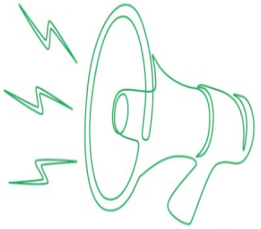
Segmentez des clients d'un site e-commerce

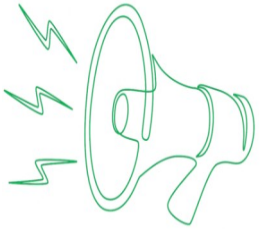
Projet 4

Sofia Velasco

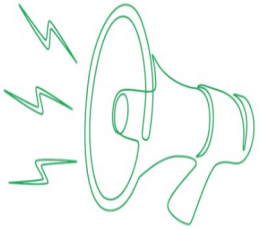


Objectif:

Identifier les bon et les moins bons clients en terme de commandes et de satisfaction



Analyse exploratoire



A. Caractéristiques générales des données

9 datasets : avec variables communes pour certains

'customer_id' → dans d1 et d6
'order_id' → dans d3, d4, d5 et d6
'product_id' → dans d3 et d7
'seller_id' → dans d8 et d3

d1: olist_customers_dataset.csv ✓
d2: olist_geolocation_dataset.csv → On ne sait pas à quoi elle fait référence ✗
d3: olist_order_items_dataset.csv ✓
d4: olist_order_payments_dataset.csv ✓
d5: olist_order_reviews_dataset.csv ✓
d6: olist_order_reviews_dataset.ods ✓
d7: olist_orders_dataset.csv ✓
d8: olist_products_dataset.csv → Inutile ✗
d9: olist_sellers_dataset.csv ✓

B. Fusion des différents dataset

1. d1 et d6
2. d3 et d7
3. d3_a (d3+d7) et d8
4. d3_b (d3_a+d8), d4, d5 et d6_a (d1+d6).

À chaque fusion :

- Nombre de lignes,
- 'NaN'
- Doublons
- Choix des variables, valeurs à retenir, et/ou calcul de nouvelles variables plus utiles:

  'payment_sequantial'→le plus grand (nombre total de moyen de paiement utilisés par le client)
  'payment_value_total', 'nombre_total_items', 'Prix_total_produits', 'freight_value_total' (total associé à chaque commande)
  'review_score'→'review_score_moyen' (moyenne associée à chaque commande)
  'Respect_temps_livraison'='order_delivered_customer_date'-'order_estimated_delivery_date'→(variable binaire 0 si respecté)

- Conversion 'dates + horaires' en dates uniquement ('datetime64[ns]')

C. Élimination de variables par rapport à notre objectif

Variables retenues éliminables :

- **Caractéristiques intrinsèques du produit:**
  Connues avant l'achat → sans influence sur satisfaction
  'product_name_lenght', 'product_description_lenght', 'product_photos_qty', 'product_weight_g', 'product_length_cm', 'product_height_cm' et 'product_width_cm'

- **Données de suivi:**
  Sans lien avec satisfaction → le client veut juste un colis livré au plus vite
  'order_approved_at' et 'order_delivered_carrier_date'

- **Identifiant de client unique à chaque commande:** 'customer_id'.
  On veut identifier les clients concrètement → **'customer_unique_id'**

- **la géolocalisation des clients:** 'customer_zip_code_prefix', 'customer_city' et 'customer_state'
  Même chose → on garde celle avec le moins de valeurs différentes possibles: 'customer_state'.

- **À plusieurs valeurs associées pour une même commande:**
  Empêche d'associer un type de produit ou un vendeur à la satisfaction des clients
  'product_category_name' et 'seller_id' → **'Nombres de catégories' et 'Nombres de vendeurs'**

  - **Liées au types de produits ou au vendeurs:**
    Devenant inutiles
    product_id', 'product_category_name', 'seller_id', 'seller_zip_code_prefix', 'seller_city' et 'seller_state'

D. Regrouper les commandes par 'customer_unique_id'

Suite au regroupement,

1. Analyse des NaN:

```
d3_final1_s.isna().sum()
```

customer_unique_id	0
order_id	0
review_score_moyen	749
payment_value	1
Prix_total_produits	0
freight_value_total	0
payment_sequential	1
payment_type	1
payment_installments	1
customer_zip_code_prefix	0
customer_city	0
customer_state	0
Nombre_total_catégories	0
Nombre_total_vendeurs	0
Nombre_total_items	0
order_status	0
order_purchase_timestamp_NEW	0
Respect_temps_livraison	0

dtype: int64

- Celles associés aux variables de ‘paiement’
  → Bug → élimination des lignes.

- Celles associés au 'review_score_moyen'
  → « 0 »

  Le ‘review score’ n'est pas obligatoire, enlever les lignes non renseignées pourrait pénaliser des clients qui peuvent être potentiellement bons

D. Regrouper les commandes par 'customer_unique_id'

2. Création des variables RFM:

L'analyse **RFM** :
- Agréger les données
- Segmenter les clients en groupes homogènes (**k-means**)

3 variables à calculer:

R (recency) → A quand remonte le dernier achat de chaque client?
(ie. Durée depuis le dernier achat)
(Date de dernière commande du dataset entier +1 jour)
-
(date de la dernière commande)

→ Engagement / Satisfaction

F (frequency) → Combien de fois un client a-t-il passé des commandes pendant la période d'analyse?
Nombre de commandes sur la période d'analyse sélectionnée pour chaque client

M (monetary) → Combien un client a-t-il dépensé au cours de la période analysée?
Somme montant achats sur la période de temps pour chaque client

→ Valeur des clients

D. Regrouper les commandes par 'customer_unique_id'

2. Dernier choix sur les variables:

- 'order_id' → 'count' → 'F-frequency'
- 'order_purchase_timestamp_NEW' → 'max' (plus récente) → R-recency
- 'payment_value' → 'sum' → 'M-monetary'

- 'order status' → plus nécessaire.

- 'Respect_temps_livraison' → 'sum' → 'Nombre_de_commandes_respectant_delais_livraison'

- 'customer_state'
- 'payment_type'

➡ Un seul ? → Non, élimination

- 'Prix_total_produits' → 'Dépenses_période_produits'
- 'freight_value_total' → 'Dépenses_période_frais'

➡ 'Diff_dépenses_produits_frais'

- Pour toutes les autres variables → 'moyenne_par_client'

➡ Impact satisfaction clients?

D. Regrouper les commandes par 'customer_unique_id'

3. Élimination des doublons.

Données d'achats depuis septembre 2016, jusqu'à septembre 2018.

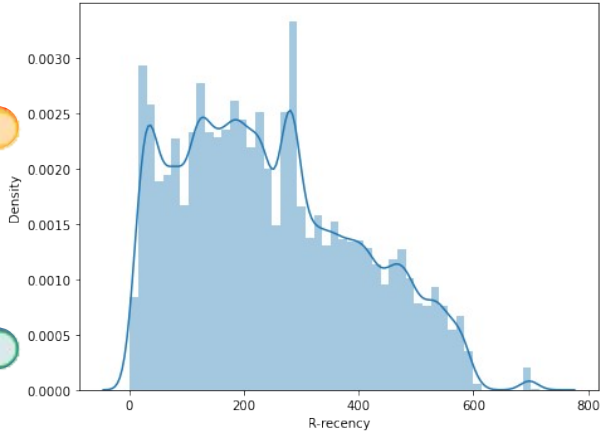
```
customer_unique_id                                    95419
R-recency                                             95419
F-frequency                                           95419
M-monetary                                            95419
Nombre_de_commandes_respectant_delais_livraison       95419
Diff_dépenses_produits_frais                          95419
review_score_moyen_moyenne_par_commande_par_client    95419
payment_sequential_moyenne_par_commande_par_client    95419
payment_installments_moyenne_par_commande_par_client  95419
Nombre_total_catégories_moyenne_par_commande_par_client  95419
Nombre_total_vendeurs_moyenne_par_commande_par_client  95419
Nombre_total_items_moyenne_par_commande_par_client    95419
dtype: int64
```

➡ Variables Finales

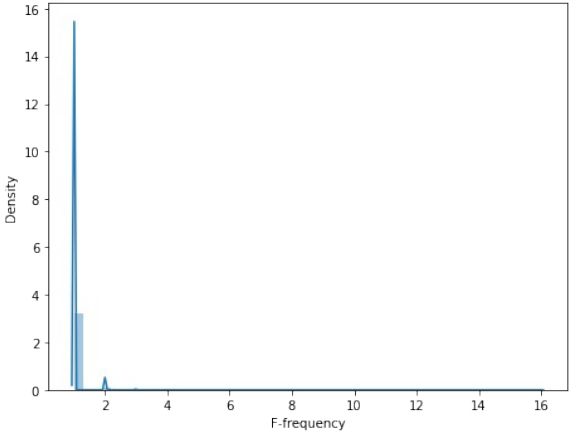
E. Distributions des variables

Variables RFM

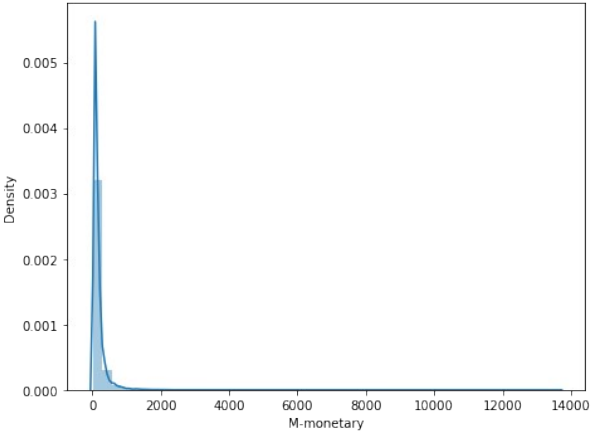
	count	mean	std	min	25%	50%	75%	max
R-recency	95419.0	244.496484	153.153950	1.00	120.0	225.00	354.00	730.00
F-frequency	95419.0	1.034018	0.211235	1.00	1.0	1.00	1.00	16.00
M-monetary	95419.0	166.070491	228.341907	9.59	63.1	107.95	183.27	13664.08



La plupart derniers achat dans les 350 jours avant le 3 septembre 2018



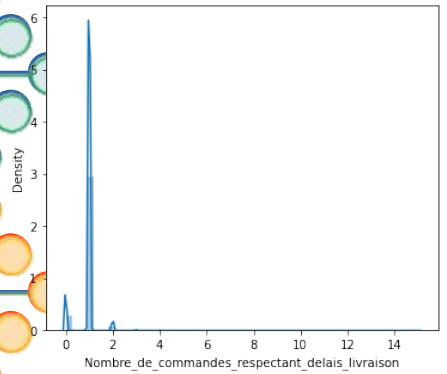
Seul 3.05% on fait au moins 2 commandes

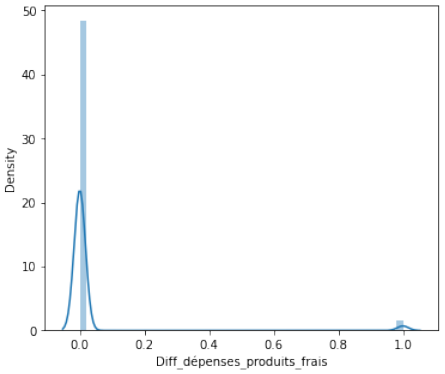


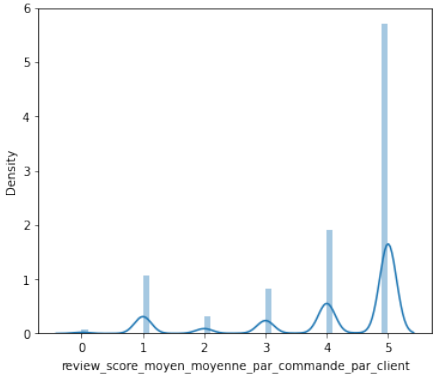
La plupart ont dépensé entre 0 et 180

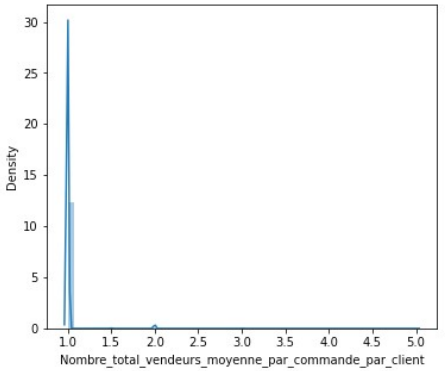
E. Distributions des variables

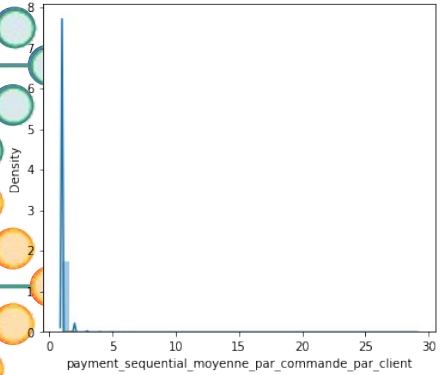
Autres variables

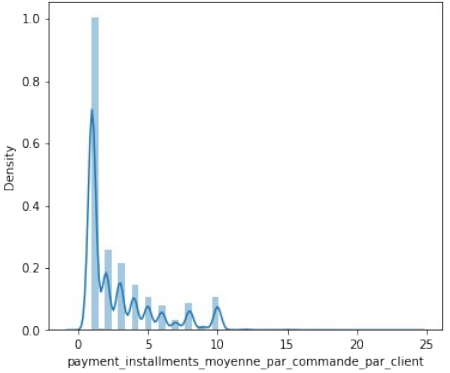


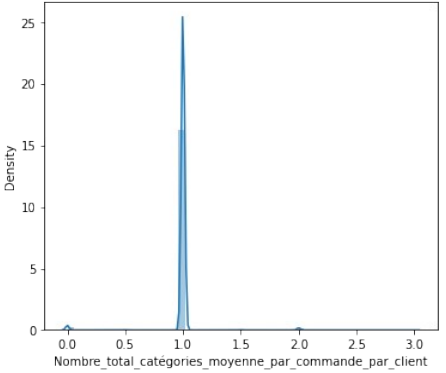


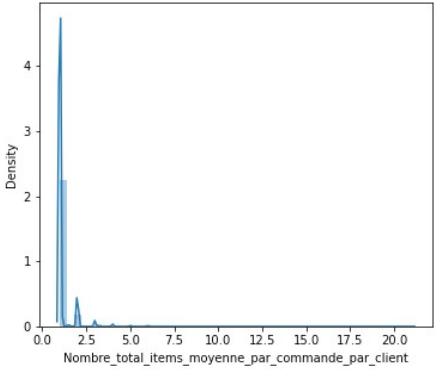






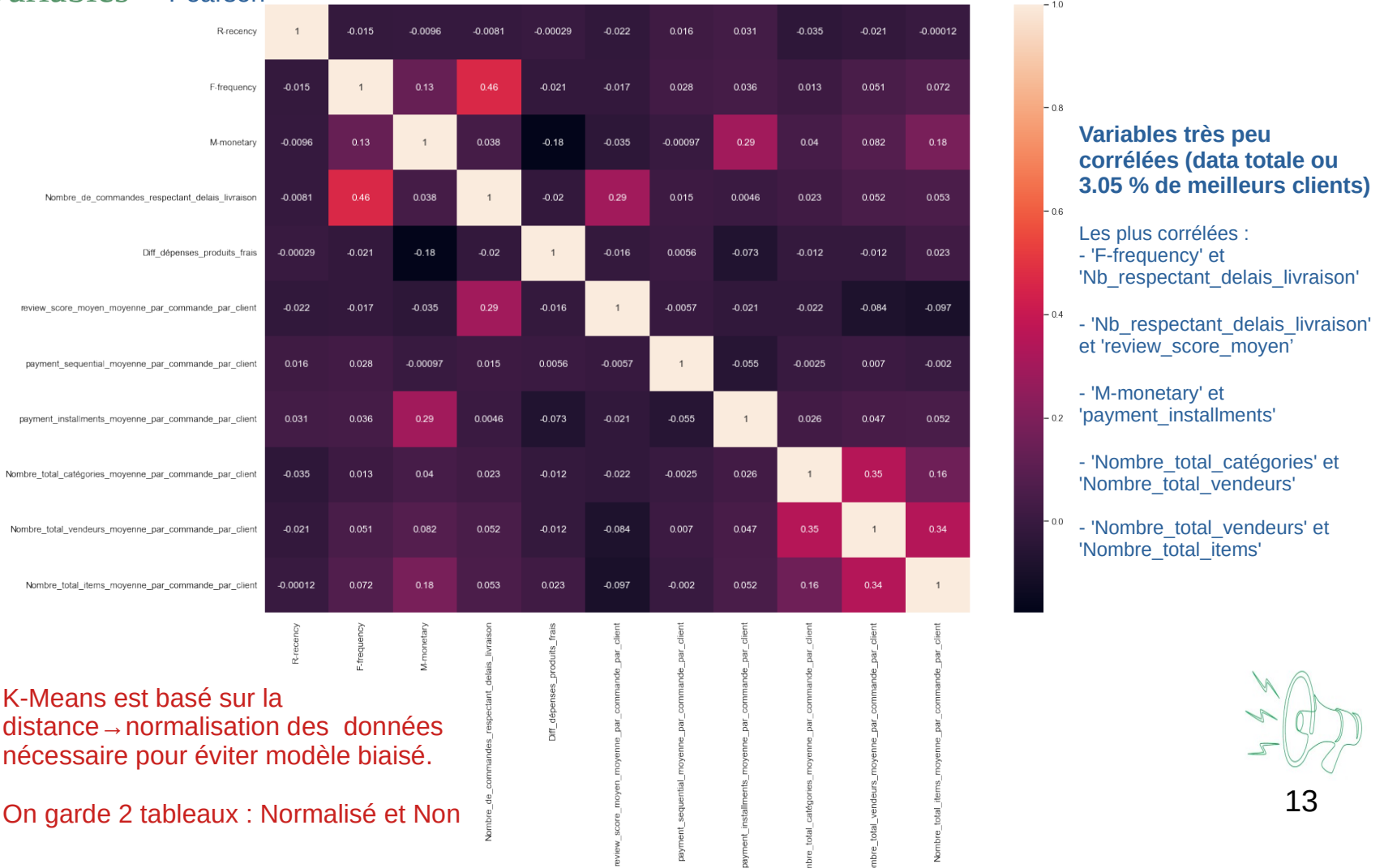






E. Analyse des données

Corrélations entre les 11 variables → Pearson



Variables très peu corrélées (data totale ou 3.05 % de meilleurs clients)

Les plus corrélées :
- 'F-frequency' et 'Nb_respectant_delais_livraison'

- 'Nb_respectant_delais_livraison' et 'review_score_moyen'

- 'M-monetary' et 'payment_installments'

- 'Nombre_total_catégories' et 'Nombre_total_vendeurs'

- 'Nombre_total_vendeurs' et 'Nombre_total_items'

K-Means est basé sur la distance → normalisation des données nécessaire pour éviter modèle biaisé.

On garde 2 tableaux : Normalisé et Non

Clustérisation

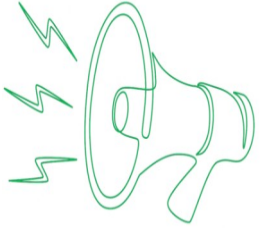
Deux grandes méthodes:

A. **K-means**
(k → avec méthode 'Elbow')

→ basé sur le centroïde partitionne tous les en K groupes de similarité, elle mesurée à l'aide de la distance euclidienne. (Besoin de données normalisées).

B. **DBScan** (Density Based Spatial Clustering of Applications with Noise)
('min_sample' >= D + 1)
('eps' → distance du 'min_sample'ème voisin le plus proche+Elbow avec 'KneeLocator')

→ basé sur la densité. Ici le voisinage de chaque point d'un cluster qui se trouve dans un rayon donné (eps) doit avoir un nombre minimum de points (min_samples). Détecte les valeurs aberrantes et gère le bruit.

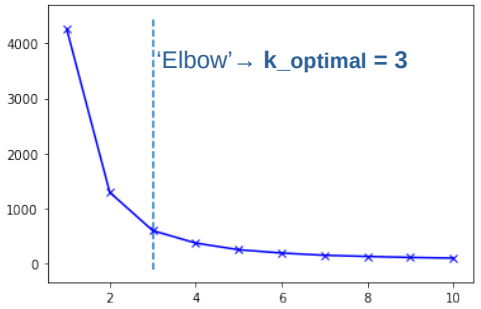


À savoir:

Dans une clustérisation,

- Homogénéité bonne (>=0.8) → les clusters ne contiennent que des points de données qui sont membres d'une seule classe.
- ARI (Adjusted Random Index) bon (>=0.8) → correspondance (un accord) entre deux segmentations (clustering).
- AMI (Adjusted Mutual Information) est bon (>=0.8) si les clusters sont purs.
- Complétude bonne (>=0.8) → tous les points membres d'une classe sont des éléments du même cluster.

A. Méthode 1: **K-means** sur les RFM continues



Cluster	R-recency mean	R-recency min	R-recency max	F-frequency mean	F-frequency min	F-frequency max	M-monetary mean	M-monetary min	M-monetary max	count
0	92.907507	1	176	1.038040	1	16	170.287758	9.59	7274.88	36619
1	464.410379	363	730	1.027055	1	6	165.626966	11.63	7571.63	22584
2	260.635796	177	362	1.034294	1	9	162.082874	10.07	13664.08	36216

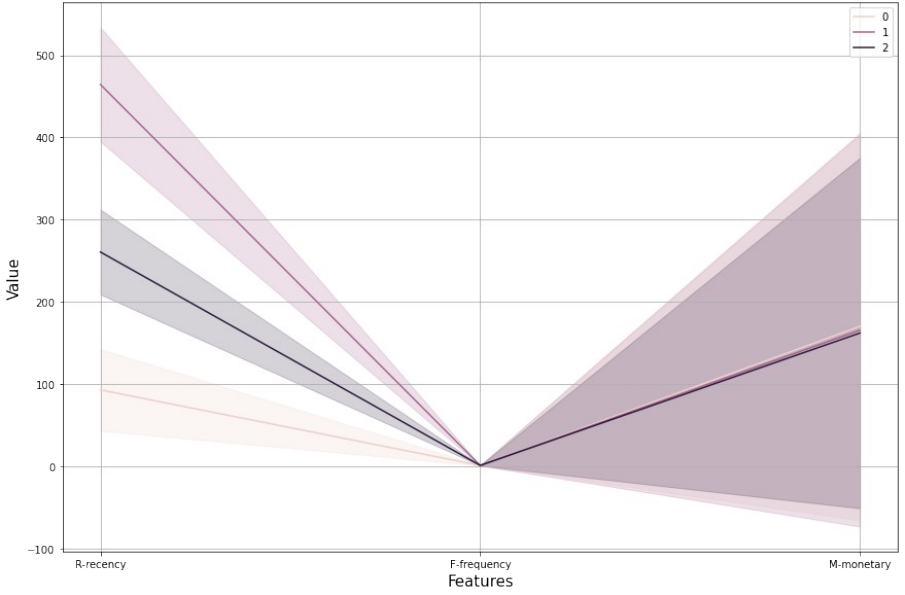
Clusters avec un **nombre d'élément équilibré!**

- **R-recency**: petit → meilleur, client plus satisfait
- **F-freqency**: plus grand → meilleur, client meilleur
- **M-monetary**: plus grand → meilleur → client meilleur

K-Means initialisé avec **K-Means++** → réduction effets aléatoires d'initialisation des centroïdes ?

Iteration	Inertia	Homo	AMI
Iter 0	601	0.96	0.965
Iter 1	601	1.00	1.000
Iter 2	601	0.96	0.965
Iter 3	601	1.00	1.000
Iter 4	601	0.96	0.965
Iter 5	601	1.00	1.000
Iter 6	601	0.96	0.965
Iter 7	601	1.00	1.000
Iter 8	601	0.96	0.965
Iter 9	601	0.96	0.965

Inerties proches, homogénéité et score AMI proches de 1 (donc bons) → **bonne stabilité des centroïdes.**



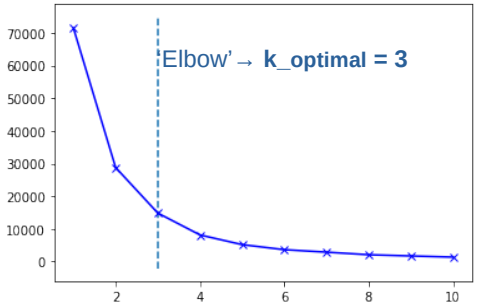
Cluster 0 → Meilleurs: F et M les plus grands (meilleurs clients) et R le plus petit (clients les plus satisfaits).

Cluster 2 → Moyens: R plus grand que cluster 0 mais plus petit que cluster 1 (clients moins satisfait que le 0 mais plus que le 1). F, plus de commandes que le cluster 1 mais moins que le 0. M le plus petit, mais la différence des dépenses moyennes des clusters 2 et 1 est minimale (du même ordre) → effort de marketing à faire pour augmenter le M.

Cluster 1 → Mauvais: R le plus grand R, F le plus petit et M suffisamment mauvais par rapport aux autres.

Sans changement ni amélioration. INUTILE

A. Méthode2: **K-means** sur les RFM scorisées



	R-recency			F-frequency			M-monetary			
Cluster_2	mean	min	max	mean	min	max	mean	min	max	count
0	465.015679	364	730	1.027127	1	6	165.354070	11.63	7571.63	22450
1	261.508343	178	363	1.034314	1	9	162.221815	10.07	13664.08	36137
2	93.393815	1	177	1.037929	1	16	170.283220	9.59	7274.88	36832

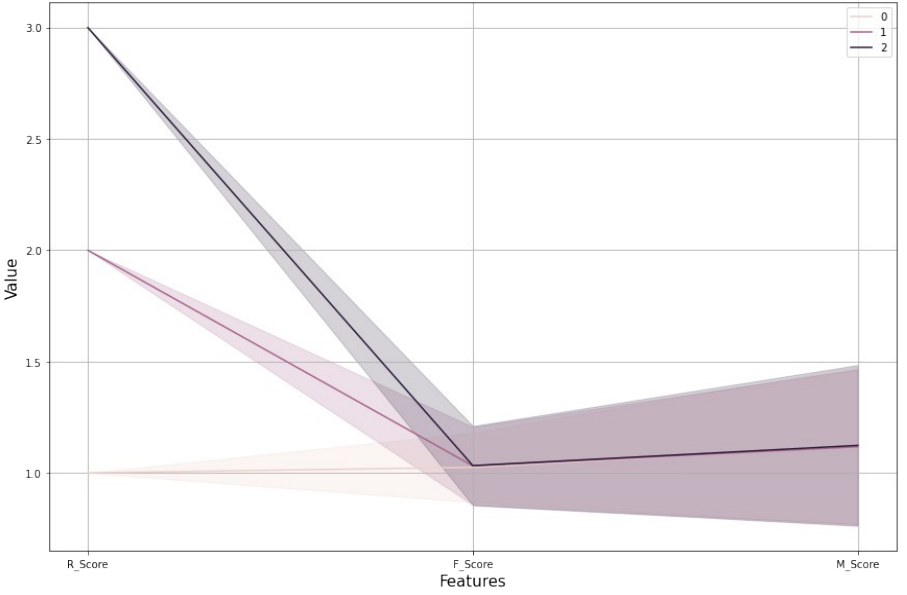
Clusters avec un **nombre d'élément équilibré!**

- **R-recency**: petit → meilleur, client plus satisfait
- **F-freqency**: plus grand → meilleur, client meilleur
- **M-monetary**: plus grand → meilleur → client meilleur

K-Means initialisé avec **K-Means++** → réduction effets aléatoires d'initialisation des centroïdes ?

Iteration	Inertia	Homo	AMI
Iter 0	21517	0.57	0.626
Iter 1	14855	1.00	1.000
Iter 2	14855	1.00	1.000
Iter 3	14855	1.00	1.000
Iter 4	21387	0.60	0.663
Iter 5	14855	1.00	1.000
Iter 6	14855	1.00	1.000
Iter 7	14855	1.00	1.000
Iter 8	14855	1.00	1.000
Iter 9	14855	1.00	1.000

Inerties proches, homogénéité et score AMI proches de 1 (donc bons) → **bonne stabilité des centroïdes.**



Cluster 0 → Meilleur: F et M les plus grands et R le plus petit.

Cluster 2 → Moyen: R plus grand que cluster 0 mais plus petit que cluster 1. F, plus de commandes que le cluster 1 mais moins que le 0. M le plus petit, mais la différence des dépenses moyennes des clusters 2 et 1 est minimale → effort de marketing à faire pour augmenter le M.

Cluster 1 → Mauvais: R le plus grand R, F le plus petit et M suffisamment mauvais par rapport aux autres.

A. Méthode 3: **K-means** sur les RFM continues + Nouvelles variables

Jusque là :

'satisfaction des clients' ('R-recency') → ressort 3 clusters bien différents.
'qualité des clients' ('F-frequency' et 'M-monetary'), → différences entre les moyennes des 3 cluster très proches.
(pour M-monetary les 2 clusters moins bon seraient inversés).

Besoin de nouvelles variables pour renforcer la 'qualité des clients'.

Seules variables non corrélées (hors variables RFM):

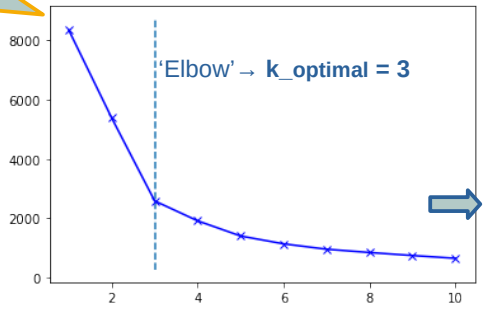
- ✓ 'Diff_dépenses_produits_frais'
- ✗ 'review_score_moyen_moyenne_par_commande_par_client' → satisfaction clients
- ✗ 'payment_sequential_moyenne_par_commande_par_client'
- ✓ 'payment_installments_moyenne_par_commande_par_client' → plus parlante

On reste donc avec les 2 variables:

'payment_installments_moyenne_par_commande_par_client'
et
'Diff_dépenses_produits_frais'.

⚠ Que 2 variables car on a déjà les 3 du RFM et beaucoup de variables fait perdre de la robustesse, 5 c'est correct.

INUTILE



K-Means initialisé avec **K-Means++** → réduction effets aléatoires d'initialisation des centroïdes ?

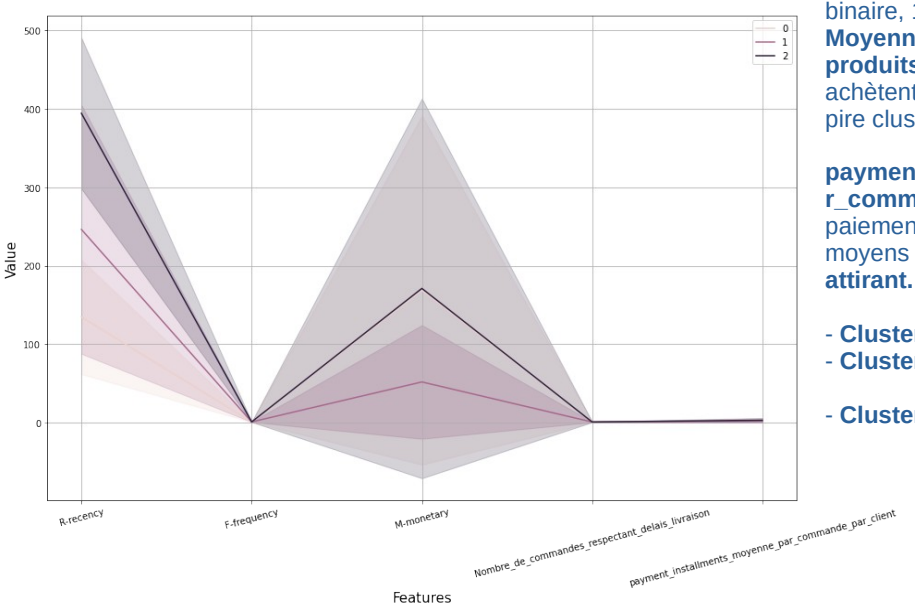
Iteration	Inertia	Homo	AMI
Iter 0	2587	1.00	1.000
Iter 1	2587	1.00	1.000
Iter 2	4668	0.51	0.436
Iter 3	2587	0.98	0.983
Iter 4	2587	0.98	0.983
Iter 5	2587	0.99	0.995
Iter 6	2587	0.97	0.973
Iter 7	2587	0.98	0.983
Iter 8	2587	0.99	0.995
Iter 9	2586	0.99	0.985

Inerties proches, homogénéité et score AMI proches de 1 (donc bons) → **bonne stabilité des centroïdes.**

Clusters avec un **nombre d'élément NON équilibré!**

	R-recency			F-frequency			M-monetary			Diff_dépenses_produits_frais			payment_installments_moyenne_par_commande_par_client		
Cluster_3	mean	min	max	mean	min	max	mean	min	max	count	mean	min	max	mean	min	max
0	134.639386	1	267	1.038182	1	16	168.808947	14.89	7274.88	53323	0.0	0	0	2.788717	0.0	24.0
1	246.207636	6	700	1.011721	1	3	51.930030	9.59	2844.96	2986	1.0	1	1	1.820496	1.0	12.0
2	394.146203	246	730	1.030043	1	6	171.051332	15.57	13664.08	39110	0.0	0	0	3.179760	1.0	24.0

- **R-recency**: petit → meilleur, client plus satisfait
- **F-freqency**: plus grand → meilleur, client meilleur
- **M-monetary**: plus grand → meilleur → client meilleur



RFM moins claires.

Diff_dépenses_produits_frais: binaire, 1 si frais > coûts produits. **Moyenne ~1 → frais > coûts produits → clients mécontents** achètent moins. Se confirme pour le pire cluster (1).

payment_installments_moyenne_par_commande_par_client: plus de paiements différés plus les clients sont moyens voir bons → **paiement différés attirant.**

- **Cluster 0 → Meilleurs**
- **Cluster 2 → Moyens** (efforts de marketing)
- **Cluster 1 → Mauvais**

B. Méthode 4: **DBScan** sur les RFM continues avec les 'min_sample' et 'eps' optimales

min_sample = 5, car 3 variables RFM (elles s'avèrent les meilleures) → 3 dimensions → optimales

eps = 0.0165 →

```
kneedle.knee_y
0.016461409992497143
```

	R-recency			F-frequency			M-monetary			
	mean	min	max	mean	min	max	mean	min	max	count
Cluster_DBScan										
-1	289.955017	12	730	2.321799	1	16	1784.103806	46.86	13664.08	289
0	243.453501	6	607	1.000000	1	1	157.179740	9.59	2713.36	92066
1	697.503448	694	702	1.000000	1	1	175.615172	18.62	982.41	290
2	227.681120	1	607	2.000000	2	2	262.197732	34.97	1432.21	2606
3	330.222222	321	343	3.000000	3	3	479.023333	383.99	588.71	9
4	444.600000	440	453	1.000000	1	1	2028.648000	1938.74	2067.42	10
5	398.200000	393	405	3.000000	3	3	244.826000	152.41	342.75	5
6	30.185185	7	49	3.000000	3	3	385.650370	88.27	806.63	27
7	89.437500	72	105	3.000000	3	3	294.023125	127.65	480.52	16
8	161.166667	117	202	3.000000	3	3	249.450278	126.33	486.23	36
9	227.428571	203	252	3.000000	3	3	288.788571	87.60	532.44	14
10	368.500000	355	379	3.000000	3	3	225.066250	139.60	335.15	8
11	122.000000	112	132	3.000000	3	3	591.817778	421.32	725.80	9
12	452.800000	447	457	3.000000	3	3	264.492000	126.33	358.89	5
13	485.111111	475	496	1.000000	1	1	1881.928889	1679.63	2150.81	9
14	109.000000	95	119	1.000000	1	1	2614.465714	2467.17	2759.95	7
15	526.400000	519	540	1.000000	1	1	2164.544000	2027.16	2324.99	5
16	189.000000	184	194	1.000000	1	1	2149.620000	2028.65	2276.10	4
17	235.500000	231	240	1.000000	1	1	2119.340000	2060.20	2204.04	4

noise points

Clusters avec un **nombre d'élément NON équilibré!** → En plus, trop de clusters nuit à l'identification des clients: Bon, Mauvais et Moyens

B. Méthode 5: **DBScan** sur les RFM continues avec valeur optimale de 'min_sample' et 'eps' plus grand

min_sample = 5, car 3 variables RFM (elles s'avèrent les meilleures) → 3 dimensions ⟶ optimale

eps = 0.1

	R-recency			F-frequency			M-monetary			
	mean	min	max	mean	min	max	mean	min	max	count
Cluster_DBScan_2										
-1	276.166667	15	607	4.166667	1	16	5247.554167	110.72	13664.08	12
0	697.701342	694	730	1.010067	1	2	188.482987	18.62	1423.55	298
1	243.072485	1	607	1.033698	1	7	165.359131	9.59	4809.44	95109

→ noise points

Clusters avec un **nombre d'élément NON équilibré!**

Un **'eps' plus grand** → **réduit** le nombre de clusters à 2.

Le cluster 0: clients avec R grands (**bas en satisfaction**), et **achetant peu** (F bas). Néanmoins c'est eux qui **dépensent le plus** (M le plus grand) → **effort de marketing à faire**.

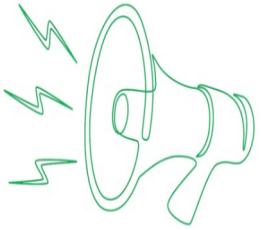
Les **noice points** représentent le **plus grand capital dépensé**, mauvais signe → c'est comme si cette segmentation ne les considérais pas.

⇩

DBScan NON adéquat pour la segmentation de nos données.

Meilleure segmentation (par rapport aux besoins et information tirées):
'K-Means' sur uniquement les variables RFM non scorées.

Simulation :
fréquence de mise à jour



A. Questions et besoins

Les **intervalles** de temps considérés doivent être **'courts mais pas trop'** → robustesse des résultats.

Période: de septembre 2016 à septembre 2018 → l'**analyse prédictions toutes les semaines adéquat**.
On a comme max 729/7=104.14 semaines (ie. **un peu plus de 104 semaines**).

Entraînement sur la base entière et à reculons pour les prédictions (.predict) → on enlèvera à chaque prédiction les semaines précédentes (lecture à faire à l'envers: 103 semaines → 1ère semaine)

Processus: On clustérise la base entière (obtention 'Cluster1' et du 'Modèle'), puis on fait un '.predict' du 'Modèle' obtenu sur une base avec x semaines en moins (obtention 'Cluster2'). Finalement on compare 'Cluster1' et 'Cluster2' en calculant leur ARI (>=0.8 bon → on garde, inférieur on change).

IMPORTANT:
Les RFM dépendent de la base (dernier achat par client, dernier achat de la base, somme dépensée durant la période), si on enlève des semaines les RFM changent.

À chaque 'semaine enlevée' on a une 'nouvelle base' avec le gros de la base conservé mais aussi avec en quelque sorte 'des nouveaux clients' (car ils auront de nouveaux RFM associés → un bon client semaine x peut être mauvais semaine x-1)

La comparaison adéquate:
"Cluster1" -> cluster résultant du modèle sur base complète,
avec
"Cluster2" -> cluster résultant du .predict du modèle calculé sur base complète,
mais appliqué sur la base sans x semaines.

B. Automatisation

```
#Automatisation
```

```
print("ARI entre les clusters")
print('Semaine   ARI')
print(53 * '_')


for i in range(103):
    #C.1.1 On commence par "loc" la base.
    df_t2=df_t.loc[(df_t['diff_jours'] >7*(i+1))]

    #C.1.2 On calcule les nouvelles variable RFM pour cette nouvelle base.
    #'R-recency'
    df_t2['Date_du_dernier_achat_par_client']=df_t2.groupby('customer_unique_id')['order_purchase_timestamp_NEW'].transform(max)
    df_t2['Dataset_last_day']=df_t2['order_purchase_timestamp_NEW'].max()
    df_t2['R-recency']= ((df_t2['Dataset_last_day']+timedelta(days=1))-df_t2['Date_du_dernier_achat_par_client']).dt.days

    df_t2=df_t2.drop(['F-frequency', 'M-monetary'], axis=1)

    #'F-frequency'
    F=df_t2.groupby(['customer_unique_id'], as_index=False)['order_id'].count()
    F.rename(columns={'order_id': 'F-frequency'}, inplace=True)
    df_t2= pd.merge(df_t2, F, how='left', on='customer_unique_id')

    #'M-monetary'
    M=df_t2.loc[:,['customer_unique_id', 'payment_value']]
    M=M.groupby(['customer_unique_id'], as_index=False).sum()
    M.rename(columns={'payment_value': 'M-monetary'}, inplace=True)
    df_t2= pd.merge(df_t2, M, how='left', on='customer_unique_id')

    #C.1.3 On calcule les prédictions.
    RFM_2=df_t2.loc[:,['R-recency','F-frequency','M-monetary']]
    prediction_1=model.predict(RFM_2)
    df_t2['Cluster2']=prediction_1

    name="-"+str(i+1)+" sem."
    res= adjusted_rand_score(df_t2['Cluster1'], df_t2['Cluster2'])
    results = [name, res]

    print(*results)
```

B. Résultats

ARI entre les clusters

Semaine	ARI
-1 sem.	-0.848902507279809
-2 sem.	0.24394310309047043
-3 sem.	-1.227966482905113
-4 sem.	-7.719263961512608
-5 sem.	7.058475749745104
-6 sem.	3.7961283896776417
-7 sem.	-2.524565396977901
-8 sem.	-1.8912713560534609
-9 sem.	1.5798088844895195
-10 sem.	1.718568244906802
-11 sem.	1.9816521949658072
-12 sem.	2.532770972434046
-13 sem.	3.858327318347242
-14 sem.	7.772287367515303
-15 sem.	-52.218558468423915
-16 sem.	-2.533727863668042
-17 sem.	-1.026258909720536
-18 sem.	-0.5775910801581022
-19 sem.	-0.36201731293190276
-20 sem.	-0.2357300603532109
-21 sem.	-0.15814268801752185
-22 sem.	0.10661076713054417
-23 sem.	0.08825635733480126
-24 sem.	0.07196614900139046
-25 sem.	0.059139123286636334
-26 sem.	0.047578724310934906
-27 sem.	0.038617414089089475
-28 sem.	0.03470018567602584
-29 sem.	0.035414319160328965
-30 sem.	0.04153651833867275
-31 sem.	0.054600785805710886
-32 sem.	0.07462711735347273
-33 sem.	0.10505806262341326
-34 sem.	0.14683801808680214
-35 sem.	0.19166823178281178
-36 sem.	0.2383520230194828
-37 sem.	0.301911236375927
-38 sem.	0.3574784731074652
-39 sem.	0.38378287196172445
-40 sem.	0.4131478656930138
-41 sem.	0.4417676708967072
-42 sem.	0.4718180009060616
-43 sem.	0.5020227660071716
-44 sem.	0.5303617213063775
-45 sem.	0.6371577042013974
-46 sem.	0.7115324145940746
-47 sem.	0.7749352680903688
-48 sem.	0.8122570964973386
-49 sem.	0.821808295734641
-50 sem.	0.8270257625769715
-51 sem.	0.8202109988090317
-52 sem.	0.8142490694284393
-53 sem.	0.8022851340451178
-54 sem.	0.7972216046608024
-55 sem.	0.7894701073557282
-56 sem.	0.782937332510906
-57 sem.	0.7780655036044573
-58 sem.	0.7720072438576676
-59 sem.	0.7660051300245877
-60 sem.	0.7592526929716268
-61 sem.	0.7566626620942646
-62 sem.	0.877885948247303
-63 sem.	0.9614992041605783
-64 sem.	0.9593429657062506
-65 sem.	0.9570714026102687
-66 sem.	0.9580505186778132
-67 sem.	0.9592904781439892
-68 sem.	0.9586385203228437
-69 sem.	0.9579306556952029
-70 sem.	0.9580931408006497
-71 sem.	0.9574670142230222
-72 sem.	0.9604665025102214
-73 sem.	0.9627622245085424
-74 sem.	0.9662207279147335
-75 sem.	0.9627539152730259
-76 sem.	0.9616962055473558
-77 sem.	0.9561875242920067
-78 sem.	0.9529838888738561
-79 sem.	0.9437869867603009
-80 sem.	0.9367049290327804
-81 sem.	0.9236238159808668
-82 sem.	0.9285458636540475
-83 sem.	0.9053723192700329
-84 sem.	0.8715698936552548
-85 sem.	0.8486340259197677
-86 sem.	0.8247459541298165
-87 sem.	0.8623375135472748
-88 sem.	0.8623375135472748
-89 sem.	0.8623375135472748
-90 sem.	0.8623375135472748
-91 sem.	0.8623375135472748
-92 sem.	0.8623375135472748
-93 sem.	0.8623375135472748
-94 sem.	0.8623375135472748
-95 sem.	0.8623375135472748
-96 sem.	0.8623375135472748
-97 sem.	0.8623375135472748
-98 sem.	0.8623375135472748
-99 sem.	0.8521709940851102
-100 sem.	1.0
-101 sem.	1.0
-102 sem.	1.0
-103 sem.	1.0

(lecture à faire à l'envers: 103 semaines → 1ère semaine)

À partir de 61 semaines (en lecture à l'envers) le ARI est >0.8,
soit au bout de 43 semaines, soit tous les 10,75 mois.

En gros tous les 10-11 mois (plutôt **10,5 mois** pour être prévoyants).

MERCI