

## Projet 4

Segmentez des clients d'un site  
e-commerce

# Loridan Adrien



student machine learning  
engineer



[medium.com/@adrien.loridan](https://medium.com/@adrien.loridan)



[linkedin.com/in/adrien-loridan](https://linkedin.com/in/adrien-loridan)



[github.com/loridan](https://github.com/loridan)

olist

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

olist

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Présentation

Olist propose une solution de vente sur les marketplaces en ligne pour les commerçants. Elle a pour objectif de mieux communiquer auprès des acheteurs.

**olist**

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Présentation

Olist propose une solution de vente sur les marketplaces en ligne pour les commerçants. Elle a pour objectif de mieux communiquer auprès des acheteurs.

## Problème

Pouvons nous mieux connaître nos clients et mieux investir le budget dans nos actions marketing ?



Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

Une solution envisagée par Olist...

Olist souhaite obtenir une segmentation de ses clients à l'aide de méthodes non supervisées sur ses propres données.

olist

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

avec une demande spécifique :

- décrire les clusters pour l'équipe marketing
- évaluer la fréquence de mise à jour de la segmentation
- respecter la convention PEP8

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Critères de segmentation

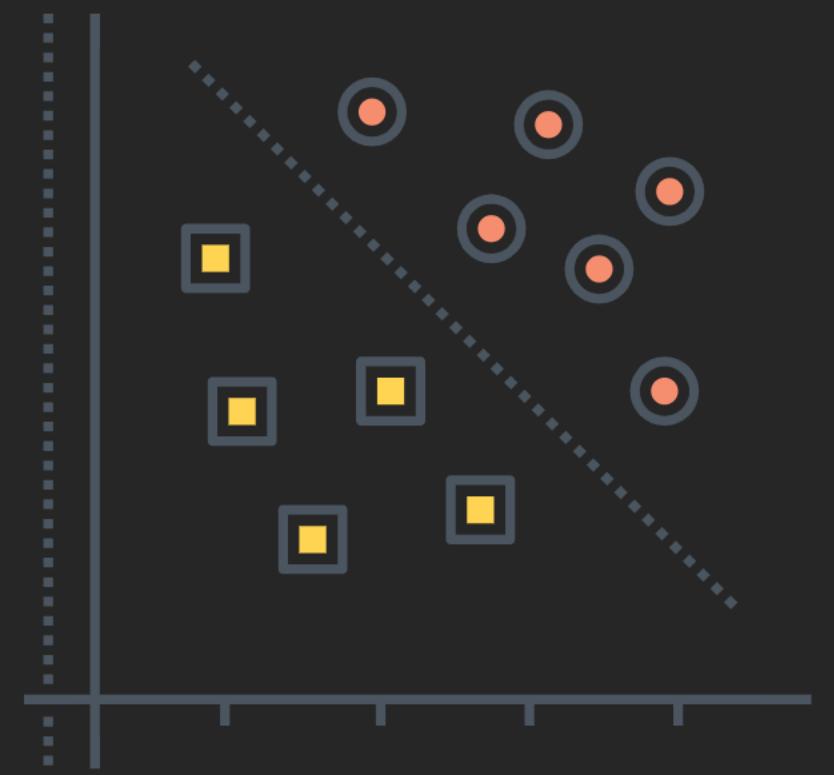
Segmenter, c'est donc créer des groupes composés de clients ayant des points communs identifiables :

- géographiques (géolocalisation, météo)
- démographiques (genre, age, nationalité)
- comportementales (nombre de visites, historique d'achat)

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Critères de segmentation

- rule-based segmentation
- cluster-based segmentation (*envisagée par Olist*)



Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Segmenter sur 3 dimensions "RFM"

On extrait des données ces 3 critères pour chacun des clients :

- Récence : *date du dernier achat*
- Fréquence : *fréquence des achats*
- Montant : *somme des achats cumulés*

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Segmenter sur 3 dimensions "RFM"

RFM is a method used for analyzing customer value.  
It is commonly used direct marketing and has received particular attention in retail and professional services industries.

Technique simple et compréhensible pour définir un segment-cible prioritaire par les gens du business

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## Segmenter sur N dimensions

On extrait des données un maximum de critères pour chacun des clients.

Méthode compliquée et moins compréhensible mais permettant de découvrir d'autres segments profitables.

Observer pour connaître, connaître pour comprendre,  
**comprendre pour agir.**

## PEP8

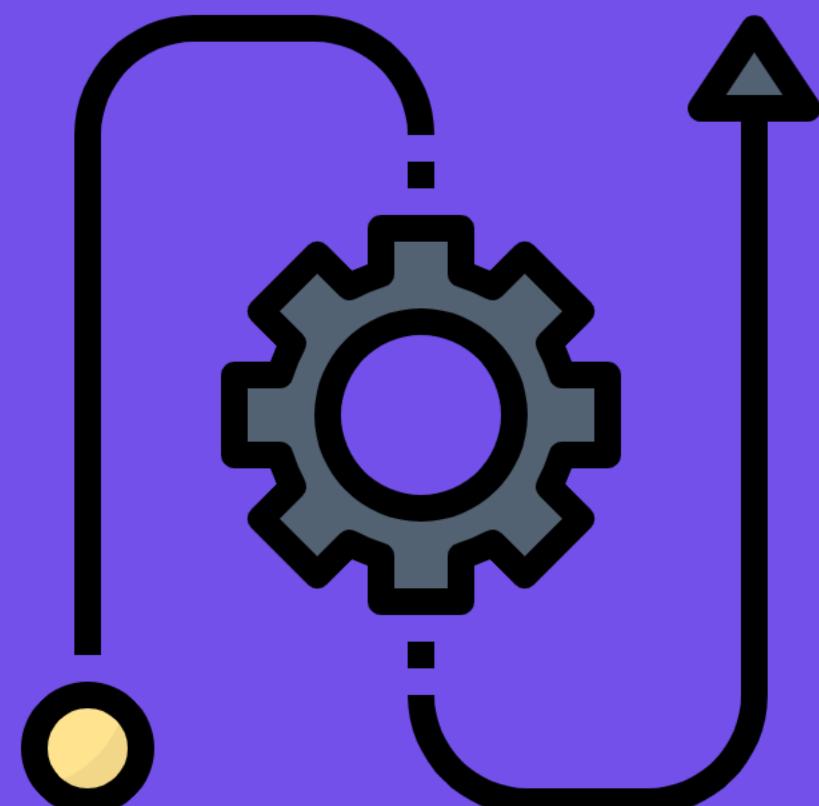
La PEP 8 définit des règles de développement communes entre développeurs.

Nous allons utiliser *Black*, pour analyser et corriger le code directement dans le jupyter-lab avec une extension appropriée.



La science des données

# Expérimentation



- 1 Collecter des données
- 2 Préparer et explorer les données
- 3 Segmenter
- 4 Comprendre

# kaggle

**Lien de téléchargement**

[kaggle.com/olistbr/brazilian-e-commerce](https://kaggle.com/olistbr/brazilian-e-commerce)

**9 fichiers**

- olist\_customers\_dataset.csv (8.62 MB)
- olist\_geolocation\_dataset.csv (58.44 MB)
- olist\_order\_items\_dataset.csv (74.72 MB)
- olist\_order\_payments\_dataset.csv (5.51 MB)
- olist\_order\_reviews\_dataset.csv (13.74 MB)
- olist\_orders\_dataset.csv (16.84 MB)
- olist\_products\_dataset.csv (2.27 MB)
- olist\_sellers\_dataset.csv (170.61 KB)
- product\_category\_name\_translation.csv (2.55 KB)

# olist

kaggle



**Quelle est la réglementation sur la protection de ces données ?**

Faible, contenu anonymisé sous licence NonCommercial (CC BY-NC-SA 4.0)

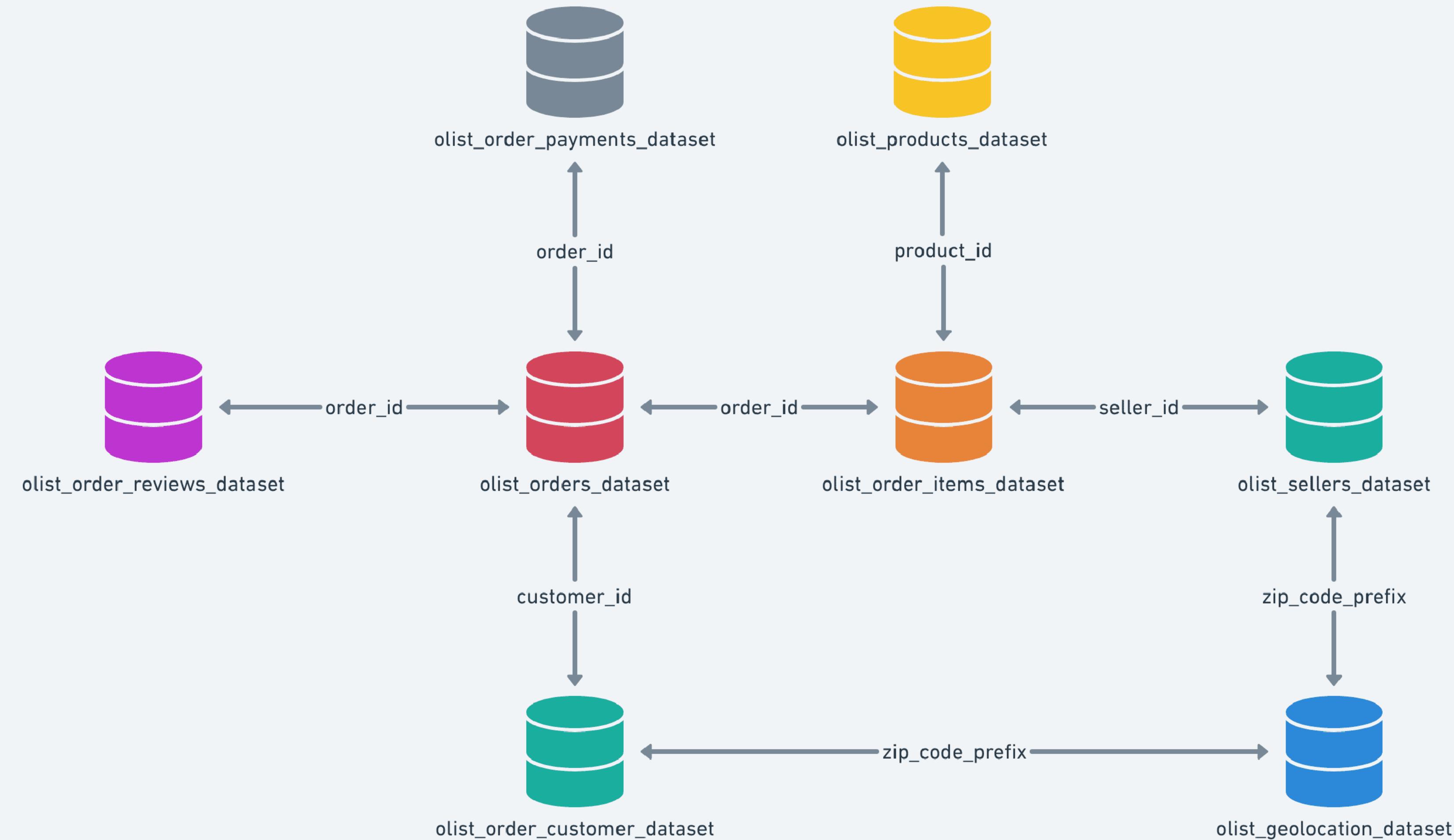
**Où sont stockées les données?**

Elles sont sur les serveurs distants publiques et sur notre ordinateur en sdd local

**Quel est le type des données**

Structurée

olist



# olist



## Nettoyage

Importer et observer les données

Identifier et traiter les doublons

Identifier et traiter les valeurs manquantes ('*review\_comment\_title*'...)

Convertir les timestamps ('*order\_delivered\_customer\_date*'...)

Analyser les données avant assemblage



## **Assemblage pour segmentations**

Merge

Feature engineering\*

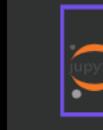
Visualisation des valeurs quantitatives

Export ("df\_rfm.csv" et "df\_customers.csv" )

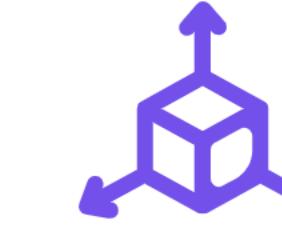


## Feature engineering pour segmentation $N=3$ dimensions - RFM

- Nombre de jours depuis dernier achat / Récence
- Nombre d'ordres / Fréquence
- Somme des paiements / Montant



## Feature engineering pour segmentation **N=105 dimensions**



- Nombre d'ordres / Fréquence
- Statistiques sur paiements ( min, max, mean, sum)
- Ratio pour les différents moyens de paiement
- Statistiques produits achetés par ordre (mean)
- Nombre de comptes clients
- Statistiques sur notes laissées ( min, max, mean )
- Ratio d'ordres pour chaque categories
- Ratio d'ordres pour differents types de produits
- Montant total dépensé par client pour differents types de produits ( 1-4)
- Statistiques sur délais de livraison ( min, max, mean, sum )
- Ratio par buckets sur délais de livraison en jours ( (0-3j] (3-7j] (7-14j] (14-maxj] )
- Nombre de jours depuis dernier achat / Récence

## Feature engineering : data augmentation

```
Entrée [53]: mapper_categories_to_four_types = {  
    'health_beauty' : 'convenience',  
    'computers_accessories' : 'shopping',  
    'auto' : 'shopping',  
    'bed_bath_table' : 'shopping',  
    'furniture_decor' : 'shopping',  
    'sports_leisure' : 'shopping',  
    'perfumery' : 'convenience',  
    'housewares' : 'shopping',  
    'telephony' : 'shopping',  
    'watches_gifts' : 'shopping',  
    'food_drink' : 'convenience',  
    'baby' : 'convenience'
```

Consumer products classification system



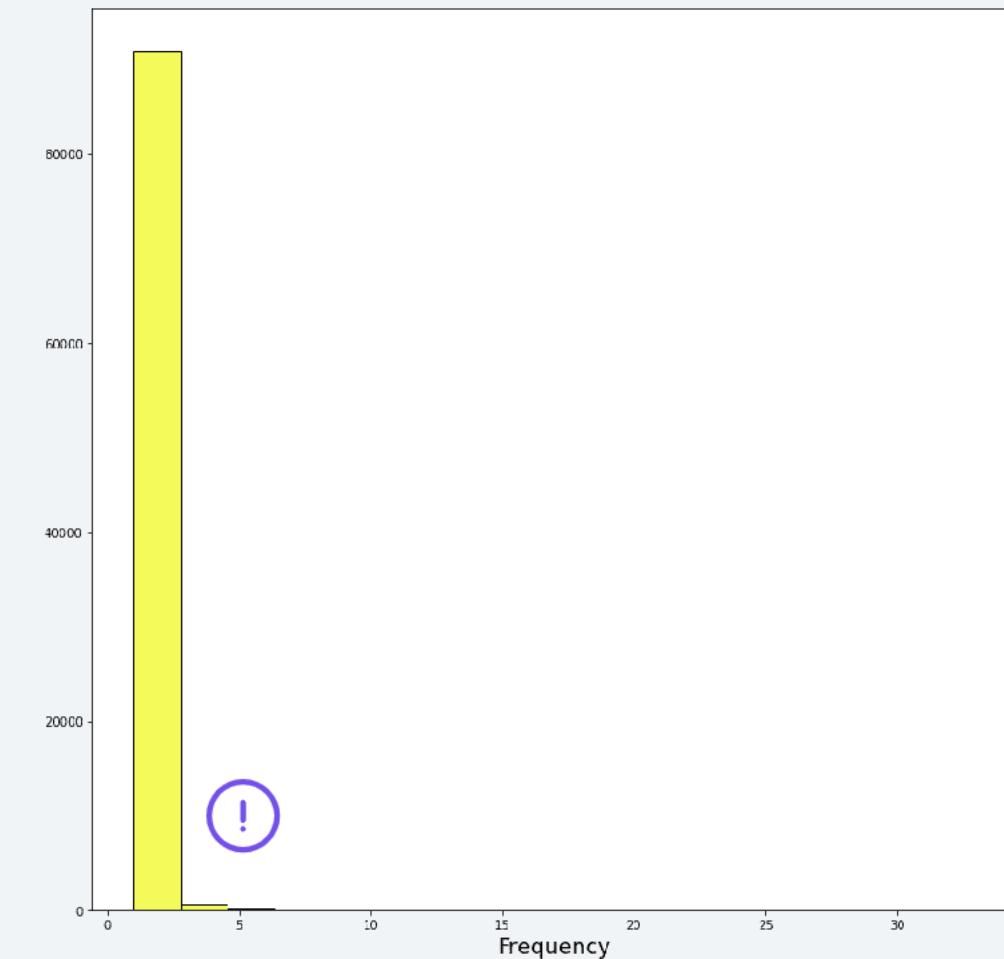
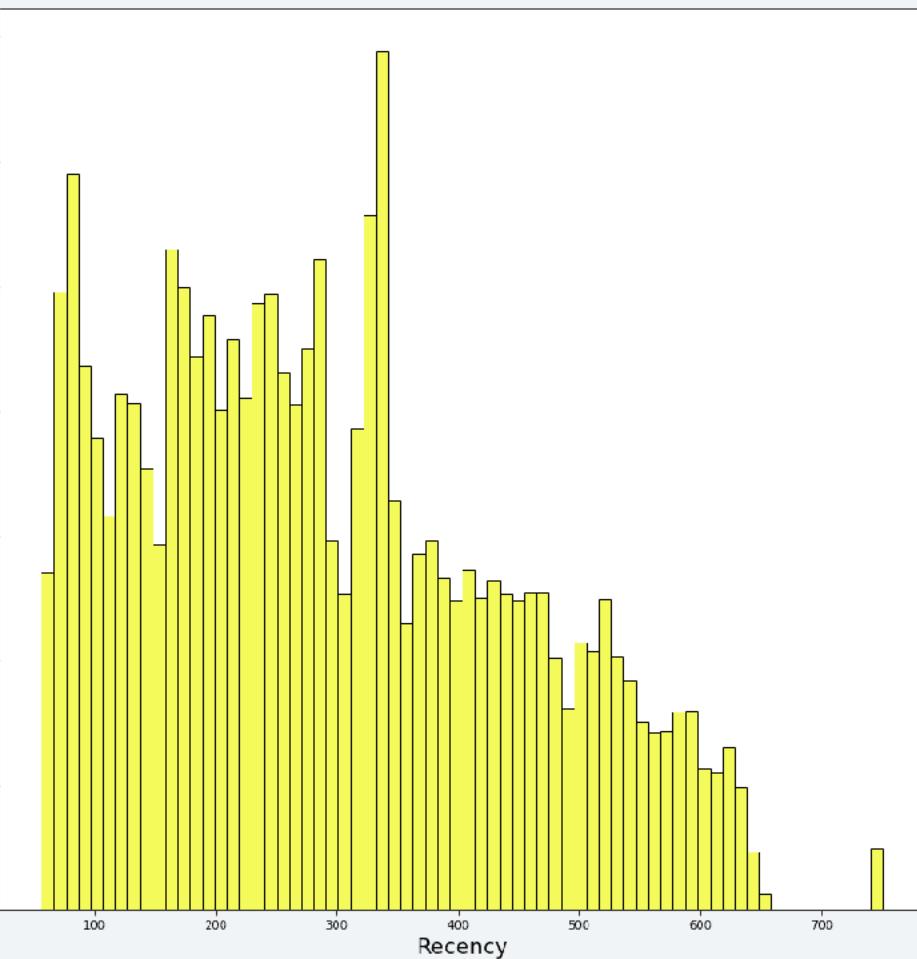
Cependant Olist ne vend pas de produits de type : 'Unsought products'

Journal of Marketing - "Retail Strategy and the Classification of Consumer Goods"  
(<https://www.jstor.org/stable/1248582?seq=1>)

## Distribution sur RFM

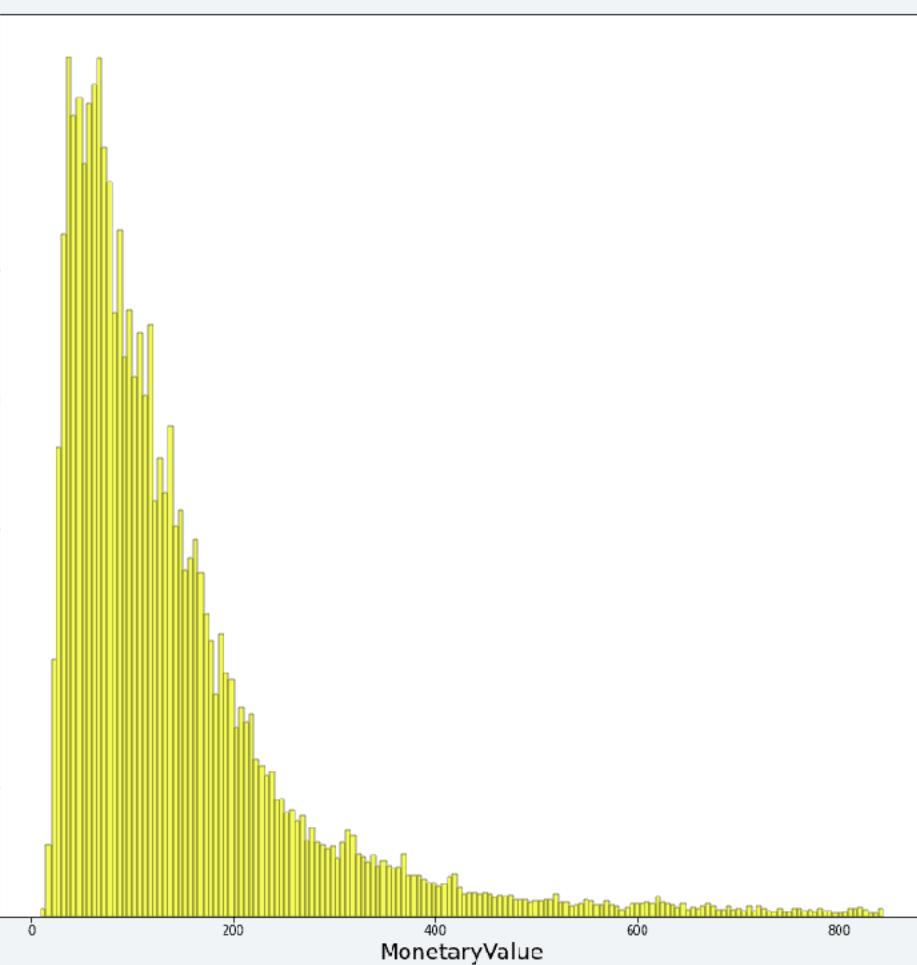
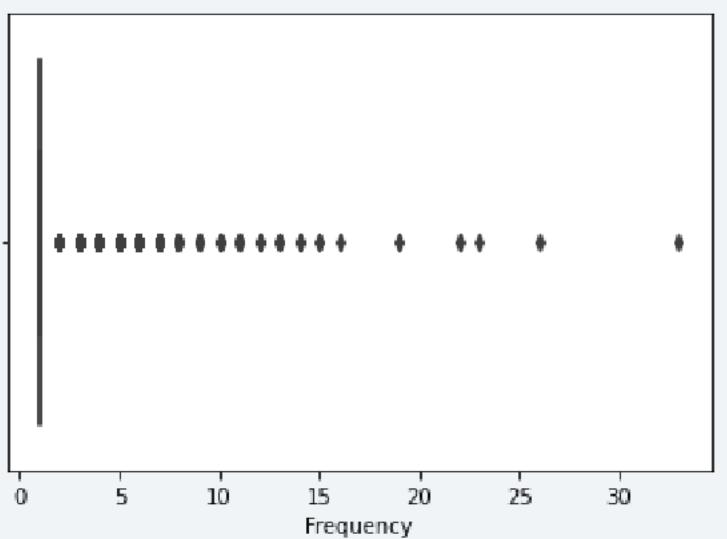


PO4\_01\_notebook\_exploration.ipynb



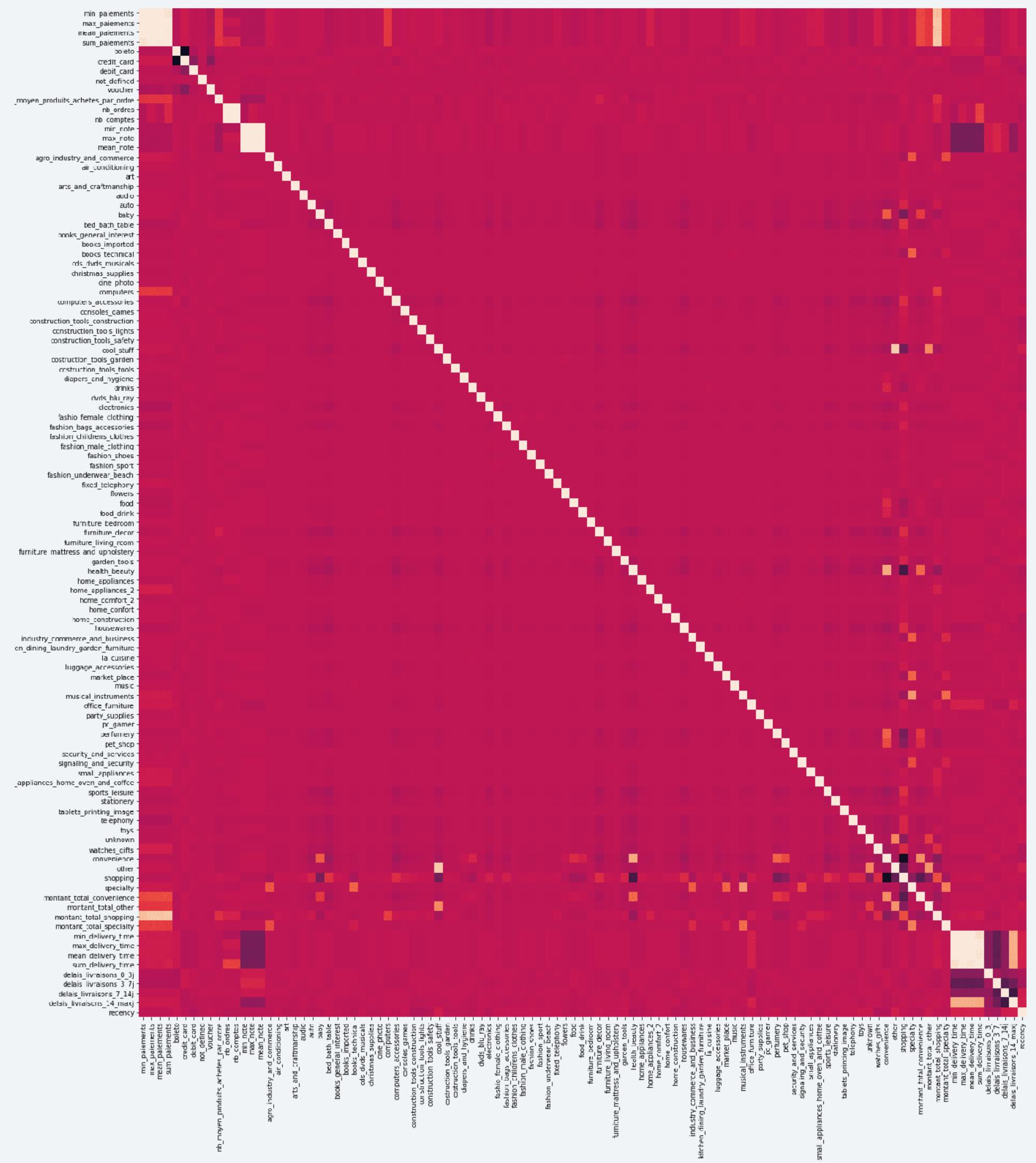
### Observations :

- ! Le nombre d'ordres est en moyenne de 1.079, avec 94% des clients qui ne passent qu'un seul ordre, le jeu de données est-il biaisé ?



olist

# Matrice de corrélation sur $N=105$ dimensions



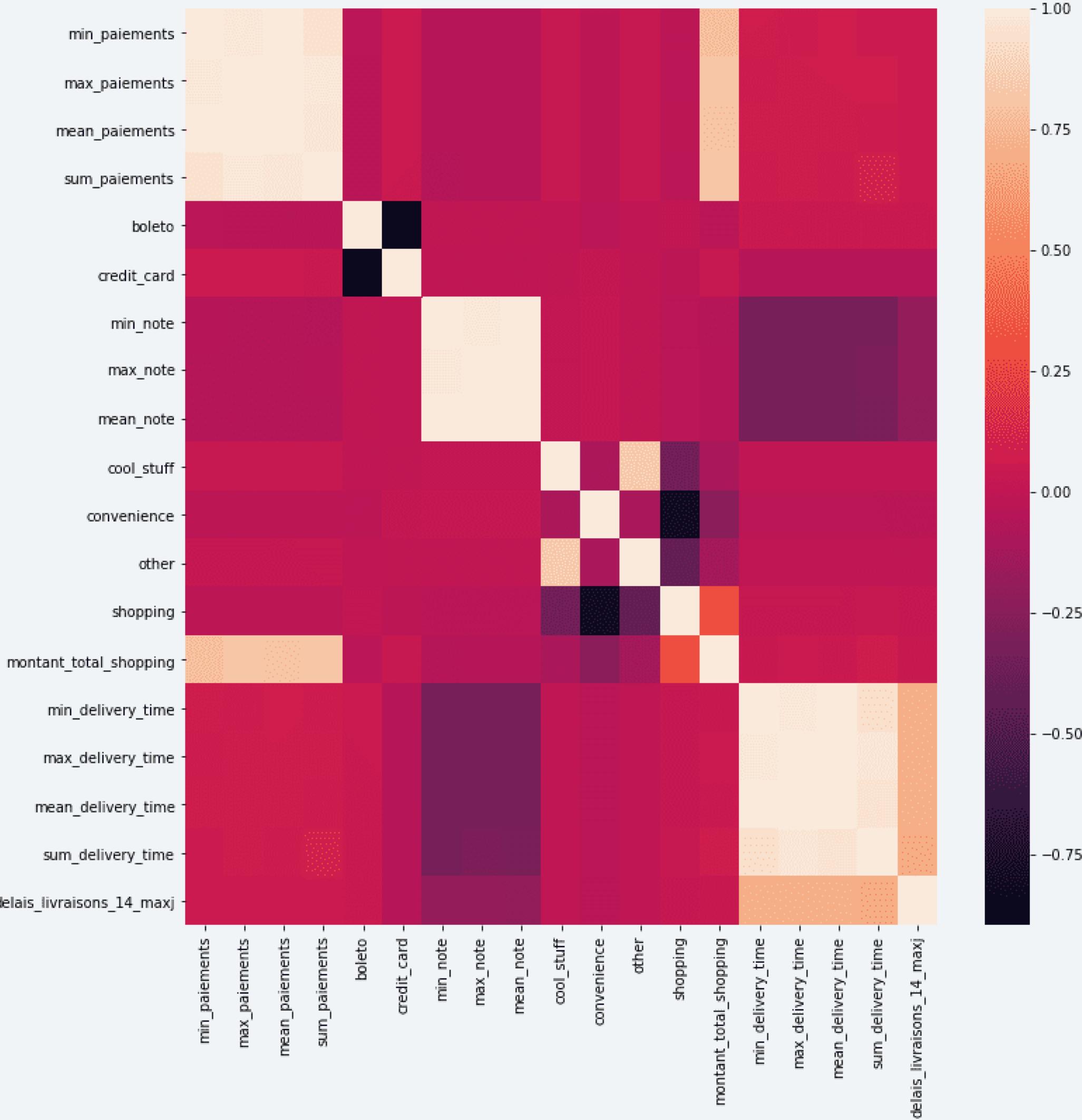
## Matrice de corrélation réduite > (+-)0.7



PO4\_01\_notebook\_exploration.ipynb

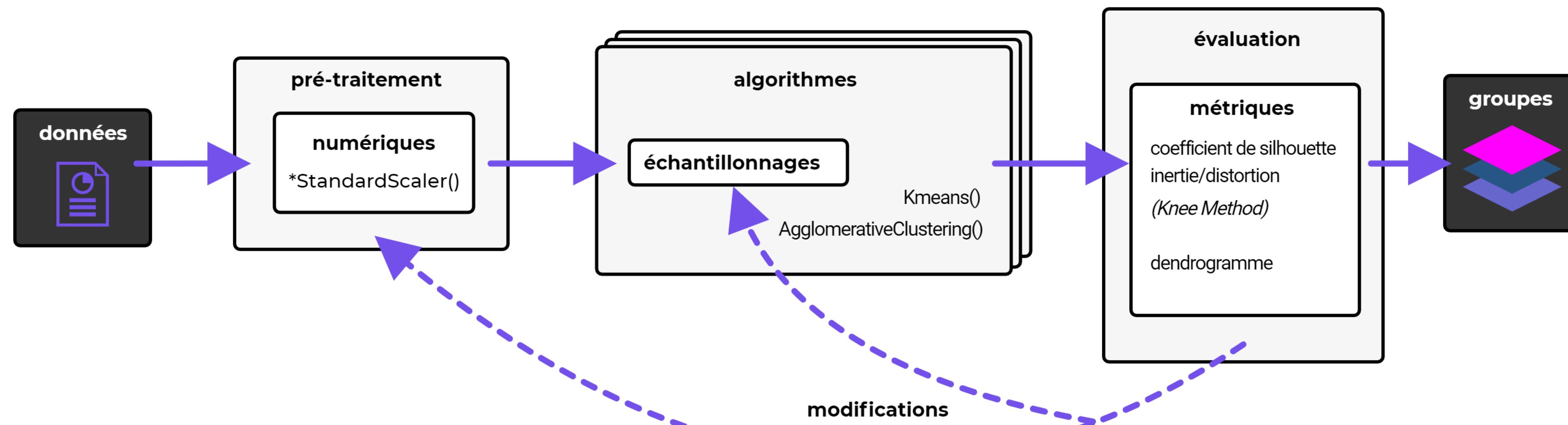
### Corrélations :

- ⊕ Délais de livraisons / Notes
- ⊕ Montant total des produits achetés de type 'shopping' / Montant total paiements
- ⊖ Paiement par billet / Paiement par CB





## Process de classification non supervisée (dév)





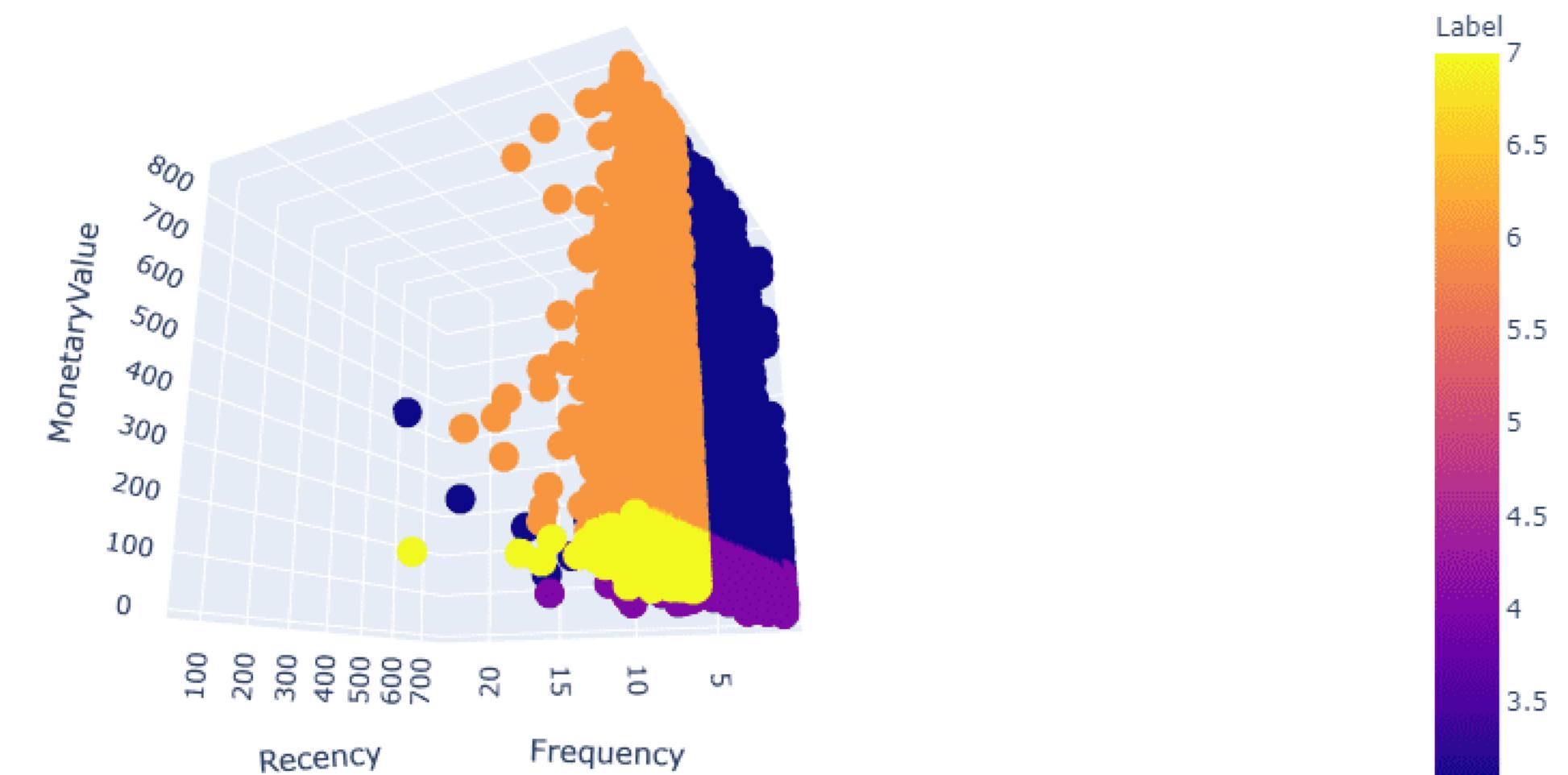
## Segmentation sur N=3 dimensions RFM

### Rules based Clustering

- sur un échantillon de 30%
- séparation en 8 classes arbitraires à priori

```
Entrée [7]: index_label_0 = df_rfmb.query('Recency < @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_1 = df_rfmb.query('Recency < @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_2 = df_rfmb.query('Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_3 = df_rfmb.query('Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_4 = df_rfmb.query('Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_5 = df_rfmb.query('Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_6 = df_rfmb.query('Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_7 = df_rfmb.query('Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue')

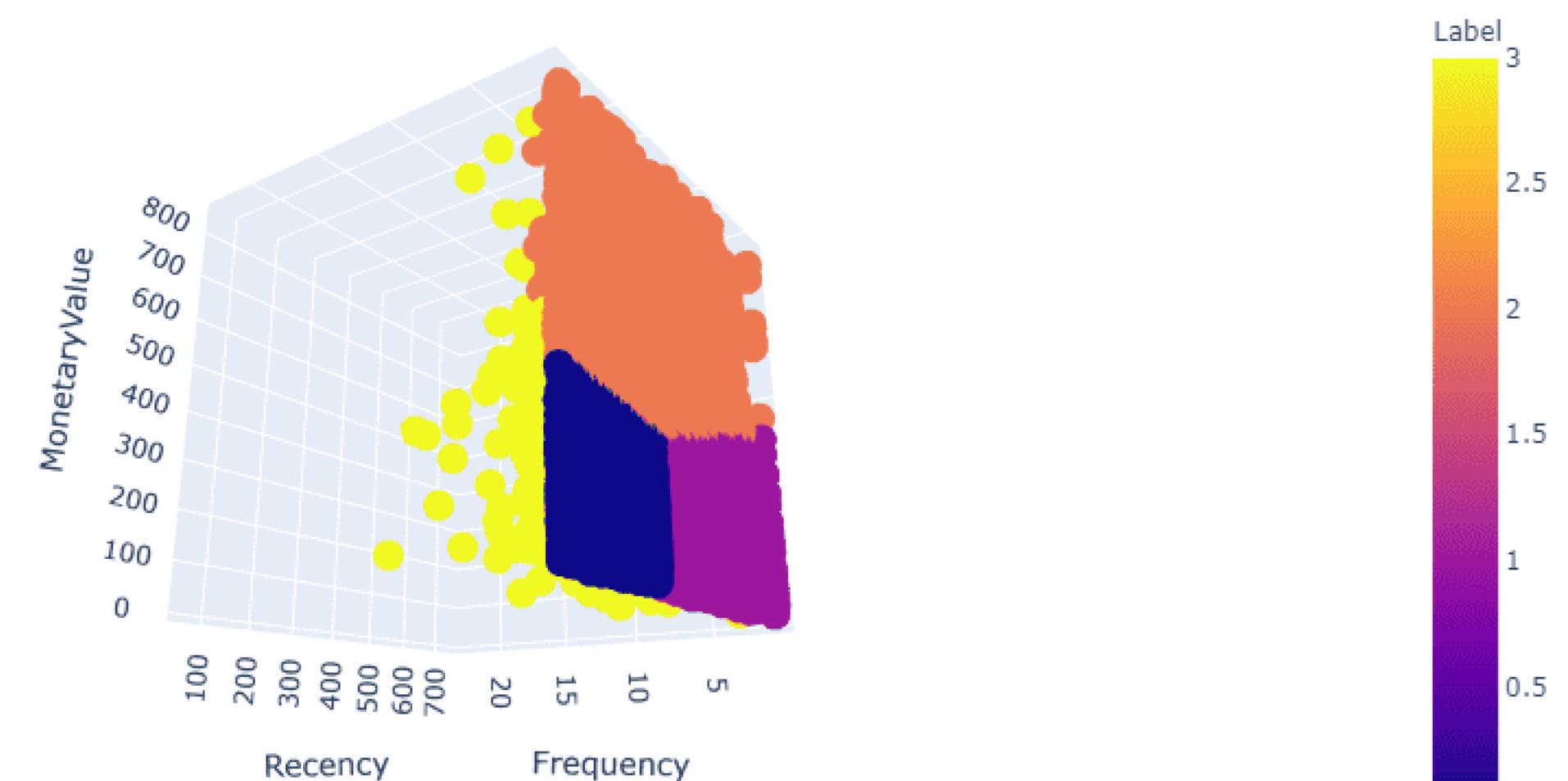
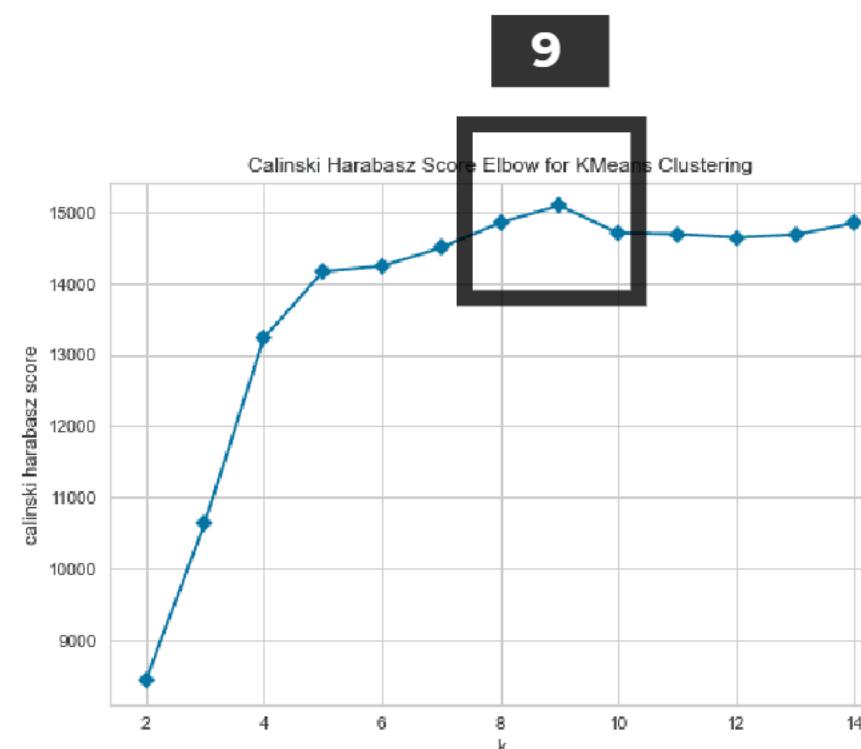
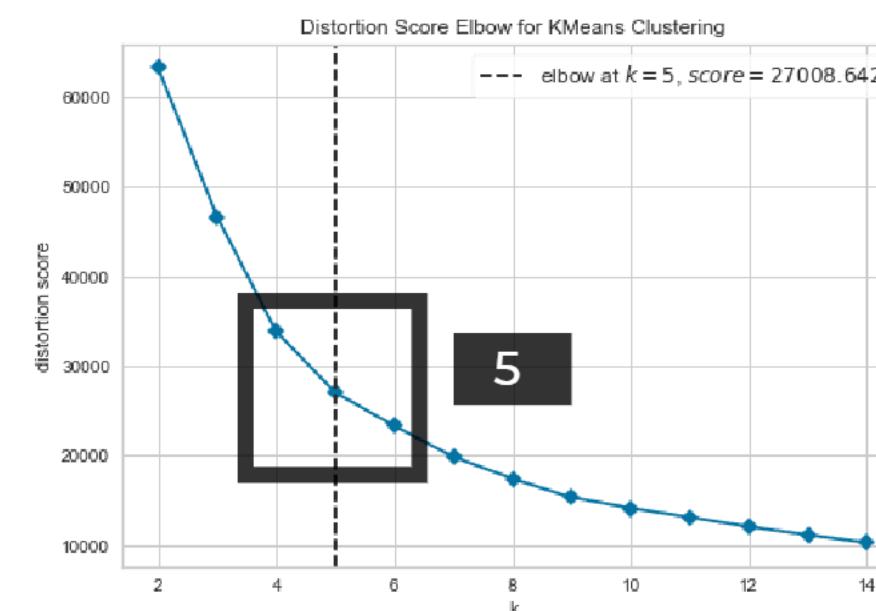
Entrée [8]: #Séparation en 8 classes arbitraires
df_rfmb.loc[index_label_0, ['Label']] = 0
df_rfmb.loc[index_label_1, ['Label']] = 1
df_rfmb.loc[index_label_2, ['Label']] = 2
df_rfmb.loc[index_label_3, ['Label']] = 3
df_rfmb.loc[index_label_4, ['Label']] = 4
df_rfmb.loc[index_label_5, ['Label']] = 5
df_rfmb.loc[index_label_6, ['Label']] = 6
df_rfmb.loc[index_label_7, ['Label']] = 7
```



## Segmentation sur $N=3$ dimensions RFM

### Kmeans Clustering

- ( K= 2 to 15 )
- sur un échantillon de 30%
- métriques divergentes - essai sur separation en 4 classes à posteriori





## Segmentation sur **N=3** dimensions RFM

On peut visuellement constater l'absence de clusters bien séparés.



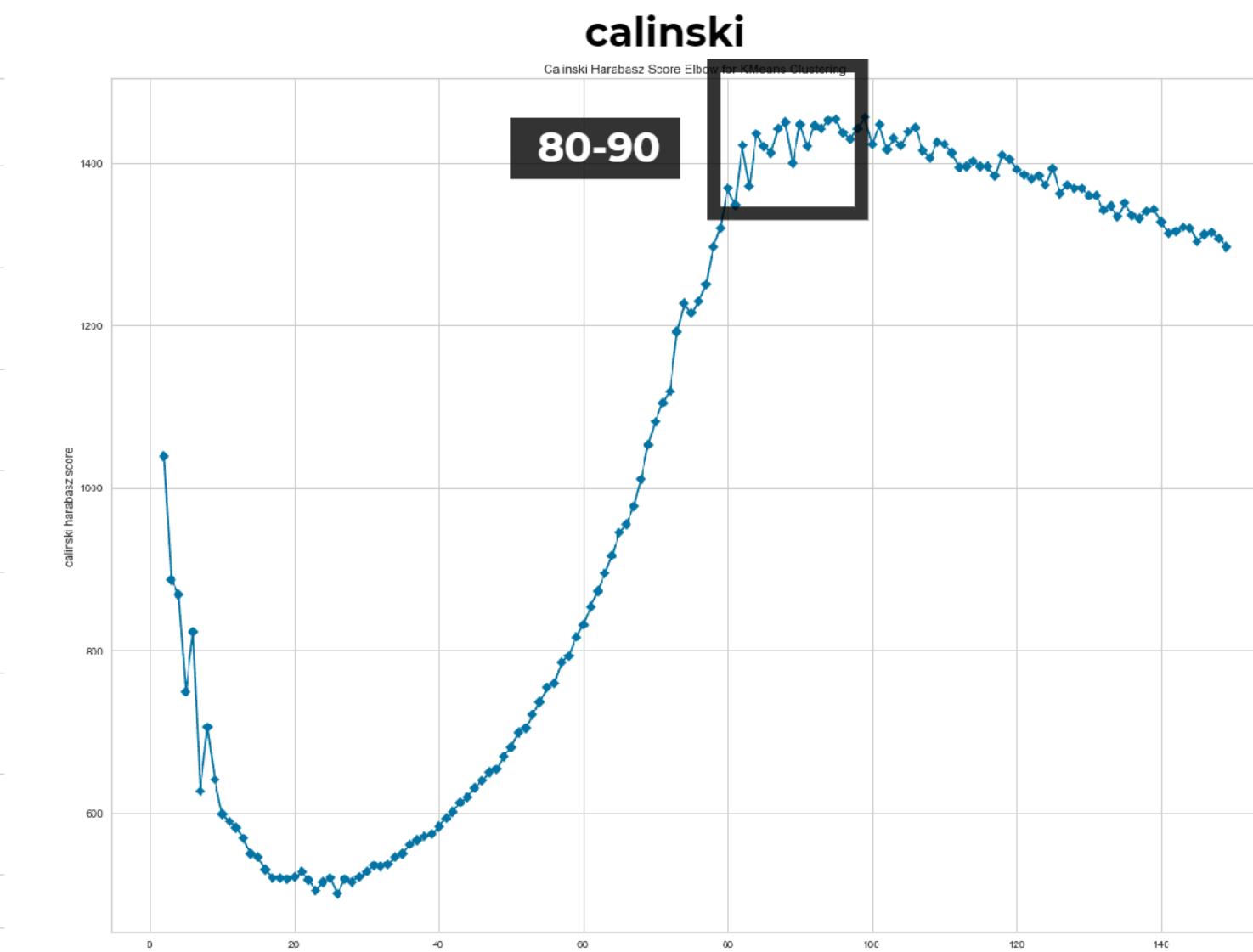
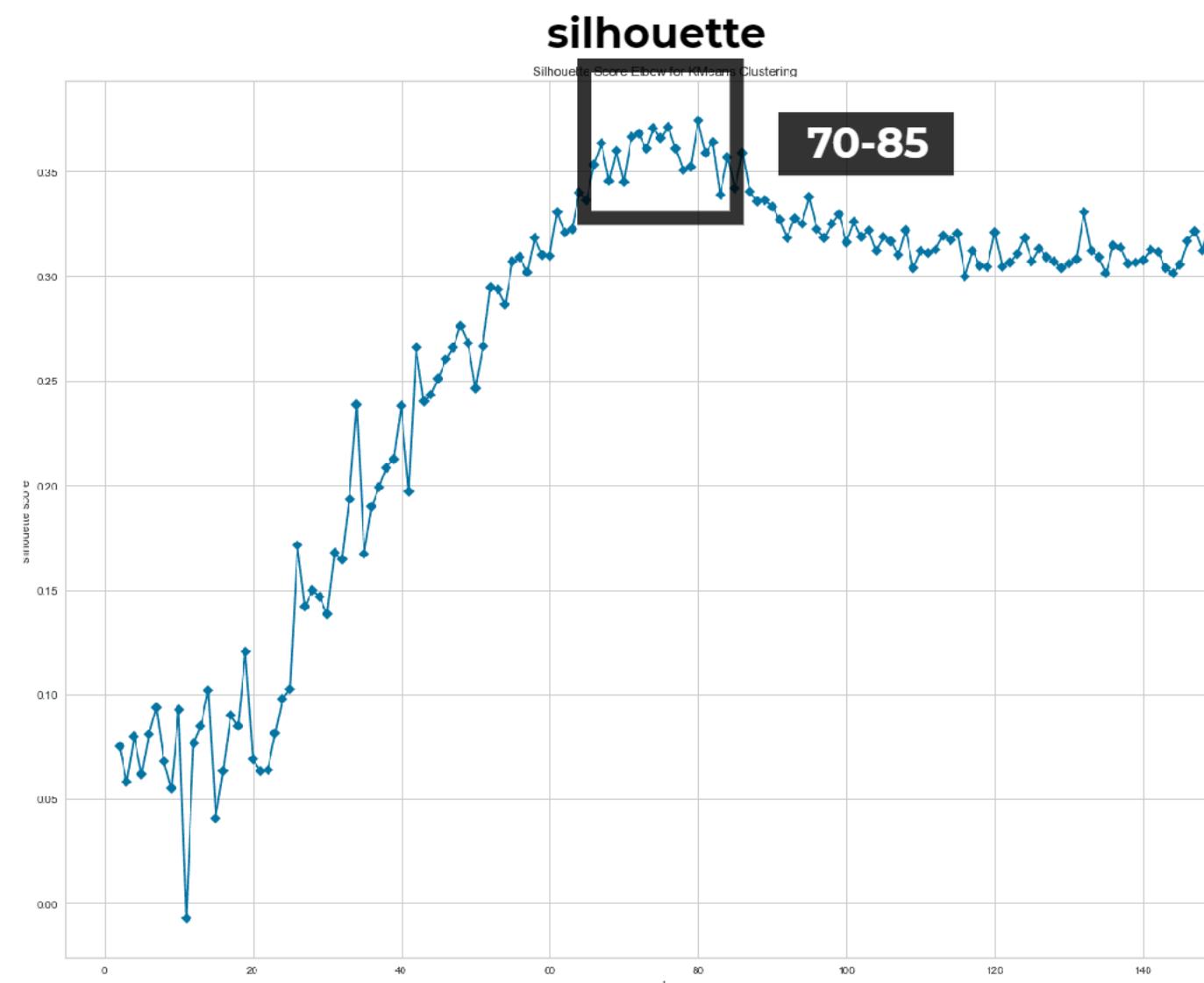
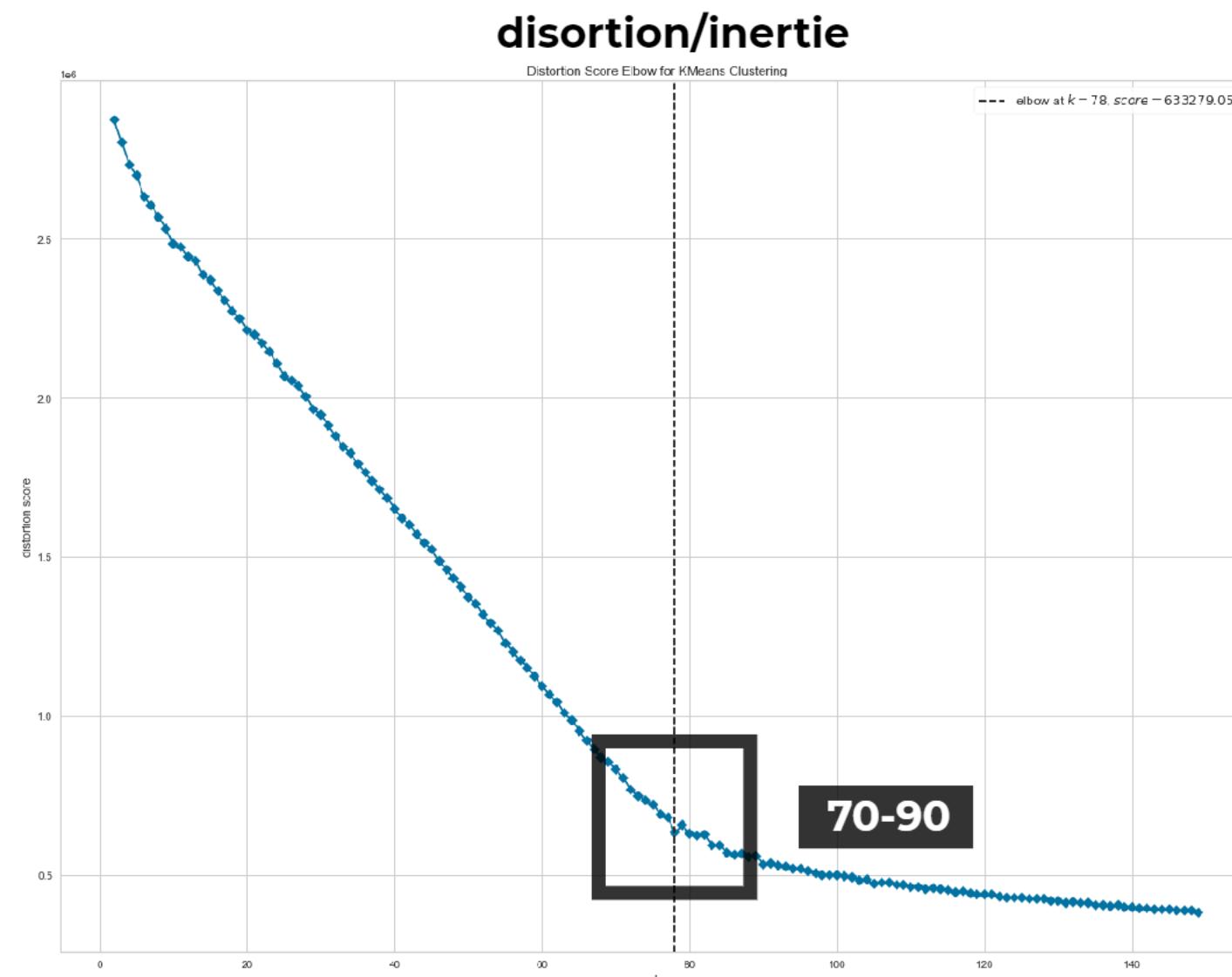
D'un point de vue marketing, une segmentation arbitraire peut cependant être utile si l'on veut cibler une partie de l'ensemble des clients.



## Segmentation sur $N=105$ dimensions

### Kmeans Clustering

- **K= 2 à 150**
- sur un échantillon de **30%**
- métriques convergentes pour une séparation de 75 à 90 classes

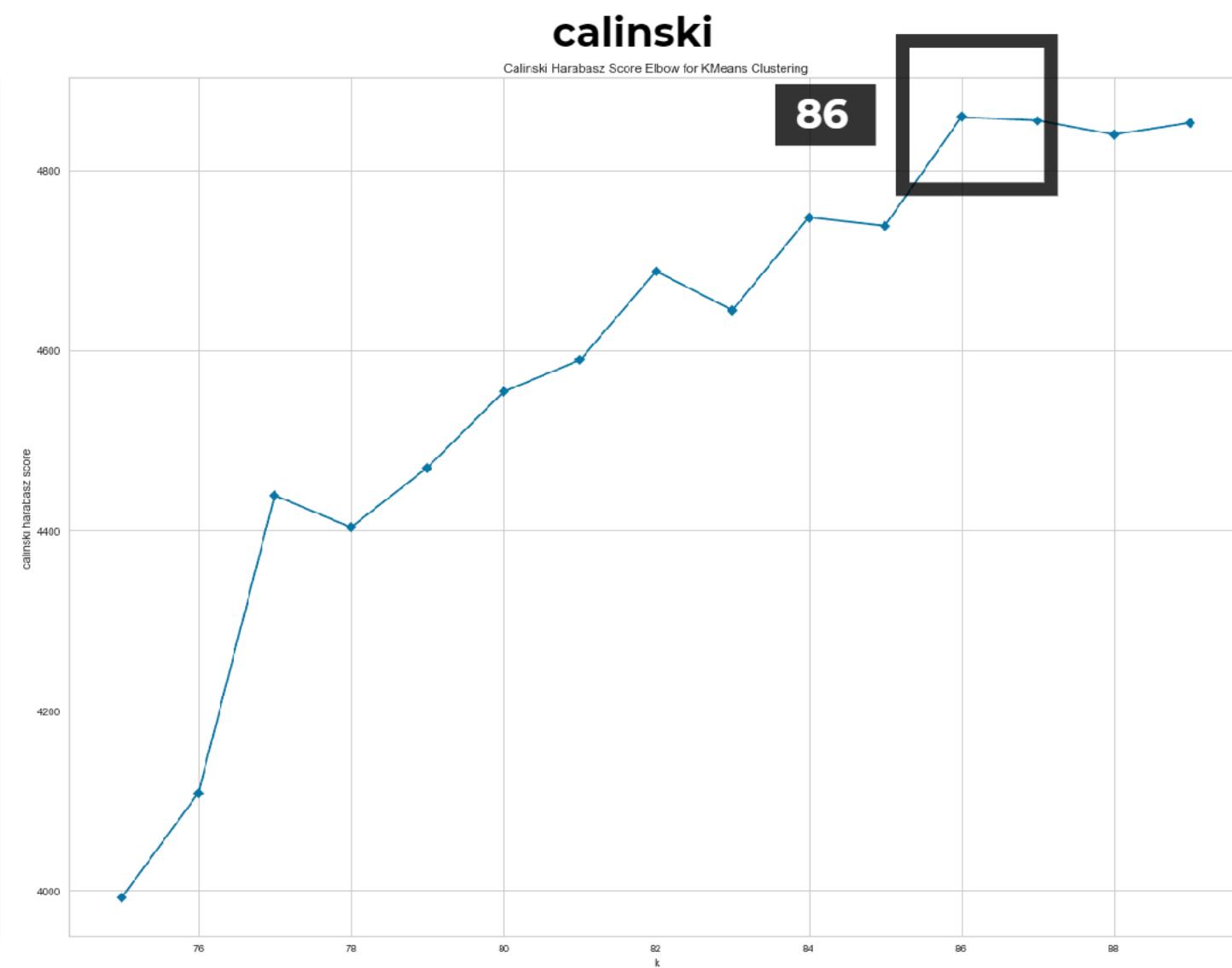
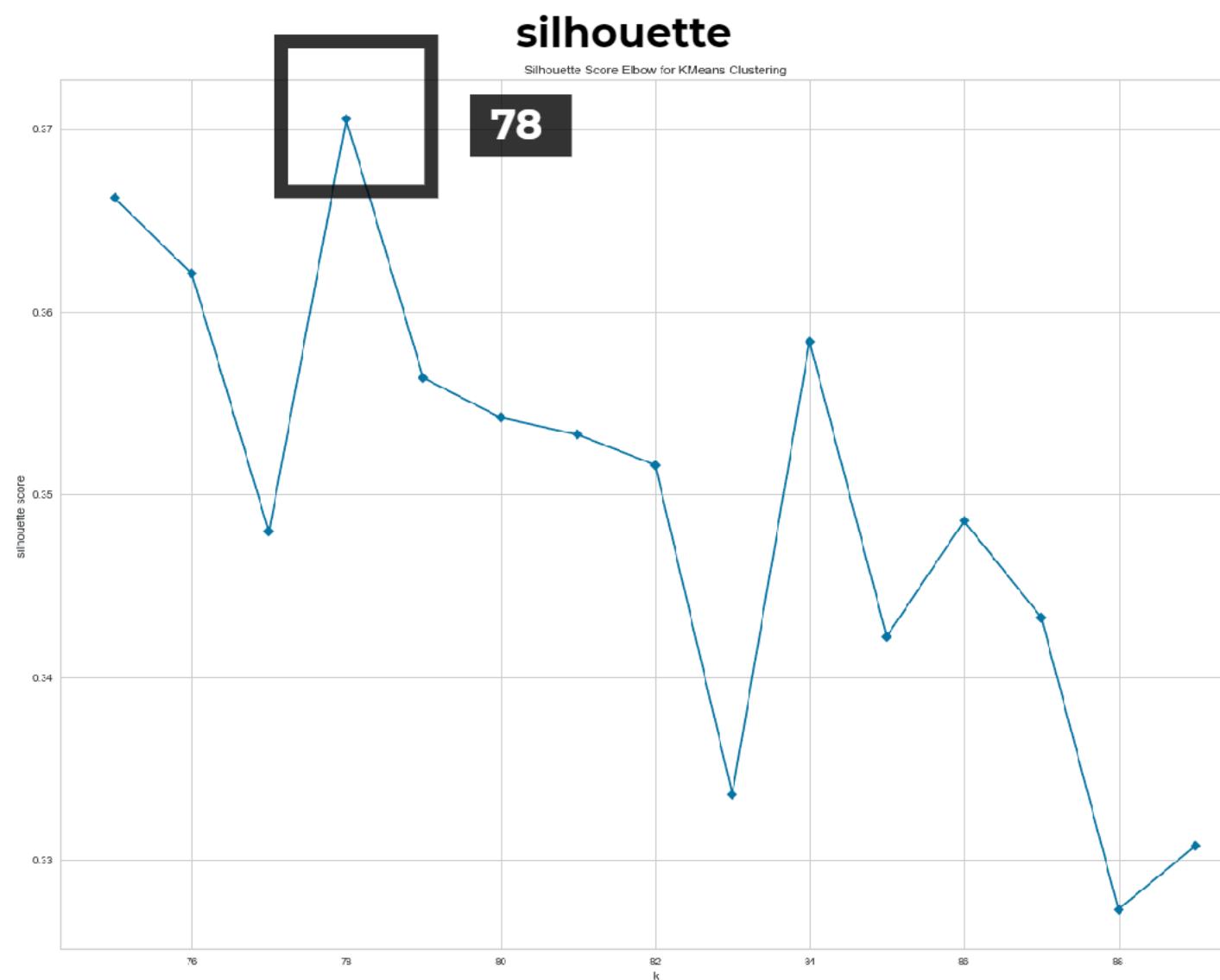
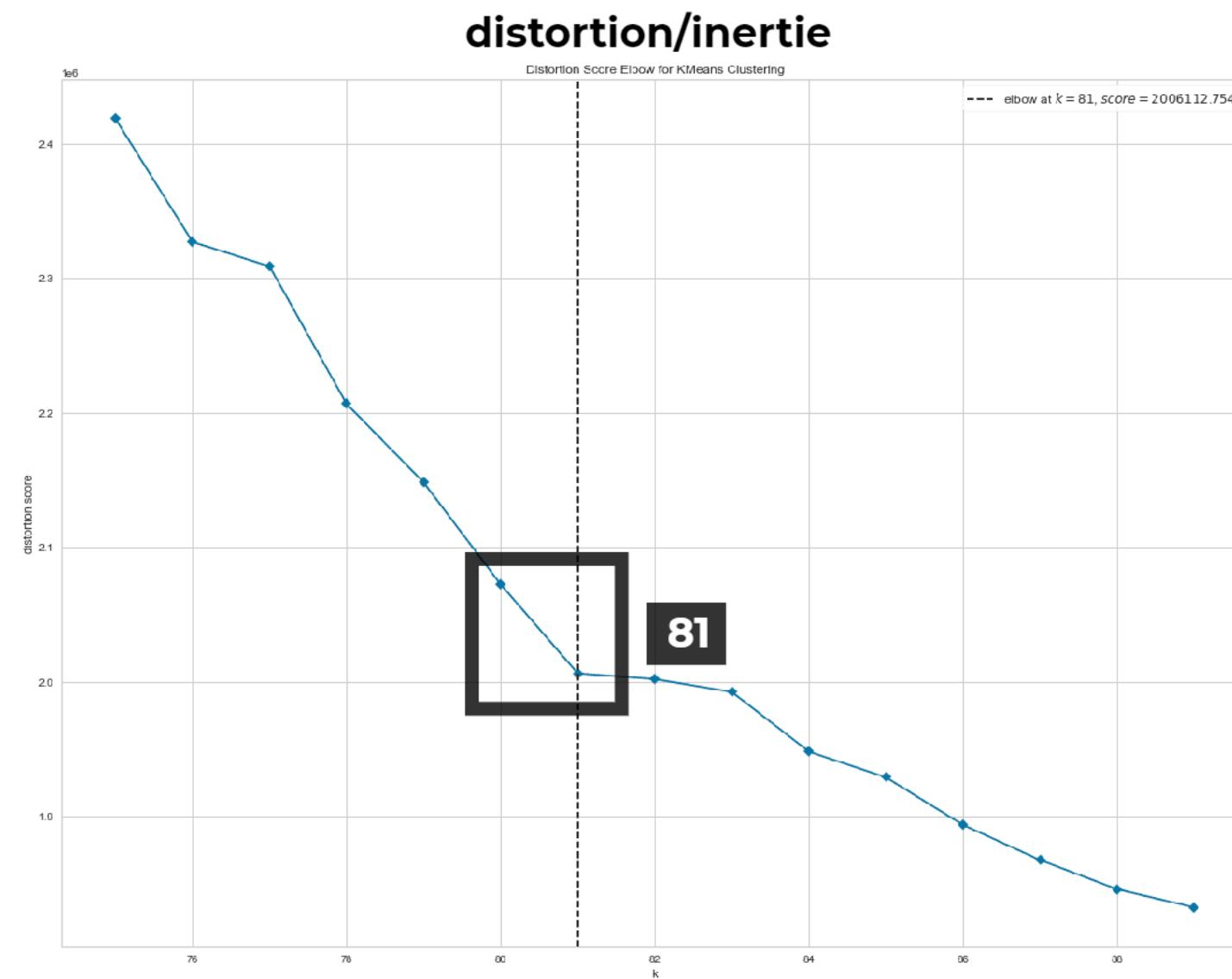




## Segmentation sur $N=105$ dimensions

### Kmeans Clustering

- **K= 75 à 90**
- sur **100%**
- métriques convergentes pour une séparation de 78 à 86 classes





**algorithme sélectionné**  
KMeans()

**hyperparamètre**  
K = 78

groupes

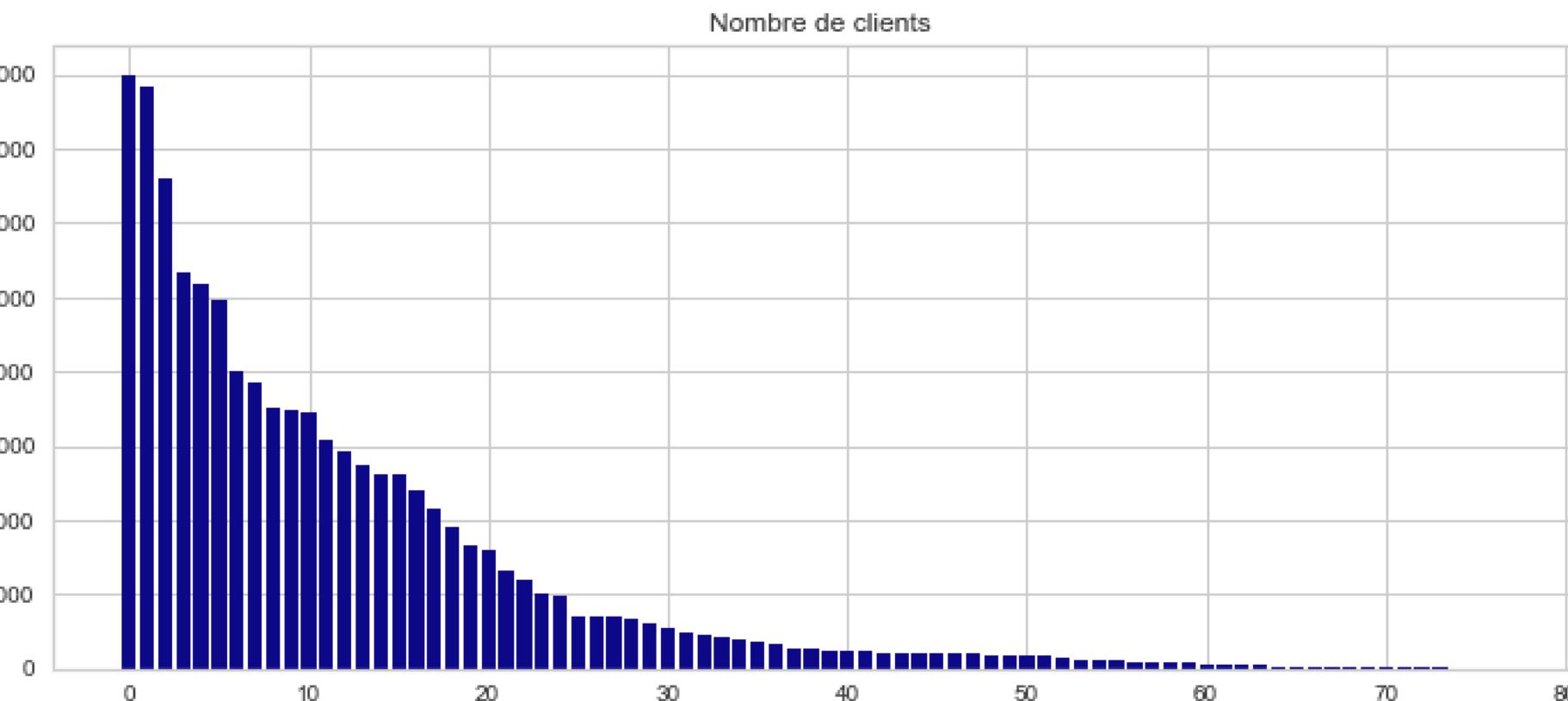
olist



### Pourcentages par cluster (trié)

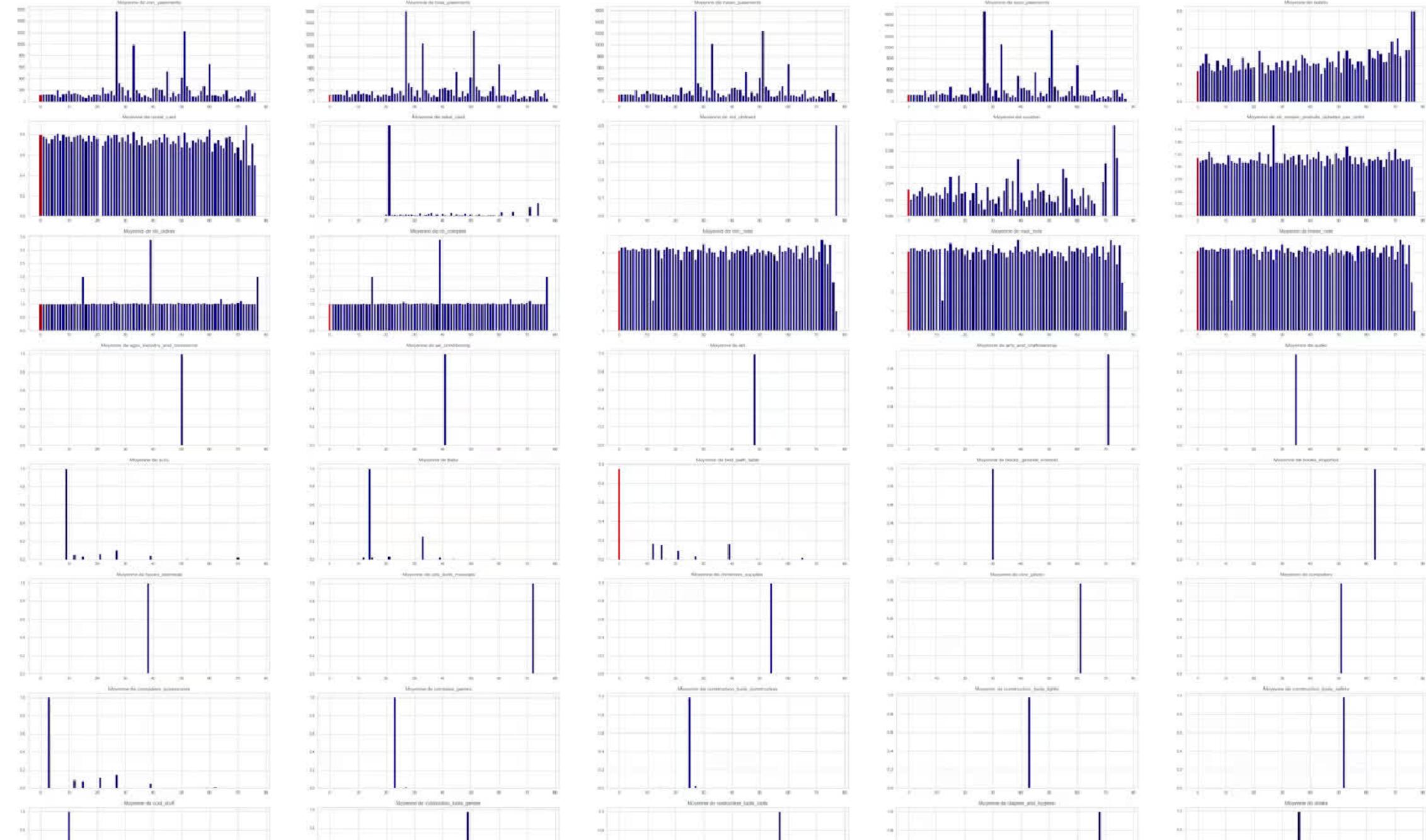
**0** 8 %  
**1** 7 %  
**2** 6 %  
**3** 6 %  
**4** 5 %  
....

### Nombre de clients par cluster (trié)





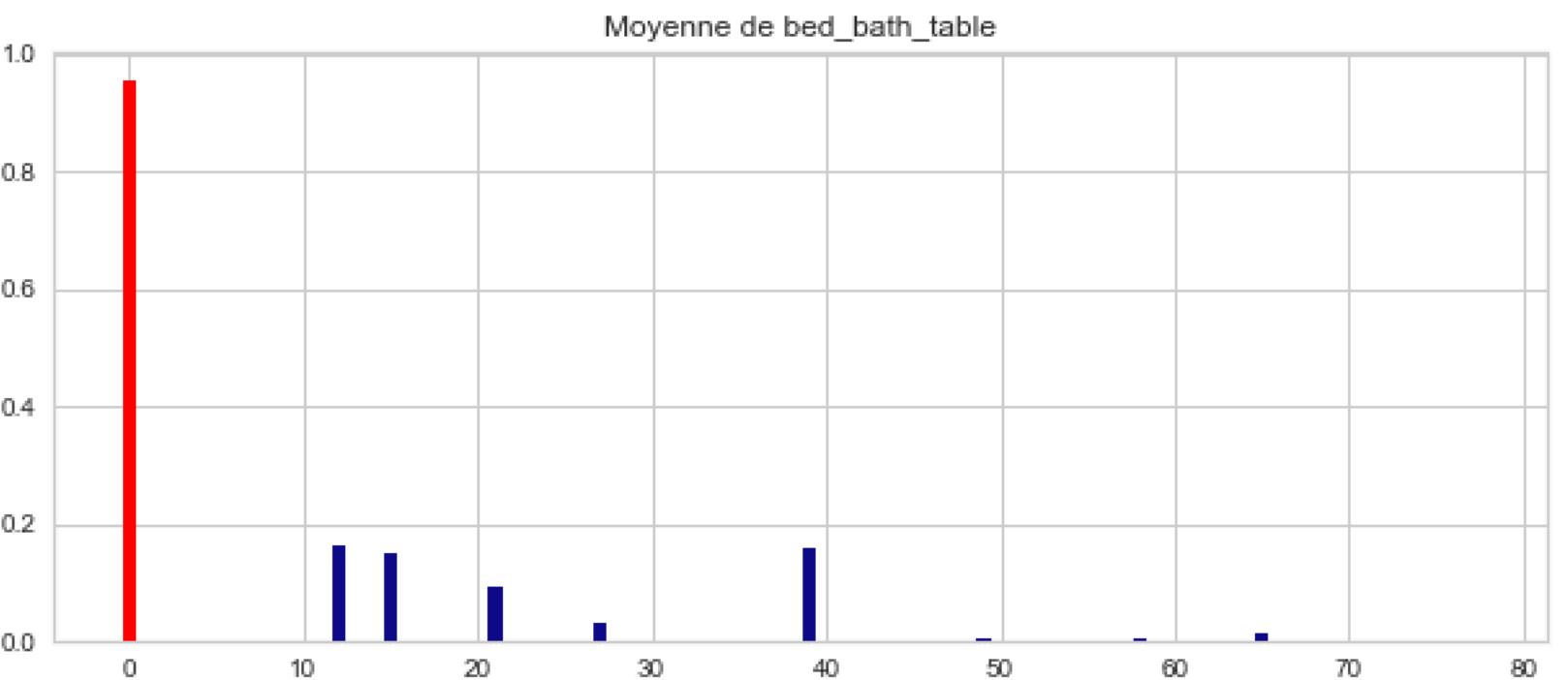
## Observations des clusters sur tous les descripteurs



olist



## Observations des clusters sur tous les descripteurs



olist



## N° Descripteurs importants

- 0.** Bed\_bath\_table, furniture\_mattress\_and\_upholstery,
- 1.** home\_confort
- 2.** Health\_beauty
- 3.** sports\_leisure
- 4.** computers\_accessories
- 5.** furniture\_decor
- 6.** housewares
- 7.** watches\_gifts
- 8.** telephony
- 9.** toys
- 10.** auto
- 11.** cool\_stuff
- 12.** note\_min, min\_delivery\_time, delais\_livraisons\_14\_max, furniture\_mattress\_and\_upholstery, home\_confort, health\_beauty, furniture\_decor, bed\_bath\_table, computers\_accessories, garden\_tools, watches\_gifts
  
- 13.** perfumery
- 14.** baby
  
- ....
- 27.** min\_paiements, max\_paiements, mean\_paiements, .... sum\_paiements, nb\_moyen\_produits\_achetes\_par\_ordre
- 39.** nb\_comptes, nb\_ordres, sum\_delivery\_time

## Description des clusters

Clients dépensiers dans une catégorie de produits ou dans un groupe de catégorie de produits

Clients ayant eu des délais de livraisons importants causes de faibles notes pour ces 8 catégories

Clients à fortes dépenses par commande

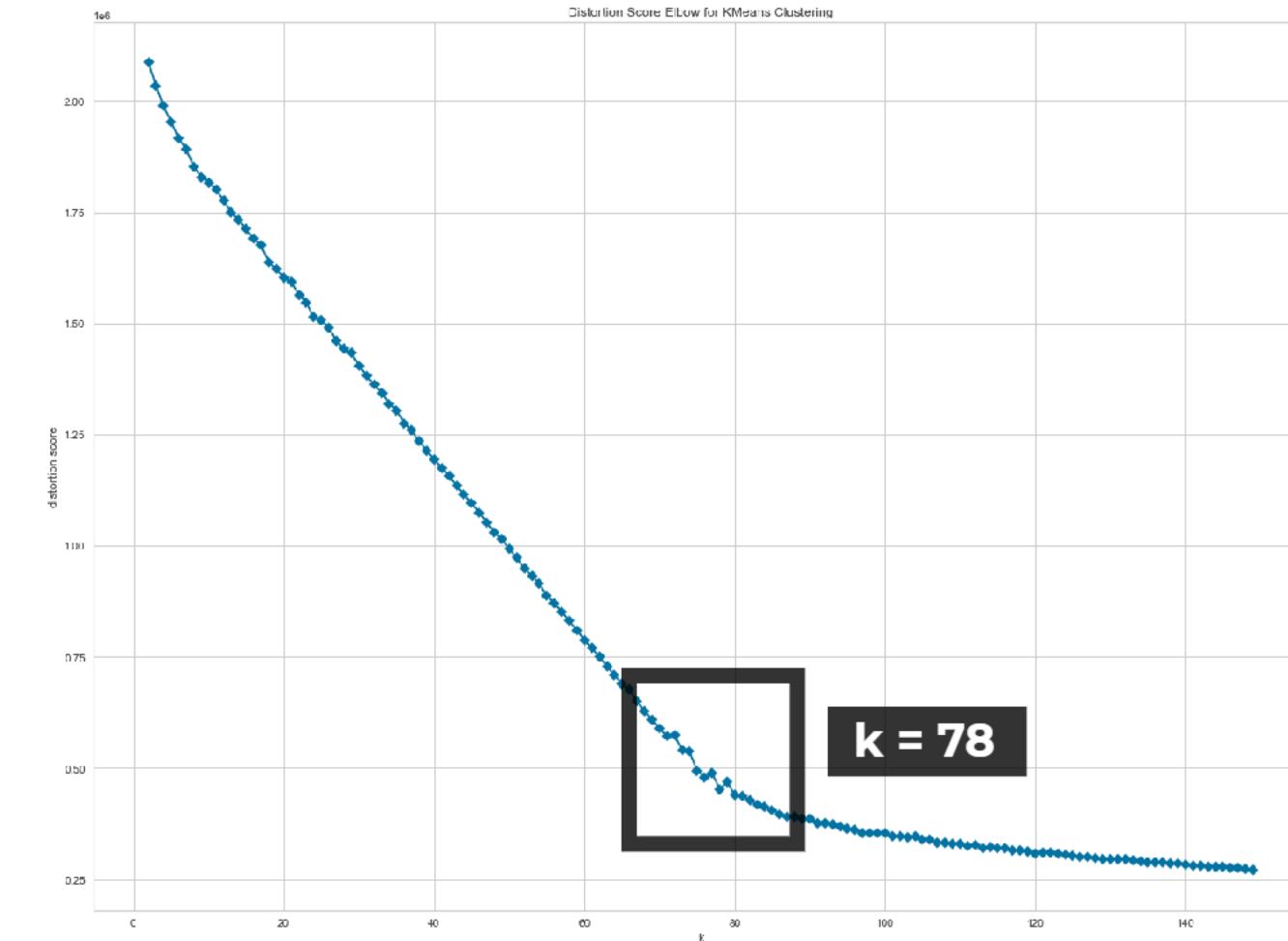
Clients qui commandent souvent

## fréquence de la segmentation et devis de contrat de maintenance

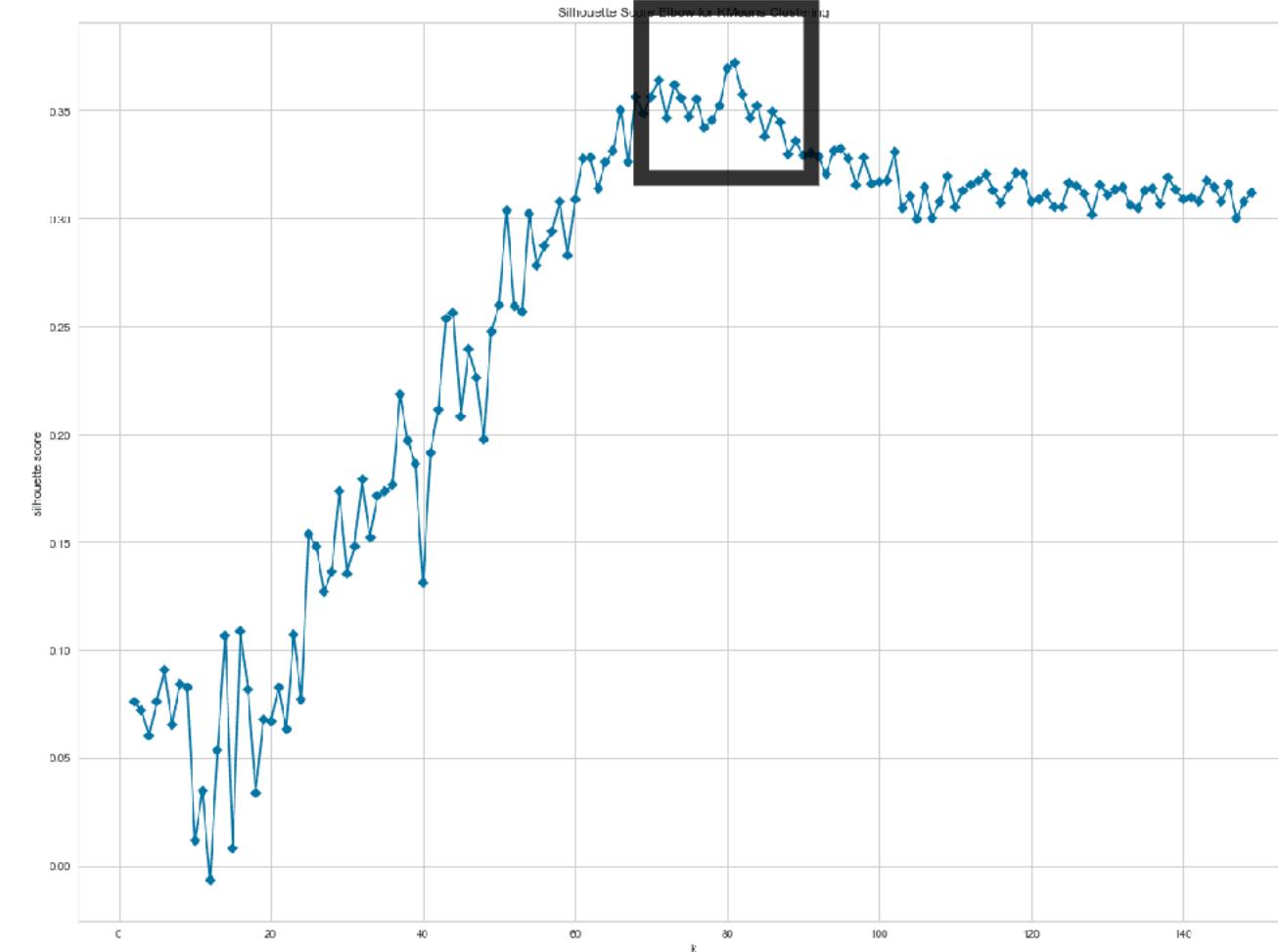
On va retirer les 6 derniers mois pour observer si nos données se segmentent différemment à l'aide du Kmeans

jeu 1 - nombre de clients 'now': 95419  
jeu 2 - nombre de clients '6 months ago': 69273

**distorsion/inertie**



**silhouette**



## fréquence de la segmentation et devis de contrat de maintenance

Le nombre de segments est identique, nous pouvons maintenant comparer si les clusters sont effectivement identiques.

La comparaison est possible en regardant les descripteurs et en utilisant pour un même jeu de données une mesure de similitarité comme l'indice de Rand sur nos modèles. (-1 à 1)

**jeu de données**  
6 months ago

**Rand score**  
0.968

### **fréquence de la segmentation et devis de contrat de maintenance**

Le score est très proche de 1, le modèle de segmentation est quasi identique.  
Il n'est donc pas nécessaire à priori de faire un entraînement régulier.

Il est cependant nécessaire d'établir un contrat de maintenance en cas de modifications des champs en base de données qui pourraient changer la segmentation des clients.

Bien sûr il faudra réaliser une segmentation automatique régulière à l'aide de notre modèle pré-entraîné afin d'identifier l'évolution des comportements ( cycle de vie d'un client)



## respect PEP8 - autoformatage du code

avant

```
[7]: index_label_0 = df_rfmb.query('Recency < @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_1 = df_rfmb.query('Recency < @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_2 = df_rfmb.query('Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_3 = df_rfmb.query('Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_4 = df_rfmb.query('Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue')
index_label_5 = df_rfmb.query('Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_6 = df_rfmb.query('Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue')
index_label_7 = df_rfmb.query('Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue')

[8]: #Separation en 8 classes arbitraires
df_rfmb.loc[index_label_0, ['Label']] = 0
df_rfmb.loc[index_label_1, ['Label']] = 1
df_rfmb.loc[index_label_2, ['Label']] = 2
df_rfmb.loc[index_label_3, ['Label']] = 3
df_rfmb.loc[index_label_4, ['Label']] = 4
df_rfmb.loc[index_label_5, ['Label']] = 5
df_rfmb.loc[index_label_6, ['Label']] = 6
df_rfmb.loc[index_label_7, ['Label']] = 7
```

après

```
[7]: index_label_0 = df_rfmb.query(
    "Recency < @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue"
).index
index_label_1 = df_rfmb.query(
    "Recency < @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue"
).index
index_label_2 = df_rfmb.query(
    "Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue < @level_monetaryValue"
).index
index_label_3 = df_rfmb.query(
    "Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue"
).index
index_label_4 = df_rfmb.query(
    "Recency >= @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue"
).index
index_label_5 = df_rfmb.query(
    "Recency >= @level_recency and Frequency < @level_frequency and MonetaryValue >= @level_monetaryValue"
).index
index_label_6 = df_rfmb.query(
    "Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue >= @level_monetaryValue"
).index
index_label_7 = df_rfmb.query(
    "Recency < @level_recency and Frequency >= @level_frequency and MonetaryValue < @level_monetaryValue"
).index

[8]: # Separation en 8 classes arbitraires
df_rfmb.loc[index_label_0, ["Label"]] = 0
df_rfmb.loc[index_label_1, ["Label"]] = 1
df_rfmb.loc[index_label_2, ["Label"]] = 2
df_rfmb.loc[index_label_3, ["Label"]] = 3
df_rfmb.loc[index_label_4, ["Label"]] = 4
df_rfmb.loc[index_label_5, ["Label"]] = 5
df_rfmb.loc[index_label_6, ["Label"]] = 6
df_rfmb.loc[index_label_7, ["Label"]] = 7
```

## Projet 4

Thank you !

