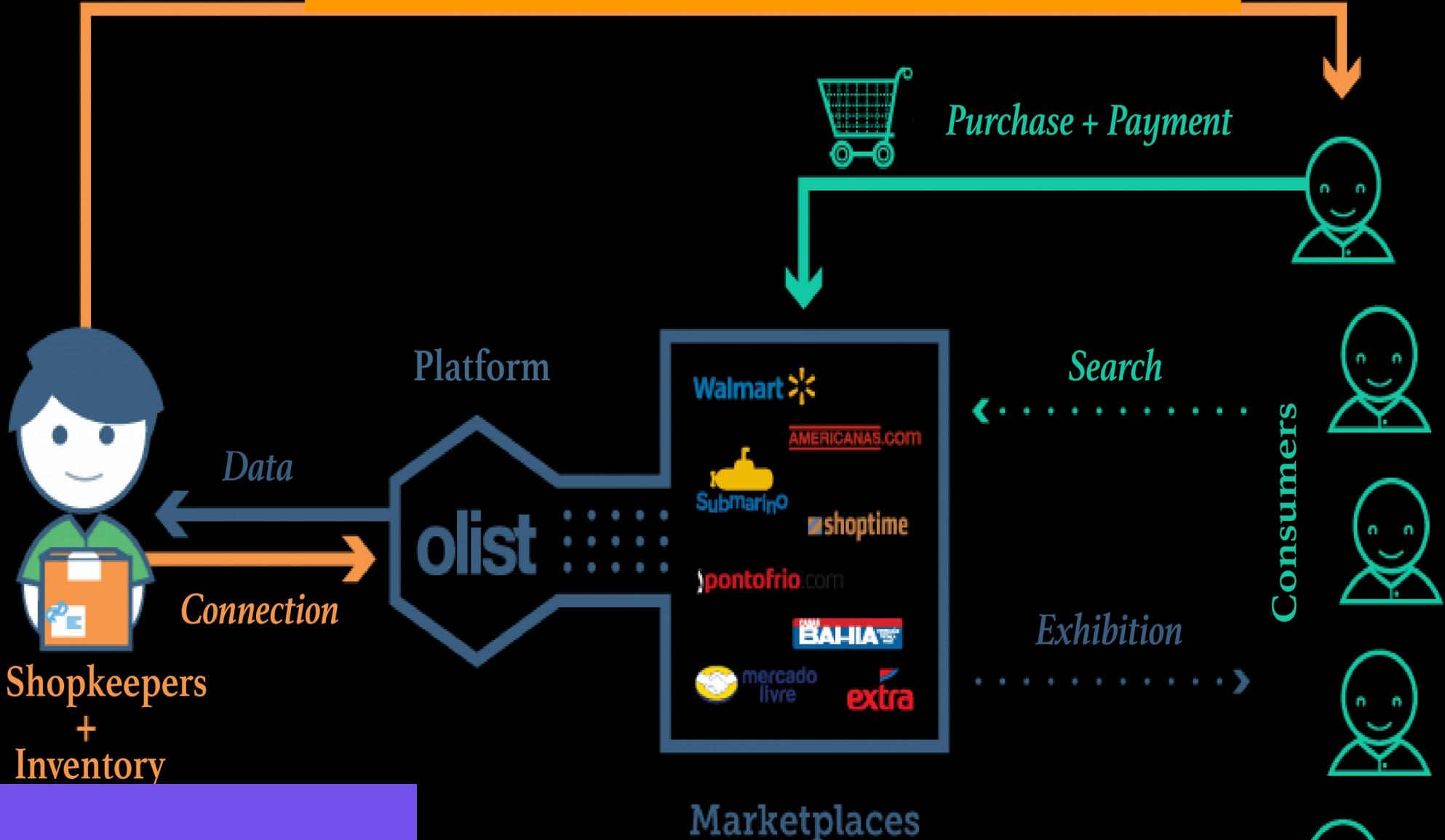


# Groupement des clients d'un site e-commerce



# Outline

- **Introduction**
  - Objectif de l'étude
  - Jeux de données
- **Nettoyage**
  - Suppression des doublons
  - Traitement des valeurs nuls
  - Suppression des **valeurs aberrants**
- **Analyse exploratoire**
  - Meilleurs vendeurs
  - Catégories des produits plus populaires
  - Analyse RFM (recency, frequency, monetary value)
- **Résultats des analyses**
  - Groupement sur données d'entraînement
  - Ajouter des valeurs test aux groupes
  - Optimisation des paramétrés
  - Réduction dimensionnelle pour visualiser les groupes
- **Conclusions**
  - Choix du meilleur algorithme de groupement

## Modalités de la soutenance

5 min - Présentation de la problématique, de son interprétation et des pistes de recherche envisagées.

5 min - Présentation du cleaning effectué, du feature engineering et de l'exploration.

10 min - Présentation des différentes pistes de modélisation effectuées.

5 min - Présentation du modèle final sélectionné et résultats.

5 à 10 minutes de questions-réponses.

# Objectif du modèle

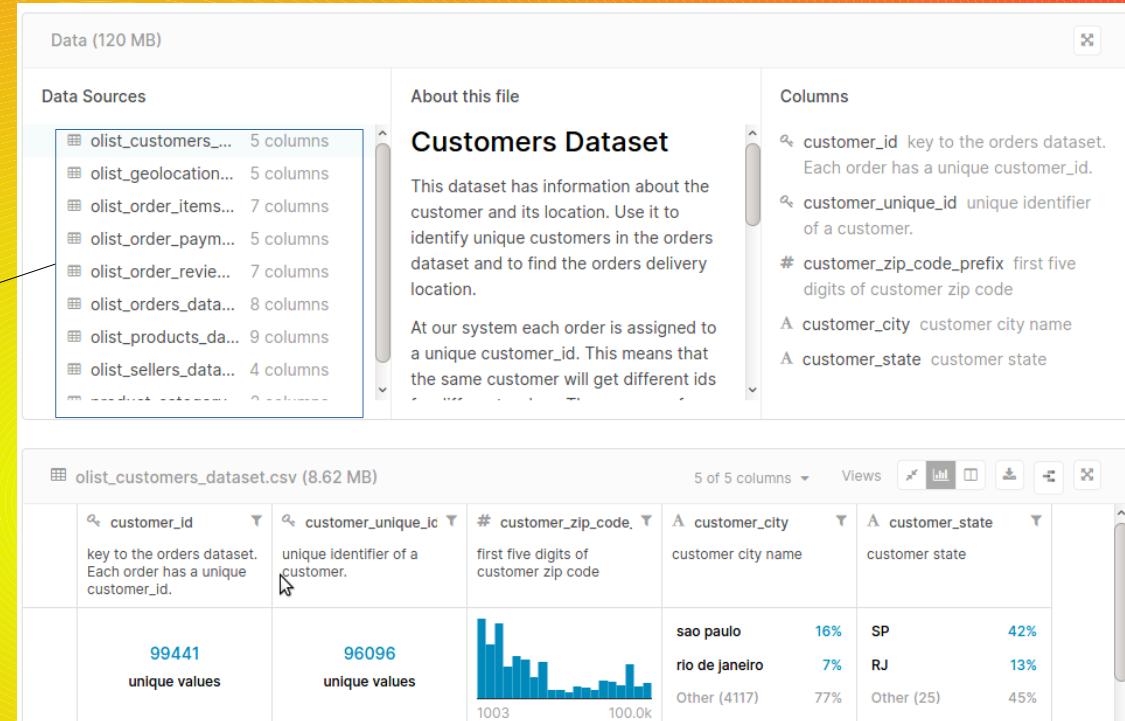
Dans cette étude avec un apprentissage non supervisé on essaye de trouver le meilleur modèle pour

- Regrouper ensemble des clients de profils similaires pour un site d' e-marketing
- Évaluer la performance du modèle de groupement de que on ajoute des nouveau clients

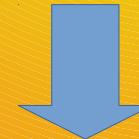
# Données du départ

Au départ environ 60 variables

Les données  
sont dans  
plusieurs fichiers  
CSV



Création d'un dataframe unique avec des features à modéliser pour chaque client



À chaque étape de jointure on vérifie que on a pas des doublons  
`print(df_order_by_customer["customer_id"].duplicated().value_counts())`

# Nettoyage des données

Présentation de la méthode et de l'outil

Présentation de l'outil

Présentation de la méthode

Présentation de l'outil

# Valeurs nuls

On a pas des valeur nuls  
après la jointure

Pas des valeurs négatifs

num	name	%null	% unique values
0	customer_id	0.0	100.000000
1	order_item_id	0.0	0.023311
2	amount_prod_categories	0.0	0.019257
3	price	0.0	7.909593
4	freight_value	0.0	8.152840
5	payment_value	0.0	30.668423
6	review_score	0.0	0.011149
7	frequency	0.0	0.018244
8	recency	0.0	0.622308

Après le nettoyage il nous restent 8 variables

alldata.describe() # stats for the final dataframe								
	order_item_id	amount_prod_categories	price	freight_value	payment_value	review_score	frequency	recency
count	98665.000000	98665.000000	98665.000000	98665.000000	98665.000000	98665.000000	98665.000000	98665.000000
mean	1.376496	1.131941	138.377845	22.947594	206.945605	4.089770	1.148523	244.811017
std	2.315509	0.567366	211.380815	21.809864	624.185201	1.342549	0.554749	153.386320
min	1.000000	0.000000	0.850000	0.000000	9.590000	1.000000	1.000000	0.000000
25%	1.000000	1.000000	45.950000	13.890000	62.920000	4.000000	1.000000	121.000000
50%	1.000000	1.000000	87.000000	17.240000	110.320000	5.000000	1.000000	226.000000
75%	1.000000	1.000000	149.980000	24.350000	196.600000	5.000000	1.000000	355.000000
max	231.000000	22.000000	13440.000000	1794.960000	109312.640000	5.000000	22.000000	728.000000

# Valeurs aberrants

On a pas des valeur nuls  
après la jointure

Pas des valeurs négatifs

Methode IQR (interquartile range)

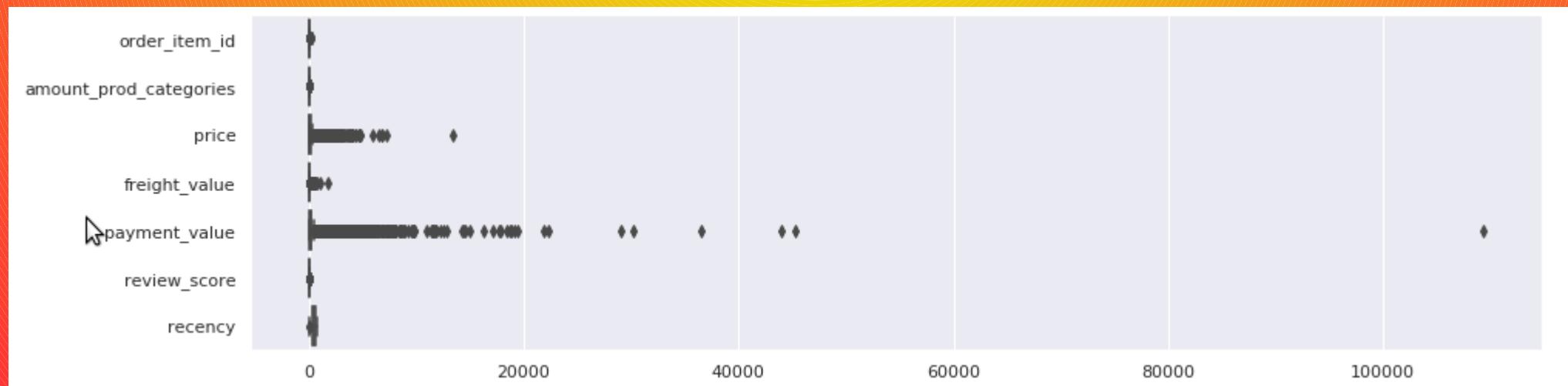
La règle  $2,0 \times$   
écart interquartile ne  
s'applique pas bien aux  
données de payement. Peux  
des valeurs ont un fort écart



Pour le nettoyage dans ce cas  
la vaut mieux imposer une  
seuil

```
alldata2 =  
alldata[alldata['payment_value'] < 6000]
```

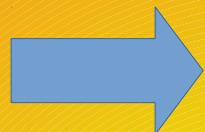
On perd que 76 datapoints mais on a plus  
les outliers



# Analyse exploratoire

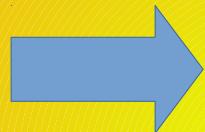
# Exploration des données

- Vendeurs



Notes et profit

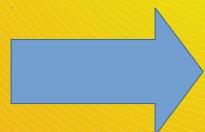
- Produits



Statistiques par  
catégorie :

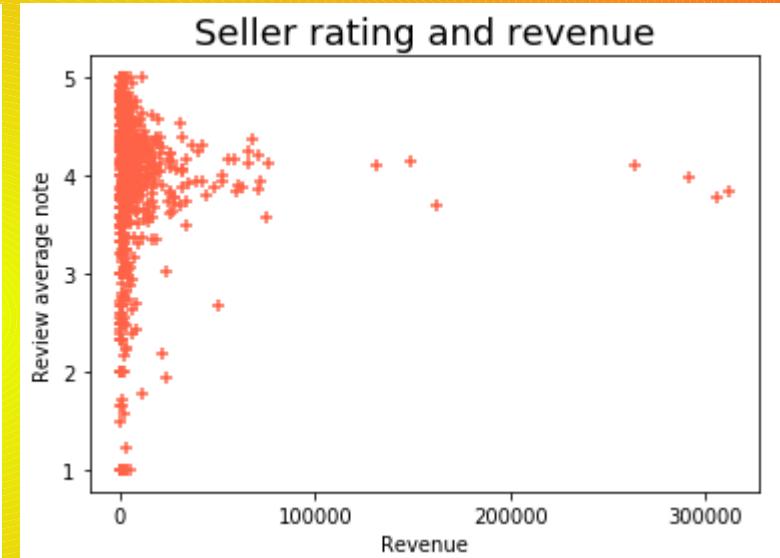
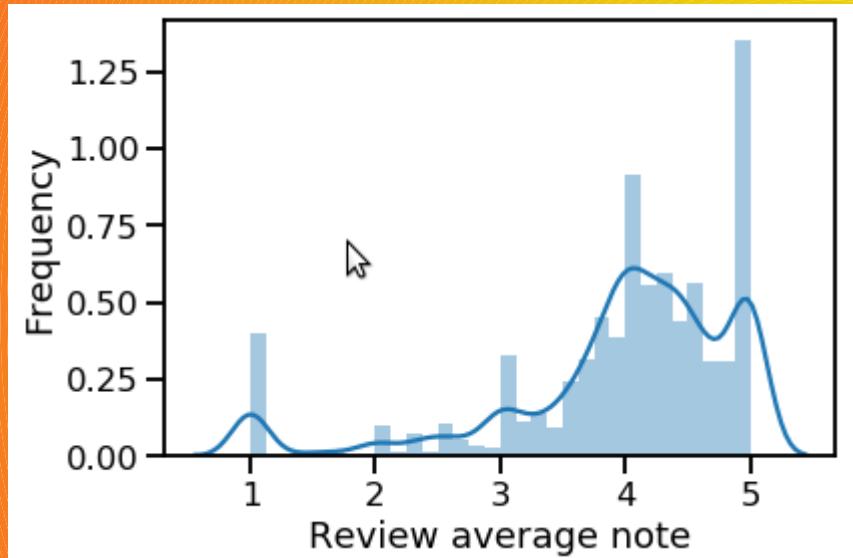
- Notes
- Num. d'achats
- Profit

- Clients



- Récence
- Fréquence
- Montant

# Les vendeurs : note et profit



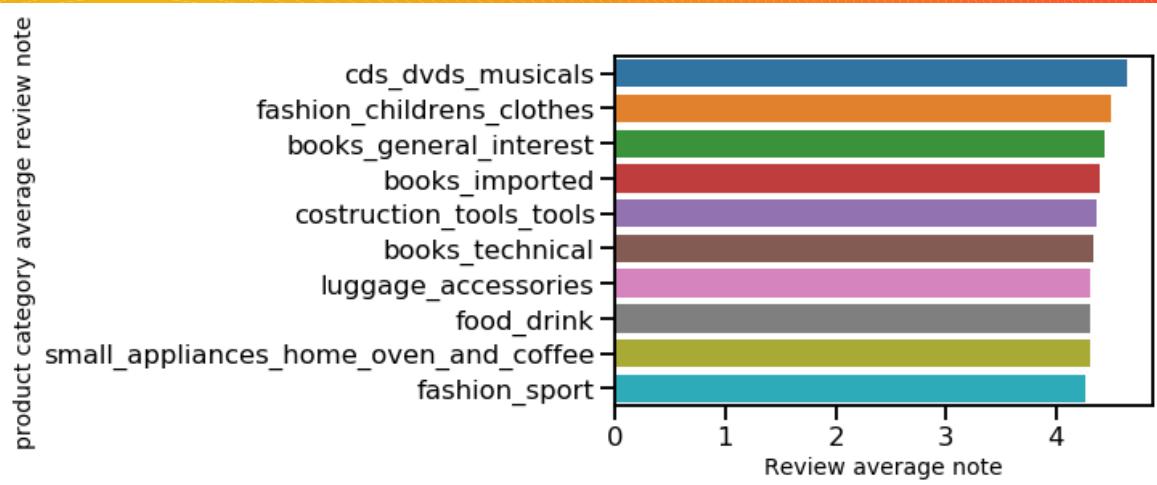
En général des bonnes notes  
Avec des mauvaises notes pas des bons profits

# Les produits : où investir ?

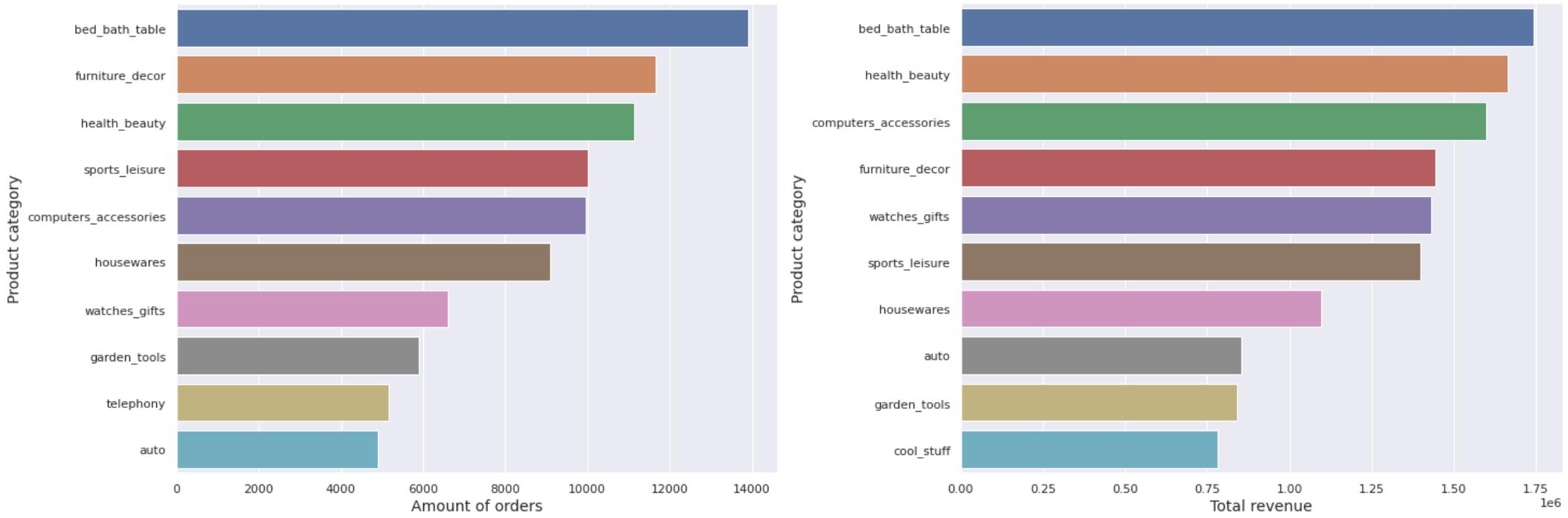


# Les top produits

## Les mieux notés



## Les plus vendus



# Les clients : classement RFM

**frequency:** approximée par le nombre d' achats  
**recency:** temps entre enregistrement et achat  
**monetary\_value:** somme des paiements

```
[ ] df_RFM.describe()
```

	index	frequency	recency	monetary_value
count	96770.000000	96770.000000	96770.000000	96770.000000
mean	49337.110912	1.118373	244.800434	162.523740
std	28476.524071	0.409977	153.357231	168.489635
min	0.000000	1.000000	0.000000	9.590000
25%	24680.250000	1.000000	121.000000	62.250000
50%	49337.500000	1.000000	226.000000	107.780000
75%	73997.750000	1.000000	355.000000	189.190000
max	98664.000000	8.000000	728.000000	1265.480000

```
[ ] print(df_RFM['frequency'].value_counts()) # check for one-time
```

1	87510
2	7708
3	1093
4	333
5	70
6	55
8	1
Name: frequency, dtype: int64	

La majorité des clients  
a acheté que une seule fois

# Regroupement des individus par comportements

- RFM
- Groupement non-supervisé

# Segmentation RFM

On transforme les valeurs  
RFM dans des notes

```
quintiles = df_RFМ[['recency', 'frequency', 'monetary_value']].quantile([.2, .4, .6, .8]).to_dict()quintiles

'frequency': {0.2: 1.0, 0.4: 1.0, 0.6: 1.0, 0.8: 1.0},
'monetary_value': {0.2: 54.53, 0.4: 86.74, 0.6: 133.85, 0.8: 218.2100000000004},
'recency': {0.2: 100.0, 0.4: 184.0, 0.6: 275.0, 0.8: 390.0}
```

Un valeur de 1 à 5 est calculé à partir des quintiles



Index	customer_id	frequency	recency	monetary_value	R	F	M	RFM Score	Segment
0	b4afeb58ac51bc903c5362286c6a5fce	8	288	992.96	2	5	5	255	can't lose
1	08a0a61f98f22952682f7e463ef2d43e	6	170	1252.08	4	5	5	455	loyal customers
2	cc3f0f9c25480b83c834bf8af1435a29	6	9	426.60	5	5	5	555	champions
3	620732fed5579e0bdfdddee02aad4c54	6	692	774.18	1	5	5	155	can't lose
4	109f80397e4897bbcf56313c25850332	6	112	1157.04	4	5	5	455	loyal customers

Selon le range des notes RFM on a  
des groupes des clients

# Les clients : 7 groupes RFM

Champions: Ils ont acheté récemment, achètent souvent et dépensent le plus (Récence : 4-5, Fréquence+Montant : 4-5)

Loyal Customers (clients fidèles): achètent régulièrement. Adaptés aux promotions. (Récence : 2-5, Fréquence+Montant : 3-5)

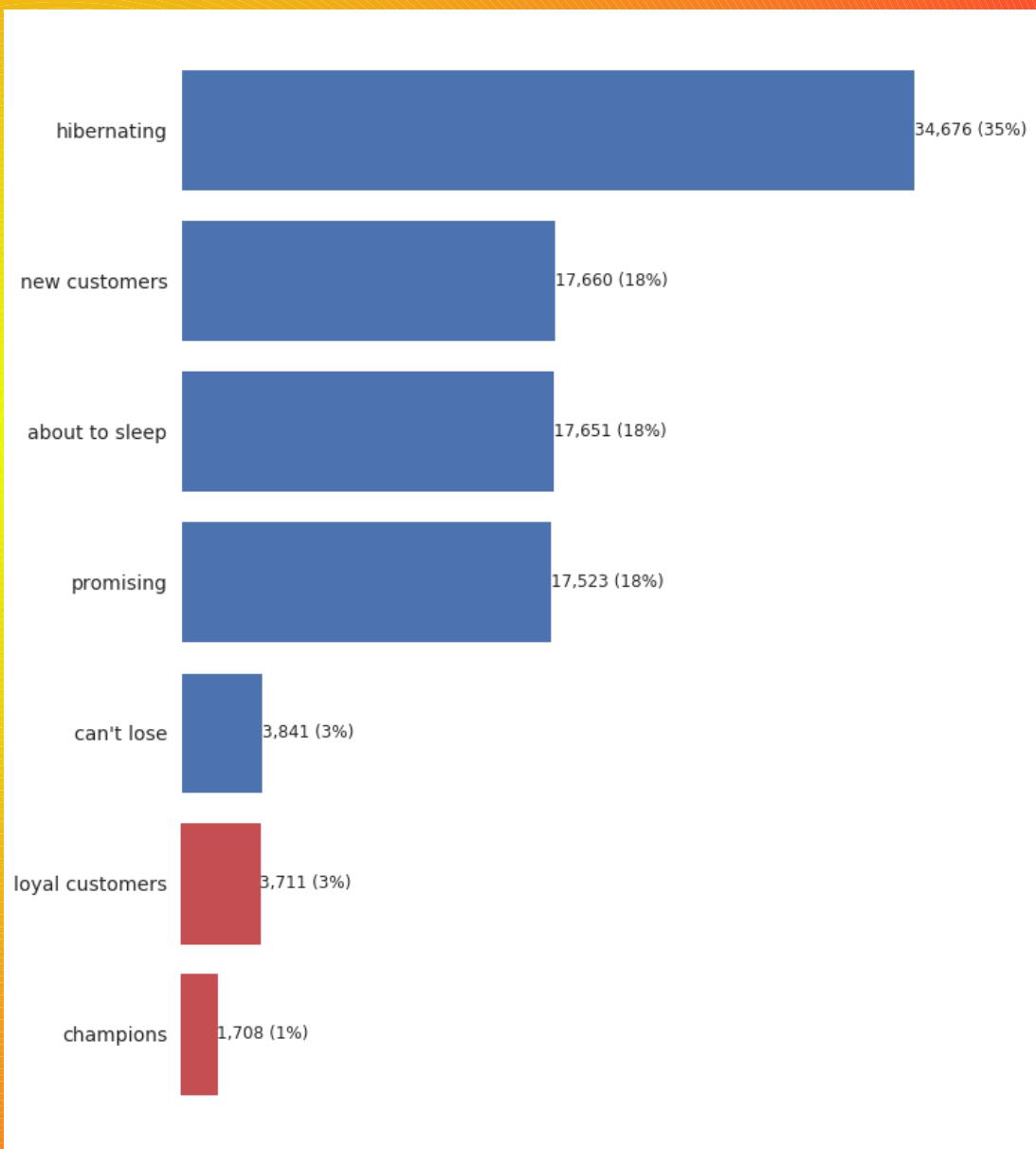
New customers (nouveaux clients) : acheteurs récents, ont assez bien dépensé (Récence : 4-5, Fréquence+Montant : 1-2)

Promising (promettant) : acheteurs récents, mais n'ont pas dépensé beaucoup. (Récence : 3-4, Fréquence+Montant : 0-1)

About To Sleep (presque oublié): Récence et fréquence inférieures à la moyenne. On les perdra s'ils ne sont pas réactivés.  
(Récence : 2-3, Fréquence+Montant : 0-2)

Can't lose (on ne peut pas les perdre): ont acheté fréquemment, mais ils ne sont pas revenu depuis longtemps.  
(Récence : 0-1, Fréquence+Montant : 4-5)

Hibernating (hibernés): Le dernier achat était de longue date et le nombre de commandes était faible. Peut être perdu.  
(Récence : 1-2, Fréquence+Montant : 1-2)



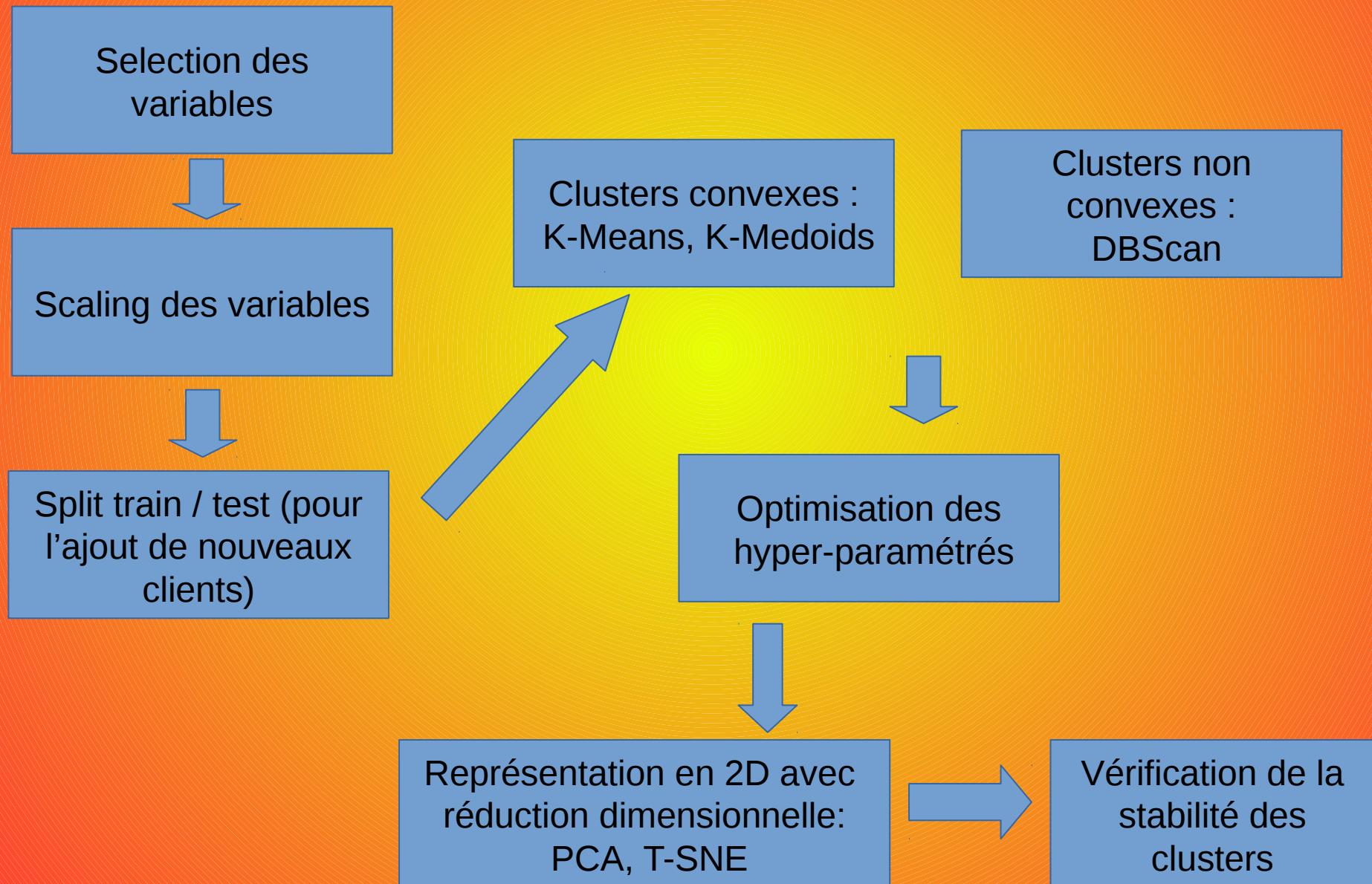
# Limites du groupement RFM

La majorité des clients a fait que un achat sur le site.

L'analyse RFM montre aussi que beaucoup des clients que ont acheté plusieurs fois sont pas réguliers.

On essaye d'avoir un meilleur groupement avec des algorithmes de classement non-supervisé

# Algorithmes de clustering non supervisées

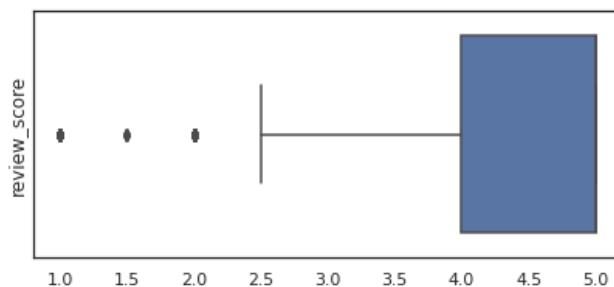
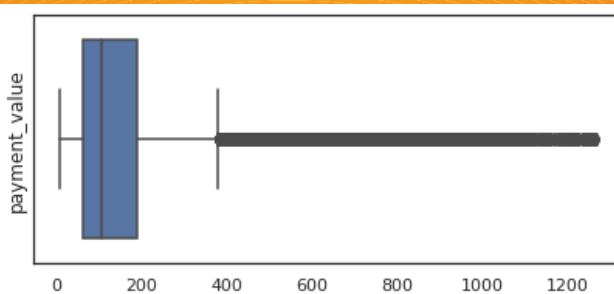
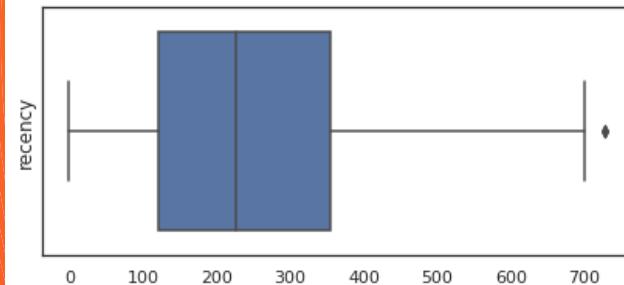
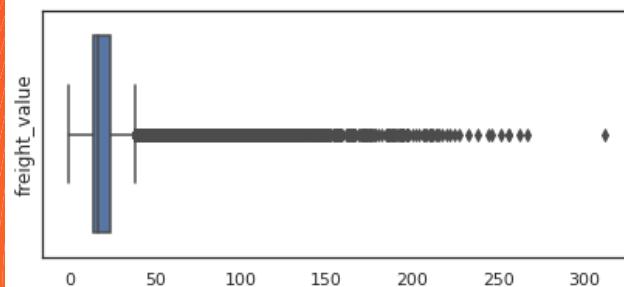
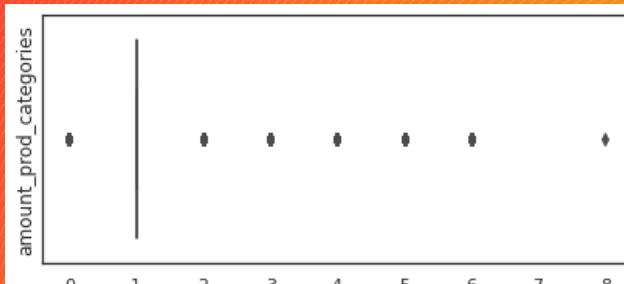


# Sélection des variables pour le classement non-supervisé

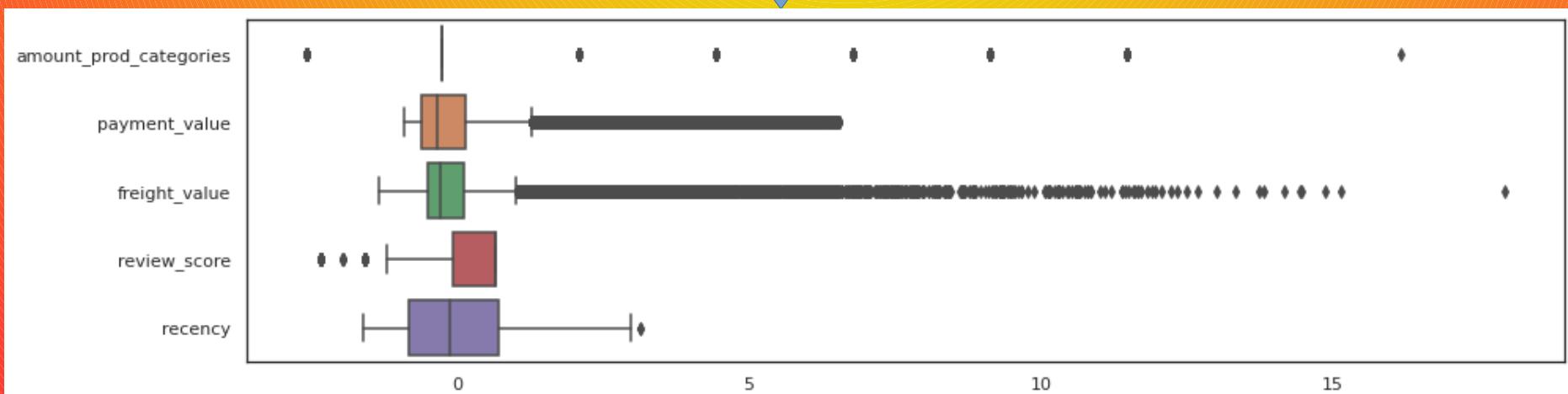
Matrice des corrélations



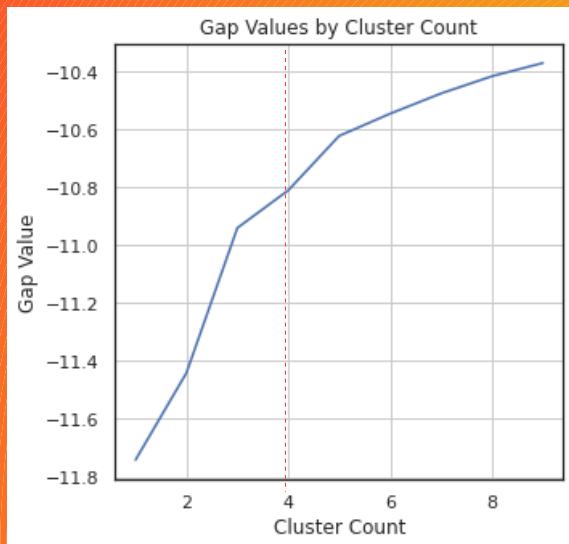
# Variables à l'échelle



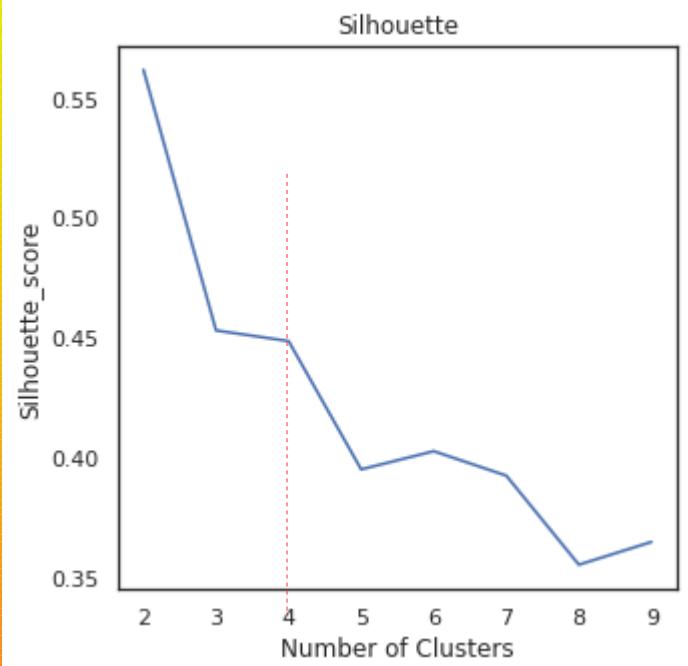
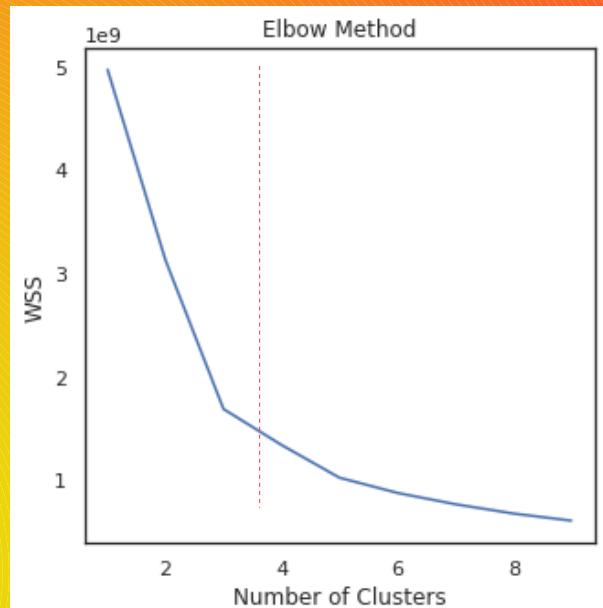
Scaling



# K-Means : mesures de performance



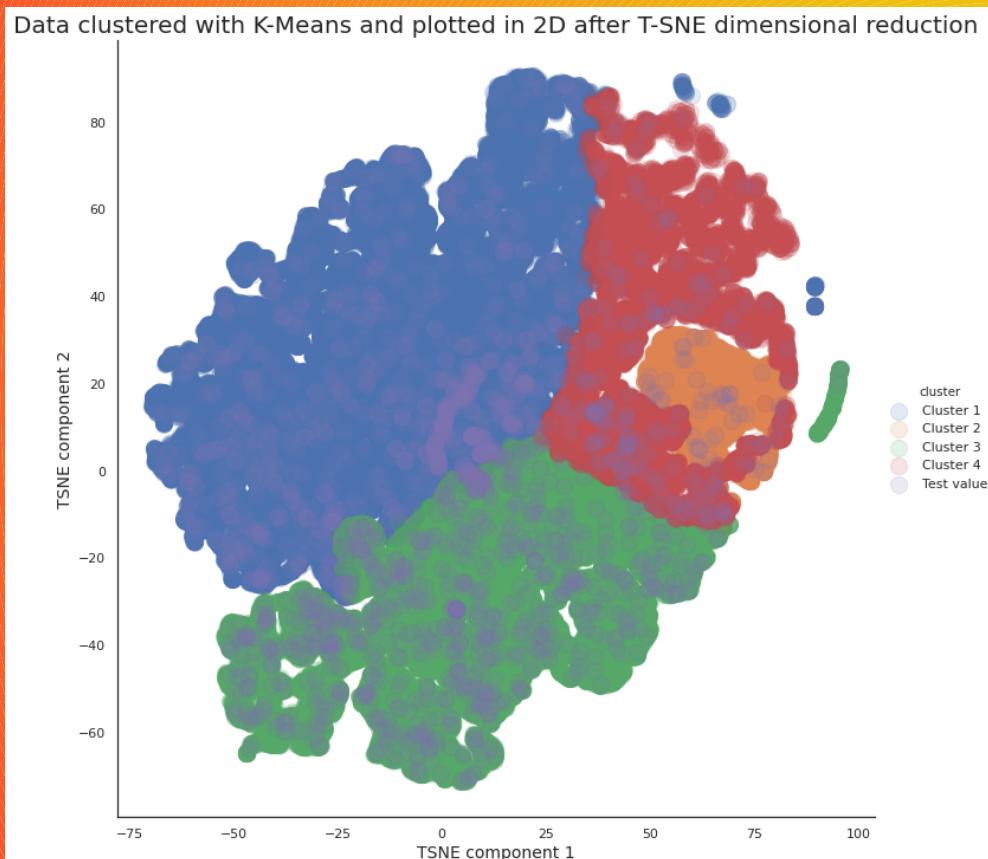
clusterCount	gap	WSS 1e8
1.0	-11.742355	49.7
2.0	-11.440970	31.2
3.0	-10.939488	16.9
4.0	-10.808794	13.4
5.0	-10.622410	10.2
6.0	-10.545387	8.8
7.0	-10.475779	7.7
8.0	-10.416236	6.8
9.0	-10.371982	6.1



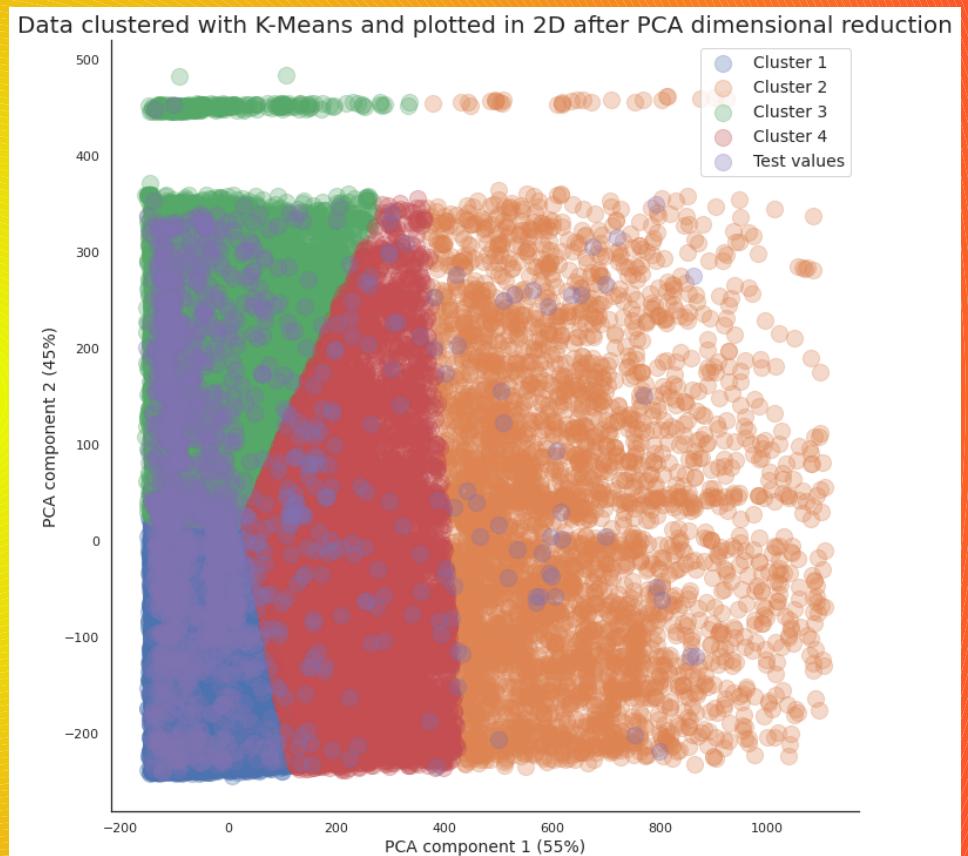
On calcule plusieurs mesures de performance : la choix est pour 4 clusters

# Réduction dimensionnelle

T-SNE (t-distributed Stochastic Neighbor Embedding)



ACP (analyse composants principaux)



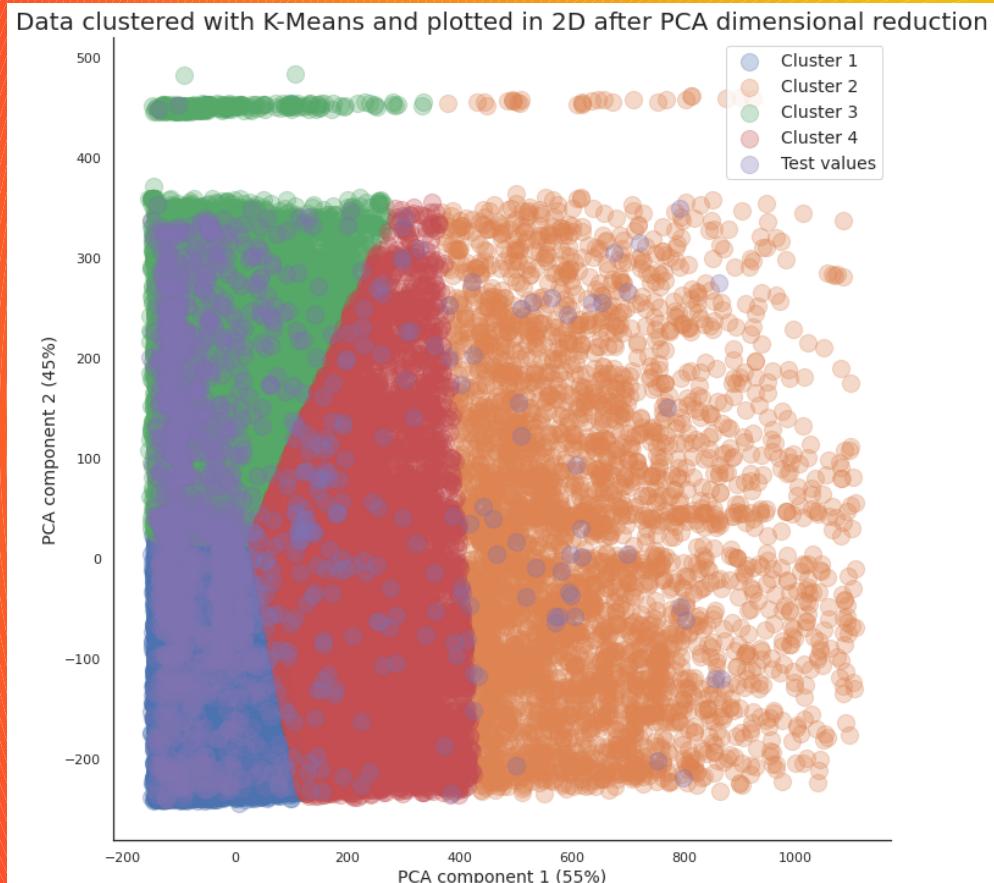
$$T_{\text{exec}} = 4.26 \text{ s}, \text{perplexity} = 28$$

Avec paquet « cuml » installé sur google colabs

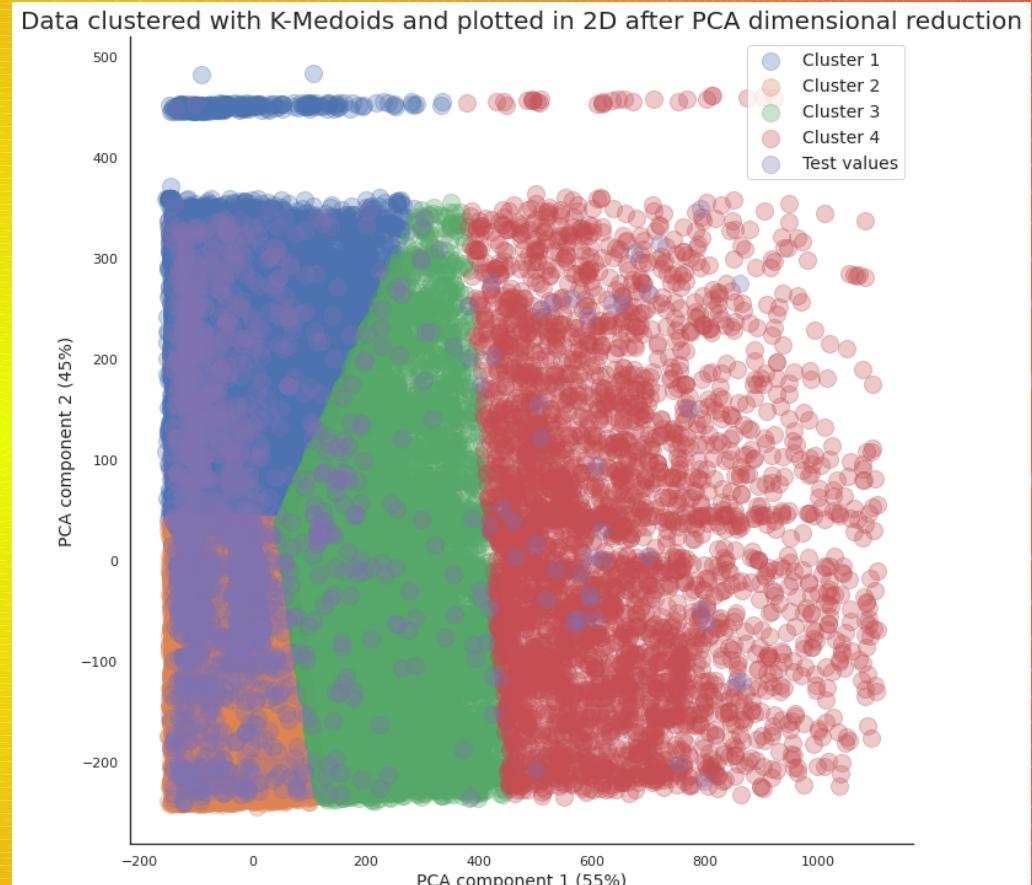
$$T_{\text{exec}} = 0.074 \text{ s}$$

Les valeurs « test » sont dans les clusters. L'ACP représente mieux les clusters que t-SNE

# K-Means vs. K-Medoids



K-Means clustering  
time elapsed: **1.2 seconds**



K-Medoids clustering  
time elapsed: **291 seconds**

# DB-Scan : groupement par densité

```
ITER | INFO | DIST | CLUS
-----|-----|-----|-----|
1 | Tested with eps = 0.01 and min_samples = 10 | 4.321 | 6
2 | Tested with eps = 0.01 and min_samples = 20 | 4.318 | 0
3 | Tested with eps = 0.01 and min_samples = 75 | 4.318 | 0
4 | Tested with eps = 0.02 and min_samples = 10 | 4.321 | 6
5 | Tested with eps = 0.02 and min_samples = 20 | 4.318 | 0
6 | Tested with eps = 0.02 and min_samples = 75 | 4.318 | 0
```

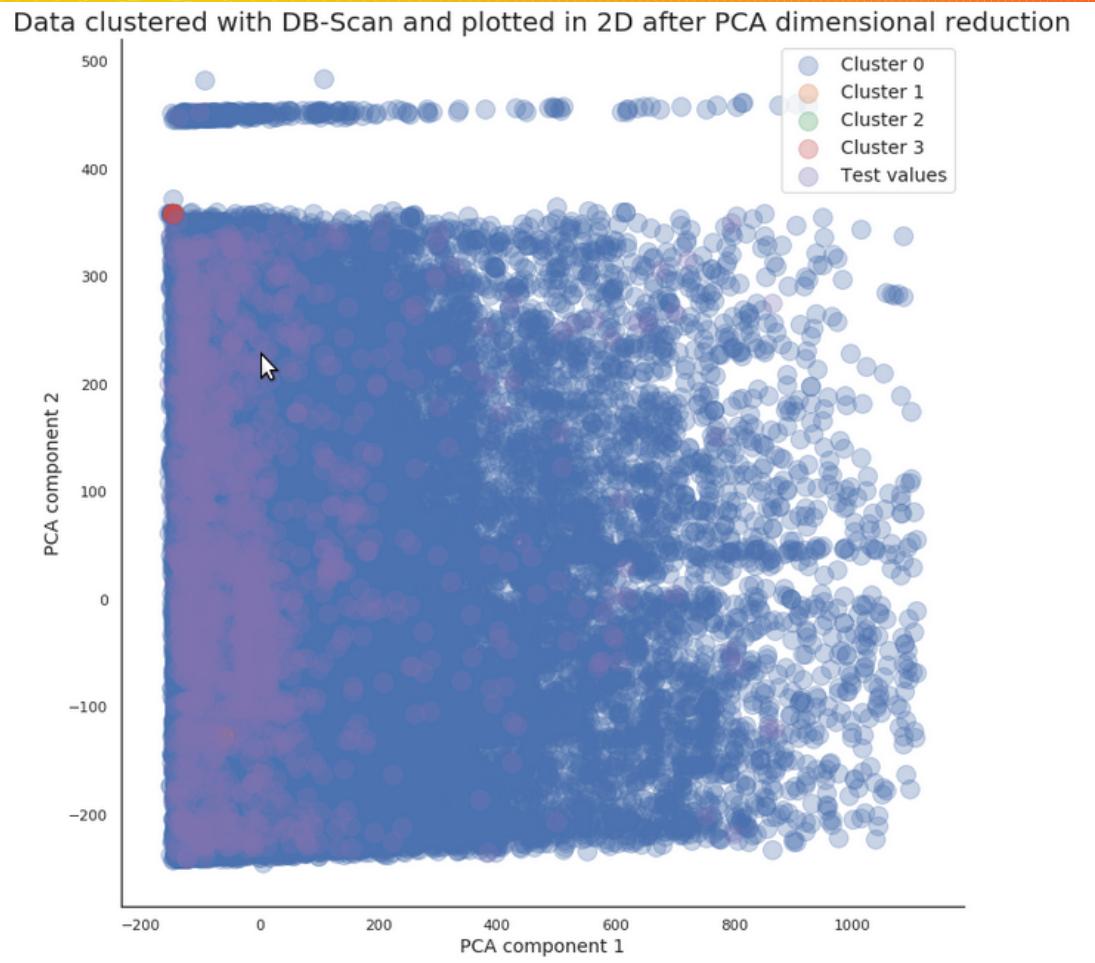
```
eps = 0.001
min_samples_to_test = 11
time_start = time.time()
dbSCAN_model = DBSCAN(eps = eps, min_samples = min_samples_to_test)
```

Time elapsed: 0.797 s

```
dfDB_PCA['cluster'].value_counts()
```

→

-1.0	94796
42.0	1936
1.0	14
2.0	12
0.0	12
Name: cluster, dtype: int64	



# Choix du meilleure algorithme

## K-Means vs. K-Medoids

Même résultat, temps de calcul réduit par K-means

## K-Means vs. DB-Clusters

Avec DB cluster on a pas une bonne segmentation,  
pire que les autres méthodes



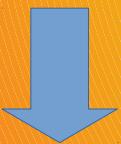
K-Means est le gagnant

# K-Means clusters stabilité

Segmentation par K-Means  
du test set



Comparaison des proportions des éléments  
par chaque cluster entre train et test set



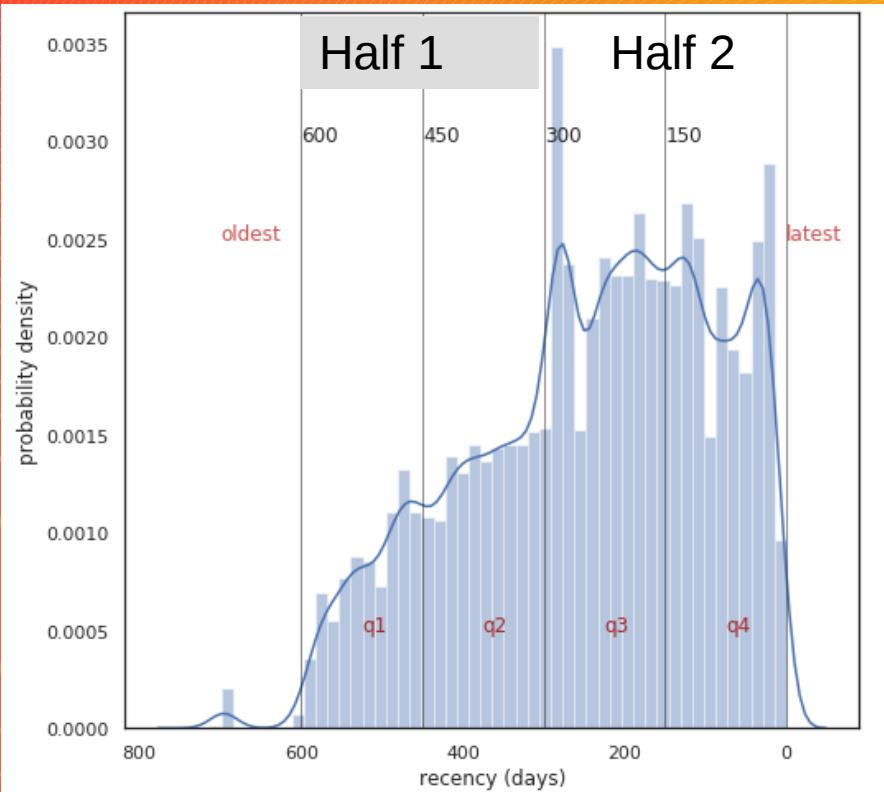
Chi square test :  
H0 est l'hypothèse de proportion similaire,  
Vérifiée avec p-value > 0,05

```
total elements in train set: 94834
Cluster 1    45544
Cluster 3    33796
Cluster 4    11478
Cluster 2    4016
Name: cluster, dtype: int64
proportions cluster/total
Cluster 1    48.024970
Cluster 3    35.637008
Cluster 4    12.103254
Cluster 2    4.234768
```

```
total elements in test set: 1936
Cluster 3    821
Cluster 1    537
Cluster 2    482
Cluster 4    96
Name: cluster, dtype: int64
proportions cluster/total
Cluster 3    42.407025
Cluster 1    27.737603
Cluster 2    24.896694
Cluster 4    4.958678
Name: cluster, dtype: float64
```

Résultat :  
Proportions are similar for clusters between  
train and test set (fail to reject H0)  
significance=0.050, **p=0.121**

# K-Means clusters durée



Chi square test :  
H0 est l'hypothèse de proportion similaire,  
Vérifiée avec p-value > 0,05

proportions cluster/total for half1  
Cluster 4 44.140103  
Cluster 3 32.114907  
Cluster 1 14.697461  
Cluster 2 9.047528

proportions cluster/total for half2  
Cluster 4 38.749788  
Cluster 3 36.241363  
Cluster 1 15.610717  
Cluster 2 9.398132

Réultat :  
Proportions are similar for clusters between  
two halves of the examined 600 days timescale  
(fail to reject H0)

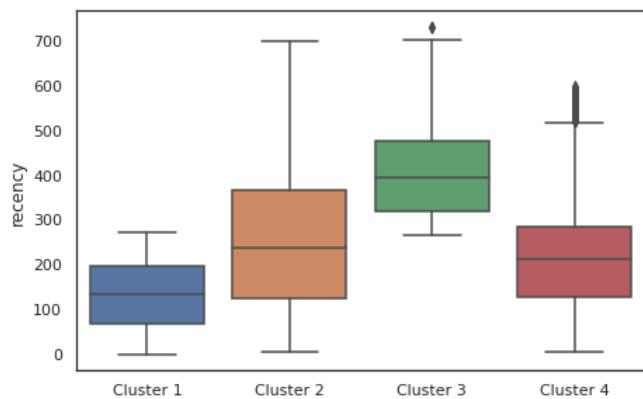
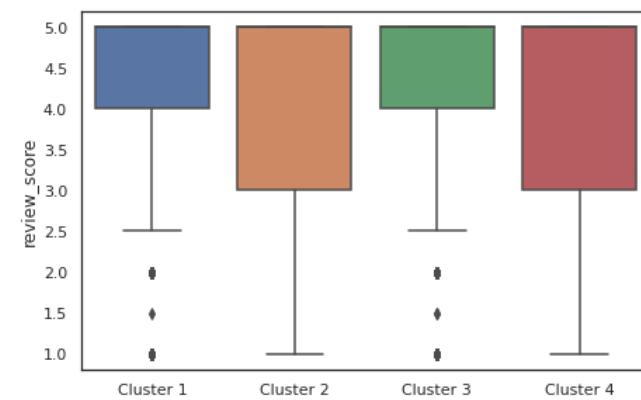
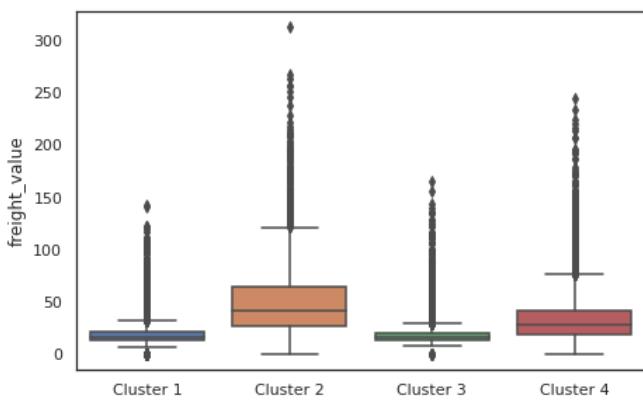
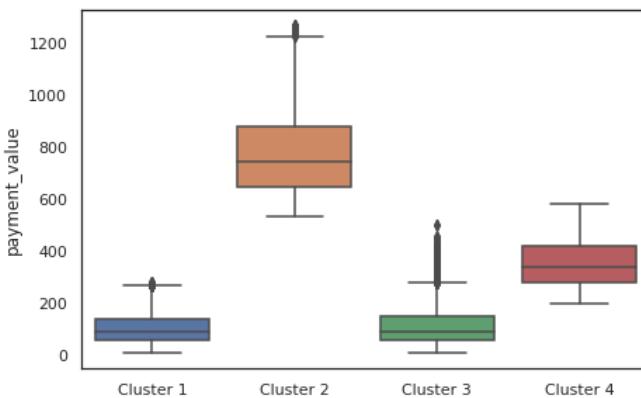
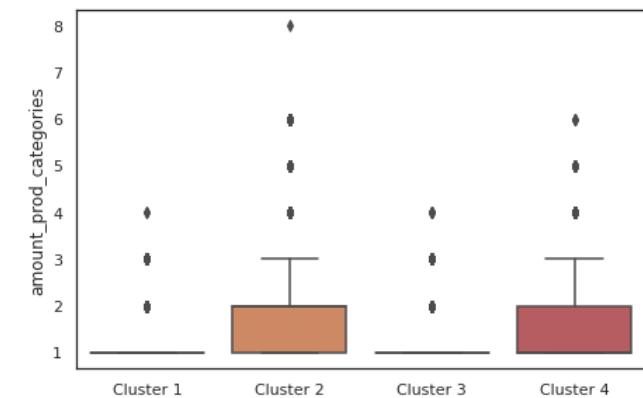
significance=0.050, **p=0.889**

proportions cluster/total for quarter 3 proportions cluster/total for quarter 4  
Cluster 2 36.627719 Cluster 4 39.875500  
Cluster 4 36.139153 Cluster 1 38.769191  
Cluster 3 17.779674 Cluster 2 11.995226  
Cluster 1 9.453454 Cluster 3 9.360083

Réultat :  
Proportions are similar for clusters between the last two quarters of the  
examined 600 days timescale (fail to reject H0)

significance=0.050, **p=0.716**

# Distribution des groupes par variable : K-Means



Le cluster 1 est déterminé surtout par la récence et note des produits

Le cluster 2 par le montant payé

Le cluster 3 par la récence

Le cluster 4 par la récence et le montant payé

Les clusters 2 et 4 sont différentiés des 1 et 3 à partir des nombres de catégories et note des produits

# Quels discours marketing pour quel cluster?

## Cluster 1

Notes : faibles  
Montants : petits  
Achats pas récents



faible priorité

## Cluster 2 et 4

Notes : bonnes  
Montants : importants  
Récence moyenne



à cibler en priorité

## Cluster 3

Notes : faibles  
Montants : petits  
Achats pas récents



à réactiver

# Conclusions

- K-Means est plus efficace que K-Medoids pour le partitionnement non-supervisé des clients.
- On a pas un bon partitionnement avec DB-Scan.
- Le Chi-Test montre que le groupement par K-means est stable. Durable sur deux ans.
- La représentation des données « test » en 2D montre que les nouveaux clients seront ajoutés automatiquement au classement
- On trouve 4 groupes comme valeur optimal pour K-Means basé sur plusieurs mesures de qualité (gap, WSS, Silhouette score)
- L'analyse aux composants principaux (ACP) est la méthode plus efficace pour visualiser le partitionnement en 2D