Synthèse de recherche

1. Pre-processing

a. Création dataset

Une fonction avec comme paramètre

- **Le dataframe d'origine contenant toutes les entrées**
- **les ID des clients (pour traiter les données de chacun)**
- **Une liste de string (Familles distincte)**

nous passons tout en tableau numpy pour une exploitation plus simple.

Initialisation du dataset final, un dataframes avec les Familles + l'ID client comme colonnes

Pour chaque client :

Calcul et ajout des statistiques autour des articles achetés, des articles selon la famille et de ses commandes.

Les statistiques sont factorisé dans des fonctions pour une meilleure lisibilité.

HYGIENE	SOINS DU VISAGE	PARFUMAGE	SOINS DU CORPS	MAQUILLAGE	CAPILLAIRES	SOLAIRES	MULTI FAMILLES	SANTE NATURELLE	NB_TICKET	TICKET_PMOYEN	ART_PMAX	ART_PMIN	NB_ARTICLE	ART_PMOYEN	CLUSTER	CLI_ID
4	3	0	2	2	1	1	0	0	4.0	28.100000	29.90	1.50	13.0	8.646154	1	994216585
3	0	1	1	0	0	0	0	0	1.0	15.950000	5.45	0.90	5.0	3.190000	0	994513006
0	4	2	0	0	0	0	0	0	2.0	52.050000	45.00	5.45	6.0	17.350000	0	977221320
0	17	0	0	0	0	0	0	0	7.0	32.214286	34.00	4.10	17.0	13.264706	1	947199614
0	0	0	0	1	0	0	0	0	1.0	4.450000	4.45	4.45	1.0	4.450000	0	988764392
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

b. Normalisation données

La transformation des données entre 0 et 1 pour scale les valeur car certaines sont des pourcentage, des entier et autres. Cela rend donc plus lisible et exploitable le dataset pour les algorithmes.

c. ACP

L'analyse en composante principales permet de réduire le nombre de features pour qu'ils soit plus optimisé au traitement.

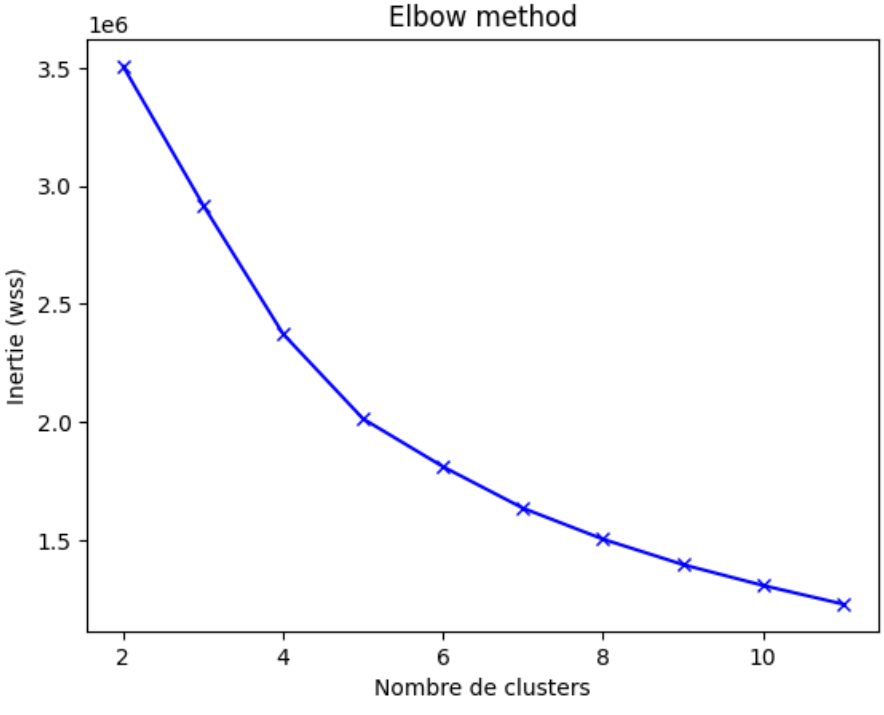
2. Segmentation

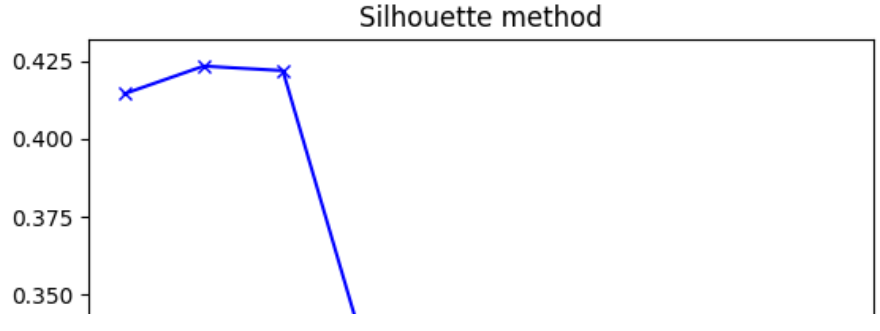
Nous avons choisi de partir sur du KMeans, algorithme la plus rependu pour du clustering qui donne accès à une grande communauté et correspond au travail demandé.

a. Clusters

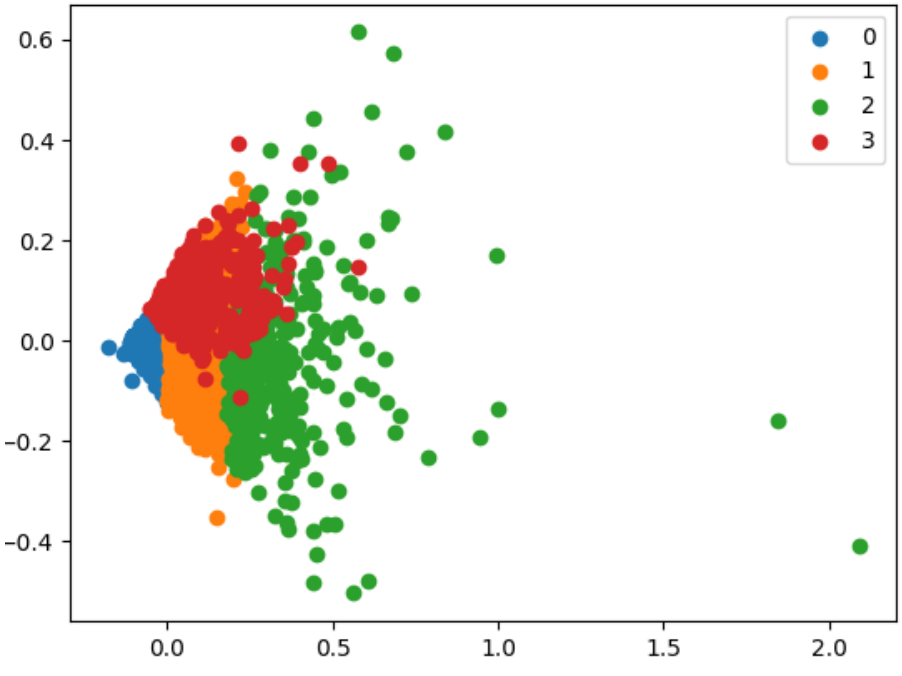
Pour que le clustering soit efficace il faut que le nombre de clusters soit adapté à notre jeu de données. Pour ce faire nous avons crée une fonction, qui selon une range donnée test pour chaque cas, l'efficacité.

De ce test sort une inertie et le score silhouette. Nous allons utilisé ces valeurs pour affiché des graphique et interprété le nombre de cluster optimale. Ce sont la "Elbow method" et "Silhouette method".





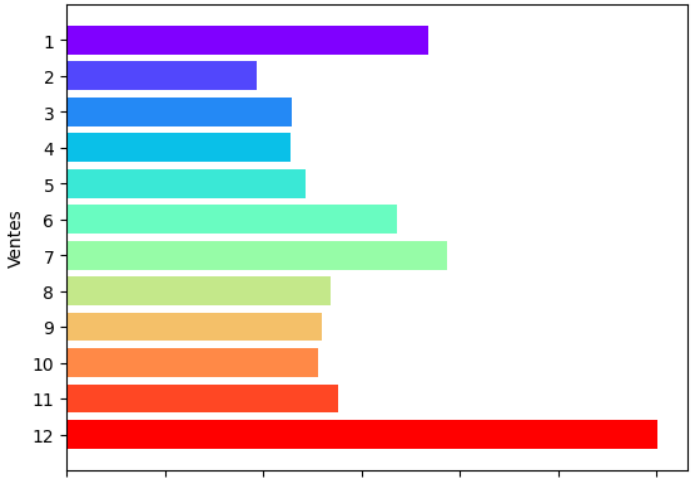
Le rendu des clusters.

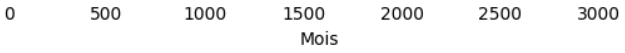


b. Interprétation des clusters

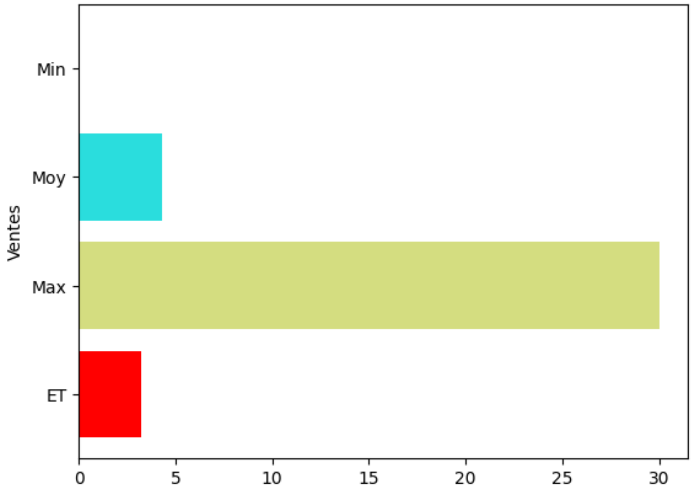
Pour comprendre les tendances de chaque clusters et établir des profils de client nous devons les comparer entre eux grâce à des graphiques.

Ventes par mois:

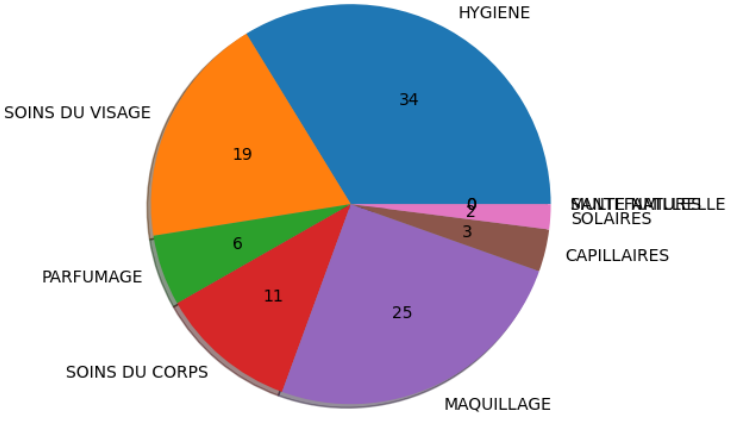




Prix moyen, minimum, maximum et écart type:

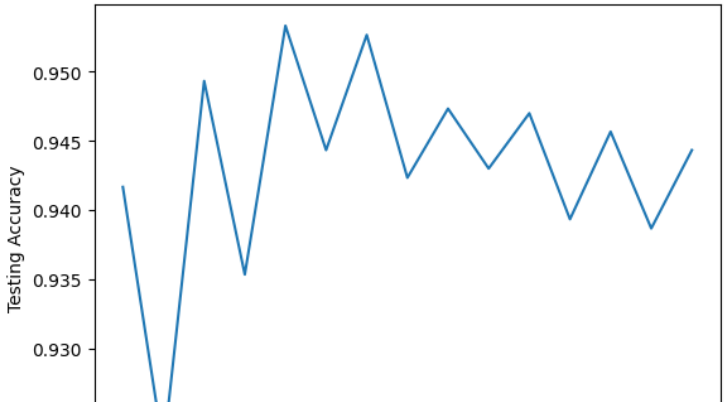


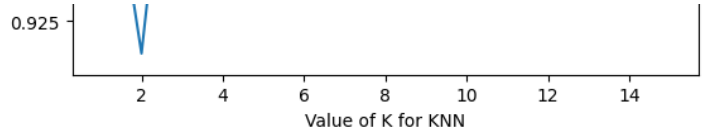
Pourcentage de ventes des familles:



c. Validation (+ cross validation)

Pour valider notre segmentation, nous allons utilisé une algorithme de classification supervisé avec comme objectif nos cluster. Nous utilisons ici un KNN. Comme pour le KMeans nous devons trouver le nombre de voisin optimale, comme précédemment selon une range, tester chaque cas et en sortir un graphique pour l'interprétation.





Le machine learning supervisé nécessite de l'entrainement avant de pouvoir passer à la prédiction, notre dataset et donc découper avec 70% du datasets réservé à l'entrainement et les 30% restant à la prédiction.

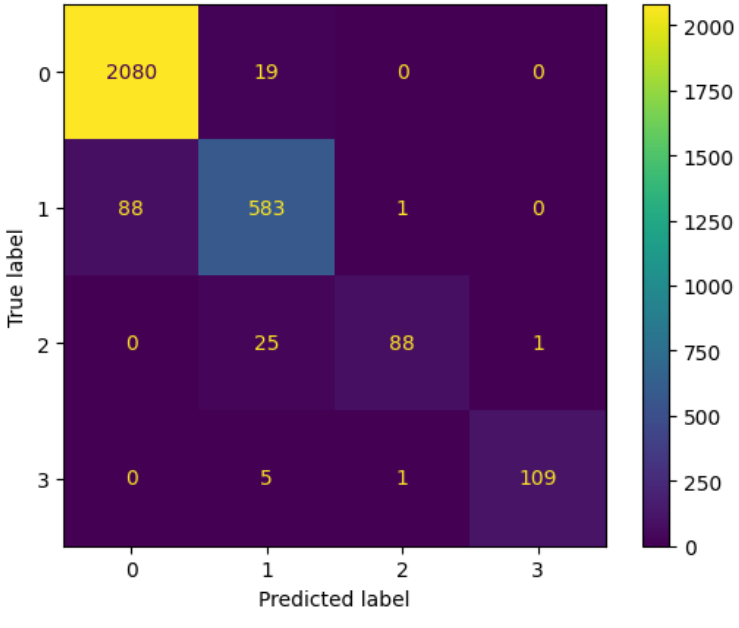
Une fois le KNN terminé, un rapport est écrit :

```
0.9533333333333334
              precision    recall  f1-score   support

           0       0.99      0.96      0.97      2168
           1       0.87      0.92      0.89       632
           2       0.77      0.98      0.86        90
           3       0.95      0.99      0.97       110

    accuracy                           0.95      3000
   macro avg       0.89      0.96      0.93      3000
weighted avg       0.96      0.95      0.95      3000
```

Mais pour plus de lisibilité nous allons utilisé une matrice de confusion (True false) qui nous permet de voir les réponses fournis.



Pour affiner encore plus, la librairie SKlearn nous propose d'utiliser de la cross validation qui permet d'estimer la fiabilité d'un modèle.

```
Accuracy: 0.93 (+/- 0.02)
```

3. Recommandation

La recommandation est un algorithme crée "from scratch" qui se base sur le cluster du client entré. Il récupère tout les produits commandé par les personnes du même segment (hors celui

de la personne souhaité) et choisi un produit de manière random, possibilité de choisir l'article le plus acheté du cluster.

Sorti de la recommandation:

PRIX_NET	FAMILLE	UNIVERS	MAILLE	LIBELLE
6.8	PARFUMAGE	PARF_EAUX PLAISIR NATURE	PARF_EDT	EDT FRAISE PN2 100ML VPM

X. Sources

a. Documentation

https://scikit-learn.org

https://seaborn.pydata.org/

https://pandas.pydata.org/docs/reference/frame.html

https://numpy.org/

https://matplotlib.org/

b. Articles

https://datascientest.com/algorithme-des-k-means

https://datascientest.com/knn

https://datascientest.com/cross-validation

https://medium.datadriveninvestor.com/data-preprocessing-for-machine-learning-188e9eef1d2c

https://medium.com/analytics-vidhya/data-preprocessing-using-sklearn-2c6fe013f594

https://medium.com/luca-chuangs-bapm-notes/principal-component-analysis-pca-using-python-scikit-learn-48c4c13e49af