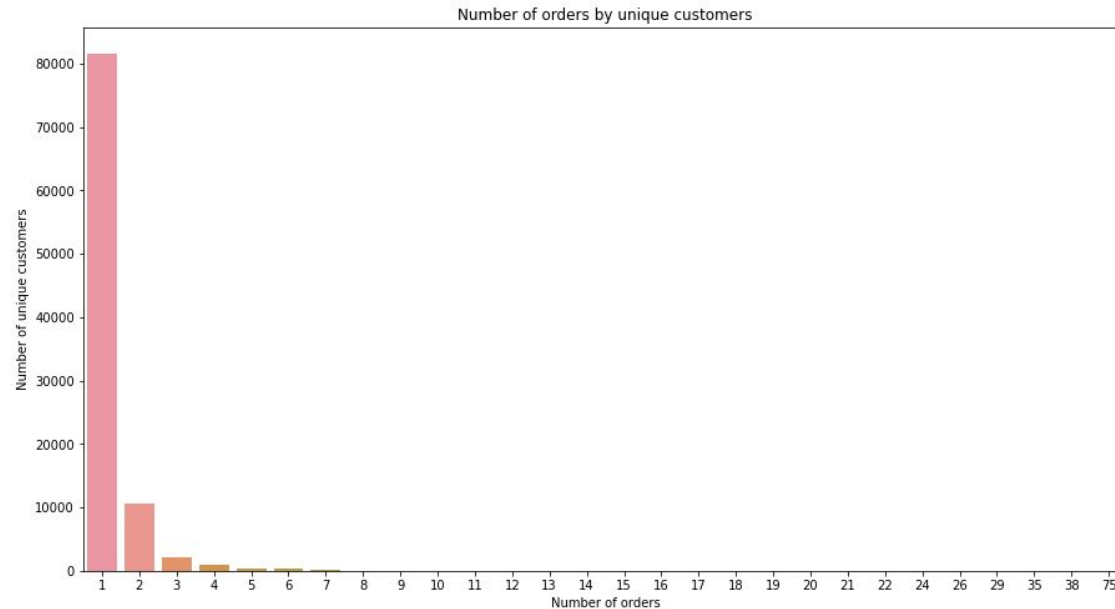
Exploration de la temporalité des commandes :

On peut voir sur le graphe ci-contre la répartition en nombre de commandes par jour avec un pic vers fin Novembre correspondant au Black Friday :



2. Préparation et exploration des données

Exploration de la fréquence d'achat des clients :



Le graphe ci-contre nous apprend
une information de contexte
importante pour notre segmentation :

**85% des clients de ce site web
n'effectuent qu'un seul
et unique achat.**

2. Préparation et exploration des données

Voici quelques conclusions de cette exploration :

- Les clients sont en majorité originaires des Etats du Sud-Est du Brésil, qui génèrent le plus de chiffre d'affaires et le plus grand nombre de commandes mais avec un prix médian plus faible que dans le reste du pays.
- 85% des clients n'achètent qu'une seule fois.
- Le moment le plus important pour Olist est le Black Friday.
- Les achats sont plus souvent réalisés en semaine que le week-end (non présenté ici).
- Le délai entre l'achat et la livraison est un facteur de satisfaction démontré dans le notebook (non présenté ici).

Dernière étape de préparation des données : recentrage du dataset autour des clients (en groupant via la variable 'customer_unique_id').



3. Premier modèle : Segmentation RFM

3. Premier modèle : segmentation RFM

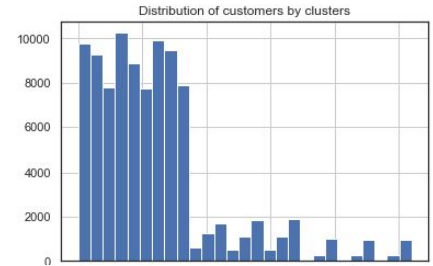
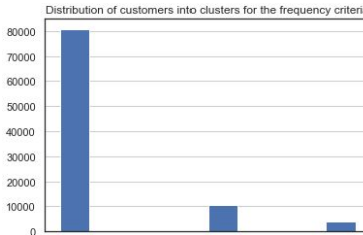
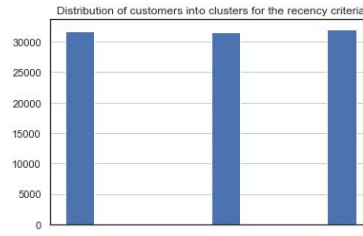
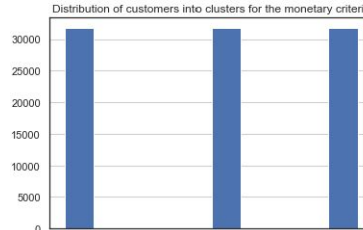
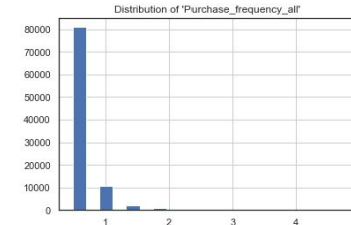
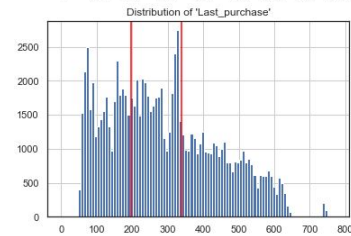
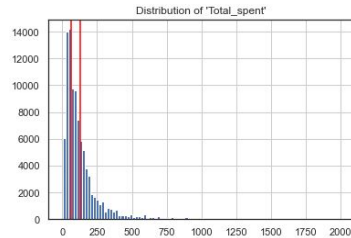
En marketing, la segmentation des clients est une tâche très courante. L'une des méthodes les plus populaires est la méthode RFM. Nous l'avons implémentée ici, avant de recourir à une méthode mettant en oeuvre des techniques de machine learning.

“RFM” signifie “Récence-Fréquence-Monétaire” en français : il s'agit globalement d'une méthodologie de segmentation des clients se basant sur ces 3 critères.

Nous construisons donc ici une segmentation en divisant en 3 la distribution des clients selon ces 3 critères, nous obtiendrons ainsi 27 clusters.

3. Premier modèle : segmentation RFM

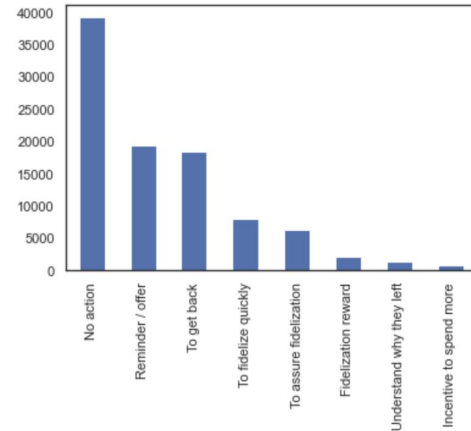
Distributions des clients selon les 3 critères :



3. Premier modèle : segmentation RFM

La segmentation obtenue dans les 27 clusters suivants peut être directement exploitée via des actions marketing pertinentes :

- Frequency = only one purchase, Recency = far past, Spendings = few amounts : no action
- Frequency = only one purchase, Recency = far past, Spendings = average amounts : no action
- Frequency = only one purchase, Recency = far past, Spendings = big amounts : to get back
- Frequency = only one purchase, Recency = average, Spendings = few amounts : no action
- Frequency = only one purchase, Recency = average, Spendings = average amounts : no action
- Frequency = only one purchase, Recency = average, Spendings = big amounts : to get back
- Frequency = only one purchase, Recency = recent, Spendings = few amounts : reminder / offer for big spendings
- Frequency = only one purchase, Recency = recent, Spendings = average amounts : reminder / offer for big spendings
- Frequency = only one purchase, Recency = recent, Spendings = big amounts : to fidelize quickly
- Frequency = >1 purchase but low frequency, Recency = far past, Spendings = few amounts : no action
- Frequency = >1 purchase but low frequency, Recency = far past, Spendings = average amounts : to get back
- Frequency = >1 purchase but low frequency, Recency = far past, Spendings = big amounts : to get back
- Frequency = >1 purchase but low frequency, Recency = average, Spendings = few amounts : no action
- Frequency = >1 purchase but low frequency, Recency = average, Spendings = average amounts : to assure fidelization
- Frequency = >1 purchase but low frequency, Recency = average, Spendings = big amounts : to assure fidelization
- Frequency = >1 purchase but low frequency, Recency = recent, Spendings = few amounts : incentive to spend more
- Frequency = >1 purchase but low frequency, Recency = recent, Spendings = average amounts : to assure fidelization
- Frequency = >1 purchase but low frequency, Recency = recent, Spendings = big amounts : to assure fidelization
- Frequency = high frequency, Recency = far past, Spendings = few amounts : reminder
- Frequency = high frequency, Recency = far past, Spendings = average amounts : to get back / understand why they left
- Frequency = high frequency, Recency = far past, Spendings = big amounts : special offer to get back / understand why they left
- Frequency = high frequency, Recency = average, Spendings = few amounts : incentive to spend more
- Frequency = high frequency, Recency = average, Spendings = average amounts : to assure fidelization
- Frequency = high frequency, Recency = average, Spendings = big amounts : special offer to get back / fidelization reward
- Frequency = high frequency, Recency = recent, Spendings = few amounts : incentive to spend more
- Frequency = high frequency, Recency = recent, Spendings = average amounts : fidelization reward
- Frequency = high frequency, Recency = recent, Spendings = big amounts : fidelization reward



3. Premier modèle : segmentation RFM

Ces clusters peuvent être regroupés, notamment comme suit :

- Best customers (have the highest values in the 3 features) : cluster 26
- Loyal customers (have the highest values for frequency) : clusters 18 to 26 (the 15% who come back, here)
- Big spenders (have the highest values for the monetary feature) : clusters 2, 5, 8, 11, 14, 17, 20, 23 and 26
- Almost lost customers (still recent but only purchased once) : clusters 6, 7 & 8
- Lost customers (have the lowest recency) : clusters 0, 1, 2, 9, 10, 11, 18, 19, 20



Best customers : 958
Loyal customers : 3933
Big spenders : 31797
Almost lost customers : 27263
Lost customers : 31734



4. Modélisation “RFM” alternative



4. Modélisation “RFM” alternative

La précédente modélisation, basée sur la méthode RFM, souffre notamment du manque de pertinence du critère de “Récence” dû au fait que 85% des clients ne reviennent pas réaliser d’achat.

Nous proposons donc une nouvelle modélisation, similaire mais en remplaçant simplement le critère de “Récence” par un critère de “Review” correspondant à la note laissée par le client suite à sa commande.

La variable créée (nommée “alt_review_cluster”) est une variable catégorielle prenant pour valeur :

- “Good review” si la moyenne des notes laissées est supérieure ou égale à 3 (*81382 clients*)
- “Bad review” si la moyenne des notes laissées est inférieure à 3 (*13998 clients*)
- “No review” si aucun avis n’est déposé (*716 clients*)

```
alt_RFM_data['alt_Review_cluster'].value_counts()
```

```
Good review    81382
Bad review     13998
No review       716
Name: alt_Review_cluster, dtype: int64
```

4. Modélisation “RFM” alternative

Cette modélisation nous permet ainsi de définir plusieurs clusters comme suit :

Interpretable clustering :

- Best customers (high spendings, high nb of orders and give good reviews)
- Loyal customers (high nb of orders)
- Bad customers (low spendings, only 1 purchase, give bad review)
- Likely satisfied customers (high spendings, high nb of orders but no reviews given)
- Good but unsatisfied customers (high spendings, high nb of orders but bad reviews)
- Fidelization potential customers (only one purchase, good review given)

Décompte par clusters :

Best customers :	2131
Loyal customers :	3941
Bad customers :	4059
Likely satisfied customers :	27
Good but unsatisfied customers :	778
Fidelization potential customers :	19943



5. Segmentation par K-Means



5. Segmentation par K-Means

a. K-means sur variables RFM

La 3ème approche proposée ici afin de réaliser un clustering des clients est l'exploitation d'un algorithme de machine learning, le K-Means.

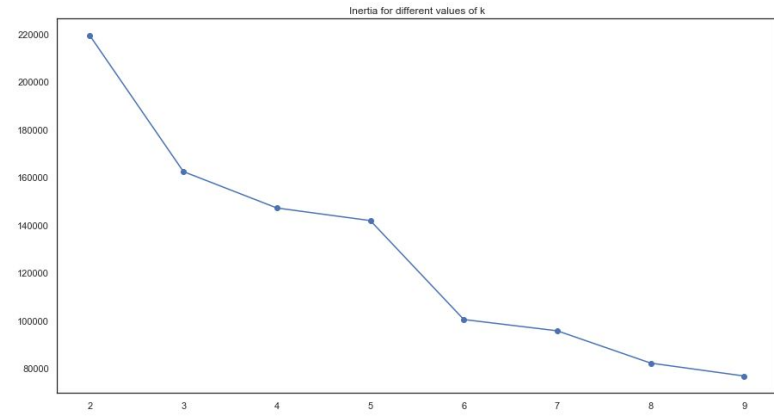
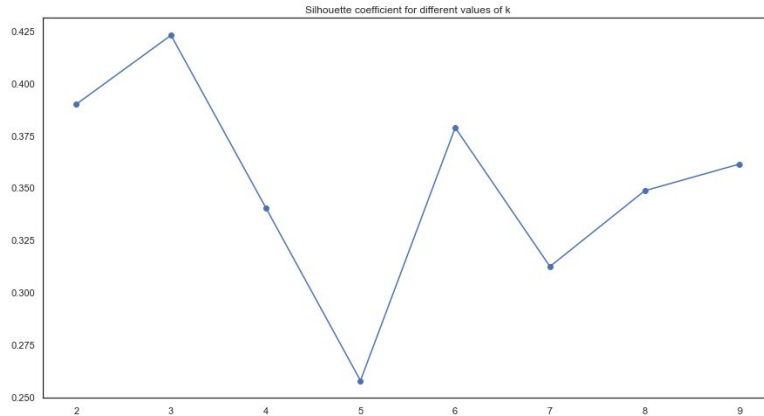
Le K-Means réalise un partitionnement en K clusters du jeu de données selon ses variables.

Nous choisissons dans un premier temps de le réaliser sur les mêmes variables que pour la méthodologie RFM :

- Total dépensé ('Total_spent')
- Fréquence d'achat ('Total_number_orders')
- Récence ('Last_purchase')

5. Segmentation par K-Means

Afin de choisir le nombre K de clusters, nous utilisons le coefficient de silhouette et l'inertie en fonction de k :

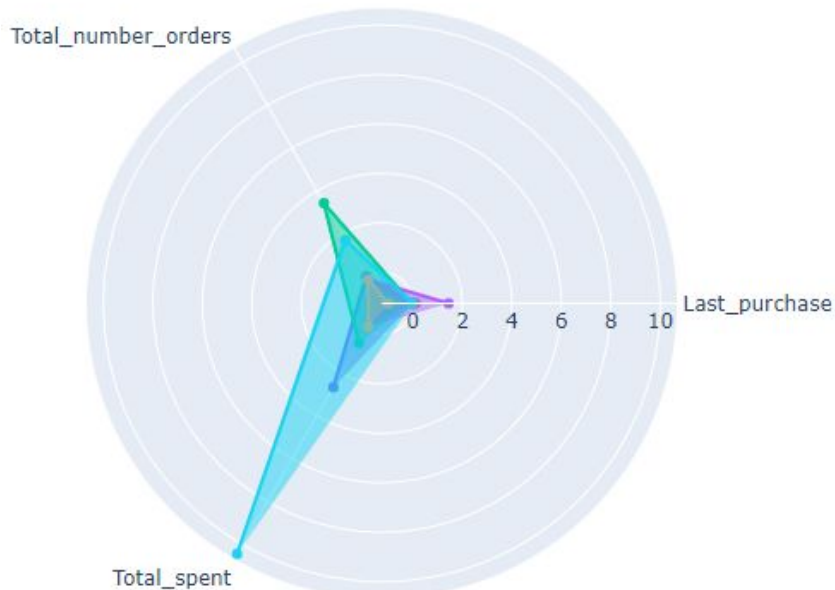


Ici, un nombre de clusters de 6 semble être un bon compromis entre coefficient de silhouette et inertie. Nous utilisons donc ce modèle pour notre segmentation.

5. Segmentation par K-Means

Comparison of clusters for Kmeans with k=6

Notre modèle K-means avec 6 clusters nous permet de réaliser la segmentation ci-contre dont l'interprétation peut être décrite telle que suit :



- Cluster 0 0 : Clients plus dépensiers que la moyenne
- Cluster 1 1 : Clients dans la moyenne sur les trois critères
- Cluster 2 2 : Clients loyaux
- Cluster 3 3 : Clients perdus
- Cluster 4 4 : Clients plutôt récents
- Cluster 5 5 : Clients particulièrement dépensiers et plus fidèles

5. Segmentation par K-Means

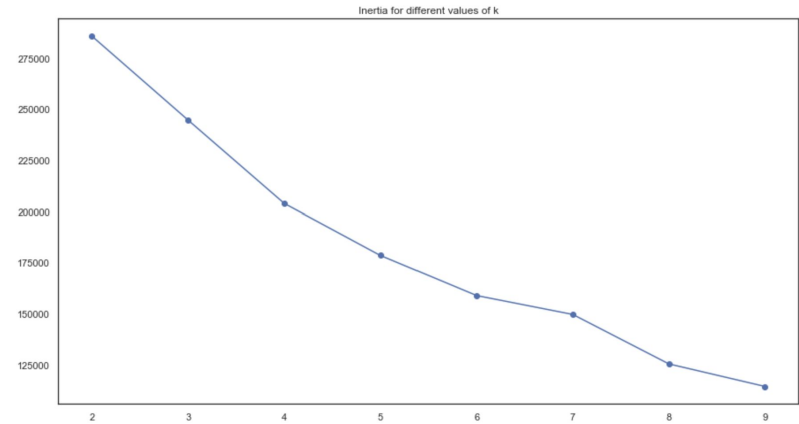
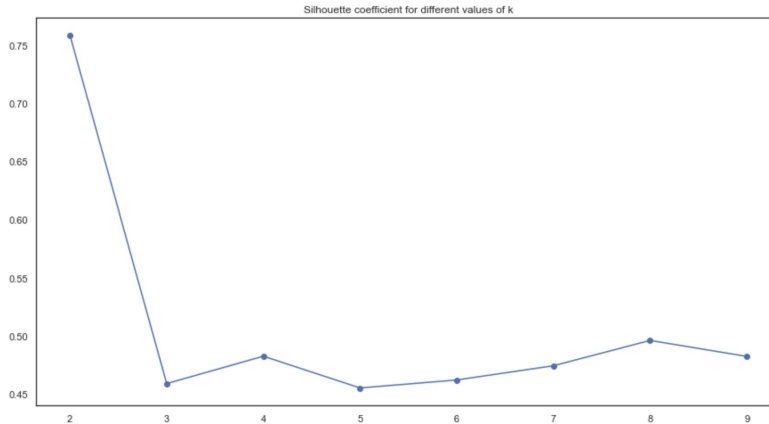
b. K-means sur variables étendues

Nous proposons dans un second temps une nouvelle implémentation du K-means en utilisant quelques autres variables potentiellement plus pertinentes car induire un intérêt marketing :

- Nombre total de commandes
- Panier moyen
- Frais de port moyen
- Total dépensé
- Variable encodée de la moyenne des notes d'avis

5. Segmentation par K-Means

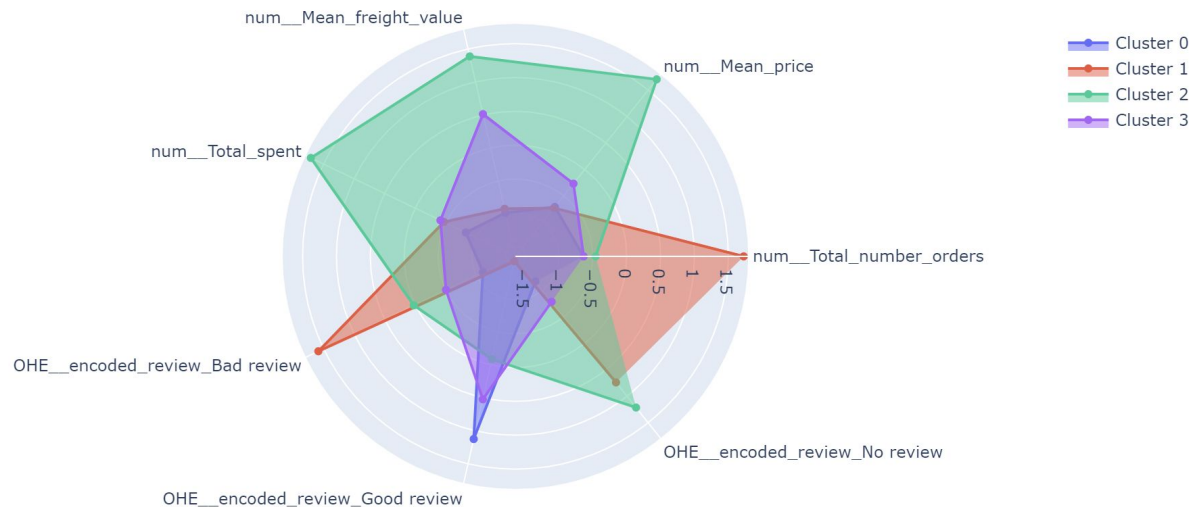
Afin de choisir le nombre K de clusters, nous utilisons le coefficient de silhouette et l'inertie en fonction de k :



Ici, nous choisissons $k=4$ qui semble être le nombre de clusters le plus faible permettant un bon compromis entre une inertie réduite et silhouette.

5. Segmentation par K-Means

Comparison of clusters for Kmeans with k=4



L'interprétation des clusters définis peut être la suivante :

- Cluster 0 : Clients ayant pour principale caractéristique de laisser de très bons avis
- Cluster 1 : Clients fidèles donnant pourtant de mauvaises notes
- Cluster 2 : Clients dépensant de fortes sommes (et ne donnant pas d'avis)
- Cluster 3 : Clients ayant des frais de port relativement élevés et donnant surtout de bons avis



6. Segmentation par DBSCAN

6. Segmentation par DBSCAN

Nous proposons ici la présentation de l'essai d'une segmentation par l'exploitation d'un autre algorithme de machine learning non-supervisé : DBSCAN.

DBSCAN est un algorithme de clustering par densité : contrairement à l'algorithme du k-means, des points "loin" l'un de l'autre dans l'espace multidimensionnel de représentation des données peuvent appartenir au même cluster si ils sont chacun atteignable par densité depuis l'autre.

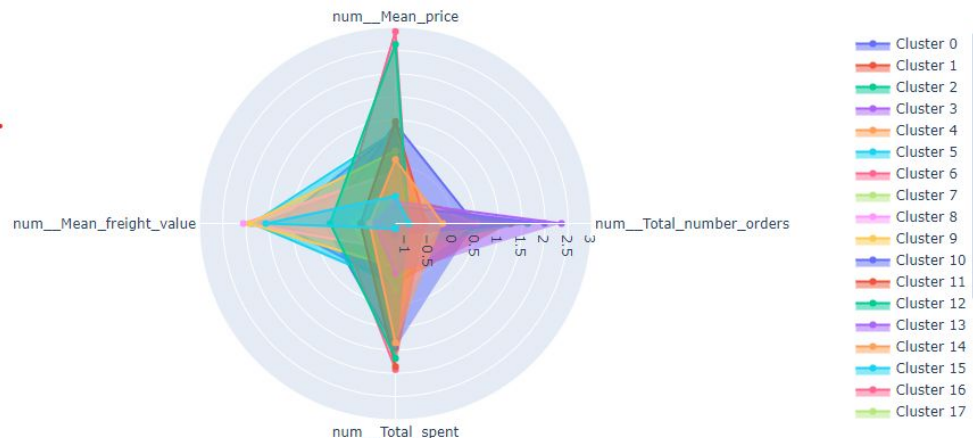
Nous avons ici mis en oeuvre cet algorithme sur l'ensemble de nos clients unique et en utilisant 4 variables déjà utilisées précédemment :

- le total dépensé ('Total_spent')
- le nombre total de commandes ('Total_number_orders')
- le panier moyen ('Mean_price')
- les frais de port moyen ('Mean_freight_value')

6. Segmentation par DBSCAN

Voici le clustering obtenu en utilisant les valeurs d'hyperparamètres par défaut, soit $\epsilon = 0.5$ et $\text{min_samples} = 5$:

Comparison of clusters for DBSCAN with default parameter values



Comme on peut le constater ici, une limite majeure de ce type d'algorithme pose problème : **le nombre de clusters (25 ici) est bien trop important pour obtenir une segmentation opérationnelle.**

6. Segmentation par DBSCAN

Nos tentatives d'optimisation du modèle DBSCAN par l'ajustement des hyperparamètres epsilon et min_samples ne nous ont pas permis d'améliorer franchement l'opérabilité de la segmentation obtenue, comme en atteste le tableau suivant de performances comparées de plusieurs modèles :

	Name	eps	min_samples	n_clusters	n_noise	silhouette
0	DBSCAN(eps=0.01, min_samples=4)	0.01	4	2195	19486	-0.333794
1	DBSCAN(eps=0.01)	0.01	5	1726	21900	-0.358682
2	DBSCAN(eps=0.01, min_samples=6)	0.01	6	1434	24004	-0.368813
3	DBSCAN(eps=0.1, min_samples=4)	0.10	4	197	2837	0.057707
4	DBSCAN(eps=0.1)	0.10	5	152	3237	0.061788
5	DBSCAN(eps=0.1, min_samples=6)	0.10	6	124	3672	0.050894
6	DBSCAN(min_samples=4)	0.50	4	43	501	0.326270
7	DBSCAN()	0.50	5	25	605	0.332622
8	DBSCAN(min_samples=6)	0.50	6	18	697	0.332962
9	DBSCAN(eps=0.7, min_samples=4)	0.70	4	33	345	0.336278
10	DBSCAN(eps=0.7)	0.70	5	30	406	0.335987
11	DBSCAN(eps=0.7, min_samples=6)	0.70	6	22	485	0.337193

Nous choisissons donc de conserver le second modèle k-means présenté en partie 5, qui est plus opérationnel et dont nous avons étudié la stabilité temporelle dans la partie suivante.



7. Fréquence de maintenance du modèle et conclusion

7. Fréquence de maintenance du modèle et conclusion

L'objectif de ce projet est de fournir une segmentation des clients pouvant être utilisée au quotidien par les équipes d'Olist pour leurs campagnes de communication.

En ce sens, nous avons proposé plusieurs segmentations, dont notamment un clustering réalisé par k-means. La question de la fréquence de maintenance de ce clustering revient à se poser la question de la stabilité temporelle du clustering; autrement dit : à partir de combien de temps l'information devient obsolète ?

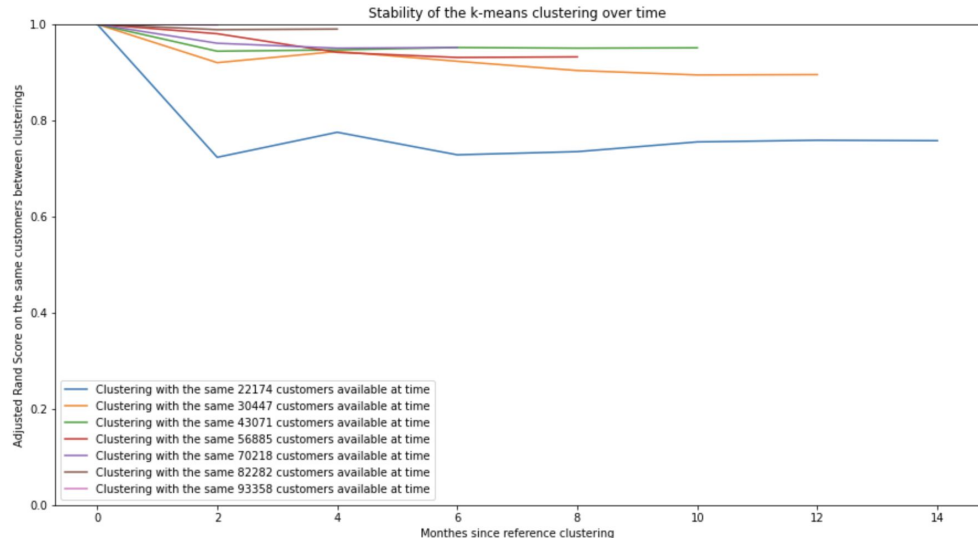
7. Fréquence de maintenance du modèle et conclusion

Pour répondre à cette question, nous avons réalisé une analyse de l'évolution temporelle des clusters.

Afin de comparer deux clusterings, nous utilisons le score de Rand ajusté (ARI) en se contentant de réaliser cette comparaison sur les mêmes clients. Nous comparerons ainsi plusieurs évolutions en fonction du moment choisi pour réaliser le clustering initial (servant de référence pour la comparaison via ARI).

Nous voyons ci-contre que les clusterings sont logiquement de plus en plus stables du point de vue de l'ARI (du fait du ratio décroissant de nouveaux clients par rapport au nombre total de clients).

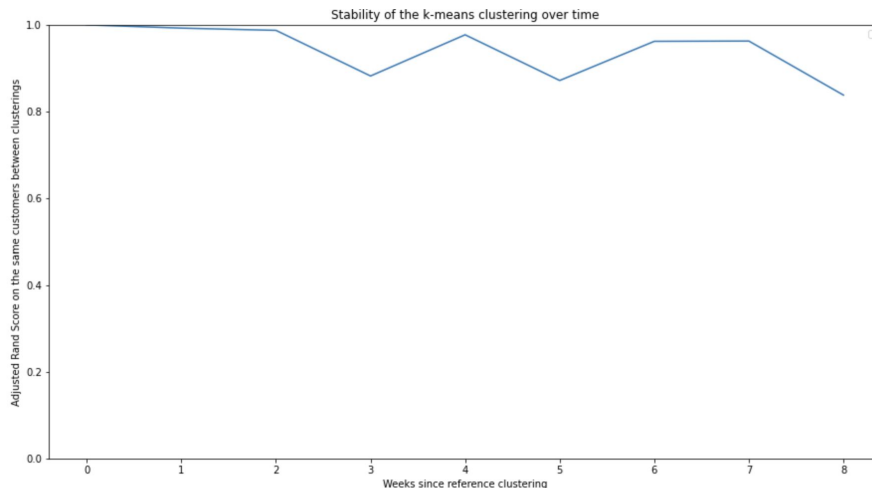
Cependant, on distingue une variation relativement importante de l'ARI sur les 2 premiers mois, particulièrement visible sur la première courbe, en bleu.



7. Fréquence de maintenance du modèle et conclusion

Nous étudions ici plus en détail l'évolution de l'ARI des clusterings réalisés semaine après semaine pendant les 8 premières semaines après le clustering de référence (réalisé quand la base de données contenait plus de 20000 clients).

On remarque que le moment correspondant environ à un ARI stable sur le graphique précédent (environ 0.8) intervient après environ 8 semaines.



Cependant, les quelques variations de l'ARI sur ces premières semaines et le relativement faible besoin de ressources de calcul nécessaire pour mettre à jour le clustering nous suggère de réaliser cette segmentation de façon plus fréquente et logique par rapport à une activité d'e-commerce. Ainsi, la mettre à jour quotidiennement nous semble plutôt adéquat.



Merci pour votre attention

