Réaliser une segmentation de clientèle

Ilaria Mereu

Soutenance Projet 5
Parcours Data Scientist

olist
empowering commerce

Mission

Réaliser une segmentation de la clientèle utile aux campagnes de communication d'entreprise.

Nos objectifs:

- Par le biais de la segmentation, différencier en catégories les clients.

- Fournir une description facile d'utilisation de notre segmentation et de sa logique pour l'équipe Marketing qui l'exploitera.

- Recommander une fréquence à laquelle la segmentation doit être mise à jour pour rester pertinente, afin de pouvoir effectuer un devis de contrat de maintenance.

olist

Entreprise brésilienne qui propose une solution de vente sur les marketplaces en ligne.

Sommaire

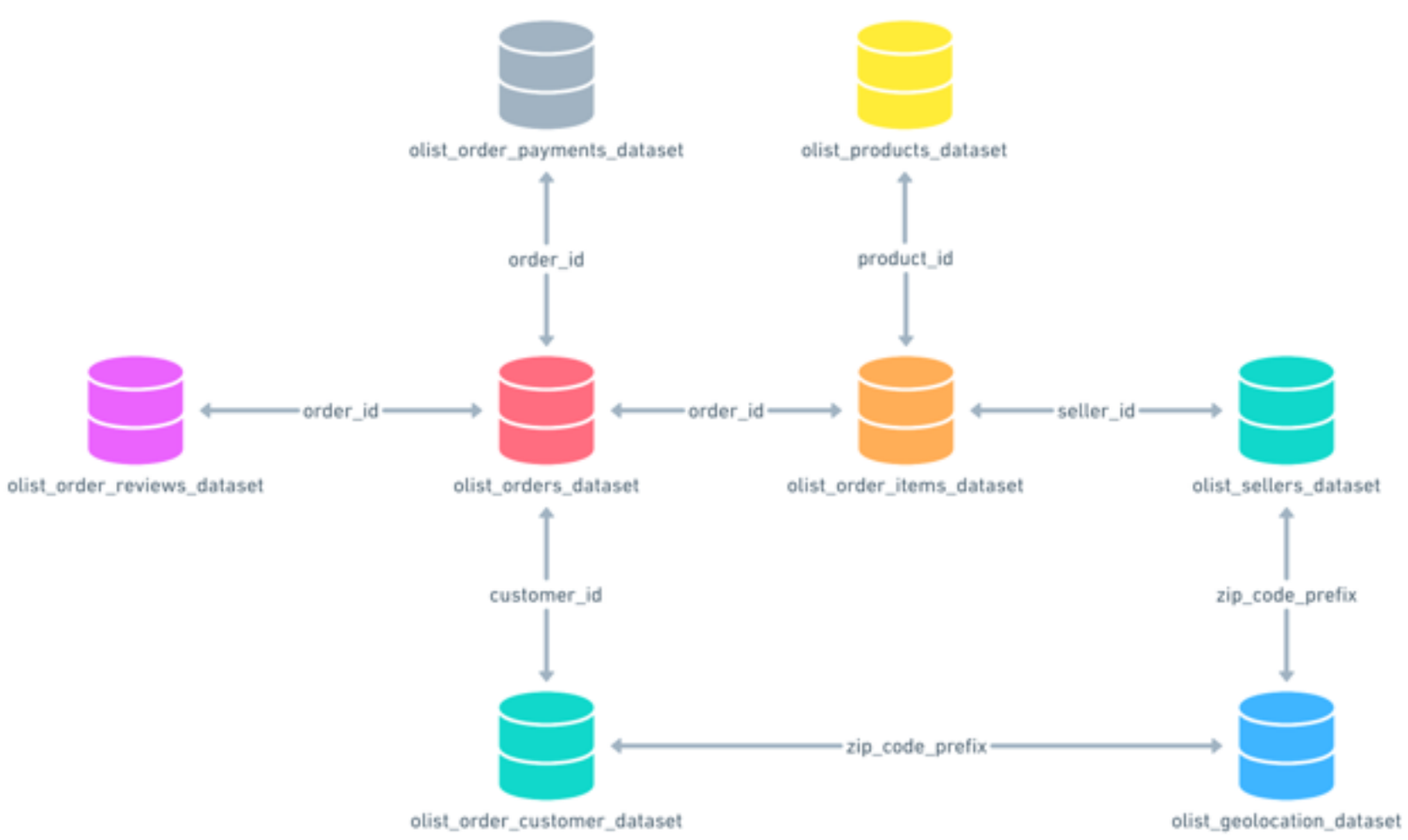
1 - Description de la base des données d'origine

olist

empowering commerce

Description de la base de données

Nom du fichier	**Taille**	
olist_customers_dataset.csv	8.6M	**Caractéristiques des clients**
olist_geolocation_dataset.csv	58M	Aide au repérage géographique des commandes géographiques
olist_order_items_dataset.csv	15M	Détails des produits par commande
olist_order_payments_dataset.csv	5.5M	**Détails des paiements**
olist_order_reviews_dataset.csv	14M	**Détails des notes des commandes**
olist_orders_dataset.csv	17M	**Détails des commandes**
olist_products_dataset.csv	2.3M	Détails des produits commandés
olist_sellers_dataset.csv	172K	**Détails sur les vendeurs**
product_category_name_translation.csv	4.0K	Traduction en anglais des catégories des produits



Commandes livrées

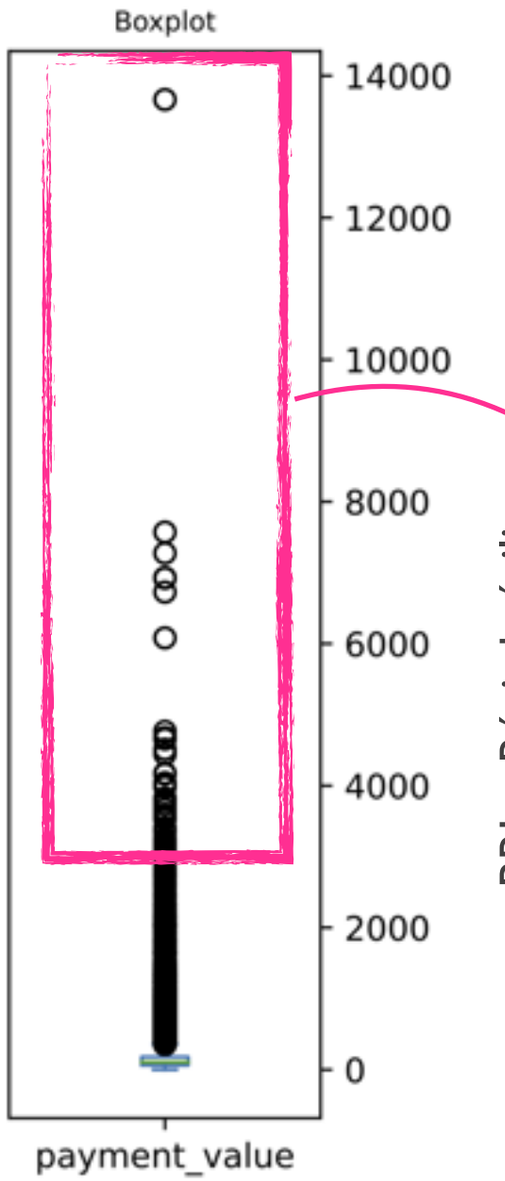
Nombre de commandes: ~99k
Nombre de commandes livrées: ~97k

CA des commandes livrées ~15M BLR

Nombre de clients uniques: 96k
94k pour les commandes livrées
Nombre de clients qui ont acheté à nouveau: ~3k
(3.11%, soit le 5.6% du CA)

Interval temporel des commandes:
04 septembre 2016 - 03 septembre 2018
soit deux ans exactes

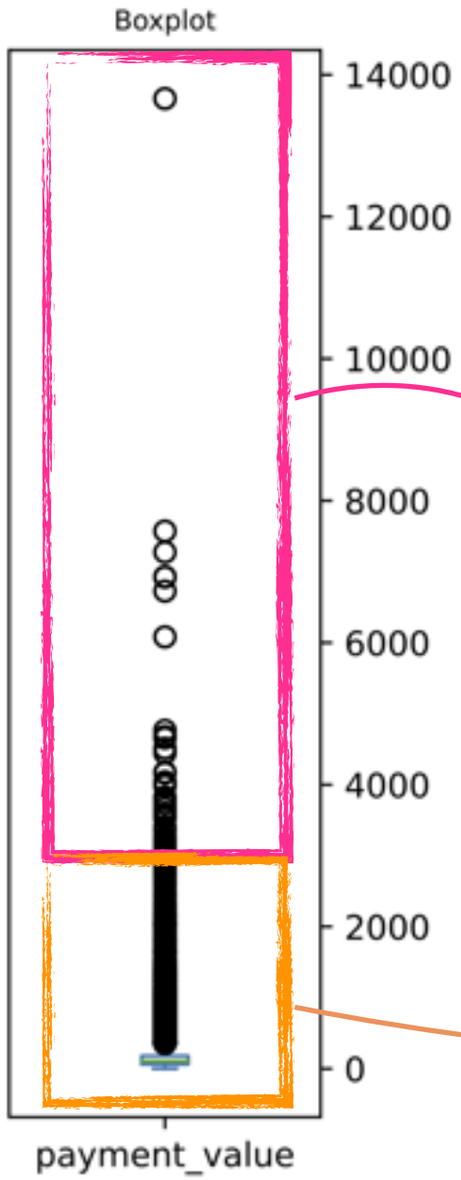
Distribution des paiements



segment separé

```
seuil: 10000 réais ~ 1950.0 euros     ->  outliers: 1 soit 13664 BRL ( 0.1 % du CA )
seuil: 8000 réais ~ 1560.0 euros      ->  outliers: 1 soit 13664 BRL ( 0.1 % du CA )
seuil: 6000 réais ~ 1170.0 euros      ->  outliers: 8 soit 62741 BRL ( 0.4 % du CA )
seuil: 4000 réais ~ 780.0 euros        ->  outliers: 20 soit 116168 BRL ( 0.8 % du CA )
seuil: 3000 réais ~ 585.0 euros       ->  outliers: 52 soit 221666 BRL ( 1.4 % du CA )
```

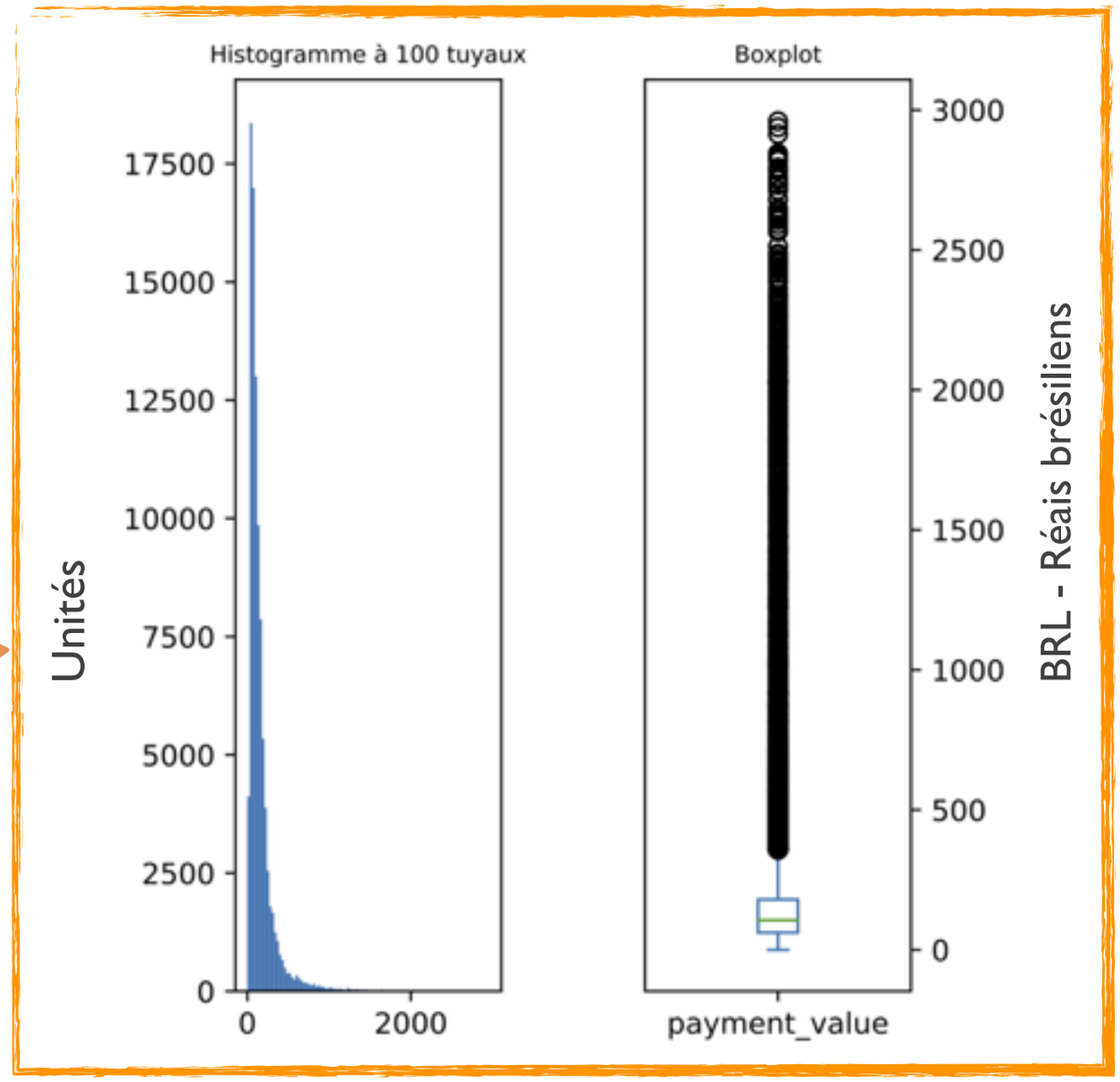
Distribution des paiements




```
seuil: 10000 réais ~ 1950.0 euros     ->  outliers: 1 soit 13664 BRL ( 0.1 % du CA )
seuil: 8000 réais ~ 1560.0 euros      ->  outliers: 1 soit 13664 BRL ( 0.1 % du CA )
seuil: 6000 réais ~ 1170.0 euros      ->  outliers: 8 soit 62741 BRL ( 0.4 % du CA )
seuil: 4000 réais ~ 780.0 euros        ->  outliers: 20 soit 116168 BRL ( 0.8 % du CA )
seuil: 3000 réais ~ 585.0 euros       ->  outliers: 52 soit 221666 BRL ( 1.4 % du CA )
```

segment separé

jeu à segmenter “< 3000 BRL”



2 - Modèles explorés et résultat de la recherche

olist

empowering commerce

Modèles explorés

Variables					
	orders	**payment value**	**recency crescent**	**review score**	**rapidite livraison**
	nombre de commande du client sur la période considérée	dépenses totales du client	mesure croissante de l'actualité du dernier achat	note du dernier achat	mesure croissante du décalage entre la date de livraison prévue et la date d'arrivé effective
RFM	☑	☑	☑		
4 features		☑	☑	☑	☑
5 features	☑	☑	☑	☑	☑

Modèle	Jeux de données	
1	clients uniques	RFM
2	"	4 features
3	"	5 features
4	clients uniques < 3000 BRL	RFM
5	"	4 features
6	"	5 features

Algorithmes
DBSCAN
agglomeratif
k-means ☑

1. Algorithme DBSCAN

Inutilisable sur les jeux entiers

ricerca-epsilon.ipynb ☆

Fichier Modifier Affichage Insérer Exécution Outils Aide Toutes les modifications ont été enregistrées

+ Code + Texte

```
        labels = db.labels_

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

        print("Epsilon= %f . Estimated number of clusters: %d . Estimated number of noise points: %d / %d ( %.2f )" % (i,

def DBS_funzione(j, label):
    strj = str(j)
    print(strj)
    strj_tex = strj.replace('_', ' ')

    x_matrix = commun.create_x_matrix(j)

    min_samples = 100

    N_lines = x_matrix.shape[0]
    # N_lines = 20000
    x_matrix = x_matrix[1:N_lines]

    x_matrix_scaled = StandardScaler().fit_transform(x_matrix)

    clusterer = DBSCAN(eps=eps_best[label], min_samples=min_samples) # Creates a K-means model to search for K different
    # cluster_labels = clusterer.fit_predict(x_matrix_scaled)  # Executes K-means on inputted data using an initialized K
    clusterer.fit(x_matrix_scaled)  # Executes K-means on inputted data using an initialized KMeans object. The returned

    cluster_labels = clusterer.labels_

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

                                                    er of noise points is %d / %d ( %.2f )" % (eps_best[label], n_

                                                    % (n_clusters_, min_samples))
```

Votre session a planté après avoir utilisé toute la mémoire RAM disponible. Si vous êtes intéressé par un accès à des environnements d'exécution à mémoire RAM élevée, nous vous conseillons Colab Pro.

Afficher les journaux d'exécution

1. Algorithme DBSCAN sur échantillon de 20k entrées

Model 1
['orders', 'payment_value', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.510000 . Estimated number of clusters: 2 . Estimated number of noise points: 810 / 20000 ( 0.04 )
Epsilon= 0.560000 . Estimated number of clusters: 2 . Estimated number of noise points: 668 / 20000 ( 0.03 )
Epsilon= 0.610000 . Estimated number of clusters: 2 . Estimated number of noise points: 520 / 20000 ( 0.03 )
Epsilon= 0.660000 . Estimated number of clusters: 2 . Estimated number of noise points: 424 / 20000 ( 0.02 )
Epsilon= 0.710000 . Estimated number of clusters: 2 . Estimated number of noise points: 371 / 20000 ( 0.02 )
Epsilon= 0.760000 . Estimated number of clusters: 2 . Estimated number of noise points: 335 / 20000 ( 0.02 )
Epsilon= 0.810000 . Estimated number of clusters: 2 . Estimated number of noise points: 299 / 20000 ( 0.01 )
Epsilon= 0.860000 . Estimated number of clusters: 2 . Estimated number of noise points: 282 / 20000 ( 0.01 )
Epsilon= 0.910000 . Estimated number of clusters: 2 . Estimated number of noise points: 265 / 20000 ( 0.01 )
```

ε optimal : [0.81, 0.71, 0.71, 0.86, 0.71, 0.71]
avec min_samples = 100

1. Limites de l'algorithme DBSCAN

1) Limité à une fraction des données
2) Le nombre des segments est déterminé par l'algorithme,
ce qui rend l'analyse moins flexible.
3) Dans le cas du jeu de trois variables, le nombre de segments trouvé est insatisfaisant.

2. Algorithme de segmentation agglomerative

Inutilisable sur les jeux entiers

ricerca-epsilon.ipynb ☆

Fichier Modifier Affichage Insérer Exécution Outils Aide Toutes les modifications ont été enregistrées

+ Code + Texte

```
    strj = str(j)
    print(strj)
    strj_tex = strj.replace('_', ' ')

    x_matrix = commun.create_x_matrix(j)

    min_samples = 100

    N_lines = x_matrix.shape[0]
    # N_lines = 20000
    x_matrix = x_matrix[1:N_lines]

    x_matrix_scaled = StandardScaler().fit_transform(x_matrix)

    clusterer = AgglomerativeClustering()
    # cluster_labels = clusterer.fit_predict(x_matrix_scaled)
    clusterer.fit(x_matrix_scaled)

    cluster_labels = clusterer.labels_

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

    print("# Epsilon= %f . Number of clusters is %s and number of noise points is

    if n_clusters_ > 10 or n_clusters_ < 3:
        print("Number of clusters is %s - min_samples = %s" % (n_clusters_, min_sa
        return

    if n_noise_ > int(N_lines*0.5):
                                                    = %s" % (n_noise_, mi
                                                    0/means-to-save-a-pytho
```

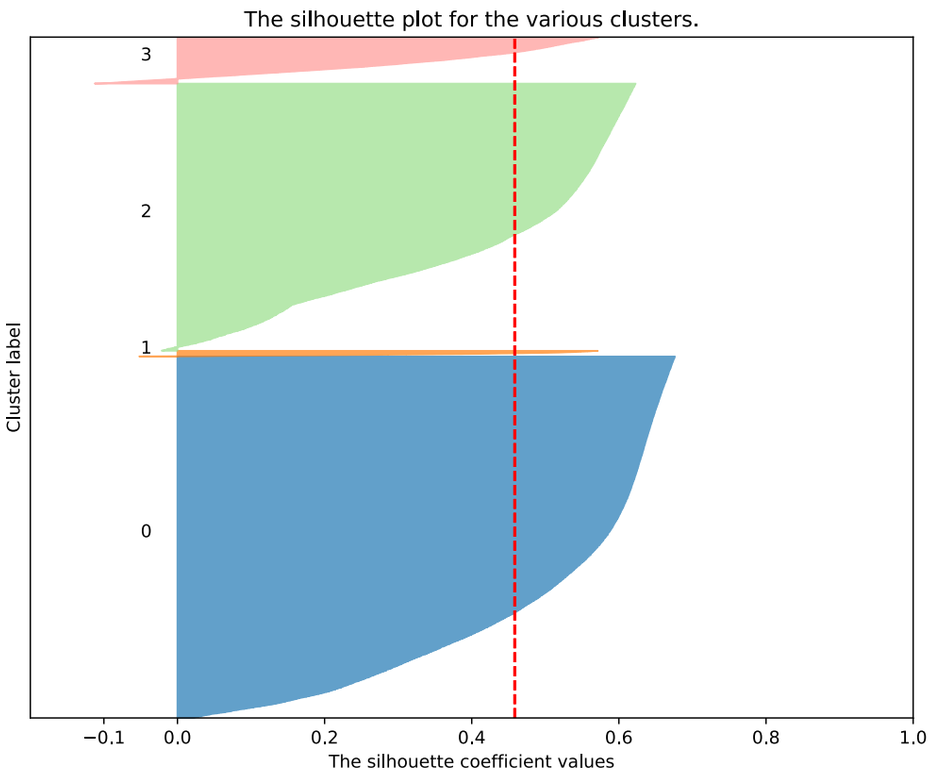
Votre session a planté après avoir utilisé toute la mémoire RAM disponible. Si vous êtes intéressé par un accès à des environnements d'exécution à mémoire RAM élevée, nous vous conseillons Colab Pro.

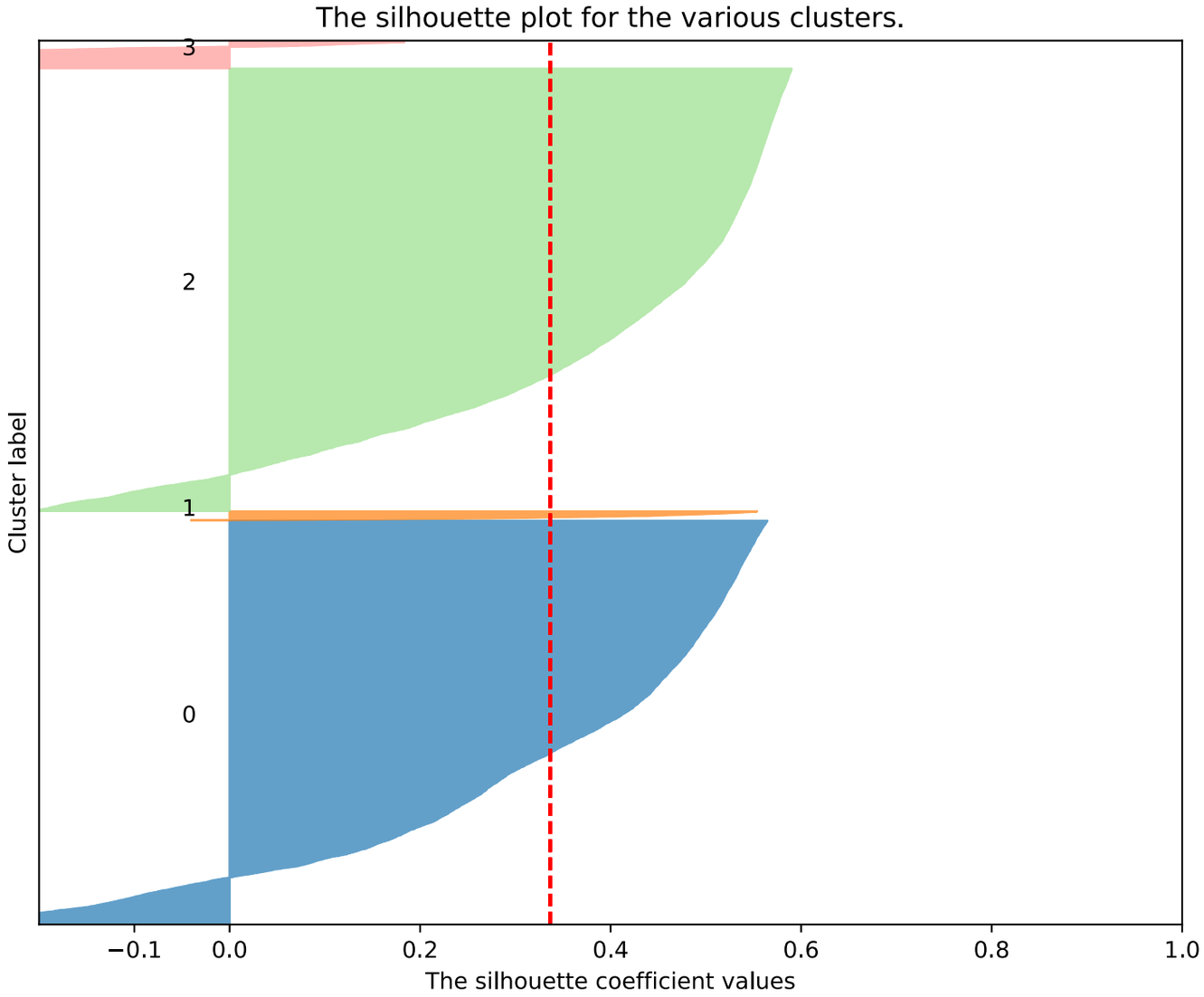
Afficher les journaux d'exécution

2. Algorithme de segmentation agglomerative

Nombre de silhouettes scores négatifs supérieur à la méthode k-means

k-means ++



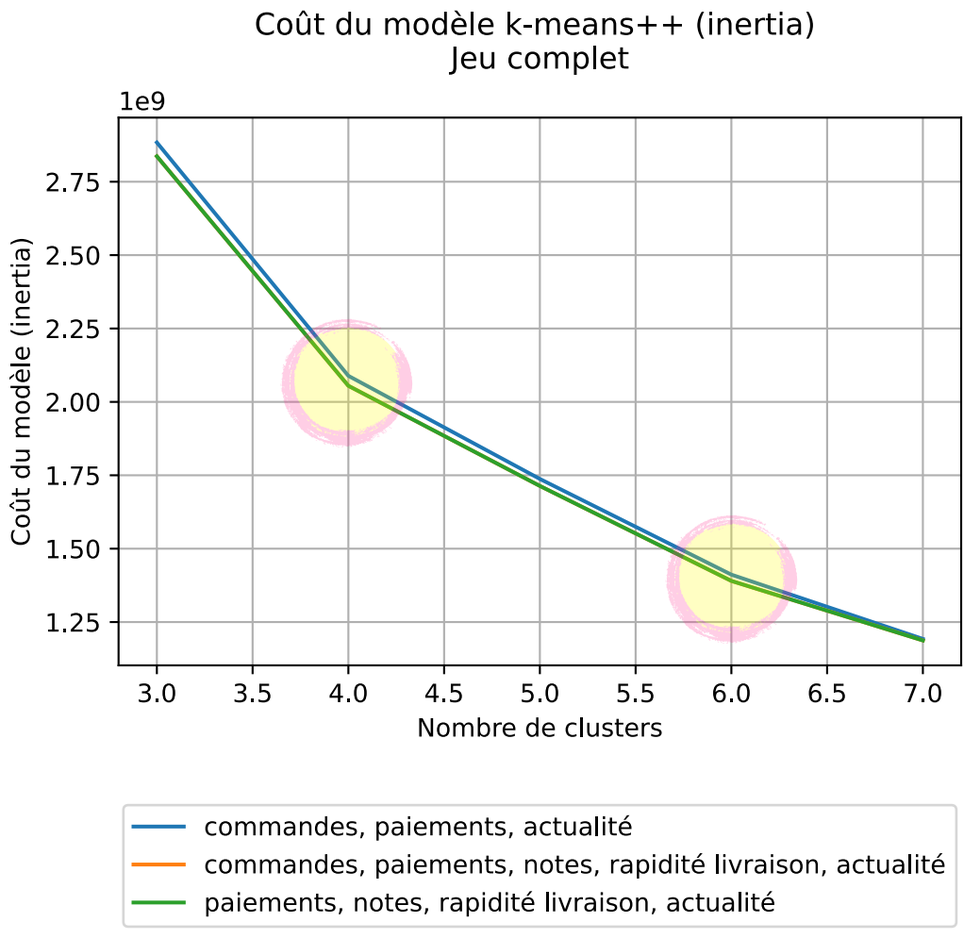


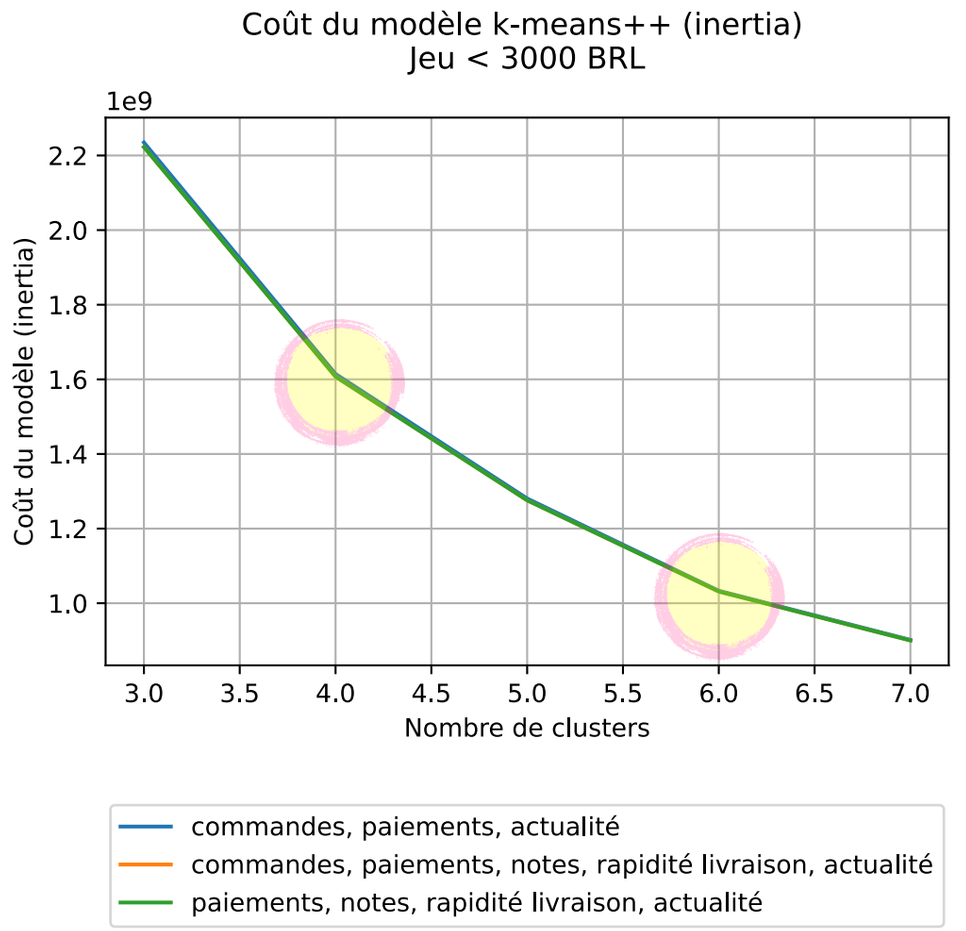
2. Limites de l'algorithme de segmentation agglomerative

1) Limité à une fraction des données
2) Silhouettes scores négatifs en quantité significative

3. Algorithme k-means++

Méthode elbow



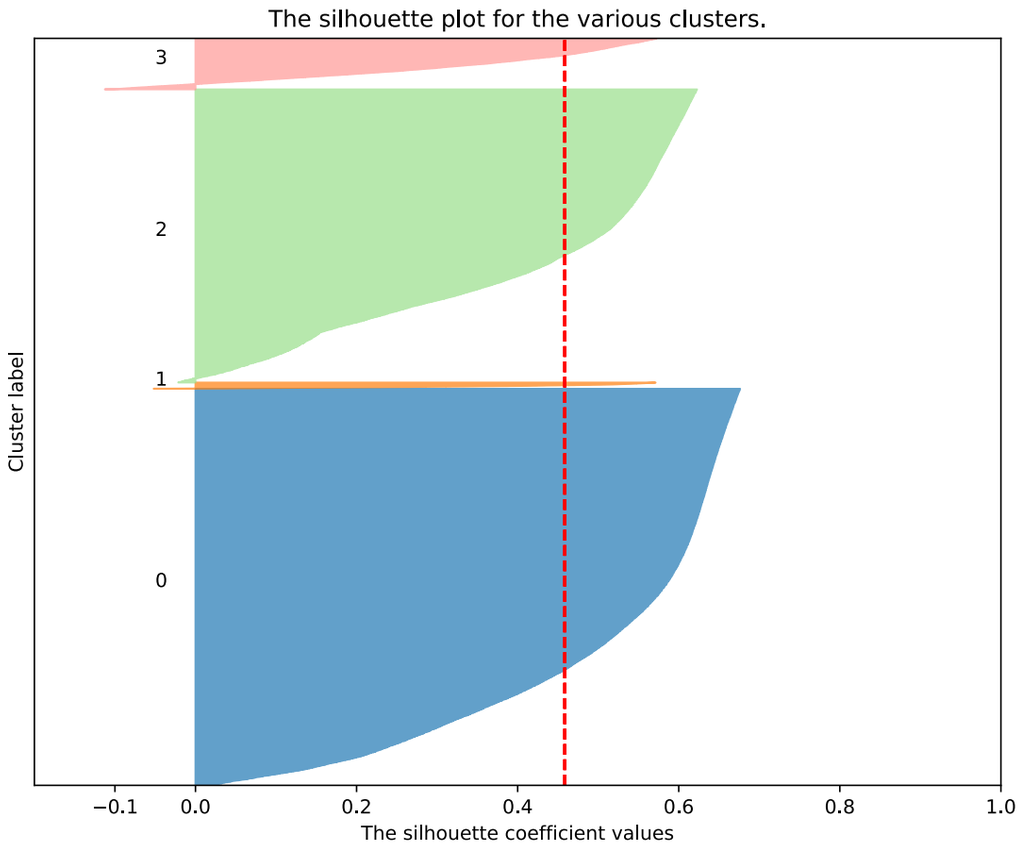


La méthode elbow identifie le nombre optimal de clusters entre 4 et 6

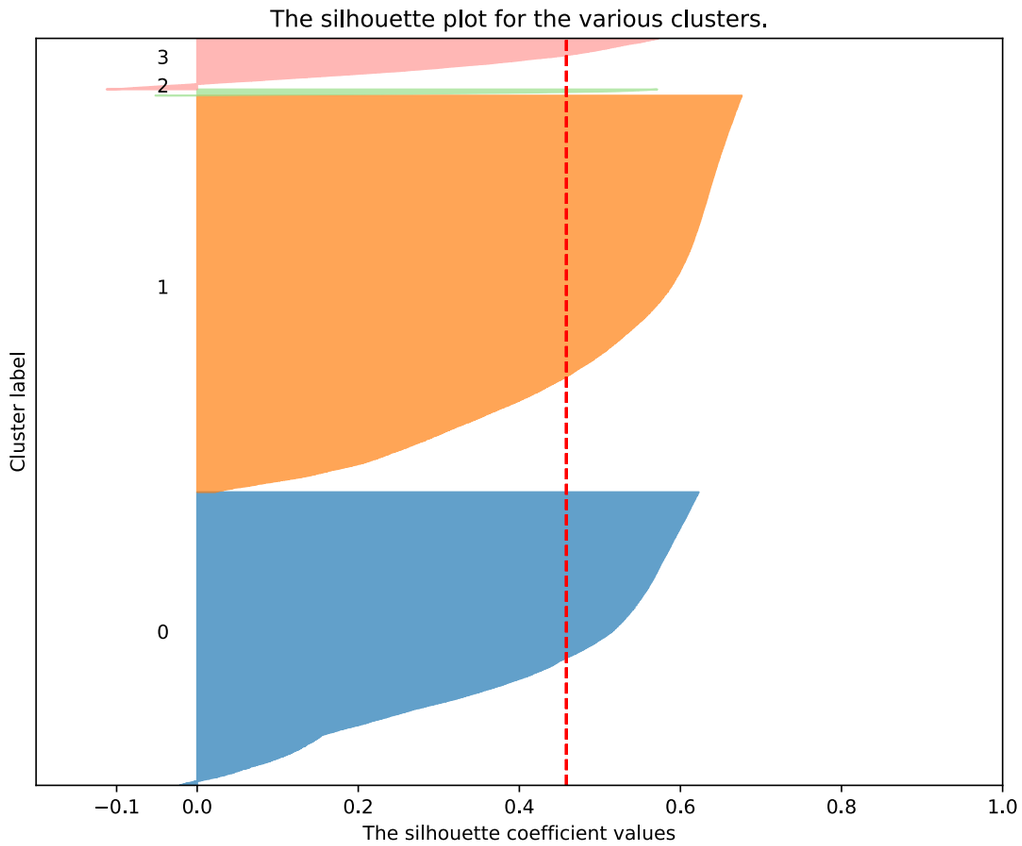
3. Algorithme k-means++

Jeu complet. Méthode silhouette

k-means++



k-means



Performance satisfaisante
k-means++ préférable parce que plus reproductible que k-means

3. Algorithme k-means++

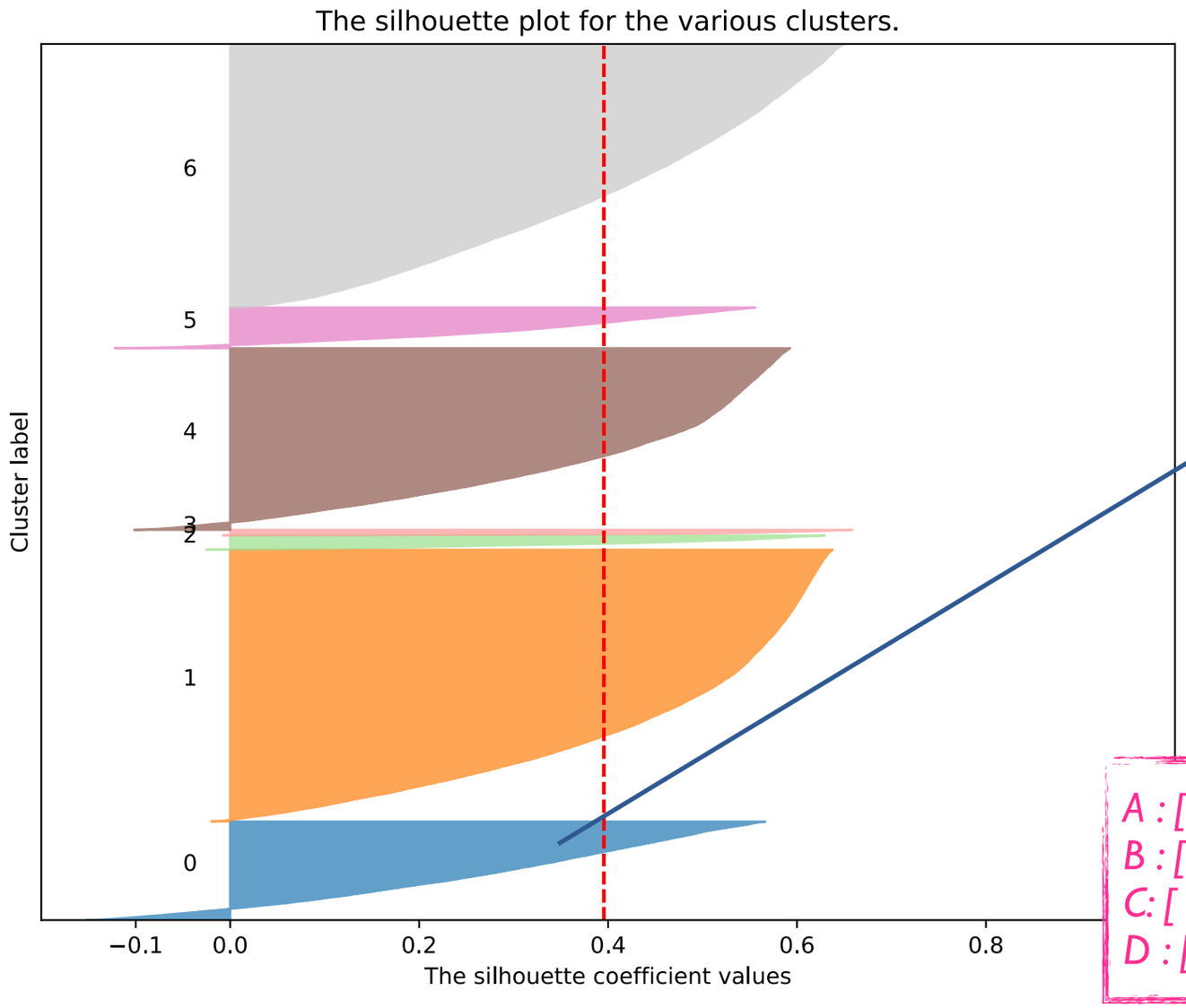
Jeu complet. Apparition de clusters non significatifs à partir de k = 5.

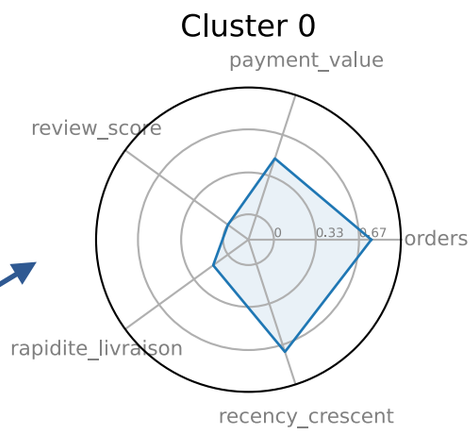
```
# Cluster 2 de 5 (model 1 - k_means): Pourcentage des clients: 0.3% (305)
# Cluster 4 de 6 (model 1 - k_means): Pourcentage des clients: 0.3% (305)
# Cluster 5 de 7 (model 1 - k_means): Pourcentage des clients: 0.0% (7)
# Cluster 6 de 7 (model 1 - k_means): Pourcentage des clients: 0.4% (401)
# Cluster 4 de 5 (model 1 - k_means_pp): Pourcentage des clients: 0.3% (305)
# Cluster 2 de 6 (model 1 - k_means_pp): Pourcentage des clients: 0.3% (305)
# Cluster 6 de 7 (model 1 - k_means_pp): Pourcentage des clients: 0.0% (21)

# Cluster 3 de 5 (model 2 - k_means): Pourcentage des clients: 0.4% (364)
# Cluster 1 de 5 (model 2 - k_means_pp): Pourcentage des clients: 0.3% (308)

# Cluster 3 de 5 (model 3 - k_means): Pourcentage des clients: 0.3% (306)
# Cluster 4 de 5 (model 3 - k_means_pp): Pourcentage des clients: 0.3% (306)
# Cluster 6 de 7 (model 3 - k_means_pp): Pourcentage des clients: 0.1% (71)
```

Le jeu des clients < 3000 BRL ne présente pas ce problème.
Il semble donc admettre une segmentation plus détaillée.

2. Mon indication

Algorithme k-means++, 5 variables, <3000 BRL, 7 segments





Pourcentage des clients: 11.2% (10484)
Poids financier du cluster: 11%

Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Dépenses modérées (162.93 BRL - [158.46 - 167.91] BRL)
Clientèle insatisfaite (4.12 - [4.12 - 4.16])
Livraison très lente (120.15 - [120.06 - 120.69])
Dernier achat assez récent (483.56 - [473.02 - 487.46])

Rating: ACDDB (ratio f/n: 1.0)

A : [troisième quartile - maximum]
B : [mediane - troisième quartile)
C: [premier quartile - mediane)
D : [minimum - premier quartile)

% chiffre d'affaire du cluster
% nombre de clients

2. Mon indication

```
Rating: ACDDB (ratio f/n: 1.0)
Rating: DDCDA (ratio f/n: 1.0)
Rating: ADCBA (ratio f/n: 1.0)
Rating: CDAAD (ratio f/n: 1.0)
Rating: DDDDA (ratio f/n: 1.0)
Rating: AAACA (ratio f/n: 1.0)
Rating: BDCDA (ratio f/n: 1.0)
```

	orders	payment value	review score	rapidite livraison	recency crescent
Cluster	nombre de commande du client sur la période considérée	dépenses totales du client	mesure croissante de l'actualité du dernier achat	note du dernier achat	mesure croissante du décalage entre la date de livraison prévue et la date d'arrivé effective
0	A	C	D	D	B
1	D	D	C	D	A
2	A	D	C	B	A
3	C	D	A	A	D
4	D	D	D	D	A
5	A	A	A	C	A
6	B	D	C	D	A

2. Mon indication

Algorithme k-means++, 5 variables, <3000 BRL, 7 segments

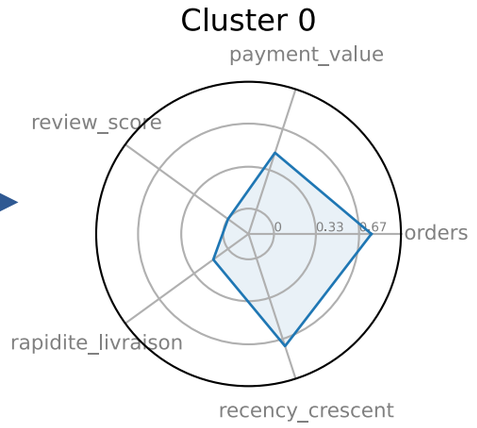
 # Cluster 0 de 7 (model 5 - k_means_pp): Pourcentage des clients: 11.2% (10484)
Poids financier du cluster: 11%
Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Dépenses modérées (162.93 BRL - [158.46 - 167.91] BRL)
Clientèle insatisfaite (4.12 - [4.12 - 4.16])
Livraison très lente (120.15 - [120.06 - 120.69])
Dernier achat assez récent (483.56 - [473.02 - 487.46])

Rating: ACDDB (ratio f/n: 1.0)
--
Cluster 1 de 7 (model 5 - k_means_pp): Pourcentage des clients: 31.1% (28954)
Poids financier du cluster: 31%
Bas nombre de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (159.45 BRL - [158.46 - 167.91] BRL)
Clientèle assez insatisfaite (4.14 - [4.12 - 4.16])
Livraison très lente (120.18 - [120.06 - 120.69])
Dernier achat récent (485.23 - [473.02 - 487.46])

Rating: DDCDA (ratio f/n: 1.0)
--
Cluster 2 de 7 (model 5 - k_means_pp): Pourcentage des clients: 1.6% (1512)
Poids financier du cluster: 2%
Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Dépenses réduites (159.68 BRL - [158.46 - 167.91] BRL)
Clientèle assez insatisfaite (4.13 - [4.12 - 4.16])
Livraison assez rapide (120.42 - [120.06 - 120.69])
Dernier achat récent (487.46 - [473.02 - 487.46])

Livraison très lente (120.14 - [120.06 - 120.69])
Dernier achat récent (485.02 - [473.02 - 487.46])

Rating: BDCDA (ratio f/n: 1.0)



Pourcentage des clients: 11.2% (10484)
Poids financier du cluster: 11%

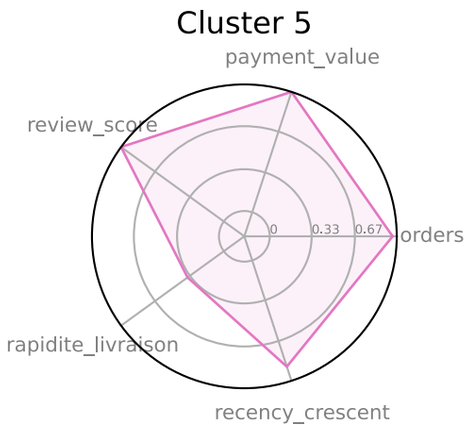
Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Dépenses modérées (162.93 BRL - [158.46 - 167.91] BRL)
Clientèle insatisfaite (4.12 - [4.12 - 4.16])
Livraison très lente (120.15 - [120.06 - 120.69])
Dernier achat assez récent (483.56 - [473.02 - 487.46])

Rating: ACDDB (ratio f/n: 1.0)

2. Mon indication

Algorithme k-means++, 5 variables, <3000 BRL, 7 segments

```
# Cluster 3 de 7 (model 5 - k_means_pp): Pourcentage des clients: 0.6% (586)
Poids financier du cluster: 1%
Nombre modéré de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (159.95 BRL - [158.46 - 167.91] BRL)
Clientèle satisfaite (4.15 - [4.12 - 4.16])
Livraison rapide (120.69 - [120.06 - 120.69])
Dernier achat ancien (473.02 - [473.02 - 487.46])

Rating: CDAAD (ratio f/n: 1.0)
--
# Cluster 4 de 7 (model 5 - k_means_pp): Pourcentage des clients: 20.7% (19343)
Poids financier du cluster: 20%
Bas nombre de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (158.46 BRL - [158.46 - 167.91] BRL)
Clientèle insatisfaite (4.13 - [4.12 - 4.16])
Livraison très lente (120.06 - [120.06 - 120.69])
Dernier achat récent (484.19 - [473.02 - 487.46])

Rating: DDDDA (ratio f/n: 1.0)
--
# Cluster 5 de 7 (model 5 - k_means_pp): Pourcentage des clients: 4.6% (4324)
Poids financier du cluster: 5%
Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Très hautes dépenses (167.91 BRL - [158.46 - 167.91] BRL)
Clientèle satisfaite (4.16 - [4.12 - 4.16])
Livraison lente (120.28 - [120.06 - 120.69])
Dernier achat récent (485.76 - [473.02 - 487.46])

Rating: AAACA (ratio f/n: 1.0)
```



Pourcentage des clients: 4.6% (4324)
Poids financier du cluster: 5%

Très haut nombre de commandes (1.04 - [1.03 - 1.04])
Très hautes dépenses (167.91 BRL - [158.46 - 167.91] BRL)
Clientèle satisfaite (4.16 - [4.12 - 4.16])
Livraison lente (120.28 - [120.06 - 120.69])
Dernier achat récent (485.76 - [473.02 - 487.46])

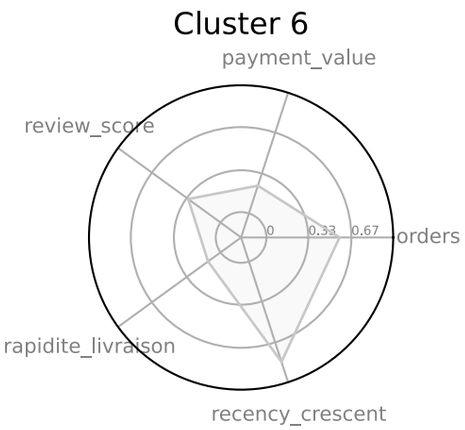
Rating: AAACA (ratio f/n: 1.0)

2. Mon indication

Algorithme k-means++, 5 variables, <3000 BRL, 7 segments

 # Cluster 6 de 7 (model 5 - k_means_pp): Pourcentage des clients: 30.1% (28031)
Poids financier du cluster: 30%
Haut nombre de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (160.63 BRL - [158.46 - 167.91] BRL)
Clientèle assez insatisfaite (4.13 - [4.12 - 4.16])
Livraison très lente (120.14 - [120.06 - 120.69])
Dernier achat récent (485.02 - [473.02 - 487.46])

Rating: BDCDA (ratio f/n: 1.0)

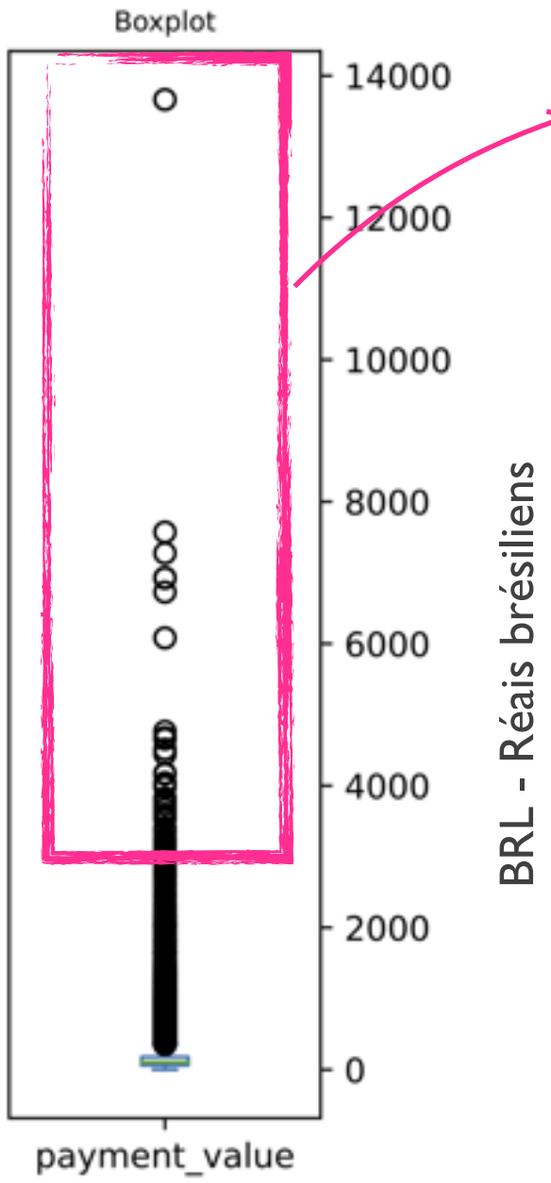


Pourcentage des clients: 30.1% (28031)
Poids financier du cluster: 30%

Haut nombre de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (160.63 BRL - [158.46 - 167.91] BRL)
Clientèle assez insatisfaite (4.13 - [4.12 - 4.16])
Livraison très lente (120.14 - [120.06 - 120.69])
Dernier achat récent (485.02 - [473.02 - 487.46])

Rating: BDCDA (ratio f/n: 1.0)

Analyse des outliers



segment separé

	Moyenne ± écart type		Rating estimé
	Jeu complet (94234 éléments)	Outliers (47 éléments)	
Nombre de commandes	1.03 ± 0.21	1.11 ± 0.47	**A**
Paiements (BRL)	162.54 ± 223.73	4182.27 ± 1869.79	**A+**
Note	4.13 ± 1.31	4.11 ± 1.40	**C**
Rapidité	120.15 ± 8.73	119.47 ± 10.28	**C**
Actualité	484.58 ± 152.91	457.06 ± 200.97	**C/D**

```
# Cluster 6 de 7 (model 5 - k_means_pp): Pourcentage des clients: 30.1% (28031)
Poids financier du cluster: 30%
Haut nombre de commandes (1.03 - [1.03 - 1.04])
Dépenses réduites (160.63 BRL - [158.46 - 167.91] BRL)
Clientèle assez insatisfaite (4.13 - [4.12 - 4.16])
Livraison très lente (120.14 - [120.06 - 120.69])
Dernier achat récent (485.02 - [473.02 - 487.46])

Rating: BDCDA (ratio f/n: 1.0)
```

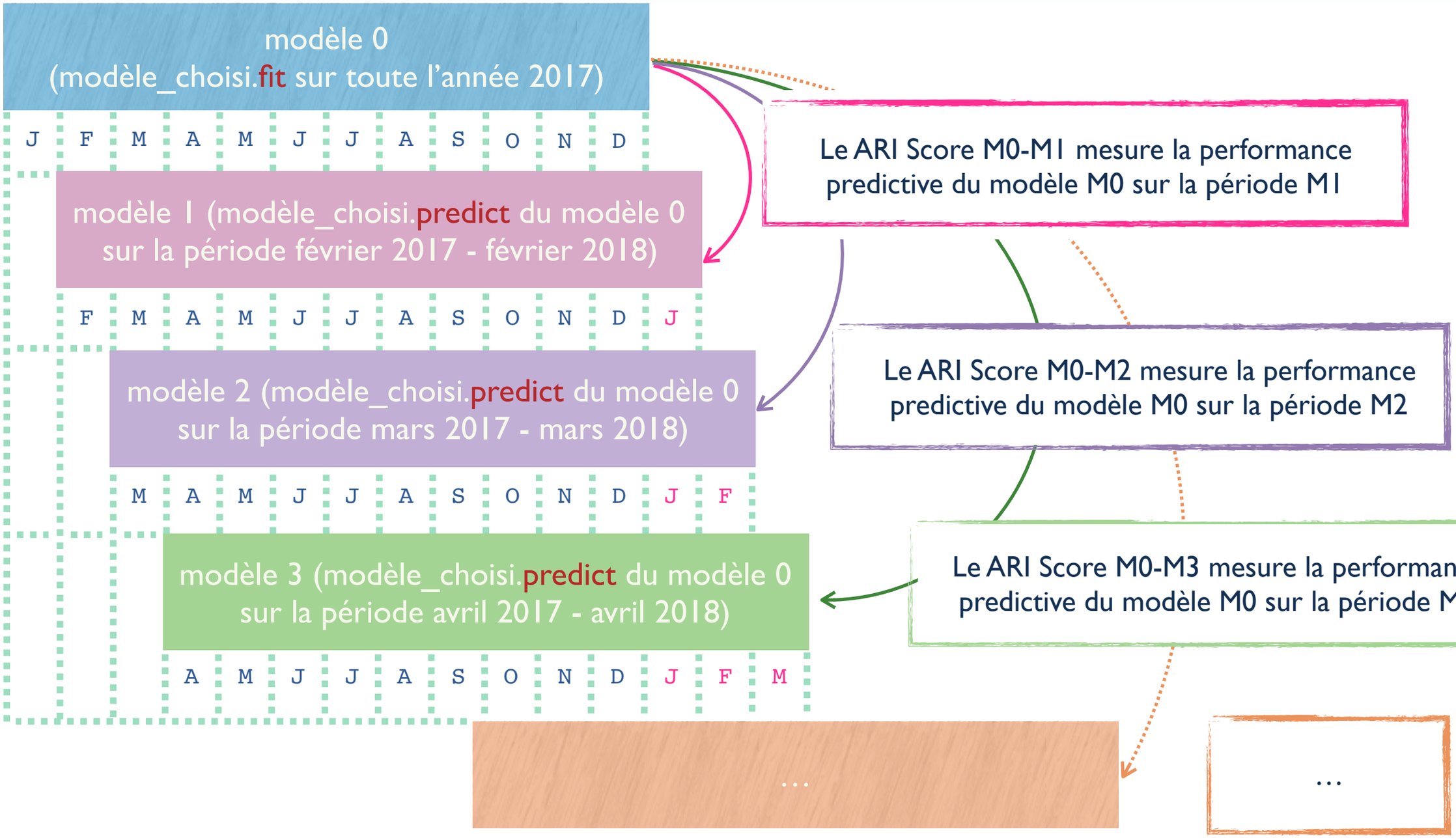
moyennes par segment

3 - Renouvellement de la segmentation

olist

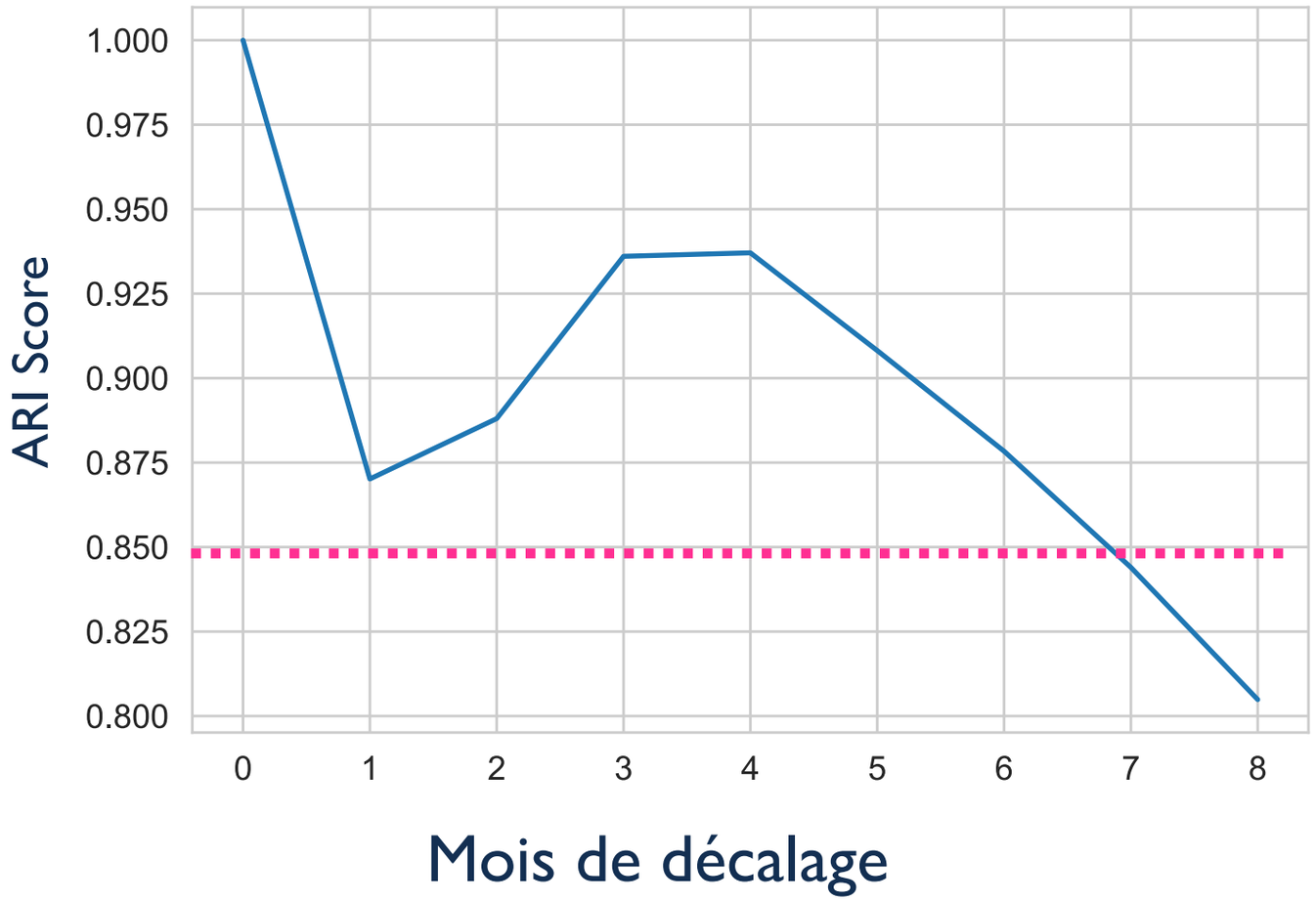
empowering commerce

Stabilité sur l'année glissante



Stabilité sur l'année glissante

```
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M1 (période 01-02-2017 --- 01-02-2018) : 0.877
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M2 (période 03-03-2017 --- 03-03-2018) : 0.894
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M3 (période 02-04-2017 --- 02-04-2018) : 0.938
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M4 (période 02-05-2017 --- 02-05-2018) : 0.936
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M5 (période 01-06-2017 --- 01-06-2018) : 0.902
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M6 (période 01-07-2017 --- 01-07-2018) : 0.871
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M7 (période 31-07-2017 --- 31-07-2018) : 0.839
ARI score entre le modèle 0 (période 02-01-2017 --- 02-01-2018) et le modèle M8 (période 30-08-2017 --- 30-08-2018) : 0.800
```



Terme proposé pour le renouvellement de la segmentation:
6 mois

4 - Perspectives

olist

empowering commerce

Points d'évolution

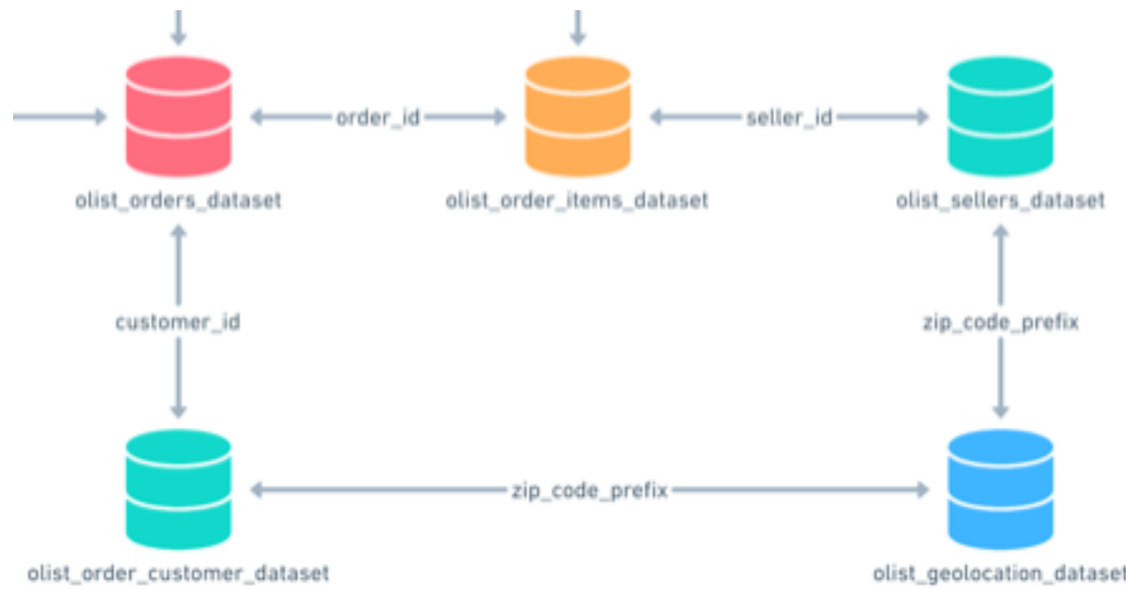
1) Analyse/segmentation par vendeur
2) Préciser l'analyse avec des nouvelles features

Merci

Description de la base de données

Nom du fichier	Taille	
olist_customers_dataset.csv	8.6M	**Caractéristiques des clients**

```
Nettoyage du jeu de données: "0"
99441 5
Après dropna, cols: 99441 5
Colonnes de moins:
 0
Après dropna, lignes: 99441 5
Lignes de moins:
 0
```

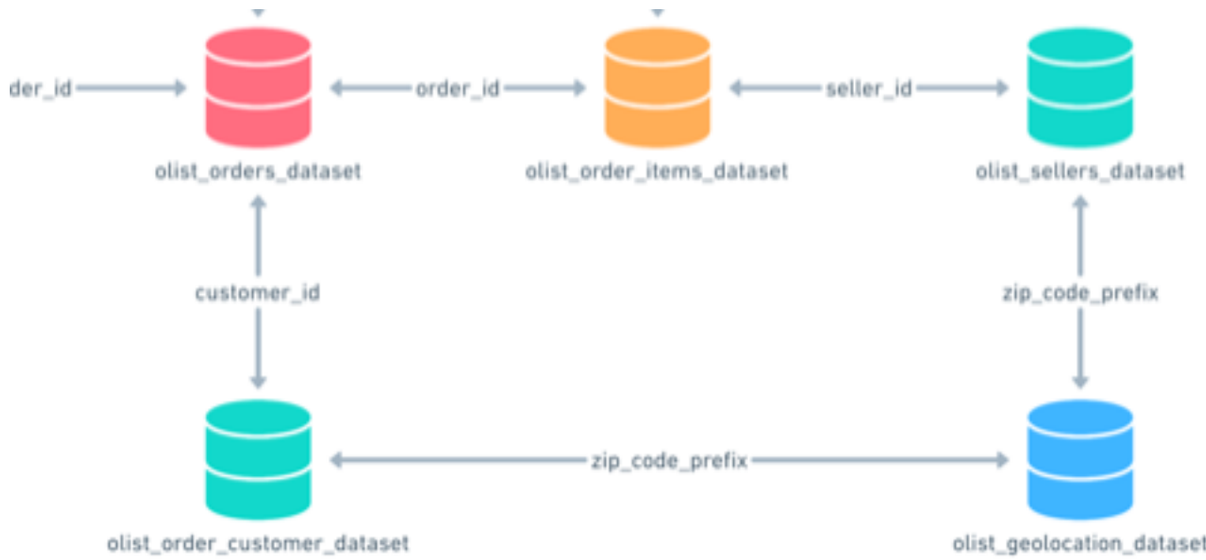



```
Remove duplicates -> Lignes de moins:0
                  Indicator  count  percentage  vides
0               customer_id  99441      100.00      0
1         customer_unique_id  99441      100.00      0
2  customer_zip_code_prefix  99441      100.00      0
3             customer_city  99441      100.00      0
4            customer_state  99441      100.00      0

--------------------------------

0 olist_customers_dataset

(99441, 5)

<bound method IndexOpsMixin.tolist of Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')>

      customer_id       customer_unique_id     customer_zip_code_prefix     customer_city     customer_state
0     06b8999e2fba1a1fbc88172c00ba8bc7    861eff4711a542e4b93843c6dd7febb0    14409       franca      SP
1     18955e83d337fd6b2def6b18a428ac77    290c77bc529b7ac935b93aa66c333dc3    9790  sao bernardo do campo   SP
```

Description de la base de données

Nom du fichier	Taille	
olist_geolocation_dataset.csv	58M	Aide au repérage géographique des commandes géographiques

```
Nettoyage du jeu de données: "1"
1000163 5
Après dropna, cols: 1000163 5
Colonnes de moins:
 0
Après dropna, lignes: 1000163 5
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:261831
                    Indicator   count  percentage   vides
0  geolocation_zip_code_prefix  738332       73.82  261831
1              geolocation_lat  738332       73.82  261831
2              geolocation_lng  738332       73.82  261831
3             geolocation_city  738332       73.82  261831
4            geolocation_state  738332       73.82  261831

--------------------------------

1 olist_geolocation_dataset

(738332, 5)

<bound method IndexOpsMixin.tolist of Index(['geolocation_zip_code_prefix', 'geolocation_lat', 'geolocation_lng',
       'geolocation_city', 'geolocation_state'],
      dtype='object')>

      geolocation_zip_code_prefix   geolocation_lat   geolocation_lng   geolocation_city  geolocation_state
0     1037  -23.55      -46.64      sao paulo   SP
1     1046  -23.55      -46.64      sao paulo   SP
```

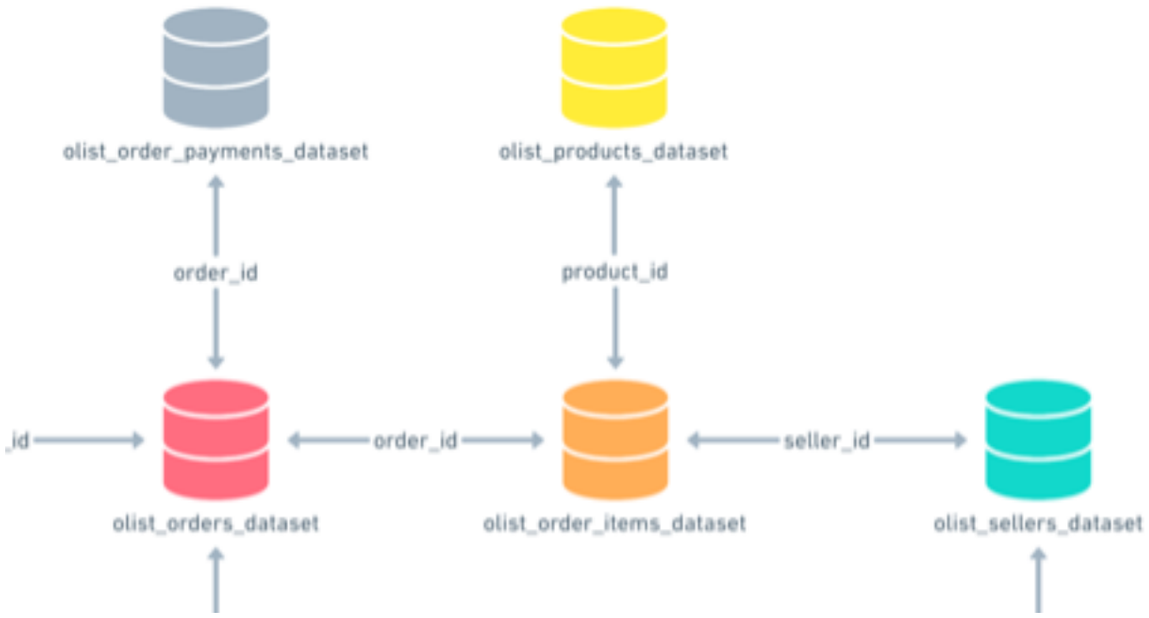


Description de la base de données

Nom du fichier	**Taille**	
olist_order_items_dataset.csv	15M	Détails des produits par commande

```
Nettoyage du jeu de données: "2"
112650 7
Après dropna, cols: 112650 7
Colonnes de moins:
 0
Après dropna, lignes: 112650 7
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:0
```

	Indicator	count	percentage	vides
0	order_id	112650	100.00	0
1	order_item_id	112650	100.00	0
2	product_id	112650	100.00	0
3	seller_id	112650	100.00	0
4	shipping_limit_date	112650	100.00	0
5	price	112650	100.00	0
6	freight_value	112650	100.00	0




```
--------------------------------

2 olist_order_items_dataset

(112650, 7)

<bound method IndexOpsMixin.tolist of Index(['order_id', 'order_item_id', 'product_id', 'seller_id',
       'shipping_limit_date', 'price', 'freight_value'],
      dtype='object')>
```

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.90	13.29
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03 11:05:13	239.90	19.93

Description de la base de données

Nom du fichier	Taille	
olist_order_payments_dataset.csv	5.5M	**Détails des paiements**

```
Nettoyage du jeu de données: "3"
103886 5
Après dropna, cols: 103886 5
Colonnes de moins:
 0
Après dropna, lignes: 103886 5
Lignes de moins:
 0
```


```
Remove duplicates -> Lignes de moins:0
              Indicator   count  percentage  vides
0              order_id  103886      100.00      0
1    payment_sequential  103886      100.00      0
2          payment_type  103886      100.00      0
3  payment_installments  103886      100.00      0
4         payment_value  103886      100.00      0

--------------------------------

3 olist_order_payments_dataset

(103886, 5)

<bound method IndexOpsMixin.tolist of Index(['order_id', 'payment_sequential', 'payment_type',
       'payment_installments', 'payment_value'],
      dtype='object')>

      order_id    payment_sequential      payment_type      payment_installments    payment_value
0     b81ef226f3fe1789b1e8b2acac839d17    1     credit_card       8     99.33
1     a9810da82917af2d9aefd1278f1dcfa0    1     credit_card       1     24.39
```

Description de la base de données

Nom du fichier	Taille	
olist_order_reviews_dataset.csv	14M	**Détails des notes des commandes**

```
Nettoyage du jeu de données: "4"
99224 7
Après dropna, cols: 99224 7
Colonnes de moins:
 0
Après dropna, lignes: 99224 7
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:0
                Indicator  count  percentage  vides
0               review_id  99224      100.00      0
1                order_id  99224      100.00      0
2            review_score  99224      100.00      0
3    review_comment_title  11568       11.66  87656
4  review_comment_message  40977       41.30  58247
5    review_creation_date  99224      100.00      0
6  review_answer_timestamp  99224      100.00      0

--------------------------------

4 olist_order_reviews_dataset

(99224, 7)

<bound method IndexOpsMixin.tolist of Index(['review_id', 'order_id', 'review_score', 'review_comment_title',
       'review_comment_message', 'review_creation_date',
       'review_answer_timestamp'],
      dtype='object')>

      review_id     order_id      review_score  review_comment_title        review_comment_message      review_creation_date
review_answer_timestamp
0      7bc2406110b926393aa56f80a40eba40   73fc7af87114b39712e6da79b0a377eb   4      NaN    NaN    2018-01-18 00:00:00  2018-01-18 21:46:59
1      80e641a11e56f04c1ad469d5645fdfde   a548910a1c6147796b98fdf73dbeba33   5      NaN    NaN    2018-03-10 00:00:00  2018-03-11 03:05:13
```

Description de la base de données

Nom du fichier	Taille	
olist_orders_dataset.csv	17M	**Détails des commandes**

```
Nettoyage du jeu de données: "5"
99441 8
Après dropna, cols: 99441 8
Colonnes de moins:
 0
Après dropna, lignes: 99441 8
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:0
                       Indicator  count  percentage  vides
0                       order_id  99441      100.00      0
1                    customer_id  99441      100.00      0
2                   order_status  99441      100.00      0
3        order_purchase_timestamp  99441      100.00      0
4                order_approved_at  99281       99.84    160
5    order_delivered_carrier_date  97658       98.21   1783
6   order_delivered_customer_date  96476       97.02   2965
7   order_estimated_delivery_date  99441      100.00      0

--------------------------------

5 olist_orders_dataset

(99441, 8)

<bound method IndexOpsMixin.tolist of Index(['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp',
       'order_approved_at', 'order_delivered_carrier_date',
       'order_delivered_customer_date', 'order_estimated_delivery_date'],
      dtype='object')>
```


```
         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        e481f51cbdc54678b7cc49136f2d6af7           9ef432eb6251297304e76186b10a928d  delivered  2017-10-02 10:56:33   2017-10-02 11:07:15   2017-10-04 19:55:00   2017-10-10 21:25:13   2017-10-18 00:00:00
1        53cdb2fc8bc7dce0b6741e2150273451           b0830fb4747a6c6d20dea0b8c802d7ef  delivered  2018-07-24 20:41:37   2018-07-26 03:24:27   2018-07-26 14:31:00   2018-08-07 15:27:45   2018-08-13 00:00:00
```

Description de la base de données

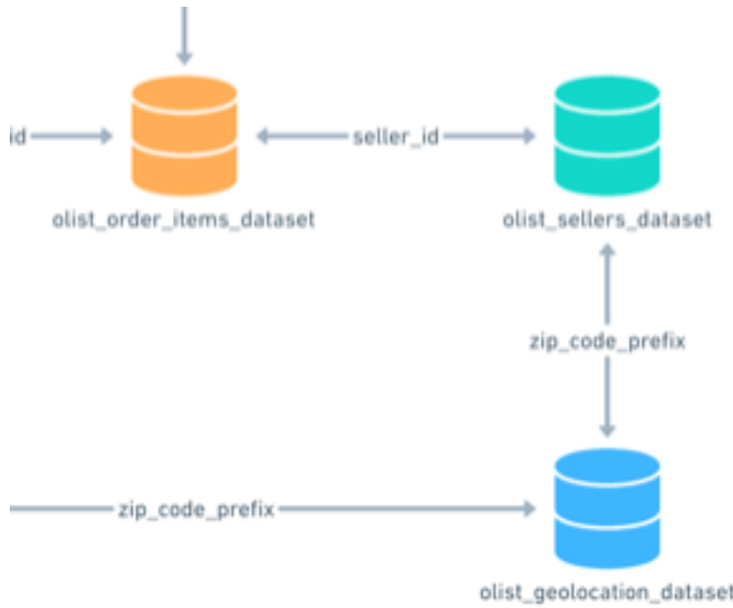
Nom du fichier	Taille	
olist_products_dataset.csv	2.3M	Détails des produits commandés

```
Nettoyage du jeu de données: "6"
32951 9
Après dropna, cols: 32951 9
Colonnes de moins:
 0
Après dropna, lignes: 32951 9
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:0
                   Indicator  count  percentage  vides
0                  product_id  32951      100.00      0
1        product_category_name  32341       98.15    610
2          product_name_lenght  32341       98.15    610
3   product_description_lenght  32341       98.15    610
4           product_photos_qty  32341       98.15    610
5             product_weight_g  32949       99.99      2
6            product_length_cm  32949       99.99      2
7            product_height_cm  32949       99.99      2
8             product_width_cm  32949       99.99      2

--------------------------------

6 olist_products_dataset

(32951, 9)

<bound method IndexOpsMixin.tolist of Index(['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'],
      dtype='object')>

       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
0      1e9e8ef04dbcff4541ed26657ea517e5   perfumaria    40.00  287.00        1.00   225.00        16.00  10.00  14.00
1      3aa071139cb16b67ca9e5dea641aaa2f   artes  44.00  276.00        1.00   1000.00       30.00  18.00  20.00
```

Description de la base de données

Nom du fichier	Taille	
olist_sellers_dataset.csv	172K	**Détails sur les vendeurs**

```
Nettoyage du jeu de données: "7"
3095 4
Après dropna, cols: 3095 4
Colonnes de moins:
 0
Après dropna, lignes: 3095 4
Lignes de moins:
 0
```




```
Remove duplicates -> Lignes de moins:0
                Indicator  count  percentage  vides
0               seller_id   3095      100.00      0
1  seller_zip_code_prefix   3095      100.00      0
2             seller_city   3095      100.00      0
3            seller_state   3095      100.00      0

--------------------------------

7 olist_sellers_dataset

(3095, 4)

<bound method IndexOpsMixin.tolist of Index(['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state'],
dtype='object')>

      seller_id   seller_zip_code_prefix  seller_city       seller_state
0     3442f8959a84dea7ee197c632cb2df15    13023       campinas    SP
1     d1b65fc7debc3361ea86b5f14c68d2e2    13844       mogi guacu  SP
```

Description de la base de données

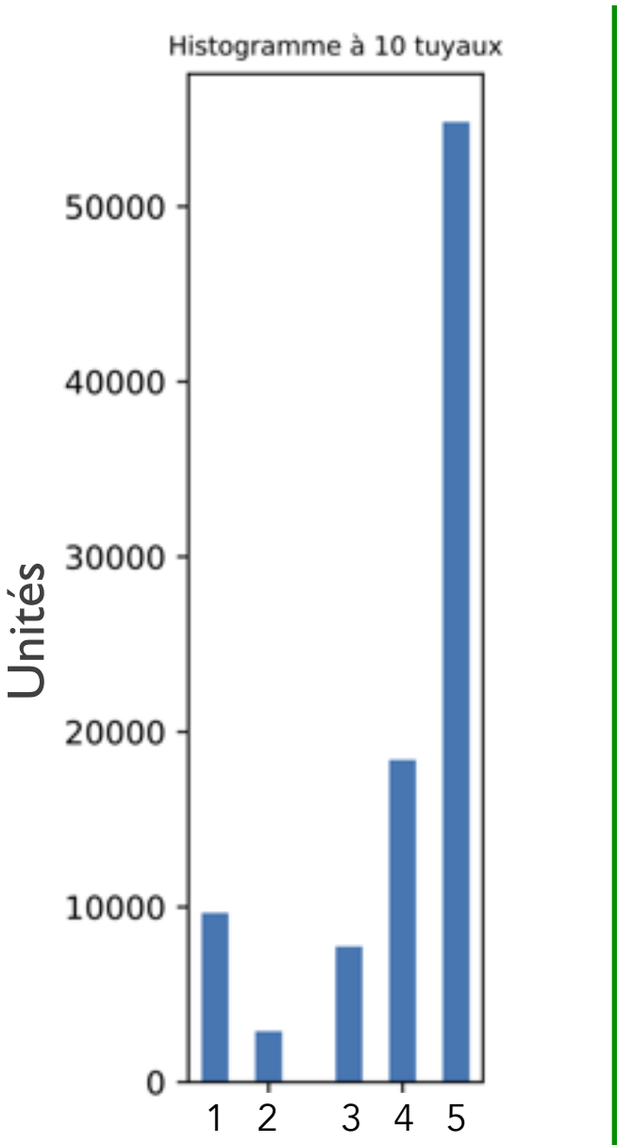
Nom du fichier	Taille	
product_category_name_translation.csv	4.0K	Traduction en anglais des catégories des produits

```
Nettoyage du jeu de données: "8"
71 2
Après dropna, cols: 71 2
Colonnes de moins:
 0
Après dropna, lignes: 71 2
Lignes de moins:
 0

Remove duplicates -> Lignes de moins:0
                       Indicator  count  percentage  vides
0          product_category_name     71      100.00      0
1  product_category_name_english     71      100.00      0

--------------------------------

8 product_category_name_translation

(71, 2)

<bound method IndexOpsMixin.tolist of Index(['product_category_name', 'product_category_name_english'], dtype='object')>

      product_category_name   product_category_name_english
0     beleza_saude      health_beauty
1     informatica_acessorios  computers_accessories
```

Distributions

Distributions: notes et commandes

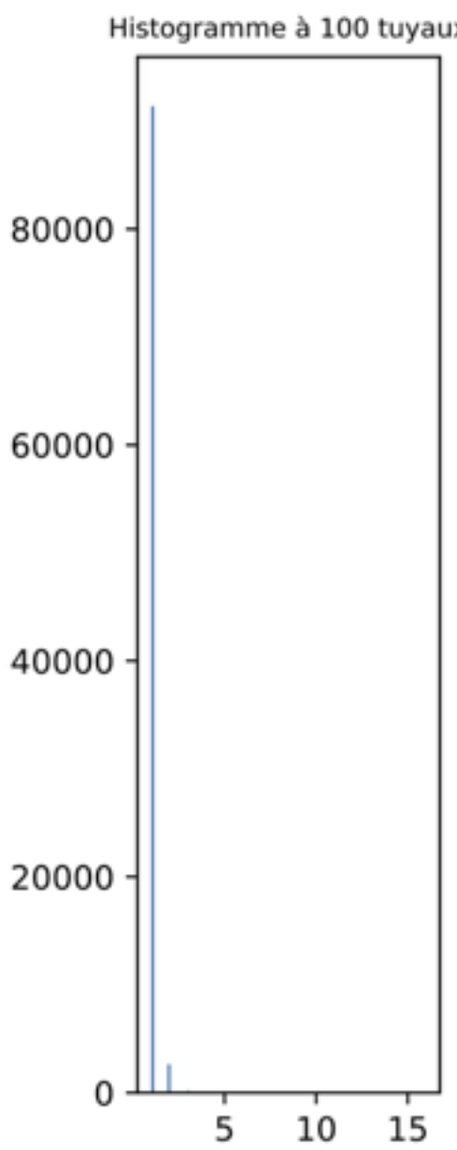
Notes de satisfaction



94234 values

Moyenne = 4.13

Écart type = 1.31

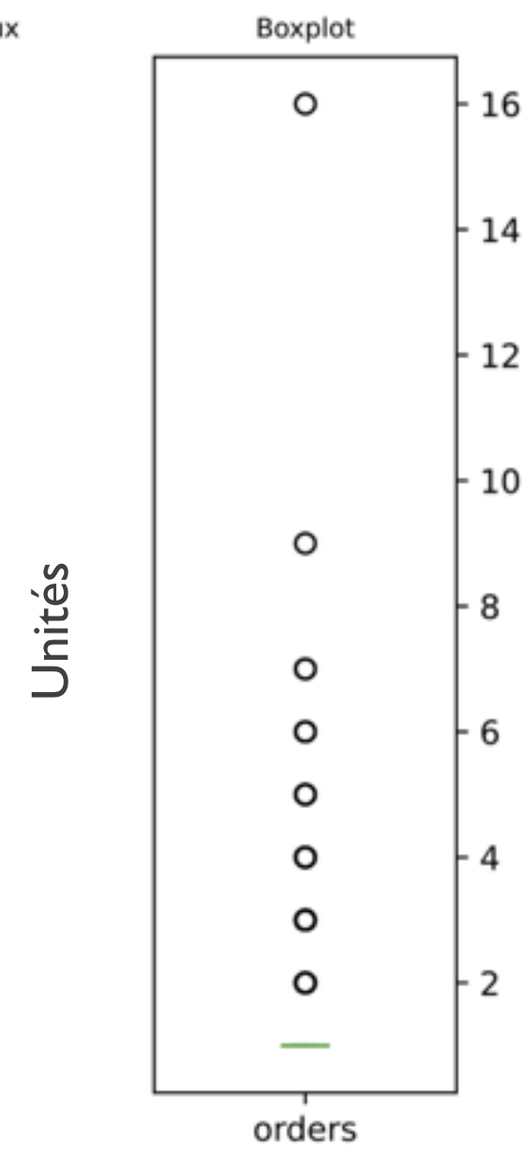


94234 values

Moyenne = 1.03

Écart type = 0.21

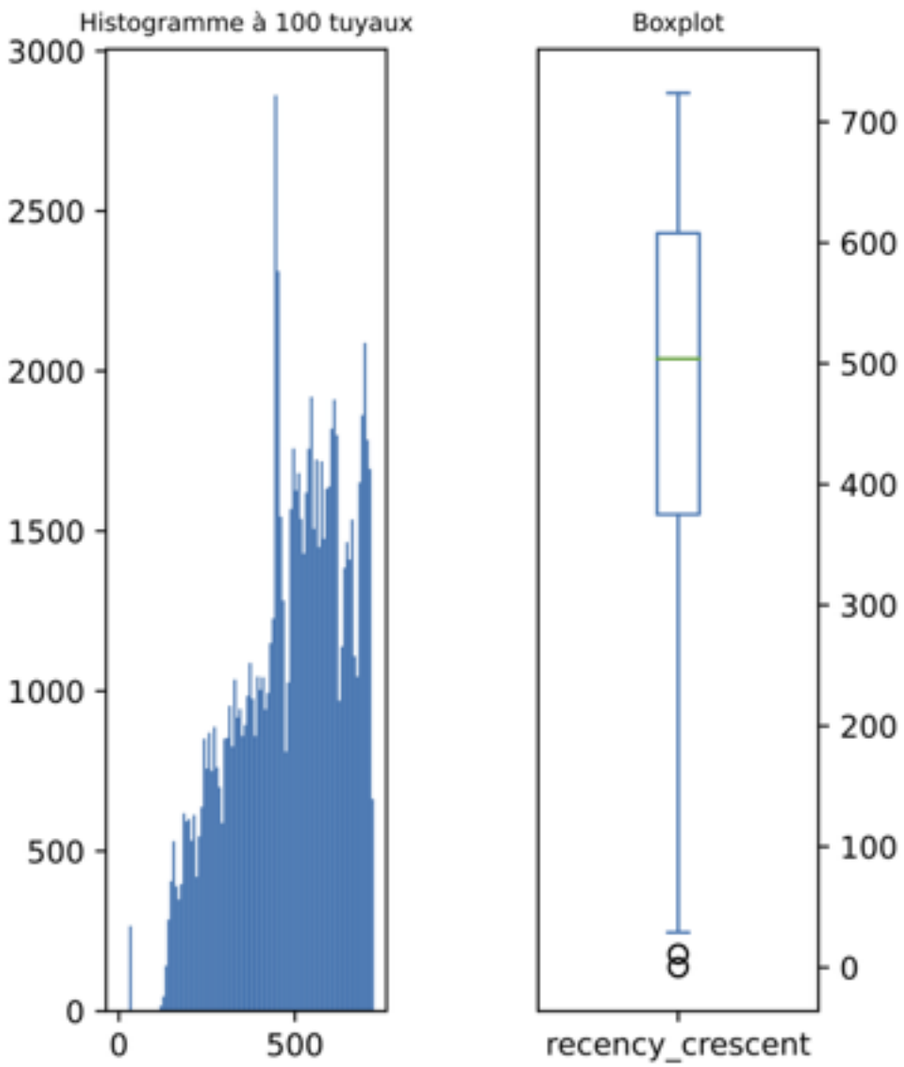
Nombre de commandes par client



~3k clients qui ont acheté à nouveau:
(3.11%, soit le 5.6% du CA)

Distributions: actualité et rapidité

Actualité de la dernière commande (mesure croissante avec l'actualité)

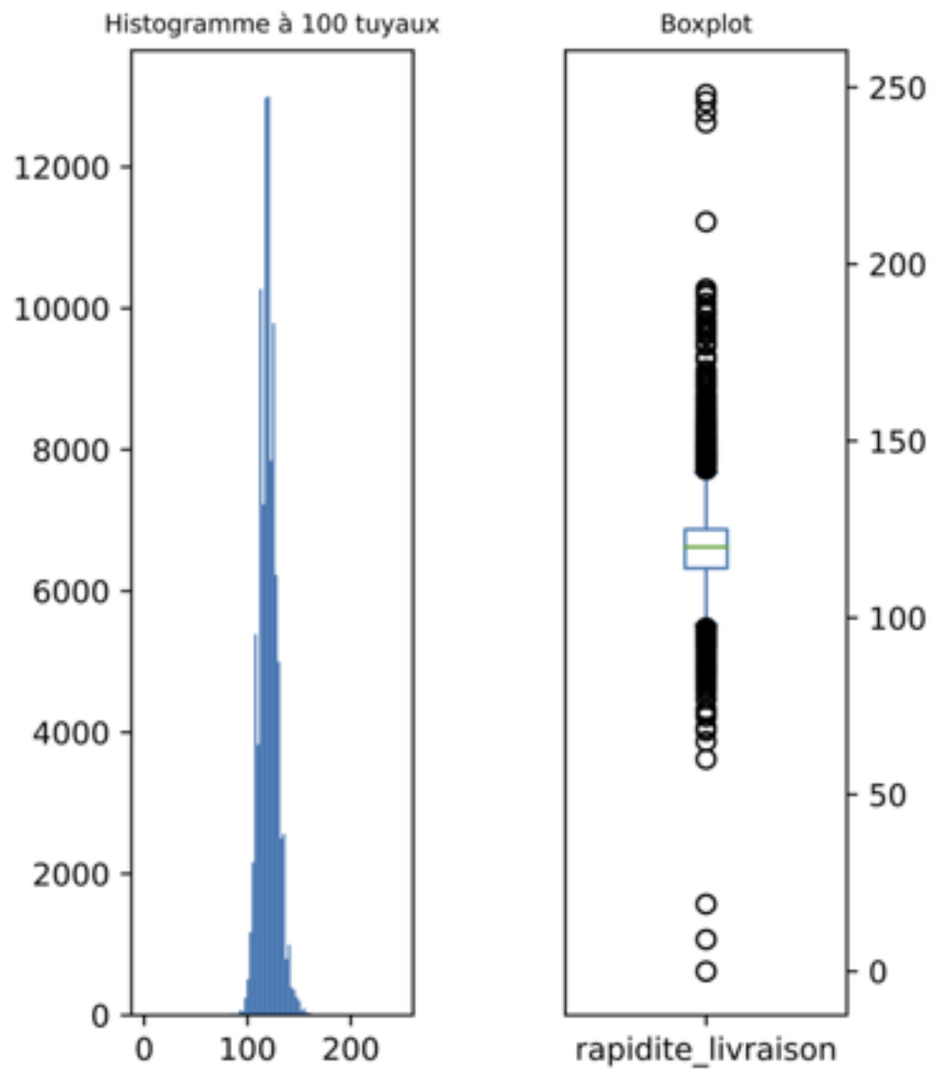


94234 values

Moyenne = 484.59

Écart type = 152.91

Rapidité de la livraison (écart entre date prévue et date effective de livraison, mesure croissante avec l'actualité)



94234 values

Moyenne = 120.15

Écart type = 8.74

DBSCAN

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 1
['orders', 'payment_value', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.510000 . Estimated number of clusters: 2 . Estimated number of noise points: 810 / 20000 ( 0.04 )
Epsilon= 0.560000 . Estimated number of clusters: 2 . Estimated number of noise points: 668 / 20000 ( 0.03 )
Epsilon= 0.610000 . Estimated number of clusters: 2 . Estimated number of noise points: 520 / 20000 ( 0.03 )
Epsilon= 0.660000 . Estimated number of clusters: 2 . Estimated number of noise points: 424 / 20000 ( 0.02 )
Epsilon= 0.710000 . Estimated number of clusters: 2 . Estimated number of noise points: 371 / 20000 ( 0.02 )
Epsilon= 0.760000 . Estimated number of clusters: 2 . Estimated number of noise points: 335 / 20000 ( 0.02 )
Epsilon= 0.810000 . Estimated number of clusters: 2 . Estimated number of noise points: 299 / 20000 ( 0.01 )
Epsilon= 0.860000 . Estimated number of clusters: 2 . Estimated number of noise points: 282 / 20000 ( 0.01 )
Epsilon= 0.910000 . Estimated number of clusters: 2 . Estimated number of noise points: 265 / 20000 ( 0.01 )
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 1
['orders', 'payment_value', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.510000 . Estimated number of clusters: 2 . Estimated number of noise points: 810 / 20000 ( 0.04 )
Epsilon= 0.560000 . Estimated number of clusters: 2 . Estimated number of noise points: 668 / 20000 ( 0.03 )
Epsilon= 0.610000 . Estimated number of clusters: 2 . Estimated number of noise points: 520 / 20000 ( 0.03 )
Epsilon= 0.660000 . Estimated number of clusters: 2 . Estimated number of noise points: 424 / 20000 ( 0.02 )
Epsilon= 0.710000 . Estimated number of clusters: 2 . Estimated number of noise points: 371 / 20000 ( 0.02 )
Epsilon= 0.760000 . Estimated number of clusters: 2 . Estimated number of noise points: 335 / 20000 ( 0.02 )
Epsilon= 0.810000 . Estimated number of clusters: 2 . Estimated number of noise points: 299 / 20000 ( 0.01 )
Epsilon= 0.860000 . Estimated number of clusters: 2 . Estimated number of noise points: 282 / 20000 ( 0.01 )
Epsilon= 0.910000 . Estimated number of clusters: 2 . Estimated number of noise points: 265 / 20000 ( 0.01 )
```

[…]
Model 2 - [['orders', 'payment_value', 'review_score',

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 2
['orders', 'payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.010000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.060000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.110000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 8 . Estimated number of noise points: 17046 / 20000 ( 0.85 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 13393 / 20000 ( 0.67 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 10906 / 20000 ( 0.55 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 8542 / 20000 ( 0.43 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 6596 / 20000 ( 0.33 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5504 / 20000 ( 0.28 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 4381 / 20000 ( 0.22 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 3758 / 20000 ( 0.19 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 3183 / 20000 ( 0.16 )
Epsilon= 0.660000 . Estimated number of clusters: 5 . Estimated number of noise points: 2733 / 20000 ( 0.14 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 2360 / 20000 ( 0.12 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 2042 / 20000 ( 0.10 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1721 / 20000 ( 0.09 )
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 2
['orders', 'payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.010000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.060000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.110000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 8 . Estimated number of noise points: 17046 / 20000 ( 0.85 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 13393 / 20000 ( 0.67 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 10906 / 20000 ( 0.55 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 8542 / 20000 ( 0.43 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 6596 / 20000 ( 0.33 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5504 / 20000 ( 0.28 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 4381 / 20000 ( 0.22 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 3758 / 20000 ( 0.19 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 3183 / 20000 ( 0.16 )
Epsilon= 0.660000 . Estimated number of clusters: 5 . Estimated number of noise points: 2733 / 20000 ( 0.14 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 2360 / 20000 ( 0.12 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 2042 / 20000 ( 0.10 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1721 / 20000 ( 0.09 )
```

[…]

Model 3 - [['payment_value', 'review_score', 'rapidite_livraison', 'recency_crescent'], 'df_olist_clean_all',

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 3
['payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.010000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.060000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.110000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 8 . Estimated number of noise points: 16987 / 20000 ( 0.85 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 13245 / 20000 ( 0.66 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 10664 / 20000 ( 0.53 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 8209 / 20000 ( 0.41 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 6145 / 20000 ( 0.31 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5045 / 20000 ( 0.25 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 3925 / 20000 ( 0.20 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 3270 / 20000 ( 0.16 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 2713 / 20000 ( 0.14 )
Epsilon= 0.660000 . Estimated number of clusters: 5 . Estimated number of noise points: 2224 / 20000 ( 0.11 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 1822 / 20000 ( 0.09 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 1508 / 20000 ( 0.08 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1156 / 20000 ( 0.06 )
Epsilon= 0.860000 . Estimated number of clusters: 1 . Estimated number of noise points: 932 / 20000 ( 0.05 )
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 3
['payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent']
Jeu complet

```
Epsilon= 0.010000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.060000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.110000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 8 . Estimated number of noise points: 16987 / 20000 ( 0.85 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 13245 / 20000 ( 0.66 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 10664 / 20000 ( 0.53 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 8209 / 20000 ( 0.41 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 6145 / 20000 ( 0.31 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5045 / 20000 ( 0.25 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 3925 / 20000 ( 0.20 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 3270 / 20000 ( 0.16 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 2713 / 20000 ( 0.14 )
Epsilon= 0.660000 . Estimated number of clusters: 5 . Estimated number of noise points: 2224 / 20000 ( 0.11 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 1822 / 20000 ( 0.09 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 1508 / 20000 ( 0.08 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1156 / 20000 ( 0.06 )
Epsilon= 0.860000 . Estimated number of clusters: 1 . Estimated number of noise points: 932 / 20000 ( 0.05 )

[…]
4 [['orders', 'payment_value', 'recency_crescent'], 'df_olist_clean_seuil_3000', '40-6-6-1-3000-rfm',
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 4
['orders', 'payment_value', 'recency_crescent']
Jeu < 3000 BRL

```
Epsilon= 0.360000 . Estimated number of clusters: 1 . Estimated number of noise points: 1478 / 20000 ( 0.07 )
Epsilon= 0.410000 . Estimated number of clusters: 1 . Estimated number of noise points: 1363 / 20000 ( 0.07 )
Epsilon= 0.460000 . Estimated number of clusters: 1 . Estimated number of noise points: 1247 / 20000 ( 0.06 )
Epsilon= 0.510000 . Estimated number of clusters: 2 . Estimated number of noise points: 1006 / 20000 ( 0.05 )
Epsilon= 0.560000 . Estimated number of clusters: 2 . Estimated number of noise points: 763 / 20000 ( 0.04 )
Epsilon= 0.610000 . Estimated number of clusters: 2 . Estimated number of noise points: 621 / 20000 ( 0.03 )
Epsilon= 0.660000 . Estimated number of clusters: 2 . Estimated number of noise points: 491 / 20000 ( 0.02 )
Epsilon= 0.710000 . Estimated number of clusters: 2 . Estimated number of noise points: 409 / 20000 ( 0.02 )
Epsilon= 0.760000 . Estimated number of clusters: 2 . Estimated number of noise points: 370 / 20000 ( 0.02 )
Epsilon= 0.810000 . Estimated number of clusters: 2 . Estimated number of noise points: 325 / 20000 ( 0.02 )
Epsilon= 0.860000 . Estimated number of clusters: 2 . Estimated number of noise points: 296 / 20000 ( 0.01 )
Epsilon= 0.910000 . Estimated number of clusters: 2 . Estimated number of noise points: 282 / 20000 ( 0.01 )
Epsilon= 0.960000 . Estimated number of clusters: 2 . Estimated number of noise points: 259 / 20000 ( 0.01 )
Epsilon= 1.010000 . Estimated number of clusters: 2 . Estimated number of noise points: 247 / 20000 ( 0.01 )
Epsilon= 1.060000 . Estimated number of clusters: 2 . Estimated number of noise points: 236 / 20000 ( 0.01 )
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 4
['orders', 'payment_value', 'recency_crescent']
Jeu < 3000 BRL

```
Epsilon= 0.360000 . Estimated number of clusters: 1 . Estimated number of noise points: 1478 / 20000 ( 0.07 )
Epsilon= 0.410000 . Estimated number of clusters: 1 . Estimated number of noise points: 1363 / 20000 ( 0.07 )
Epsilon= 0.460000 . Estimated number of clusters: 1 . Estimated number of noise points: 1247 / 20000 ( 0.06 )
Epsilon= 0.510000 . Estimated number of clusters: 2 . Estimated number of noise points: 1006 / 20000 ( 0.05 )
Epsilon= 0.560000 . Estimated number of clusters: 2 . Estimated number of noise points: 763 / 20000 ( 0.04 )
Epsilon= 0.610000 . Estimated number of clusters: 2 . Estimated number of noise points: 621 / 20000 ( 0.03 )
Epsilon= 0.660000 . Estimated number of clusters: 2 . Estimated number of noise points: 491 / 20000 ( 0.02 )
Epsilon= 0.710000 . Estimated number of clusters: 2 . Estimated number of noise points: 409 / 20000 ( 0.02 )
Epsilon= 0.760000 . Estimated number of clusters: 2 . Estimated number of noise points: 370 / 20000 ( 0.02 )
Epsilon= 0.810000 . Estimated number of clusters: 2 . Estimated number of noise points: 325 / 20000 ( 0.02 )
Epsilon= 0.860000 . Estimated number of clusters: 2 . Estimated number of noise points: 296 / 20000 ( 0.01 )
Epsilon= 0.910000 . Estimated number of clusters: 2 . Estimated number of noise points: 282 / 20000 ( 0.01 )
Epsilon= 0.960000 . Estimated number of clusters: 2 . Estimated number of noise points: 259 / 20000 ( 0.01 )
Epsilon= 1.010000 . Estimated number of clusters: 2 . Estimated number of noise points: 247 / 20000 ( 0.01 )
Epsilon= 1.060000 . Estimated number of clusters: 2 . Estimated number of noise points: 236 / 20000 ( 0.01 )
```

[…]

Model 5 - [['orders', 'payment_value', 'review_score', 'rapidite_livraison', 'recency_crescent'],

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 5
['orders', 'payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent’]
Jeu < 3000 BRL

```
Epsilon= 0.010000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.060000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.110000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 7 . Estimated number of noise points: 18160 / 20000 ( 0.91 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 14163 / 20000 ( 0.71 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 11727 / 20000 ( 0.59 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 9167 / 20000 ( 0.46 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 7305 / 20000 ( 0.37 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5984 / 20000 ( 0.30 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 4822 / 20000 ( 0.24 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 4040 / 20000 ( 0.20 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 3438 / 20000 ( 0.17 )
Epsilon= 0.660000 . Estimated number of clusters: 4 . Estimated number of noise points: 3042 / 20000 ( 0.15 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 2578 / 20000 ( 0.13 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 2225 / 20000 ( 0.11 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1847 / 20000 ( 0.09 )
```

2. Algorithme DBSCAN sur échantillon de 20k entrées

Model 6
['payment_value', 'review_score', 'rapidite_livraison', ‘recency_crescent']
Jeu < 3000 BRL

```
Epsilon= 0.160000 . Estimated number of clusters: 0 . Estimated number of noise points: 19999 / 20000 ( 1.00 )
Epsilon= 0.210000 . Estimated number of clusters: 6 . Estimated number of noise points: 16633 / 20000 ( 0.83 )
Epsilon= 0.260000 . Estimated number of clusters: 1 . Estimated number of noise points: 13080 / 20000 ( 0.65 )
Epsilon= 0.310000 . Estimated number of clusters: 2 . Estimated number of noise points: 10595 / 20000 ( 0.53 )
Epsilon= 0.360000 . Estimated number of clusters: 3 . Estimated number of noise points: 8277 / 20000 ( 0.41 )
Epsilon= 0.410000 . Estimated number of clusters: 4 . Estimated number of noise points: 6398 / 20000 ( 0.32 )
Epsilon= 0.460000 . Estimated number of clusters: 4 . Estimated number of noise points: 5397 / 20000 ( 0.27 )
Epsilon= 0.510000 . Estimated number of clusters: 4 . Estimated number of noise points: 4306 / 20000 ( 0.22 )
Epsilon= 0.560000 . Estimated number of clusters: 4 . Estimated number of noise points: 3707 / 20000 ( 0.19 )
Epsilon= 0.610000 . Estimated number of clusters: 4 . Estimated number of noise points: 3095 / 20000 ( 0.15 )
Epsilon= 0.660000 . Estimated number of clusters: 4 . Estimated number of noise points: 2678 / 20000 ( 0.13 )
Epsilon= 0.710000 . Estimated number of clusters: 5 . Estimated number of noise points: 2166 / 20000 ( 0.11 )
Epsilon= 0.760000 . Estimated number of clusters: 1 . Estimated number of noise points: 1780 / 20000 ( 0.09 )
Epsilon= 0.810000 . Estimated number of clusters: 1 . Estimated number of noise points: 1394 / 20000 ( 0.07 )
Epsilon= 0.860000 . Estimated number of clusters: 1 . Estimated number of noise points: 1130 / 20000 ( 0.06 )
```

order_id

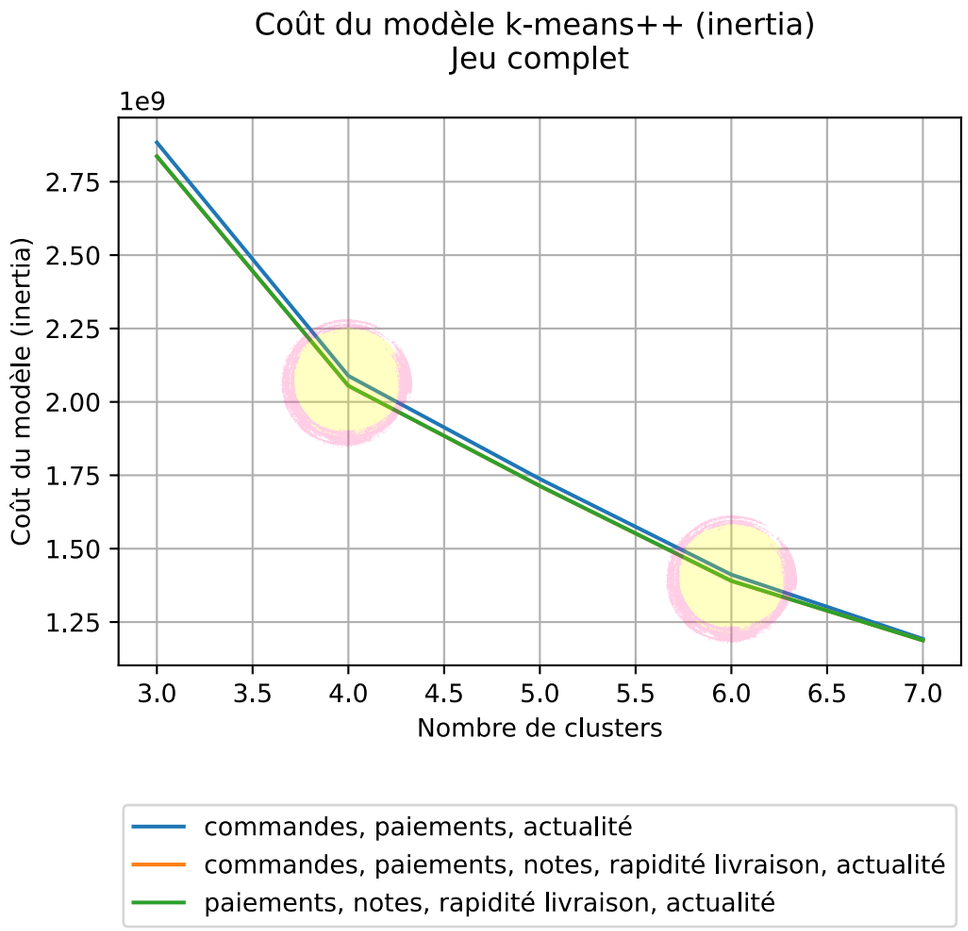
On retient seulement le premier order_id

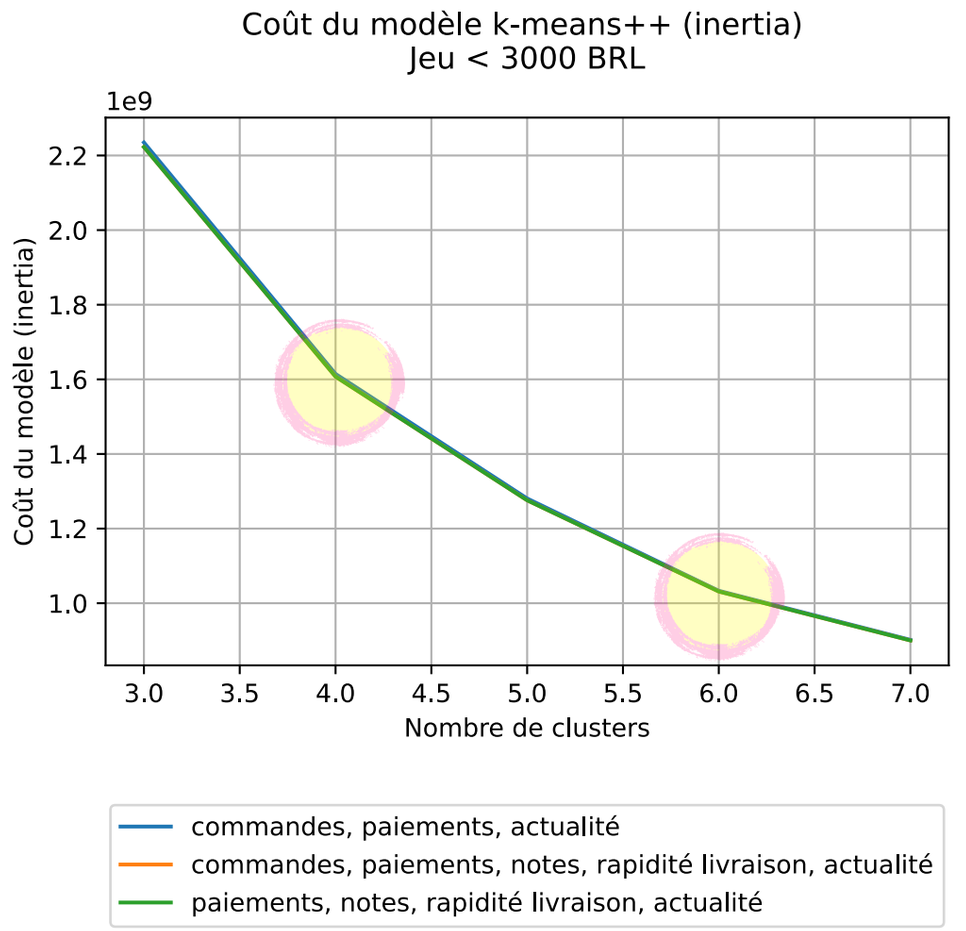
	customer_id	customer_unique_id	order_id	order_status	order_purchase_timestamp
6	4d3abb73ceb86353aeadbe698aa9d5cb	f736308cd9952b33b90b9fe94da9c8f5	fffb9224b6fc7c43ebb0904318b10b5f	delivered	2017-10-27
7	4d3abb73ceb86353aeadbe698aa9d5cb	f736308cd9952b33b90b9fe94da9c8f5	fffb9224b6fc7c43ebb0904318b10b5f	delivered	2017-10-27
8	4d3abb73ceb86353aeadbe698aa9d5cb	f736308cd9952b33b90b9fe94da9c8f5	fffb9224b6fc7c43ebb0904318b10b5f	delivered	2017-10-27
9	4d3abb73ceb86353aeadbe698aa9d5cb	f736308cd9952b33b90b9fe94da9c8f5	fffb9224b6fc7c43ebb0904318b10b5f	delivered	2017-10-27

order_delivered_carrier_date	order_estimated_delivery_date	order_item_id	product_id	price	freight_value	review_score	payment_sequential	payment_type	payment_installments	payment_value	dup
2017-11-10 19:31:52	2017-11-27 00:00:00	4	43423cdffde7fda63d0414ed38c11a73	55.0	34.19	4	1	boleto	1	356.76	True
2017-11-10 19:31:52	2017-11-27 00:00:00	3	43423cdffde7fda63d0414ed38c11a73	55.0	34.19	4	1	boleto	1	356.76	True
2017-11-10 19:31:52	2017-11-27 00:00:00	2	43423cdffde7fda63d0414ed38c11a73	55.0	34.19	4	1	boleto	1	356.76	True
2017-11-10 19:31:52	2017-11-27 00:00:00	1	43423cdffde7fda63d0414ed38c11a73	55.0	34.19	4	1	boleto	1	356.76	True

3. Algorithme k-means++

Méthode elbow





La méthode elbow identifie le nombre optimal de clusters entre 4 et 6

2. Mon indication

```
Rating: ACDDB (ratio f/n: 1.0)
Rating: DDCDA (ratio f/n: 1.0)
Rating: ADCBA (ratio f/n: 1.0)
Rating: CDAAD (ratio f/n: 1.0)
Rating: DDDDA (ratio f/n: 1.0)
Rating: AAACA (ratio f/n: 1.0)
Rating: BDCDA (ratio f/n: 1.0)
```

	orders	payment value	recency crescent	review score	rapidite livraison
Cluster	nombre de commande du client sur la période considérée	dépenses totales du client	mesure croissante de l'actualité du dernier achat	note du dernier achat	mesure croissante du décalage entre la date de livraison prévue et la date d'arrivé effective
0	A	C	D	D	B
1	D	D	C	D	A
2	A	D	C	B	A
3	C	D	A	A	D
4	D	D	D	D	A
5	A	A	A	C	A
6	B	D	C	D	A