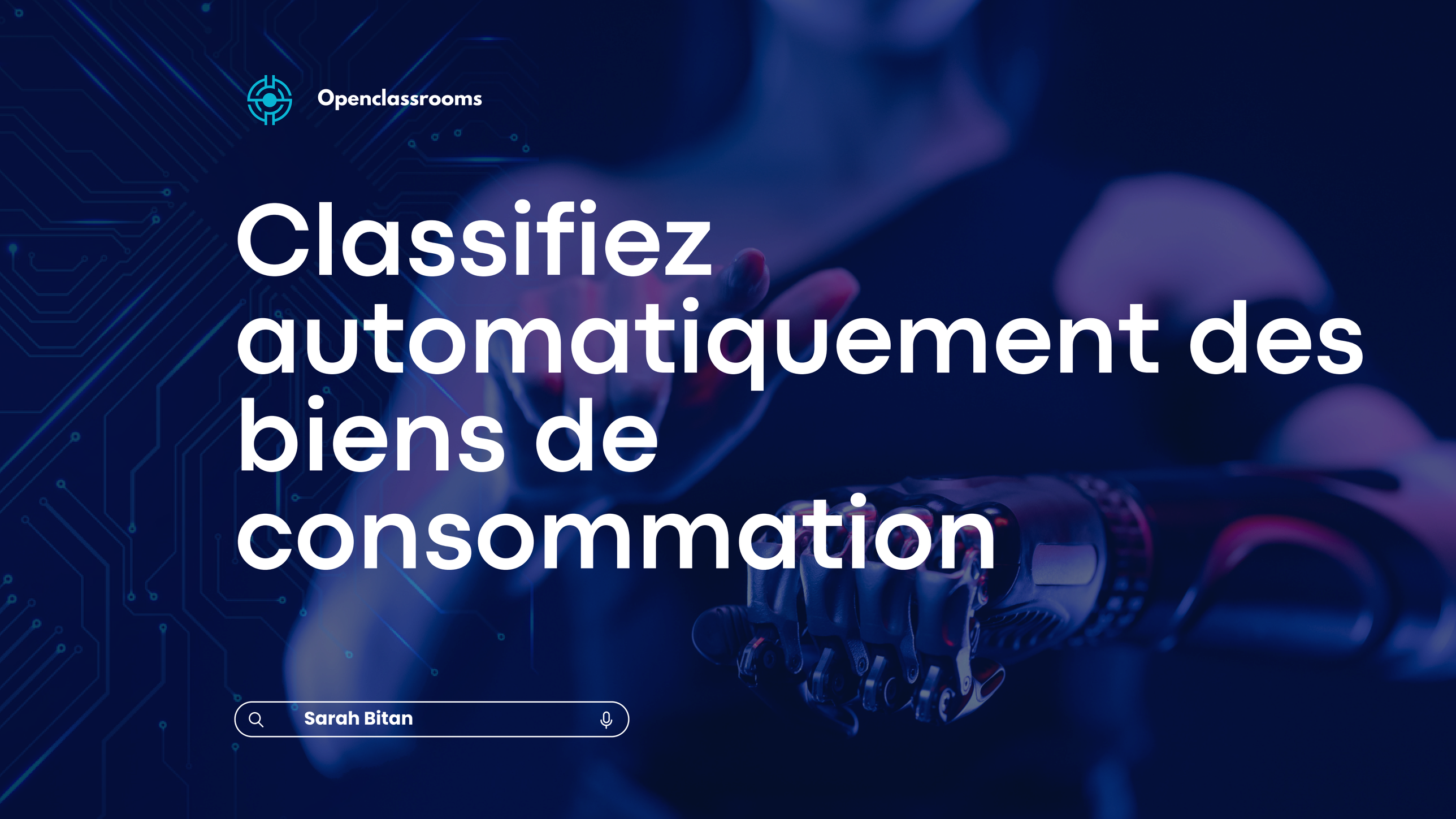
Openclassrooms

Classifiez automatiquement des biens de consommation



Sarah Bitan

Notre mission

En tant que Data Scientist chez "Place de Marché", ma mission est d'automatiser la classification des articles en utilisant leurs descriptions textuelles et images pour améliorer l'expérience utilisateur.

Actuellement, cette tâche est réalisée manuellement par les vendeurs, ce qui est peu fiable et difficile à gérer à grande échelle.

Problématique

Quelles sont les attentes ?

	Acteurs			
	Avant		Apres	
Missions	Vendeur	Place de marche	Vendeur	Place de marche
Rediger la description	✓		✓	
Poster la photo	✓		✓	
Choisir une categorie	✓			✓

Plan

1.Présentation de nos données

2.Traitement de texte (bag of words, TF-IDF, Word2Vec, GloVe, BERT et USE)

3.Traitement d’images (algo SIFT, Transfer Learning avec des CNN)

4.Classification supervisée d’images via CNN Transfer Learning

5.API

Présentation de nos données

Type de nos données

```
df.dtypes
```

```
uniq_id                    object
crawl_timestamp            object
product_url                object
product_name               object
product_category_tree      object
pid                        object
retail_price              float64
discounted_price          float64
image                      object
is_FK_Advantage_product      bool
description                object
product_rating             object
overall_rating             object
brand                      object
product_specifications     object
dtype: object
```

Nettoyage

```
# Vérifier s'il y a des doublons dans le DataFrame
if df.duplicated().any():
    print("Il y a des doublons dans le jeu de données.")
else:
    print("Il n'y a pas de doublons dans le jeu de données.")
```

```
Il n'y a pas de doublons dans le jeu de données.
```

```
# Compter le nombre de valeurs NaN dans chaque colonne
nan_count = df.isna().sum()

print("Nombre de valeurs NaN dans le jeu de données :")
print(nan_count)

# Supprimer les lignes contenant des valeurs NaN
df_cleaned = df.dropna()

# Afficher la forme du DataFrame après suppression des lignes contenant des NaN
print("\nForme du DataFrame après suppression des lignes contenant des NaN :", df_cleaned.shape)
```

```
Nombre de valeurs NaN dans le jeu de données :
uniq_id                    0
crawl_timestamp            0
product_url                0
product_name               0
product_category_tree      0
pid                        0
retail_price               1
discounted_price           1
image                      0
is_FK_Advantage_product    0
description                0
product_rating             0
overall_rating             0
brand                    338
product_specifications     1
dtype: int64

Forme du DataFrame après suppression des lignes contenant des NaN : (710, 15)
```

Prétraitement du texte

Tokenization

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
0     [Key, Features, of, Elegance, Polyester, Multi...
1     [Specifications, of, Sathiyas, Cotton, Bath, T...
3     [Key, Features, of, SANTOSH, ROYAL, FASHION, C...
4     [Key, Features, of, Jaipur, Print, Cotton, Flo...
33    [Key, Features, of, SANTOSH, ROYAL, FASHION, C...
34    [Key, Features, of, House, This, Queen, Cotton...
35    [Buy, Riva, Carpets, Cotton, Free, Bath, Mat, ...
36    [Myesquire, Ceramic, Burner, Pot, Lemongrass, ...
37    [Key, Features, of, Dungri, India, Craft, Ducj...
38    [Key, Features, of, BFT, 6, W, LED, Bulb, Pack...
Name: description_tokens, dtype: object
```

Nettoyage (stopwords,ponctuation)

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
0     [Key, Features, Elegance, Polyester, Multicolo...
1     [Specifications, Sathiyas, Cotton, Bath, Towel...
3     [Key, Features, SANTOSH, ROYAL, FASHION, Cotto...
4     [Key, Features, Jaipur, Print, Cotton, Floral,...
33    [Key, Features, SANTOSH, ROYAL, FASHION, Cotto...
34    [Key, Features, House, Queen, Cotton, Duvet, C...
35    [Buy, Riva, Carpets, Cotton, Free, Bath, Mat, ...
36    [Myesquire, Ceramic, Burner, Pot, Lemongrass, ...
37    [Key, Features, Dungri, India, Craft, Ducjug01...
38    [Key, Features, BFT, 6, W, LED, Bulb, Pack, 1,...
Name: clean_description_tokens, dtype: object
```

Lemmatization

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
0     [Key, Features, Elegance, Polyester, Multicolo...
1     [Specifications, Sathiyas, Cotton, Bath, Towel...
3     [Key, Features, SANTOSH, ROYAL, FASHION, Cotto...
4     [Key, Features, Jaipur, Print, Cotton, Floral,...
33    [Key, Features, SANTOSH, ROYAL, FASHION, Cotto...
34    [Key, Features, House, Queen, Cotton, Duvet, C...
35    [Buy, Riva, Carpets, Cotton, Free, Bath, Mat, ...
36    [Myesquire, Ceramic, Burner, Pot, Lemongrass, ...
37    [Key, Features, Dungri, India, Craft, Ducjug01...
38    [Key, Features, BFT, 6, W, LED, Bulb, Pack, 1,...
Name: lemmatized_description_tokens, dtype: object
```

Etude de faisabilite de la classification

méthode :

1. encodage du texte (bow, tf-idf, word2vec, bert, use)
2. réduction de la dimensionnalité des caractéristiques textuelles à 2d avec t-sne pour la visualisation.
3. entraînement d'un classificateur kmeans sur l'ensemble du dataset.
4. évaluation du classificateur (ari).
5. création d'une visualisation par nuage de points comparant les étiquettes réelles et prédites.

Représentation des données textuelles

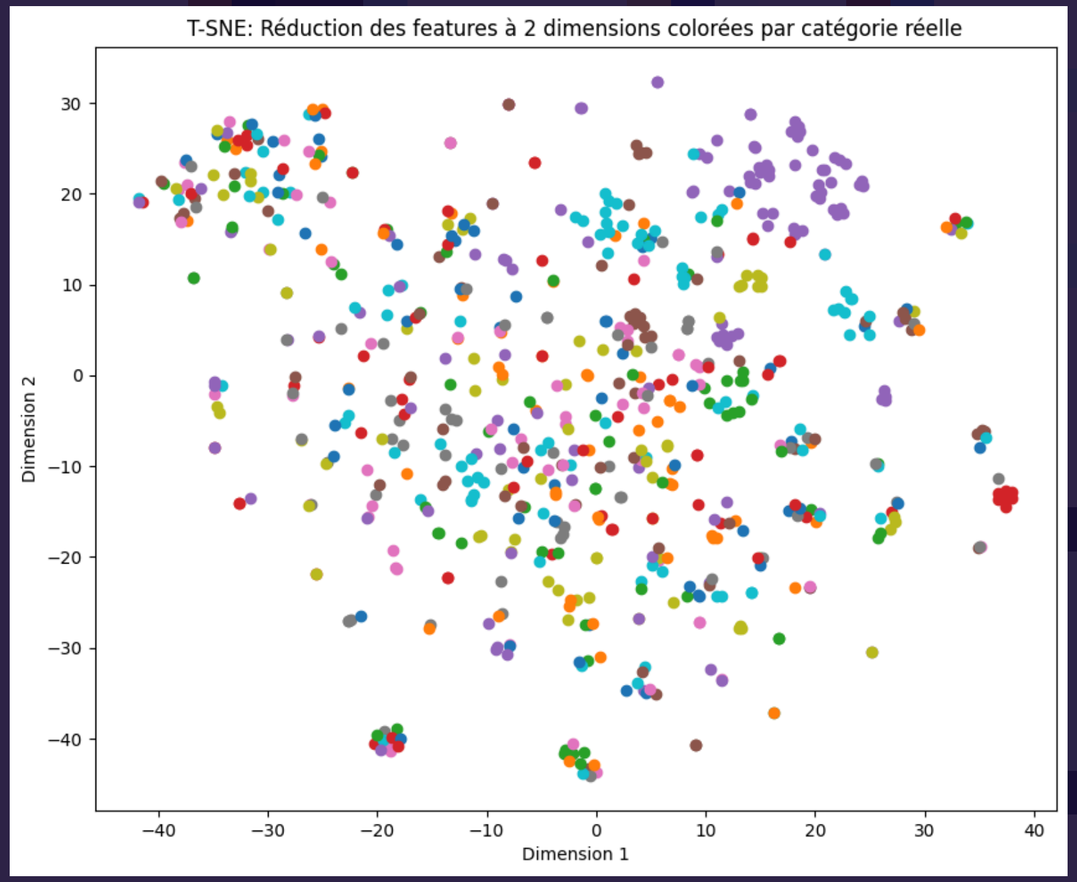
Création d’ un Bag of Words et calcule duTF-IDF (Term Frequency-Inverse Document Frequency) pour représenter chaque document textuel

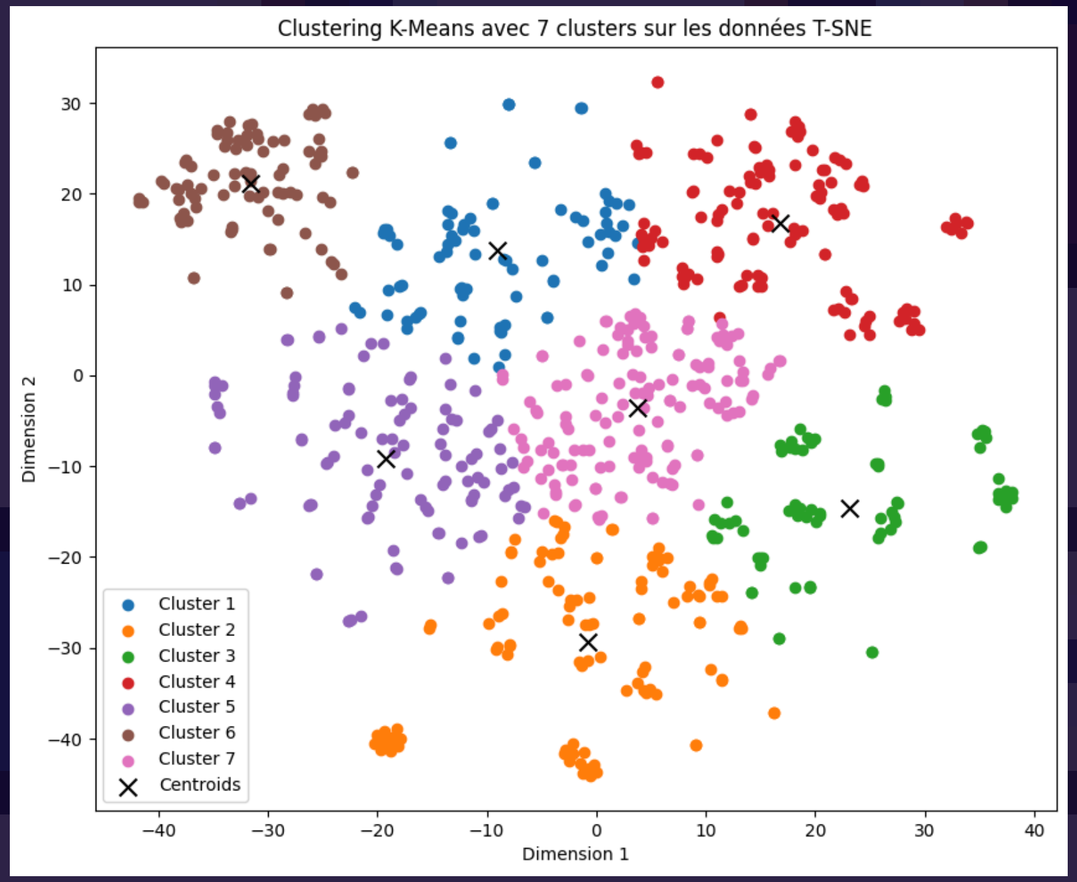
```
[ ]  from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

     # Créer un vectoriseur de mots en sac (Bag of Words)
     count_vectorizer = CountVectorizer()
     bag_of_words = count_vectorizer.fit_transform(df['lemmatized_description_tokens'].apply(lambda x: ' '.join(x)))

     # Calculer le TF-IDF
     tfidf_transformer = TfidfTransformer()
     tfidf_matrix = tfidf_transformer.fit_transform(bag_of_words)

     # Afficher la taille de la matrice TF-IDF
     print("Taille de la matrice TF-IDF :", tfidf_matrix.shape)
```

```
Taille de la matrice TF-IDF : (710, 4991)
```

Nuage de mots

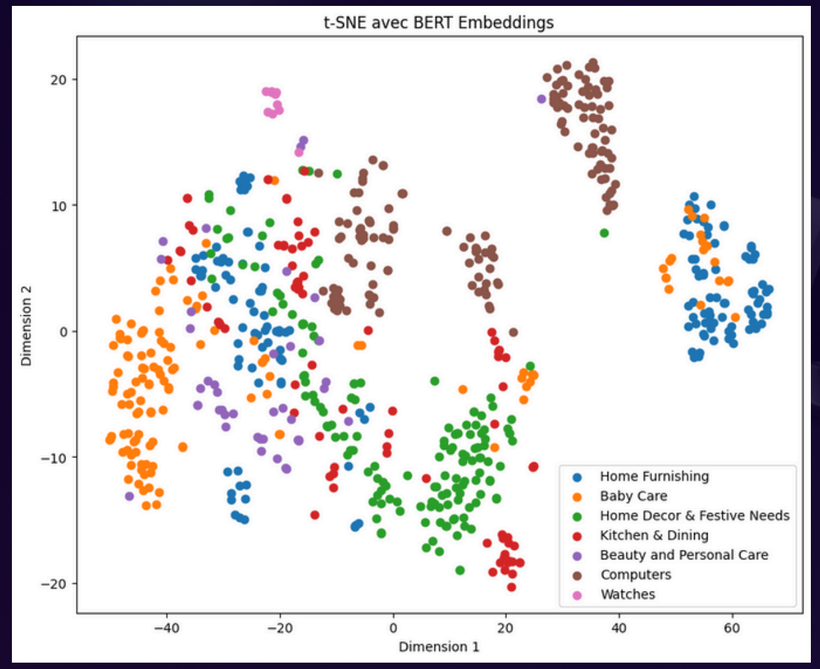
Visualisation TSNE

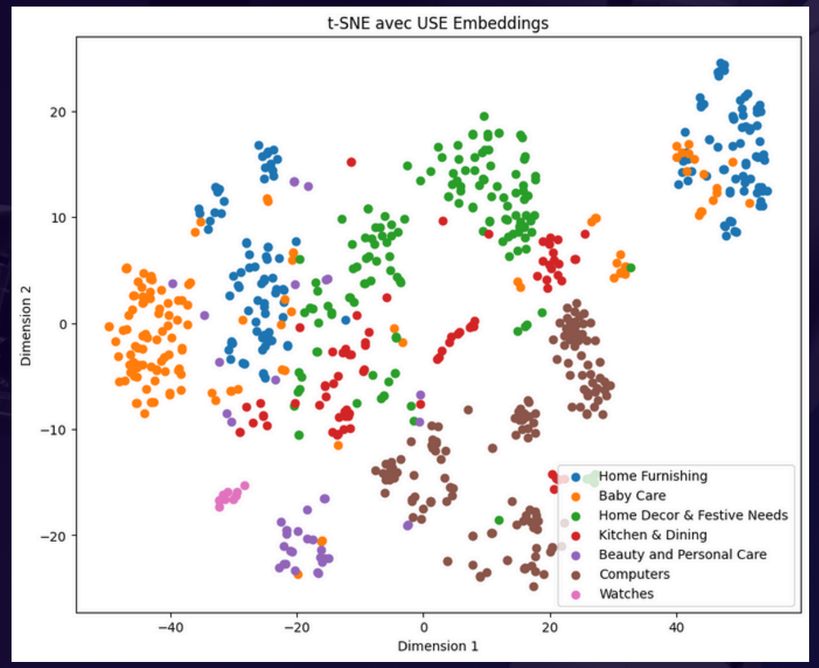


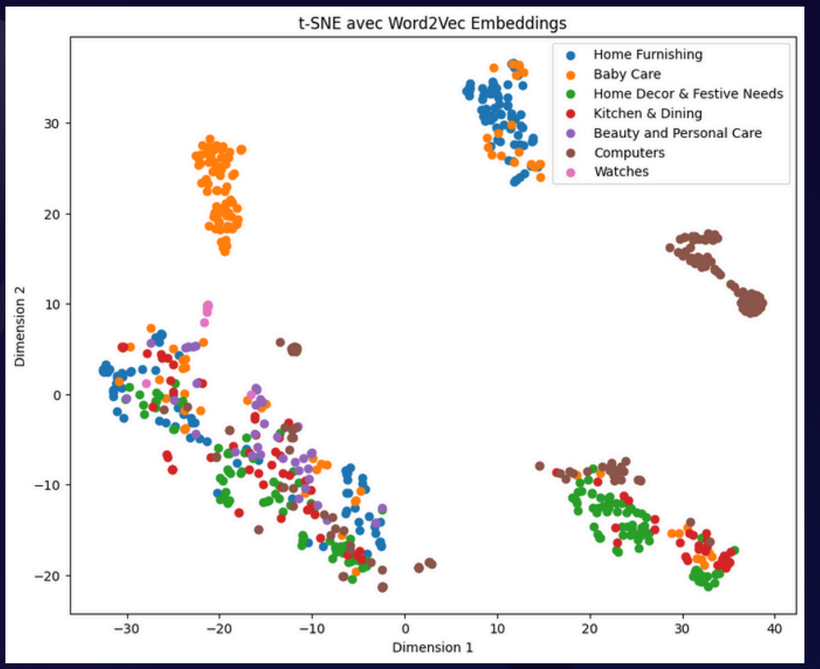



```
ARI pour l'approche BoW: 0.012991031878840497
ARI pour l'approche TF-IDF: 0.028060752988776774
```

Embedding de mots








```
ARI avec TF-IDF: 0.028060752988776774
ARI avec Word2Vec: 0.2390172759426602
ARI avec BERT: 0.3160916833723786
ARI avec USE: 0.34010444964501146
```

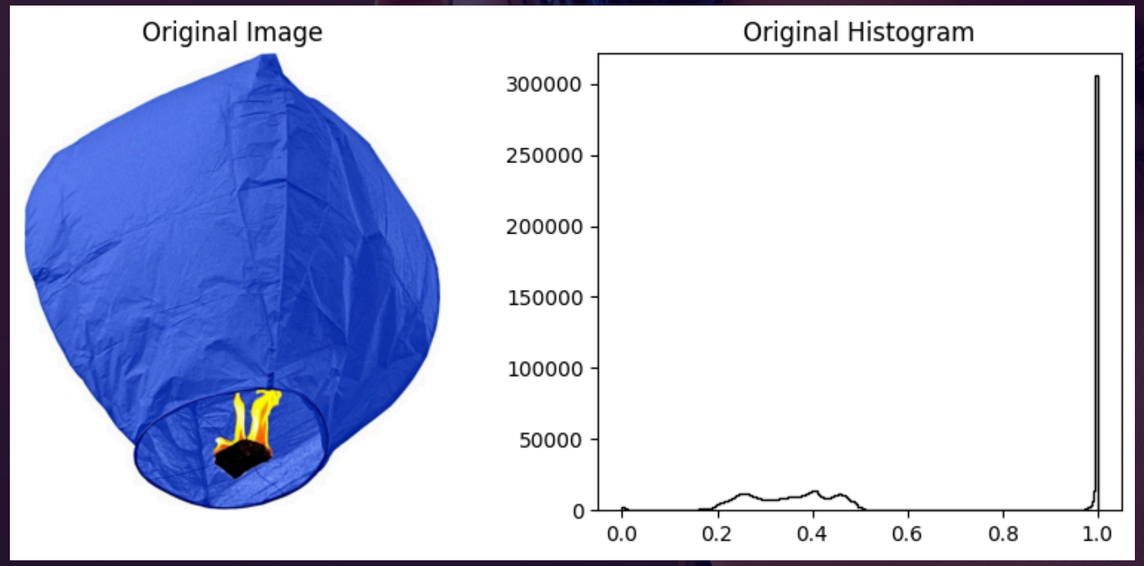
Traitement d'images

méthode :

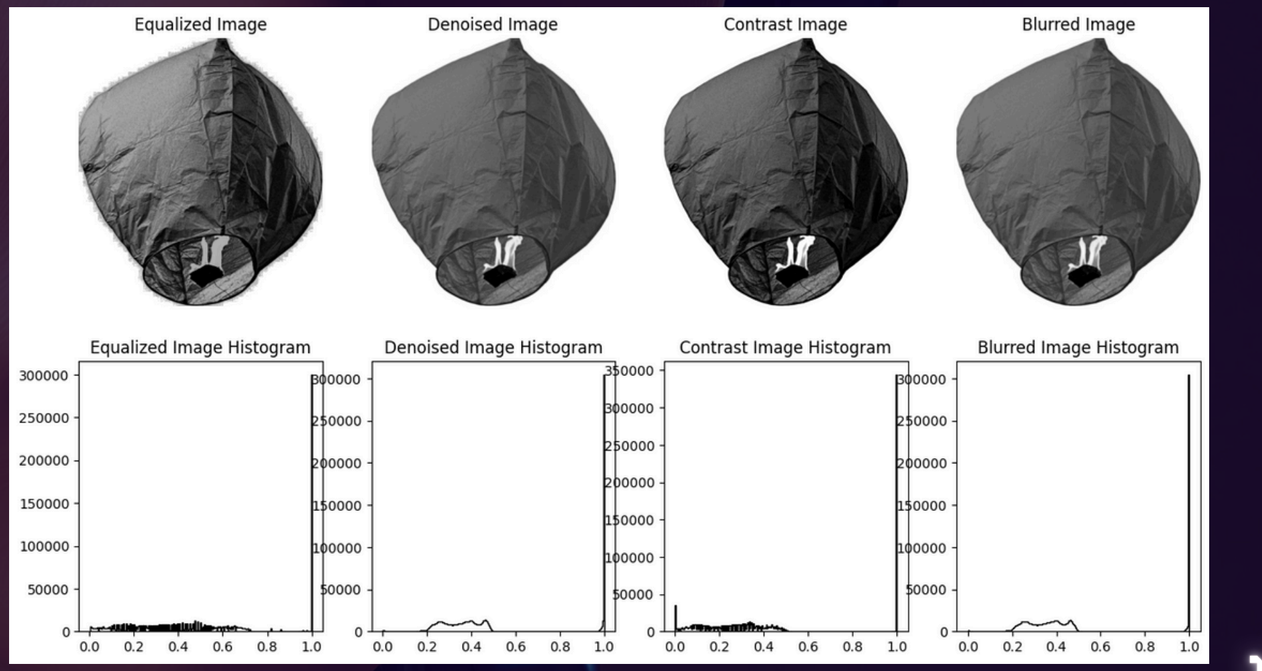
- pré-traitement d'images
- extraction de descripteurs (SIFT)
- création de clusters de descriptors
- création de features <-> bag of visual word (bovw)
- représentation en 2d (t-sne)
- analyse visuelle puis évaluation de l'ari

Traitement d’images

Affichage de l’image d’exemple

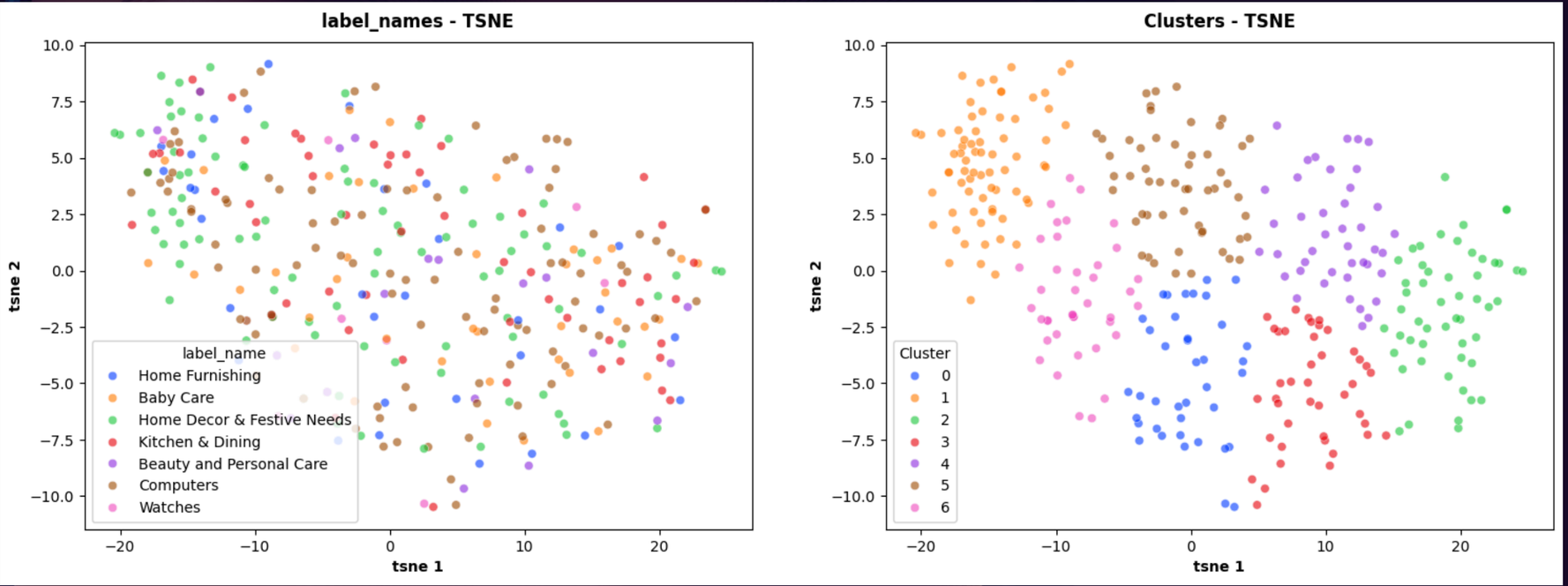


Et ses transformations



Faisabilité de la classification

Nombre de descripteurs SIFT : 1644



Visulations TSNE valeurs réelles et valeurs prédites avec le clustering
Adjusted Rand Index (ARI) Score: 0.014611995302782 27

Analyse de faisabilité de classification automatique d’images via CNN Transfer Learning

Extraction des caracteristiques d’images avec Resnet50

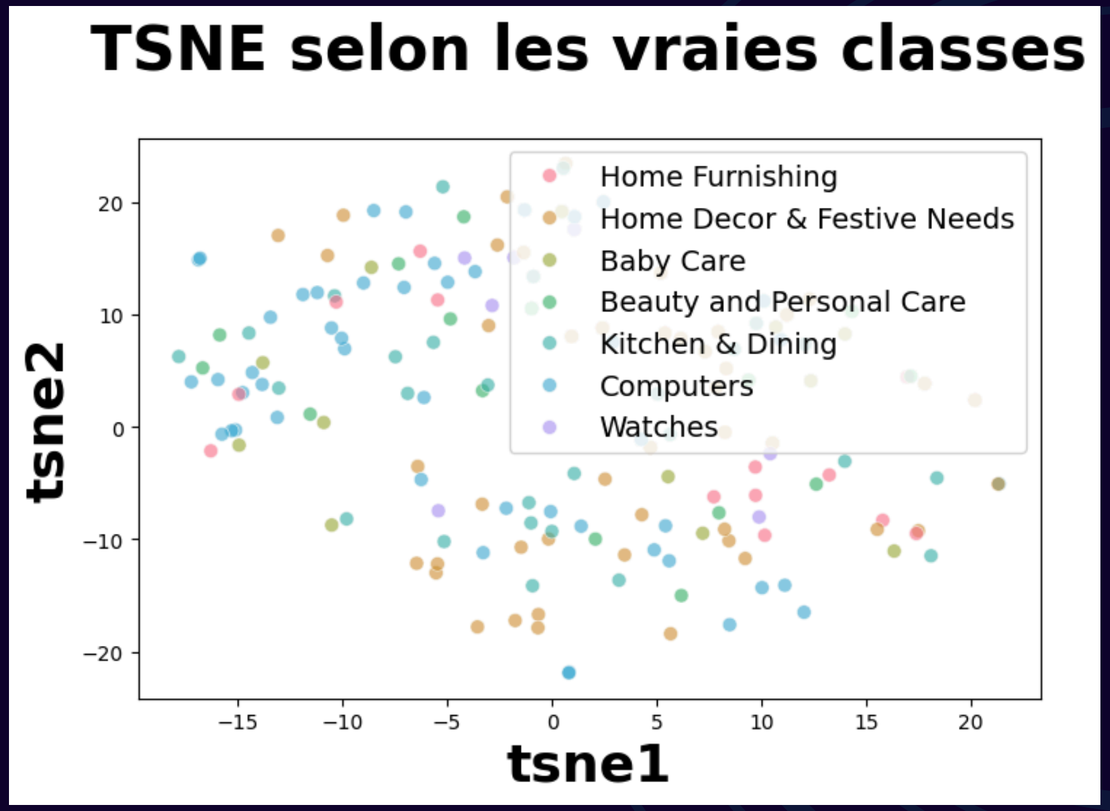
Model: "model"

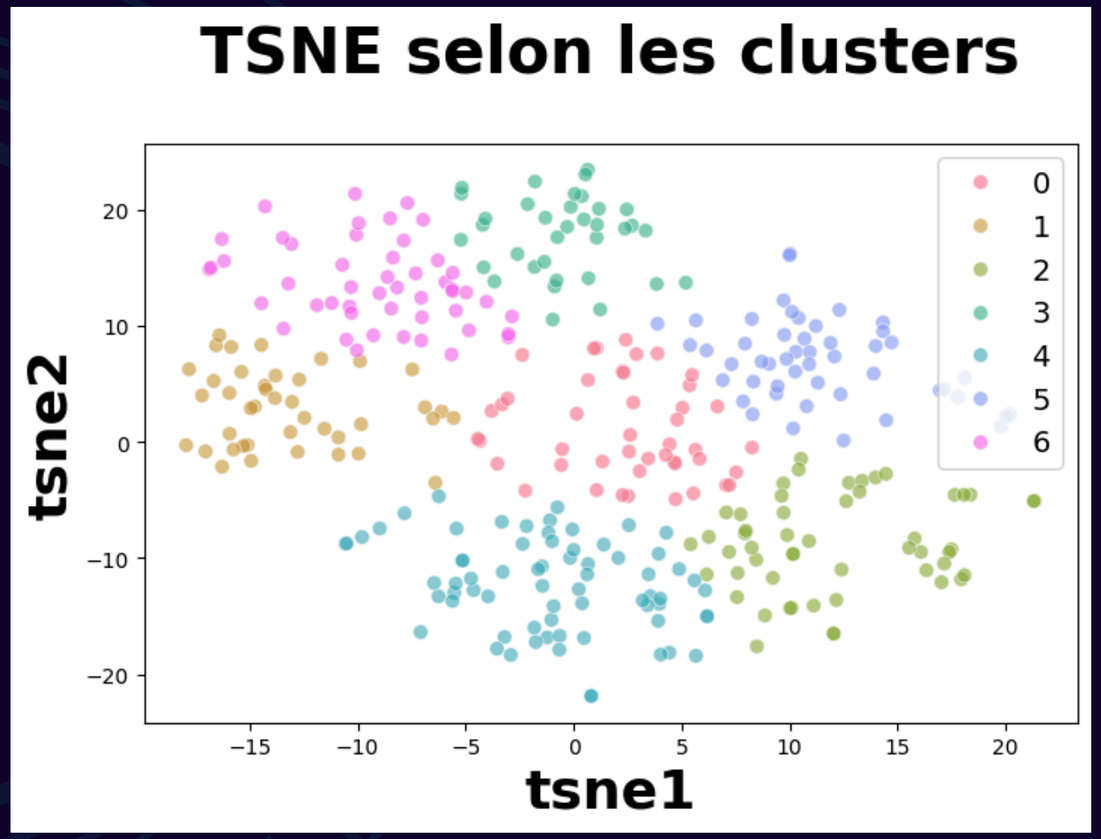
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv1_pad (ZeroPadding2D)	(None, 238, 238, 3)	0	['input_1[0][0]']
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	['conv1_pad[0][0]']
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	['conv1_conv[0][0]']
conv1_relu (Activation)	(None, 112, 112, 64)	0	['conv1_bn[0][0]']
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	['conv1_relu[0][0]']
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	['pool1_pad[0][0]']
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	['pool1_pool[0][0]']
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_1_conv[0][0]']
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_1_bn[0][0]']
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	['conv2_block1_1_relu[0][0]']
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_2_conv[0][0]']
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_2_bn[0][0]']
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	['pool1_pool[0][0]']
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	['conv2_block1_2_relu[0][0]']
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_0_conv[0][0]']
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_3_conv[0][0]']

Réduction de la dimmensionnalité avec l’acp

```
[ ]  # Réduction de la dimensionnalité avec PCA
     pca = PCA(n_components=0.99)
     feat_pca = pca.fit_transform(images_features)
     print(feat_pca.shape)

     (340, 291)
```

Analyse de faisabilité de classification automatique d’images via CNN Transfer Learning

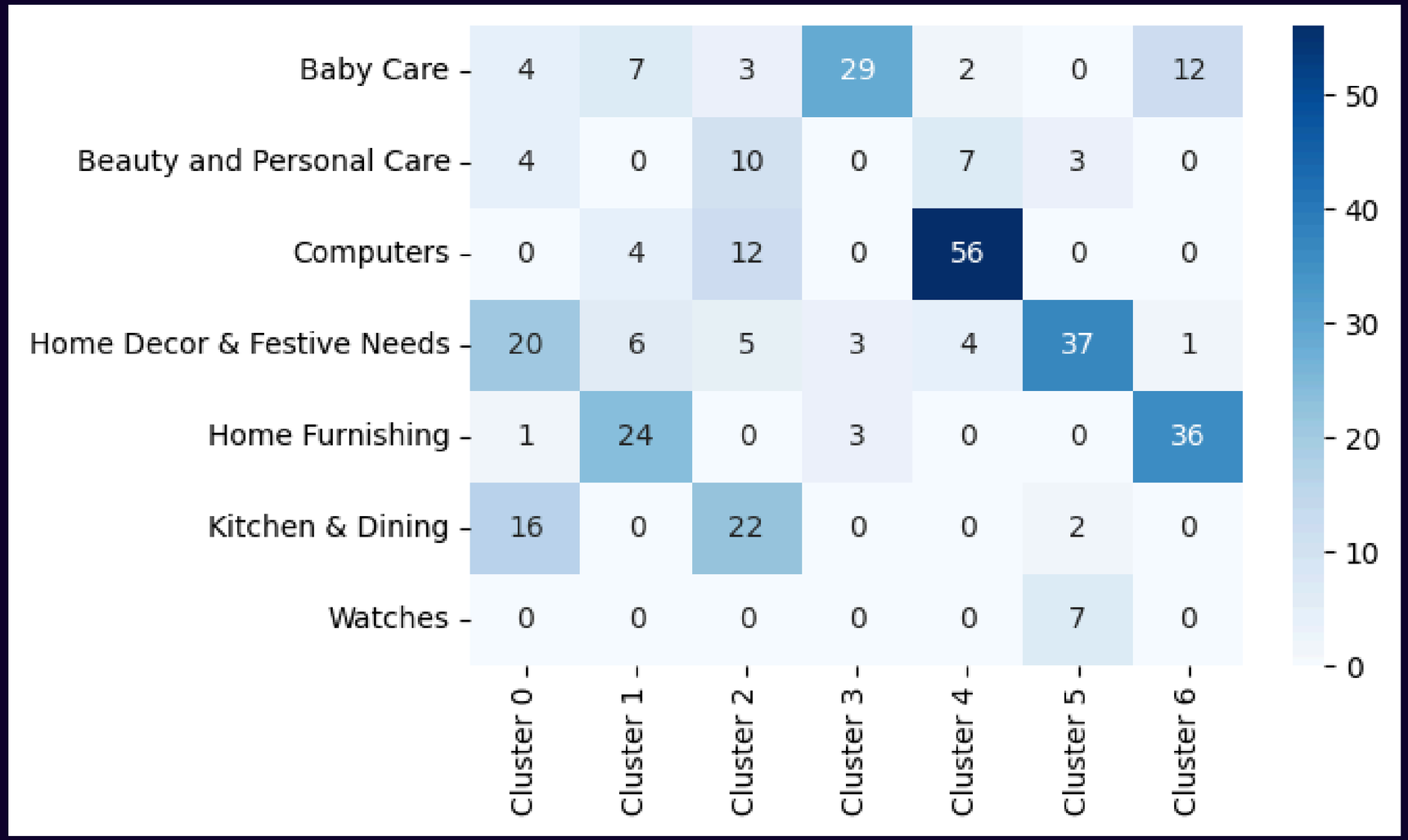




TSNE selon les vraies classes et les clusters

ARI : 0.3706247954199613

Analyse de faisabilité de classification automatique d’images via CNN Transfer Learning



Matrice de confusion

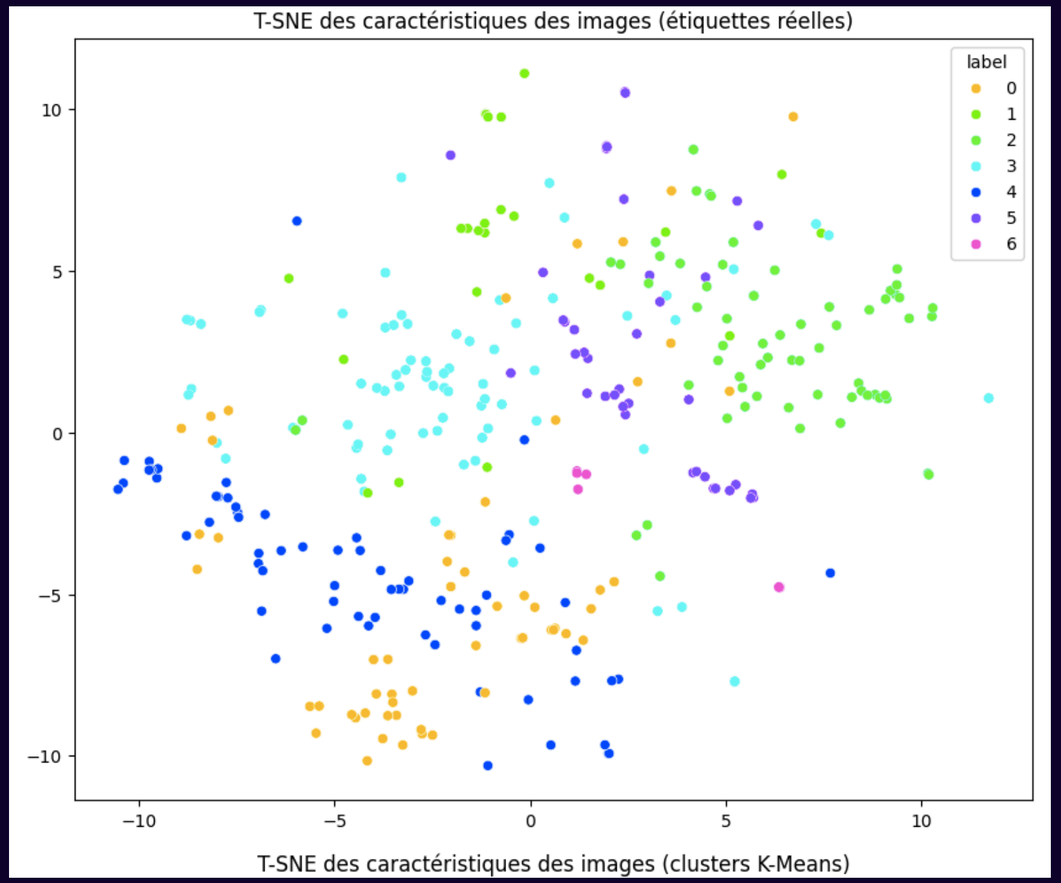
Classification supervisée d’images via CNN transfer Learning

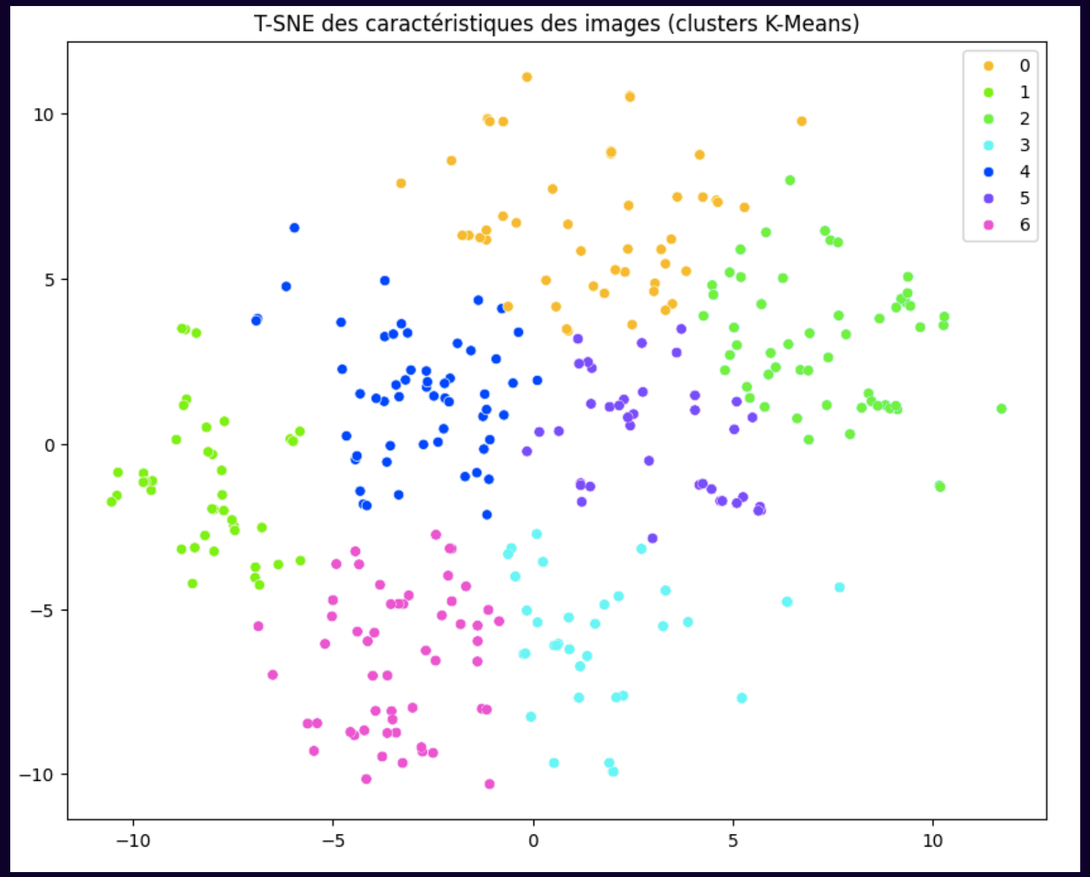
```python
batch_size = 32
num_classes = len(df["label"].unique())

# Création des générateurs de données avec Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

train_generator = train_datagen.flow_from_dataframe(
    df,
    x_col='image_path',
    y_col='label_name',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_dataframe(
    df,
    x_col='image_path',
    y_col='label_name',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
8/8 [==============================] - ETA: 0s - loss: 1.6476 - accuracy: 0.3667WARNING:tensorflow:Can save best model only with val_los-=s available, skipp
8/8 [==============================] - 70s 9s/step - loss: 1.6476 - accuracy: 0.3667 - val_loss: 1.8993 - val_accuracy: 0.1719
3/3 [==============================] - 16s 3s/step - loss: 1.9277 - accuracy: 0.2353
Validation loss: 1.92767333984375
Validation accuracy: 0.23529411852359772
```

Modèle Resnet50

Classification supervisée d’images via CNN transfer Learning





ARI: 0.34893069431111196

Test de l’API

méthode:

création d’un compte

obtention de x-rapidapi-key et x-rapidapi-host.

fetch data

extraction des informations importantes sauvegarde (csv)

Test de l’API

	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_am5egz6aq3fpjlaf8xpkdbc2asis	Champagne Truffles	Generic meals	butter; cocoa; sweetened condensed milk; vanil...	NaN
8	food_bcz8rhiajk1fuva0vkfmeakbouc0	Champagne Vinaigrette	Generic meals	champagne vinegar; olive oil; Dijon mustard; s...	NaN
9	food_a79xmnya6togreaeukbroa0thhh0	Champagne Chicken	Generic meals	Flour; Salt; Pepper; Boneless, Skinless Chicke...	NaN

Conclusion

À travers l'utilisation avancée de techniques de traitement de texte et d'images, nous avons démontré la faisabilité d'améliorer significativement l'expérience utilisateur actuelle, souvent entravée par une classification manuelle sujette à l'erreur.

En adoptant des approches telles que le traitement du langage naturel avec USE et l'utilisation de CNN transfer learning pour l'analyse d'images, nous avons non seulement optimisé la précision des classifications mais aussi facilité la gestion à grande échelle des produits sur la plateforme.

merci pour votre écoute !

place aux questions ...