SEGMENTER DES CLIENTS D'UN SITE E-COMMERCE

MARWA EL HOURI

MISSION

- Olist (entreprise brésilienne de vente en ligne) souhaite faire une segmentation des clients pour leurs campagnes de communication
- Objectif:
  - Classifier les différents types d'utilisateurs
  - Proposer un contrat de maintenance

PLAN

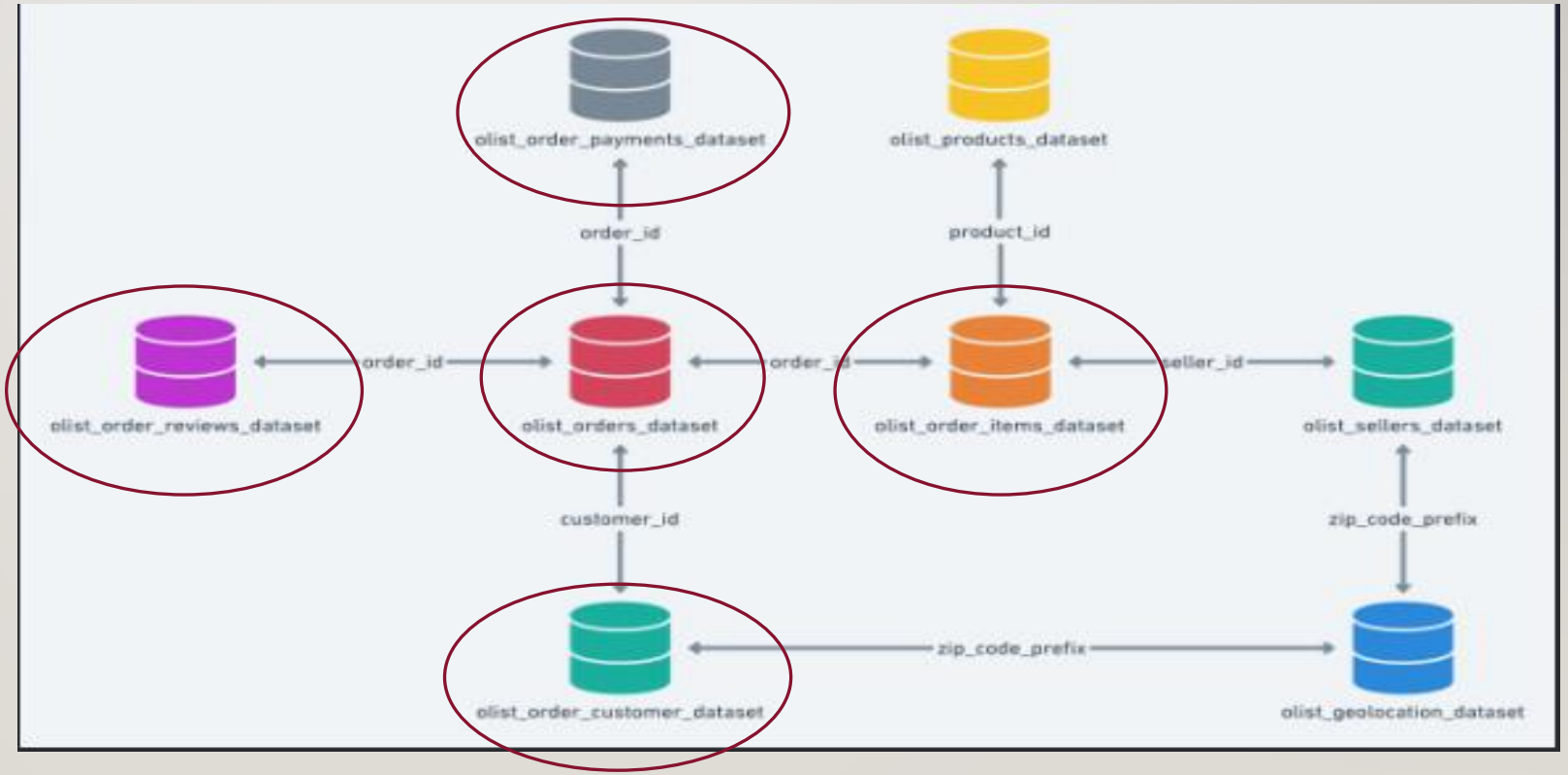
1. Exploration des données et feature engeneering
2. Méthodes de classifications
3. Contrat de maintenance

1- EXPLORATION DES DONNÉES

- Description du jeu de donnée
- Préparation des données (Feature engeneering)
- Analyse exploratoire

1.1 - DESCRIPTION DU JEU DE DONNÉE

- 9 jeux de données
- 5 jeux de données potentiellement intéressants



1.1 - DESCRIPTION DU JEU DE DONNÉE

- Jeux de donnees a explorer
  - Olist_orders_dataset : Information sur les dates de commande et de livraison
  - Oliste_order_custumer_dataset : Information sur les clients (récurrents ou non grâce a customer_unique_id)
  - Olist_order_payments_dataset : Information sur le paiement
  - Olist_order_reviews_dataset: Information sur les reviews
  - Olist_order_itemes_dataset : Information sur le nombre d'articles par commande

1.2 - PRÉPARATION DES DONNÉES (FEATURE ENGENEERING)

- Identifier les informations potentiellement importantes pour la segmentation
  - **Recency** : La récence de la dernière commande effectuée
  - **Frequency** : Le nombre de commandes effectué par un même client
  - **Monetary** : Le montant total dépensé par un même client
  - **Payment installement** : Le nombre de paiement moyen effectué pour l'achat par un même client
  - **Review score** : Le score moyen donné par un même client
  - **Nbre of items** : Le nombre total d'article acheté

1.2 - PRÉPARATION DES DONNÉES RECENCY

- Olist_orders_dataset

```
df_orders.shape
(99441, 8)
```

	order_id	customer_id	order_status	order_purchase_timestamp	c
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	

- Conversion de la variable « order_purchase_timestamp en datetime

```
from datetime import timedelta as td
df_orders['order_purchase_timestamp'] = pd.to_datetime(
    df_orders['order_purchase_timestamp'], format="%Y-%m-%d %H:%M:%S").dt.date
```

1.2 - PRÉPARATION DES DONNÉES FREQUENCY

- olist_customers_dataset

	customer_id	customer_unique_id
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066

- Un peu plus de 3% de clients récurrents

```
len(df_customers['customer_id'].unique())
```

```
99441
```

```
len(df_customers['customer_unique_id'].unique())
```

```
96096
```

1.2 - PRÉPARATION DES DONNÉES REVIEW SCORE

- Olist_order_reviews_dataset

	review_id	order_id	review_score
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5

- Enlever les duplicatats

```
df_review[df_review['order_id'] == 'c88b1d1b157a9999ce368f218a407141']
```

	review_id	order_id	review_score
1985	ffb8cff872a625632ac983eb1f88843c	c88b1d1b157a9999ce368f218a407141	3
82525	202b5f44d09cd3cfc0d6bd12f01b044c	c88b1d1b157a9999ce368f218a407141	5
89360	fb96ea2ef8cce1c888f4d45c8e22b793	c88b1d1b157a9999ce368f218a407141	5

	order_id	order_item_id	product_id
88316	c88b1d1b157a9999ce368f218a407141	1	b1acb7e8152c90c9619897753a75c973

```
df_review = df_review.drop_duplicates(subset='order_id')
df_review.shape
```

```
(98673, 7)
```

1.2 - PRÉPARATION DES DONNÉES MONETARY ET PAYMENT INSTALLEMENTS -

- olist_order_payments_dataset

	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
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

```
df_order_payments['order_id'].value_counts()
```

```
fa65dad1b0e818e3ccc5cb0e39231352    29
ccf804e764ed5650cd8759557269dc13    26
285c2e15bebd4ac83635ccc563dc71f4    22
895ab968e7bb0d5659d16cd74cd1650c    21
fedcd9f7ccdc8cba3a18defedd1a5547    19
                                    ..
6d2a30c9b7dcee3ed507dc9a601f99e7     1
a7737f6d9208dd56ea498a322ed3c37f     1
646e62df54f3e236eb6d5ff3b31429b8     1
e115da7a49ec2acf622e1f31da65cfb9     1
28bbae6599b09d39ca406b747b6632b1     1
Name: order_id, Length: 99440, dtype: int64
```

```
df_order_payments['payment_installments'].describe()
```

```
count    103886.000000
mean          2.853349
std           2.687051
min           0.000000
25%           1.000000
50%           1.000000
75%           4.000000
max          24.000000
Name: payment_installments, dtype: float64
```

- Agrégation sur le montant total de chaque commande et le nombre de paiement

```
payment = df_order_payments.groupby('order_id').agg({'payment_value': lambda x: x.sum(
), 'payment_installments': lambda x: x.max()})
```

order_id	payment_value	payment_installments
00010242fe8c5a6d1ba2dd792cb16214	72.19	2
00018f77f2f0320c557190d7a144bdd3	259.83	3
000229ec398224ef6ca0657da4fc703e	216.87	5
00024acbcdf0a6daa1e931b038114c75	25.78	2
00042b26cf59d7ce69dfabb4e55b4fd9	218.04	3

1.2 - PRÉPARATION DES DONNÉES
NBRE OF ITEMS -

Agrégation des données sur le nombre maximal dans nbr_items

- olist_order_items_dataset

```
nbre_items = df_order_items.groupby('order_id').agg(
    {'order_item_id': lambda x: x.max()})
nbre_items.describe()
```

	order_item_id
count	98666.000000
mean	1.141731
std	0.538452
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	21.000000

	order_id	order_item_id	product_id
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089

```
df_order_items['order_id'].value_counts
```

```
8272b63d03f5f79c56e9e4120aec44ef    21
1b15974a0141d54e36626dca3fdc731a    20
ab14fdcfbe524636d65ee38360e22ce8    20
9ef13efd6949e4573a18964dd1bbe7f5    15
428a2f660dc84138d969ccd69a0ab6d5    15
                                    ..
5a0911d70c1f85d3bed0df1bf693a6dd     1
5a082b558a3798d3e36d93bfa8ca1eae     1
5a07264682e0b8fbb3f166edbbffc6e8     1
5a071192a28951b76774e5a760c8c9b7     1
fffe41c64501cc87c801fd61db3f6244     1
```

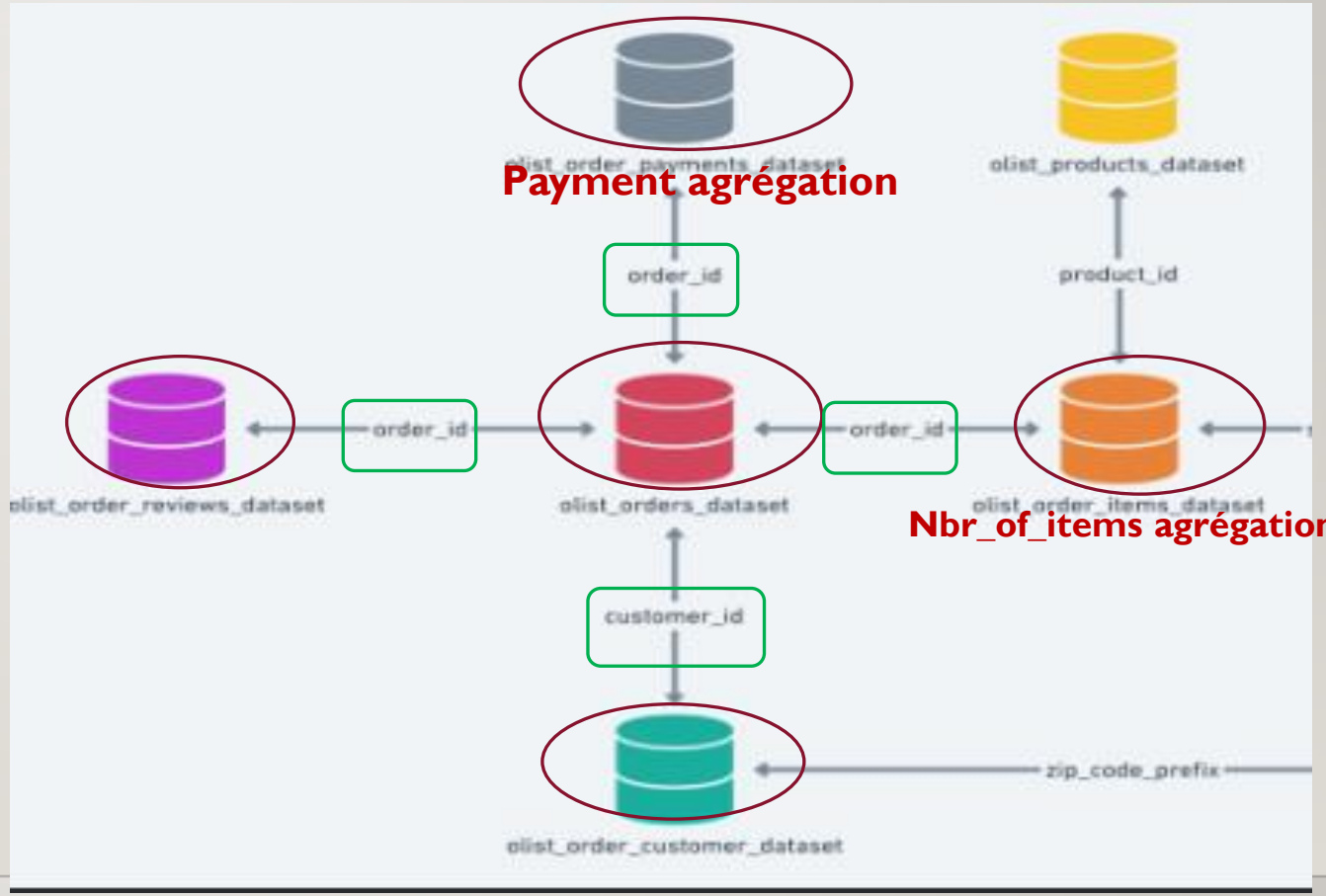
1.3 - SYNTHÈSE DES RÉSULTATS MERGE DES JEUX DE DONNÉES

```
df = df_orders.merge(df_customers, how='inner', on='customer_id')
```

```
df = df.merge(payment, how='inner', on='order_id')
```

```
df = df.merge(df_review, how='inner', on='order_id')
```

```
df = df.merge(nbre_items, how='inner', on='order_id')
```



1.3 - SYNTHÈSE DES RÉSULTATS CALCUL DES VARIABLES RFM-PLUS

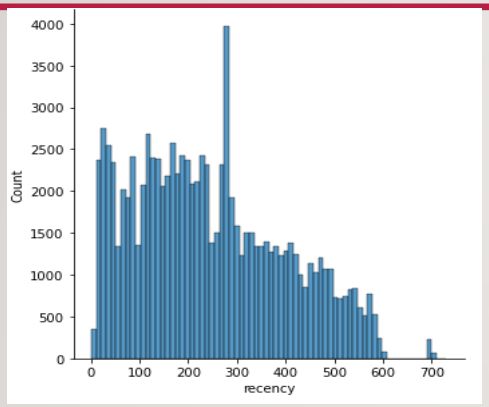
- Groupement par « customer_unique_id »

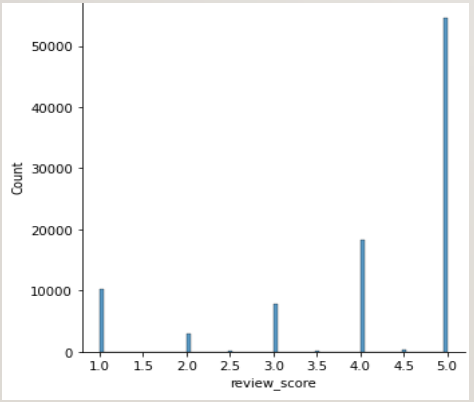
```
rfm_plus = df.groupby('customer_unique_id').agg(
    {'order_purchase_timestamp': lambda x: (latest_date - x.max()),
     'order_id': lambda x: x.count(),
     'payment_value': lambda x: x.sum(),
     'payment_installments': lambda x: x.mean(),
     'review_score': lambda x: x.mean(),
     'order_item_id': lambda x: x.sum()})
```

```
rfm_plus.head()
```

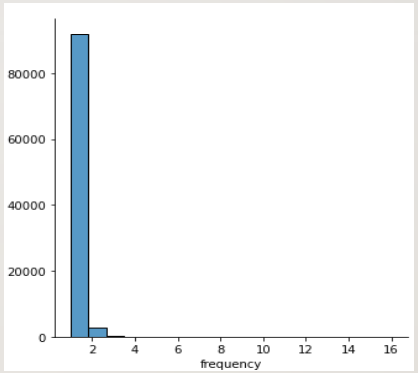
customer_unique_id	recency	frequency	monetary	payment_installments	review_score	nbre_of_items
0000366f3b9a7992bf8c76cfdf3221e2	117	1	141.90	8.0	5.0	1
0000b849f77a49e4a4ce2b2a4ca5be3f	120	1	27.19	1.0	4.0	1
0000f46a3911fa3c0805444483337064	543	1	86.22	8.0	3.0	1
0000f6ccb0745a6a4b88665a16c9f078	327	1	43.62	4.0	4.0	1
0004aac84e0df4da2b147fca70cf8255	294	1	196.89	6.0	5.0	1

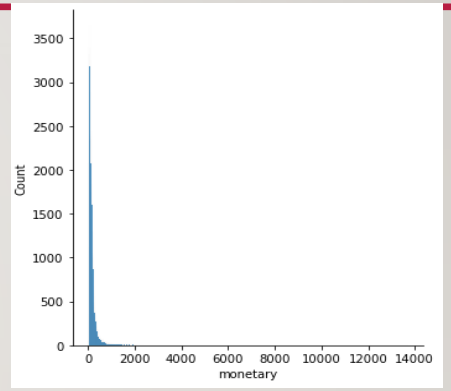
1.4 - ANALYSE EXPLORATOIRE

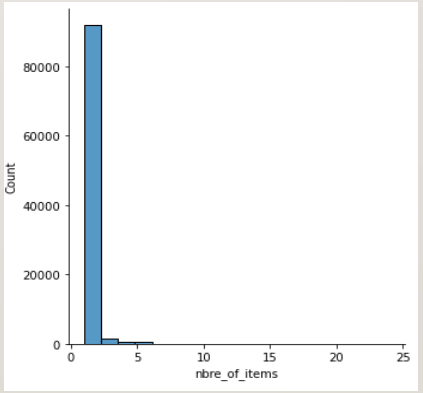


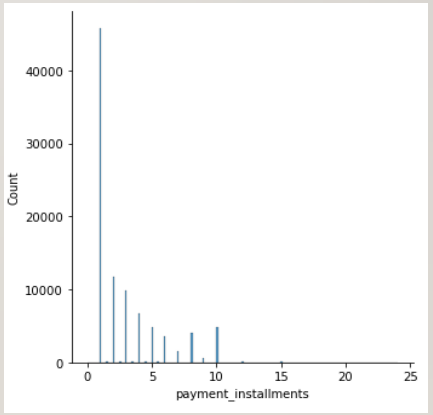


Variables Normales



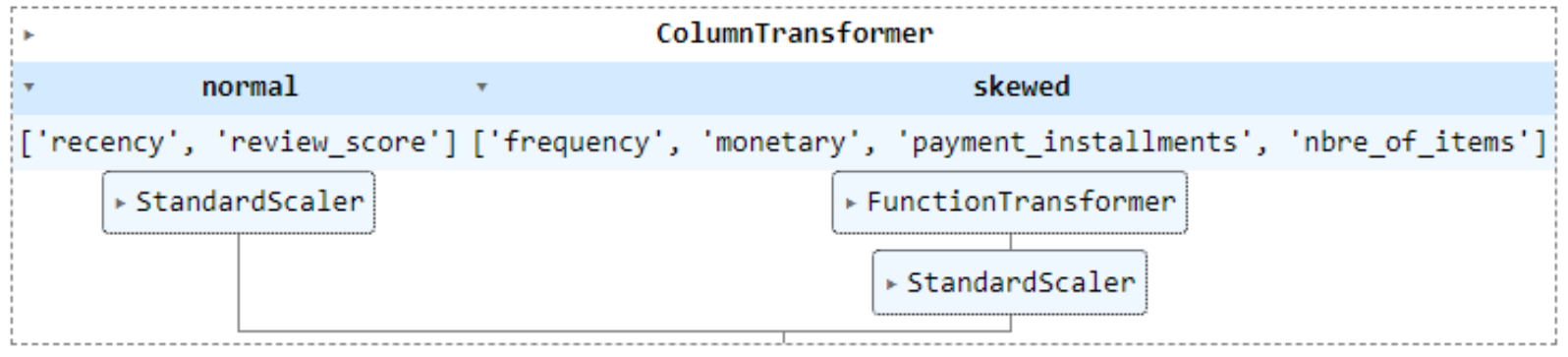






Variables asymétriques (skewed)

1.5 - TRANSFORMATION DES VARIABLES



2- MÉTHODES DE CLASSIFICATIONS

- PCA
- Kmeans
- DBSCAN

2.1 - PCA

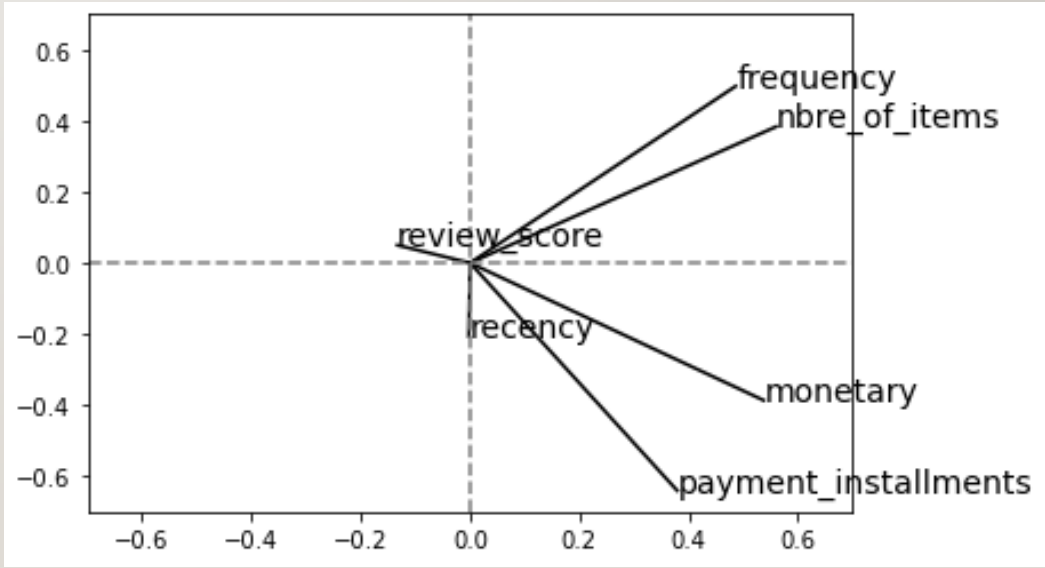
- On commence par une PCA avec 3 composante pour avoir une idée sur la réduction des dimensions

```
PCA
PCA(n_components=3)
```

```
print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.sum())
```

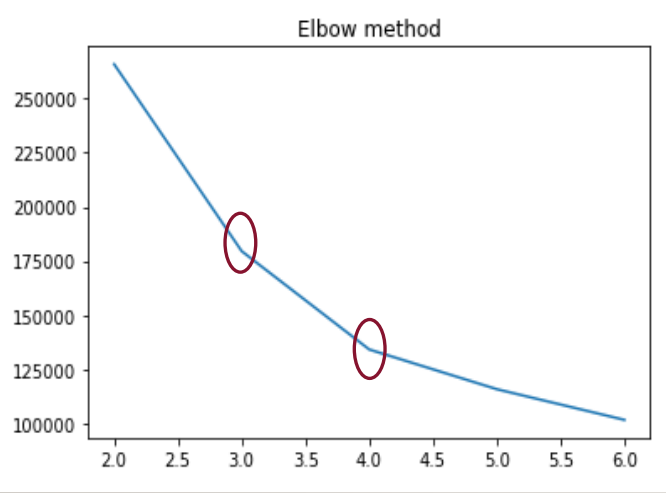
```
[0.28986039 0.19967257 0.16900666]
0.6585396189386463
```

Une PCA avec 3 composantes donne 66% d'explicabilité de la variance

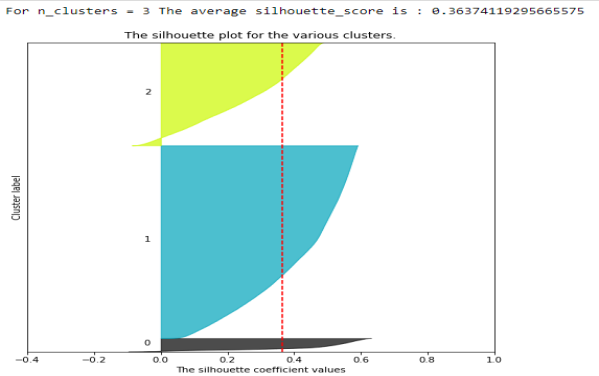


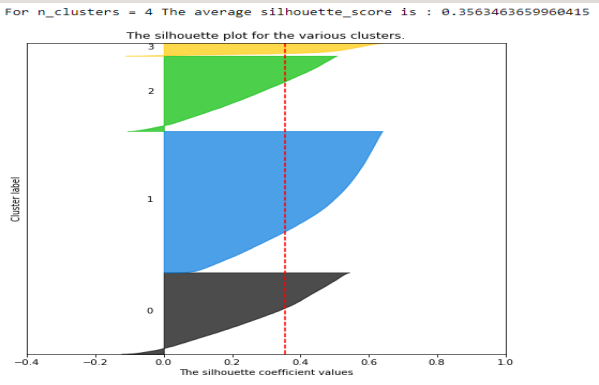
2.1 - PCA PUIS KMEANS

- On essaie de faire une classification kmeans sur le PCA du jeu de donnée
- Methode du coude (elbow method) pour choisir le nombre de clusters k



- Score silhouette score analysis





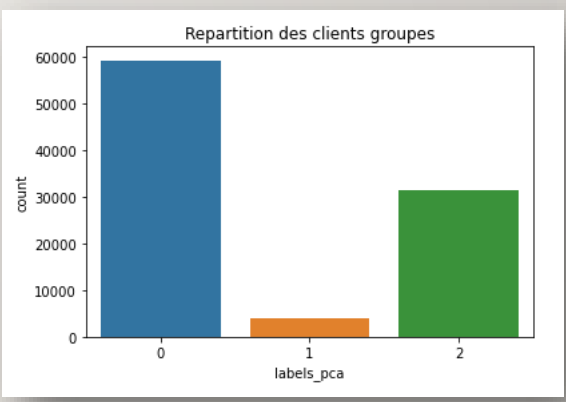
- Decision : n_clusters =3

2.1 - PCA PUIS KMEANS

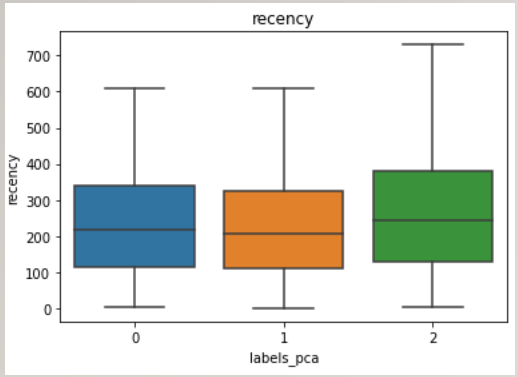
- Kmeans avec n_clusters = 3

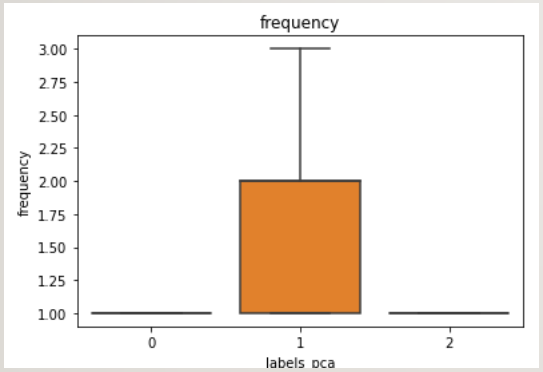
```
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(df_pca)
silhouette_avg = silhouette_score(
    df_pca, kmeans.labels_, sample_size=5000)
print('Silhouette score:', silhouette_avg)
```

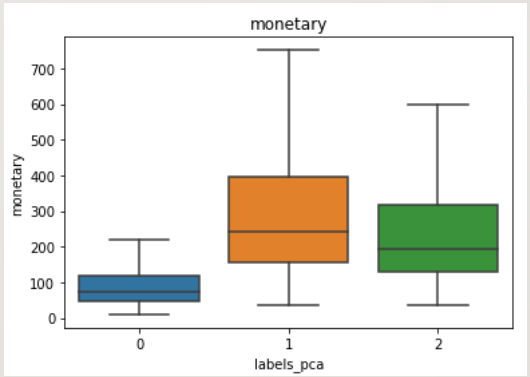
```
Silhouette score: 0.36504113684521106
```

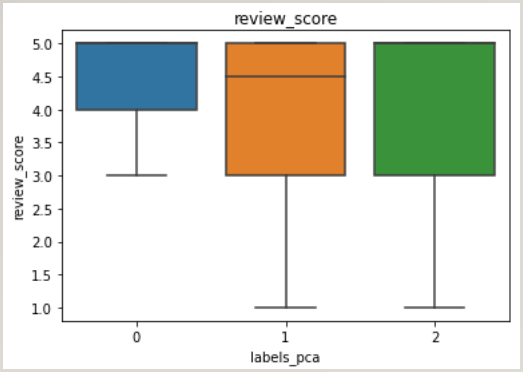


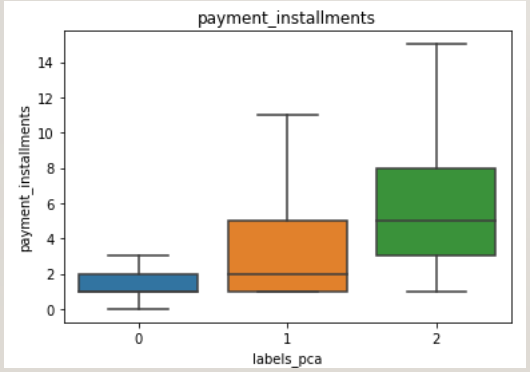
2.1- PCA PUIS KMEANS
EXPLORATION DES CLUSTERS

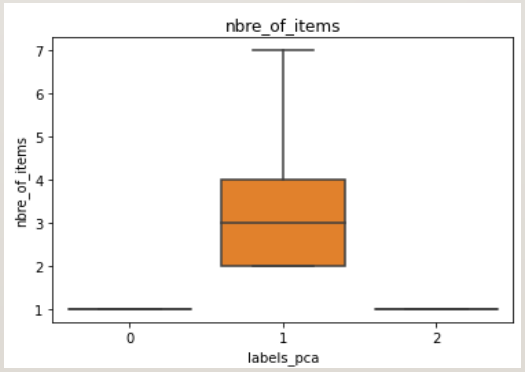












Conclusion:

Les clusters ne sont pas pertinents pour l'objectif de la segmentation client

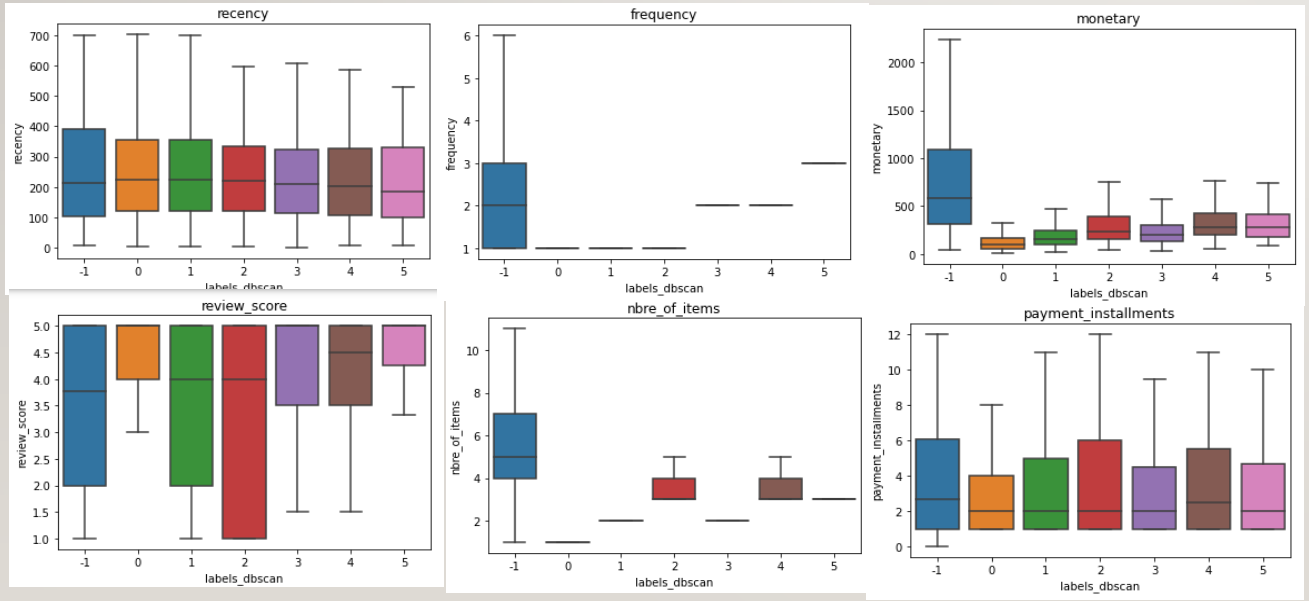
- Groupe 1 représente les clients fréquents ayant acheté plusieurs articles
- Groupe 0 et 2 sont divises par le montant de dépense. La récence n'est pas prise en compte de cette classification.

3 – MODÈLES DE CLASSIFICATION

- Etudes de différents classifications sur un ensemble de variables du jeu de donnée en utilisant les classificateurs kmeans et DBSCAN
- Fonctions pour la recherche des hyperparamètres et le calcul du model
  - Kmeans:
    - **get_score(X, a, b)** qui donne le graphe de l'inertie pour k entre a et b (elbow method)
    - **get_model(X, k, nom_du_model)** donne la classification kmeans avec n_clusters = k et affiche le silhouette score ainsi que les boxplot de la distribution des différentes variables par cluster
  - DBSCAN:
    - **search_dbscan(X, range_eps, min_samples)** recherche pour chaque eps dans range_eps le silhouette score et le nombre de labels
    - **get_dbscan_model(X, eps, min_samples)** donne la classification DBSCAN avec les hyperparamètres correspondants et affiche le silhouette score ainsi que les boxplot de la distribution des différentes variables par cluster

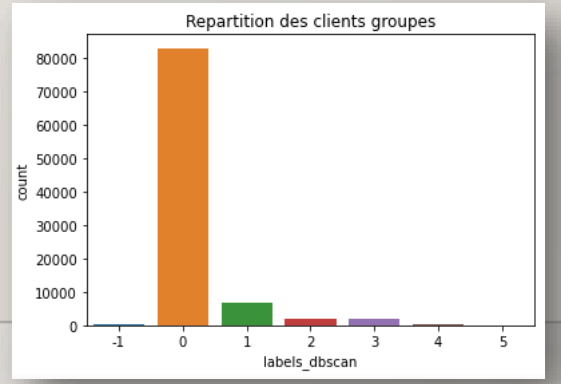
3.1 – DBSCAN SUR L'ENSEMBLE DES VARIABLES

- Une petite recherche des hyperparamètres a permis de choisir eps=1.2, min_sample= 2*(nbre de variables)



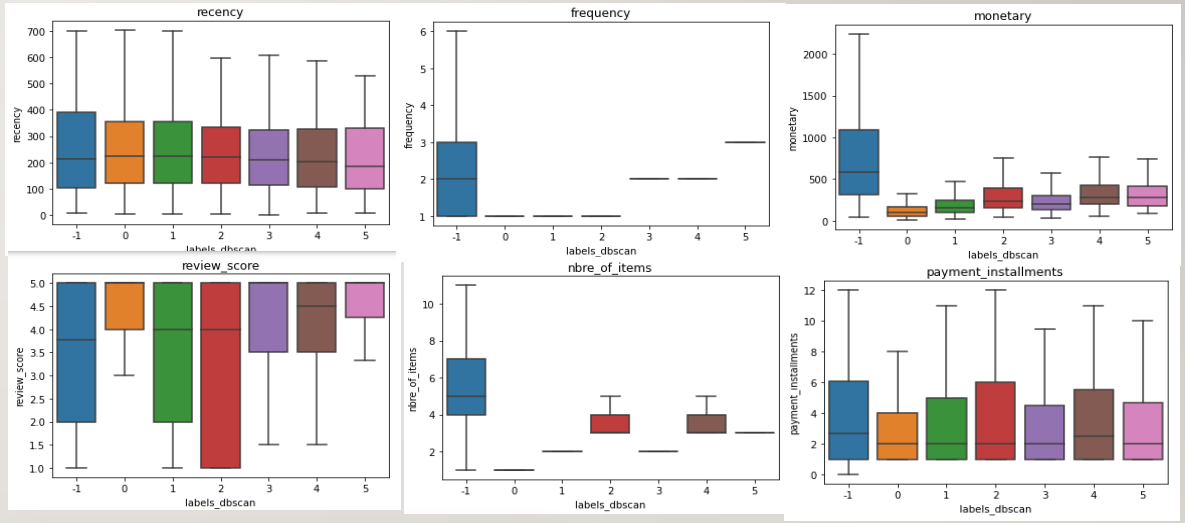

```
search_dbscan(df_scaled, range_eps, df_scaled.shape[1]*2)
```

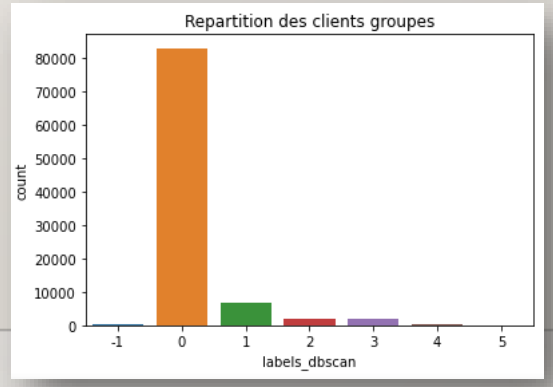
```
eps_value :0.6
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
3, 34, 35, 36, -1}
for eps0.6 silhouette score is 0.026192222367832713
eps_value :0.8
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, -1}
for eps0.8 silhouette score is 0.2393775347159524
eps_value :1
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -1}
for eps1 silhouette score is 0.26167243505617965
eps_value :1.2
{0, 1, 2, 3, 4, 5, -1}
for eps1.2 silhouette score is 0.2750610642308364
```



3.1 – DBSCAN SUR L'ENSEMBLE DES VARIABLES

- Le nombre de clusters est assez grand
  - 6 clusters avec la classe des intrus
- La classification est surtout faite en fonction de la fréquence et du nombre d'articles. Pour le reste des variables on ne trouve pas une vrai segmentation
- La classification est très disproportionnée avec le group 0 contenant la majorité des clients.
- DBSCAN n'est pas adapter pour ce model.





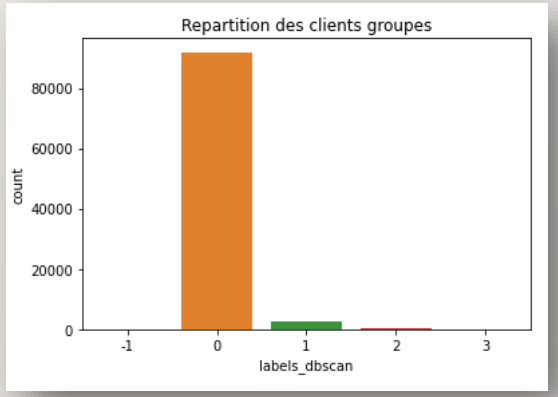
3.2 - DBSCAN SUR LE MODEL RFM DE BASE

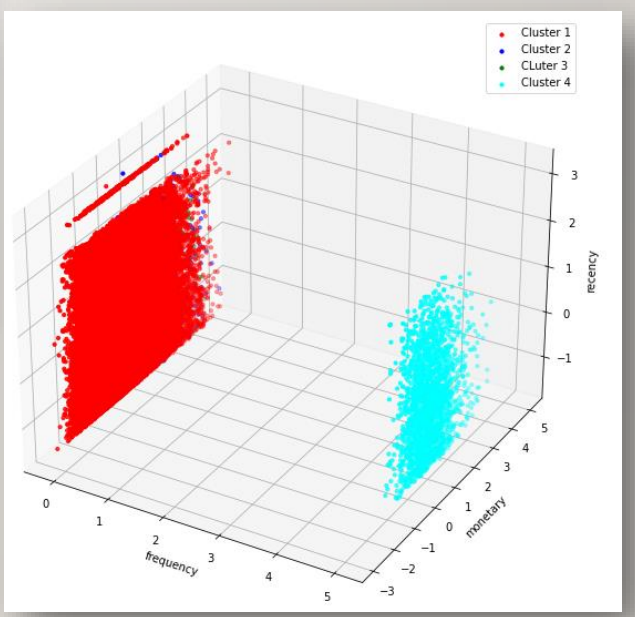
```
range_eps = [0.4, 0.6, 0.8]
search_dbscan(df_rfm, range_eps, df_rfm.shape[1]*2)
```

```
eps_value :0.4
{0, 1, 2, 3, 4, 5, 6, -1}
for eps0.4 silhouette score is 0.4327597953572456
eps_value :0.6
{0, 1, 2, 3, -1}
for eps0.6 silhouette score is 0.6847874482432005
eps_value :0.8
{0, 1, 2, 3, 4, -1}
for eps0.8 silhouette score is 0.6886994206323178
```

```
get_dbscan_model(df_rfm, 0.6, df_rfm.shape[1]*2)
```

```
Le score silhouette est :  0.686460362661407
```

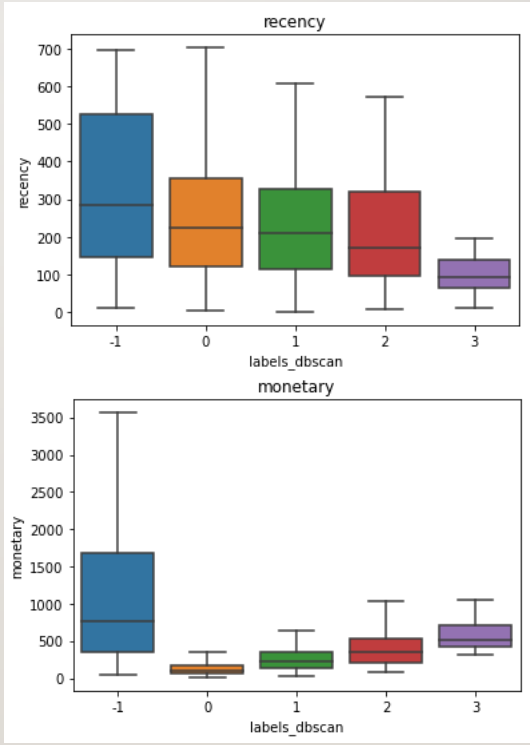


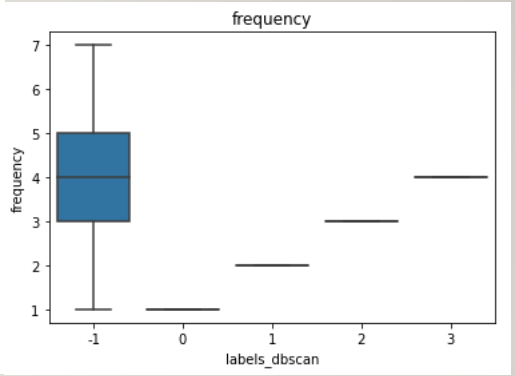


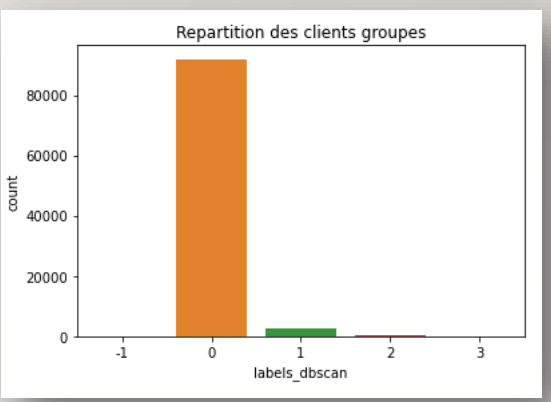
- eps=0.8, min_samples = 6
- Nombre de groupes 4
- Silhouette score 0.67

3.2 - DBSCAN SUR LE MODEL RFM DE BASE

- Les groupes sont très hétérogènes
- Ils sont surtout divises par fréquence et montant de dépenses
- Groupe 0 : Les clients a faibles dépenses et qui ont acheté une seule fois (majorité des clients)
- Groupe 1 : les clients qui ont acheté 2 fois, faibles dépenses
- Groupe 2 : Les clients qui ont acheté 3 fois
- Groupe 3 : Les clients qui ont acheté 4 fois et leur dernier achat est assez récent







3.3 KMEANS

- Etude kmeans sur différentes combinaisons de variables
- Etude du silhouette score et de la distribution des variables par clusters dans chaque cas
- Conclusion sur les modèles les plus pertinents pour la segmentation des clients

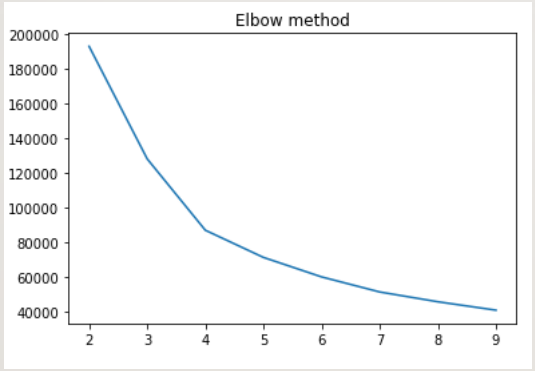
3.3.1- RÉSUMÉ DES RÉSULTATS

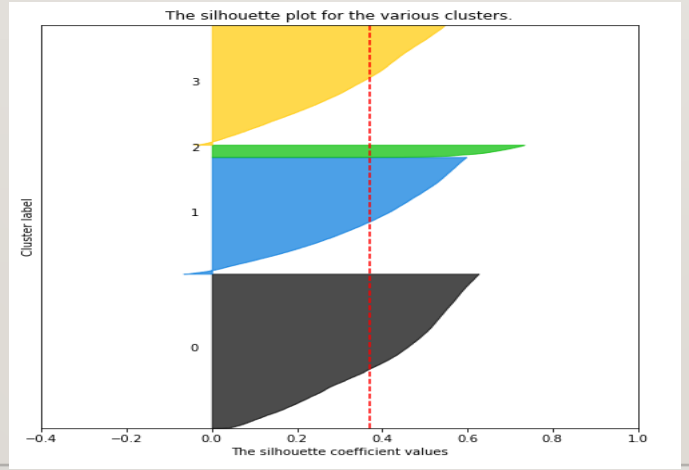
- Synthèse des résultats des différents combinaison par
  - Silhouette score moyen
  - Nombre de clusters optimal

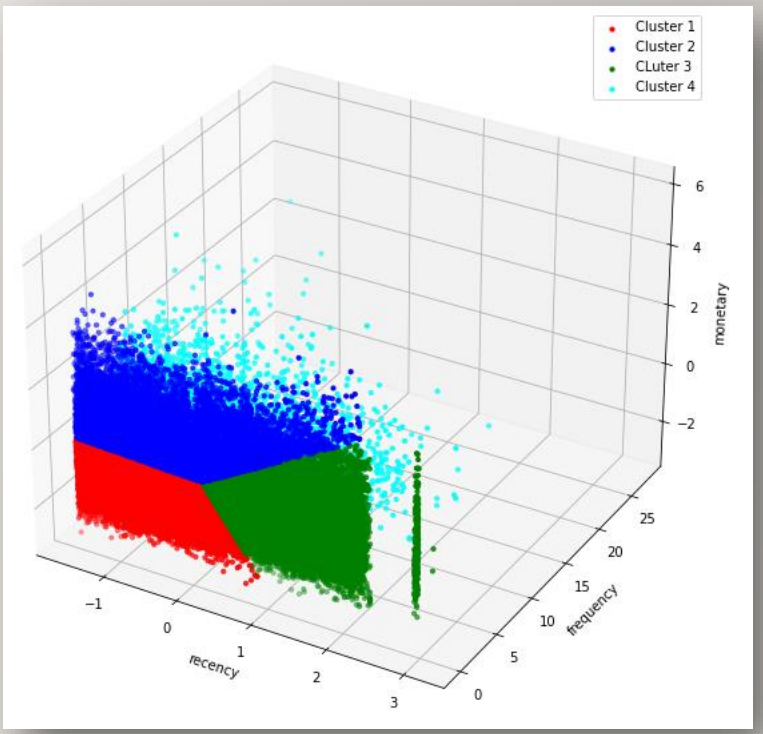
	all	Recency-Frequency-Monetary	Recency-Frequency-Monetary-Nbre_of_items	Recency-Monetary-Nbre_of_items	Recency - Monetary - Nbre_of_items - Review	Recency - Monetary - Nbre_of_items - Payment_installments
Silhouette score	0.279906	0.376168	0.356941	0.352642	0.307781	0.290151
Number of clusters	4.000000	4.000000	5.000000	4.000000	4.000000	4.000000

3.3.2 - RECENCY FREQUENCY MONETARY

- N_clusters=4
- Silhouette score = 0.37

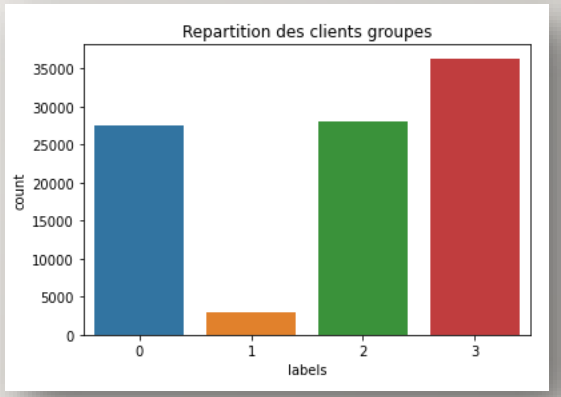


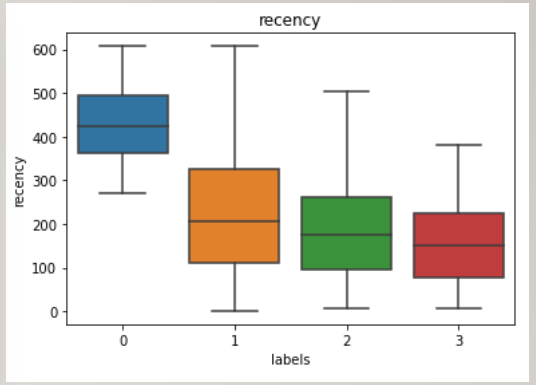


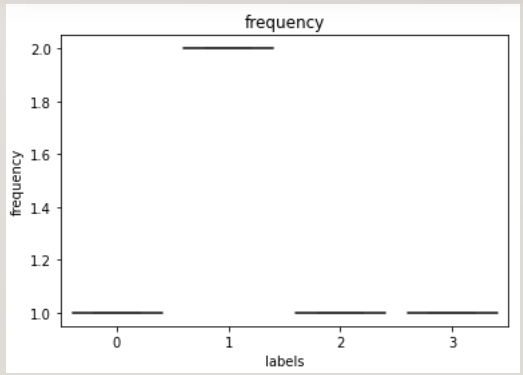


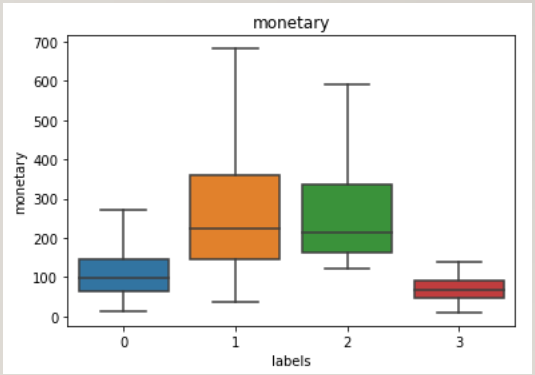
3.3.2 - RECENCY FREQUENCY MONETARY

- Group 0 : les anciens clients a faibles dépenses
- Groupe 1 : Les clients récurrents récents et dépensent plus
- Groupe 2 : Les clients récents qui dépensent plus
- Groupe 3 : Les clients récents a faibles dépenses
- Conclusion : Cette classification est satisfaisante:
  - elle catégorise les clients par fréquence, récence et budget.
  - Elle peut satisfaire les objectifs de l'équipe de communication



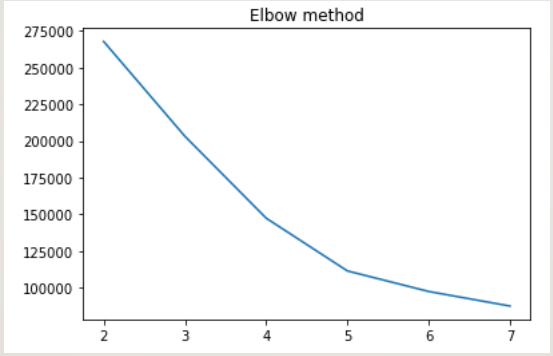


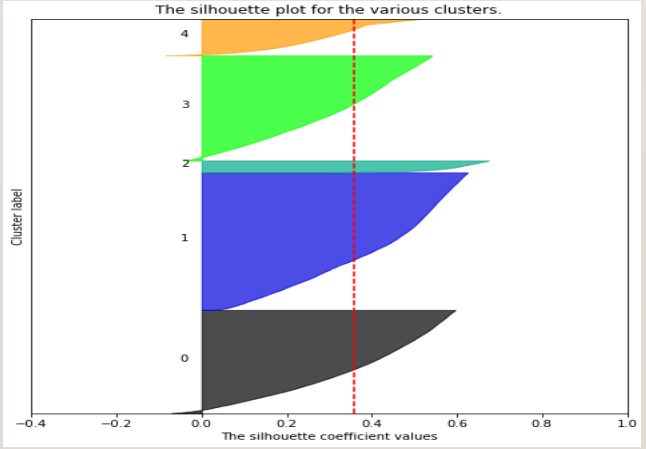


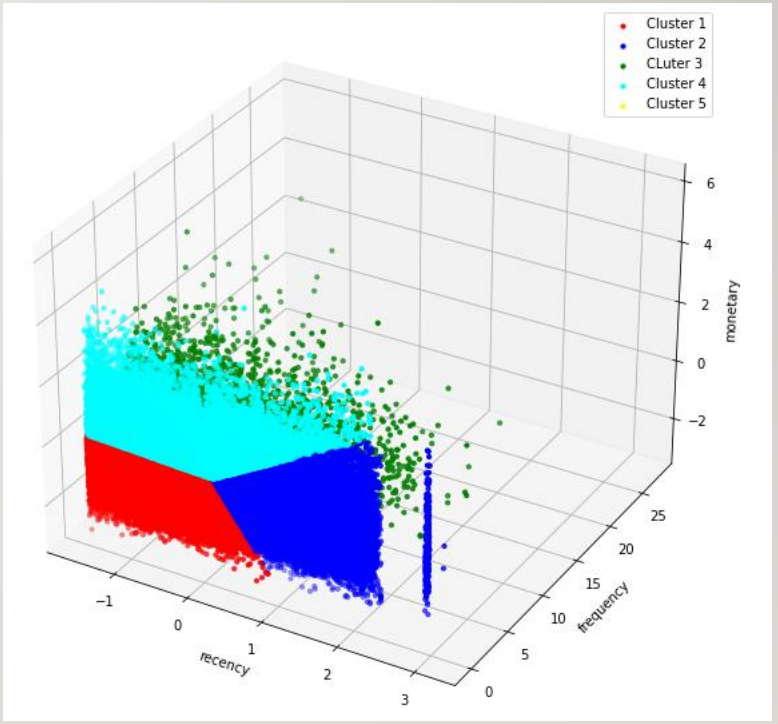


3.3.3 - RECENCY FREQUENCY MONETARY NBRE_OF_ITEMS

- N_clusters=5
- Silhouette score = 0.367

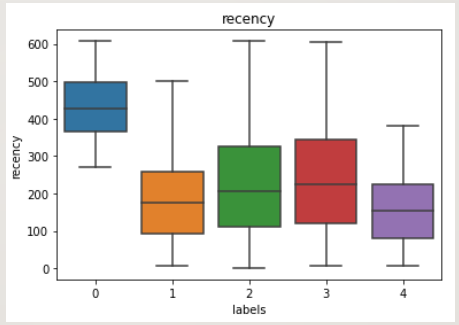


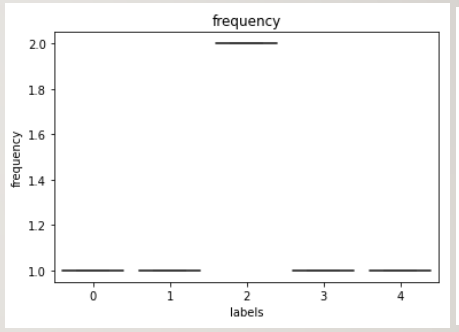


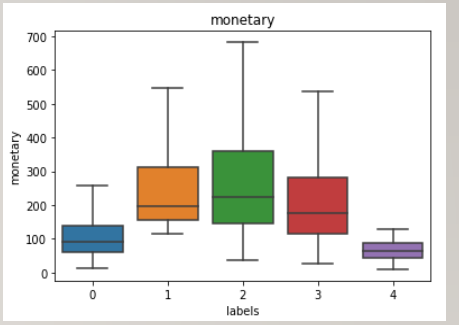


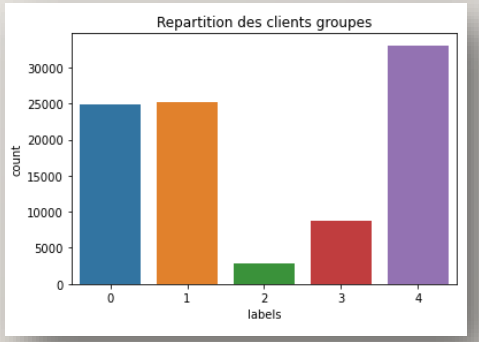
3.3.3 - RECENCY FREQUENCY MONETARY NBRE_OF_ITEMS

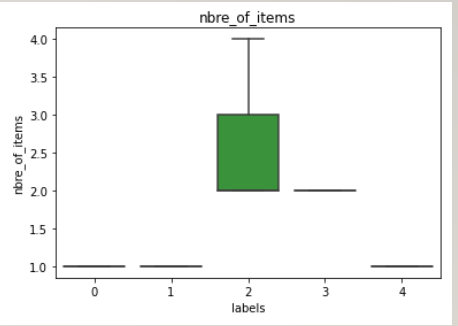
- Group 0: les anciens clients a faibles dépenses
- Groupe 1 : Les clients récents qui dépensent plus
- Groupe 2 : Les clients récurrents qui achètent plusieurs articles et dépensent plus
- Groupe 3 : : Les clients qui achètent au moins 2 articles
- Groupe 4 : Les clients récents a faibles dépenses
- Conclusion : Cette segmentation apporte une nouvelle classe importante de clients, les grand acheteurs c'est donc aussi une alternative a prendre en compte.











3.4 - CONCLUSION

- Le classificateur DBSCAN n'est pas adapté pour la segmentation des client dans ce modèle
- Le classificateur kmeans ne donne des classifications satisfaisantes pour :
  - Le model RFM
  - Le model RFM + nombre d'articles achetés
  - Le model RM + nombre d'articles achetés (ce dernier modèle est très similaire au model 2 mais la classe clients fréquents et clients non fréquents mais qui ont acheté plusieurs articles sont groupes en une seuls classe)
- Une fois le model entraine, le kmeans permet de classifier facilement de nouveaux clients a l'aide de la fonction kmeans.predict()

4- CONTRAT DE MAINTENANCE

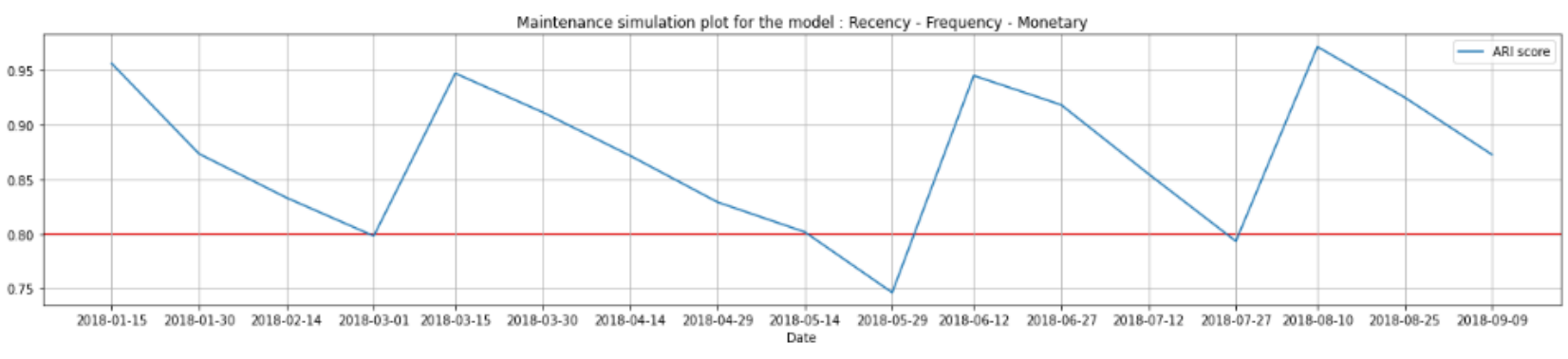
- Nous allons étudier la stabilité du modèle pour les 2 modèles choisis
  - Le model RFM
  - Le model RFM + nombre d'articles achetés
- Jeu de données initial : df0 jusqu'à 31/12/2017
- Simulation de maintenance:
  - Période : tous les 15 jours
  - Méthode de mesure et décision : ARI score

4.1 - MÉTHODOLOGIE:

- Modèle initial M0
  - df0 : jeu de donnée jusqu'à t0=31/12/2017
  - rfm0 calculée a partir de df0
  - rfm0_scaled=fullpipeline0. fit_transform(rfm0)
  - Kmeans0.fit(rfm0_scaled)
- Nouveau modèle M1
  - df1 : jeu de donnée jusqu'à t1=t0+15jours
  - rfm1 calcule a partir de df1
  - rfm1_scaled=fullpipeline1 fit_transform(rfm1)
  - C1=kmeans1.fit_predict(rfm1_scaled)
- Prédiction a partir de M0
  - Rfm1_init_scaled = fullpipeline0.transform(rfm1)
  - C1_init=kmeans0.predict(rfm1_scaled)
  - ARI score(C1_init, C1)
  - Stop si ARI<0.8

4.2 - CONTRAT DE MAINTENANCE RECENCY – FREQUENCY - MONETARY

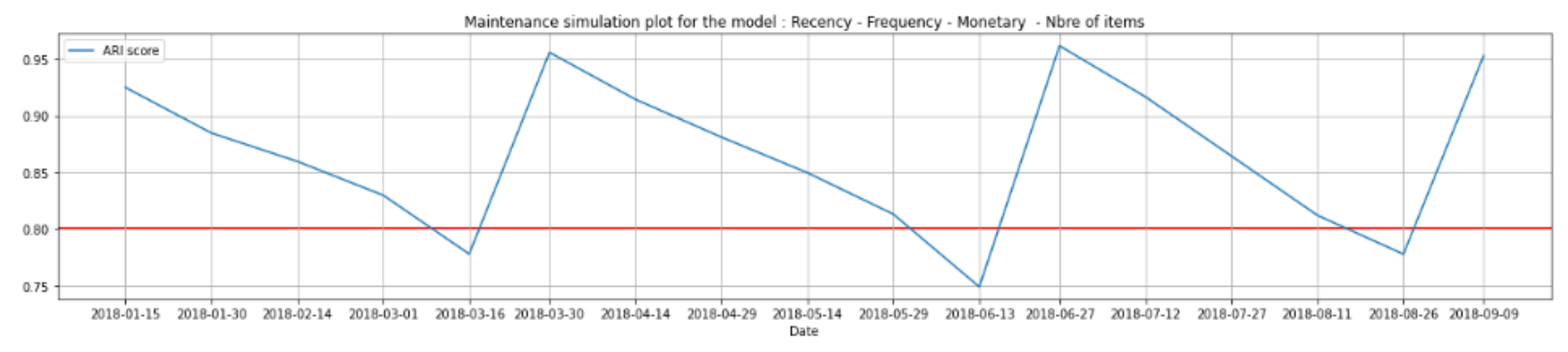
- Résultat: Mise a jours de la modélisation tous les 2 mois en moyenne



	Date	T1	ARI score
15	2018-01-15	15	0.956574
30	2018-01-30	30	0.873567
45	2018-02-14	45	0.833252
60	2018-03-01	60	0.798077
75	2018-03-15	15	0.947489
90	2018-03-30	30	0.911704
105	2018-04-14	45	0.871857
120	2018-04-29	60	0.82919
135	2018-05-14	75	0.801713
150	2018-05-29	90	0.746079
165	2018-06-12	15	0.94542
180	2018-06-27	30	0.918548
195	2018-07-12	45	0.855058
210	2018-07-27	60	0.793292
225	2018-08-10	15	0.971926
240	2018-08-25	30	0.925293
255	2018-09-09	45	0.87272

4.3 - CONTRAT DE MAINTENANCE RECENCY – FREQUENCY – MONETARY – NBRE_OF_ITEMS

- Résultat: Mise a jours de la modélisation tous les 2 mois et demi a 3 mois



	Date	T1	ARI score
15	2018-01-15	15	0.925116
30	2018-01-30	30	0.884796
45	2018-02-14	45	0.859686
60	2018-03-01	60	0.829727
75	2018-03-16	75	0.777771
90	2018-03-30	15	0.956063
105	2018-04-14	30	0.914529
120	2018-04-29	45	0.88111
135	2018-05-14	60	0.84966
150	2018-05-29	75	0.813021
165	2018-06-13	90	0.748722
180	2018-06-27	15	0.961895
195	2018-07-12	30	0.916733
210	2018-07-27	45	0.864385
225	2018-08-11	60	0.81198
240	2018-08-26	75	0.777647
255	2018-09-09	15	0.953244

Merci!