Classer automatiquement des biens de consommation

Problématique

L'entreprise « Place de marché » souhaite lancer une marketplace e-commerce. Actuellement, les vendeurs y proposent des articles en postant une photo et une description. **La catégorie d'un article est entrée manuellement** par celui qui le met en vente et, pour l'heure, le volume de produits proposés sur la plateforme demeure faible. Dans le but de rendre l'expérience utilisateur des vendeurs et des acheteurs la plus simple et la plus fluide possible, ainsi que dans l'optique d'un passage à l'échelle sur un volume d'articles plus important, il devient nécessaire d'**automatiser cette tâche**.

Objectifs

Pour répondre à notre problématique, ce projet se déroule en 3 temps :

1. Tout d'abord, nous allons **étudier la faisabilité d'un moteur de classification d'articles** basé sur les images et les descriptions textuelles qui leur sont associées.

2. Ensuite, si le résultat de l'étude s'avère positif pour la création d'un **moteur de classification basé sur les images de description des produits,** une étude plus approfondie, visant à comparer différentes techniques et modèles pour extraire les features des images, sera réalisée.

3. Enfin**,** dans le but d'élargir la gamme de produits de la plateforme dans l'épicerie fine, il est souhaité de **tester la collecte de produits** à base de « champagne » **via une API RapidAPI** en extrayant les 10 premiers produits exportés dans un fichier « .csv », contenant les données suivantes : *foodId, label, category, foodContentsLabel,* et *image* pour chaque produit.

Compétences acquises

- **Cibler les features utiles** au développement des modèles de classification et les transformer lorsque nécessaire (EDA).
- Mettre en œuvre des **techniques de réduction de dimension**.
- Manipuler des **données non structurées** (textes et images).
- **Déterminer la faisabilité d'un moteur de classification automatique** en amont de son développement.
- Emploi de techniques de ***Natural Language Processing (NLP)*** et de **réseaux de neurones avec transfer learning**.

- Utiliser des techniques de type ***bag-of-words (BoW)***, *de* ***text embedding*****,** de type ***bag-of-visual-word (BoVW)*** et de **data augmentation**.
- **Interagir avec une API** en ligne.

Table des matières

I) Découverte du jeu de données

1) Description du jeu de données et choix des features

Le but étant de classer les articles selon leurs textes et leurs images de description, il faut tout d'abord identifier les features textuelles, images et catégorielles.

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   uniq_id                 1050 non-null   object
 1   crawl_timestamp         1050 non-null   object
 2   product_url             1050 non-null   object
 3   product_name            1050 non-null   object
 4   product_category_tree   1050 non-null   object
 5   pid                     1050 non-null   object
 6   retail_price            1049 non-null   float64
 7   discounted_price        1049 non-null   float64
 8   image                   1050 non-null   object
 9   is_FK_Advantage_product 1050 non-null   bool
 10  description             1050 non-null   object
 11  product_rating          1050 non-null   object
 12  overall_rating          1050 non-null   object
 13  brand                   712 non-null    object
 14  product_specifications  1049 non-null   object
dtypes: bool(1), float64(2), object(12)
memory usage: 116.0+ KB
```

Figure 1 : *Jeu de données brut.*

On peut remarquer que ces 3 features ne contiennent pas de valeur nulle nous facilitant l'étape de nettoyage.

product_category_tree
["Home Furnishing >> Curtains & Accessories >>...
["Baby Care >> Baby Bath & Skin >> Baby Bath T...
["Baby Care >> Baby Bath & Skin >> Baby Bath T...

Figure 2 : *Extrait des premières entrées de la feature catégorielle.*

On peut aussi noter que la feature catégorielle contient plusieurs niveaux de catégories allant de gauche à droite, du plus global au plus spécifique, qu'il va falloir extraire.

2) Extraction des niveaux de catégorie et choix du niveau sur lequel baser l'étude

```
product_category_tree 1050 non-null object
```

```
cat_lvl_0    1050 non-null   object
cat_lvl_1    1050 non-null   object
cat_lvl_2    1047 non-null   object
cat_lvl_3    679 non-null    object
```

Figure 3 : *Extraction des 4 premiers niveaux de catégorisation.*

Une fois les 4 premiers niveaux extraits, on peut constater que seuls les 2 premiers sont complètement remplis. Le 3e niveau l'est pratiquement aussi mais au-delà on passe à moins de 60 % de remplissage.

Tableau 1 : *Nombre de catégories par niveau.*

Categories levels	Number of categories' labels
Level 0	7
Level 1	62
Level 2	242
Level 3	350

Dans le tableau, on peut remarquer que plus le niveau de catégorisation est élevé plus le nombre de catégorie l'est également.

⇨ Ainsi, pour l'étude de faisabilité, le niveau 0 est sélectionné car un nombre de catégories réduit devrait permettre de mettre plus facilement en évidence la faisabilité d'une classification.

II) Étude de faisabilité

L'étude de faisabilité de la catégorisation des articles à partir de leurs descriptions se découpe en 2 parties :

1. La première étude se base sur les données textuelles.
2. Tandis que la seconde se base sur les images.

1) Données textuelles

Dans cette partie, 3 approches mettant en place plusieurs modèles ont été testées :

1. Une première simple de type « bag-of-words » qui essaie d'associer les catégories disponibles aux articles par rapport au nombre d'occurrences de certains mots-clés et leur importance dans les descriptions : **Comptage et fréquence de tokens**.

2. Une seconde plus sophistiquée qui se base sur la vectorisation des mots et des phrases dont la proximité des vecteurs correspondants dans leur espace de définition suppose une association du sens des termes observés proportionnel à cette proximité : **Vectorisation de texte (*« Text embedding »*)**.
   Dans cette approche un texte est un vecteur de vecteurs de mots préalablement créés puis stockés dans un dictionnaire à partir des *« bags-of-words »* (BoWs) représentant le ou les textes à encoder.

3. Une troisième plus complexe encore, basée sur les transformeurs (une architecture *« deep learning »*) qui proposent des modèles pré-entraînés sur divers types de jeux de données à gros voir très gros volume et permettent de capter le contexte : **Ajout du contexte**.

NB : A noter les prétraitements de texte communs comme la suppression des mots courants (stopwords), la mise en minuscule de tous les caractères et le traitement des caractères spéciaux sont réalisés au préalable pour chaque description de produit.

Étant une étude de faisabilité les algorithmes ont aussi été utilisés à leur valeur par défaut ou à des valeurs standards sans optimisation de leurs hyperparamètres.

a) Structure du texte d'une description de produit

> "[key features of elegance polyester multicolor abstract eyelet door curtain floral curtain elegance polyester multicolor abstract eyelet door curtain 213 cm in height pack of 2 price...
>
> … designs the surreal attention is sure to steal hearts these contemporary eyelet and valance curtains slide smoothly so when you draw them apart first thing in the morning to welcome the bright sun rays you want to wish good morning to the whole world and when you draw them close in the evening you create the most special moments of joyous beauty given...]"

Figure 4 : *Echantillon d'une description de produit.*

Si on jette un œil à une description de produit on peut constater que certaines parties sont écrites par mots-clés et d'autres avec beaucoup d'erreurs grammaticales qui pourraient gêner la capture du contexte.

⇨ Par conséquent, il est possible que les algorithmes utilisant le contexte pour générer leurs prédictions se voient gêner.

b) Comptage du nombre d'occurrences des mots du corpus par document

i. Comptage et tf-idf

Tableau 2 : *Scores ARI obtenus après comptage et tf-idf après différents prétraitements de texte.*

Preprocessed BoW	Count		Tf-idf	
	ARI	Time (s)	ARI	Time (s)
Not-normalized	0.411	6	0.439	5
Stemmed	0.417	6	0.431	5
Lemmatized	0.445	6	0.512	5

On peut constater que la normalisation apporte une amélioration du score ARI : la lemmatisation tout particulièrement.
De plus, le nombre d'occurrences d'un mot dans un document ramené à sa fréquence d'apparition dans tout le corpus (tf-idf), améliore le score par rapport à un simple comptage de la fréquence d'apparition du mot dans le document.

⇨ Par conséquent, il semble que la **meilleure configuration** est de réaliser un **tf-idf** sur une description de produit **lemmatisée** au préalable.

NB : Le score ARI mesure la correspondance entre les points appartenant à une même catégorie réelle (voir figure 4.1 ci-dessous) et ceux regroupés dans une même catégorie par K-Means (voir figure 4.2).

ii. Lemmatisé + tf-idf

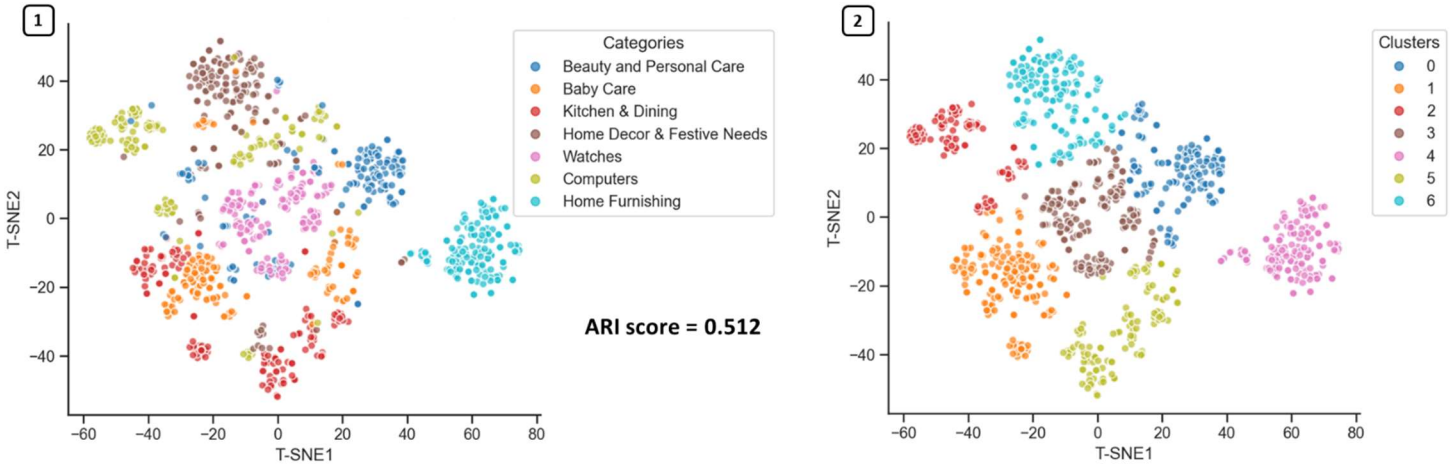


Figure 5 : *Projection t-SNE des descriptions lemmatisées puis représentées par la méthode tf-idf.*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

Si on jette un œil sur la projection t-SNE des produits colorisés à partir de leur catégorie réelle sur la figure de gauche, on remarque immédiatement qu'ils sont regroupés en plusieurs clusters et certains d'entre eux correspondent en grande majorité, voir pratiquement exclusivement, à une seule catégorie.
On observe aussi, des groupes de plus petits clusters ou des petits clusters plus éparpillés qui correspondent aussi à une même catégorie globale.

⇨ Ils représentent probablement des sous-catégories de la catégorie principale.

Sur la figure de droite, on peut observer le résultat de la clusterisation réalisé par K-Means (K-Means++). Ce type de figure aide à mieux distinguer les différents clusters de produits sans se soucier de leurs catégories. On peut remarquer que la majorité des gros clusters de produits faisant partie d'une même catégorie sont bien isolés par la clusterisation K-Means. Il semble même possible de pouvoir affiner la clusterisation en passant à un niveau de catégorie supérieur et/ou en modifiant les paramètres du modèle de clusterisation.

⇨ Ainsi, il semble que la majorité des clusters représentent une catégorie spécifique de produit et la clusterisation Kmeans tend à montrer qu'un algorithme est capable de distinguer les différents clusters de produits les uns des autres.

⇨ Il apparaît donc que catégoriser les produits part leurs descriptions respectives semble possible.

Cette méthode d'encodage montre qu'il est possible de prédire les différentes catégories de produits grâce à leurs descriptions textuelles. Cependant, cette technique se concentre simplement sur la présence de tokens dans une description, alors que des méthodes plus complexes permettent d'encoder plus d'informations telle que l'association de textes selon leurs « sens vectoriel global » : Word/sentence embedding.

c) Word/Sentence embedding

En quelques mots, la vectorisation de mots (le word embedding), que l'on peut généraliser à la vectorisation de textes, consiste à représenter chaque mot d'un dictionnaire par un vecteur de nombres réels. Les mots apparaissant dans des contextes similaires dont les vecteurs correspondants sont relativement proches dans leur espace de définition.

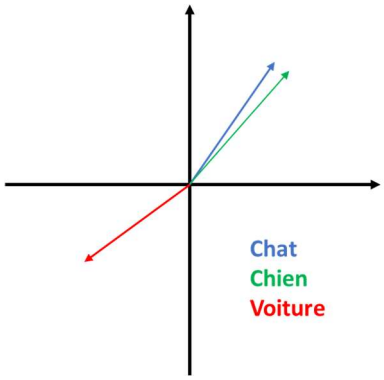


Figure 6 : *Espace des vecteurs représentant les mots (embedding space).*

Par exemple, on pourrait s'attendre à ce que les mots « chien » et « chat » soient représentés par des vecteurs relativement peu distants dans l'espace dans lequel ils sont définis, contrairement au mot « voiture ».

⇨ **Cette technique repose** sur **l'hypothèse de Harris** qui suppose que les mots qui apparaissent dans des **contextes similaires** ont des **significations apparentées**.

i. Vectorisation des descriptions avec Word2Vec

Ici, le but est de vectoriser les descriptions des produits puis de regrouper leurs vecteurs par leurs similarités spatiales dans le but de générer des clusters de produits similaires et donc de mêmes catégories.

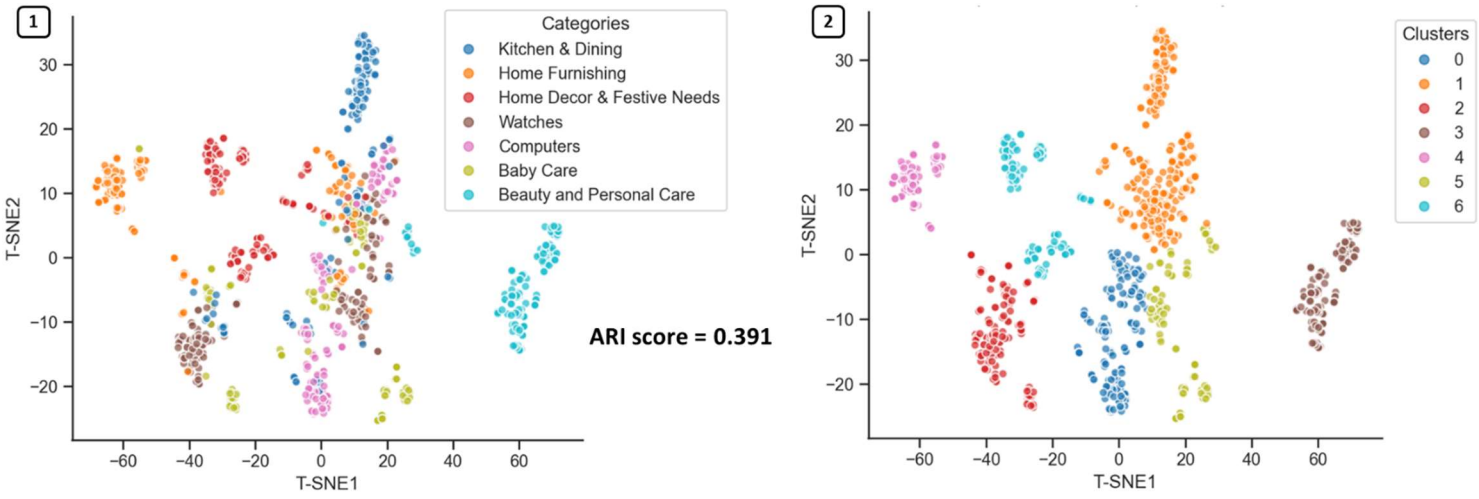


Figure 7 : *Projection t-SNE des descriptions lemmatisées puis vectorisées par Word2Vec variante « continuous bag-of-words » (CBOW).*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

Comme précédemment, on constate bien que différents clusters correspondent à une même catégorie. Certains paraissent même mieux définis et plus distants des autres. Cependant, certains clusters de produits semblent appartenir à plus de catégories différentes ce qui se remarque dans la valeur du score ARI 20% plus bas que précédemment.
NB : A noter que l'étape de lemmatisation améliore le score de 10 % pour atteindre les 0.39 et que la variante skip-gram affiche un score ARI de 0.35 dans la même configuration.

⇨ En sélectionnant des catégories de niveau supérieures et plus nombreuses, il serait peut-être possible d'améliorer les résultats. Toutefois, avec ce nouveau jeu de catégories le score ARI obtenu précédemment s'améliorerait probablement aussi.
L'augmentation de l'hétérogénéité globale de la répartition des catégories de produits peut aussi s'expliquer par le fonctionnement propre à Word2Vec. Le vecteur représentant un token est formé en regardant ses voisins contextuels mais est utilisé par la suite de manière non-contextuelle au cours d'une tâche NLP. Autrement dit, en sortie Word2Vec n'enregistre qu'un seul vecteur pour représenter le token à encoder limitant sa capacité à saisir le sens d'un mot dans deux contextes différents (**ex :** "river bank" et "bank deposit" ou "apple macbook" et "sweet apple"). Ainsi, cette limitation peut engendrer de mauvaises associations (ou dissociations) de tokens selon le contexte et entraîner une mauvaise catégorisation du produit.

Dans le but d'améliorer le score, il peut être intéressant de considérer une nouvelle approche de vectorisation qui, cette fois-ci, tient aussi compte du contexte dans lequel se situent les mots.

d) Ajout de la contextualité

Word2Vec enregistre une seule représentation vectorielle d'un mot, tandis que BERT génère un vecteur pour représenter un mot en fonction de la façon dont le mot est utilisé dans une phrase.

Supposons que nous ayons deux phrases : 1. *Le chat a peur du chien*, 2. *Le chien a peur du chat*. Les deux phrases contiennent les mêmes mots-clés (*Chat, Chien* et *Peur*) mais dans des arrangements différents. Si on fait la moyenne de tous les vecteurs représentant chaque mot pour obtenir le vecteur de la phrase. On obtiendra certainement le même vecteur pour représenter ces 2 phrases puisque les mots qui les constituent sont les mêmes. Cependant, ces 2 phrases ne signifient pas la même chose : le contexte est différent dans chacun des deux cas.

⇨ Ainsi, pour capturer le contexte de la phrase entière nous allons utiliser le modèle pré-entraîné *« Bidirectional Encoder Representations from Transformers »* (BERT) et l'Encodeur de Phrase Universel (USE) développés par Google pour générer des incorporations de phrases.

i. Bidirectional Encoder Representations from Transformers (BERT)

Dans le cas de BERT, 2 versions ont été testées celle de :
- HuggingFace
- Tensorflow hub

Dans chacun des 2 cas, on part d'un modèle BERT pré-entraîné sur des mots de la langue anglaise (grâce au module *« bert-base-uncased »*) par la méthode de *« Masked Language Modeling »* (MLM).

➢ **HuggingFace**

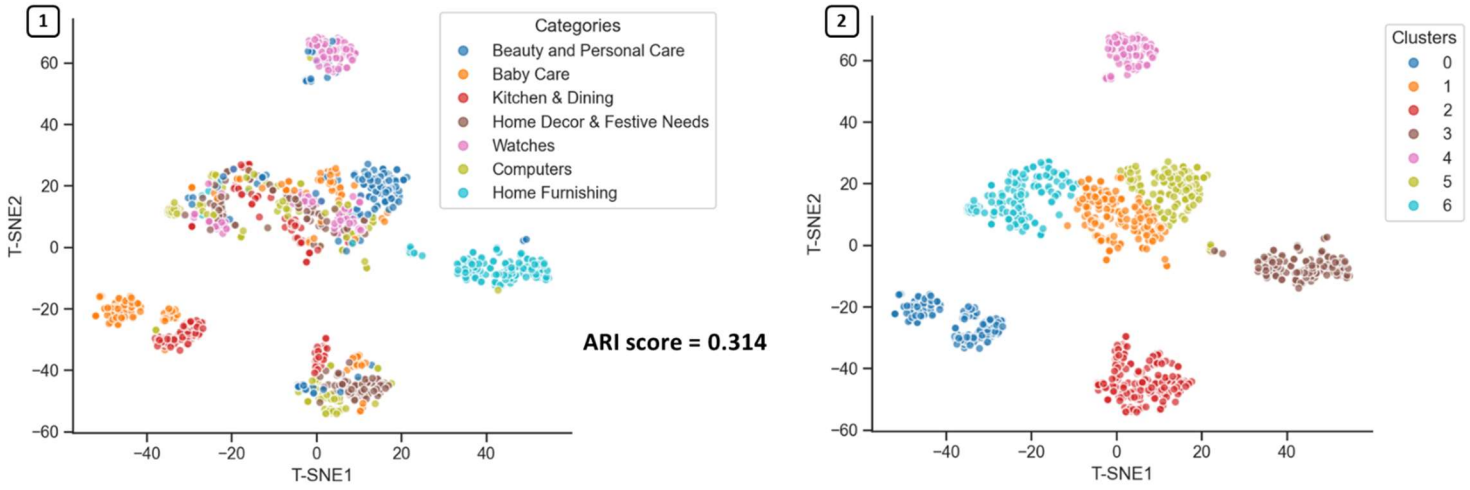


Figure 8 : *Projection t-SNE des descriptions non normalisées avant la vectorisation.*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

À première vue, les clusters semblent mieux isolés que précédemment. Toutefois, la répartition des catégories est plus hétérogène encore, comme le montre le score ARI (0.31).

➢ **Tensorflow hub**

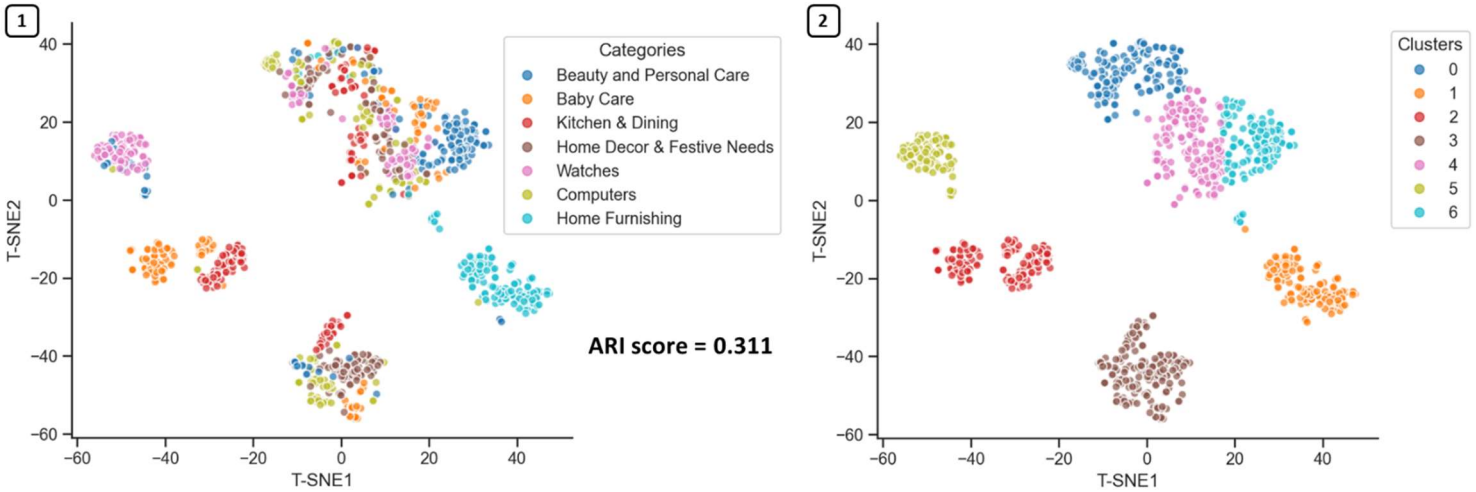


Figure 9 : *Projection t-SNE des descriptions non normalisées avant la vectorisation.*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

Bien que la position des clusters ait sensiblement changé sur la projection t-SNE, les résultats sont finalement très proches de ceux obtenus avec la version de HuggingFace.

NB : Il serait certainement possible d'améliorer les résultats obtenus avec BERT si une version pré-entraînée sur un jeu de données structurellement similaire à nos descriptions était trouvée.

ii. Universal Sentence Encoder (USE)

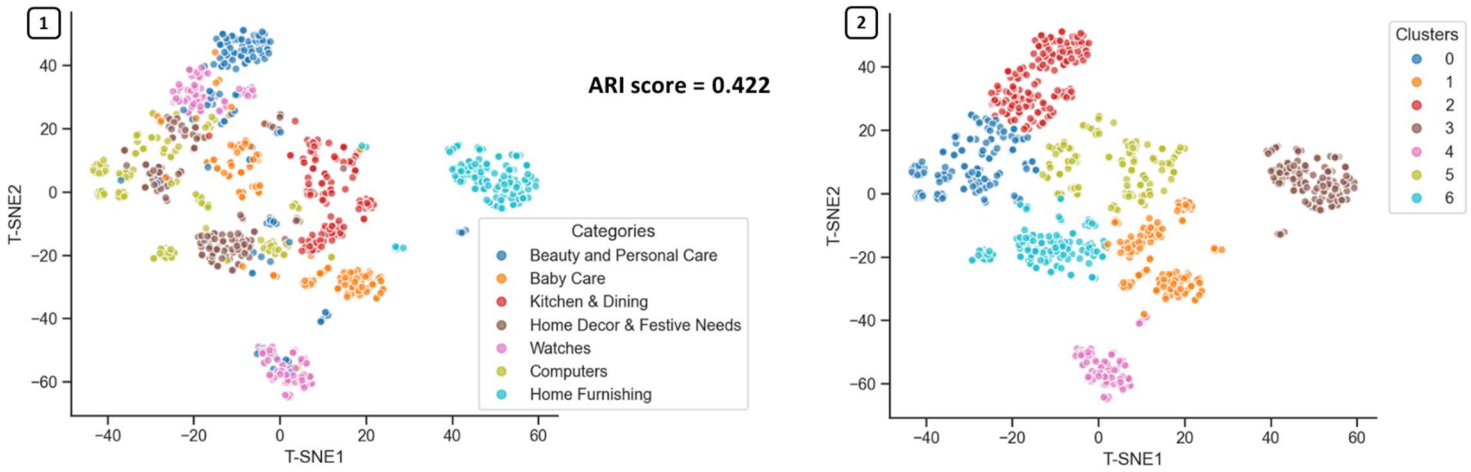


Figure 10 : *Projection t-SNE des descriptions non normalisées avant la vectorisation.*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

Il apparaît qu'USE se débrouille mieux que BERT même si visuellement les clusters paraissent plus diffus. Toutefois, USE affiche un score toujours inférieur à celui obtenu avec tf-idf et reste proche, en termes de répartition, de celui obtenu avec Word2Vec.

⇨ On peut penser que les scores obtenus s'expliquent par des descriptions davantage structurées pour accrocher rapidement le regard et susciter l'envie chez le lecteur par des mots-clés ou de petits morceaux de phrases mis bout à bout plutôt qu'en texte grammaticalement et syntaxiquement correctes. De plus, ce genre de structure peut favoriser des techniques comme tf-idf qui se concentre seulement sur la fréquence d'apparitions de mots-clés dans une description sans tenir compte du contexte.

e) Conclusion

Tableau 3 : *Scores ARI obtenus pour les différents modèles testés avec et sans lemmatisation préalable.*

Models	Preprocessed BoW	ARI scores
Tf-idf	**Lemmatized**	0.51
W2V	**Not-normalized**	0.30
	Lemmatized	0.39
BERT	**Not-normalized**	0.31
	Lemmatized	0.31
USE	**Not-normalized**	0.42
	Lemmatized	0.45

Si l'on compare les résultats obtenus avec les différents modèles d'embedding on peut remarquer que la lemmatisation apporte un gain de performance significatif avec Word2Vec et USE mais pas avec BERT.

> ⇨ **Finalement, cette première analyse tend à montrer qu'il est bien possible de classer les produits d'après leurs descriptions et que la méthode le plus prometteuse est l'application d'un tf-idf sur un BoW lemmatisé qui saura caractériser chaque description par les mots-clés les plus pertinents.**

NB : Bien sûr, ces résultats sont à relativiser. Il existe bien d'autres modèles plus spécifiques dont la plupart sont des forks de ceux utilisés ici. Sans parler de l'optimisation des hyperparamètres ou l'affinement du niveau de catégorisation qui n'est pas évalué dans ce projet.

1) Données visuelles

a) Extraction des features visuelles

Pour évaluer la faisabilité d'une classification avec les données visuelles, il nous faut récupérer les features de chaque image.

Figure 11 : *Echantillon d'images de produits appartenant à la catégorie « Watches » et à la catégorie « Home decors & festive needs ».*

Pour ce faire, l'avant-dernière couche du modèle VGG16 est sélectionnée comme couche de sortie pour récupérer les 4096 features de l'image puis, le nombre de features est réduit aux plus importantes par ACP, pour arriver à 24, dans le but d'alléger grandement la consommation en ressource pour la suite.

b) Mesure du score ARI

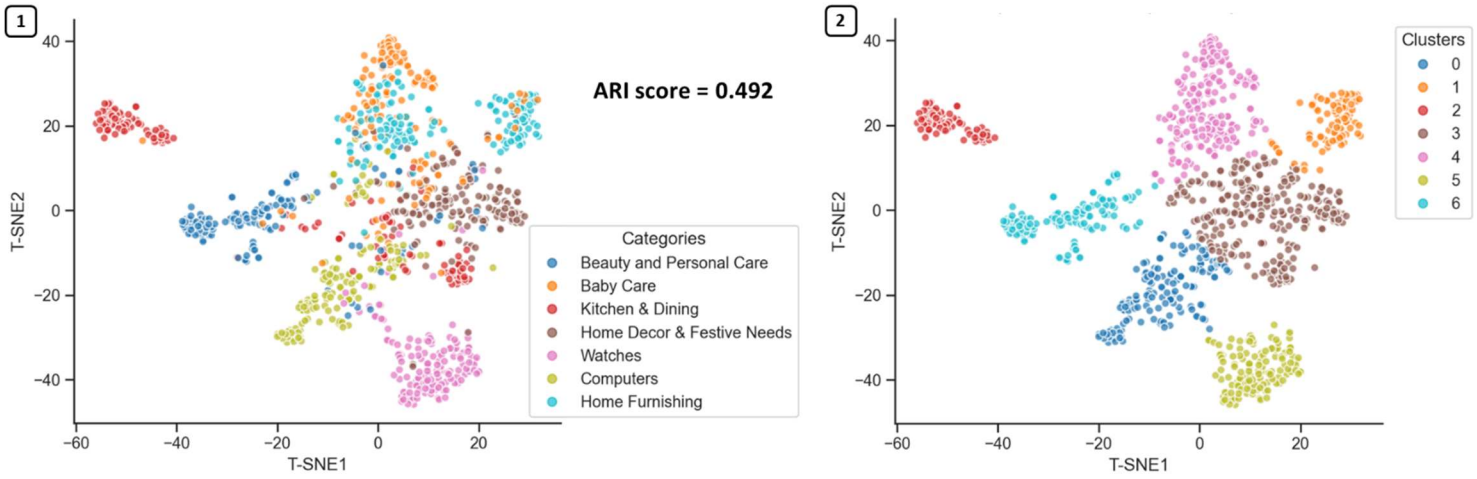


Figure 12 : *Projection t-SNE des produits regroupés d'après leurs images de présentation.*

1. *Une couleur par catégorie d'appartenance réelle.*
2. *Une couleur par cluster (Kmeans++ avec K = Nombre de catégories possibles).*

On constate tout de suite que le score ARI est similaire à celui obtenu lors de l'étude précédente par l'emploi de techniques d'extraction de features de type BoW. Il est possible de distinguer certains clusters sur la projection t-SNE, quelques-uns sont même exclusifs à une catégorie. Bien que quelques points répartis de façon plus diffuse entraînent des chevauchements entre clusters. Comme discuté dans la partie précédente, une catégorisation des articles plus spécifiques devrait certainement atténuer ce phénomène.

⇨ **La répartition des catégories en clusters montre bien qu'il est aussi possible de classer les produits grâce à leurs images de descriptions et pourrait donner des résultats, à priori, équivalent voir meilleurs à ceux que l'on obtiendrait avec l'analyse des descriptifs textuels selon la méthode utilisée.**

III) Classification supervisée des données visuelles

Deux approches d'extraction des features et de classification sont testées :

1. Une première approche, **sans réseau de neurones**, de type ***« bag-of-visual-words »*** **(BoVW)** suivi d'un classifieur **k-NN**.

2. Une seconde approche, **avec emploi d'un réseau de neurone**, de type ***« Convolutional neural network »*** **(CNN) transfert learning**.

1) Création des sous-datasets d'images

Avant toute chose, le dataset d'images est scindé en 3 sous-datasets :
- Un train set avec 70 % des images pour entraîner le modèle.
- Un validation set avec 15 % des images pour évaluer et comparer les modèles.

- Un test set avec 15 % des images pour évaluer la capacité de généralisation des modèles.

NB : Chaque séparation s'est faite avec stratification pour conserver au mieux la proportion relative entre chaque catégorie dans chacun des sous-datasets.

2) Sans réseaux de neurones : Solution de type BoVW

a) Extraction des features visuelles

Tout d'abord, pour extraire les features, plusieurs techniques peuvent être utilisées. Le choix s'est toutefois porté sur SIFT qui permet de stocker les descripteurs sur 128 dimensions contrairement à ORB qui ne permet que de les stocker sur 32.
NB : SURF n'est actuellement pas présent dans la librairie opencv-contrib-python.

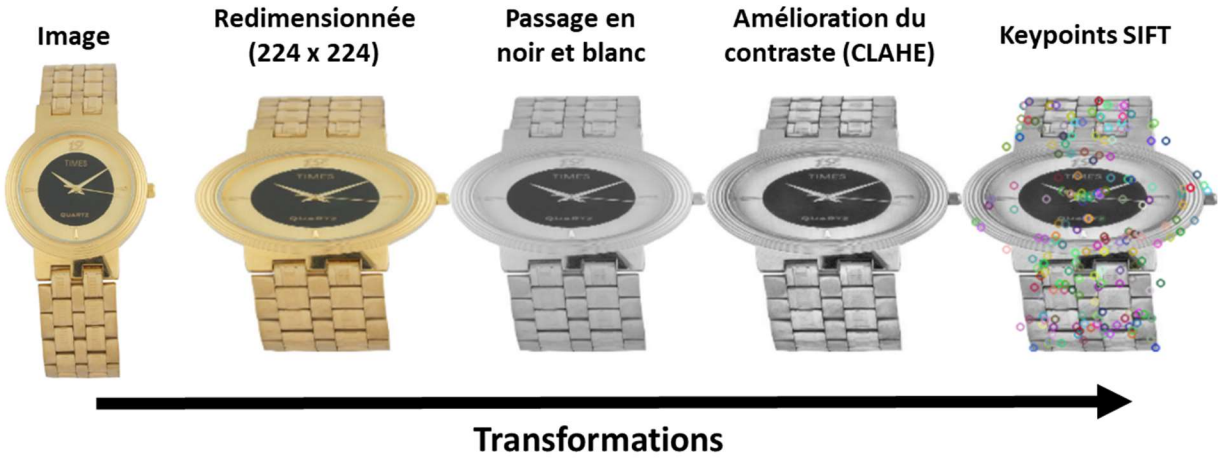


Figure 14 : *Image échantillon d'origine et à chaque étape du prétraitement avec les zones des keypoints trouvées.*

Une fois chargée le prétraitement d'une image se fait en 3 étapes avant l'extraction de ses features :

1. En premier lieu, elle est redimensionnée puis,
2. On ne sélectionne que le channel des niveaux de gris,
3. Le contraste est amélioré avec la méthode CLAHE ensuite,
4. Enfin, on passe à l'extraction des features, à proprement parler, avec la méthode SIFT.

b) Réduction de la dimension des features extraites

Pour simplifier les calculs, une réduction de dimension est appliquée sur les features extraites par ACP.

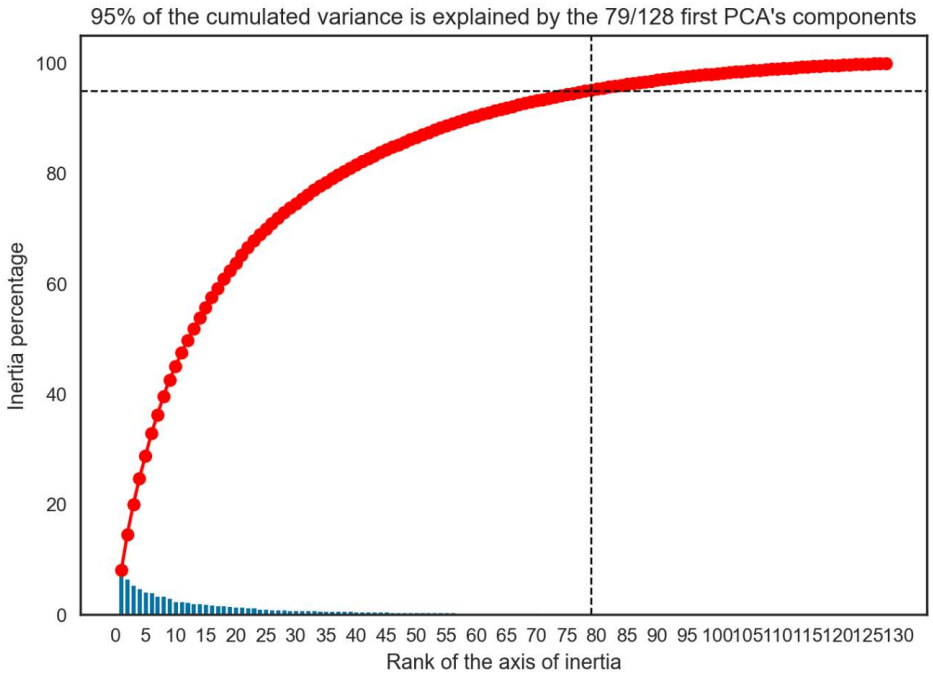


Figure 15 : *Eboulis des valeurs propres : 95 % de la variance cumulée est expliquée par les 79 premiers composants de l'ACP.*

D'après l'éboulis des valeurs propres, on peut voir qu'il est possible de n'utiliser que 61 % des dimensions des features extraites.

c) Création du bag of visual words (BoVW)

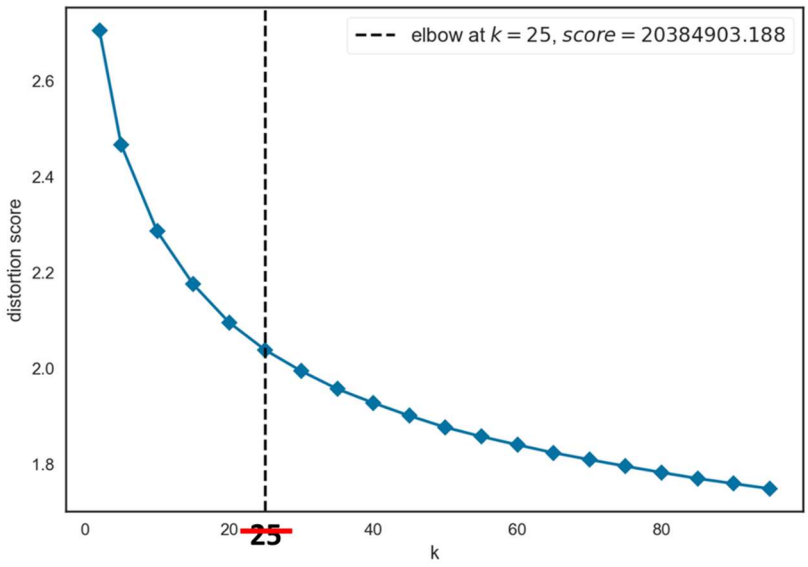


Figure 16 : *Coefficient silhouette calculé pour plusieurs nombres de clusters (K) prédéfinis.*

Dans un premier temps, pour connaître le nombre de *visual-words* à mettre dans le dictionnaire, on mesure le coefficient silhouette sur une gamme de nombres de clusters à l'initialisation puis, on choisit le nombre correspondant au coefficient silhouette le plus faible comme nombre de visual words à utiliser dans le dictionnaire.

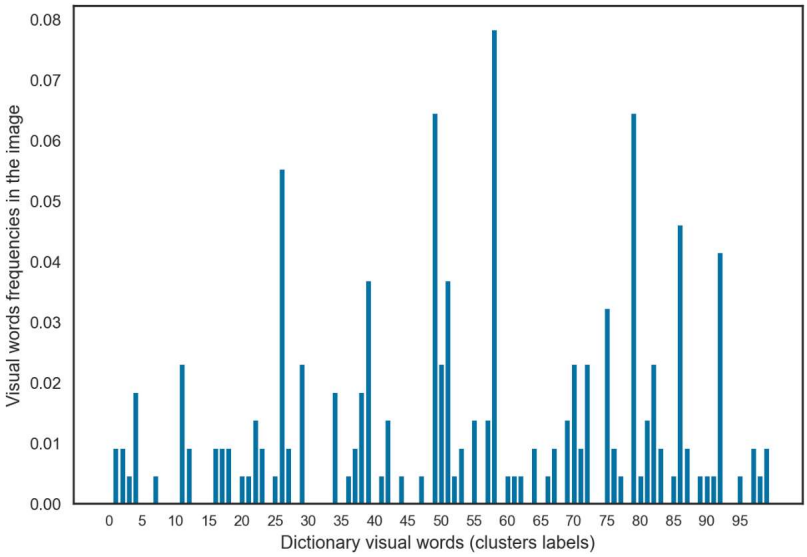


Figure 17 : *Histogramme de la fréquence de présence des 100 visual words pour l'image échantillon.*

Ensuite, il ne reste plus qu'à représenter chaque image en vecteurs dont la fréquence de présence de chaque visual word du dictionnaire est un composant.

NB : A noté que finalement le K-Means n'a pas su fournir la valeur la plus optimale qui n'est finalement pas de 25 mais plutôt de 100 mots visuels. C'est pourquoi, l'histogramme ne correspond pas à la valeur déterminée par K-Means.

d) k-NN classifieur

On va classer les images grâce à un classifieur de type k-NN.

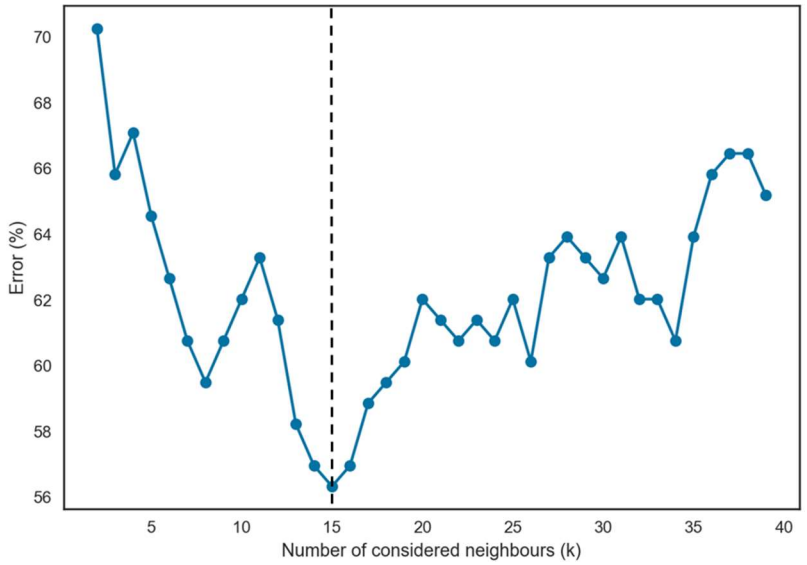


Figure 18 *: Courbe représentant l'erreur de classification selon le nombre de proches voisins considérés.*

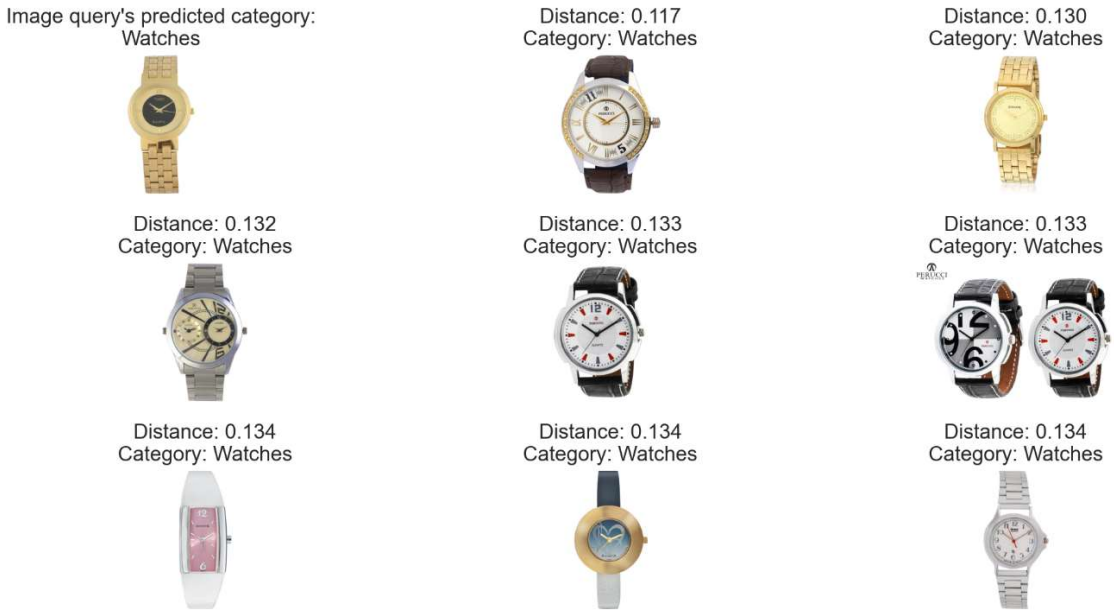


Figure 19 *: Plus proches voisins qui ont permis de déterminer la catégorie de l'image échantillon.*

Pour ce faire, on commence par mesurer l'erreur sur la précision (c'est-à-dire, le nombre d'images mal classées) à plusieurs valeurs de k pour sélectionner le meilleur.

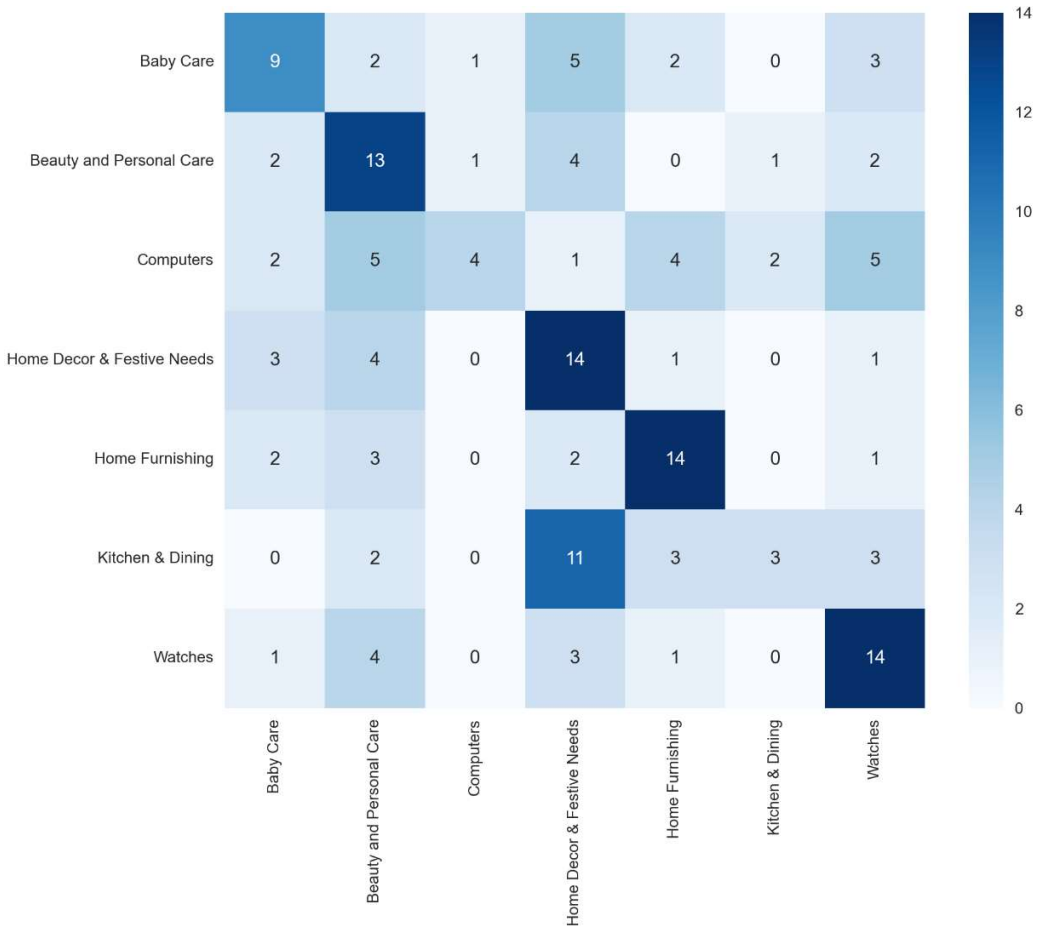


Figure 20 : *Matrice de confusion du jeu de validation sur les catégories attribuées aux produits par rapport à leurs images de description.*

On remarque que seule la moitié des catégories ont bien été attribuées et qu'il y a beaucoup de confusions pour toutes les catégories.

Figure 21 : *Exemples d'images mal classées dans « Watches » et qui devraient se trouver dans « Computers ».*

Si on regarde la catégorie la moins bien prédite (« Computers »), 7 de ses éléments ont été classés dans « Watches ». On peut remarquer que la majorité de ces articles possèdent des caractéristiques propres aux montres, comme les antennes assimilables à des aiguilles, et peuvent induire la machine en erreur.

⇨ Les catégories sont généralistes pour certaines (comme *« Home decor & festive needs »*) ou présentent des produits de formes et contrastes variables (comme *« Beauty and personal care »*) pouvant générer des images avec des vecteurs proches au niveau de leurs caractéristiques visuelles mais d'utilités bien différentes.
NB : De manière analogue, ce problème fait penser à celui rencontré précédemment avec la technique de vectorisation de type Word2Vec dans la partie NLP.

3) Avec réseau de neurones : CNN (Convolutionnal Neural Network)

3 approches ont été testées (2 sans data augmentation et une avec) :

1. La première consiste à effectuer l'étape de prétraitement des images avant leur passage dans le réseau de neurones.

2. La seconde à charger les images de chaque sous-dataset (le dataset d'entraînement, de validation et de test) en tant que dataset tensorflow avant de les passer au CNN.

3. La troisième reprend la structure précédente à laquelle une couche de data augmentation est ajoutée en amont de leur passage au CNN.

a) Modifications apportées au CNN

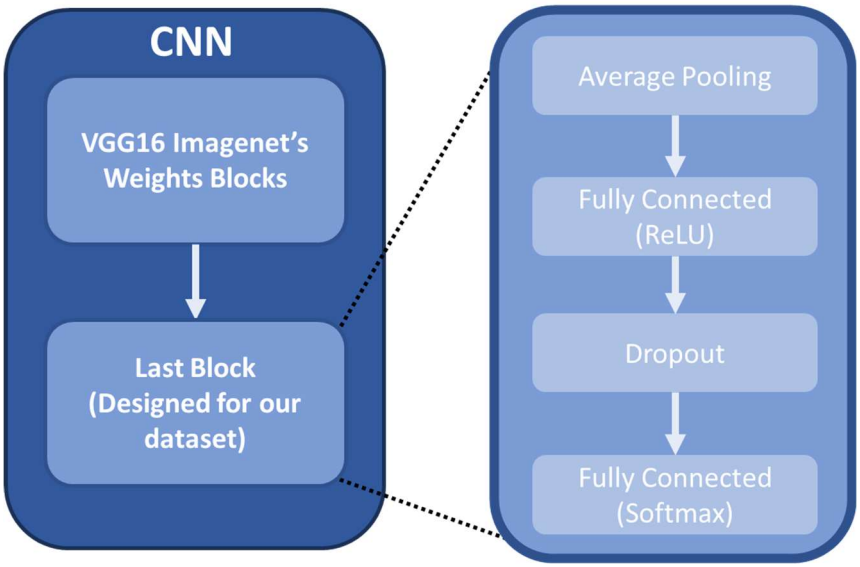


Figure 22 : *Structure du modèle CNN utilisé.*

Comme vous pouvez le voir sur la figure de droite, le modèle est construit à partir d'un VGG16 dont les poids obtenus à l'entraînement sur Imagenet ont été conservés et seul le dernier bloc a été remplacé par un bloc de 4 nouvelles couches :

1. Une couche de pooling sur la moyenne qui fait la moyenne des résultats obtenus en sortie des couches de convolution. Autrement dit, cette couche reconstitue une image avec les caractéristiques importantes enregistrées jusque-là.

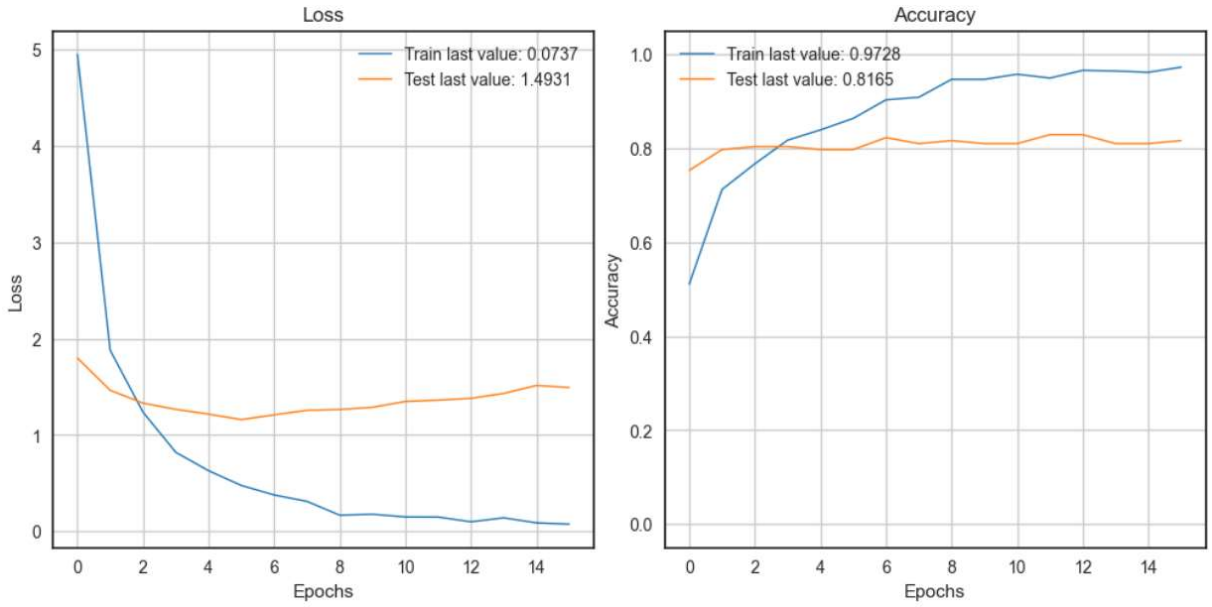
2. Une couche fully connected avec une fonction d'activation de type ReLU pour accélérer l'entraînement du réseau.

3. Une couche dropout pour la partie MLM réglée à 50%.

4. Et la dernière couche fully connected avec une fonction d'activation de type softmax qui va classer les images selon les données lues provenant de l'avant-dernière couche.

b) Approche simple en 2 temps

Dans la première approche, les images sont chargées avec Keras, puis sont encodées en tableau numpy auxquels on ajoute une dimension pour que le modèle puisse leur attribuer un batch. Enfin, elles sont prétraitées pour être utilisées par le modèle. Ce dernier est réglé sur 50 epochs avec un *« early stopping »* qui arrête l'entraînement lorsque le score donné par la fonction loss (basée sur l'entropie croisée) n'a pas diminué pendant plus de 10 epochs consécutives.



0.98	1.0
0.80	0.82
0.80	0.82

Figure 23 : *Historique de l'apprentissage du modèle à chaque epoch.*
NB : *« Test » dans la légende correspond en fait à « Validation » sur le validation set.*

Sur la figure ci-dessus, on peut remarquer qu'on obtient de plutôt bons résultats sur les prédictions et une bonne capacité du modèle à généraliser, toutefois il reste sujet à l'overfitting.

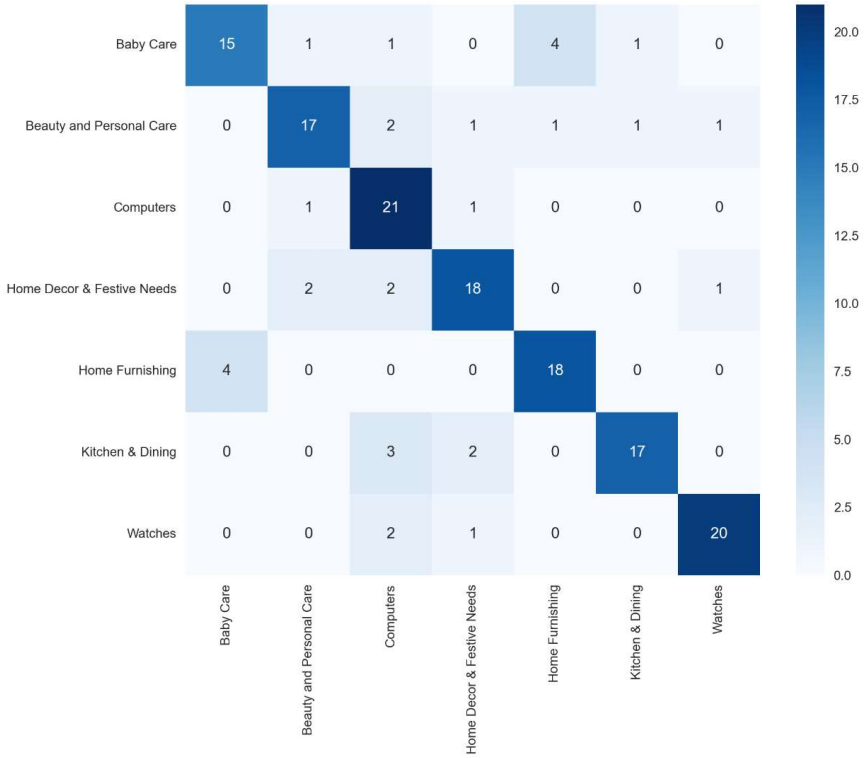
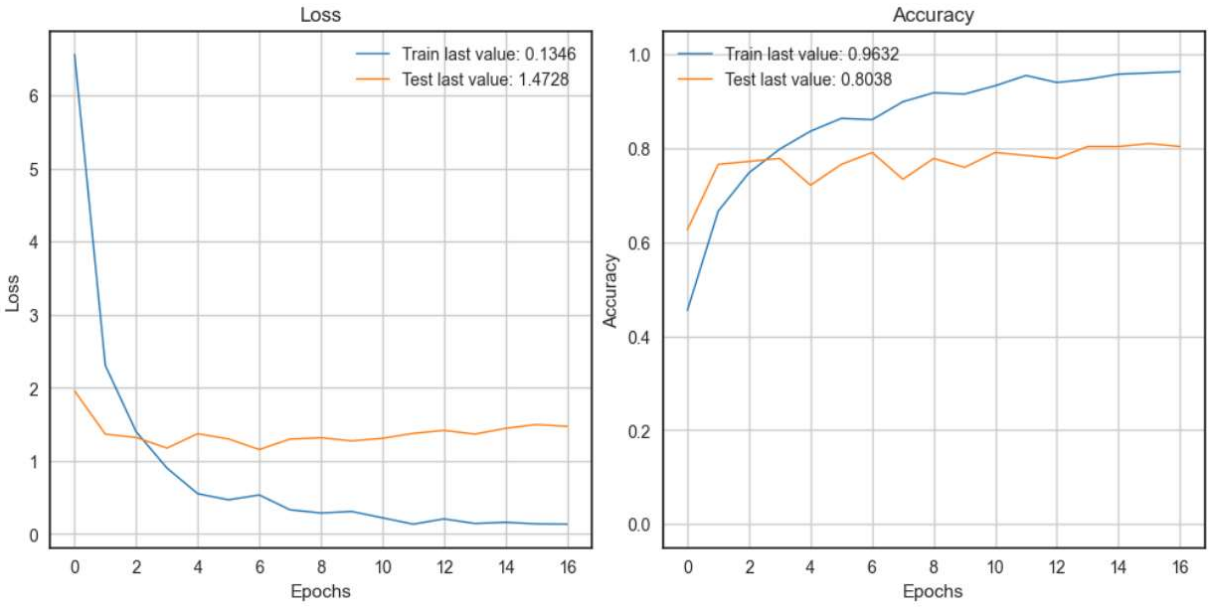


Figure 24 : *Matrice de confusion du jeu de validation sur les catégories attribuées aux produits par rapport à leurs images de description.*

La matrice de confusion obtenue sur le jeu de données de validation confirme les résultats et montre de meilleurs résultats que ceux obtenus par la solution de type BoVW.

c) Approche par dataset tensorflow

Dans cette approche, les datasets sont directement chargés en tant que datasets tensorflow. Il n'est donc plus nécessaire de charger et de prétraiter les images à la main avant de les donner au modèle.



0.98	1.0
0.79	0.80
0.79	0.78

Figure 25 : *Historique de l'apprentissage du modèle à chaque epoch.*
NB : *« Test » dans la légende correspond en fait à « Validation » sur le validation set.*

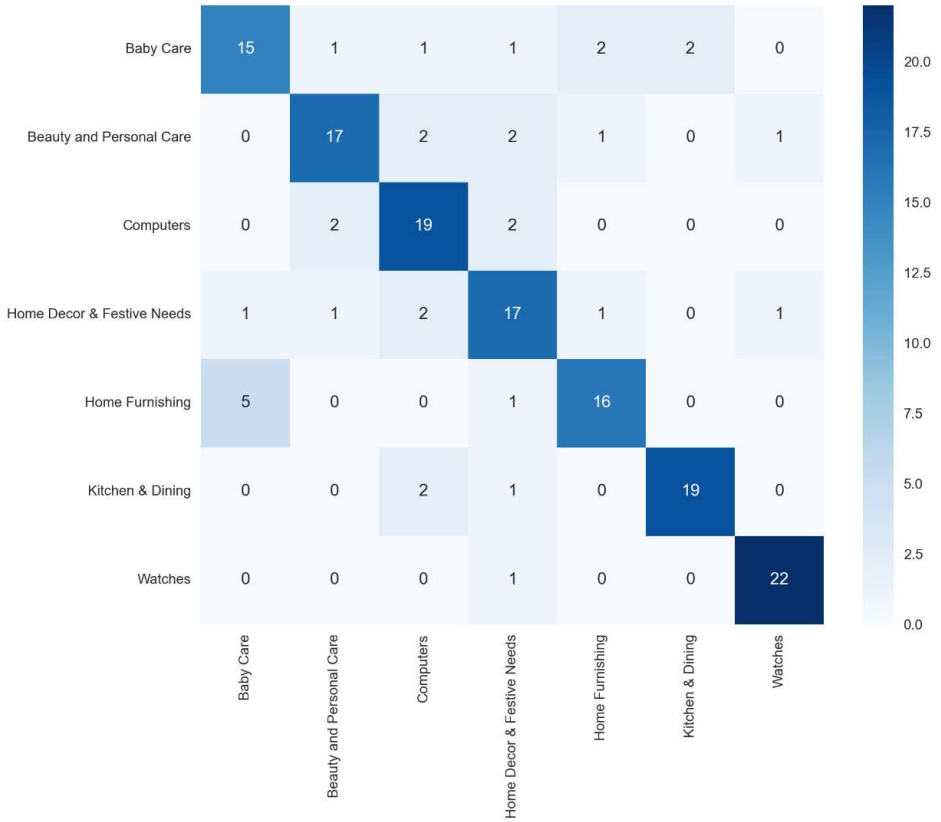


Figure 26 : *Matrice de confusion du jeu de validation sur les catégories attribuées aux produits par rapport à leurs images de description.*

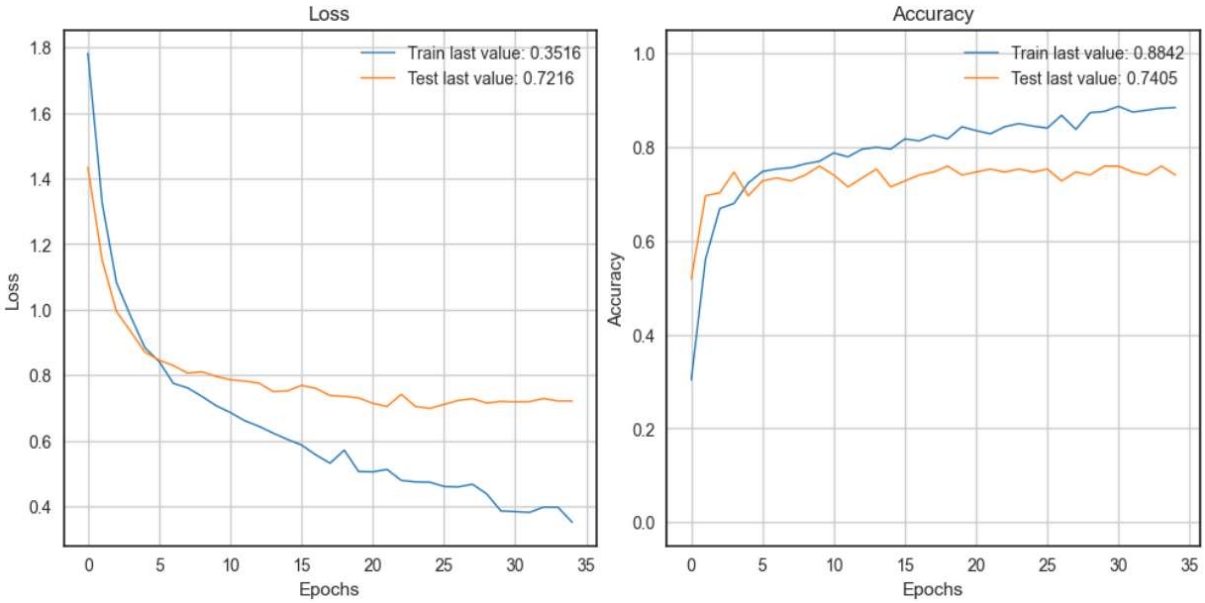
On peut faire les mêmes observations que précédemment.

⇨ Il est possible que l'overfitting soit dû à un volume de données d'entraînement trop petit.

⇨ C'est pourquoi, une troisième approche avec data augmentation est testée.

d) Approche par dataset tensorflow avec data augmentation

La structure du modèle précédent est récupérée et on y ajoute la data augmentation en amont.



0.90	0.91
0.75	0.74
0.82	0.81

Figure 27 : *Historique de l'apprentissage du modèle à chaque epoch.*
NB : *« Test » dans la légende correspond en fait à « Validation » sur le validation set.*

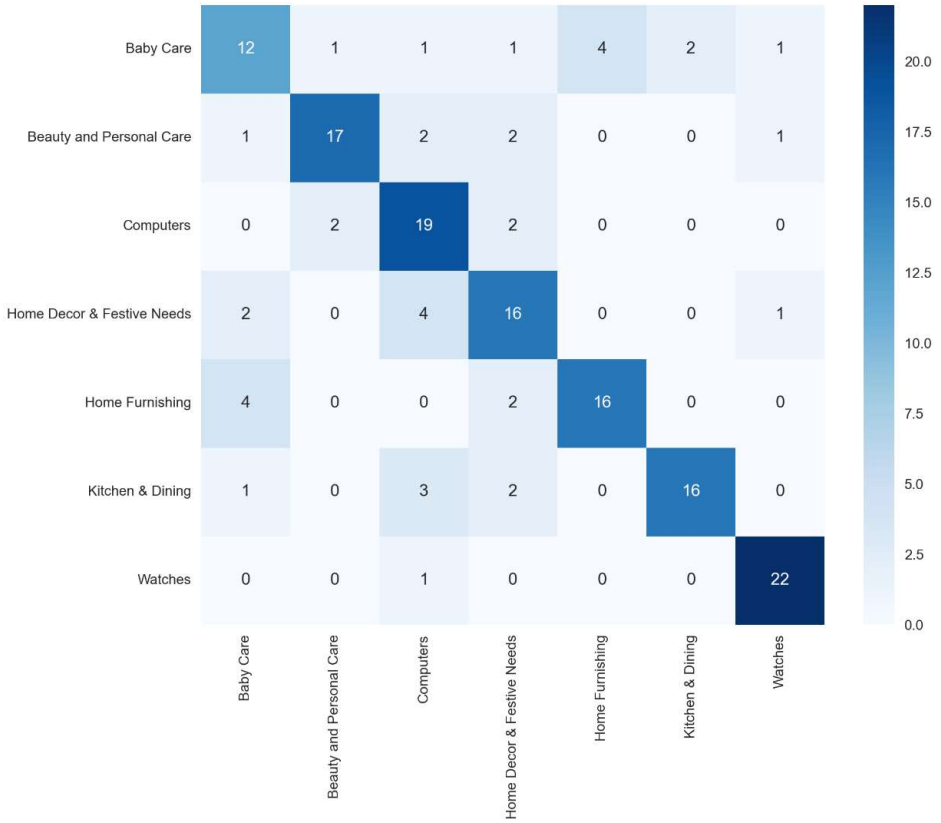


Figure 28 : *Matrice de confusion du jeu de validation sur les catégories attribuées aux produits par rapport à leurs images de description.*

Bien que sur test set le score de précision obtenu demeure très similaire à ceux obtenus au cours des approches précédentes, celui obtenu sur le jeu de validation est quant à lui plus bas d'à peu près 8 %. Néanmoins, il semble que l'étape de data augmentation a bien joué son rôle puisque le modèle overfitte 2 fois moins même au cours de la dernière epoch.

e) Conclusion

Tableau 4 : *Comparaison des modèles selon leurs scores de précision sur le jeu de validation à leurs scores de loss le plus optimal.*

Accuracy	k-NN	CNN 1	CNN 2	CNN 3
Train set	x	0.98	0.99	0.90
Validation set	0.43	0.79	0.79	0.74
Test set	x	0.79	0.78	0.81

En résumé, on constate tout de suite que les réseaux de neurones de type CNN donnent de bien meilleurs résultats comparés à l'approche BoVW + kNN.
Cependant, les 3 approches testées avec ce type de réseaux neuronaux ont abouti à des résultats très similaires. Bien que l'approche par data augmentation réduise l'overfitting de moitié.

IV) Collecte de données provenant d'une API

Avant de passer au test de l'API en lui-même, rappelons ce qu'est le RGPD et les 5 grands principes sur lesquels il repose.

1) Principes du RGPD

Le **RGPD** est le **Règlement Général sur la Protection des Données** du Parlement européen relatives à au traitement des données à caractère personnel (des personnes physiques). Il a pour but de renforcer et unifier la protection des données pour les individus au sein de l'Union européenne.
Il est basé sur 5 principes fondamentaux qui couvrent la confidentialité, le stockage, la collecte, l'entretien, la sécurité, la transparence et la souveraineté sur les données concernant les personnes physiques :

1. **Principe de finalité :** Le responsable d'un fichier ne peut enregistrer et utiliser des informations sur des personnes physiques que dans un but bien précis, légal et légitime.

2. **Principe de proportionnalité et de pertinence :** Les informations enregistrées doivent être pertinentes et strictement nécessaires au regard de la finalité du fichier.

3. **Principe d'une durée de conservation limitée :** Il n'est pas possible de conserver des informations sur des personnes physiques dans un fichier pour une durée indéfinie. Une durée

de conservation précise doit être fixée, en fonction du type d'information enregistrée et de la finalité du fichier.

4. **Principe de sécurité et de confidentialité :** Le responsable du fichier doit garantir la sécurité des informations qu'il détient. Il doit en particulier veiller à ce que seules les personnes autorisées aient accès à ces informations.

5. **Respect du droit des personnes à disposer de leurs données :** Transparence, accès, rectification, effacement, limitation du traitement et opposition.

2) Requête & réponse

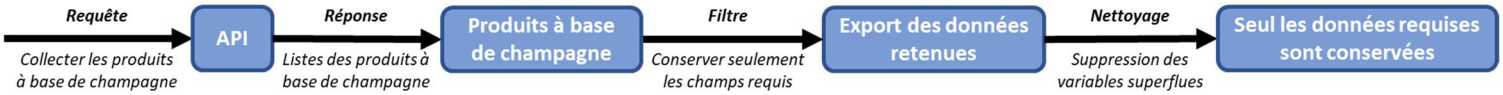


Figure 29 : *Schéma représentant les principales étapes de la collecte, du traitement et du stockage des données.*

Pour tester l'API, la requête formulée comprenait un filtre pour sélectionner uniquement les produits contenant du champagne, puis un second filtre a été appliqué pour collecter les champs d'intérêts, que vous pouvez voir en-tête du tableau.
Les variables contenant les données ont été supprimées une fois les 10 premiers produits exportés au format csv pour rester en conformité avec le RGPD.

Tableau 5 : *Tableau des 10 premiers produits obtenus en utilisant le mot-clé « Champagne » provenant de l'API RapidAPI.*

	foodId	label	category	foodContentsLabel	image
0	food_a656mk2a5dmqb2adiamu6beihduu	Champagne	Generic foods	NaN	https://www.edamam.com/food-img/a71/a718cf3c52...
1	food_b753ithamdb8psbt0w2k9aquo06c	Champagne Vinaigrette, Champagne	Packaged foods	OLIVE OIL; BALSAMIC VINEGAR; CHAMPAGNE VINEGAR...	NaN
2	food_b3dyababjo54xobm6r8jzbghjgqe	Champagne Vinaigrette, Champagne	Packaged foods	INGREDIENTS: WATER; CANOLA OIL; CHAMPAGNE VINE...	https://www.edamam.com/food-img/d88/d88b64d973...
3	food_a9e0ghsamvoc45bwa2ybsa3gken9	Champagne Vinaigrette, Champagne	Packaged foods	CANOLA AND SOYBEAN OIL; WHITE WINE (CONTAINS S...	NaN
4	food_an4jjueaucpus2a3u1ni8auhe7q9	Champagne Vinaigrette, Champagne	Packaged foods	WATER; CANOLA AND SOYBEAN OIL; WHITE WINE (CON...	NaN
5	food_bmu5dmkazwuvpaa5prh1daa8jxs0	Champagne Dressing, Champagne	Packaged foods	SOYBEAN OIL; WHITE WINE (PRESERVED WITH SULFIT...	https://www.edamam.com/food-img/ab2/ab2459fc2a...
6	food_alpl44taoyv11ra0lic1qa8xculi	Champagne Buttercream	Generic meals	sugar; butter; shortening; vanilla; champagne;...	NaN
7	food_byap67hab6evc3a0f9w1oag3s0qf	Champagne Sorbet	Generic meals	Sugar; Lemon juice; brandy; Champagne; Peach	NaN
8	food_am5egz6aq3fpjlaf8xpkdbc2asis	Champagne Truffles	Generic meals	butter; cocoa; sweetened condensed milk; vanil...	NaN
9	food_bcz8rhiajk1fuva0vkfmeakbouc0	Champagne Vinaigrette	Generic meals	champagne vinegar; olive oil; Dijon mustard; s...	NaN

Le tableau affiche les 10 premiers produits que retourne l'API RapidAPI lorsqu'une requête avec le mot-clé « Champagne » lui est envoyée.

On peut déjà remarquer que la feature « foodContentsLabel » n'est pas tout à fait homogène dans sa nomenclature et que les images de description sont majoritairement manquantes pour ces produits.

⇨ Par conséquent, l'utilisation de cette API pour étoffer notre dataset et faire de la classification pourrait ne pas être possible si on se base uniquement sur les images de description. De plus, s'il s'avère que pour les autres produits les images sont fournies et que cette API est utilisable alors, il faudra d'abord veiller à bien homogénéiser la nomenclature de chaque valeur présente dans les features.

NB : Toutes les variables utilisées contenant les données récupérées ont été supprimées et seul le contenu du tableau que vous avez sous les yeux a été exporté au format csv pour être en conformité avec le règlement RGPD sur le traitement et le stockage de données externes.

V) Perspectives

Pour la suite, il serait pertinent d'essayer d'améliorer les résultats de classification du moteur en combinant l'approche NLP et l'approche visuelle.

D'autres algorithmes de NLP ou de type CNN existent (tel que MobilNetV2…) et pourraient être testés dans le but d'améliorer les performances du moteur.

D'autres jeux de paramètres, résultant de l'entraînement des algorithmes sur des datasets différents, pourraient aussi s'avérer plus adaptés que ceux utilisés dans ce projet.

Toute cette étude s'appuie sur le premier niveau de catégories (niveau 0), mais il en existe d'autres exploitables comprenant davantage de classes possibles. Cette classification plus fine permettrait éventuellement de classer les produits dans une sous-catégorie qui leur est plus spécifique tout en améliorant les performances du moteur ayant à disposition plus de caractéristiques discriminantes.