SEGMENTEZ LES CLIENTS D'UN SITE E-COMMERCE

Projet 5 - Parcours Data Scientist – OpenClassrooms

Christophe ABALLEA – Mai 2024

CONTEXTE & MISSION

Olist : entreprise brésilienne proposant une solution de vente sur les marketplaces en ligne

- Requêtes SQL pour le Dashboard
- Mission principale :
  Réaliser une **segmentation de la clientèle de type RFM**
- Compléter par une simulation de contrat de maintenance

PLAN

1 — 2 — 3 — 4

Les données

- Analyse
- Feature ingeniering
- Dataset final

Les modèles

- KMeans
- DBSCAN
- AgglomerativeClustering

Le clustering

- Focus sur le modèle retenu
- Interprétations métiers

La maintenance

- Démarche
- Évolution des scores ARI

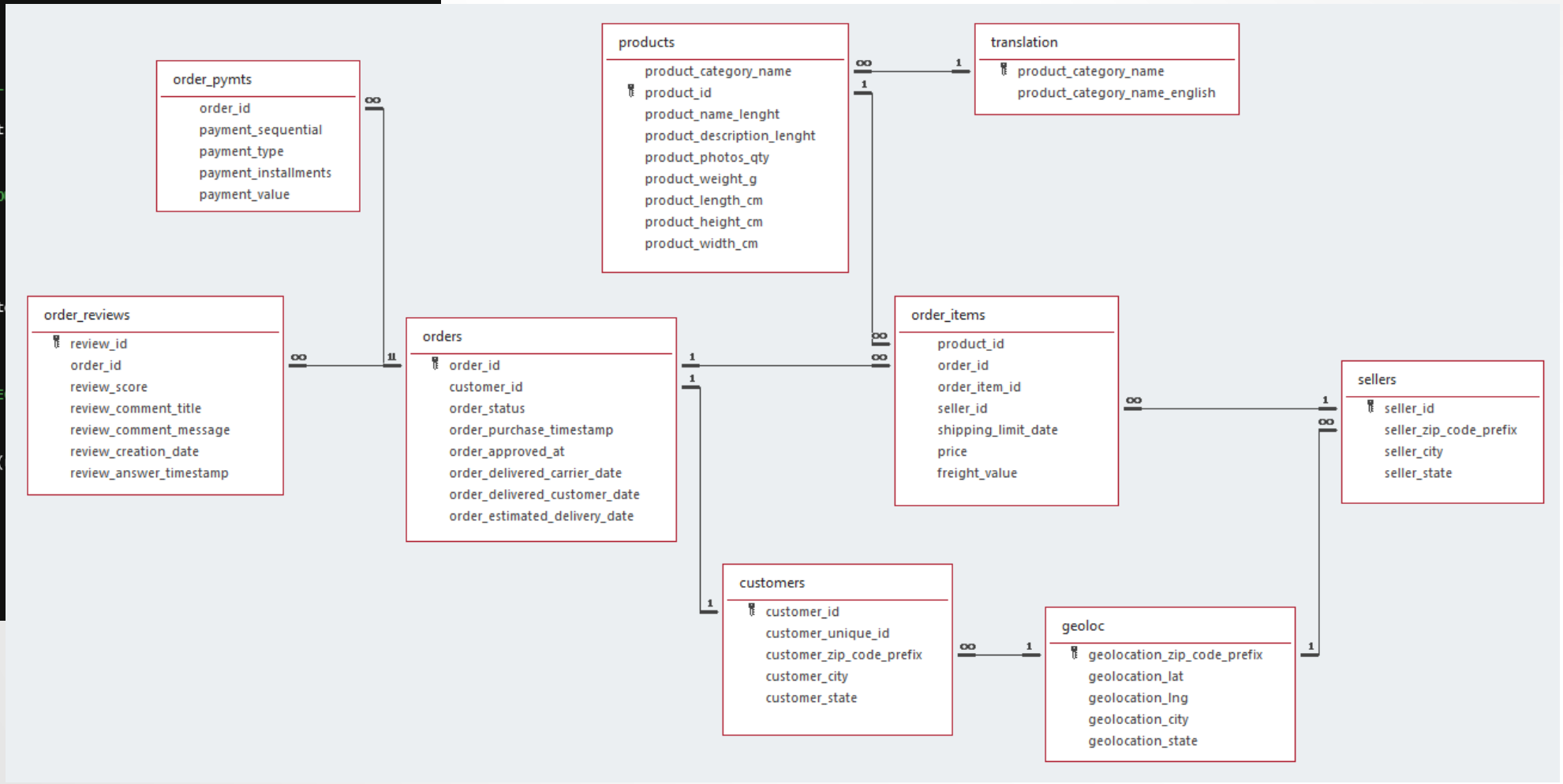
1

Les données

1 Les données

```sql
-- Quels sont les 5 codes postaux, enregistrant plus de 30 commandes, avec le pire review scor
WITH orders_zip_codes AS (
    WITH
    valid_orders AS (
        SELECT order_id, customer_id
        FROM orders
        WHERE order_approved_at IS NOT NULL
            AND order_status != 'canceled'
            AND julianday(order_approved_at
    ),
    zip_codes AS (
        SELECT customer_zip_code_prefix, CO
        FROM customers
        GROUP BY customer_zip_code_prefix
        HAVING COUNT(customer_id) > 30
    )
    SELECT customer_zip_code_prefix, c.cust
    FROM valid_orders vo
    INNER JOIN customers c
    ON vo.customer_id = c.customer_id
    WHERE customer_zip_code_prefix IN (SELE
    ORDER BY customer_zip_code_prefix
)
SELECT customer_zip_code_prefix, ROUND(AVG(
FROM orders_zip_codes ozc
LEFT JOIN order_reviews o
ON o.order_id = ozc.order_id
GROUP BY customer_zip_code_prefix
ORDER BY moyenne_review_scores ASC
LIMIT 5;
```

SCHÉMA RELATIONNEL

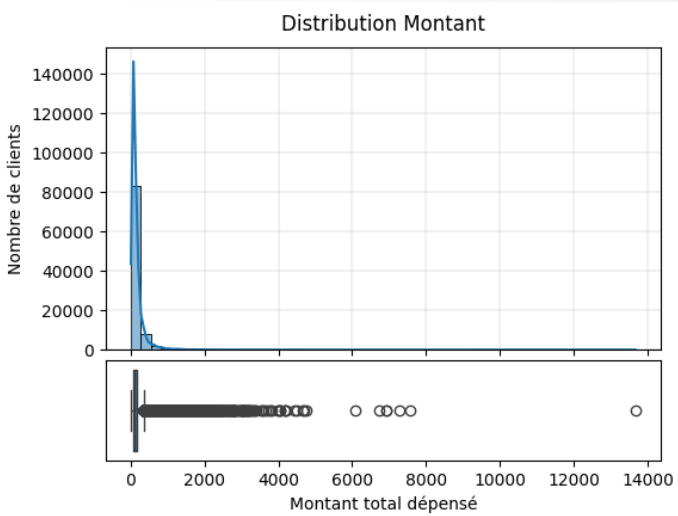


1 Les données

ANALYSE EXPLORATOIRE

Sélection des features pour réaliser une segmentation des clients de type **RFM** :

- **R**écence : Nombre de jours écoulés depuis le dernier achat
- **F**réquence : Nombre total d'achats effectués par un client
- **M**ontant : Montant total dépensé par un client

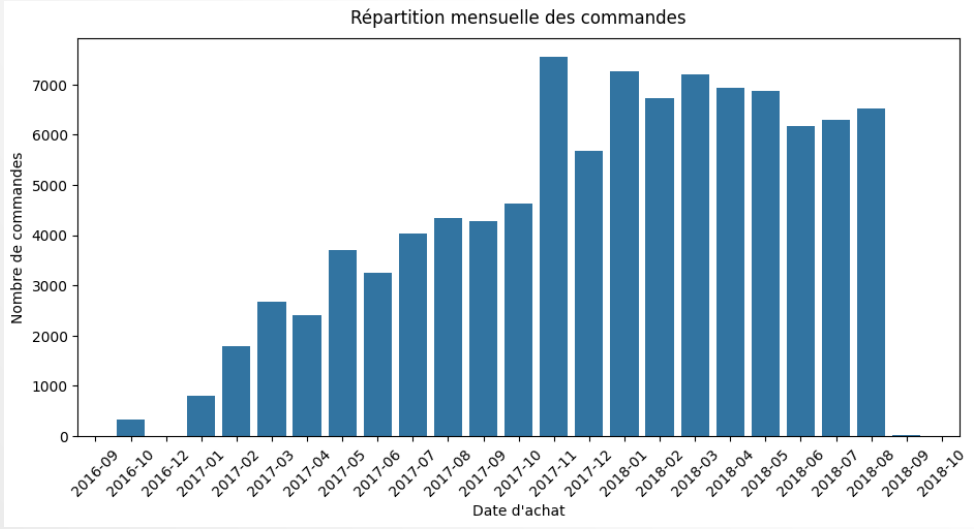


Création d'un dataset complet
94 703 observations X 24 features

1 observation = 1 commande

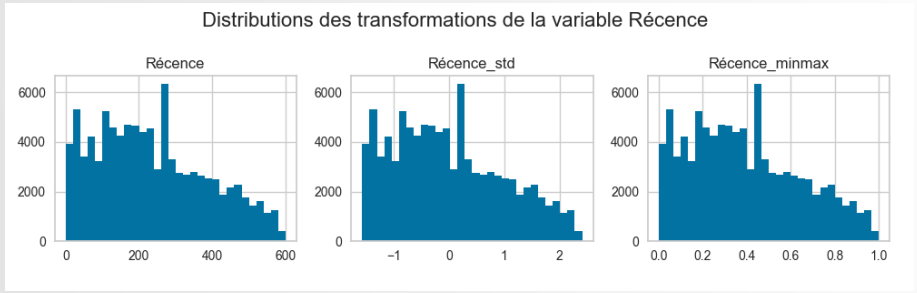
Basé sur les tables :
- 'customers'
- 'orders'
- 'order_items'
- 'order_reviews'

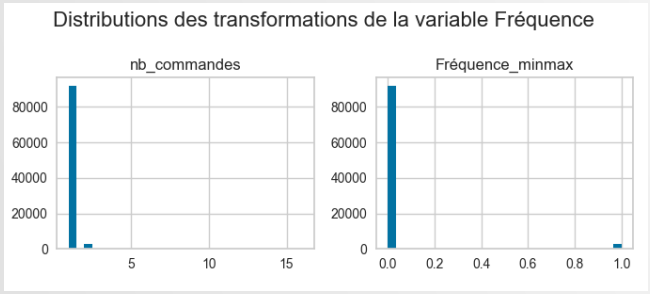
Méthode : groupby + merge



	nb_commandes	Nbre de clients	%
0	1	91829	96.97
1	2	2639	2.79
2	3	187	0.20
3	4	29	0.03
4	5	9	0.01
5	6	5	0.01
6	7	3	0.00
7	9	1	0.00
8	16	1	0.00

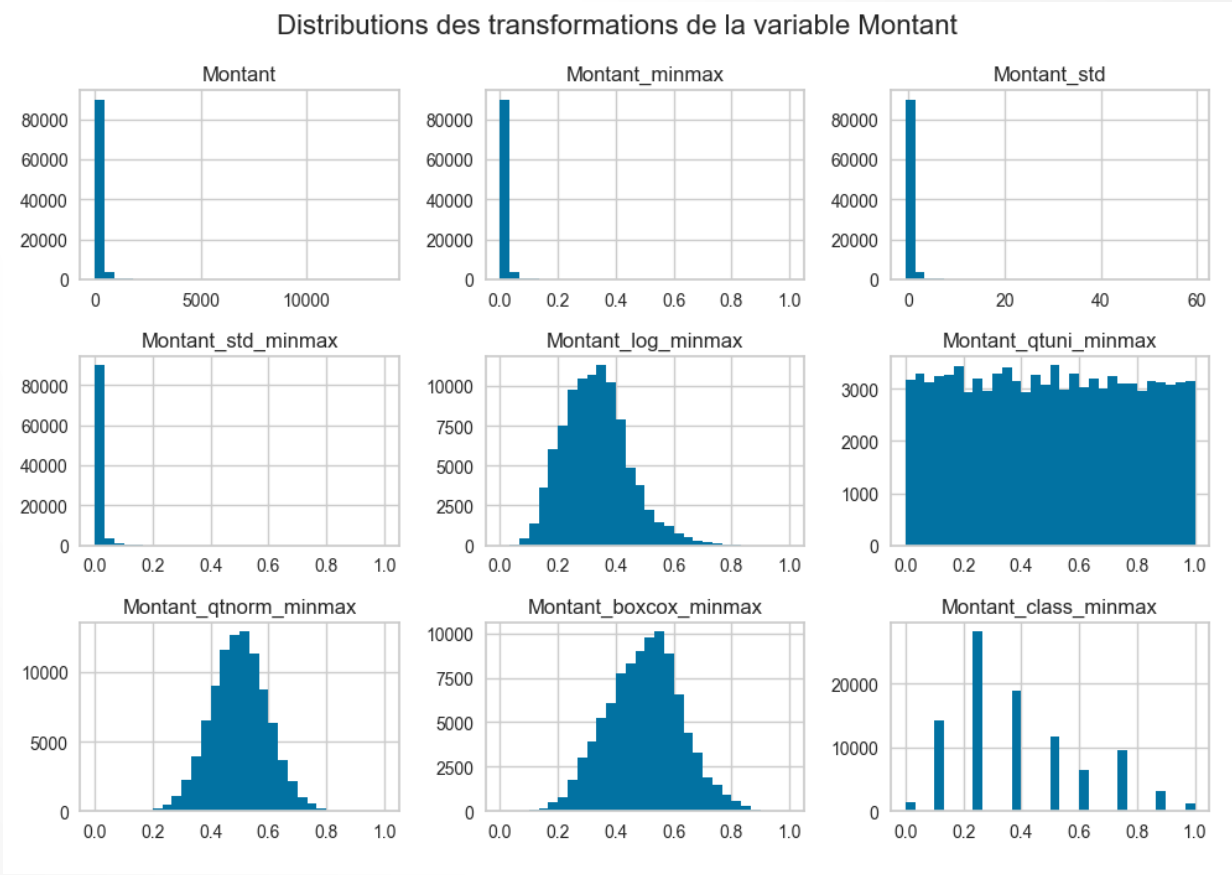
1 Les données





- MinMaxScaler systématique
- Binarisation feature ‘Fréquence’
- Création de la variable Satisfaction

FEATURE INGENIERING



- Classements des montants en tranches facilement interprétables par l'équipe marketing

```
# Définition des bornes pour les intervalles de classement
bins = [0, 25, 50, 100, 150, 200, 250, 500, 1000, float('inf')]
labels = range(len(bins) - 1)
```

```
df_features['Montant_class'] = pd.cut(df_features['Montant'], bins=bins, labels=labels, right=False)
```

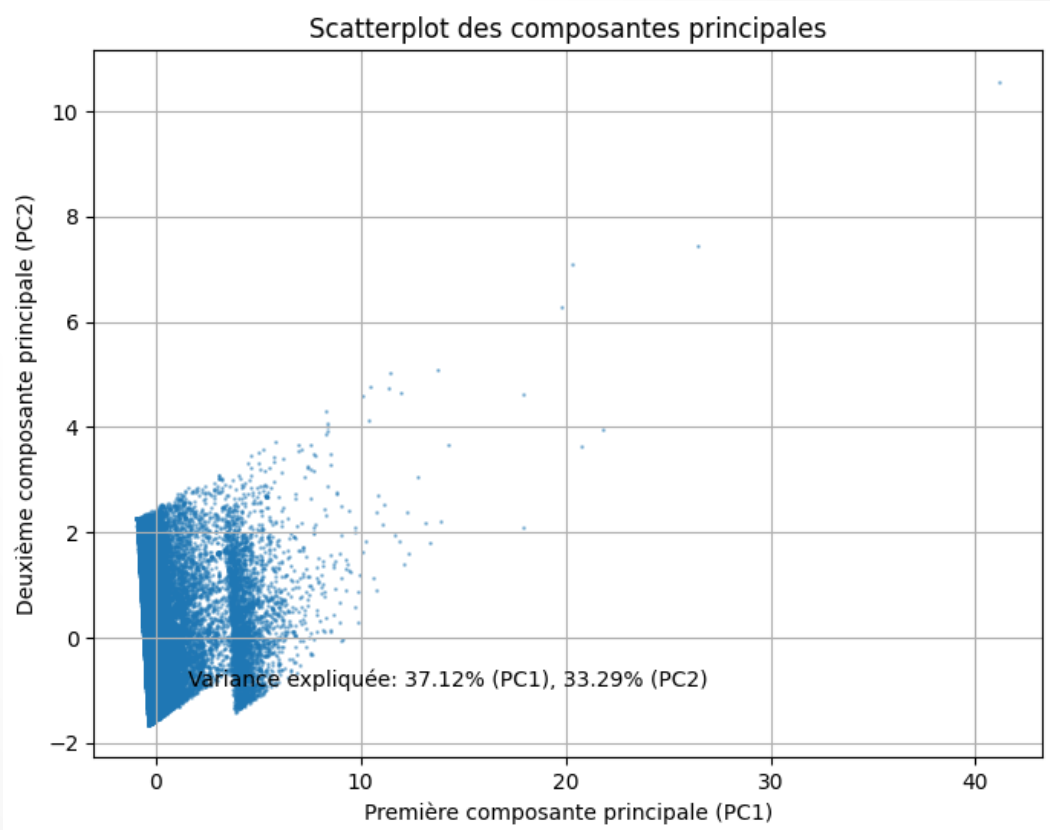
1 Les données

DATASET FINAL

- **Récence :** date_dernière_commande, Récence, Récence_minmax
- **Fréquence :** Fréquence, Fréquence_minmax
- **Montant :** Montant, Montant_log_minmax Montant_qtuni_minmax, Montant_qtnorm_minmax, Montant_boxcox_minmax, Montant_class_minmax
- **Satisfaction :** Satisfaction

- 97 905 commandes
- 94 703 clients

+ 1 version du dataset avant transformations pour la simulation du contrat de maintance



Récence, Fréquence (binarisée), Montant

2

Les modèles

2 Les modèles

DBSCAN

Algorithme : **DBSCAN**

Biblitohèque : **Scikit-learn**

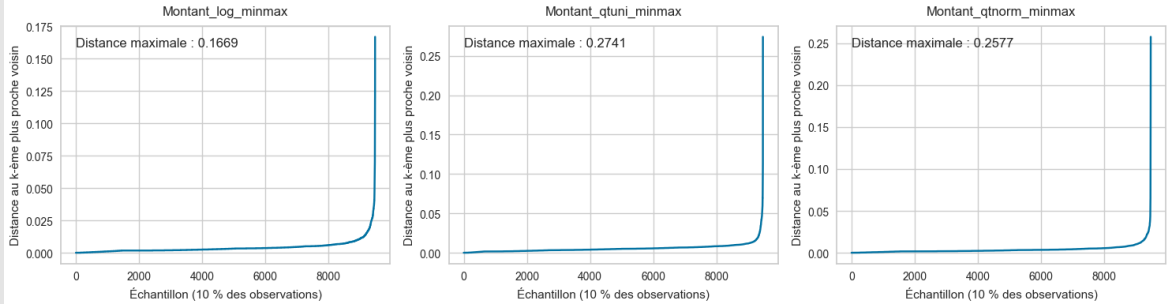
Algorithme de clustering basé sur la densité qui peut identifier un nombre arbitraire de clusters de formes variées

Fonctionnement :

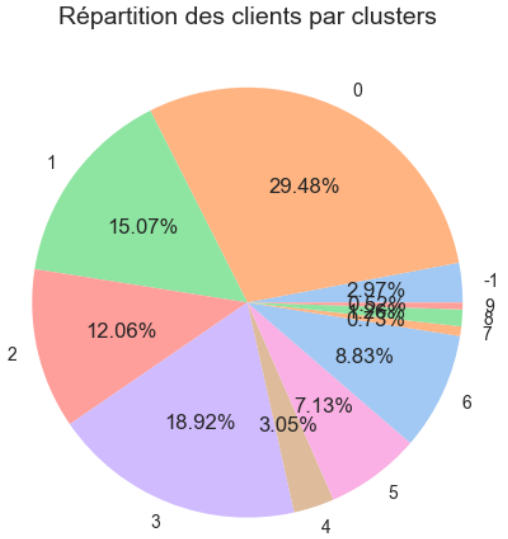
- Recherche des points voisins dans un rayon <= **epsilon**
- S'il en trouve au moins **min_samples**, le point est considéré comme point central d'un cluster
- Propagation par répétition du processus
- Tous les points trop éloignés d'un cluster sont considérés comme des points de bruit

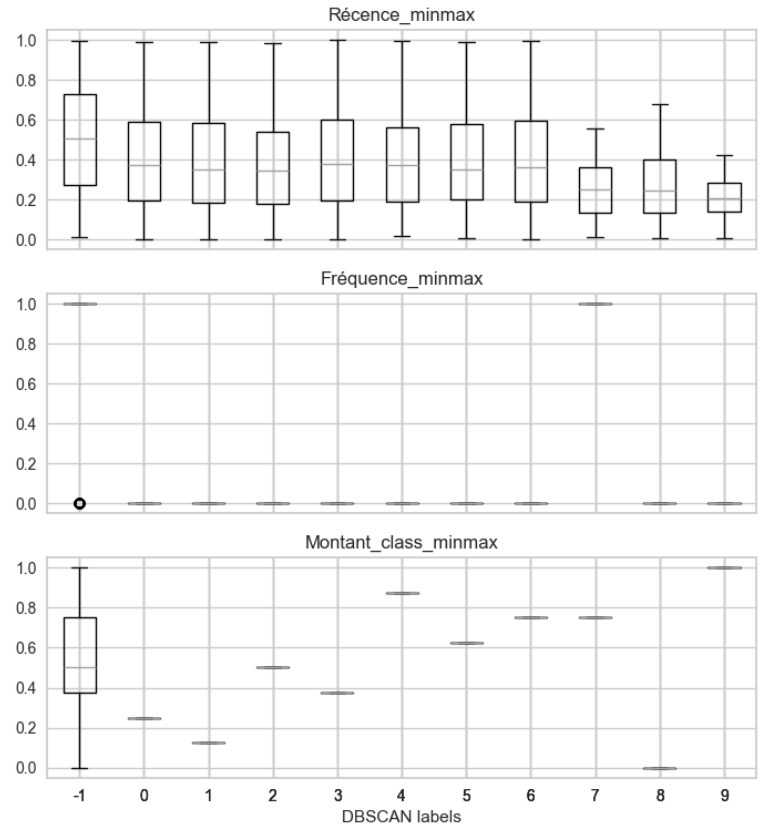
2 Les modèles

DBSCAN



	Unnamed: 0	feature_montant	eps	min_samples_value	n_clusters	n_noise
245	245	Montant_class_minmax	0.110000	20	11	244
244	244	Montant_class_minmax	0.110000	15	12	142
251	251	Montant_class_minmax	0.112222	20	12	222
257	257	Montant_class_minmax	0.114444	20	12	222
263	263	Montant_class_minmax	0.116667	20	12	222
250	250	Montant_class_minmax	0.112222	15	13	127
281	281	Montant_class_minmax	0.123333	20	13	178
269	269	Montant_class_minmax	0.118889	20	13	202





2 Les modèles

AGGLOMERATIVE CLUSTERING

Algorithme : **AgglomerativeClustering**

Biblitohèque : **Scikit-learn**

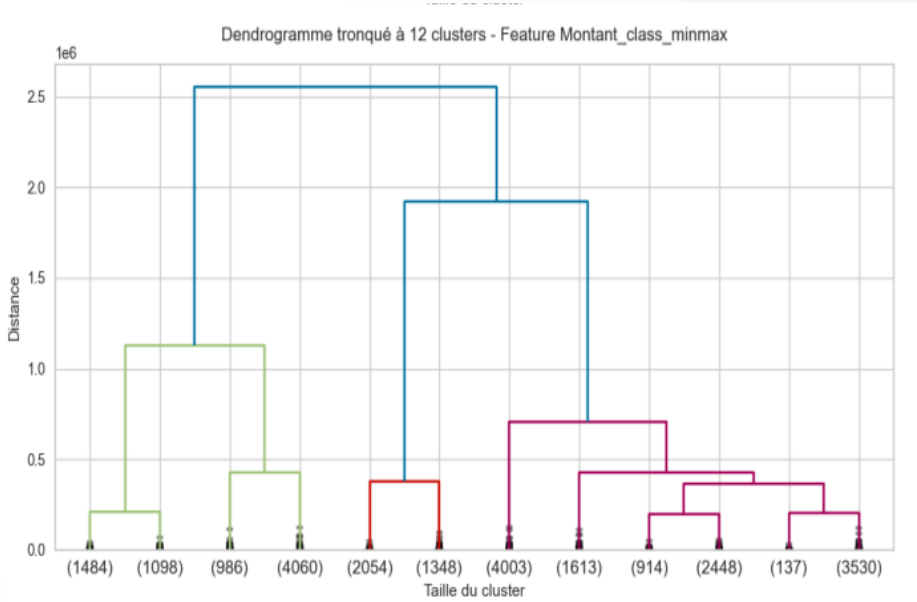
Algorithme de clustering hiérarchique qui considère chaque observation comme un cluster, et qui fusionne les clusters deux à deux jusqu'à n'en obtenir plus qu'un.

Fonctionnement :

- Au départ, chaque point est considéré comme un cluster individuel
- L'algorithme fusionne les clusters deux à deux en fonction de leur proximité
- Répétition de l'étape jusqu'à ce que le nombre de clusters désiré soit atteint ou que tous les points ne forment qu'un seul cluster

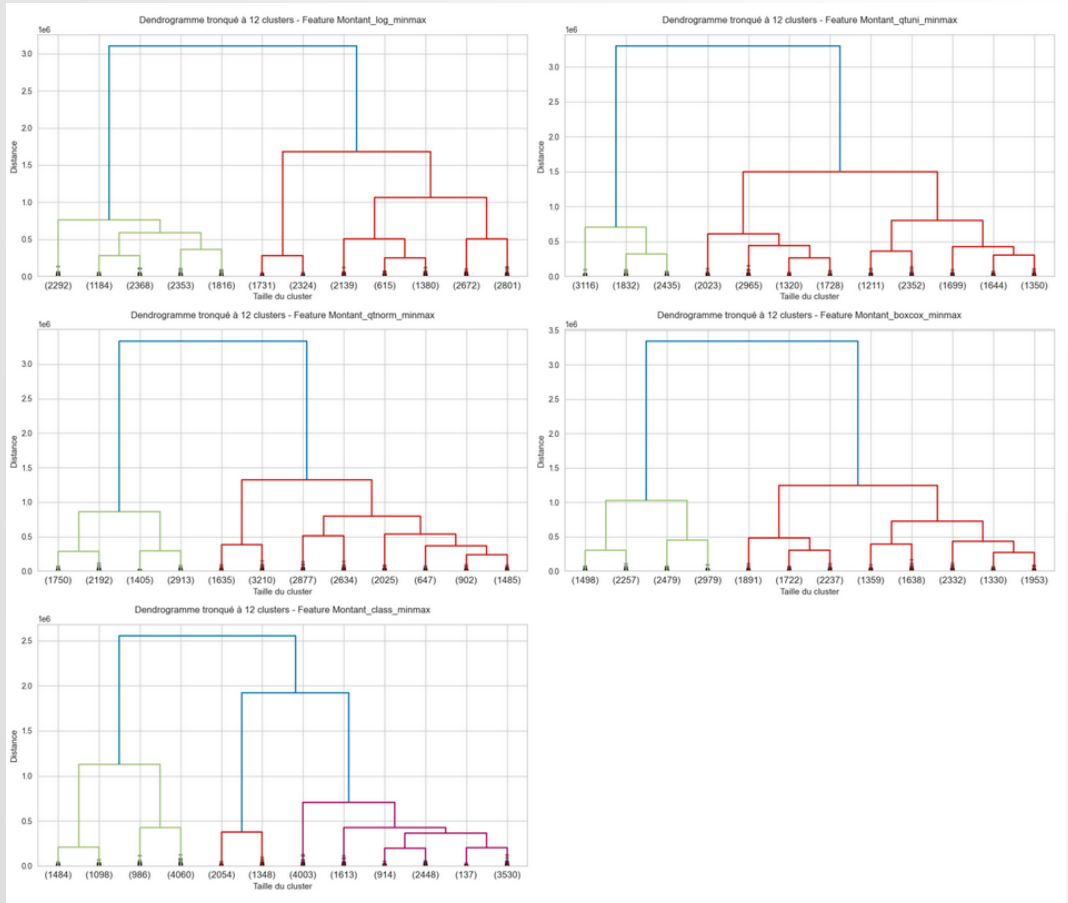
- Hyperparamètres :

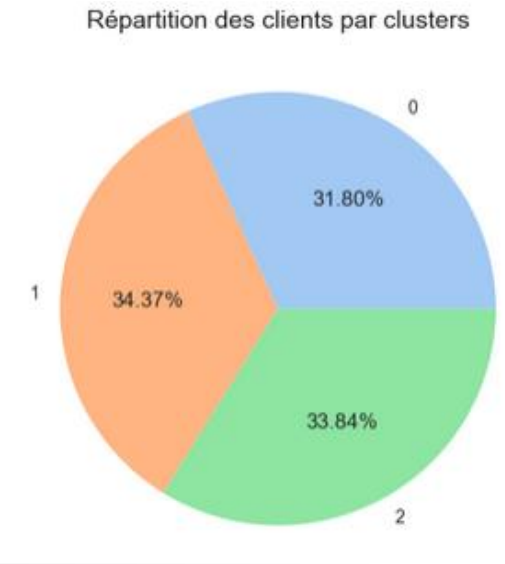
  linkage (ward) & **metric** (euclidean)

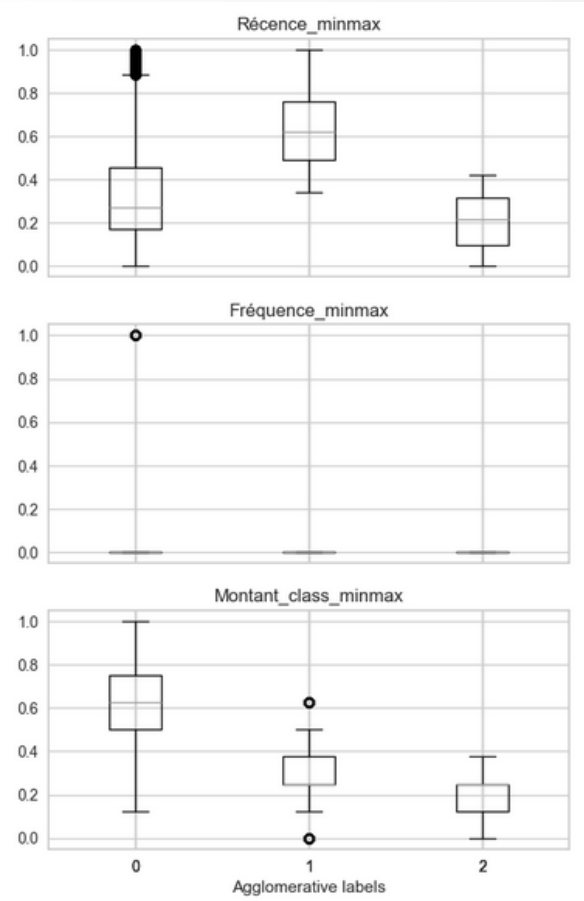


2 Les modèles

AGGLOMERATIVE CLUSTERING







2 Les modèles

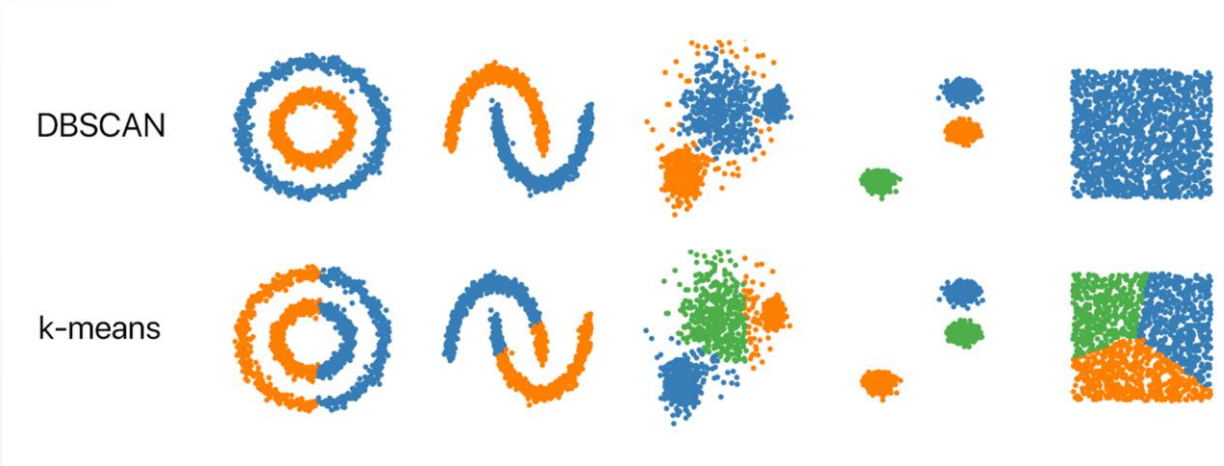
KMEANS

Algorithme : **Kmeans** Biblitohèque : **Scikit-learn**

Algorithme de clustering hiérarchique qui regroupe des ensembles de données en k groupes distincts basés sur des similarités. Particulièrement efficace pour identifier des clusters bien séparés si les données sont relativement bien distribuées.

Fonctionnement :

- L'algorithme commence par initialiser k points aléatoires appelés centroïdes, chacun représentant le centre initial d'un cluster
- Ensuite, chaque point du jeu de données est assigné au cluster dont le centroïde est le plus proche
- Les positions des centroïdes sont recalculées en tant que moyenne de tous les points assignés à leur cluster respectif
- Répétition du processus jusqu'à que les positions des centroïdes n'évoluent plus



Hyperparamètres : **n_clusters** & **init** (k-means++)

3

Le clustering

3 Le Clustering

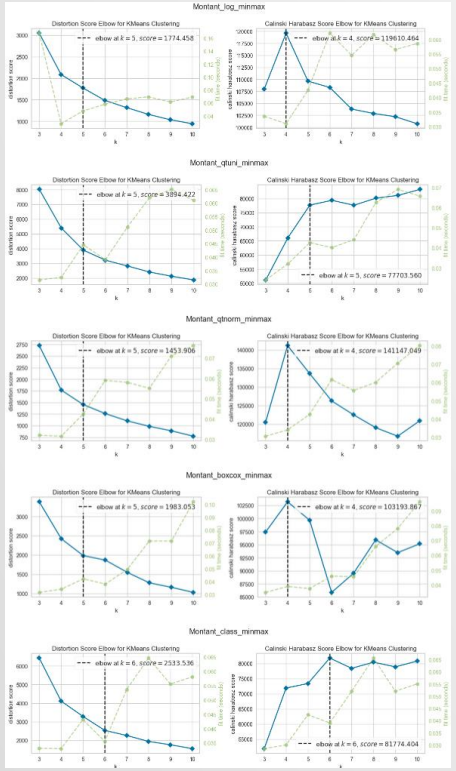
KMEANS

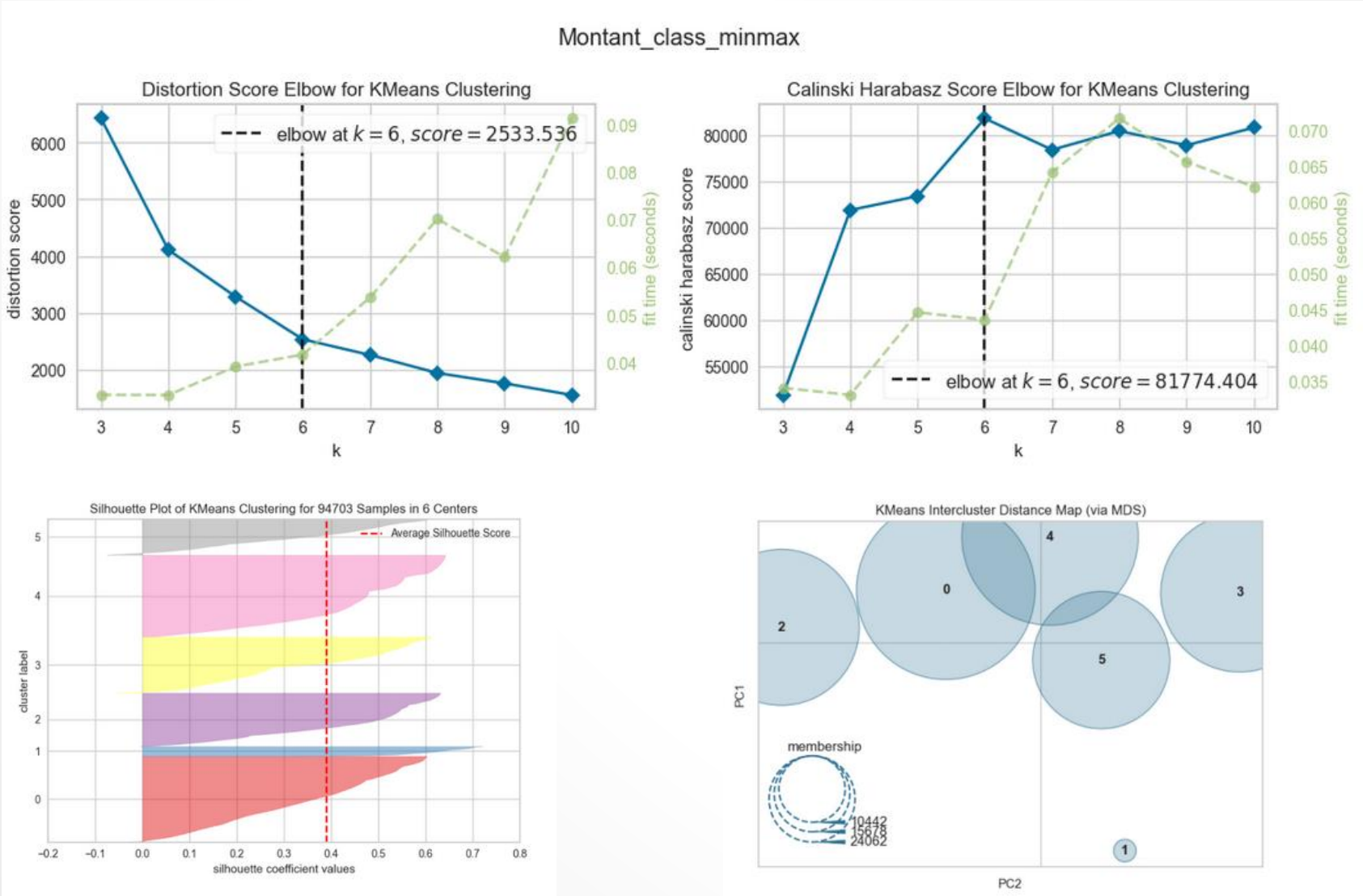
Recherche du nombre optimal de clusters :

n_clusters entre 3 et 10

init = ‘k-means++’

random_state = 0



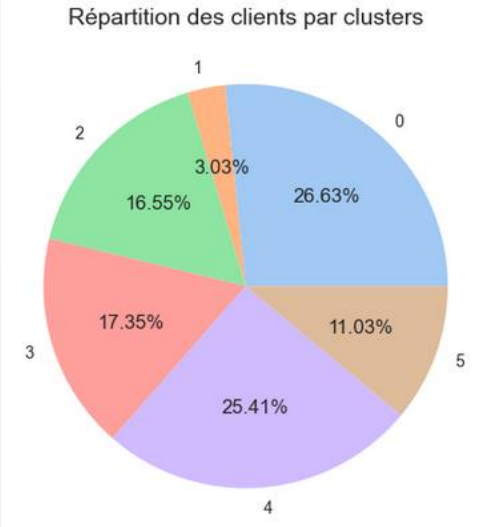


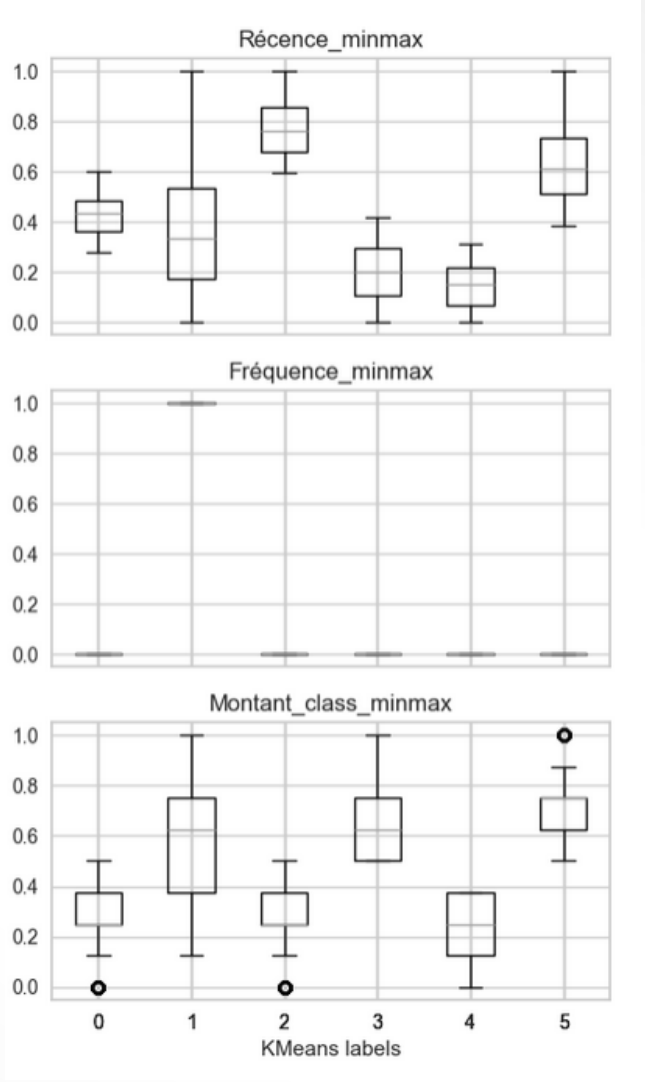
Pourcentage de variance expliquée par les 2 premières composantes principales : 72.26 %

3 Le Clustering

KMEANS

	KMeans labels	Client
0	0	25217
1	1	2874
2	2	15678
3	3	16430
4	4	24062
5	5	10442





- Cluster 0 : clients ayant commandé il y relativement longtemps (Récence moyenne), n'ayant passé qu'une seule commande (Fréquence à 0), ayant peu dépensé (Montant dans la tanche basse)
- Cluster 1 : clients ayant passé plusieurs commandes
- Cluster 2 : clients ayant commandé il y a longtemps, ayant passé une seule commande, dépense plus faible
- Cluster 3 : clients ayant commandé plus récemment, une seule commande, dépense plus élevée
- Cluster 4 : clients ayant commandé plus récemment, une seule commande, dépense faible
- Cluster 5 : clients ayant commandé il y a longtemps, une seule commande, dépense plus élevée

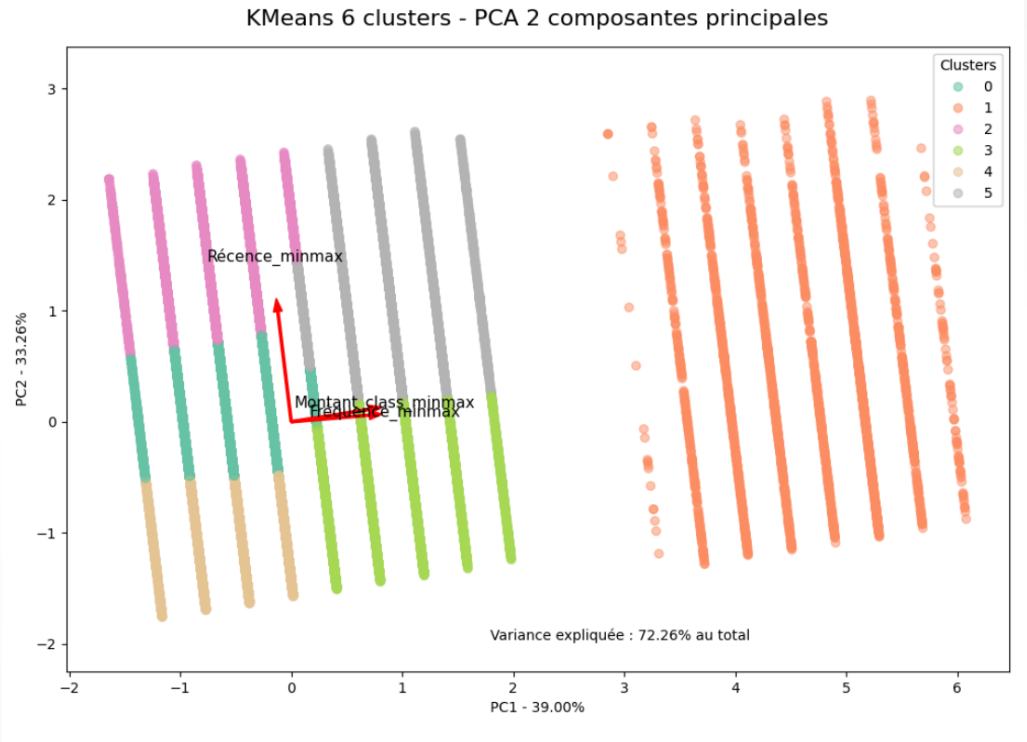
3 Le Clustering

KMEANS

L'algorithme a très bien réagi par rapport au classement des montants par tranches

	Récence_mois				Fréquence				Montant			Client	Percentage
	min	max	mean	median	min	max	mean	median	min	max	mean	median	count	
KMeans labels														
0	6.0	12.0	8.617480	9.0	0	0	0.0	0.0	10.07	199.95	87.080396	79.640	25217	26.627456
1	0.0	20.0	7.361169	7.0	1	1	1.0	1.0	35.94	7571.63	308.360480	225.620	2874	3.034751
2	12.0	20.0	15.362419	15.0	0	0	0.0	0.0	11.63	199.98	85.041647	77.570	15678	16.554914
3	0.0	8.0	4.014851	4.0	0	0	0.0	0.0	150.00	7274.88	328.044922	226.180	16430	17.348975
4	0.0	6.0	2.945516	3.0	0	0	0.0	0.0	9.59	149.89	78.464869	73.610	24062	25.407854
5	8.0	20.0	12.646045	12.0	0	0	0.0	0.0	150.00	13664.08	382.157604	266.695	10442	11.026050

```
# Définition des bornes pour les intervalles de classement
bins = [0, 25, 50, 100, 150, 200, 250, 500, 1000, float('inf')]
```



3 Le Clustering

KMEANS

Le VIP — 3 % des clients

8,34 / 10

- A passé plusieurs commandes
- 50 % des VIP ont dépensé plus de 225 $R et jusqu'à plus de 7 500 $R

Le Récent à fort potentiel — 17 %

8,40 / 10

- N'a passé qu'une commande
- Il y a moins de 8 mois
- A dépensé plus de 150 $R

Le Nouveau économe — 25 %

8,47 / 10

- N'a passé qu'une commande
- Il y a moins de 6 mois
- A dépensé moins de 150 $R

L'Occasionnel — 17 %

8,40 / 10

- N'a passé qu'une commande
- Il y a plus d'1 an
- A dépensé moins de 200 $R

L'Ancien de valeur — 11 %

8,12 / 10

- N'a passé qu'une commande
- Il y a plus de 8 mois
- A dépensé plus de 150 $R et jusqu'à plus de 10 000 $R

Le Dormant — 27 %

8,12 / 10

- N'a passé qu'une commande
- Entre 6 mois et 1 an
- A dépensé moins de 200 $R

4

La maintenance

4 Simulation maintenance

DÉMARCHE

« Est-ce qu'un modèle entraîné à la date D0 ferait une segmentation pertinente des clients à la date D1 ? »

D0 — D1

Calculer les features RFM jusqu'à D0

Les transformer (MinMaxScaler…)
Et conserver les transformers : **transformers_D0**

Entrainer le modèle et le conserver : **model_D0**

Calculer les features RFM : **RFM_D1**

Les transformer (MinMaxScaler…)

Appliquer le modèle

Récupérer les labels de clusters : **labels_D1**

Récupérer les **RFM_D1**

Les transformer avec les **transformers_D0**

Récupérer les labels par un predict
labels_simul_D1 = model_D0. predict(transformer_D0.transform(RFM_D1))

ari_score = adjusted_rand_score(**labels_simul_D1**, **labels_D1**)

RI : Random index

ARI : Adjusted Random index

$$ARI = \frac{RI - Expected\ RI}{\max(RI) - Expected\ RI}$$

4 Simulation maintenance

SCORES ARI