

Projet FRAGRAI



Lien du projet : <https://github.com/Sykzen/FRAGRAI>

Réalisé par :

Rachad Waïl Said

Seydou Guindo

Sermed Mabrouk

2021-2022

Table des matières

I - Introduction

- Description du Projet
- Présentation de la Pipeline

II - L'extraction des données

III - Le Data Cleaning (Data Pre-Processing)

IV - One Hot encoding (Feature Engineering)

V - K-means (Model Selection)

- Pourquoi K-means
- Choix du N_clusters avec elbow et inertia score (Parameters optimization)
- Etude de la silhouette score (Parameters optimization 2)
- Choix du Cluster

VI - 3D Plotting

VII - Création de la super Recette

VIII - RNN Attribution d'une marque et d'une référence

IX - Outils de Workflow utilisé

X - Conclusion

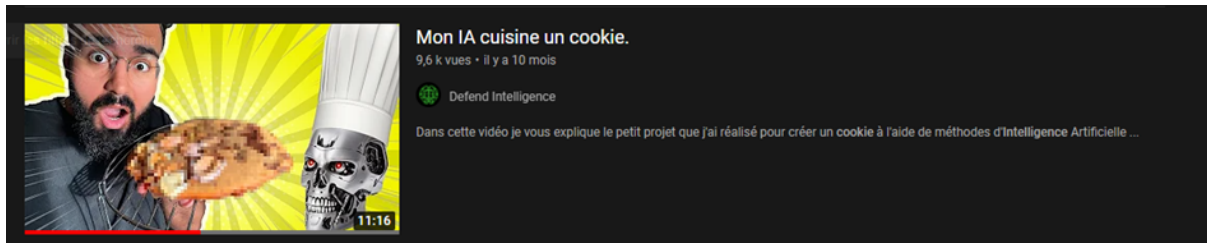
- Ce qu'on a appris

XI - Glossaire/Sources

I. Introduction

1. But

Le but de notre projet est de construire une recette de parfum à partir de plusieurs recettes de parfums et de leurs notes attribuées entre 0 et 5 par des clients. L'idée de base nous est parvenue d'un youtubeur qui à prédit la recette d'un cookie.

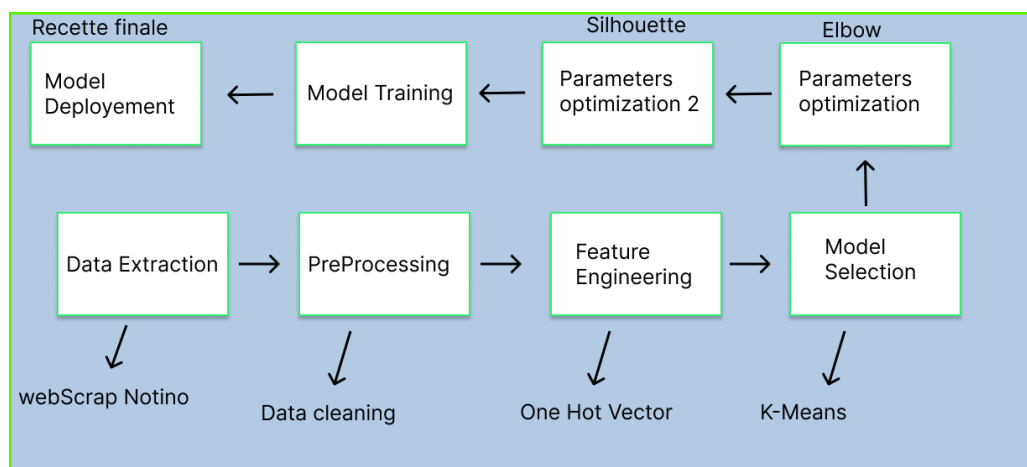


Nous avons donc commencé par chercher un dataset préconstruit et nous avons aussi demandé à des distributeurs d'avoir accès à leurs API ou nous fournir une database de parfum sans succès, ce qui nous a poussé alors à créer notre propre dataset (WebScrapping).

La recette prédit est appelée le “jus” et elle constitue environ 10% du flacon.

Nous n'avons malheureusement pas les proportions de chaque ingrédient car cela pousserait la concurrence à produire le même parfum, cependant les recettes sont obligatoirement partagées si le produit est mis en vente pour éviter des allergies aux clients. La note de chaque recette (le rating) est la moyenne des notes des clients qu'ils ont attribuer à ce parfums

2. La Pipeline



II. L'extraction des données

Notre première étape était alors de collecter des données des différents distributeurs en ligne de parfums, on a repéré 3 sites avec un large catalogue de produits de parfum.

Voici l'aperçu des données sur le site :

Google qui héberge votre site

Liste complète des ingrédients

ALCOHOL DENAT (SD ALCOHOL 99-C) - PARFUM (FRAGRANCE) - AQUA (WATER), ETHYLHEXYL METHOXYCINNAMATE, DIETHYLAMINO HYDROXYBENZOYL HEXYL BENZOATE -, BHT, CI 17200 (RED 33), CI 60730 (EXT. VIOLET 2), CI 19140 (YELLOW 5), BENZYL SALICYLATE, LIMONENE, HEXYL CINNAMAL, COUMARIN, LINALOOL, CITRAL, CINNAMAL, CITRONELLOL, GERANIOL Le fabricant est responsable des ingrédients du produit. Nous vous recommandons de vérifier la liste des ingrédients du produit directement sur son emballage en raison de changements potentiels.

[TEXTE INTÉGRAL](#)

Notre site internet utilise des cookies. En naviguant sur ce site, vous acceptez l'utilisation de cookies. [En savoir plus](#)

J'accepte

DevTools is now available in French! [Always match Chrome's language](#) [Switch DevTools to French](#) Don't show again

Elements Console Recorder Sources Network Memory Performance Application Security Lighthouse AdBlock Plus

Styles Computed Layout Event Listeners

Filter

show .cls +

element.style {

}

appnot(.accessibility) @0faebadd25...8ca81.c

* {

outline: 2px solid #f00; font-weight: bold;

Description La marque Avis 19

Avis Boucheron Quatre Absolu de Nuit

4.3 ★★★★★

Donner mon avis

17 ★★★★★ aspect du flacon

2 ★★★★★ tenue du parfum

3 ★★ intensité

1 ★ impression générale

Les plus récents

Une très belle découverte Tout ce que j'aime dans un parfum je suis adepte de parfums orientaux mais chaud Je suis ravie de cet achat je le recommanderai sans hésitez Envoi rapide et sur Je suis devenue adepte de Notino J'ai déjà passer plusieurs commandes sur le site surtout avec des prix très compétitifs je recommande

★★★★★ Nathalie F. 31 mars 2022 Avez-vous trouvé cet avis utile ? [Oui](#) 0 | [Non](#) 0

On a utilisé la librairie BeautifulSoup4 de python afin de récupérer le DOM html et en extraire les informations qui nous intéressaient.

Concernant les difficultés que nous avons rencontré, cette étape fût pénible car le site changeait l'arbre DOM tous les 4-8h et donc un code qui marchait le matin ne fonctionnait plus le soir. De plus, l'index des produits changeait chaque seconde et

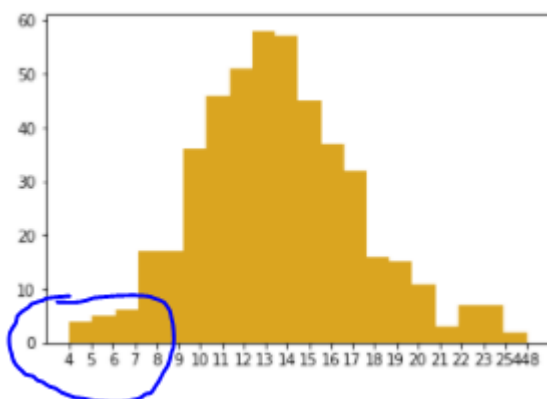
donc à chaque nouvel appel d'extraction de page, on pouvait avoir des produits qui reviennent plusieurs fois.

Cette étape nous a pris environ 3 jours, on a pu collecté un dataset de 2000 parfums, (nom,recette,référence,rating) qui après enlèvement des valeurs vides, est descendu à 500 parfums.

III. Le Data Cleaning (Pre-Processing)

Après l'étape de l'extraction, nous avons beaucoup de recettes qui contenaient des commentaires, des publicités liées aux sites, des chaînes de caractères incompréhensibles ou encore des ingrédients qui était épelé avec des noms différents (Aqua/water) (La dirty Data).

On a alors décidé de créer un ensemble d'expressions régulier pour (éliminer/regrouper) ces ingrédients grâce à cela, on a réussi à obtenir le graphe suivant qui regroupe le nombre d'occurrence d'un ingrédient parmi tous les parfums du dataset.



On remarque alors qu'il y a certains ingrédients qui apparaissent moins de 7 fois dans toutes les recettes, on a donc décidé de les supprimer pour la suite de notre modèle.

Ensuite, on a supprimé la colonne "nom" du parfum et la référence du DF pour faciliter la prochaine étape de notre Pipeline : Le Feature engineering (Encodage de données) avec One hot Encoder.

Voici les données qu'a obtenu après cleaning :

```
Entrée [12]: print(df.isnull().sum())
print(df.nunique())
print(df.shape)
```

```

Marque      0
Ingr         0
Rating      0
Reference    0
dtype: int64
Marque      160
Ingr        506
Rating      29
Reference    505
dtype: int64
(506, 4)
```

| | | Ingr | Rating |
|-----|--|------|--------|
| 0 | RICINUS COMMUNIS (CASTOR) SEED OIL, STEARETH... | | 2.9 |
| 1 | DIMETHICONE, CI 42090 (FD&C BLUE NO. BUTYLPHE... | | 3.6 |
| 2 | lnCINNAMAL, SODIUM PHOSPHATE, ALPHA-ISOMETHYL... | | 3.9 |
| 3 | ALCOHOL DENAT, TOCOPHEROL, #13380 ALCOHOL, ... | | 5.0 |
| 4 | CI 42090 (BLUE 1), PARFUM (FRAGRANCE) -, SODI... | | 4.9 |
| ... | ... | ... | ... |
| 501 | PARAFFINUM LIQUIDUM, PEG-90 GLYCERYL ISOSTEAR... | | 4.0 |
| 502 | CI 42090 (BLUE 1),#13877 ALCOHOL, (SD ALCOHOL... | | 4.5 |
| 503 | ZUR ÜBERPRÜFUNG DER ZUSAMMENSETZUNG EINES BES... | | 4.0 |
| 504 | HYDROLYZED CORALLINA OFFICINALIS EXTRACT,DIEN... | | 4.0 |
| 505 | ALCOHOL OF VEGETAL ORIGIN 80% VOL, CALCIUM LA... | | 3.7 |

506 rows x 2 columns

IV. One Hot encoding (Feature Engineering)

Après l'étape du pré-processing, notre dataset était presque prêt à être entraîné mais avant tout, nous devons nous assurer que les algorithmes utilisés fonctionnent bien avec notre dataset car ce genre d'algorithmes sont fait de telle manière à recevoir seulement des nombres (0,1).

Donc, on a dû transformer notre dataset qui est de base une liste d'ingrédients avec un rating en binaires. On a procédé comme suit.

Tous les ingrédients existants se placent en intitulés de colonne, les index des lignes représentent les recettes et on place un 1 si la recette contient l'ingrédient ou 0 sinon. On ajoute à la fin une colonne "ratings". Ces étapes ont modifié notre dataset qui est maintenant divisé en 4550 colonnes (4549 ingrédients + la colonne ratings). Puis à l'aide des librairies importées, (comme pandas ici), nous avons accompli cette étape et obtenu les résultats suivants :

| 0 lnCINNAMAL | 0 lnCOUMARIN | 0 (SD ALCOHOL 39-C) | 0 17200 (RED 33) | 0 19140 (YELLOW 5) | 0 2 (CI 60730) | 0 2 0 | 0 4 (CI 14700) | 0 60730 | 0 78 | ... | 18 WATER (AQUA) | 18_DIENTS PEUT FAIRE L'OBJET DE MODIFICATIONS | 18_EUGENOLCITRONELLOL | 18_M OCTY |
|-----------------|-----------------|---------------------------|---------------------------|--------------------------|----------------------|----------|----------------------|------------|------|-----|-----------------------|--|-----------------------|--------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 501 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 502 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 503 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 504 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| 505 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |

506 rows x 4550 columns

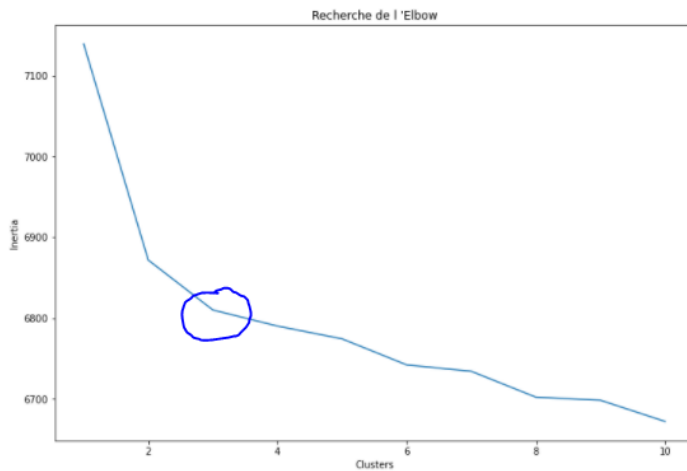
V. K-Means

1. Pourquoi K-means

On a opté pour k-means comme algorithme non-supervisé car il s'agit de l'algorithme le plus utilisé qui donne de très bons résultats à chaque fois. Nous n'avons choisi qu'un seul algorithme car notre prédiction porte sur la recette d'un produit et donc nous devons tâcher de produire le meilleur résultat possible. Nous portons une grande importance aux choix de l'algorithme et les hyperparamètres au lieu d'utiliser tous les algorithmes non-supervisé et obtenir plusieurs recettes à la fin sans savoir lequel choisir. Dans notre pipeline il n'y aura pas de model validation ce qui signifie dans notre cas la création de la recette dans le monde réel et la faire tester par des humains. Ceci explique la grande importance que nous accordons au choix des hyperparamètres (elbow+silhouette).

2. Choix du N_clusters (elbow + inertia) (Parameters optimization)

Nous avons commencé par choisir le nombre de clusters idéal avec elbow (la méthode du coude). Nous avons "fit" notre algorithme 11 fois en calculant l'inertia à chaque parcours voici le graphe qu'on a obtenu :

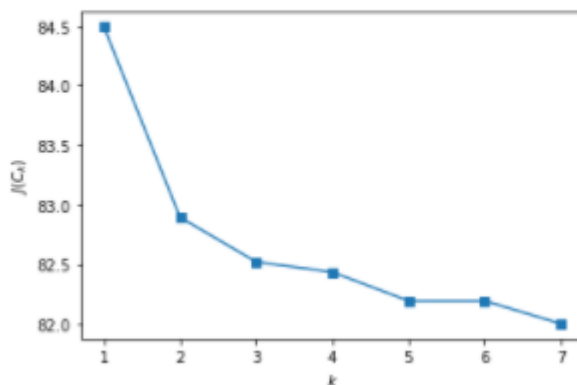


On remarque alors, le point elbow à $n = 3$ donc $n_clusters = 3$.

On remarque aussi que 2 peut aussi être un point elbow on pousse les recherches pour choisir le bon cluster

Pour vérifier notre conclusion, on a ajouté un checkout avec la racine de l'inertie au lieu du score et on a obtenu le même résultat :

```
2]: # Plot Les k valeurs avec inertia
plt.plot(range(1, 8), inertia, marker='s');
plt.xlabel('$k$');
plt.ylabel('$J(C_k)$');
```



ie 2 et 3 peuvent être un point elbow

3. Choix du $N_clusters$ (silhouette) (Parameters optimization 2)

Après la première recherche du $n_cluster$ paramètre, on a essayé d'optimiser encore plus avec silhouette score.

on calcule le score silhouette des $n_cluster$ de 2 à 10

et voici ce qu'on obtient


```
Entrée [47]: for idx, score in zip(n_clusters,s_scores):  
             print(idx,"clusters >> ", "silhouette score is",score)
```

```
2 clusters >> silhouette score is 0.036227103885610346  
3 clusters >> silhouette score is 0.015523125890598889  
4 clusters >> silhouette score is 0.0006592104532288375  
5 clusters >> silhouette score is 0.012233705344115223  
6 clusters >> silhouette score is 0.0001487231062930258  
7 clusters >> silhouette score is 0.006851682219485666  
8 clusters >> silhouette score is 0.003593647294055814  
9 clusters >> silhouette score is -0.007031348292321058  
10 clusters >> silhouette score is -0.0013430085018403715
```

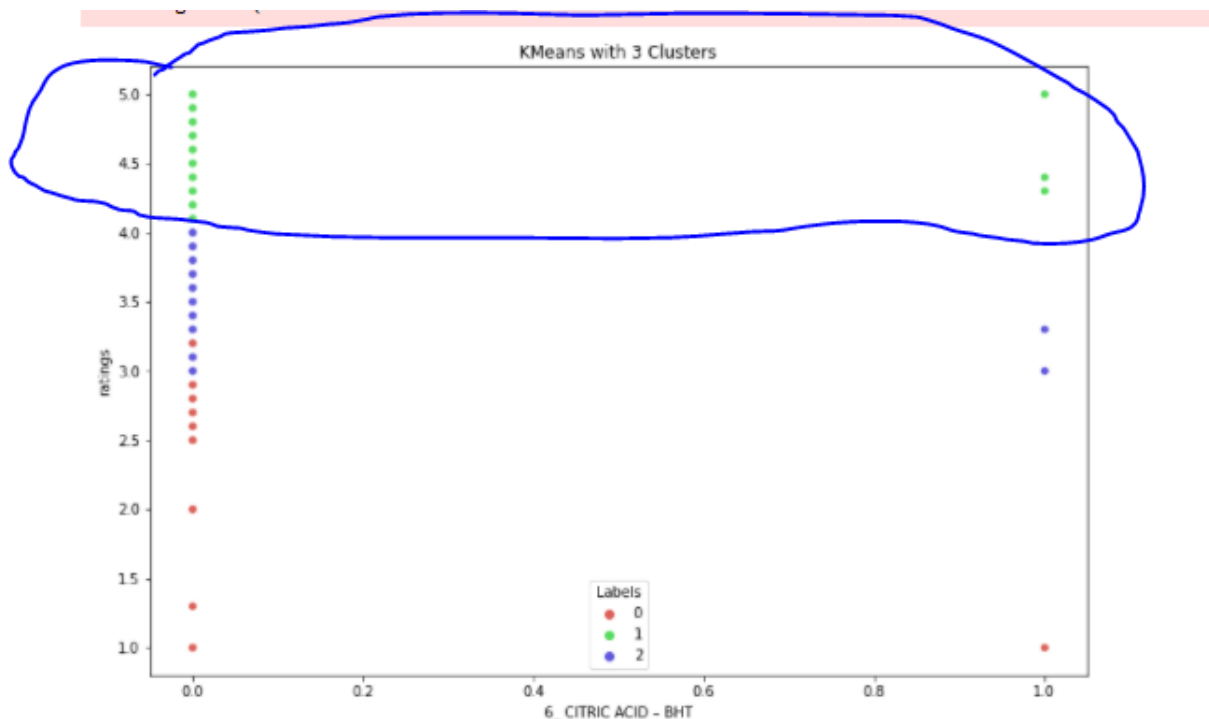
```
Entrée [48]: for idx, score in zip(n_clusters,clusters_inertia):  
             print(idx,"clusters >> ", "inertia score is", score)
```

```
2 clusters >> inertia score is 6871.797303370762  
3 clusters >> inertia score is 6811.198678129689  
4 clusters >> inertia score is 6801.74726537001  
5 clusters >> inertia score is 6781.726211538463  
6 clusters >> inertia score is 6781.85582430413  
7 clusters >> inertia score is 6722.844323140624  
8 clusters >> inertia score is 6724.586640347098  
9 clusters >> inertia score is 6707.871834152666  
10 clusters >> inertia score is 6683.30306354298
```

On remarque quand même que n=2 clusters représente le double du score de n=3

4. Choix du Label d'ingrédients

Ensuite, nous avons "plot" nos 3 clusters sur 1 des ingrédients (on a pris un des ingrédients ayant la plus grosse moyenne de rating) et obtenu le graphe suivant.



On remarque que la plupart des points avec un rating 4-5 sont verts et appartiennent donc au label 1 (1 des 3 clusters).

(L'axe des X ne représente que les valeurs du one hot vector qui sont toujours soit 0 soit 1)

on affiche le size de chaque cluster pour voir si ya un cluster à éliminer

```
[28]: KM_clust_sizes = KMS_clustered.groupby(
      KM_clust_sizes.columns = ["KM_size"]
      KM_clust_sizes

      # Cluster 0 - 60
      # Cluster 1 - 258
      # Cluster 2 - 188
```

```
jt[28]:
```

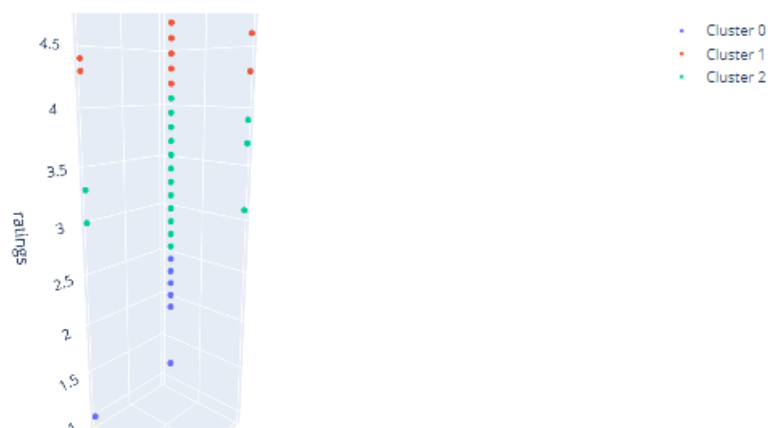
| KM_size | |
|---------|-----|
| Cluster | |
| 0 | 60 |
| 1 | 258 |
| 2 | 188 |

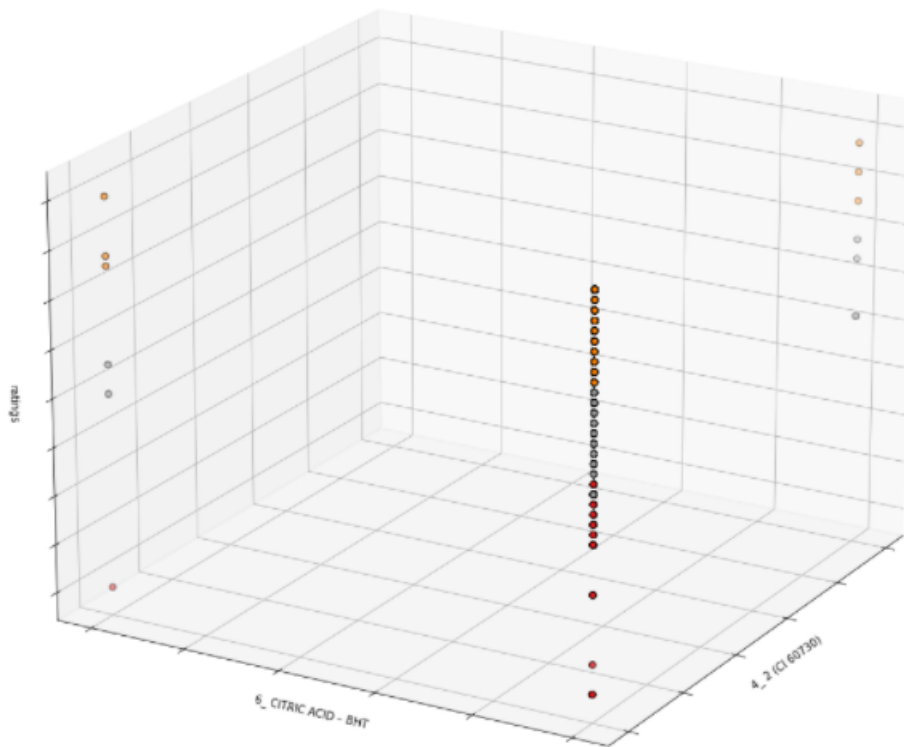
on décide alors de prendre le cluster 1 pour notre model deployment

VI. 3D Plotting

On affiche un 3D plot avec deux ingrédients pour s'assurer de notre cluster choisi

Clusters by K-Means





on remarque aussi que cluster 1 représente les meilleurs ratings

VII - Création de la super Recette

Après les étapes faites précédemment, nous arrivons à la dernière étape du Pipeline (Le Model deployment) ou la super recette finale.

Voici la sortie de nos algorithmes avec les hyperparamètres qu'on a défini :
(On voit qu'on a spécifié Labels==1 pour les raisons cité au dessus)

```
np.sum(output.loc[output.Labels == 1], axis = 0).sort_values(ascending = False)[2:20]
```

| | |
|--|-----|
| 5_ CI 47005 (D&C YELLOW NO | 5.0 |
| 6_ PARFUM (FRAGRANCE) - AQUA (WATER/EAU) | 4.0 |
| 1_ BENZYL BENZOATE | 4.0 |
| 8_ HYDROXYCITRONELLAL | 4.0 |
| 11_ B217785/1) | 4.0 |
| 7_ ALPHA-ISOMETHYL IONONE | 4.0 |
| 3_ CI 77163 (BISMUTH OXYCHLORIDE) | 4.0 |
| 6_ CI 60730 (EXT VIOLET 2) | 4.0 |
| 1_ SUCROSE ACETATE ISOBUTYRATE | 4.0 |
| 5_ C YELLOW NO | 3.0 |
| 10_ \nCINNAMAL | 3.0 |
| 9_ CETEARYL ALCOHOL | 3.0 |
| 1_ BENZYLALCOHOL | 3.0 |
| 2_ ALPHA-ISOMETHYL IONONE | 3.0 |
| 4_ EXT VIOLET 2 (C | 3.0 |
| 7_ CETYL PEG PPG-10 1 DIMETHICONE | 3.0 |
| 2_ METHYL ANTHRANILATE | 3.0 |
| 8_ HYDROGENATED COCO-GLYCERIDES | 3.0 |

dtype: float64

Comme mentionné ci-dessus, nous ne pourrons pas procéder à un modèle validation car une machine ne pourra déterminer si un parfum “sent bon” ou pas (à moins de créer un modèle qui prédit le rating d’une recette).

VIII - RNN Attribution d’une marque et d’une référence

Après les étapes faites précédemment, nous voulions ajouter quelques retouches pour avoir un produit créé à 100% par une machine et ainsi donné du sens au nom du projet (Fragrai).

Nous avons donc utilisé un réseau de neurones pour générer du texte, ici, nous voulons générer le nom du parfum ainsi que le nom de sa marque associée à partir du dataset extrait au préalable (dans la colonne référence et marque).

On ne détaillera pas beaucoup cette partie car nous n'avons pas encore des bases assez solides sur le Deep Learning, mais voici ce qu'on a compris et qu'on a pu appliqué :

- On s'est penché sur le modèle séquentiel de Keras car il était plus facile à utiliser et beaucoup de projets Open Source ,qui portaient sur la même problématique, l'ont également utilisé.
- On a commencé par déterminer le corpus length de notre chaîne de caractère concaténée, ainsi que la longueur de la plus grosse séquence et le nombre de caractères utilisés sur tous les chars du dataset (ces informations seront utiles pour le modèle de tf)
- On a ensuite converti nos données sur une matrice en codant pour chaque chaîne de caractère avec le One hot. De ce fait, chaque chaîne de caractère possède son propre code par exemple le mot "dior" est représenté par la liste [1,...0*(le nombre de noms différents)] et ainsi de suite pour créer nos deux matrices X,Y.
- On a choisi Adam optimizer comme hyperparamètres pour un entraînement rapide, ce dernier combine les deux descentes de gradient Momentum et RMSprop (Root mean square propagation) en 1 seule équation.
- On entraîne alors notre modèle avec sequential de tensorflow avec 64 entraînements par itération (batch size).
- et on obtient les résultats en dessous :

```
:rée [*]: import random
for _ in range(500):
    word=""
    for i in range(random.randint(1,4)):
        word+=generate_Ref()+" "
    print(word)
#Forever Suprême
#Vences Faith Ambre Bouquet |
#J'adore Ambre
#Intense Flowers
#Miris Paris
#Elixir Rebella
#Celline Man x Olyfet
#Flowers La vie
#Vanilla Legend

Debut Oudh 36 La nyc
Vital
Faith Blossom Legend
Debut Medlove Dossom Be gres
My vi Miss ap Shanima
Ck ont Never Milam Unne me
Candy k Cinema Oiseau Maxima
Rebelle Maxima
Magnia Rogur
Forever
Cinema
Nisean Ralph Pinea
Bergam
Night J'ado
Dohér Peeple Lilac Ambre
Ellion
```

Pour les marques :

Generate nom d'une marque

```
Entrée [*]: for _ in range(200):
             if generate_marque() in Marque_input:
                 pass
             else:
                 print(generate_marque())

#-----Nom intéressant à choisir-----
#Camuto
#Gabred
#Licobs
#Revlon

Rasasi
Lalique
Jean
Gucci
Versace
The 1
Lancome
Desel
Kenzo
Versace
Versace
Armaf
Amouage
Barlow
```

Cependant, notre modèle arrive à générer du texte qui résonne dans le même type de dataset, c'est-à-dire qu'il arrive à respecter le placement des voyelles et respecte le pattern des pseudo marketing utilisé par ces distributeurs pour éviter d'avoir des mots incompréhensibles comme "krtknkj" mais il ne sait pas si un mot a du sens ou non, ceci peut ne pas gêner pour l'attribution du nom d'une marque mais pas pour le nom du parfum (la référence) qui pour le coup, lui, a toujours un sens.

Nous avons donc fait tourner la machine plus longtemps que prévu pour générer plusieurs références ainsi pouvoir filtrer manuellement celles qui ont du sens.

Voici donc notre produit final :

Gabred - Vanilla legend Eau de Parfum Mixte 100ml.

Recette : CI 47005 (D&C YELLOW NO,PARFUM (FRAGRANCE) ,AQUA (WATER/EAU) ,BENZYL BENZOATE,HYDROXYCITRONELLAL ,B217785/1) ,ALPHA-ISOMETHYL IONONE,CI 77163 (BISMUTH OXYCHLORIDE) ,CI 60730 (EXT VIOLET 2),SUCROSE ACETATE ISOBUTYRATE ,C YELLOW NO

Pour aller plus loin, nous voulions également générer un design de parfum avec un dataset de photo de parfum.



IX. Outils de Workflow utilisé:

Github :

The screenshot shows a GitHub repository page for a project named 'FRAGRAI'. The repository is owned by 'sylkex' and has 33 commits. The main branch is 'main'. The repository contains several files and folders, including 'DataScrap', 'Static', 'FRAGRAI.py', 'README.md', 'ENV_task.py', 'dataset.csv', and 'requirements.txt'. The repository is described as 'Create fragrance with IA'. The 'About' section shows the repository is public and has 0 stars and 1 fork. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows the repository is written in Python (2.2%) and Jupyter Notebook (97.8%). The repository is currently in a state of 'Ready to publish'.

FRAGRAI

Create fragrance with IA












But du Projet

Prédire la recette et générer un nom d'un parfum à partir d'un dataset de recette de parfum + la note des clients + nom du parfum et référence

Résultat Final

Gabriel - Vanilla Legend Eau de Parfum Mixte 100ml

Monday :

| | | | |
|--|--|------------|--------------------|
| <div>> FRAGRAI ● ☆</div> <div>Ajouter une description au tableau</div> <div><div>🏠 Par défaut</div><div>+</div></div> | | | |
| <div>Ajouter Élément</div> <div>Recher...</div> <div>Personne</div> <div>Filtre</div> <div>Trier</div> <div></div> <div></div> <div></div> <div></div> | | | |
| Web Scrapping | Personne | Statut | Période |
| Collecte data Notino |  RW | Fait | mars 12 |
| Collecte data Sephora |  | Pas encore | févr. 18 - 24 |
| Collecte data Nocibe |  | Pas encore | févr. 10 - 19 |
| + Ajouter Élément | | | |
| | | | févr. 10 - mars 12 |
| Data Cleaning | Personne | Statut | Période |
| Enlever les recette qui n'ont pas d'avis (note de 0) |  | Fait | mars 13 |
| Enlever les ingrédients non représentatif (moins de 2ingr) |  | Fait | mars 15 |
| Enlever tout les doublons de recette |  | Fait | mars 15 |
| + Ajouter Élément | | | |
| | | | mars 13 - 15 |
| Prédiction de la recette | Personne | Statut | Période |
| Optimisation KMeans |  | Fait | mars 24 - avr. 18 |
| Add hyperparamètre |  | Fait | mars 24 - avr. 18 |
| + Ajouter Élément | | | |
| | | | mars 24 - avr. 18 |
| RNN (générations d'un nom de parfum) | Personne | Statut | Période |
| Prog complet |  | Fait | mars 24 - avr. 18 |
| filtrer les nom |  | Fait | - |
| éliminer les nom existant |  | Fait | - |

Pour l'extraction de données : python (Librairie BS4)

Pour l'algorithme de la machine et le deep learning : Les librairies sickit learn et tensorflow.

X. Conclusion

Durant ce projet, nous avons utilisé au maximum les fonctions build in des librairies importées pour apporter un maximum de clarté dans notre code et éviter d'avoir un notebook avec plusieurs centaines de lignes. De plus, on a importé des librairies qui sont écrites en C, contenant probablement 10 000 lignes de code et bénéficient d'un calcul rapide donc autant en profiter.

On a été confronté à un nouveau type de travail, nous avons recherché beaucoup d'informations sur des sites divers et regardé un grand nombre de tutoriels informatiques. Le travail consistait plus à un avancement collectif qu'à une répartition des tâches vu la nature du travail, cela a demandé une forte communication au sein du groupe.

Nous avons beaucoup aimé faire ce projet en groupe, nous avons beaucoup appris. Depuis le début, nous avons été très inspiré pour la fin, dans le meilleur des cas avec plus de temps, nous aurions voulu générer un design pour le parfum afin

de dire qu'on a vraiment généré un parfum de A à Z. Nous essayerons sûrement de le faire durant notre temps libre.

XI. Glossaire

- DL/data cleaning : Processus de détection et de corrections d'erreurs présent dans notre dataset.
- RNN : Réseau de neurones récurrents (Algorithme de deep learning).
- Feature-Engineering : Transformer les données en format acceptable par les algos.
- Github/monday: Outils de Workflow (travail de groupe).
- Pipeline: Code d'avancement d'un projet de machine learning.
- DF/DataFrame : Tableau de données.
- Pre-processing : Etape de la pipeline comprend le data cleaning.
- TF/TensorFlow : Librairie python qui regroupe beaucoup de modèle de réseau de neurones.

Sources :

- <https://www.kaggle.com/code/ishivinal/arabic-name-generator> (RNN inspiration)
- <https://github.com/simon-larsson/pokemon-name-generator> (RNN inspiration)

- <https://github.com/antonio-f/Generating-names-with-RNN/blob/master/Generating%20names%20with%20recurrent%20neural%20networks/RNN-task.ipynb> (RNN inspiration)
- <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/> (Adam optimization)
- https://keras.io/guides/sequential_model/ (Keras)(RNN inspiration)
- <https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/discussion/62442#410577> (Tensorflow)
- [https://fr.wikipedia.org/wiki/Silhouette_\(clustering\)](https://fr.wikipedia.org/wiki/Silhouette_(clustering)) (Silhouette)
- <https://www.scikit-yb.org/en/latest/api/cluster/elbow.html#:~:text=The%20elbow%20method%20runs%20k,point%20to%20its%20assigned%20center.> (Elbow)
- <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> (KMeans)