DATA SCIENTIST - PROJET 5

Segmentez des clients d'un site e-commerce

Arnaud CHOUX

Formation OPENCLASSROOMS
Financement région IdF

Novembre 2022

Plan

Plan

I.1. Objectifs

1. Établir un clustering raisonnable (3 à 6 clusters de plus de 100 clients chacun et présentant des caractéristiques différentes) d'un dataset de clients d'une pateforme web.

2. Identifier les caractéristiques de chaque cluster et proposer une action commerciale adéquate sur chacun.

3. Prédire la durée de validité du modèle calculé au bout de laquelle il faut mettre à jour la modélisation.

I.2. Compétences abordées

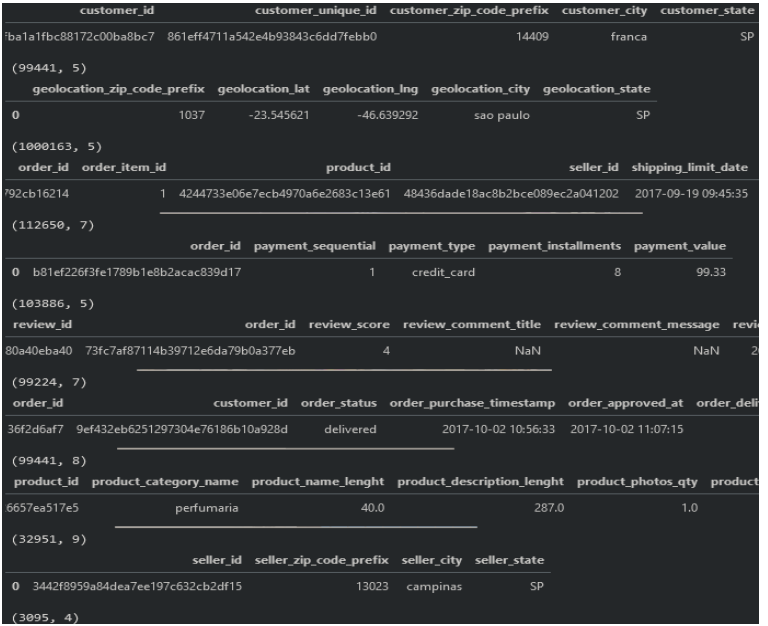
- collecter la data (sans phase de recherche car la source est donnée)

- mettre en forme et nettoyer la data (RDBMS $\rightarrow$ RFM)

- appliquer un clustering et évaluer sa performance

- évaluer la similarité entre deux résultats de clustering

I.3. Dataset



customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
²ba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP

(99441, 5)

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	1037	-23.545621	-46.639292	sao paulo	SP

(1000163, 5)

order_id	order_item_id	product_id	seller_id	shipping_limit_date
²92cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35

(112650, 7)

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33

(103886, 5)

review_id	order_id	review_score	review_comment_title	review_comment_message	revi
80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	NaN	2

(99224, 7)

order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_deli
36f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	

(99441, 8)

product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product
6657ea517e5	perfumaria	40.0	287.0	1.0	

(32951, 9)

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP

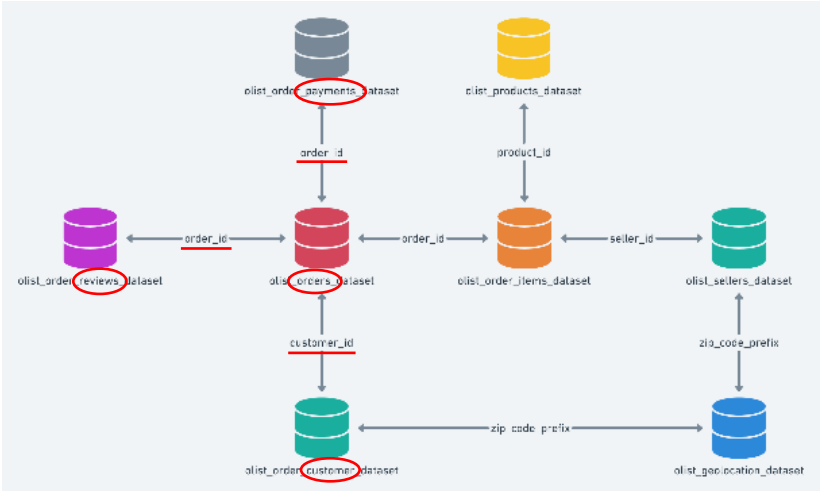
(3095, 4)

→ RDBMS de 9 csv.

Principales keys pour ce problème: customer_unique_id, order_id

Plan

II.1. Merging



	customer_unique_id	order_purchase_timestamp	payment_value	review_score
0	861eff4711a542e4b93843c6dd7febb0	2017-05-16 15:05:35	146.87	4.0
1	290c77bc529b7ac935b93aa66c333dc3	2018-01-12 20:48:24	335.48	5.0

→ Je forme un dataset ne gardant que les features pertinentes du RDBMS pour mon étude, *via* merging (joining).

II.2. Nettoyage

```
1  df.isna().sum()
```

```
customer_unique_id          0
order_purchase_timestamp    0
payment_value               0
review_score                0
```

→ Je supprime les doublons (facilement repérables car ils ont une NaN).

II.3. Réorganisation de la data en RFM(S)

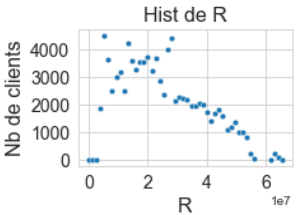
	customer_unique_id	order_purchase_timestamp	payment_value	review_score
0	861eff4711a542e4b93843c6dd7febb0	2017-05-16 15:05:35	146.87	4.0
1	290c77bc529b7ac935b93aa66c333dc3	2018-01-12 20:48:24	335.48	5.0

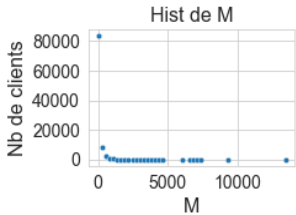
```
df["recency_2"] = pd.to_numeric(pd.to_datetime(df.order_purchase_timestamp))
df["recency_2"] = (df.recency_2.max() - df.recency_2)*10**-9
dfs["R"] = df.groupby("customer_unique_id").recency_2.min()
dfs["F"] = df.groupby(
    ["customer_unique_id","order_purchase_timestamp"]
).agg("count").groupby("customer_unique_id").count().payment_value
dfs["M"] = df.groupby("customer_unique_id").payment_value.sum()
dfs["rs"] = df.groupby("customer_unique_id").review_score.min()
dfs.head()
```

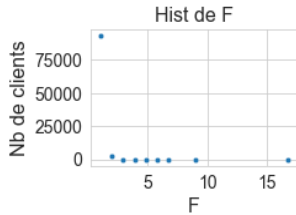
customer_unique_id	R	F	M	rs
66f3b9a7992bf8c76cfdf3221e2	13847631.0	1	141.90	5.0
249f77a49e4a4ce2b2a4ca5be3f	14105931.0	1	27.19	4.0
5a3911fa3c0805444483337064	50617515.0	1	86.22	3.0
ccb0745a6a4b88665a16c9f078	31957237.0	1	43.62	4.0
ac84e0df4da2b147fca70cf8255	29108676.0	1	196.89	5.0

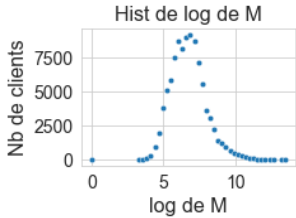
→ En groupant les commandes par client je tire les caractéristiques de chacun.

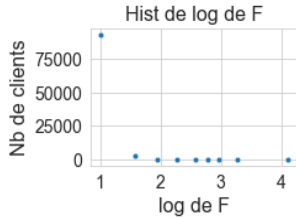
II.4. Visualisation







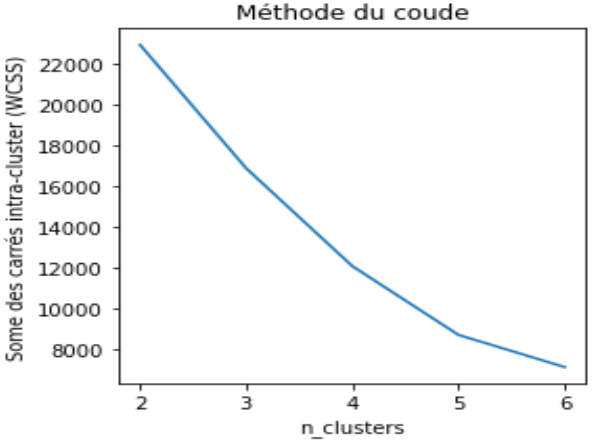




→ Je trouve que seuls 2.8% des clients présents dans cette database ont fait plus d'un achat.

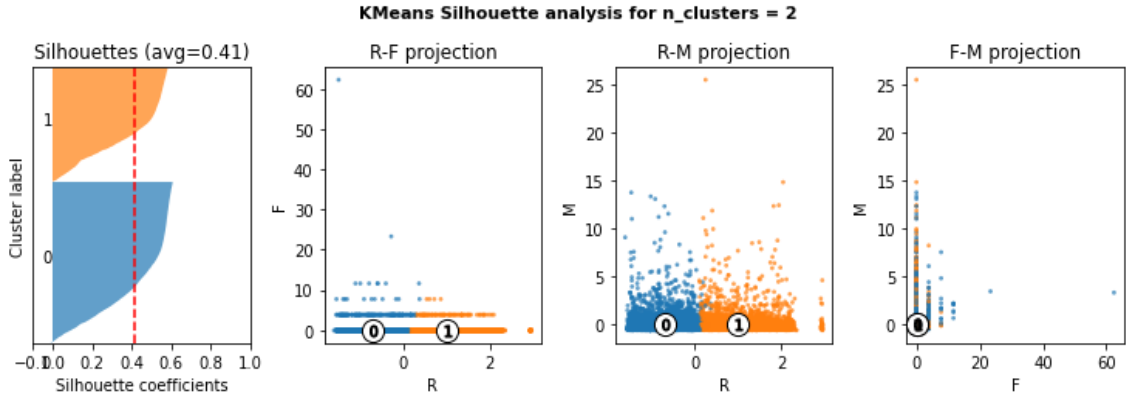
Plan

III.1.1. KMeans elbow (RFM)



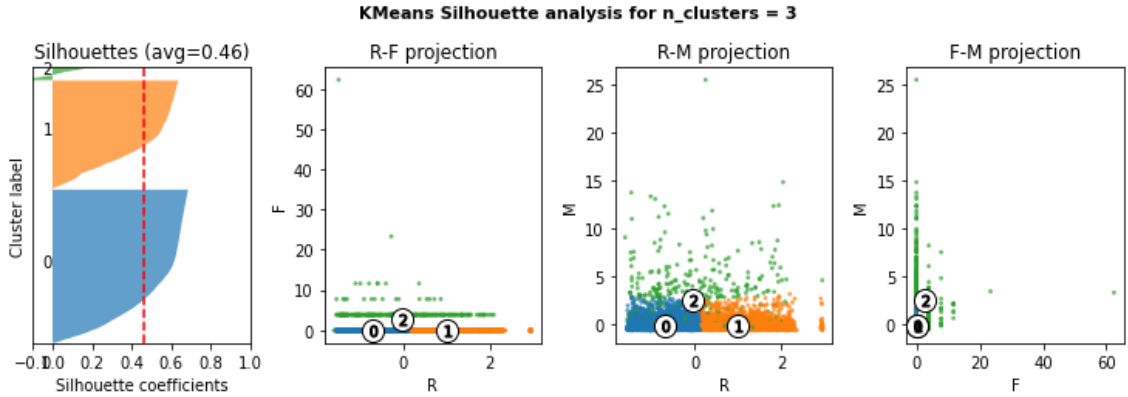
→ optimum à 4 ou 5 clusters

III.1.2.a. KMeans silhouette (RFM, n_clusters=2)



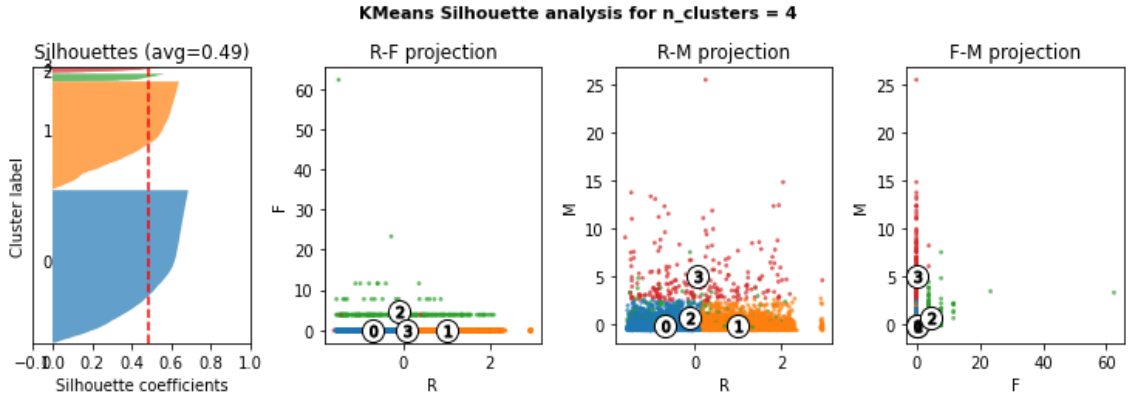
→ clustering sur R uniquement

III.1.2.b. KMeans silhouette (RFM, n_clusters=3)



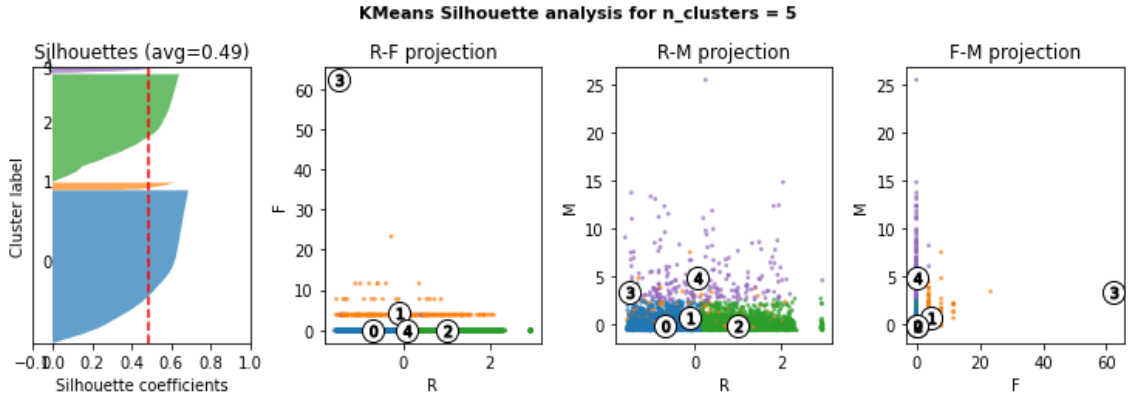
→ clustering sur R, et sur (F+M)

III.1.2.c. KMeans silhouette (RFM, n_clusters=4)



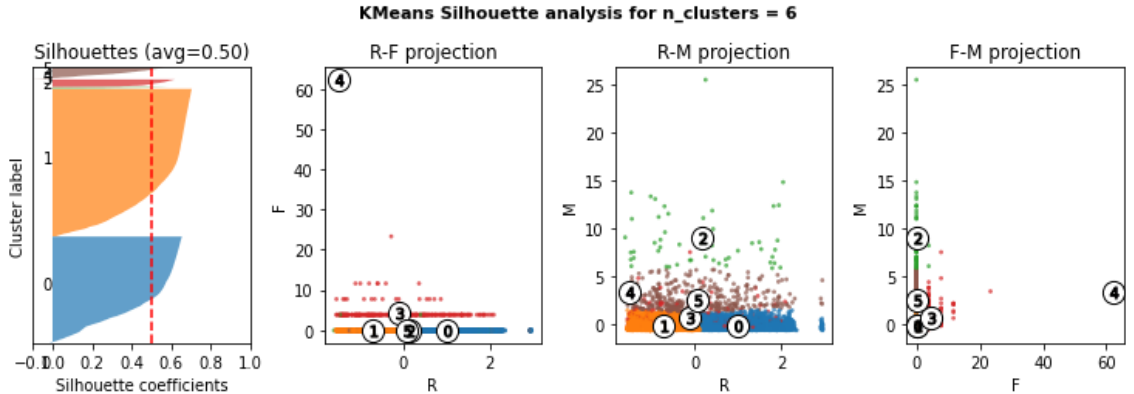
→ clustering sur R, F et M, c'est parfait.

III.1.2.d. KMeans silhouette (RFM, n_clusters=5)



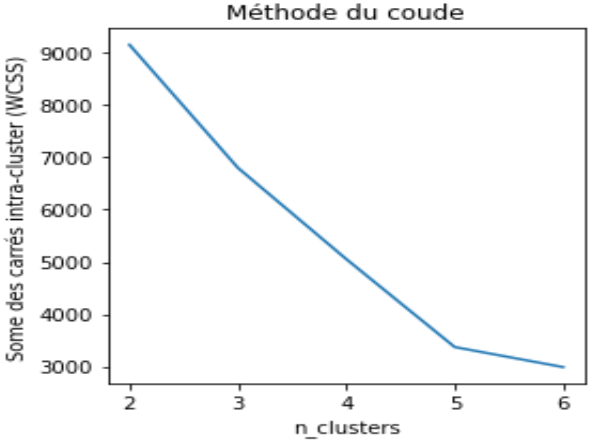
→ cluster 3 négligeable

III.1.2.e. KMeans silhouette (RFM, n_clusters=6)



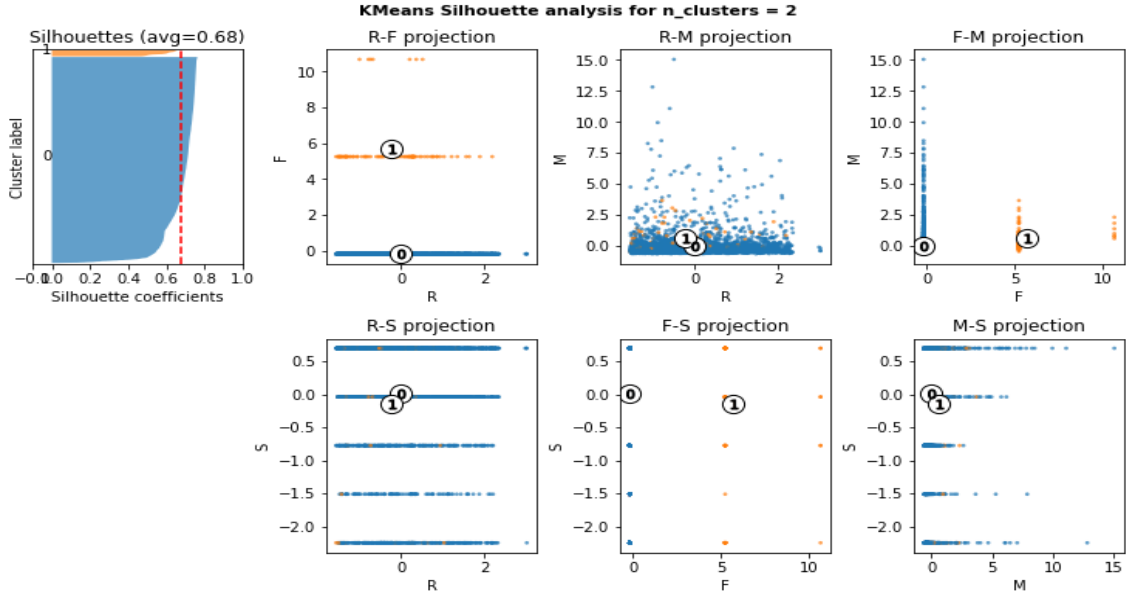
→ cluster '3' négligeable, mais silhouette maximale

III.1.3. KMeans elbow (RFMS)



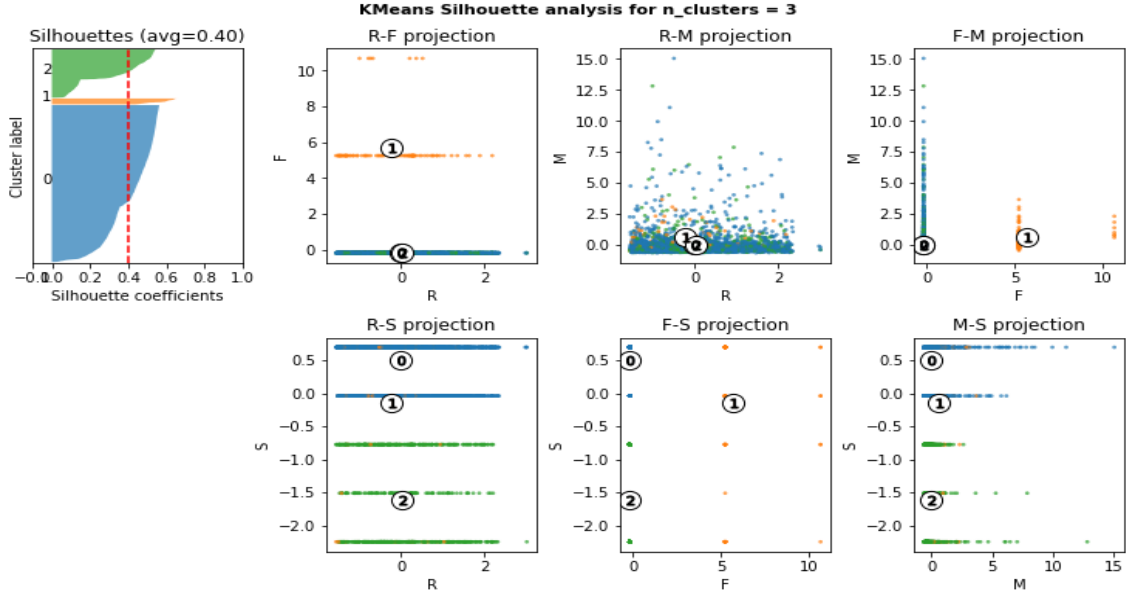
→ optimum à 4 ou 5 clusters

III.1.4.a. KMeans silhouette (RFM, n_clusters=2)



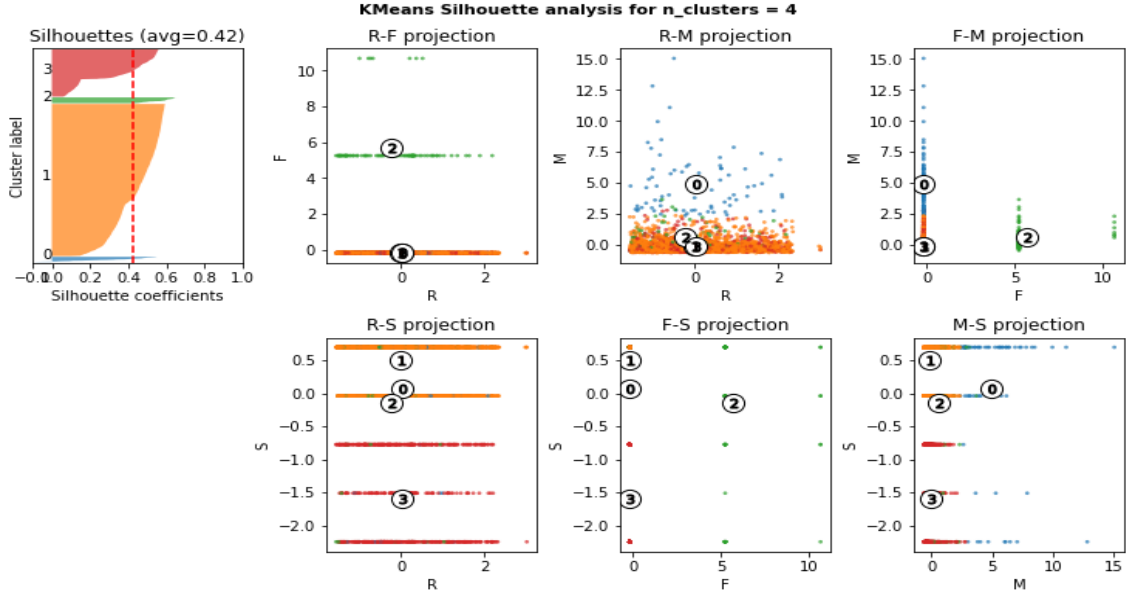
→ clustering sur F uniquement

III.1.4.b. KMeans silhouette (RFM, n_clusters=3)



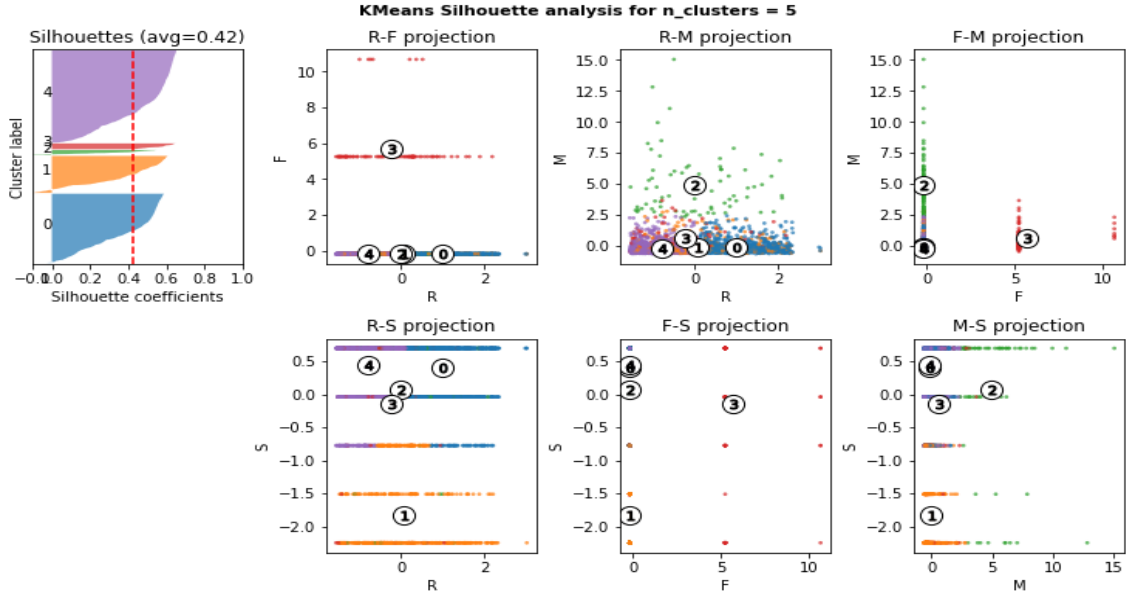
→ clustering sur F et S uniquement

III.1.4.c. KMeans silhouette (RFM, n_clusters=4)



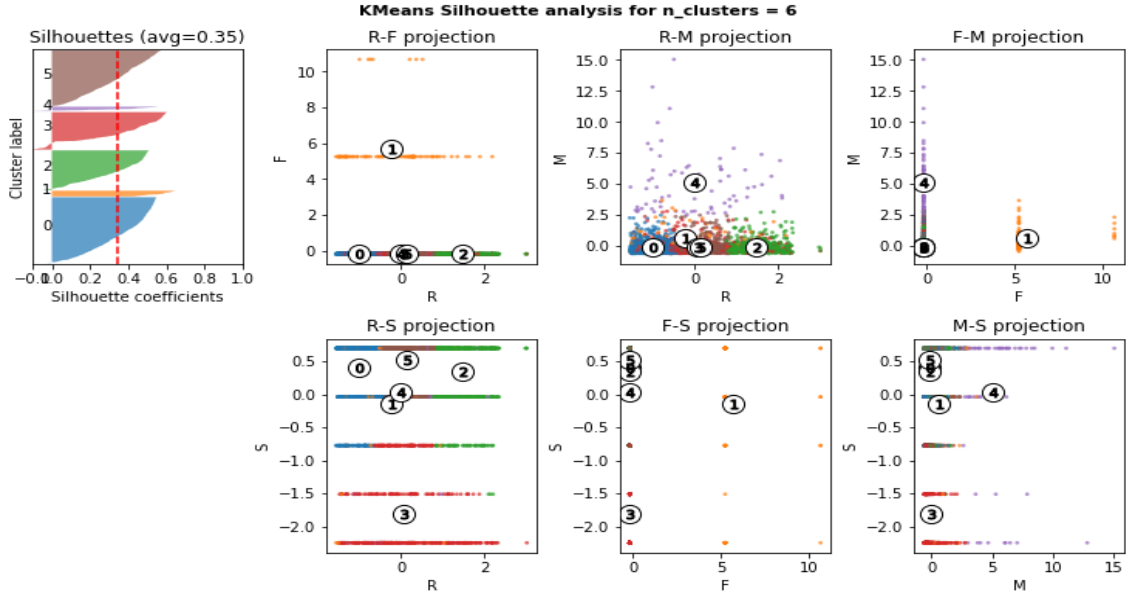
→ clustering sur F, M et S uniquement

III.1.4.d. KMeans silhouette (RFM, n_clusters=5)



→ clustering sur R, F, M et S, c'est parfait

III.1.4.e. KMeans silhouette (RFM, n_clusters=6)



→ clustering sur R, F, M et S, c'est parfait

III.2. Analyse des clusters obtenus

label	R	F	M	S	count
0	2.480367e+07	1.019753	1287.599783	3.945679	2025
1	2.285980e+07	2.114200	308.853573	3.723892	2662
2	1.480017e+07	1.000000	133.810101	4.668549	42139
3	3.822613e+07	1.000000	135.004696	4.631493	31766
4	2.529335e+07	1.000000	153.387360	1.588729	16787

Le KMeans trouve les clusters suivants:

- S bas (1.6/5), RFM moyens. -> client déçu. -> Excuses et bon d'achat.
- S haut, achat récent, FM moyens. -> nouveau client -> pas d'action.
- S haut, achat daté, FM moyens. -> client satisfait mais perdu -> news/discount.
- M haut, RFS moyens. -> un seul gros achat -> news.
- F haut, RMS moyens. -> client fréquent -> proposition d'abonnement VIP.

III.3. Stabilité (reproductibilité) du clustering KMeans

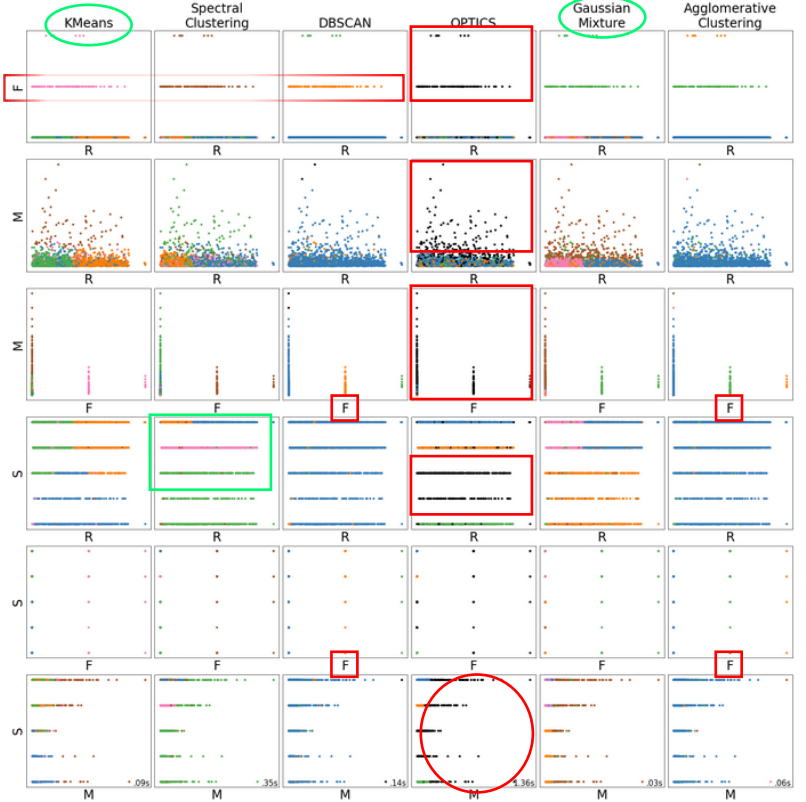
```
7  print(pd.Series(ARIs).mean(), pd.Series(ARIs).std())
```

```
0.9998779504580605 0.00012521339799416l2
```

Excellent résultat ! Le KMeans à 5 clusters trouve toujours les mêmes clusters malgré le init="random".

→ Avec n_clusters=5, init='random', je répète 5 fois le clustering.

III.4.


KMeans
Spectral Clustering
DBSCAN
OPTICS
Gaussian Mixture
Agglomerative Clustering
F
M
S
R
F
M
S
M
.09s
.35s
.14s
1.36s
.03s
.06s

Plan

IV.1. Calcul de la stabilité temporelle

1. Fabriquer des datasets sur une plage temporelle croissante. (t0 – (1an + n mois))

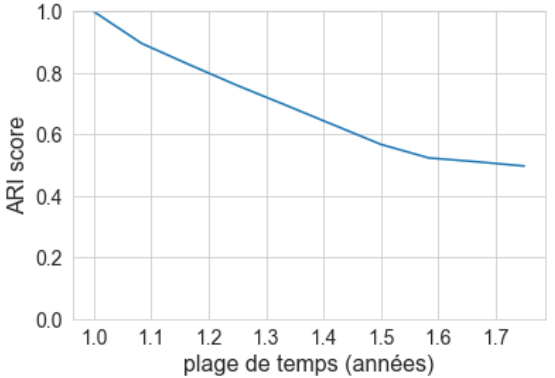
2. Fit un KMeans sur le dataset 0, l'utiliser pour predict sur chaque dataset.

3. Fit_predict n algorithmes KMeans (un par dataset).

4. Comparer (par calcul d'Ajusted Rand Index score) deux à deux les prédictions.

5. Représenter temporellement l'évolution du désaccord entre les prédictions.

IV.2. Stabilité temporelle du clustering KMeans



→ Le clustering reste valide (ARI $>$ .8) pendant 2 mois et 12 jours (10 semaines).

CCL

- Un clustering KMeans optimal a été obtenu.
- Gaussian Mixture et Spectral Clustering ont l'air également adéquats ici.
- Le délai de maintenance du clustering délivré a été estimé.

Questions ?