# CLASSIFICATION AUTOMATIQUE DE BIENS DE CONSOMMATION

## Contexte

Je suis Data Scientist au sein de l'entreprise "Place de marché", qui souhaite lancer une marketplace e-commerce.Les vendeurs proposent des articles à des acheteurs en postant une photo et une description. Pour l'instant, l'attribution de la catégorie d'un article est effectuée manuellement par les vendeurs et est donc peu fiable. De plus, le volume des articles est pour l'instant très petit. Pour rendre l'expérience plus confortable pour les utilisateurs(vendeurs et acheteurs) et dans l'optique d'un passage à l'échelle, il devient nécessaire d'automatiser cette tâche.

## Demande

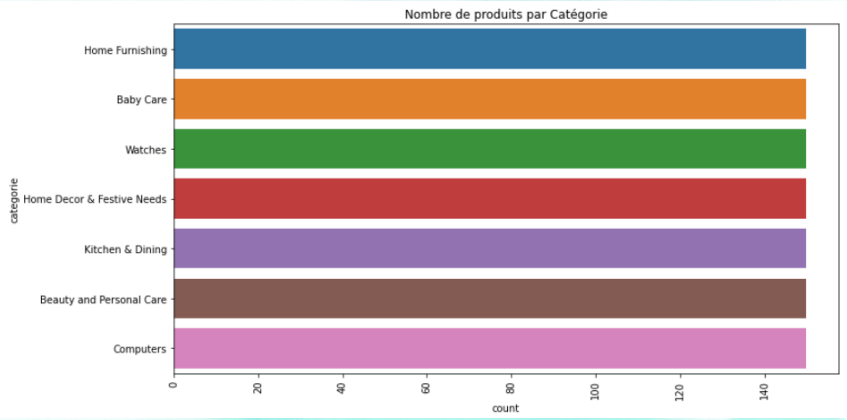
Il s'agit d'étudier la faisabilité d'un moteur de classification des articles basée sur un texte et une image pour l'automatisation de l'attribution de la catégorie de l'article.

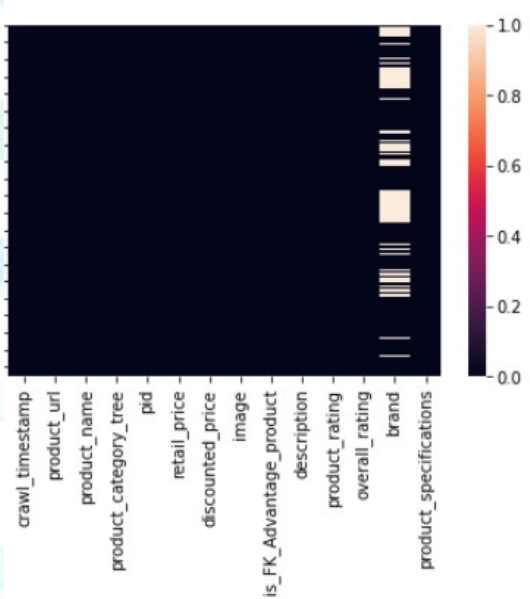
Il y a lieu d' analyser les descriptions textuelles et les images des produits au travers de :

- Un prétraitement des données texte ou image suivant le cas
- Une extraction de features
- Une réduction en 2 dimensions, afin de projeter les produits sur un graphique 2D
- Analyse du graphique afin d'en déduire ou pas, à l'aide des descriptions ou des images, la faisabilité de regrouper automatiquement des produits de même catégorie .
- Réalisation d'une mesure pour confirmer l' analyse visuelle, en calculant la similarité entre les catégories réelles et les catégories issues d'une segmentation en clusters.

# Analyse exploratoire des données

- pas de données manquantes pour les colonnes product_name ,image , description et product_category_name
- 1050 produits répartis uniformément en 7 catégories



## Création du corpus à partir de la variable description

```
# creation corpus
raw_corpus = ' '.join(df_['description'].values)
print("total nb of words in the whole corpus: ", len(raw_corpus.split()))
```

```
total nb of words in the whole corpus:  80013
```

## Exemple de quelques traitements appliqués au corpus

**The oldest lion was sleeping while the two youngers were playing !!**

- Mise en minuscule

**the oldest lion was sleeping while the two youngers were playing !!**

- Tokenisation +suppression ponctuation

**the oldest lion was sleeping while the two youngers were playing**

- Suppression des stop words

**oldest lion sleeping two youngers playing**

- Suppression des mots < 3 caractères

**oldest lion sleeping two youngers playing**

- Stemming(PorterStemmer)

**oldest lion sleep two younger play**

- Lemmatization(WordNetLemmatizer)

**oldest lion sleeping two younger playing**

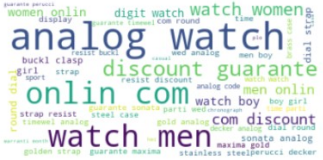
Autres traitements :

Suppression des mots rares : fréquence d'apparition <= 5
Suppression des tokens avec des caractères numériques
Suppression des  extra-words : mots communs aux 7 corpus parmi les 1000 mots les plus fréquents de chaque corpus

WordClouds des 7 catégories après nettoyage

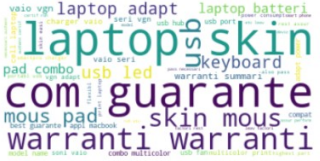

Home Furnishing


Baby Care




Watches


Home Decor & Festive Needs




Computers


Kitchen & Dining


Beauty and Personal Care

# Analyse supervisée du texte

- SVM linéaire pour comparer l'impact des méthodes de vectorisation sur la performance finale.
- normalisation des données d'entrée Pour le bow et le TF-IDF .
- pipeline avec grid_search et validation croisée (cv=5) avec meilleur modèle unigrammes/bigrammes
- métrique utilisée : accuracy.

1. **Bag of words** chaque document est représenté par un vecteur qui compte le nombre d'apparitions de chaque mot du vocabulaire dans le document.

2. **TF-IDF:** cette méthode de vectorisation tient compte de la fréquence relative des mots.On multiplie la fréquence d'apparition du mot dans le document par un terme qui pénalise une fréquence élevée du mot dans le corpus.

3. **Word2vec :** on représente les documents avec des vecteurs denses de dimension réduite. La performance chute fortement parce que la taille du corpus est très petite.

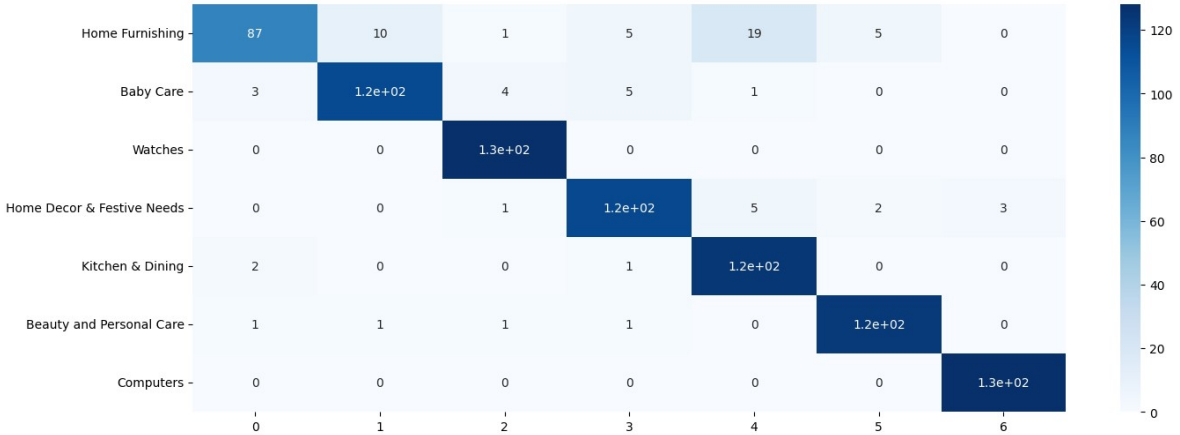
4. **Word2vec pré-entrainé :** on utilise un modèle pré-entrainé sur un très gros corpus pour faire du transfert learning. Le modèle utilisé est GLOVE (Global Vectors for Words Representation)

5. **Contextual embeddings :** la représentation vectorielle du mot n'est plus fixe mais est calculée dynamiquement en fonction des mots du voisinage ; ce qui permet de tenir compte du sens de la phrase.(BERT..)

# Résultats des modélisations de l'analyse supervisée

| | Mean accuracy |
|---|---|
| **Contextual Embedding** | 0.920357 |
| **TF-IDF** | 0.914782 |
| **Bag-of-Words** | 0.905838 |
| **Word2Vec pré-entraîné** | 0.884489 |
| **Word2Vec non pré-entraîné** | 0.450650 |

Matrice de confusion Contextual Embedding




Le modèle Avec Contextual Embedding obtient le meilleur score devant la vectorisation TF-IDF.
Ceci montre l'importance de tenir compte du contexte .

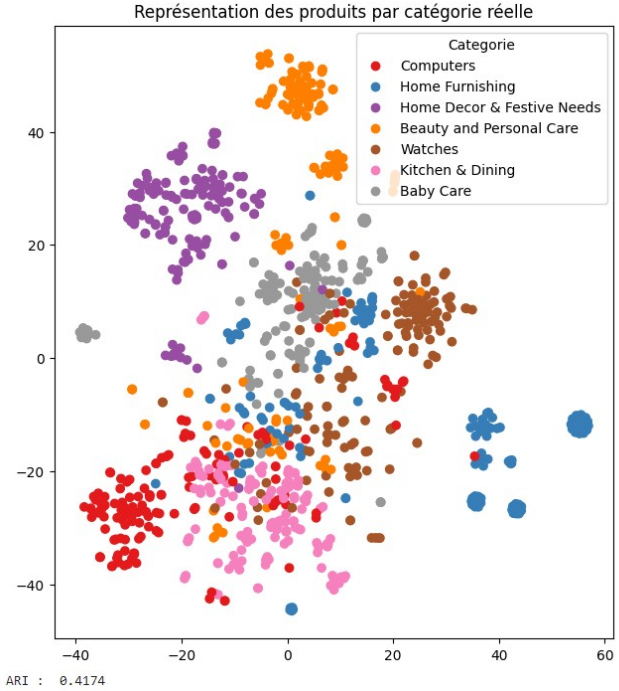
# Analyse non supervisée du texte

- création de description_bow_lem et description_dl
- kmeans pour déterminer les clusters après Tsne
- visualisation de Tsne selon les vraies catégories et selon clustérisation
- calcul de ARI entre les vraies catégories et les numéros de clusters.

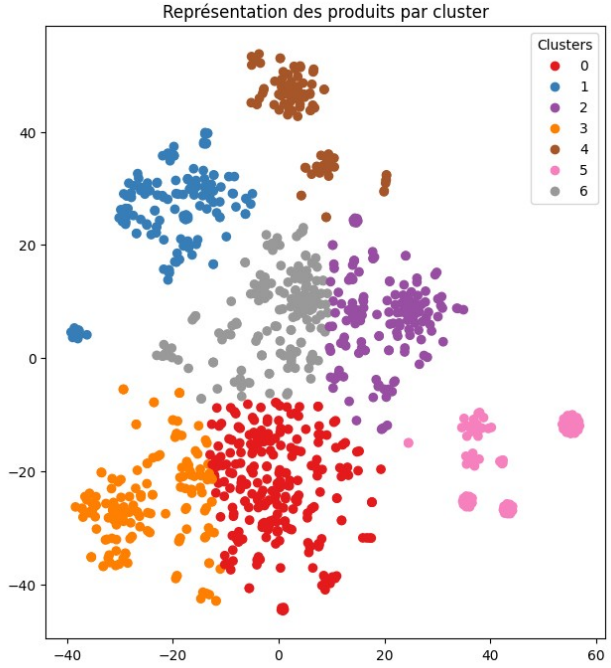
1. Bag of Words
2. TF-IDF
3. Word2Vec
4. BERT hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
5. BERT model_type = 'bert-base-uncased'
6. BERT model_type = 'cardiffnlp/twitter-roberta-base-sentiment'
7. BERT model_url = 'https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4'
8. USE – Universal Sentence Encoder

# Analyse non supervisée du texte

## T-SNE de Bag of Words

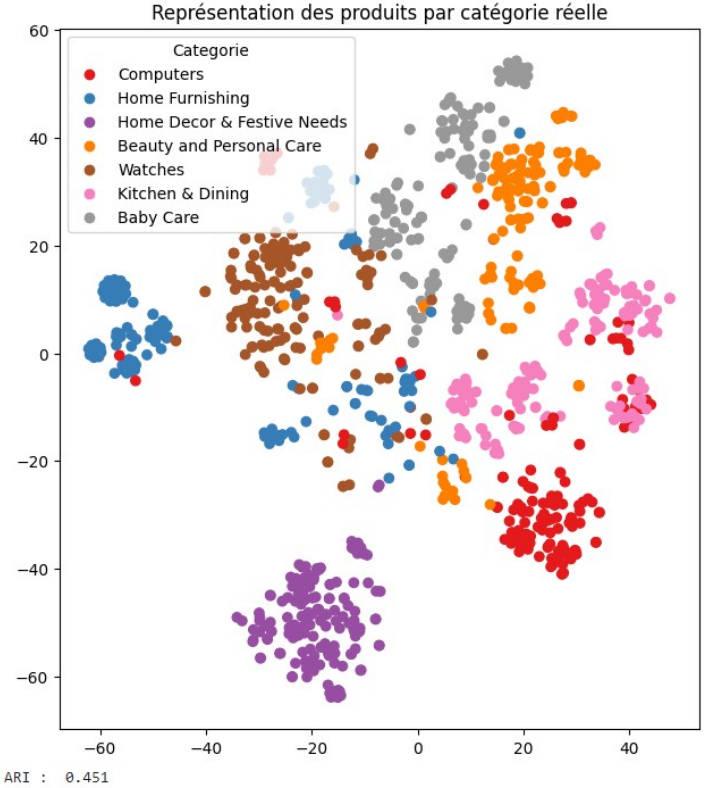



ARI : 0.4174

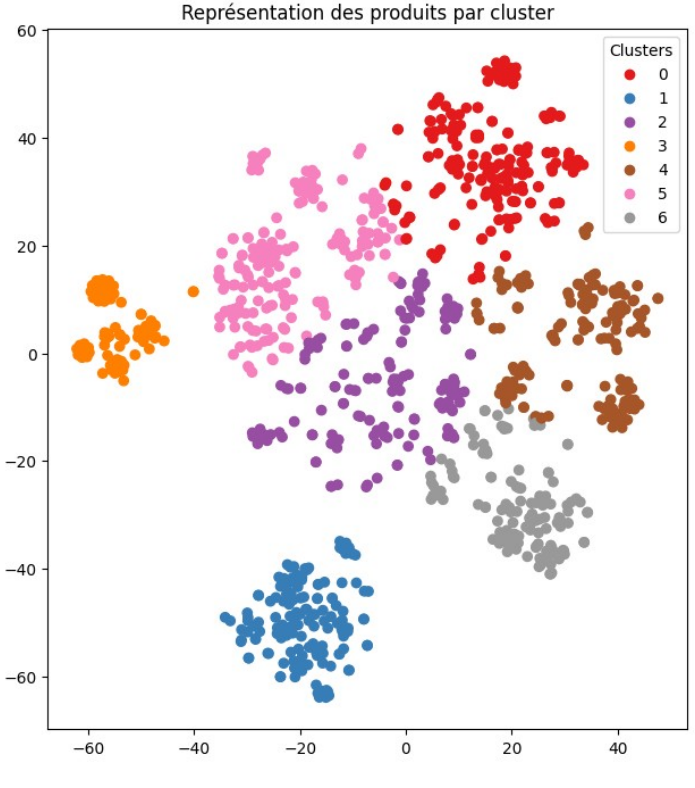


# Analyse non supervisée du texte

T-SNE de TD-IDF

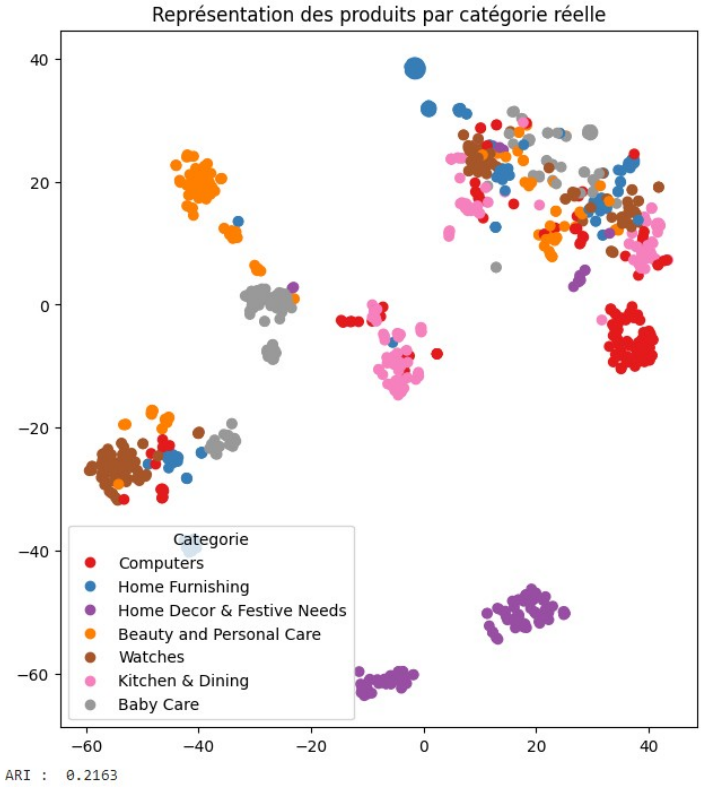


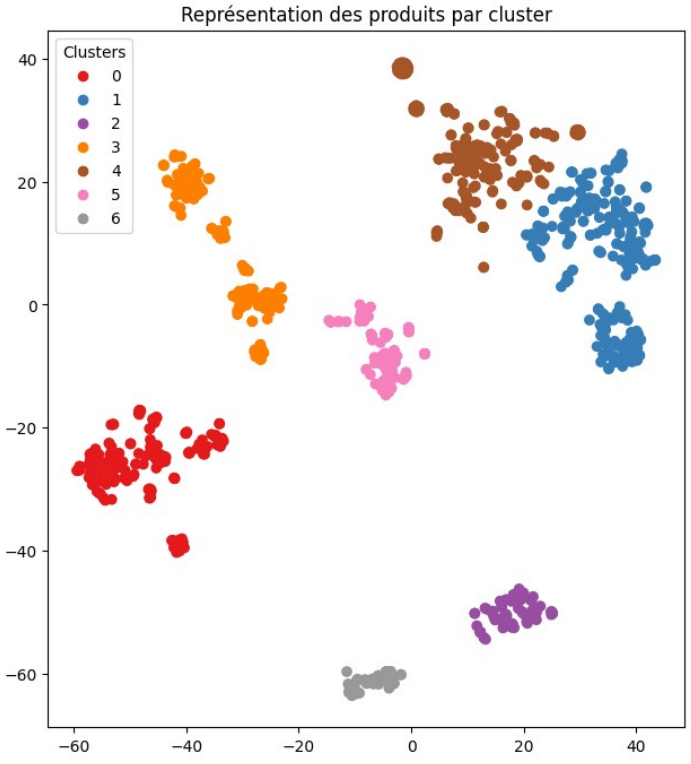

ARI : 0.451



# Analyse non supervisée du texte

T-SNE de Word2Vec

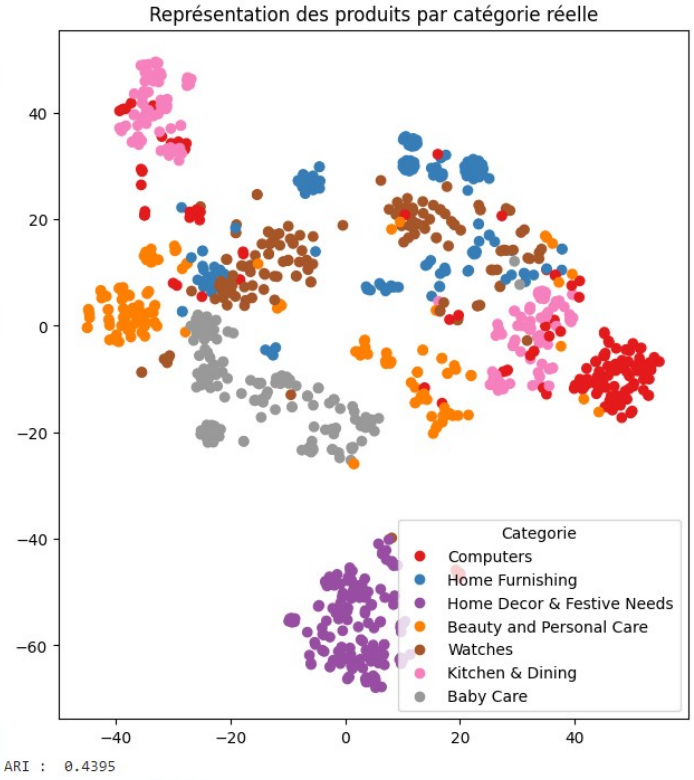


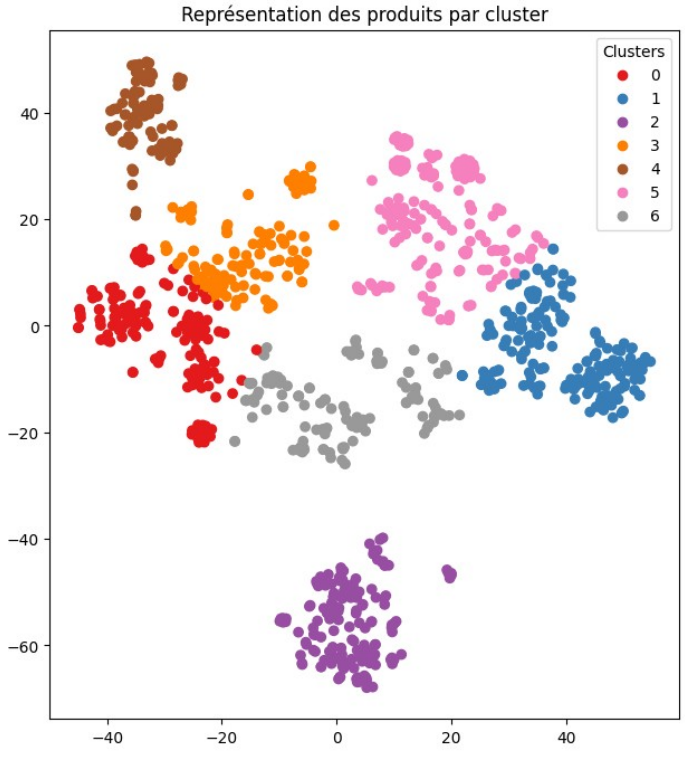
# Analyse non supervisée du texte

T-SNE de BERT

```
embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
```



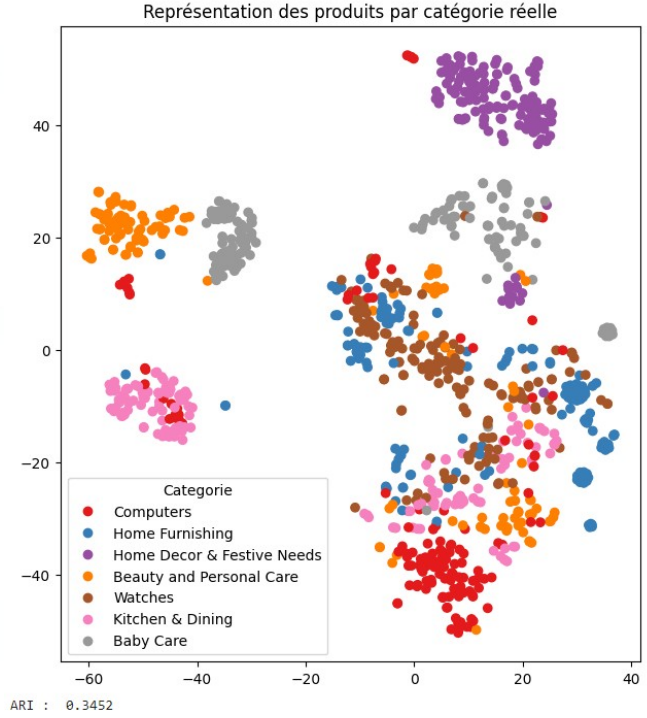

ARI : 0.4395

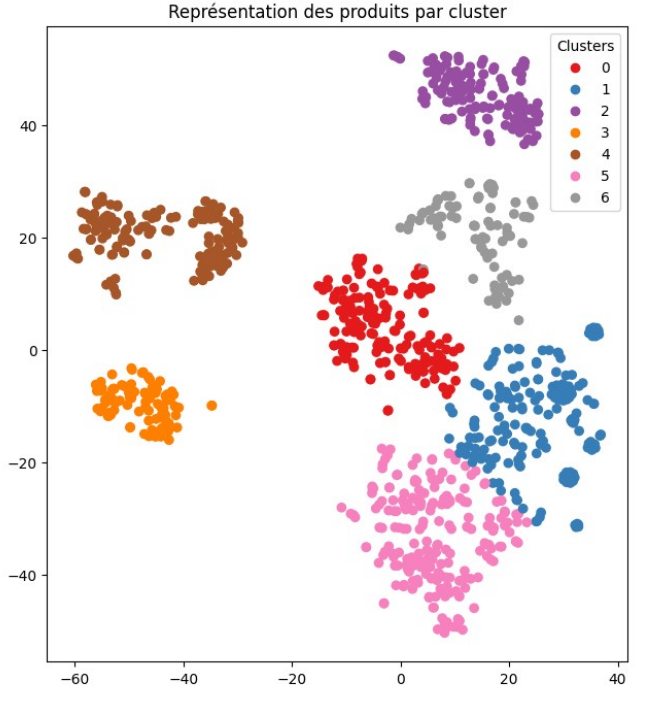


# Analyse non supervisée du texte

T-SNE de BERT HuggingFace

```
model_type = 'bert-base-uncased'
```



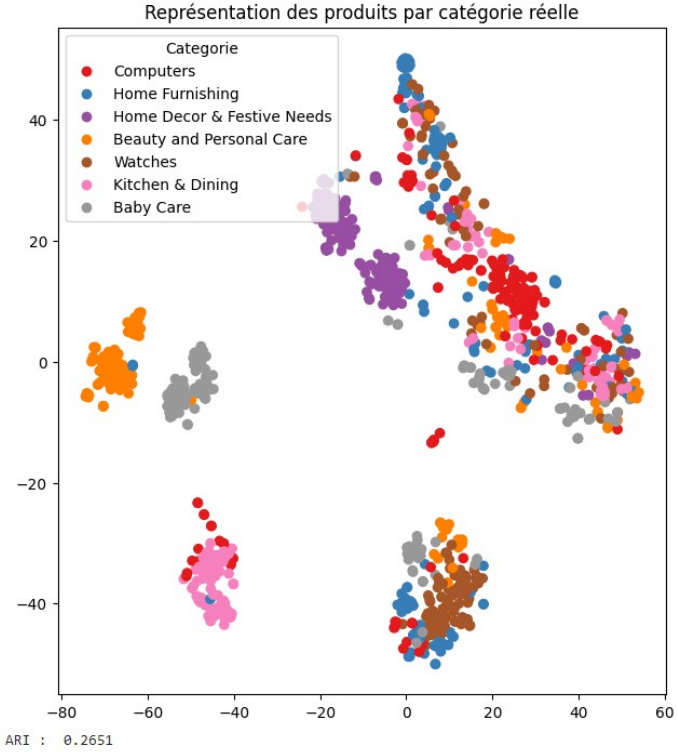




ARI : 0.3452

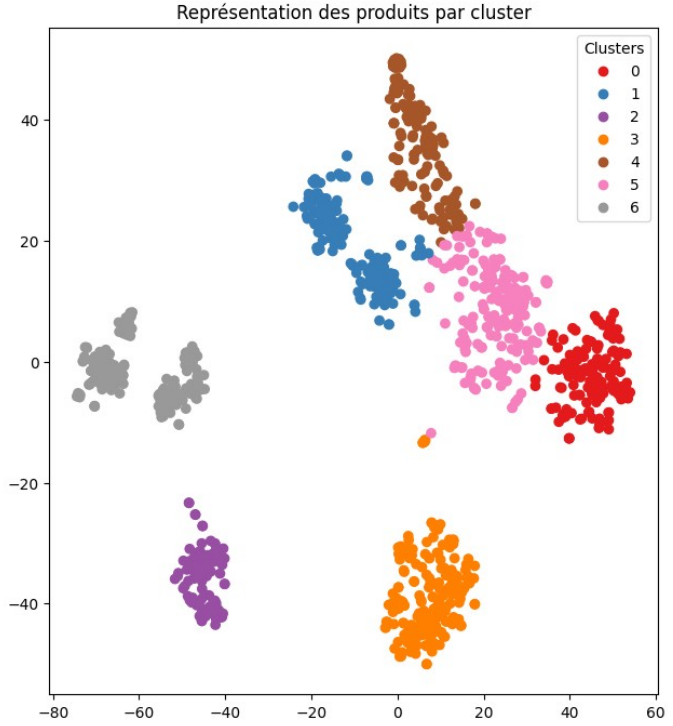
# Analyse non supervisée du texte

T-SNE de BERT HuggingFace

```
model_type = 'cardiffnlp/twitter-roberta-base-sentiment'
```



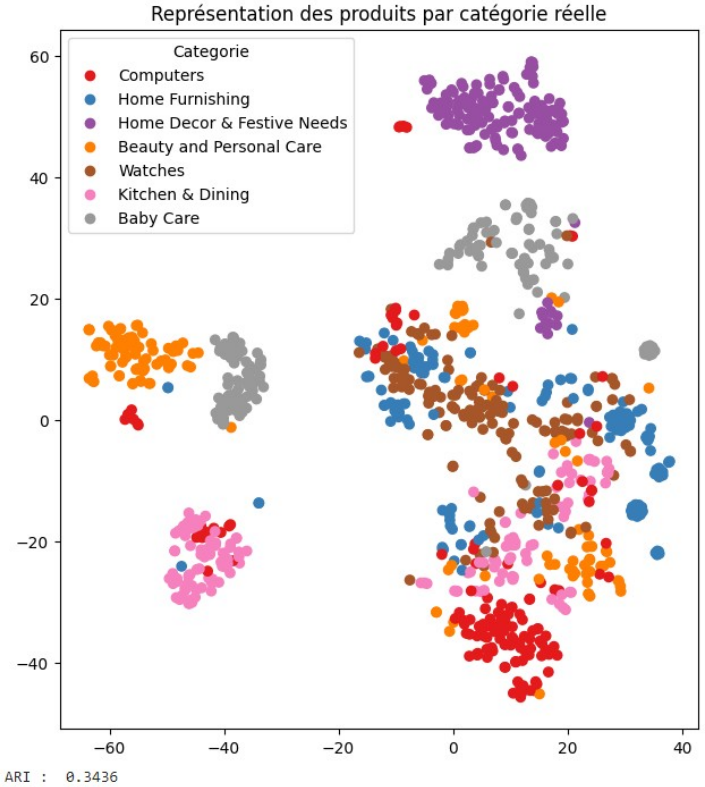

ARI : 0.2651



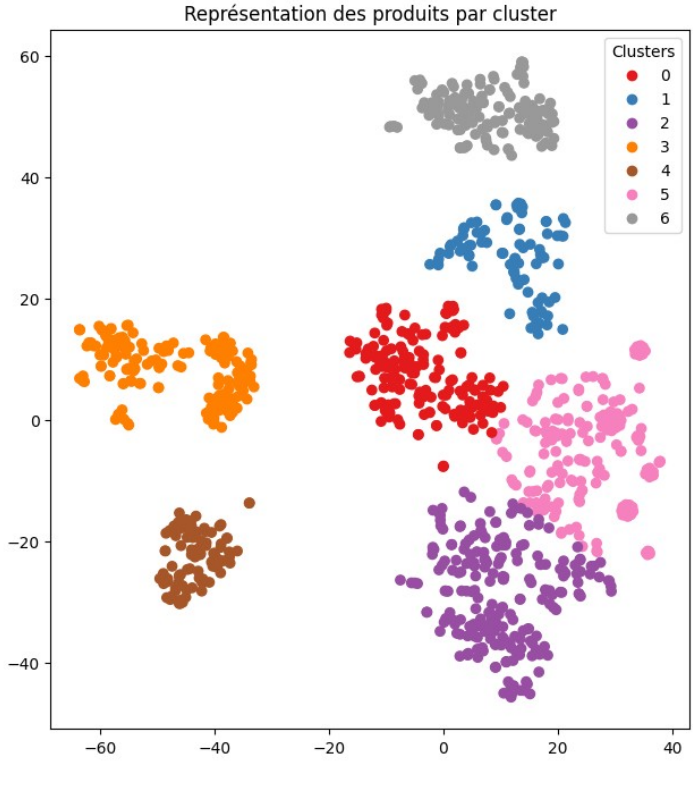
# Analyse non supervisée du texte

T-SNE de BERT hub Tensorflow

```
model_url = 'https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4'
```

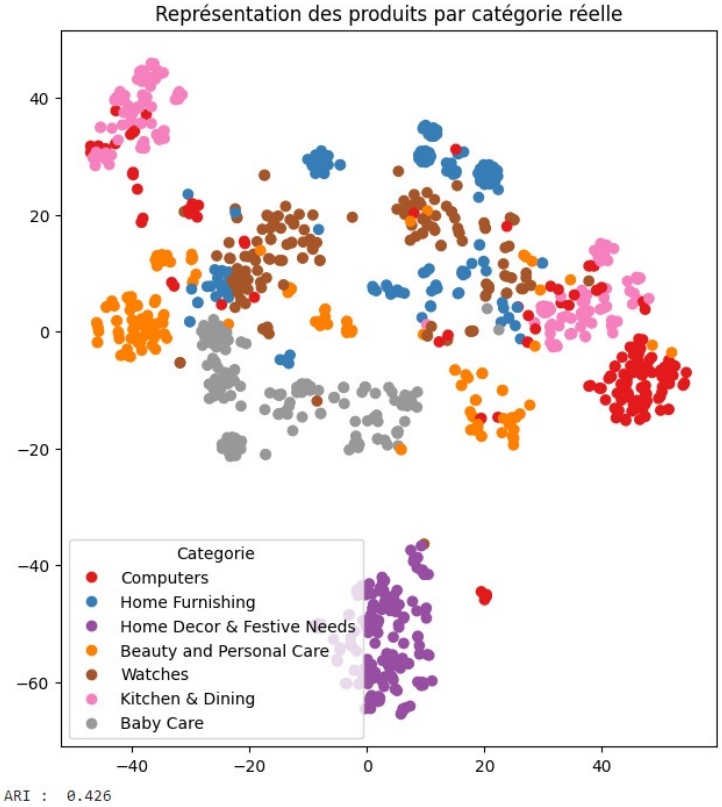


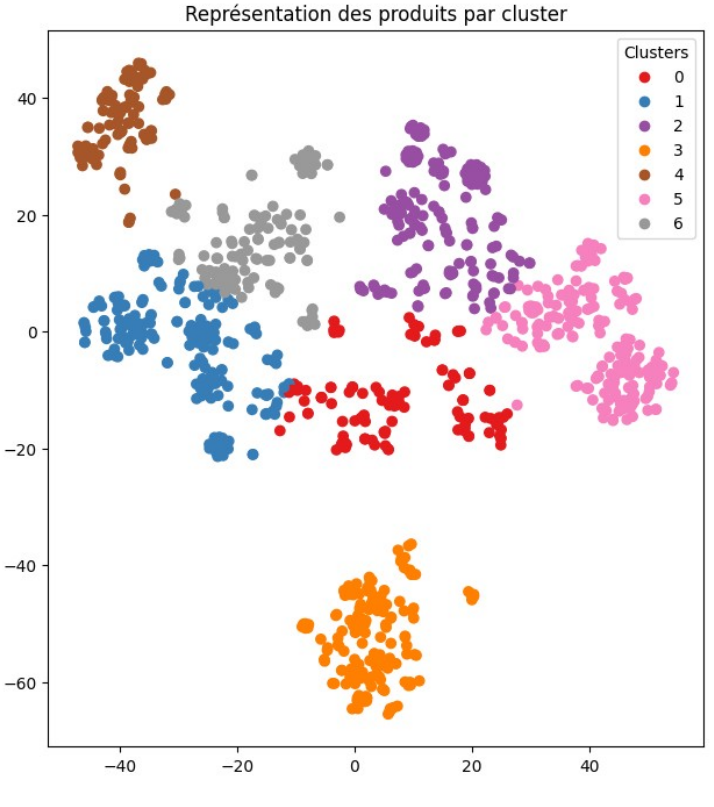

ARI : 0.3436



# Analyse non supervisée du texte

## T-SNE de USE - Universal Sentence Encoder



# Analyse non supervisée des images

Création du modèle pré-entraîné : VGG16()
Création des features des images
Réduction dimension et analyse

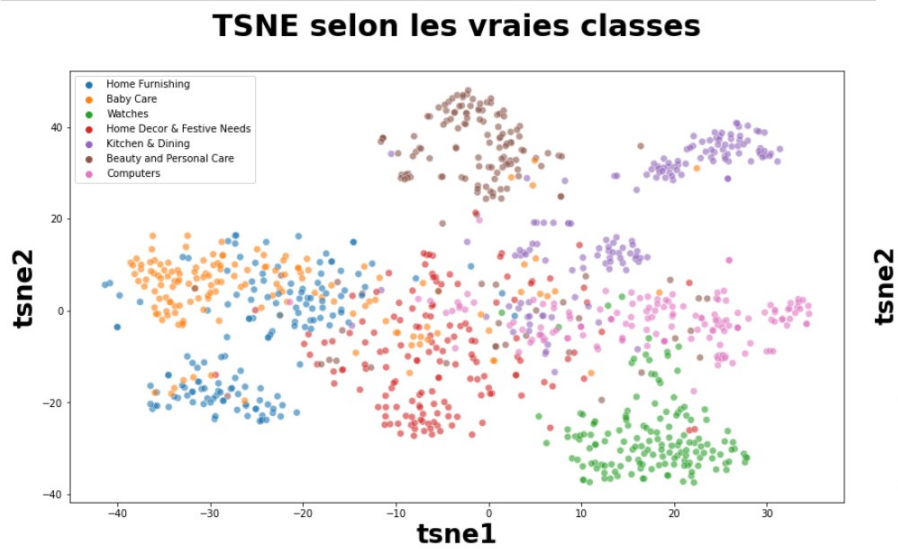
```
base_model = VGG16()
model = Model(inputs=base_model.inputs, outputs=base_model.layers[-2].output)
```

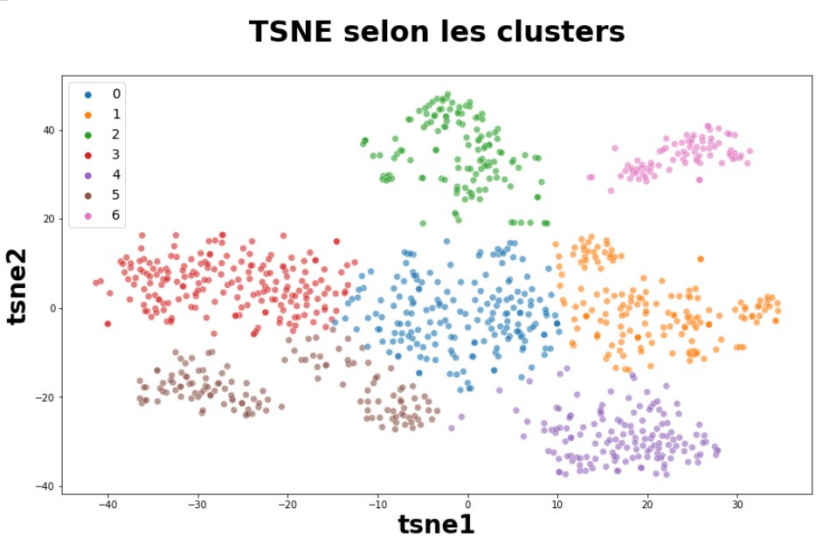
```
Nombre de descripteurs VGG : (1050, 4096)
```

```
pca = decomposition.PCA(n_components=0.99)
feat_pca= pca.fit_transform(images_features)
print(feat_pca.shape)
```

```
(1050, 803)
```

Réduction de dimension T-SNE et affichage des images selon vraie classe et selon clustering kmean



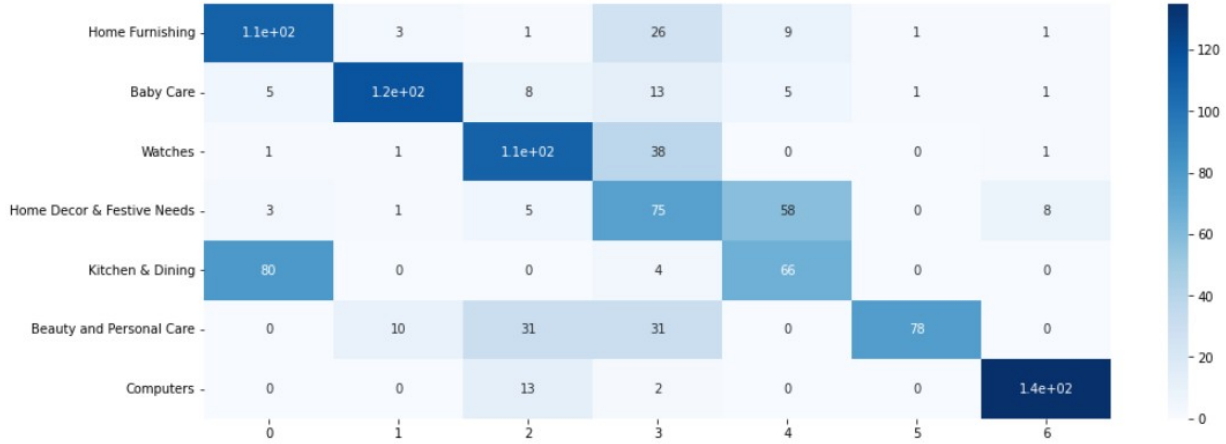




```
ARI :  0.4482725638250802
```

L'analyse graphique montre visuellement qu'**on peut séparer automatiquement les images selon leurs vraies classes**.

Ceci suffit à démontrer la **faisabilité de réaliser ultérieurement une classification supervisée** pour déterminer automatiquement les classes des images.

# Analyse des classes : Matrice de confusion




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.73 | 0.63 | 150 |
| 1 | 0.89 | 0.78 | 0.83 | 150 |
| 2 | 0.65 | 0.73 | 0.69 | 150 |
| 3 | 0.40 | 0.50 | 0.44 | 150 |
| 4 | 0.48 | 0.44 | 0.46 | 150 |
| 5 | 0.97 | 0.52 | 0.68 | 150 |
| 6 | 0.92 | 0.90 | 0.91 | 150 |
| accuracy | | | 0.66 | 1050 |
| macro avg | 0.69 | 0.66 | 0.66 | 1050 |
| weighted avg | 0.69 | 0.66 | 0.66 | 1050 |

# Détermination et affichage des descripteurs SIFT

un exemple avec cv2

```
sift = cv2.SIFT_create(500)
image = cv2.imread(list_photos[50])

plt.figure(figsize=(18, 14))
```

```
plt.subplot(2,3,1)
plt.axis('off')
plt.title('image depart')
plt.imshow(image)

plt.subplot(2,3,2)
#retirer le bruit
image = cv2.GaussianBlur(image, (5, 5), 0)
plt.axis('off')
plt.title('debruite')
plt.imshow(image)

#niveaux de gris
plt.subplot(2,3,3)
gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.axis('off')
plt.title('noir et blanc')
plt.imshow(gray_img,cmap='gray')

#augmentation du contraste par égalisation
plt.subplot(2,3,4)
image = cv2.equalizeHist(gray_img)   # equalize image histogram
plt.axis('off')
plt.title('rajout contraste')
plt.imshow(image, cmap='gray')

#normaliser en redimensionnant image
plt.subplot(2,3,5)
dsize = (224, 224)
image = cv2.resize(image, dsize)
plt.axis('off')
plt.title('redimensionnement')
plt.imshow(image, cmap='gray')

#detection feature SIFT
plt.subplot(2,3,6)
kp, des = sift.detectAndCompute(image, None)
img=cv2.drawKeypoints(image,kp,gray_img)
plt.axis('off')
plt.title('features SIFT')
plt.imshow(img)
```


```
Descripteurs :  (211, 128)

[[ 22.   0.   0. ...   0.   0.   0.]
 [ 34.   1.   0. ...   0.   0.   0.]
 [ 19.   0.   0. ...   0.   0.   1.]
 ...
 [120.  16.   0. ...   0.   0.   0.]
 [ 59.  20.   0. ...   0.   0.   7.]
 [ 25.   0.   0. ...   0.   0.   1.]]
```

# Pré-traitement des images via SIFT

## 1. Création des descripteurs

- Pour chaque image
- débruitage
- passage en gris
- equalisation
- redimensionnement de l'image
- création d'une liste de descripteurs par image ("sift_keypoints_by_img") qui sera utilisée pour réaliser les histogrammes par image
- création d'une liste de descripteurs pour l'ensemble des images ("sift_keypoints_all") qui sera utilisé pour créer les clusters de descripteurs

```
Nombre de descripteurs :  (347152, 128)
temps de traitement SIFT descriptor :            38.22 secondes
```

## 2. Création des clusters de descripteurs

```
Nombre de clusters estimés :  589
Création de 589 clusters de descripteurs ...
temps de traitement kmeans :             3.23 secondes
```

## 3. Création des features des images

Pour chaque image :

- prédiction des numéros de cluster de chaque descripteur
- création d'un histogramme = comptage pour chaque numéro de cluster du nombre de descripteurs de l'image

## 4. Réduction de dimension

### 4.1 Réduction de dimension PCA

- La réduction PCA permet de créer des features décorrélées entre elles, et de diminuer leur dimension, tout en gardant un niveau de variance expliquée élevé (99%)
- L'impact est une meilleure séparation des données via le T-SNE et une réduction du temps de traitement du T-SNE

```
Dimensions dataset avant réduction PCA :  (1050, 589)
Dimensions dataset après réduction PCA :  (1050, 475)
```

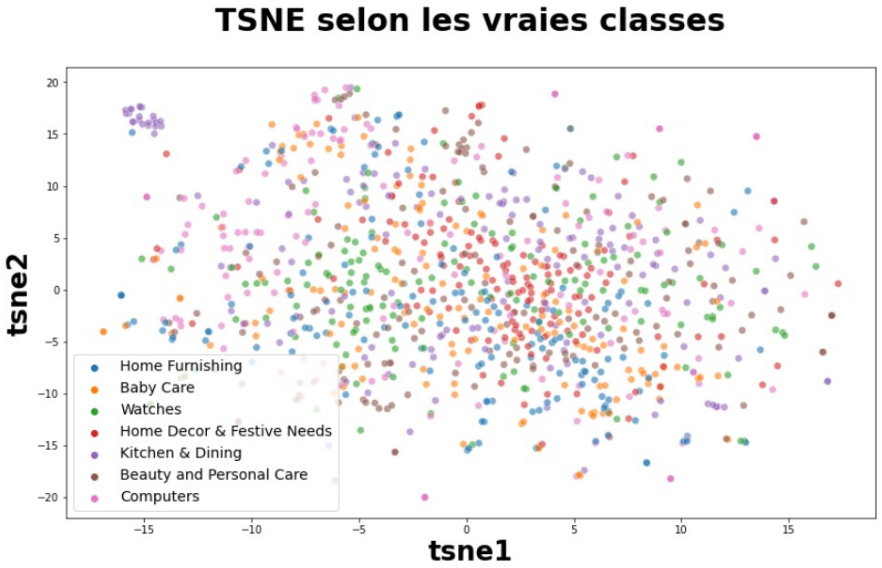
### 4.2 Réduction de dimension T-SNE

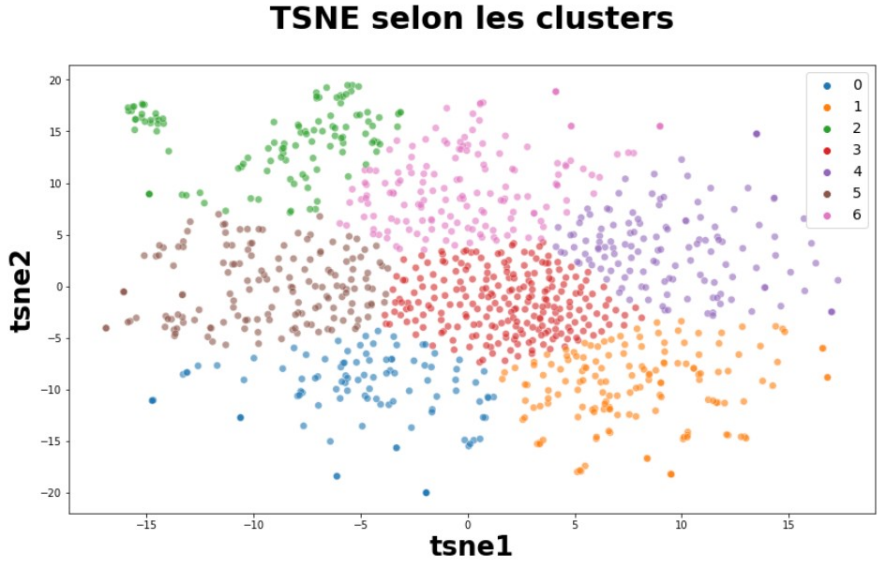
- Réduction de dimension en 2 composantes T-SNE pour affichage en 2D des images

## 5.Affichage T-SNE selon catégories d'images

## 6.Affichage T-SNE selon clusters et calcul ARI de similarité catégories images / clusters

# Pré-traitement des images via SIFT… le résultat




ARI : 0.03428293217186819

Il y a de très grosses différences entre vraies classes et clusters.Le faible résultat provient du jeu de données trop restreint.

# Analyse supervisée des images

5 approches sont présentées :

Une approche simple par préparation initiale de l'ensemble des images avant classification supervisée
Une approche par data generator sans data augmentation
Une approche par data generator avec data augmentation. Les images sont directement récupérées à la volée dans le répertoire des images
Une approche par DataSet(Tensorflow.org) sans data augmentation
Une approche par DataSet, avec data augmentation intégrée au modèle

## Création des 2 modèles de classification

Pour les 4eres approches

```
def create_model_fct() :
    # Récupération modèle pré-entraîné
    model0 = VGG16(include_top=False, weights="imagenet", input_shape=(224, 224, 3))

    # Layer non entraînables = on garde les poids du modèle pré-entraîné
    for layer in model0.layers:
        layer.trainable = False

    # Récupérer la sortie de ce réseau
    x = model0.output
    # Compléter le modèle
    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(7, activation='softmax')(x)

    # Définir le nouveau modèle
    model = Model(inputs=model0.input, outputs=predictions)
    # compilation du modèle
    model.compile(loss="categorical_crossentropy", optimizer='rmsprop', metrics=["accuracy"])

    print(model.summary())

    return model
```

Pour la 5eme approche

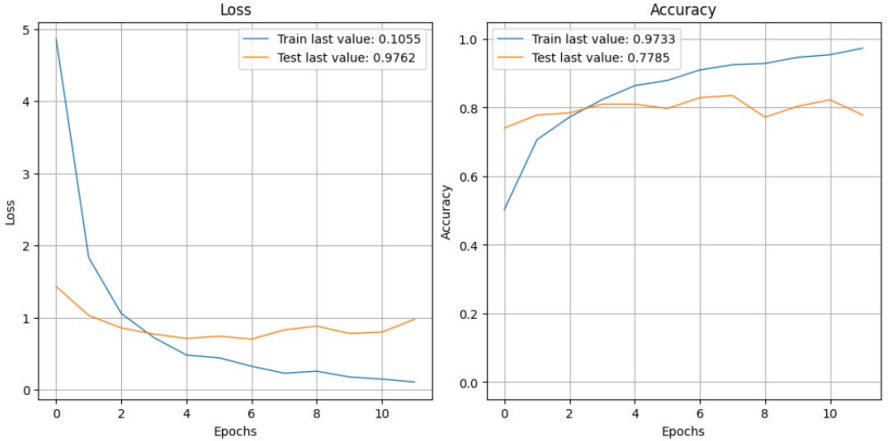
```
def create_model_fct2() :
    # Data augmentation
    data_augmentation = Sequential([
        RandomFlip("vertical", input_shape=(224, 224, 3)),
        RandomRotation(0.2),
        RandomZoom(0.2),

      ])

    # Récupération modèle pré-entraîné
    model_base = VGG16(include_top=False, weights="imagenet", input_shape=(224, 224, 3))
    for layer in model_base.layers:
        layer.trainable = False

    # Définition du nouveau modèle
    model = Sequential([
                data_augmentation,
                Rescaling(1./127.5, offset=-1),
                model_base,
                GlobalAveragePooling2D(),
                Dense(256, activation='relu'),
                Dropout(0.5),
                Dense(7, activation='softmax')
                ])

    # compilation du modèle
    model.compile(loss="categorical_crossentropy", optimizer='adam', metrics=["accuracy"])
    print(model.summary())

    return model
```

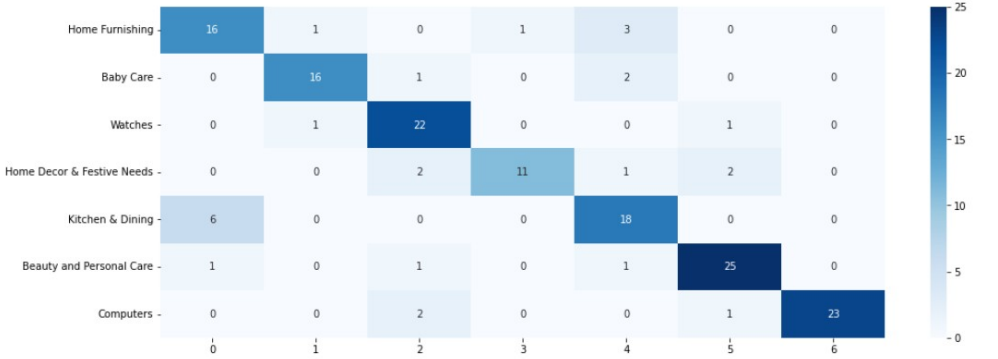
# Analyse supervisée des images : Modèle avec préparation initiale des images

```
# Score de l'epoch optimal

model1.load_weights(model1_save_path1)

loss, accuracy = model1.evaluate(X_val, y_val, verbose=False)
print("Validation Accuracy :  {:.4f}".format(accuracy))

loss, accuracy = model1.evaluate(X_test, y_test, verbose=False)
print("Test Accuracy       :  {:.4f}".format(accuracy))

```

```
Validation Accuracy :  0.8291
Test Accuracy       :  0.7905
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.70 | 0.76 | 0.73 | 21 |
| 1 | 0.89 | 0.84 | 0.86 | 19 |
| 2 | 0.79 | 0.92 | 0.85 | 24 |
| 3 | 0.92 | 0.69 | 0.79 | 16 |
| 4 | 0.72 | 0.75 | 0.73 | 24 |
| 5 | 0.86 | 0.89 | 0.88 | 28 |
| 6 | 1.00 | 0.88 | 0.94 | 26 |
| accuracy | | | 0.83 | 158 |
| macro avg | 0.84 | 0.82 | 0.82 | 158 |
| weighted avg | 0.84 | 0.83 | 0.83 | 158 |

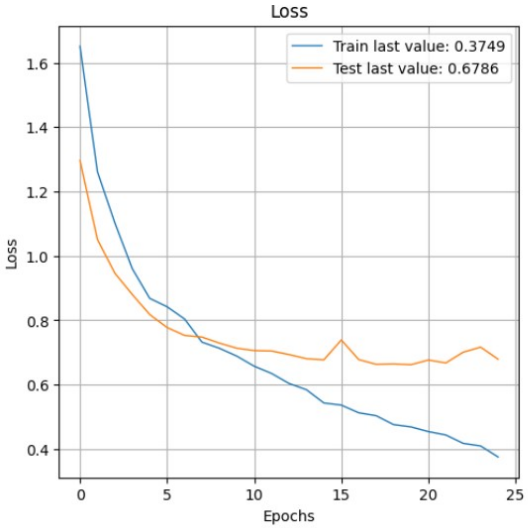


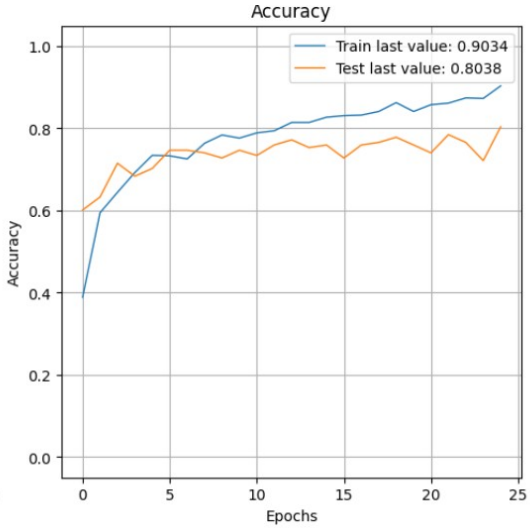



# Analyse supervisée des images : Modele avec ImageDataGenérator sans Data Augmentation

```
# Score de l'epoch optimal

model2.load_weights(model2_save_path)

loss, accuracy = model2.evaluate(validation_generator, verbose=False)
print("Validation Accuracy :  {:.4f}".format(accuracy))

loss, accuracy = model2.evaluate(test_generator, verbose=False)
print("Test Accuracy       :  {:.4f}".format(accuracy))

```

```
Validation Accuracy :  0.7595
Test Accuracy       :  0.8952
```




Sur les **données d'entraînement**, le modèle est **de plus en plus performant** mais sur les **données de validation** il stagne, voir devient **de moins en moins bon**.

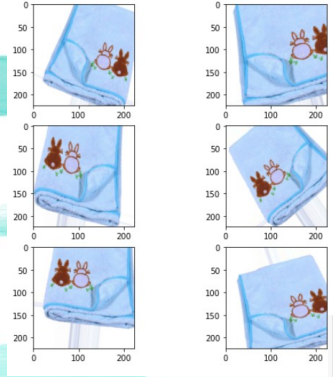
Le modèle **se spécialise sur les données d'entraînement** mais n'est plus capable d'**interpréter** les données de validation (et les données réelles en général), il est en **overfitting**. Sur les données de test,la précision est de 0.89

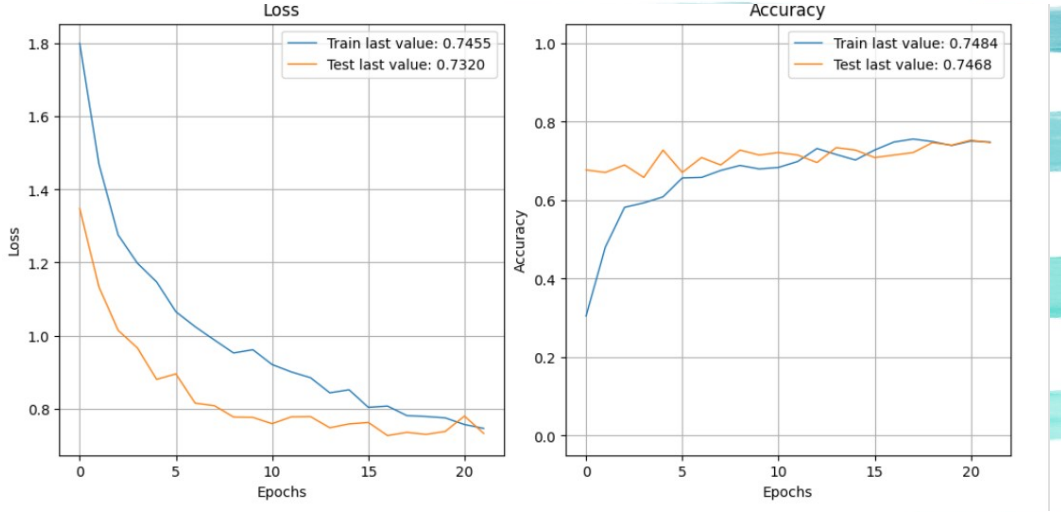
L'overfitting est dû au manque de données pour l' entraînement du modèle. Pour augmenter le nombre de données , on va créer de nouvelles images grâce à la Data Augmentation. L'idée est de reproduire les données pré-existantes en leur appliquant une transformation aléatoire. Lors de l'entraînement, le modèle aura plus de données à sa disposition.Ce qui devrait lui permettre de mieux généraliser.

# Analyse supervisée des images : Modèle avec ImageDataGenerator avec Data Augmentation

On va utiliser la fonction ImageDataGenerator en modifiant certains paramètres pour créer de nouvelles images.

```python
# exemple de filtres pour générer nouvelles images à partir d'images existantes
augmented_datagen = ImageDataGenerator(
        rotation_range=45,
        width_shift_range=0.25,
        height_shift_range=0.25,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')    # pour affichage
```






La Data Augmentation concerne seulement les données d'entraînement... pour les autres données on initialise un générateur simple.

```python
train_generator = augmented_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

validation_generator = datagen.flow_from_directory(
    validation_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical')
```

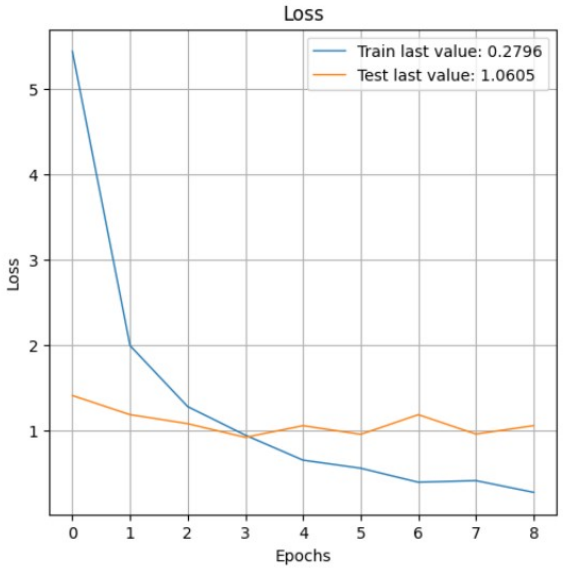
```python
# Score de l'epoch optimal

model3.load_weights(model3_save_path)

loss, accuracy = model3.evaluate(validation_generator, verbose=False)
print("Validation Accuracy :  {:.4f}".format(accuracy))

loss, accuracy = model3.evaluate(test_generator, verbose=True)
print("Test Accuracy       :  {:.4f}".format(accuracy))
```

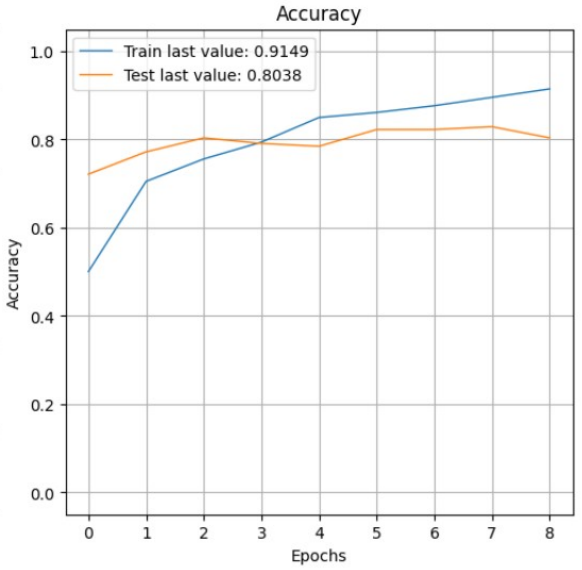
```
Validation Accuracy :  0.7152
4/4 [==============================] - 21s 5s/step - loss: 0.4902 - accuracy: 0.8571
Test Accuracy       :  0.8571
```

# Analyse supervisée des images : Modèle avec Dataset sans data augmentation

```
# Score de l'epoch optimal

model4.load_weights(model4_save_path)

loss, accuracy = model4.evaluate(dataset_val, verbose=False)
print("Validation Accuracy :  {:.4f}".format(accuracy))

loss, accuracy = model4.evaluate(dataset_test, verbose=False)
print("Test Accuracy       :  {:.4f}".format(accuracy))

```

```
Validation Accuracy :  0.7911
Test Accuracy       :  0.8667
```

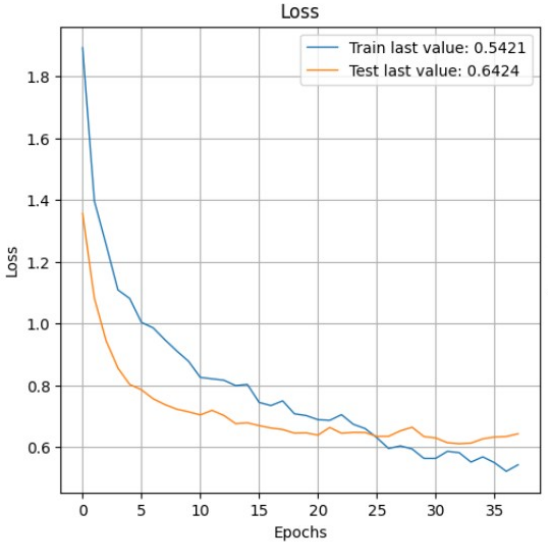


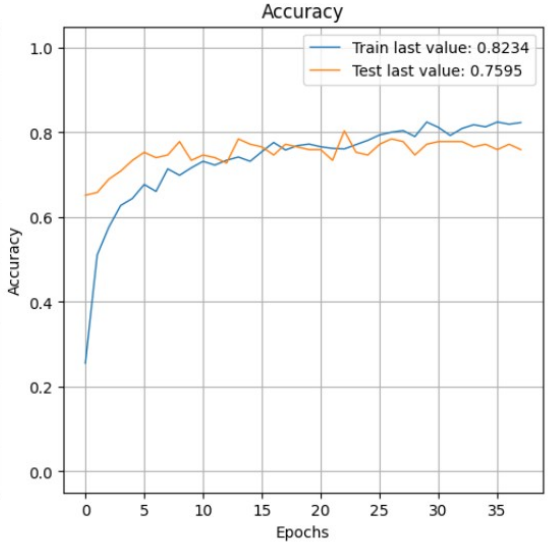



# Analyse supervisée des images : Modèle avec Dataset et Data Augmentation intégrée au modèle

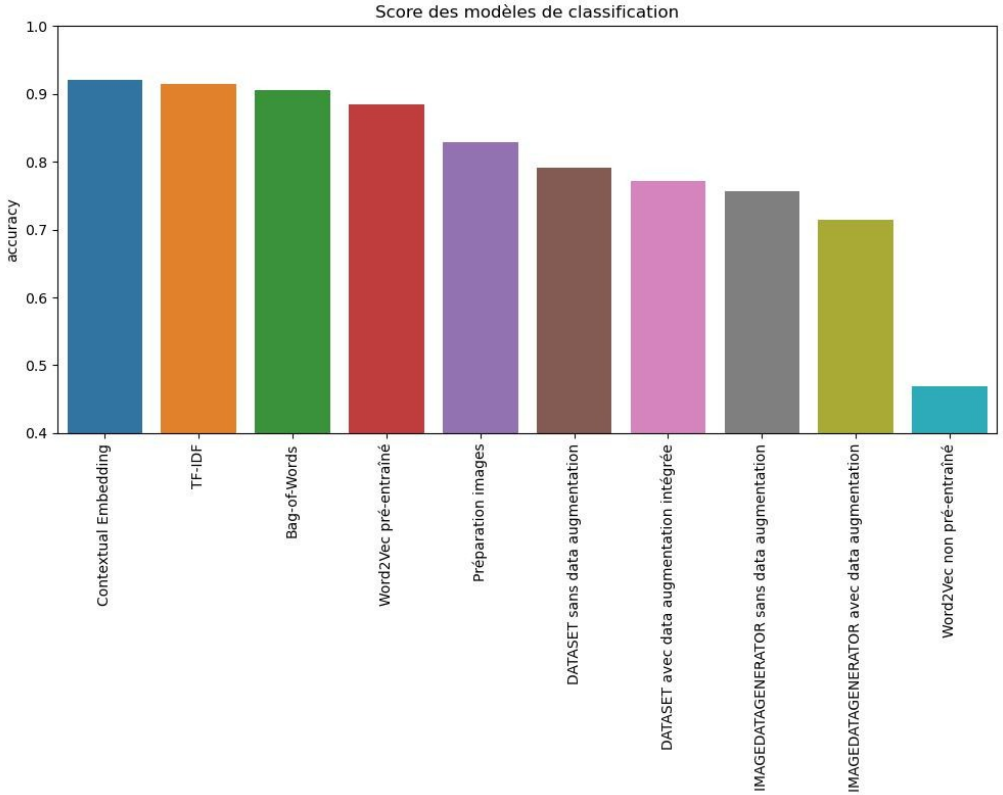
```python
# Score de l'epoch optimal

model15.load_weights(model15_save_path)

loss, accuracy = model15.evaluate(dataset_val, verbose=False)
print("Validation Accuracy :  {:.4f}".format(accuracy))

loss, accuracy = model15.evaluate(dataset_test, verbose=False)
print("Test Accuracy       :  {:.4f}".format(accuracy))

```

```
Validation Accuracy :  0.7785
Test Accuracy       :  0.8857
```



# Comparatifs score des différents modèles de vectorisation



# Test d’une API

```
import requests
import pandas as pd
import json
url = "https://edamam-food-and-grocery-database.p.rapidapi.com/api/food-database/v2/parser"

querystring = {"ingr":"champagne"}

headers = {"X-RapidAPI-Key": "eea546e3c0msh22a68ee5e0e8e57p1d5922jsne487409fe309",
"X-RapidAPI-Host": "edamam-food-and-grocery-database.p.rapidapi.com"}

response = requests.get(url, headers=headers, params=querystring)
dict = response.json()

champs = ["foodId","label","category","foodContentsLabel","image"]

def get_food(data):
    food = data['food']
    return [ food.get(f) for f in champs ]

def get_foods(foods):
    return pd.DataFrame(
        [get_food(parsed)
        for parsed in foods ],
        columns=champs)

all_foods = None
foods = get_foods(dict.get('hints') or [])
if all_foods is None:
    all_foods = foods
else:
    all_foods = pd.concat([all_foods, foods])
all_foods=all_foods.head(10)
#on conserve uniquement les 10 premiers
all_foods.to_csv('api_champagne.csv',index=False)
```

Le fichier csv

```
df=pd.read_csv(r'C:\Users\serge\OneDrive\Documents\Documents\DATA SCIENTIST\P6\api_champagne.csv')
df
```

| | 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 |

un rappel des 5 grands principes des règles de protection des données personnelles

Les 5 grands principes des règles de protection des données personnelles sont les suivants :

- **Le 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 ;
- **Le principe de proportionnalité et de pertinence** : les informations enregistrées doivent être pertinentes et strictement nécessaires au regard de la finalité du fichier ;
- **Le 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 ;
- **Le 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 ;
- **Les droits des personnes.**