Projet 5 : Segmentez des clients d'un site e-commerce

Problématique

Problématique : comprendre les différents types d’utilisateurs grâce à leur comportement et à leurs données personnelles.

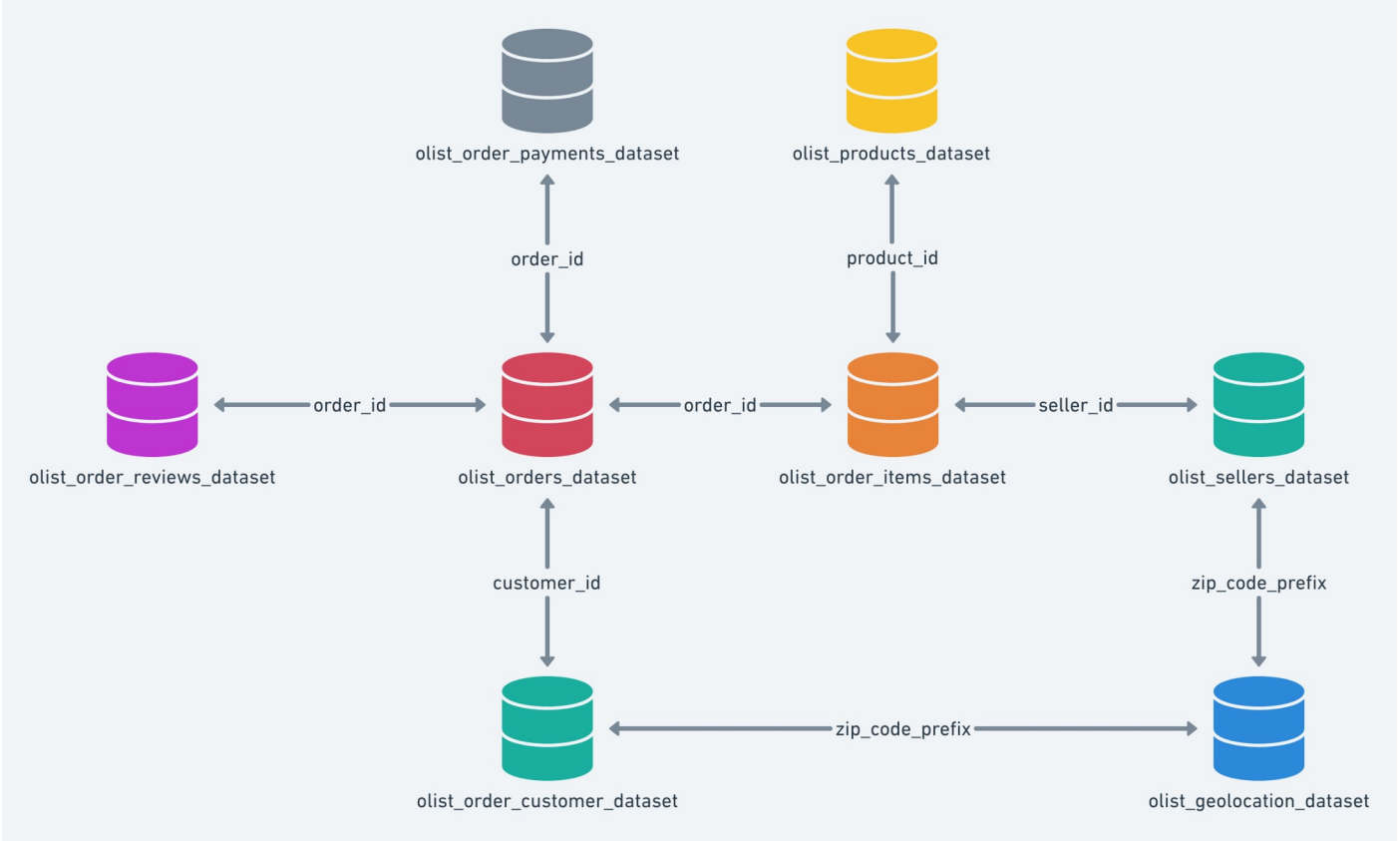
Faire une description actionnable de votre segmentation et de sa logique sous-jacente pour une utilisation optimale

Faire une proposition de contrat de maintenance basée sur une analyse de la stabilité des segments

Préparation des données

- OLIST publiait les données utilisées sur le site Kaggle.
- Ces données correspondent à 100 000 commandes effectuées entre 2016 et 2018.
- Les données récupérées consistait en 9 datasets contenant diverses informations sur les produits vendus, les vendeurs, les clients, les reviews_score de chaque achat, les dates de ventes…
- Les données étaient fournies avec un plan permettant de trouver la clef reliant chaque dataset, ainsi qu'une description succincte de chaque dataset.

Préparation des données



Préparation des données

- On a commencé par le dataset data_customer qui contenait une liste des clients et des commandes qu'ils avaient effectuées.
- On l'a fusionné avec data_orders, data_order_item et data_order_reviews, pour obtenir le coût de chaque commande, les dates associé à chaque commande, et le review_score associé.
- On a utilisé uniquement les commandes qui avaient été délivrées.
- On a converti les colonnes temporelles en date_time, et on a créé la colonnes wait_time et delay_vs_expected.

Préparation des données

- On a regroupé les données pour avoir des informations par client et non plus par commande.
- On a ainsi récupéré :

- la date de la première et dernière commande

- le nombre de commandes par client (fréquence)

- les dépenses du client (montant total)

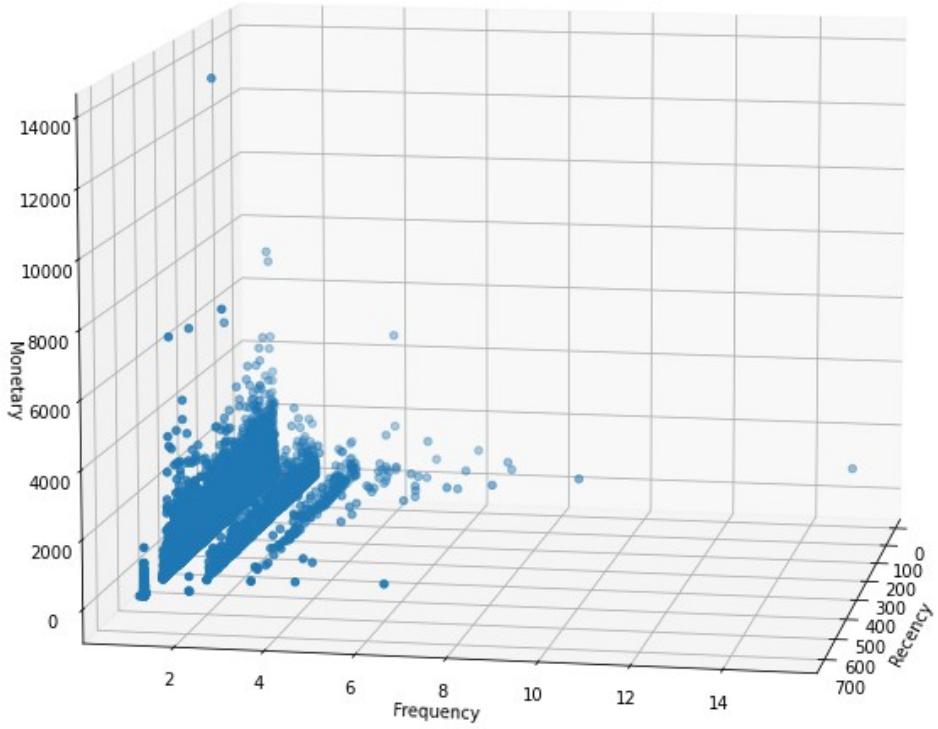
- le temps d'attente moyen, et le délai attendu contre temps d'attente réel (moyen)

- la récence du client

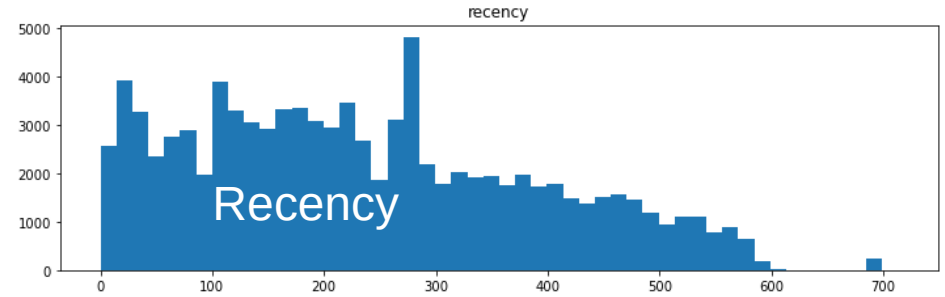
- le temps passé sur la plateforme

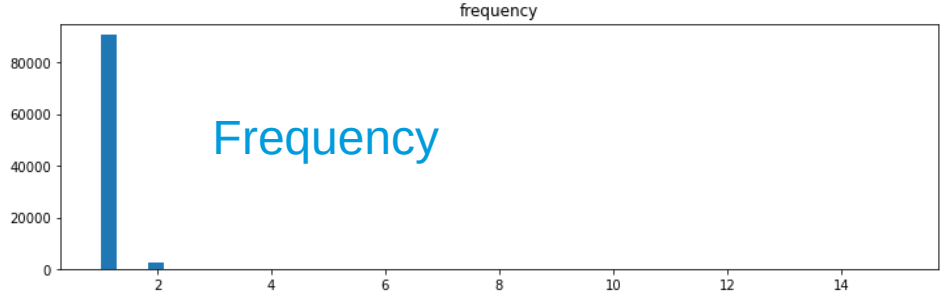
Exploration des données

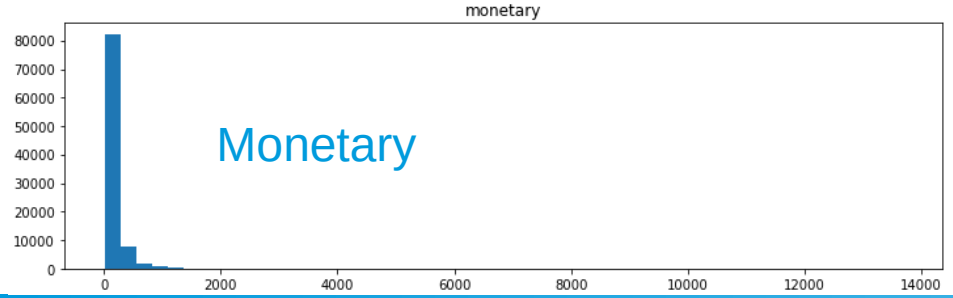
- On a utilisé la RFM pour représenter nos données :

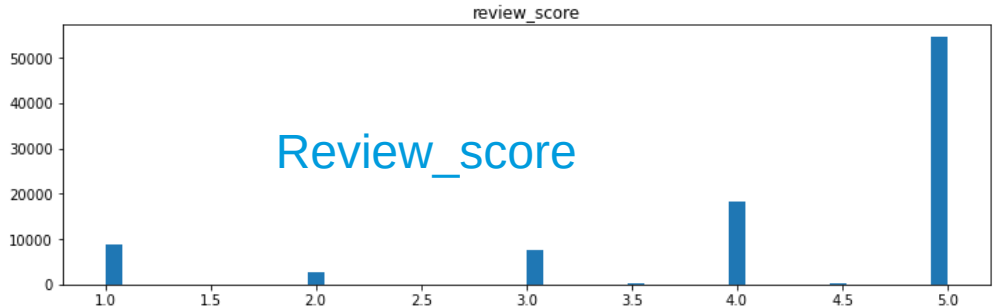


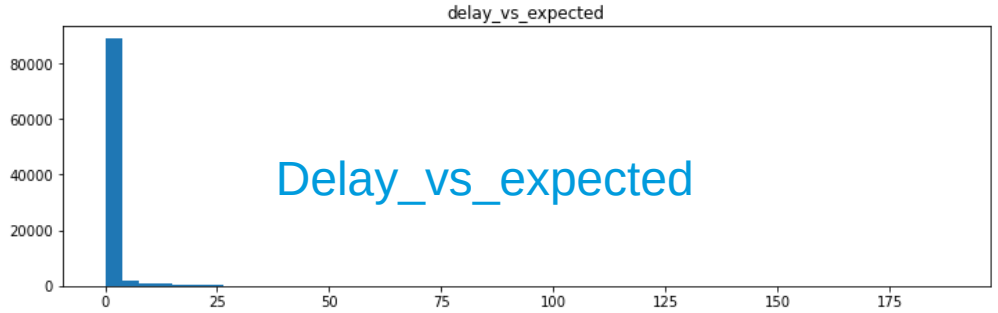
Exploration des données







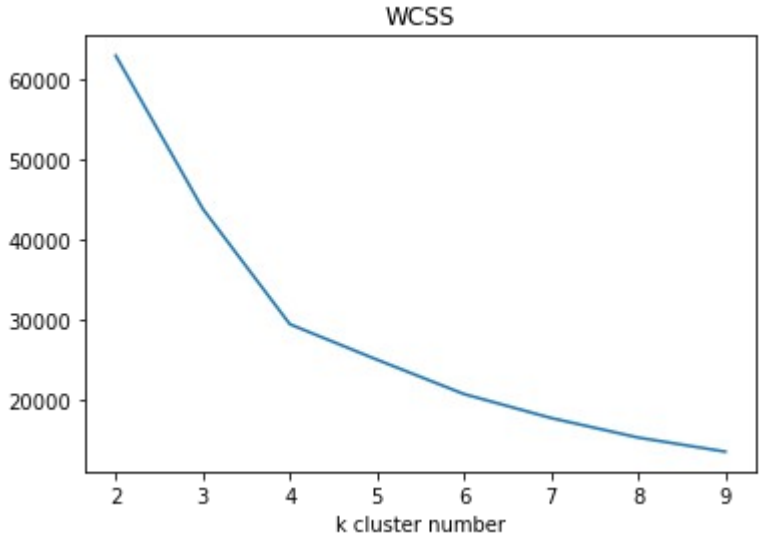


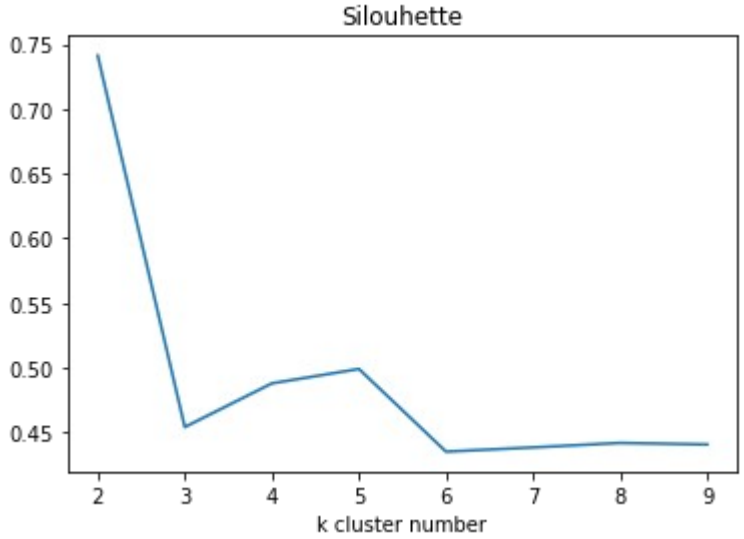


Modélisation - K_means

- On commence nos modélisations par K_means avec les features RFM.
- On normalise nos données pour qu'elles soient à la même échelle .
- On teste le nombre de kernels entre 2 et 9 et on regarde l'évolution du score de l'Inertia et de la Silouhette.

Modélisation - K_means





Modélisation - K_means

Nombre de cluster : 5

- [[0 1 2 3 4]
- [49564 36075 2777 4397 545]]
- Pourcentage :
- [53.09 38.64 2.97 4.71 0.58]

- Nombre de cluster : 4
- [[0 1 2 3]
- [37190 51019 2376 2773]]
- Pourcentage :
- [39.84 54.65 2.55 2.97]

Nombre de cluster : 3

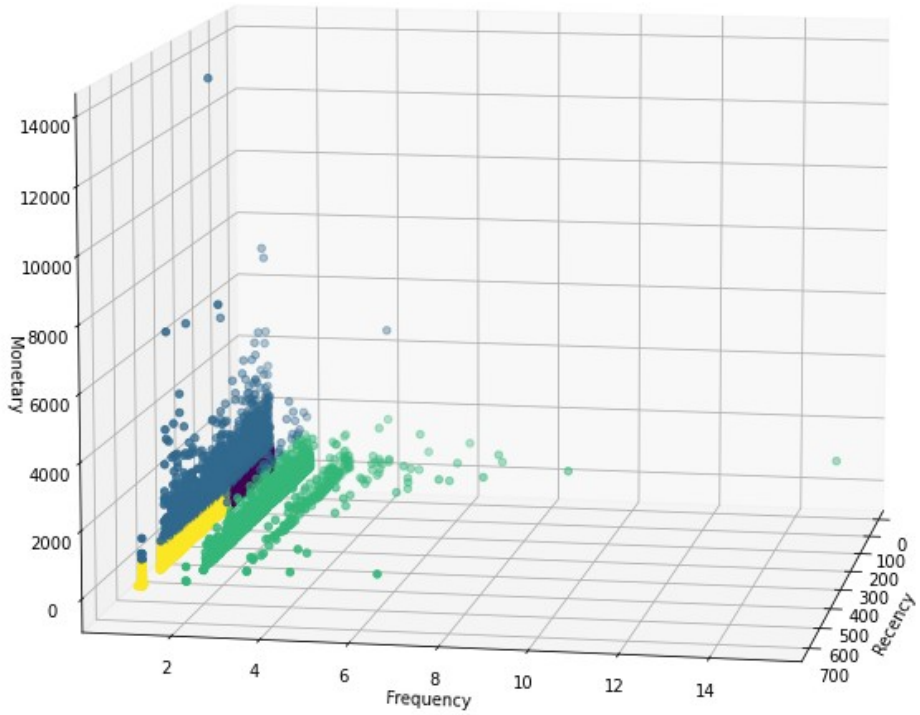
- [[0 1 2]
- [52273 38273 2812]]
- Pourcentage :
- [55.99 41. 3.01]

Modélisation - K_means

- Avec 5 clusters, on a un cluster très petit qui ne nous apportera pas beaucoup d’informations.
- On choisit d’utiliser 4 clusters.
- On initialise un K_means à 4 clusters avec 50 % et 60 % des données.
- Le score ARI obtenu entre les 2 modèles est supérieur à 80 %
- Le modèle est stable.

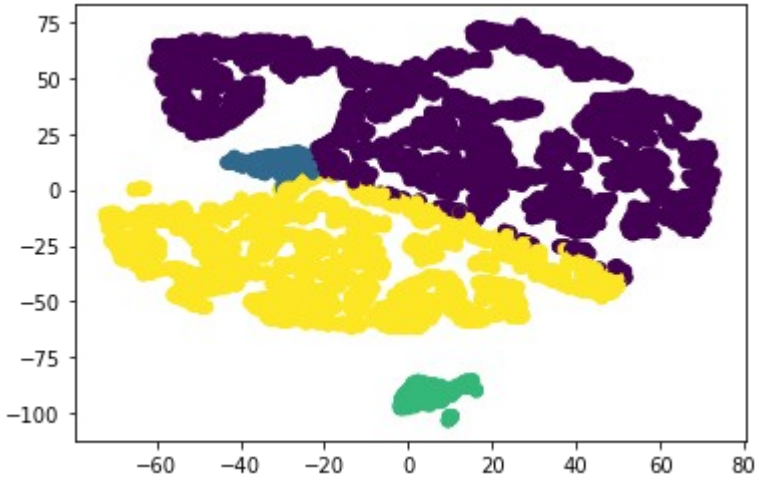
Modélisation - K_means

- On représente graphiquement notre classification avec 4 clusters :



Modélisation - K_means

- On fait également une représentation en 2d avec une t-SNE :



Modélisation - K_means

- En étudiant la moyenne des valeurs de chaque feature et leur répartition, on en déduit sur quel critère le K_means à créer les différentes catégories :

  Feature 0 : Client ayant passé une seule commande, récente, et ayant fait des dépenses modérées.

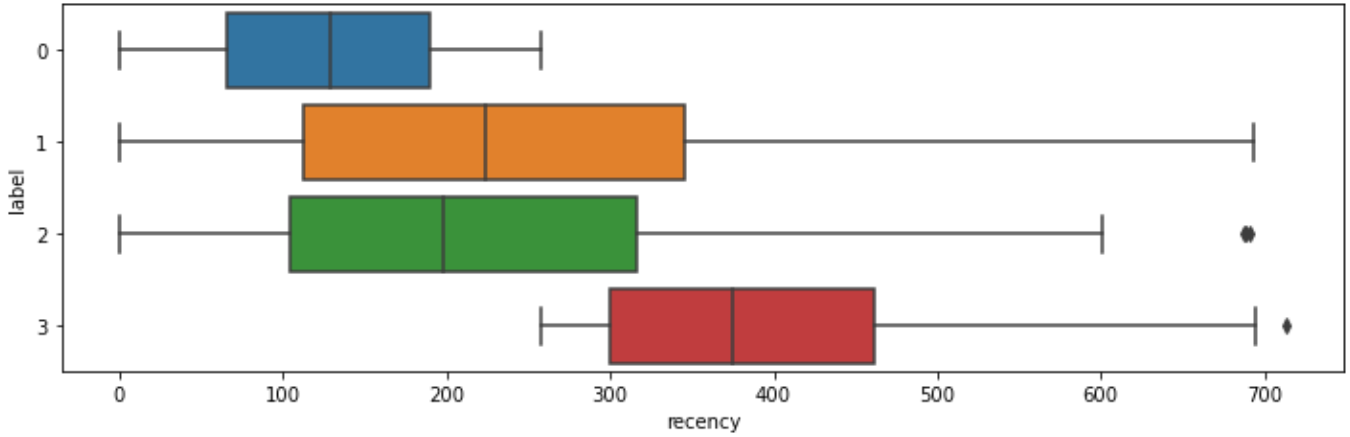
  Feature 1 : Client ayant dépensé beaucoup sur la plateforme.

  Feature 2 : Client ayant passé plus d'une commande et ayant dépensé modérément.

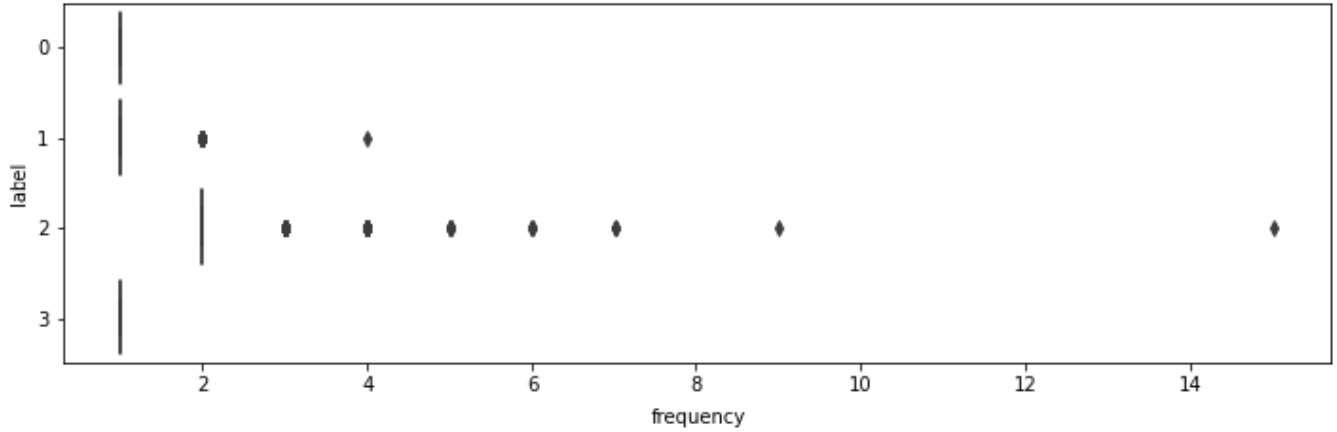
  Feature 3 : Comme Feature 0 mais les clients sont anciens.

Modélisation - K_means

Récence :

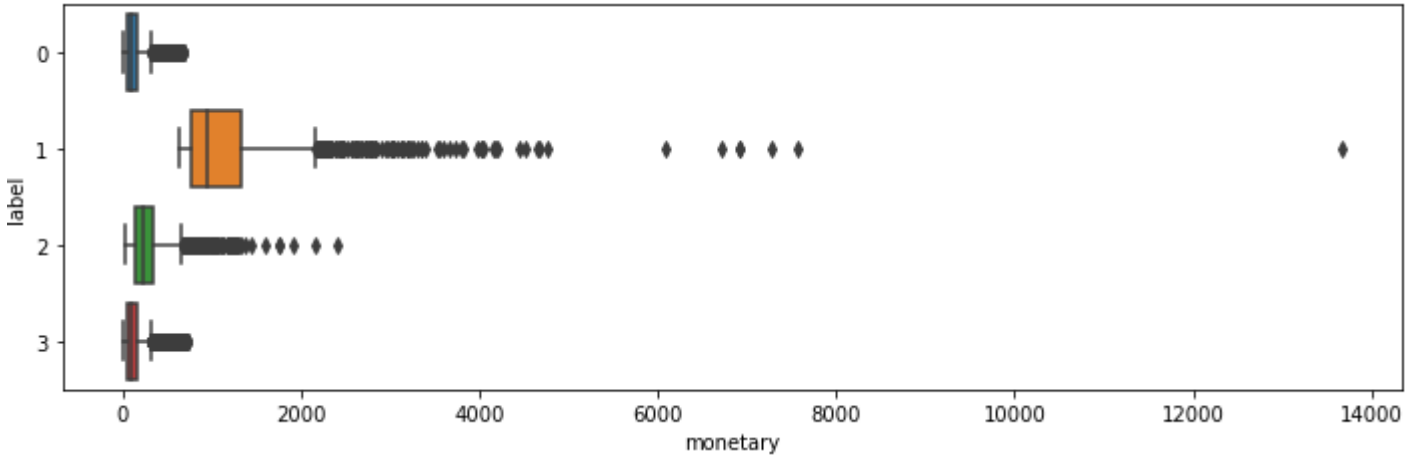


Fréquence :



Modélisation - K_means

Dépense :



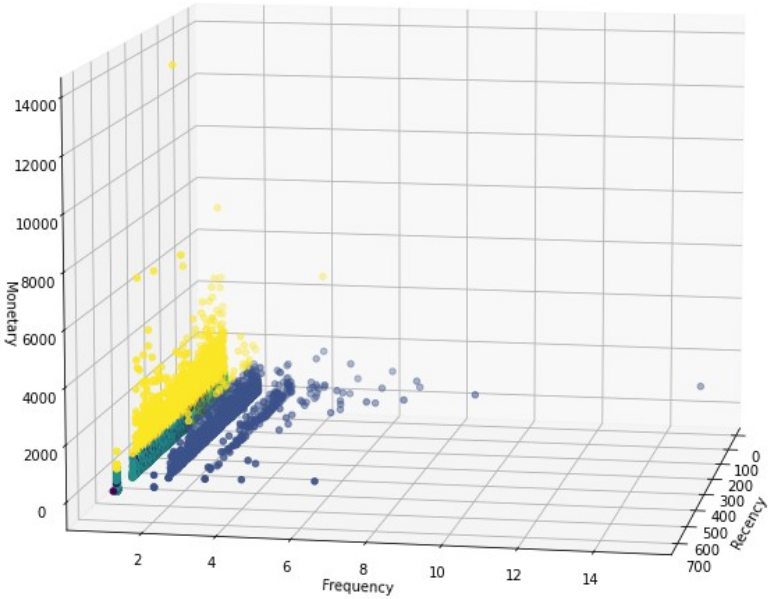
Modélisation - K_means

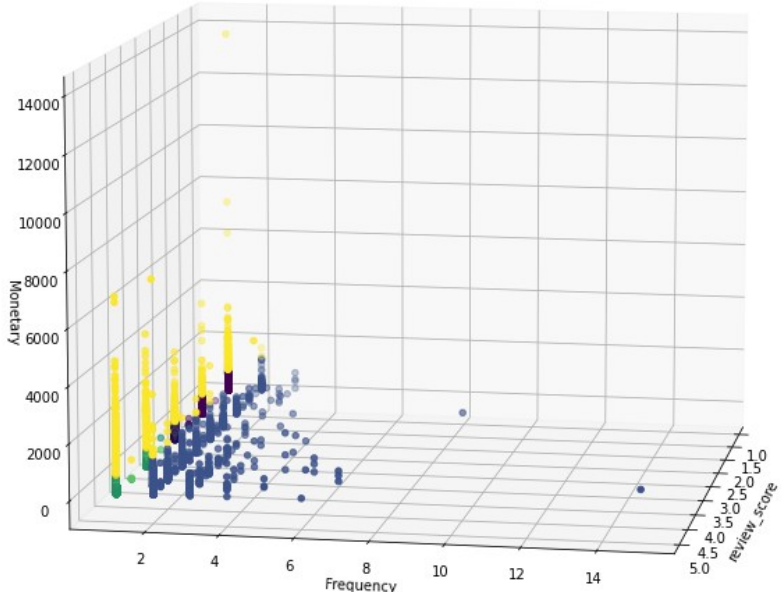
On essaie avec plus de features. Cette fois, on rajoute le review_score moyen de chaque client.

On fait les mêmes testes que précédemment
Cette fois, on trouve 5 kernels.

Modélisation - K_means

- On représente graphiquement notre classification avec 5 clusters :



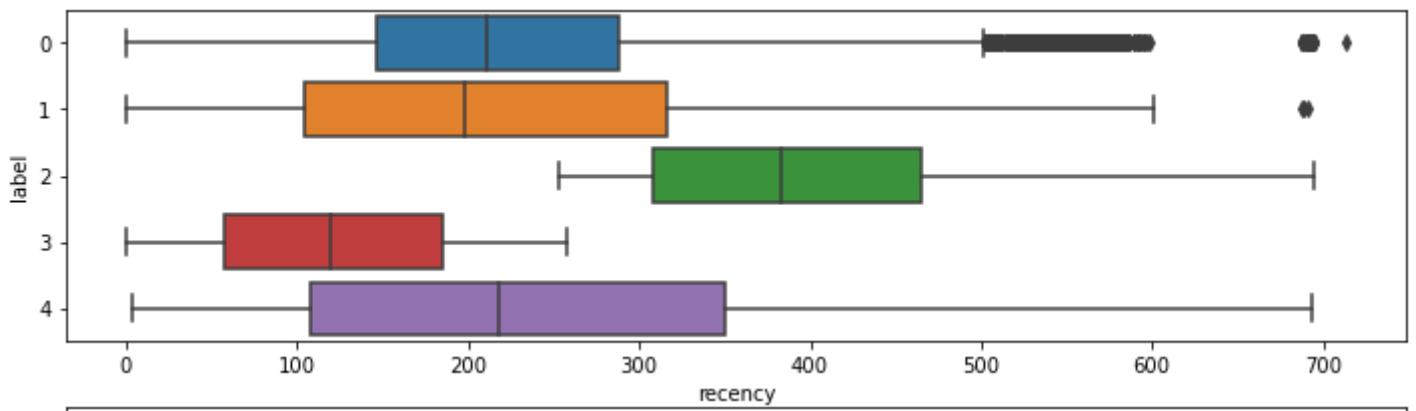


Modélisation - K_means

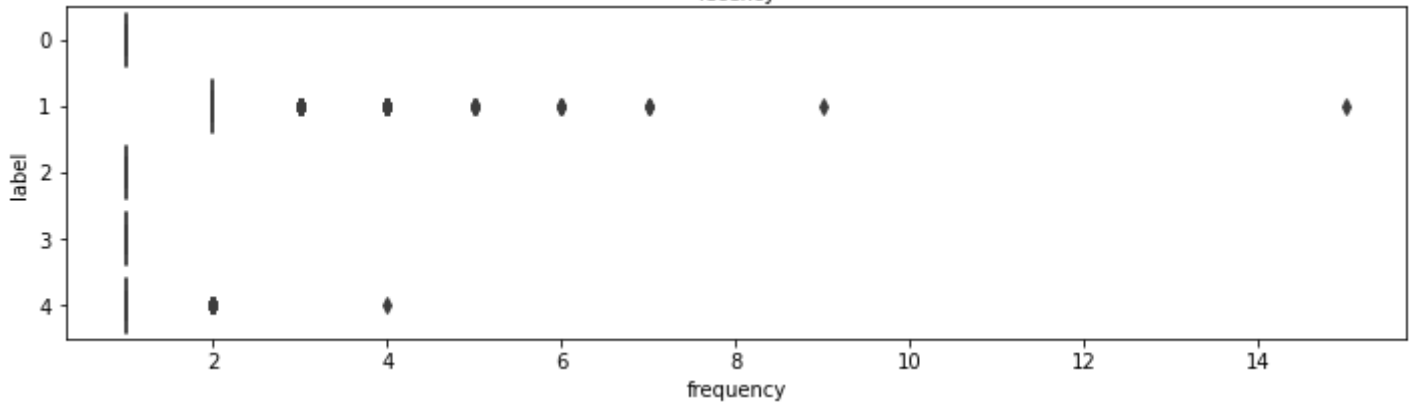
- Feature 0 : Client ayant laissé une mauvaise review et étant venus qu'une seule fois.
- Feature 1 : Client ayant passé plus d'une commande.
- Feature 2 : Client ayant passé une seule commande il y a longtemps.
- Feature 3 : Client récent ayant dépensé modérément, ayant passé une seule commande et laissé une bonne review.
- Feature 4 : Client ayant dépensé beaucoup d'argent.

Modélisation - K_means

Récence :

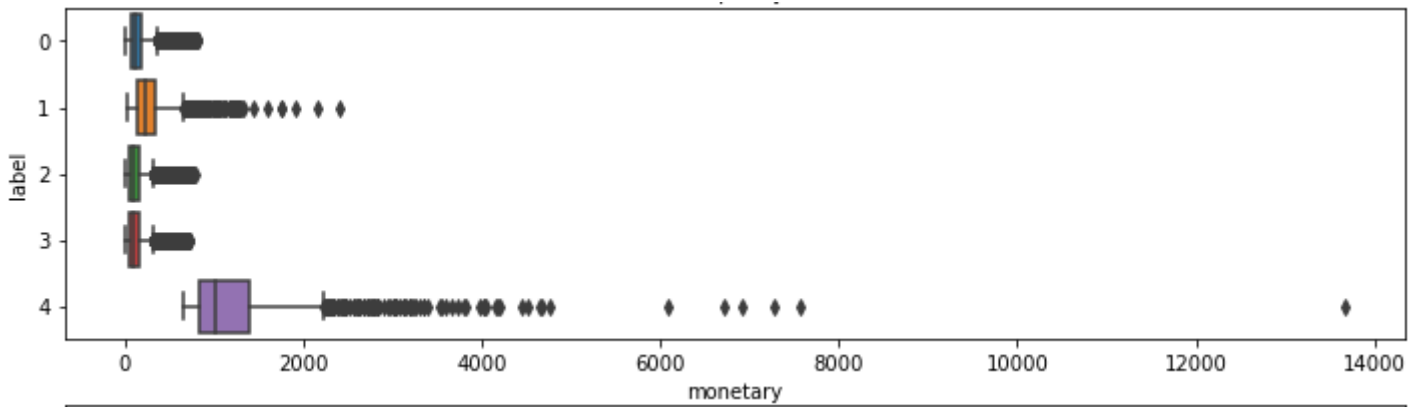


Fréquence :

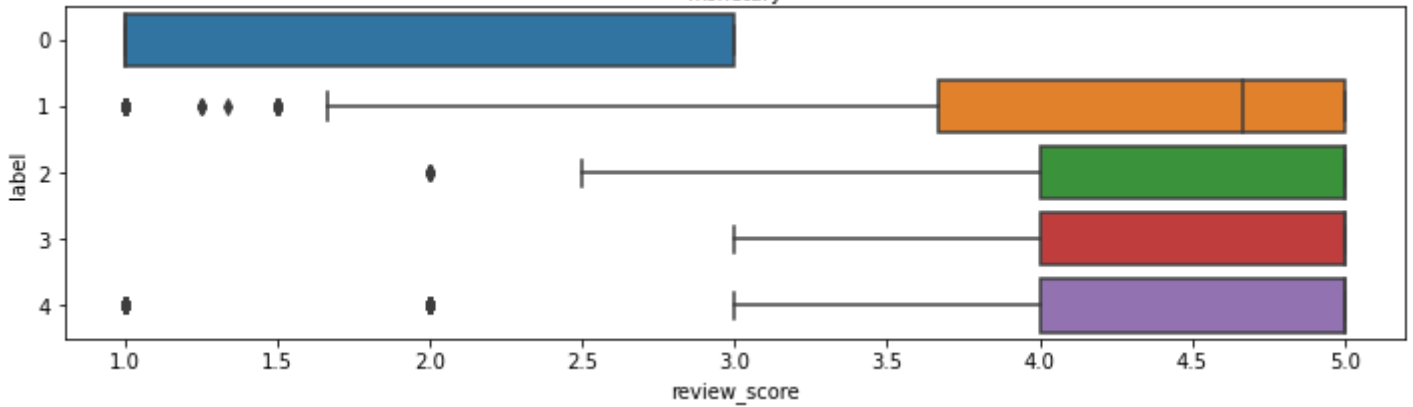


Modélisation - K_means

Dépense :



Review_Score :

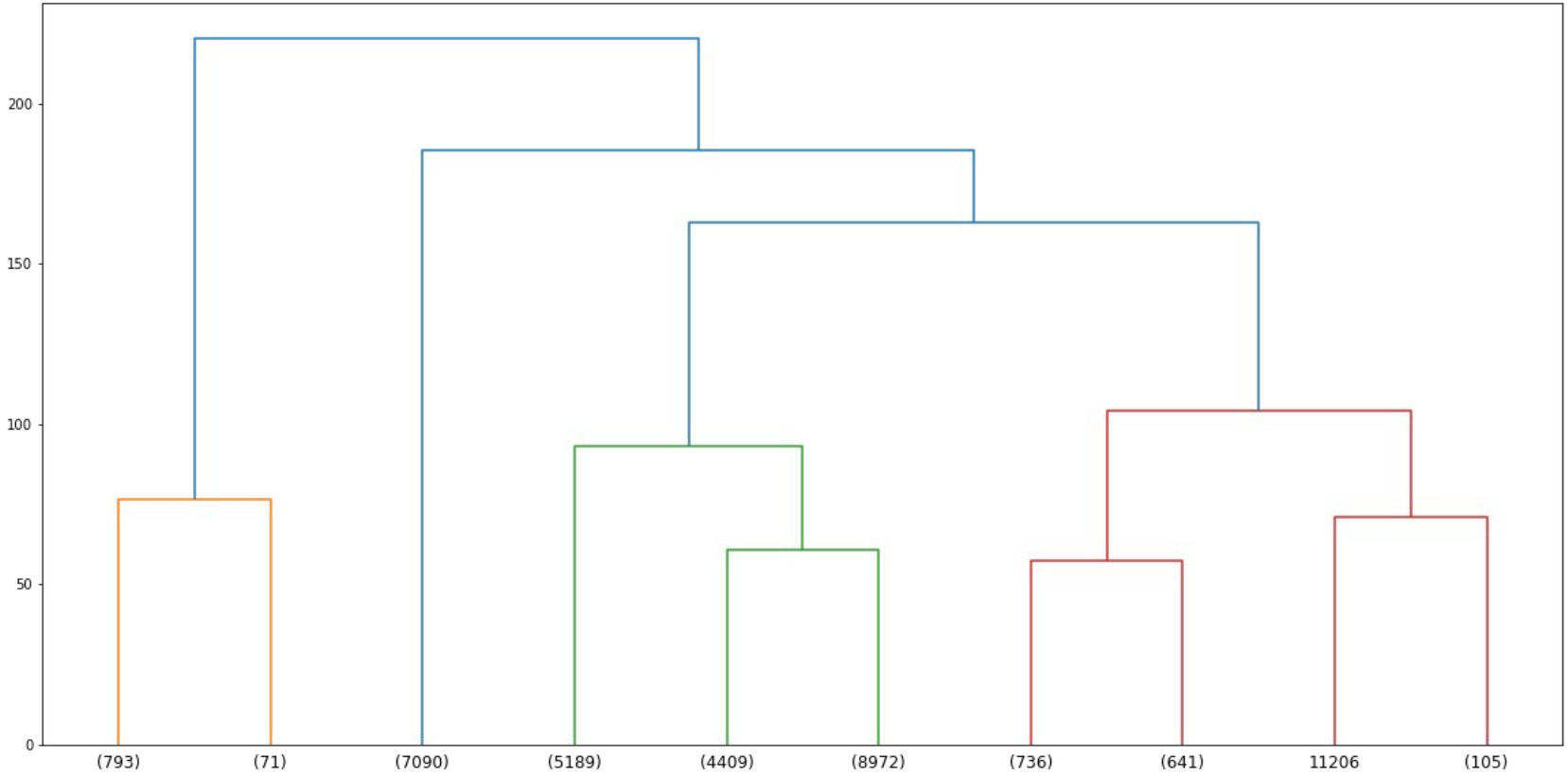


Modélisation – CAH Clustering

- On essaie de faire fonctionner un algorithme de CAH Clustering avec les features RFM.

- On commence par faire un dendrogramme pour trouver le bon nombre de kernels.

Modélisation – CAH Clustering



Modélisation – CAH Clustering

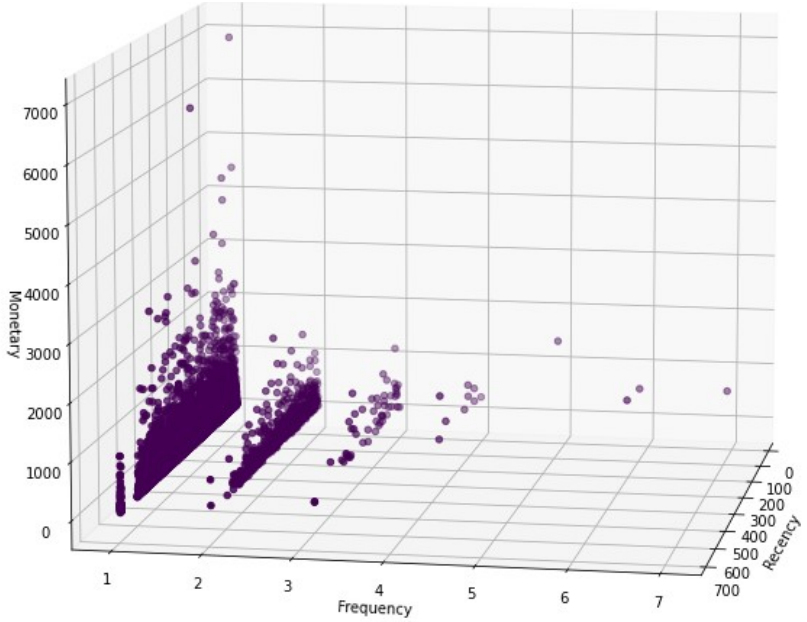
- D’après le dendogramme, on a 4 kernels.
- Feature 0 : Les clients ayant dépensé beaucoup et ayant passé une seul commande.
- Feature 1 : Les clients récent ayant dépensé peu et ayant passé une seule commande.
- Feature 2 : Les clients ancien ayant dépensé peu et ayant passé une seule commande.
- Feature 3 : Les clients ayant passé plusieurs commande.

Modélisation - DBSCAN

- On finit avec un DBSCAN sur les features RFM.
- DBSCAN trouve 5 cluster.

Modélisation - DBSCAN

- On représente graphiquement notre classification avec 5 clusters :



Modélisation - DBSCAN

- Répartition :
- [[-1 0 1 2 3 4 5]
- [27965 10 8 8 5 6 5]]
- Pourcentage :
- [99.9 0. 0. 0. 0. 0. 0.]

Modélisation - Conclusion

- De toute évidence, la classification DBSCAN n’est pas adaptée au problème.
- Le CAH clustering n’est pas assez performant pour traiter de grosse base de données.
- Le K_means donne de bonne catégorie, et est suffisamment performant pour fonctionner sur l’ensemble de gros Dataset.
- On choisit donc K_means avec 5 kernels, et fonctionnant avec le RFM et le review_score comme features.

Mise à jour du modèle

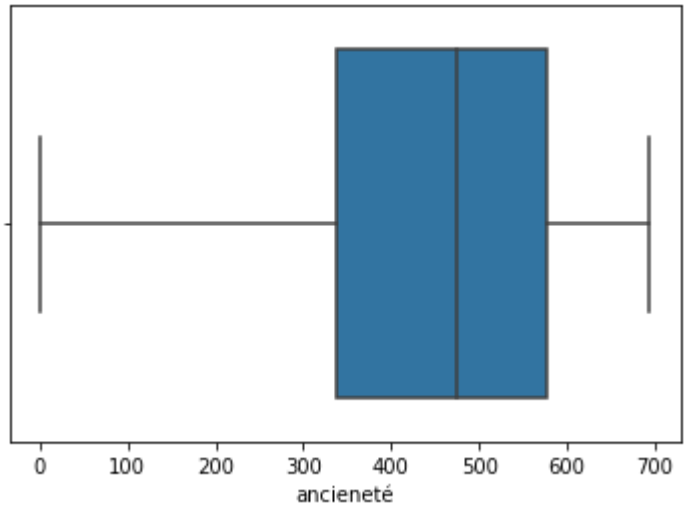
- Il reste à savoir quand doit on mettre à jour le modèle.

  On se base sur la date de la première commande de chaque client pour déterminer son ancienneté.
  L'ancienneté est calculée selon cette formule : combien de jours se sont écoulés entre l'arrivée du client et l'arrivée du premier client du dataframe.

  On regarde la répartition de l'ancienneté pour avoir une date qui contient au moins 50 % du DataFrame.

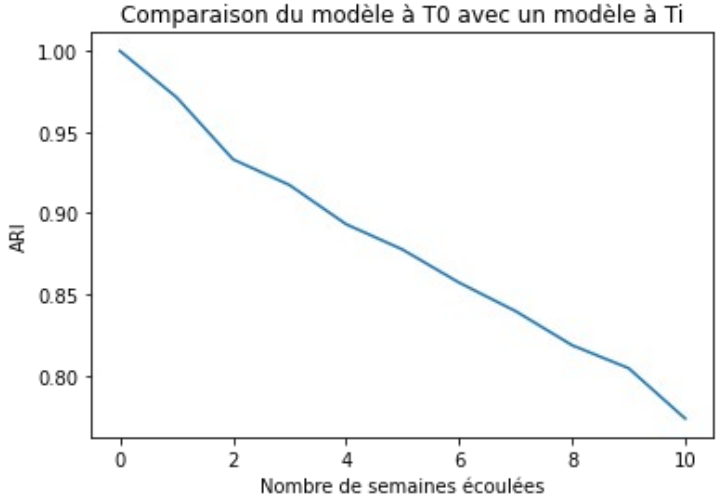
Mise à jour du modèle

D’après la répartition, 500 jours après l’arrivée du premier client fera l’affaire.



Mise à jour du modèle

- On fait tourner un algorithme qui comparera un modèle original contre un nouveau modèle toutes les semaines en l'initialisant à 500 jours après l'arrivée du premier client et nous renverra l'ARI obtenu à chaque comparaison. L'algorithme s'arrêtera lorsque l'ARI descendra en dessous de 0.8



Mise à jour du modèle

- Le modèle devra être mis à jour toutes les 9 semaines.