2020

OPENCLASSROOMS

Parcours Data-Scientist – Projet 5

CentraleSupélec

SEGMENTEZ DES CLIENTS D'UN SITE E-COMMERCE

CHANGEMENTS

- OBLIGATOIRES :
  - Respect de la convention PEP8
  - Ajout d'une étude de stabilité temporelle (k–means)

- OPTIONNELS :
  - Ajout d'une étude de stabilité à la réinitialisation (k–means)
  - Ajout d'un diagramme de Kiviat

SOMMAIRE

- Problématique
- Présentation des données (structure / contenu du dataset)
- Nettoyage des données – Feature engineering
- Segmentation RFM
- Analyse univariée – multivariée
- Clustering
- Conclusion et perspectives

PROBLEMATIQUE

olist

- **Historique :** base de données anonymisée comportant l'historique de commandes depuis janvier 2017
- **Problème :** comment adapter les campagnes de communication ?
- **Objectif :** comprendre les différents types d'utilisateurs pour mieux communiquer.
- **Méthodologie :**
  - utiliser des méthodes non supervisées pour regrouper ensemble des clients de profils similaires.

PRESENTATION DES DONNEES

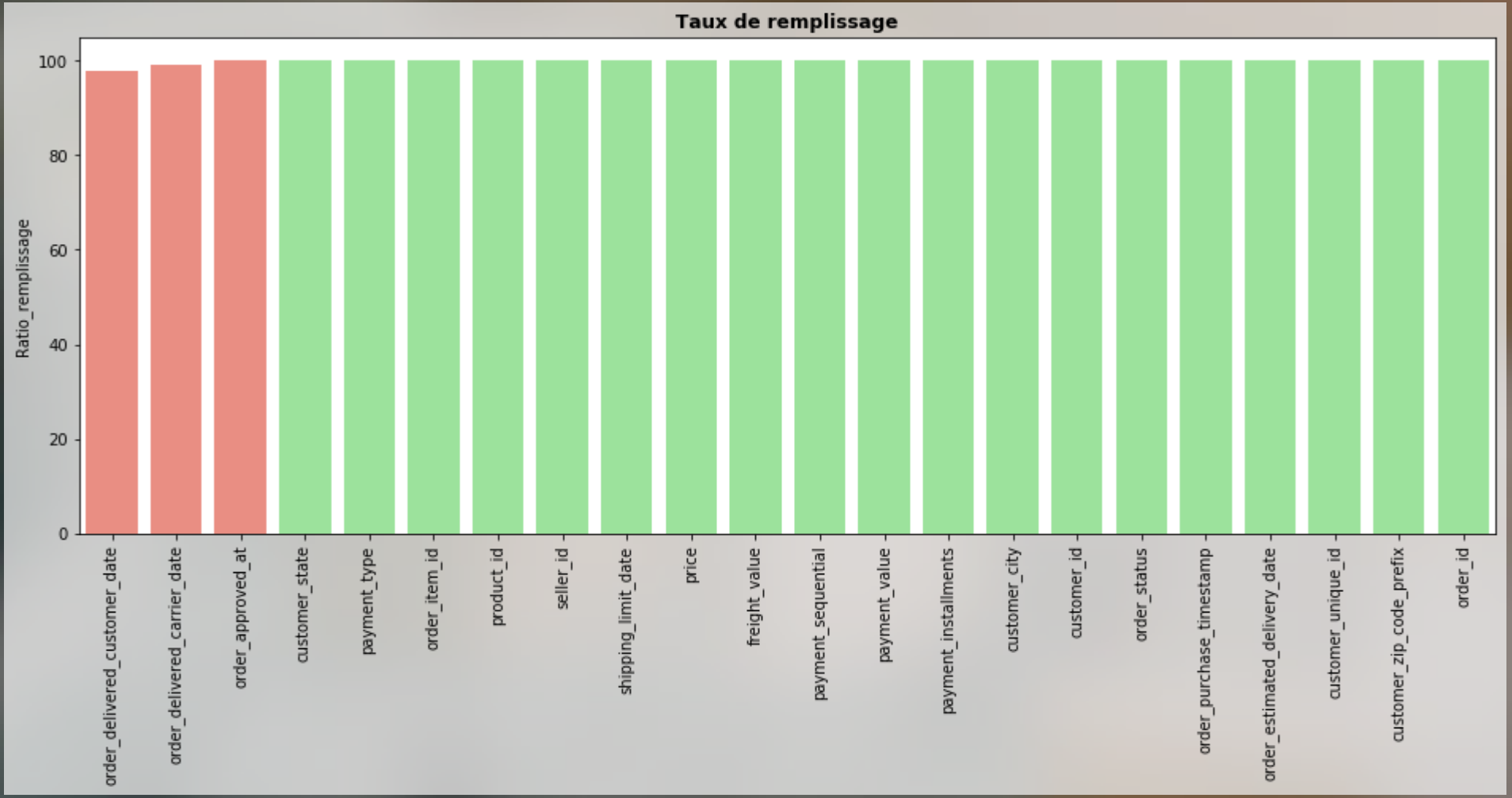
Structure du dataset, d'où proviennent les données ?

Contenu du dataset, taux de remplissage

STRUCTURE DU DATASET

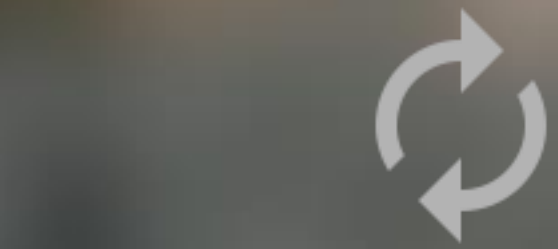
- **Jeu de données :** https://www.kaggle.com/olistbr/brazilian-ecommerce/download
- **Format de fichier :** .csv
- **Taille :** 120.3 MB
- **Nombre de colonnes / lignes :**
  - olist_customers_dataset.csv : 99441 lignes & 5 colonnes
  - olist_orders_dataset.csv : 99441 lignes & 8 colonnes
  - olist_order_items_dataset.csv : 112650 lignes & 7 colonnes
  - olist_order_payments_dataset.csv : 103886 lignes & 5 colonnes

CONTENU DU DATASET



- Majorité de colonnes remplies
- 3 colonnes avec peu de valeurs manquantes
  => imputation possible sans biais

NETTOYAGE DES DONNEES & FEATURE ENGINEERING



NETTOYAGE DES DONNEES – FEATURE ENIGEERING

- **Respect de la convention PEP8**

- PEP signifie Python Enhancement Proposal, et il en existe plusieurs dont PEP8 (https://www.python.org/dev/peps/pep-0008/)
- Objectif : définir des règles de développement communes entre développeurs.
- Avantages : améliorer la lisibilité et la cohérence du code Python

```
# Commencement de la vérification PEP8 :
%pycodestyle_on
```

```
1:41: W291 trailing whitespace
1:41: W291 trailing whitespace
```

```
# Pour l'affichage des erreurs :
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.simplefilter("ignore")

# Pandas
import pandas as pd
pd.set_option('display.max_columns', 100)
```

```
7:1: E402 module level import not at top of file
7:1: E402 module level import not at top of file
```

NETTOYAGE DES DONNEES – FEATURE ENIGEERING

4 étapes principales de Nettoyage

2 étapes de Feature Engineering

NETTOYAGE DES DONNEES – FEATURE ENIGEERING

Fusion des datasets

Format des colonnes

Suppression des doublons

Imputation des valeurs manquantes

Ajout de variables

```
# Fusion des dataframes :

df_tmp1 = dfi.merge(dfp, on='order_id')
df_tmp2 = df_tmp1.merge(dfo, on='order_id')
df = df_tmp2.merge(dfc, on='customer_id')
```

NETTOYAGE DES DONNEES – FEATURE ENIGEERING

Fusion des datasets

Format des colonnes

Suppression des doublons

Imputation des valeurs manquantes

Ajout de variables

```
# Conversion en objet :
df[['order_item_id', 'customer_zip_code_prefix']] = df[['order_item_id', 'customer_zip_code_prefix']].astype(object)

# Conversion en date :
import datetime

df[['shipping_limit_date', 'order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date', 'order_del
```

NETTOYAGE DES DONNEES – FEATURE ENIGEERING


```
# Suppression des doublons
# ATTENTION : ne pas oublier .reset_index() sinon il efface en conservant les index de lignes effacées :
rfm = rfm.drop_duplicates().reset_index()
```

NETTOYAGE DES DONNEES – FEATURE ENIGEERING

Fusion des datasets → Format des colonnes → Suppression des doublons → **Imputation des valeurs manquantes** → Ajout de variables

```
# On ajoute donc ces valeurs au valeurs manquantes (les valeurs étant négatives, on rajoute un signe '-'):

df['order_approved_at'] = df['order_approved_at'].fillna(df['order_purchase_timestamp'] + (-df['Dif_purch_approval'].mean()))
df['order_delivered_carrier_date'] = df['order_delivered_carrier_date'].fillna(df['order_purchase_timestamp'] + (-df['Dif_purch_del_carr'].mean()))
df['order_delivered_customer_date'] = df['order_delivered_customer_date'].fillna(df['order_purchase_timestamp'] + (-df['Dif_purch_del_cust'].mean()))
```

NETTOYAGE DES DONNEES – FEATURE ENIGEERING

Fusion des datasets

Format des colonnes

Suppression des doublons

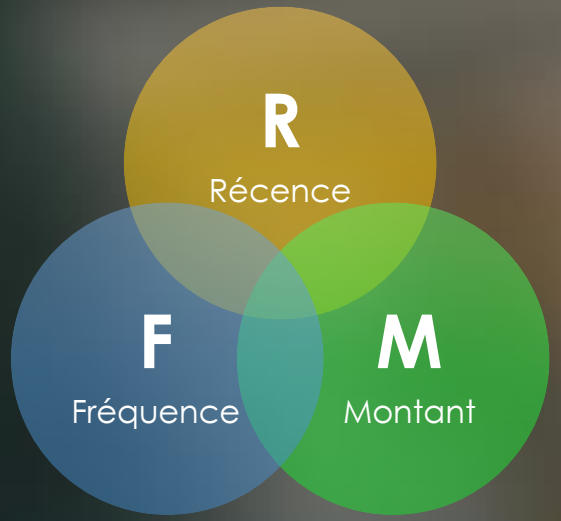
Imputation des valeurs manquantes

Ajout de variables

```
# Ajout de la colonne Récence :
df['Récence'] = (df['Récence_date'].max() - df['Récence_date']).dt.days


# Ajout de la colonne Fidélité (en 3 étapes) :
df['Premier_achat_date'] = df.groupby('customer_unique_id')['order_purchase_timestamp'].transform('min')
df['Ancienneté'] = (df['Récence_date'].max() - df['Premier_achat_date']).dt.days
df.drop('Premier_achat_date', axis=1, inplace=True)

# Ajout de la colonne Montant_m :
df['Panier_m'] = df.groupby('customer_unique_id')['payment_value'].transform('mean')

# On peut vérifier en cherchant les doublons de customer_unique_id :
df[df['customer_unique_id'].duplicated()]
```

SEGMENTATION RFM

SEGMENTATION RFM



- Classement des clients en fonction de leurs habitudes d'achat.
- Permet d'optimiser une stratégie marketing en fonction des clients.
- 3 critères :
  - **Récence** : date du dernier achat ou dernier contact client
  - **Fréquence** : fréquence des achats sur une période de référence donnée
  - **Montant** : somme des achats cumulés sur cette période

Sources :
https://fr.wikipedia.org/wiki/R%C3%A9cence%2C_Fr%C3%A9quence%2C_Montant
https://www.e-marketing.fr/Definitions-Glossaire/-recence-frequence-valeur-238820.htm#GfAlbVX2J8XQHczW.97

```
# Ajout de la colonne Récence_date :
df['Récence_date'] = df.groupby('customer_unique_id')['order_purchase_timestamp'].transform('max')

# Ajout de la colonne Fréquence :
df['Fréquence'] = df.groupby('customer_unique_id')['order_purchase_timestamp'].transform('count')

# Ajout de la colonne Montant :
df['Montant'] = df.groupby('customer_unique_id')['payment_value'].transform('sum')
```

```
# Ajout de la colonne Récence :
df['Récence'] = (df['Récence_date'].max() - df['Récence_date']).dt.days


# Ajout de la colonne Fidélité (en 3 étapes) :
df['Premier_achat_date'] = df.groupby('customer_unique_id')['order_purchase_timestamp'].transform('min')
df['Ancienneté'] = (df['Récence_date'].max() - df['Premier_achat_date']).dt.days
df.drop('Premier_achat_date', axis=1, inplace=True)

# Ajout de la colonne Montant_m :
df['Panier_m'] = df.groupby('customer_unique_id')['payment_value'].transform('mean')
```

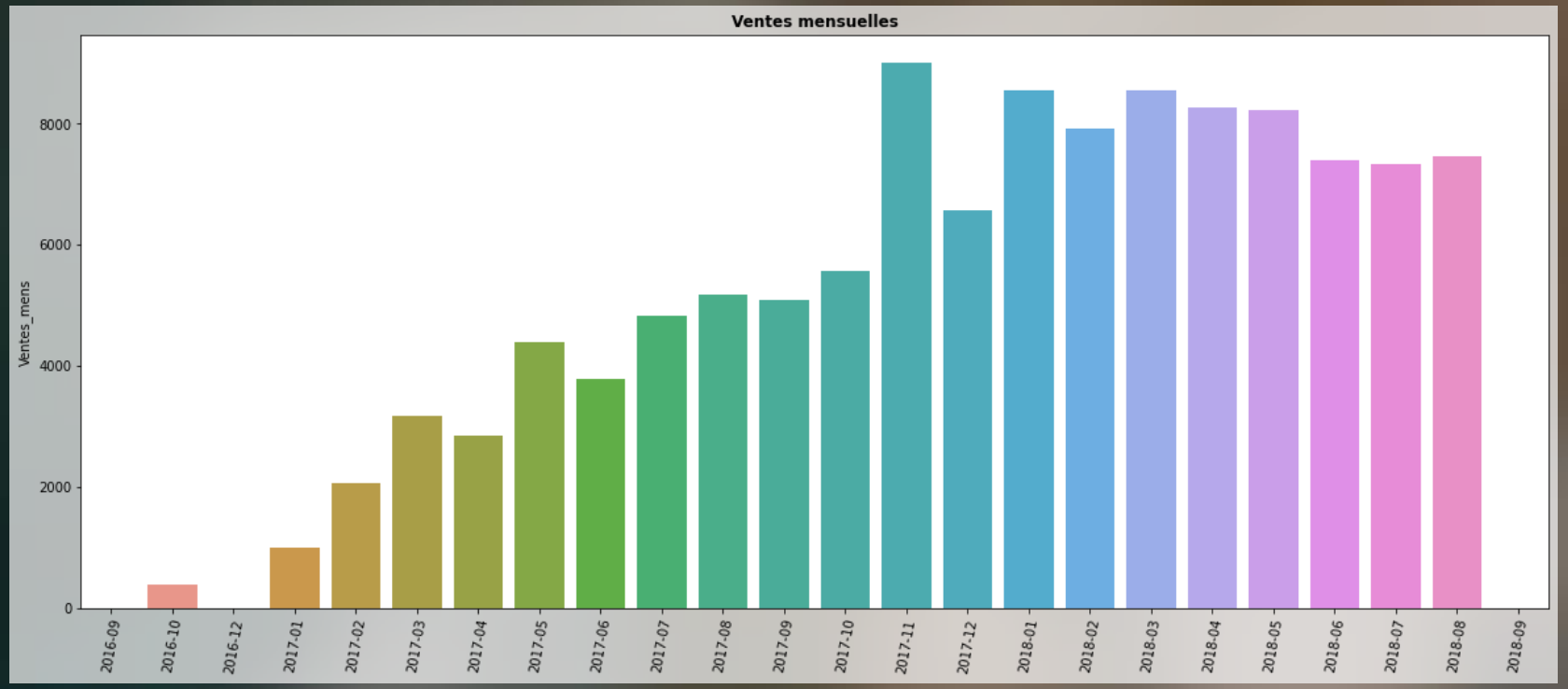
ANALYSE UNIVARIÉE – MULTIVARIÉE

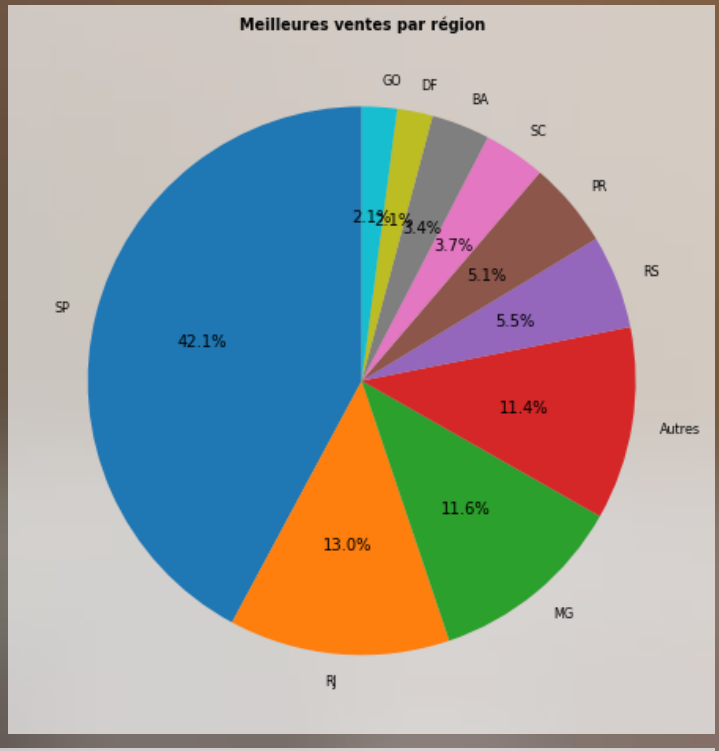
Quelques statistiques

Distribution des variables

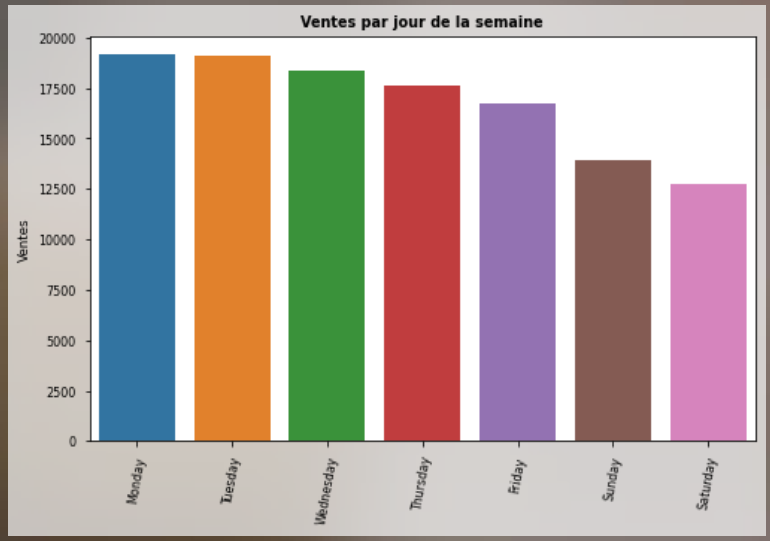
Corrélations de variables

ANALYSE UNIVARIÉE – MULTIVARIÉE

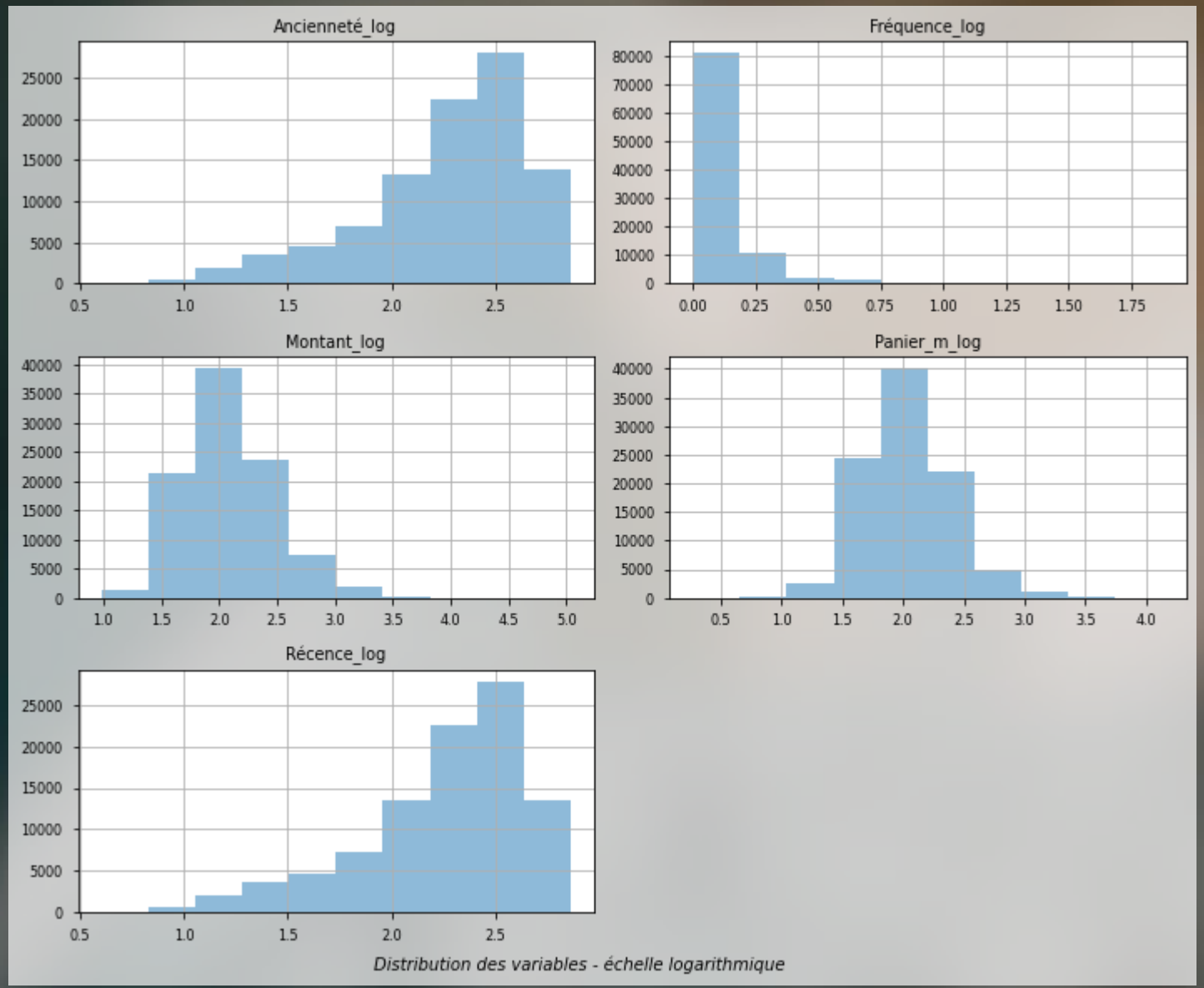




- Nombre de ventes mensuelles en croissance constante, pas de saisonnalité visible hormis une stagnation de Juin à Aout 2017 et 2018 (congés ?)
- Pic maximal en Novembre 2017 (fêtes de fin d'année ?)
- Meilleures ventes à Sao Polo (~majorité), Rio de Janeiro, et Minas Gerais
- Les achats diminuent avec les jours de la semaine, les week-ends sont les jours ayant le moins d'achat.
  => campagnes promotionnelles préférables les Lundi



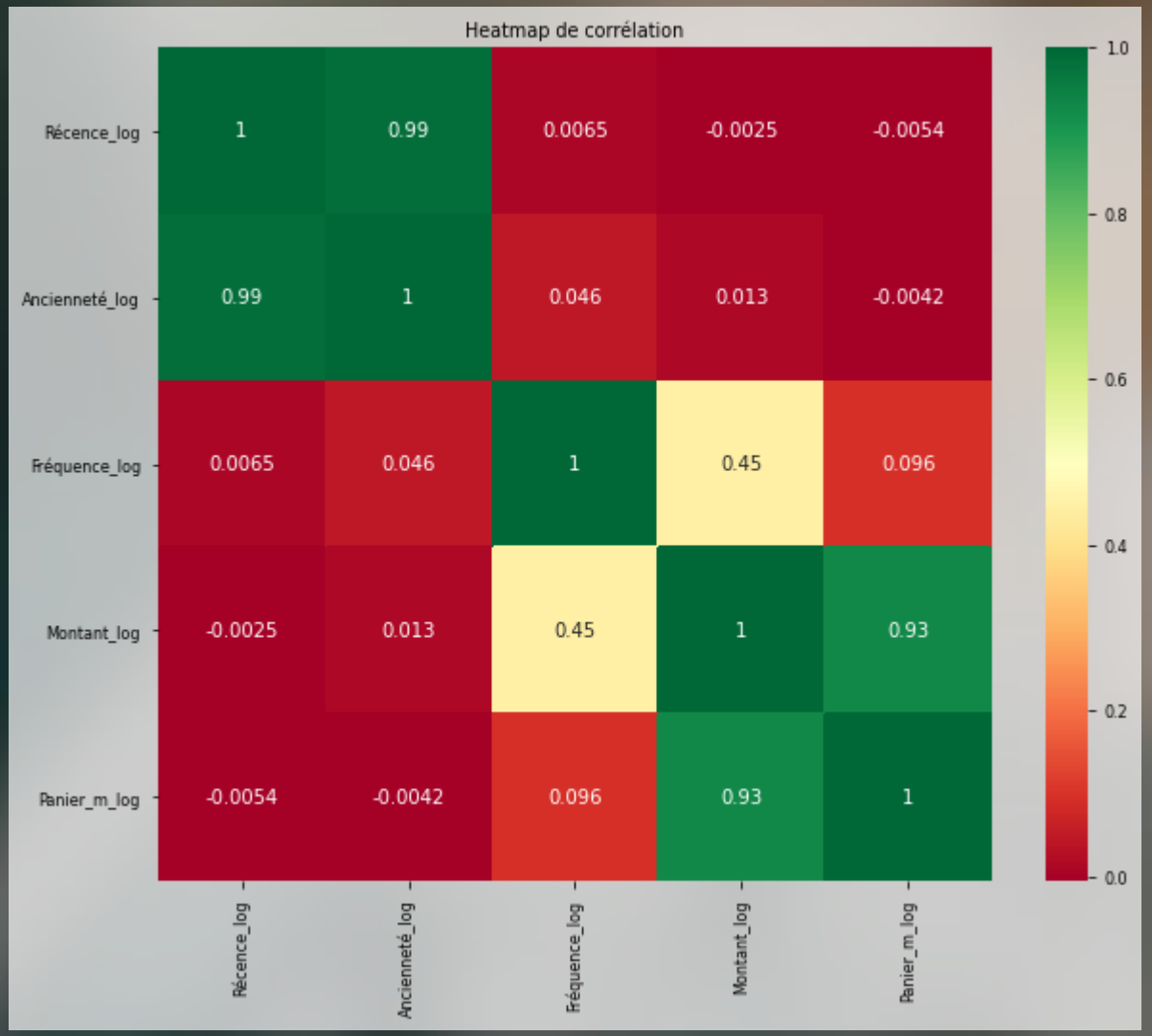
ANALYSE UNIVARIÉE – MULTIVARIÉE



Distribution des variables - échelle logarithmique

- On cherche à comprendre si la variable est unimodale, car la standardisation fonctionne bien avec les variables unimodales
- Passage en Log décimal nécessaire pour visualiser
- Peu de gaussiennes
- Variables unimodales
  => On confirme bien qu'il est possible de les standardiser.

ANALYSE UNIVARIÉE – MULTIVARIÉE



- Effectuée après standardisation des données
- Peu de corrélations utiles
- Quelques colinéarités

=> On peut se passer de certaines variables telles que Montant (le panier moyen étant plus révélateur) ou l'Ancienneté (pour conserver la notion de Récence).

CLUSTERING

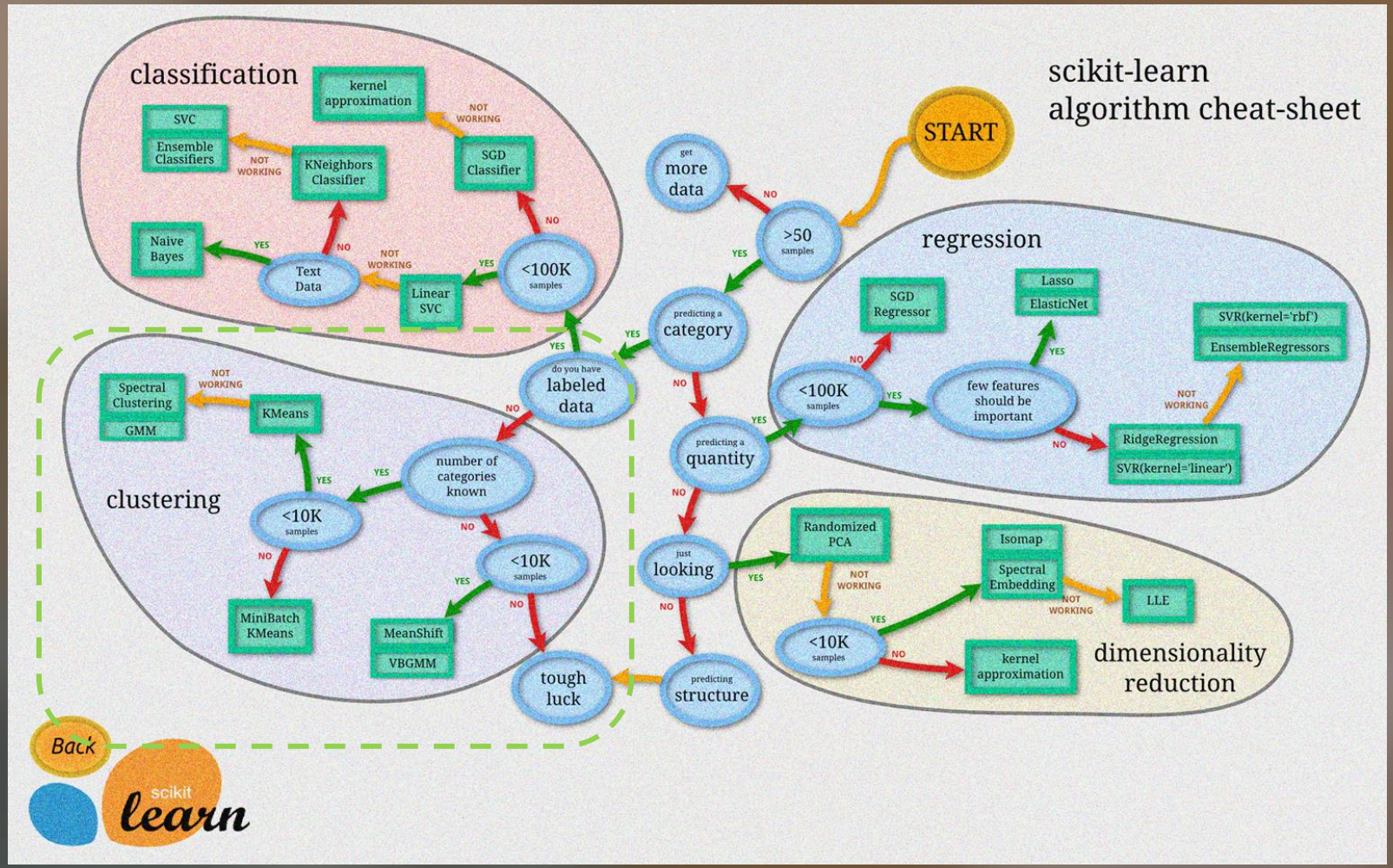
CLUSTERING

- On souhaite regrouper les clients en catégories basées sur leur RFM

Choix de l'algorithme :

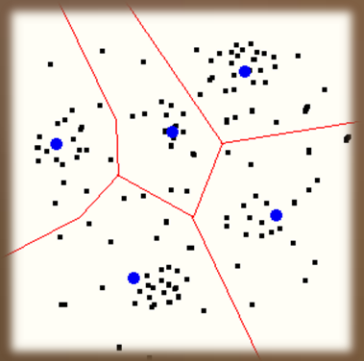
- Plus de 50 échantillons
- Les clusters sont des catégories
- Pas de labélisation (non supervisé)
- Nombre de catégories inconnue
- Echantillon supérieur à 10 000

- => Algorithme **Kmeans**



CLUSTERING

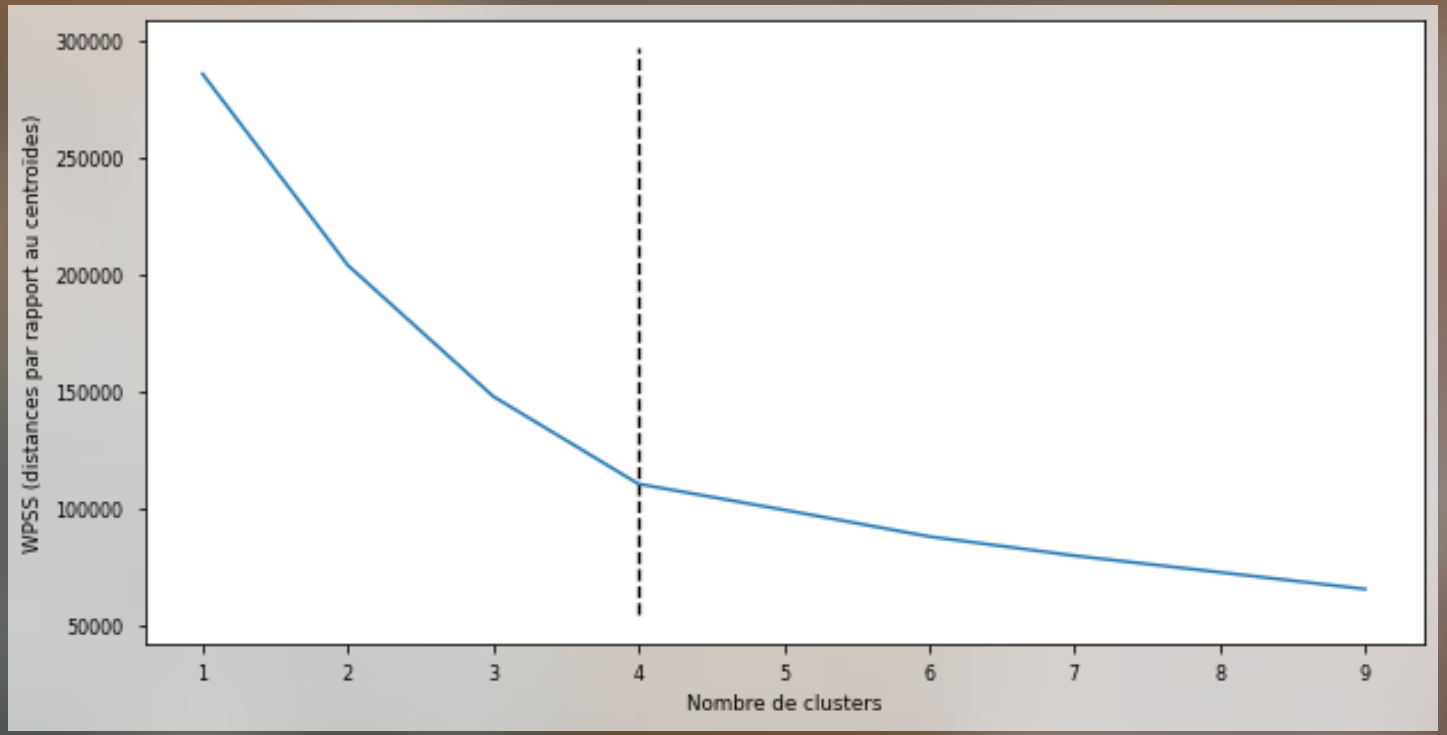
- On souhaite connaitre le nombre optimal de clusters
- Obtenir la meilleur inertie possible (i.e. convergence des différentes populations formant un cluster)
- Mathématiquement, on appelle cette inertie le WCSS (ou Within Cluster Sum of Squares) = somme des carrés des distances de chaque point par rapport à leurs centroïde respectif



Source : mnemstudio.org

```
# Importation de l'algorithme k-means :
from sklearn.cluster import KMeans

# Initialisation du WCSS sous la forme d'un dict(key: value) :
wcss={}

# Sélection des colonnes souhaitées pour le k-means :

rfm_k = rfm_s[['Récence_log', 'Fréquence_log', 'Panier_m_log']].copy()

# Entraînement :
# On se donne 10 comme nombre maximal de clusters, et 300 (par défaut) comme nombre max d'
# Chaque itération recalcule le centroïde le plus proche pour chaque point de données.
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=300).fit(rfm_k)
    rfm_k['Clusters'] = kmeans.labels_
    wcss[k] = kmeans.inertia_
```

- Utilisation de la **Méthode Elbow** : graphique de l'estimation du WCSS en fonction du nombre de clusters
- La modification des hyperparamètres a peu d'impact
- Nombre optimal de clusters : 4
- Cependant, 3 clusters reste envisageable !
- **=> Comparer les analyses avec 3 et 4 clusters**



CLUSTERING

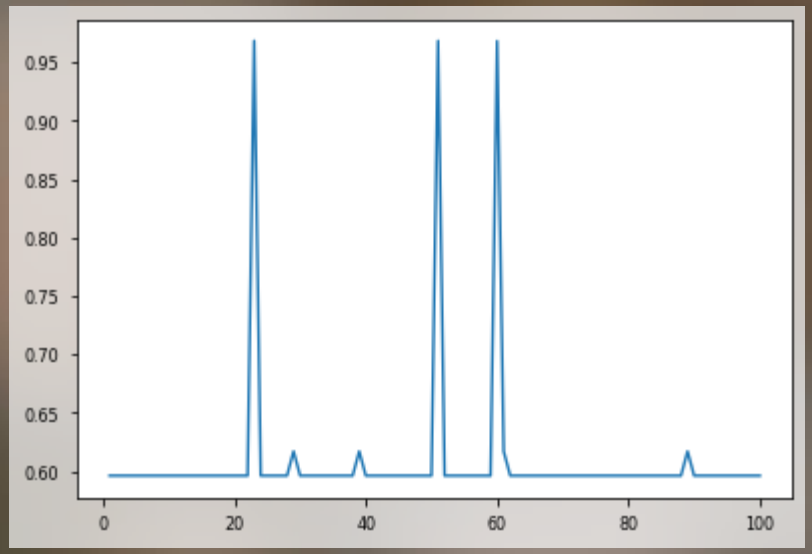
Stabilité de k-means à la réinitialisation:

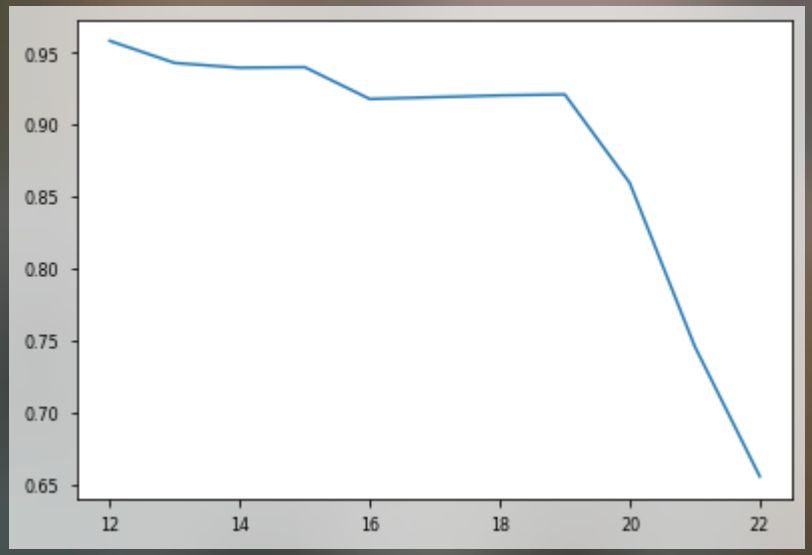
- Meilleure stabilité pour 4 clusters
- 3 pics importants pour 3 clusters, le reste étant stable.

Stabilité temporelle de k-means :

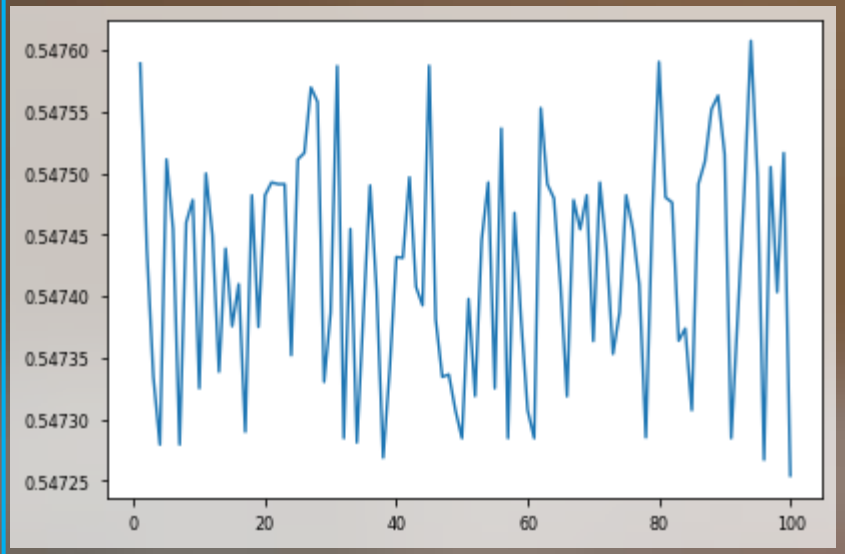
- Meilleure stabilité pour 4 clusters (20 mois) que pour 3 clusters (19 mois)
- Le contrat de maintenance doit donc couvrir moins de 20 mois afin de s'assurer de la fiabilité des résultats..

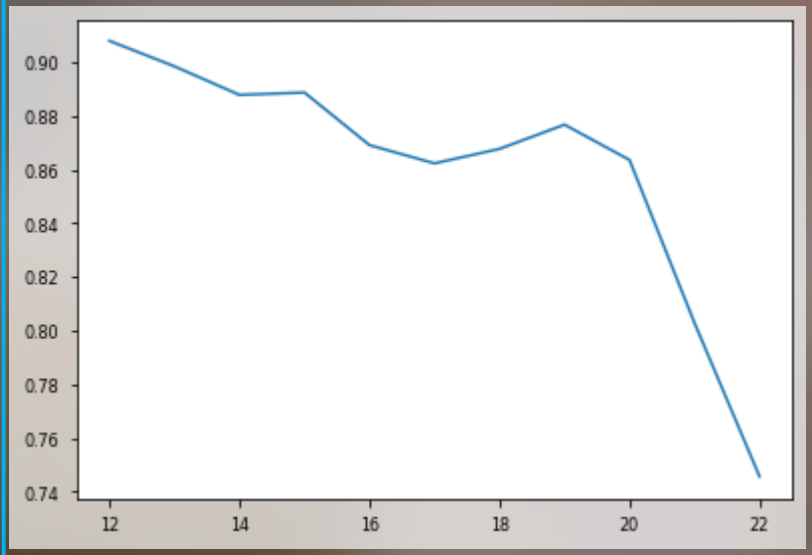
3 clusters



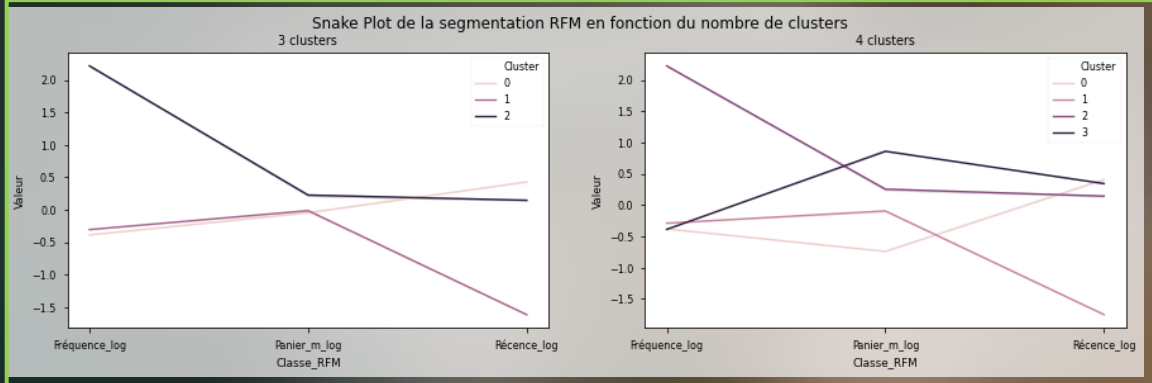


4 clusters



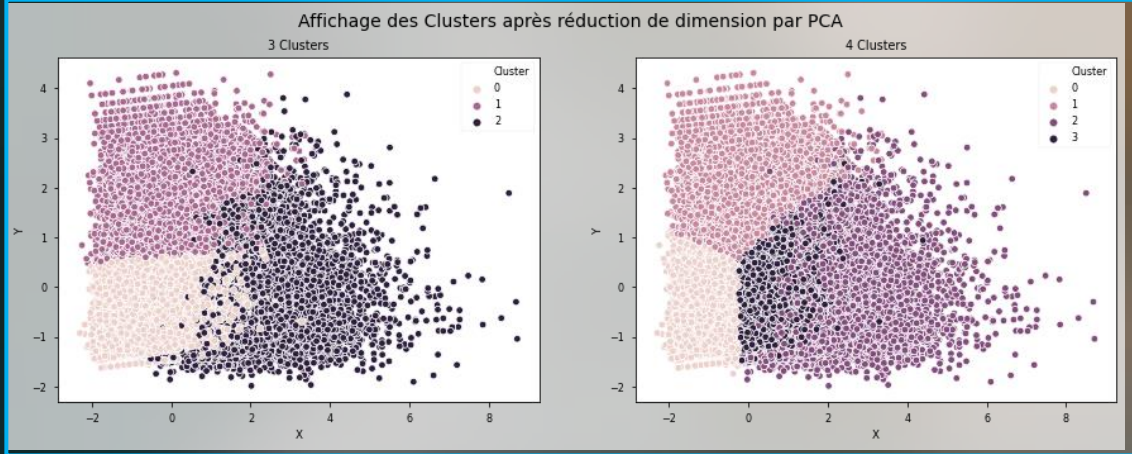


CLUSTERING



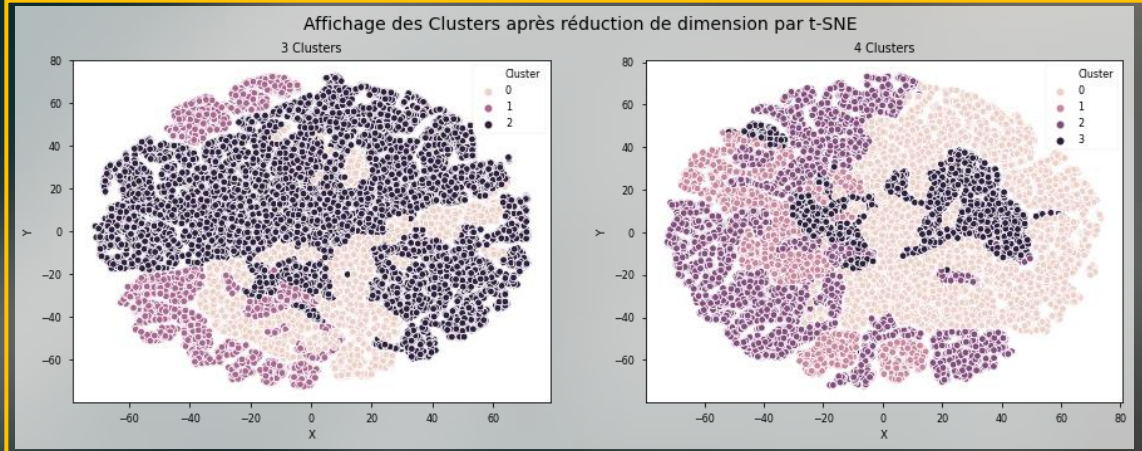
Snake plot :

- Technique issu d'étude de marché permettant de comparer visuellement différents segments
- Le cluster 0 du graphique à 3 clusters semble se décomposer en clusters 0 et 3 du graphique à 4 clusters.



Visualisation après PCA :

- Technique de réduction (linéaire) de dimensions
- Les clusters 0 et 2 du graphique à 3 clusters semblent former le cluster 3 du graphique à 4 clusters.

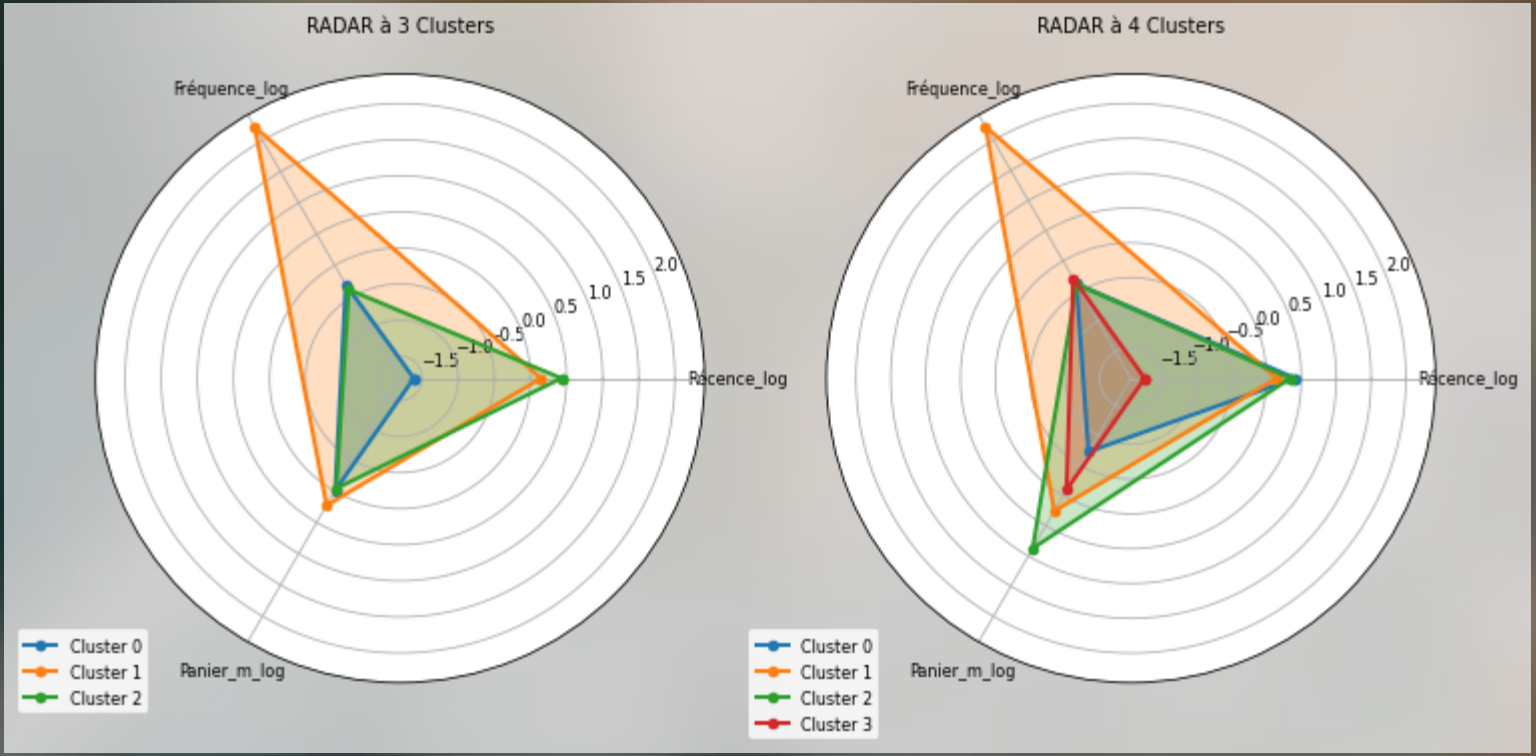


Visualisation après t–SNE :

- Technique de réduction (non–linéaire) de dimensions
- Le cluster 2 du graphique à 3 clusters semble former les cluster 0 et 2 du graphique à 4 clusters.

CLUSTERING

Diagramme de Kiviat:



- Meilleure visibilité qu'un Snake plot, notamment avec beaucoup de variables
- Le cluster 2 du graphique à 3 clusters semble se décomposer en clusters 0 et 3 du graphique à 4 clusters.

CLUSTERING

Interprétation

Cluster	0	1	2
Récence_min	4.000000	5.000000	103.000000
Récence_moy	50.261003	253.496149	294.442477
Récence_max	111.000000	728.000000	728.000000
Fréquence_min	1.000000	2.000000	1.000000
Fréquence_moy	1.077867	3.581403	1.000000
Fréquence_max	3.000000	75.000000	1.000000
Montant_min	9.590000	15.180000	10.070000
Montant_moy	168.168286	898.211550	150.205631
Montant_max	6922.210000	109312.640000	6929.310000

Cluster	0	1	2	3
Récence_min	60.000000	5.000000	24.000000	4.000000
Récence_moy	292.631455	252.903308	278.642245	44.241709
Récence_max	728.000000	728.000000	699.000000	111.000000
Fréquence_min	1.000000	2.000000	1.000000	1.000000
Fréquence_moy	1.006882	3.595930	1.000826	1.092809
Fréquence_max	2.000000	75.000000	2.000000	3.000000
Montant_min	10.070000	17.380000	109.750000	9.590000
Montant_moy	64.109033	902.278963	280.627538	142.606608
Montant_max	117.860000	109312.640000	9618.880000	3351.350000

Nbre de clusters	0	1	2	3
3 clusters	▪ viennent récemment sur le site ▪ concrétisent moins leurs visites par un achat ▪ payent moyennement cher que les autres consommateurs ▪ *=> Action marketing peu rentable*	▪ ne sont pas tous récemment venus sur le site ▪ concrétisent plus leurs visites par un achat ▪ payent plus cher que tous les autres consommateurs *=> Action marketing rentable*	▪ ne sont pas tous récemment venus sur le site ▪ concrétisent moins leurs visites par un achat ▪ payent moins cher que les autres consommateurs ▪ *=> Action marketing peu rentable*	
4 clusters	▪ ne sont pas tous récemment venus sur le site ▪ concrétisent moins leurs visites par un achat ▪ payent moins cher que les autres consommateurs ▪ *=> Action marketing peu rentable*	▪ ne sont pas tous récemment venus sur le site ▪ concrétisent plus leurs visites par un achat ▪ payent plus cher que tous les autres consommateurs *=> Action marketing rentable*	▪ ne sont pas tous récemment venus sur le site ▪ concrétisent moins leurs visites par un achat ▪ payent moyennement cher que les autres consommateurs ▪ *=> Action marketing moyennement rentable*	▪ viennent récemment sur le site ▪ concrétisent moins leurs visites par un achat ▪ payent moyennement cher que les autres consommateurs ▪ *=> Action marketing moyennement rentable*

CONCLUSION ET PERSPECTIVES

CONCLUSION ET PERSPECTIVES

Il a été effectué :

- Nettoyage d'un jeu de données et du Feature engineering (respect de PEP8)
- Segmentation en fonction de 3 critères : Récence, Fréquence, Montant
- Analyse univariée et multivariée de ces données (statistiques, distributions, corrélations & colinéarité, etc.)
- Un algorithme de clustering (Kmeans) et ses visualisations (Snake Plot, PCA, t-SNE), et l'étude de sa stabilité

Perspectives pour Olist:

- Investir dans une campagne marketing pour :
  - Segmentation à 3 clusters : les clients appartenant au cluster 1
  - Segmentation à 4 clusters (préférable) : les clients appartenant au clusters 1 et 2
- Déclencher cette campagne un Lundi
- Prioriser les clients de Sao Paulo
- Effectuer une maintenance tous les moins de 20 mois