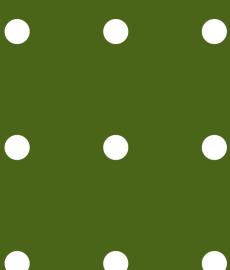


Place de marché

Classification automatique
des biens de consommation

Présenté par Thierry KAPPE.



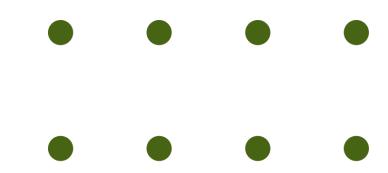
Le contexte

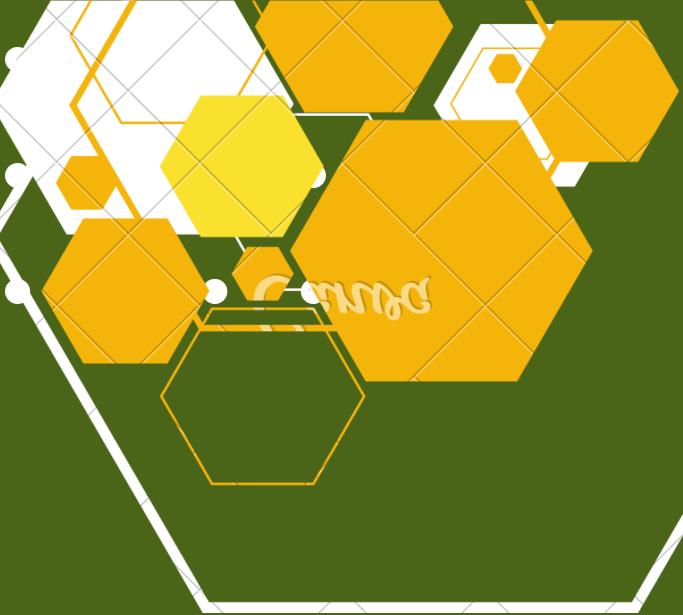
L'entreprise Place de marché souhaite lancer une marketplace qui permettra à des vendeurs de proposer des articles à des acheteurs en postant une photo et une description.

Afin de fluidifier l'expérience des utilisateurs, l'attribution de la catégorie d'un article qui est actuellement manuelle doit être automatisée.

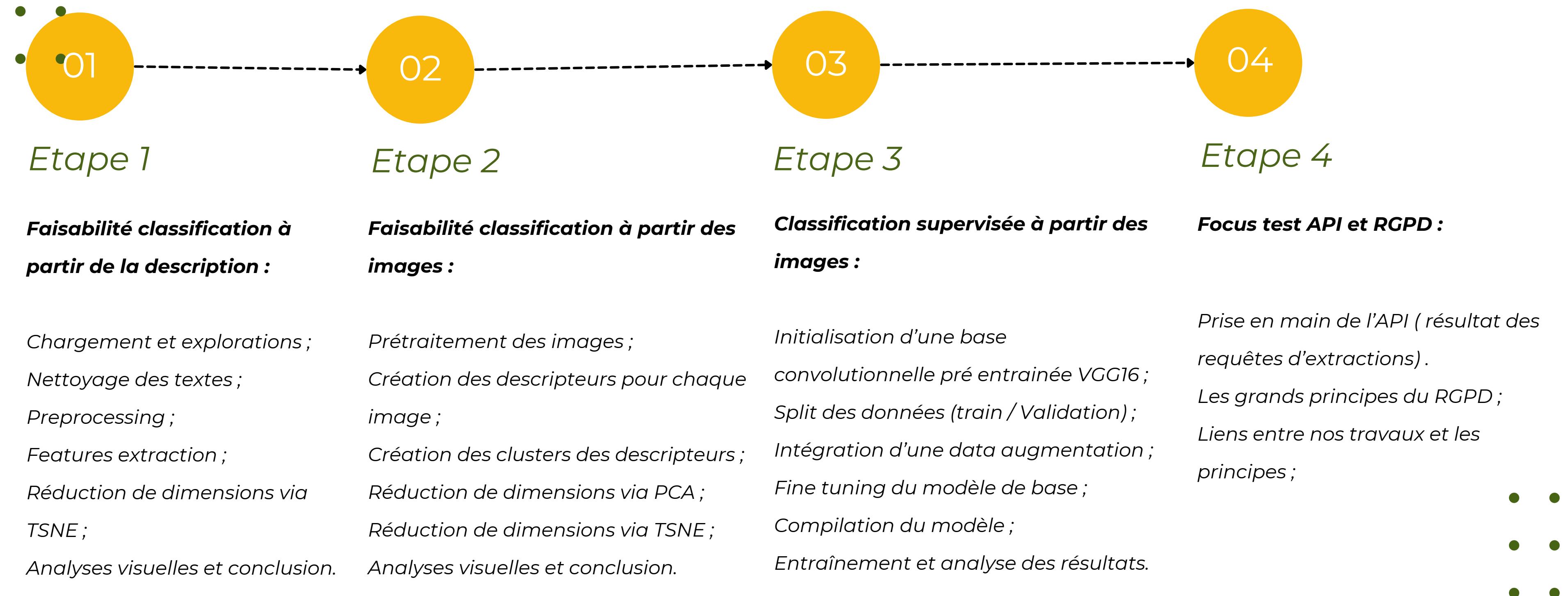
Pour ce faire, nous devons dans un premier temps étudier la faisabilité d'une classification automatique des articles en fonction des descriptifs et des images, ensuite réaliser une classification automatique en se basant sur les images des différents produits.

Cette présentation est soutenue par : un notebook de prétraitement (études de faisabilité), un notebook de classification supervisée des images, un script python de test de l'API ainsi qu'un fichier.csv des produits extraits.





Vue globale des étapes clés



Faisabilité classification à partir de la description



- **Chargement et exploration des données**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1050 entries, 0 to 1049
Data columns (total 15 columns):
 #   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
```

- **Données après extraction des catégories et des seules colonnes nécessaires pour nos analyses**

uniq_id	image	description	product_category
55b85ea15a1536d46b7190ad6fff8ce7	55b85ea15a1536d46b7190ad6fff8ce7.jpg	Key Features of Elegance Polyester Multicolor Abstract Eyelet Door Curtain,Door Curtain (210 cm in Height, Pack of 2) Price: Rs. 899 Th...	Home Furnishing
Tb72c92c2f6c40268628ec5f14c6d590	Tb72c92c2f6c40268628ec5f14c6d590.jpg	Specifications of Sathiya's Cotton Bath Towel (3 Bath Towel, Red, Yellow, Blue) Bath Towel Features Machine Washable Yes Material...	Baby Care
64d5d4a258243731dc7bbb1eeff49ad74	64d5d4a258243731dc7bbb1eeff49ad74.jpg	Key Features of Europa Cotton Terry Face Towel Set Size: small Height: 9 inch GSM: 360.Europa Cotton Terry Face Towel Set (20...	Baby Care
d4684dc759dd9cdf41504698d737d8	d4684dc759dd9cdf41504698d737d8.jpg	Key Features of SANTOSH ROYAL FASHION Cotton Printed King sized Double Bedsheet Royal Bedsheet Perfect for Wedding & Gifting.Specifications of SANTOSH ROYAL FASHION Cotton Printed King sized Doub...	Home Furnishing
6325b6870c54cd47be6ebfbffa620ec7	6325b6870c54cd47be6ebfbffa620ec7.jpg	Key Features of Jaipur Print Cotton Floral King sized Double Bedsheet 100% cotton,Jaipur Print Cotton Floral King sized Double Bedsheet (1 bed sheet 2 pillow cover, White) Price: Rs. 998 This nice...	Home Furnishing

- **Taille et composition des catégories**

```
Counter({'Home Furnishing': 150,
         'Baby Care': 150,
         'Watches': 150,
         'Home Decor & Festive Needs': 150,
         'Kitchen & Dining': 150,
         'Beauty and Personal Care': 150,
         'Computers': 150})
```

Faisabilité classification à partir de la description



- **Nettoyage des textes de description**

==> Conversion en minuscule
 ==> Suppression des URLs
 ==> Suppression des html
 ==> Suppression des caractères non ASCII
 ==> Suppression des éléments de ponctuation
 ==> Suppression des chiffres

- **Preprocessing des textes de description**

==> Suppression des mots vides
 ==> Suppression des mots pas assez ou trop fréquents
 ==> Tokenisation
 ==> Stemming via PorterStemmer
 ==> Lemmatisation

- **Features extraction**

==> BoW (Countvectorizer, Tf-idf, Word2Vec)
 ==> BoW avec Lemmatisation
 ==> Deep Learning (USE et BERT)

- **Création de la matrice d'embedding**

```
Create Embedding matrix ...
Word embedding rate : 1.0
Embedding matrix: (3054, 300)
```

- **Création du modèle d'embedding**

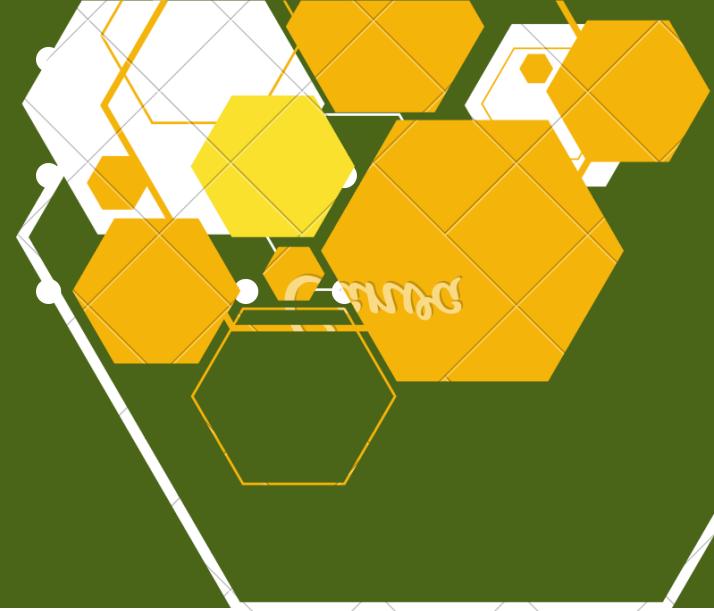
Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 270)	0
embedding (Embedding)	(None, 270, 300)	916,200
global_average_pooling1d (GlobalAveragePooling1D)	(None, 300)	0

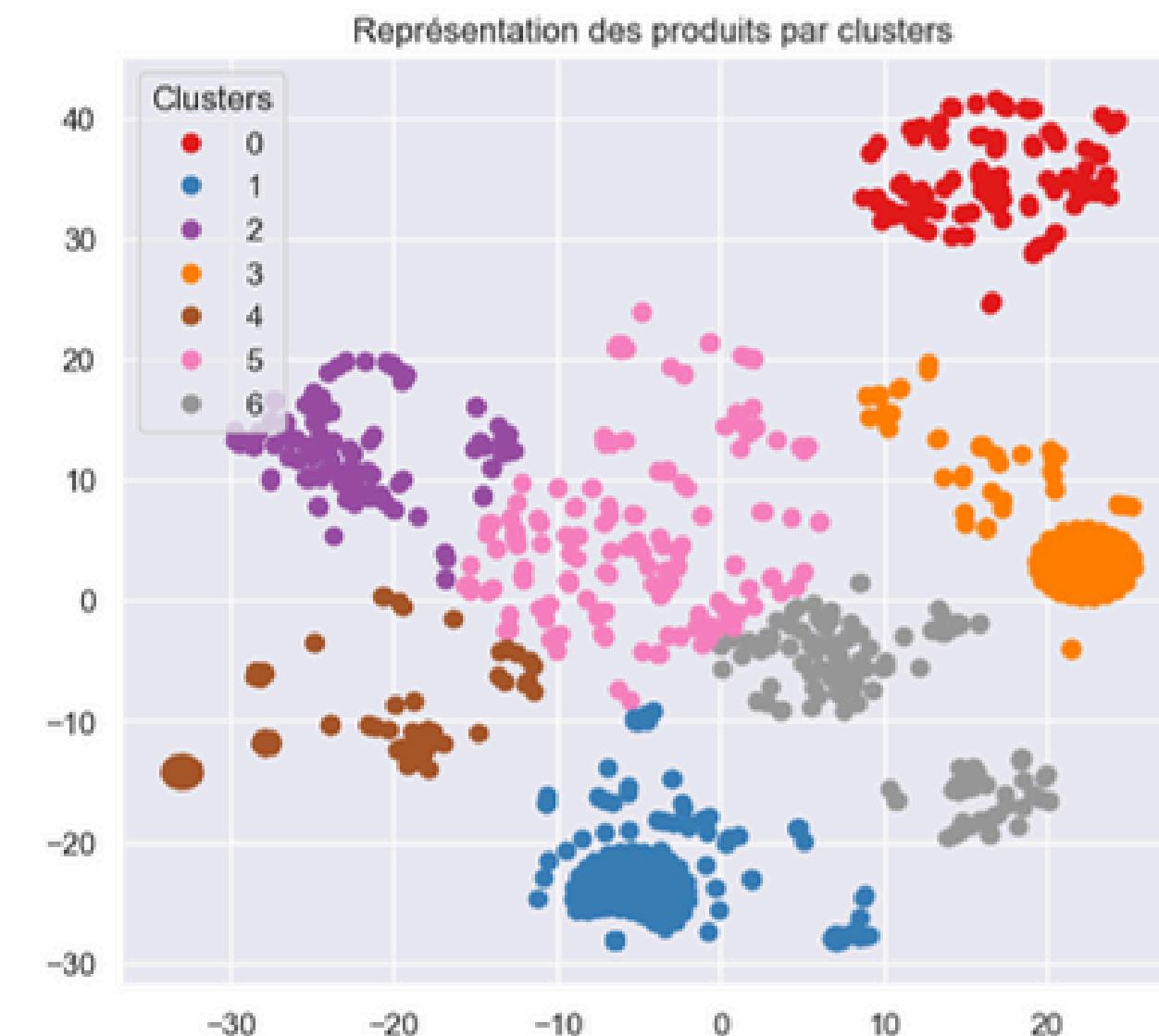
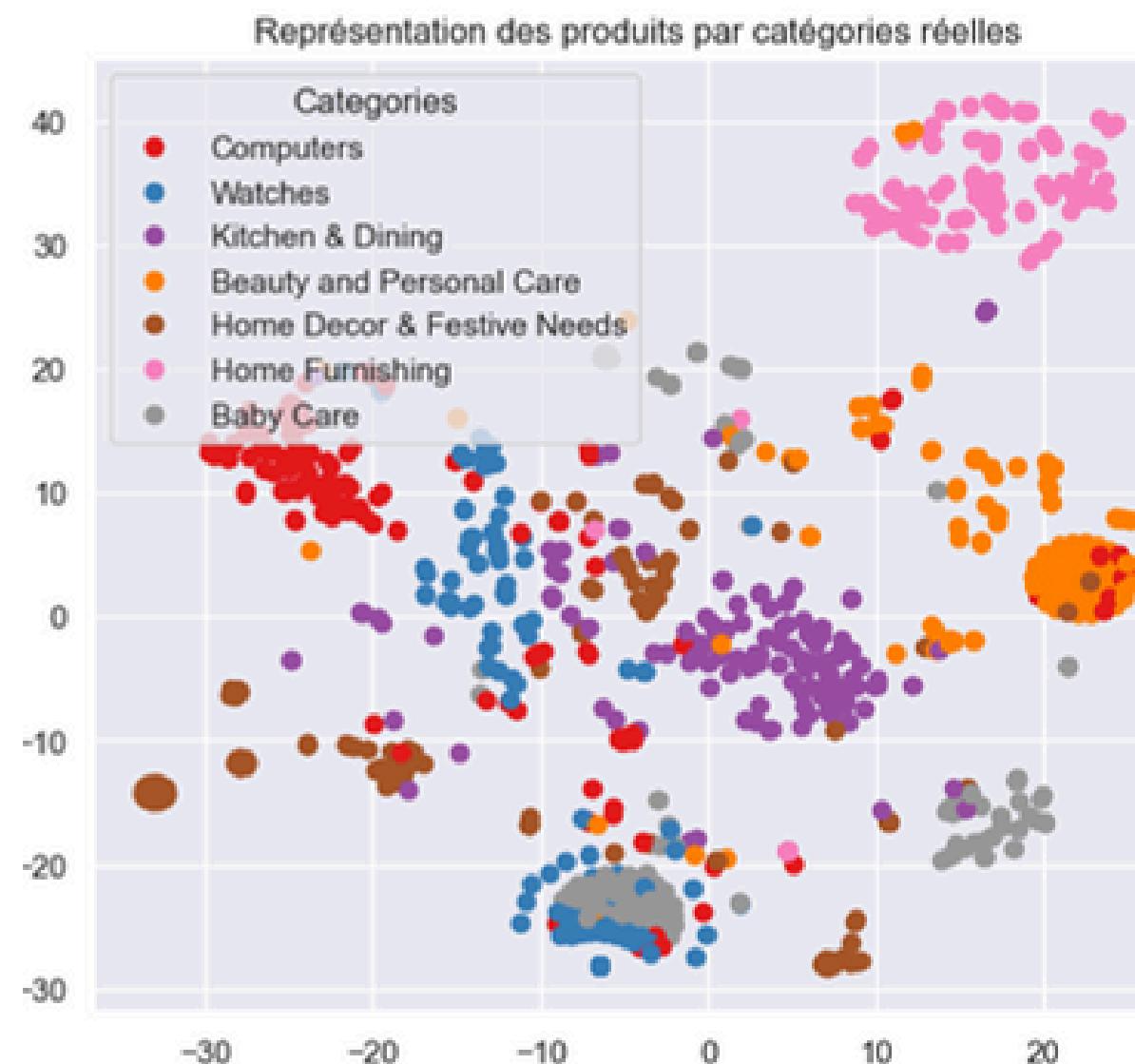
- **Synthèse des résultats**

Technique	Taille
BoW avec Countvectorizer	792
BoW avec Tf-idf	792
BoW avec Word2Vec	3053
matrice d'embedding	(3054, 300)
modèle d'embedding	(1050, 300)
T-SNE	(1050, 2)

Faisabilité classification à partir de la description



- Réduction de dimensions via T-SNE et analyses visuelles



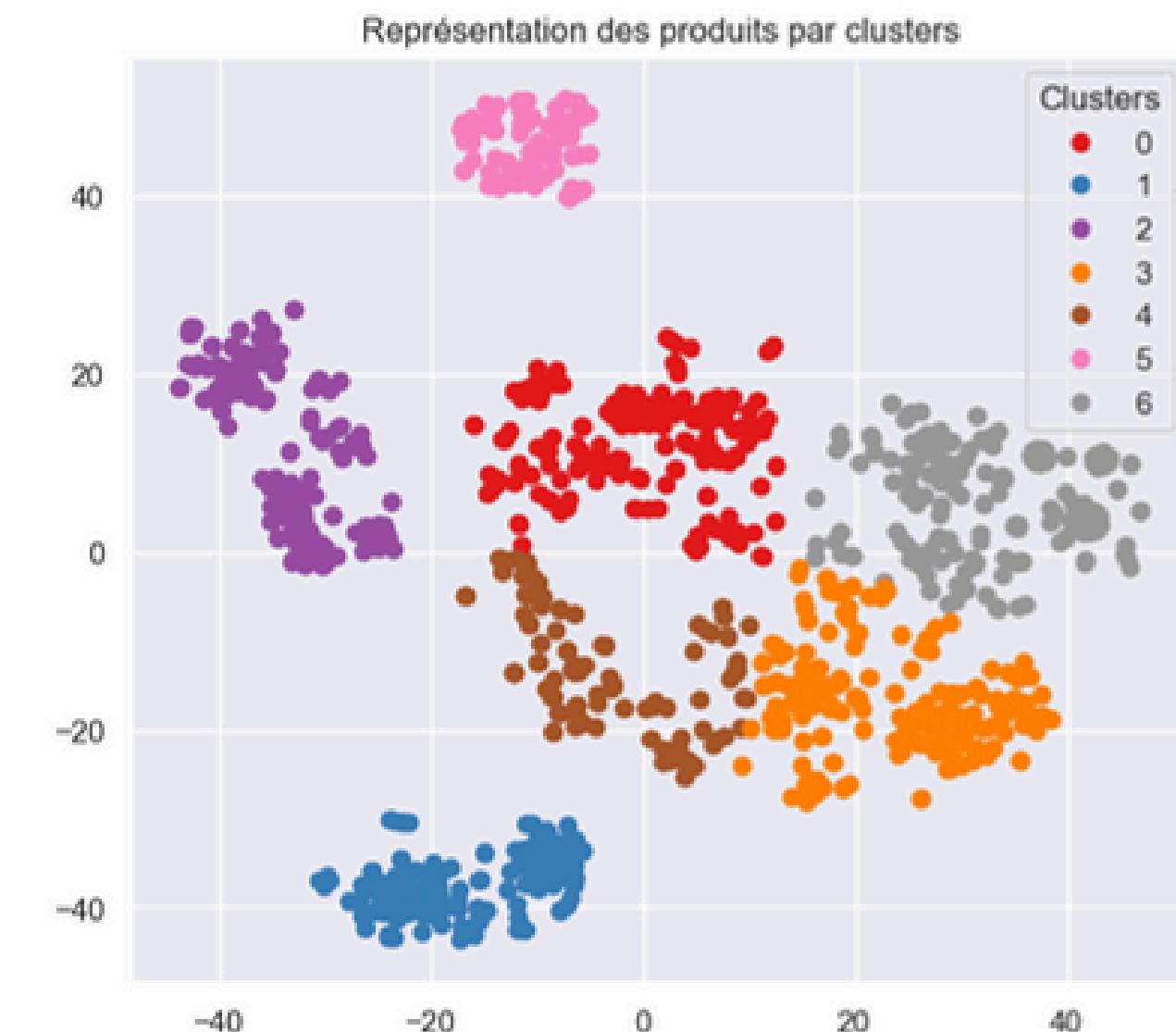
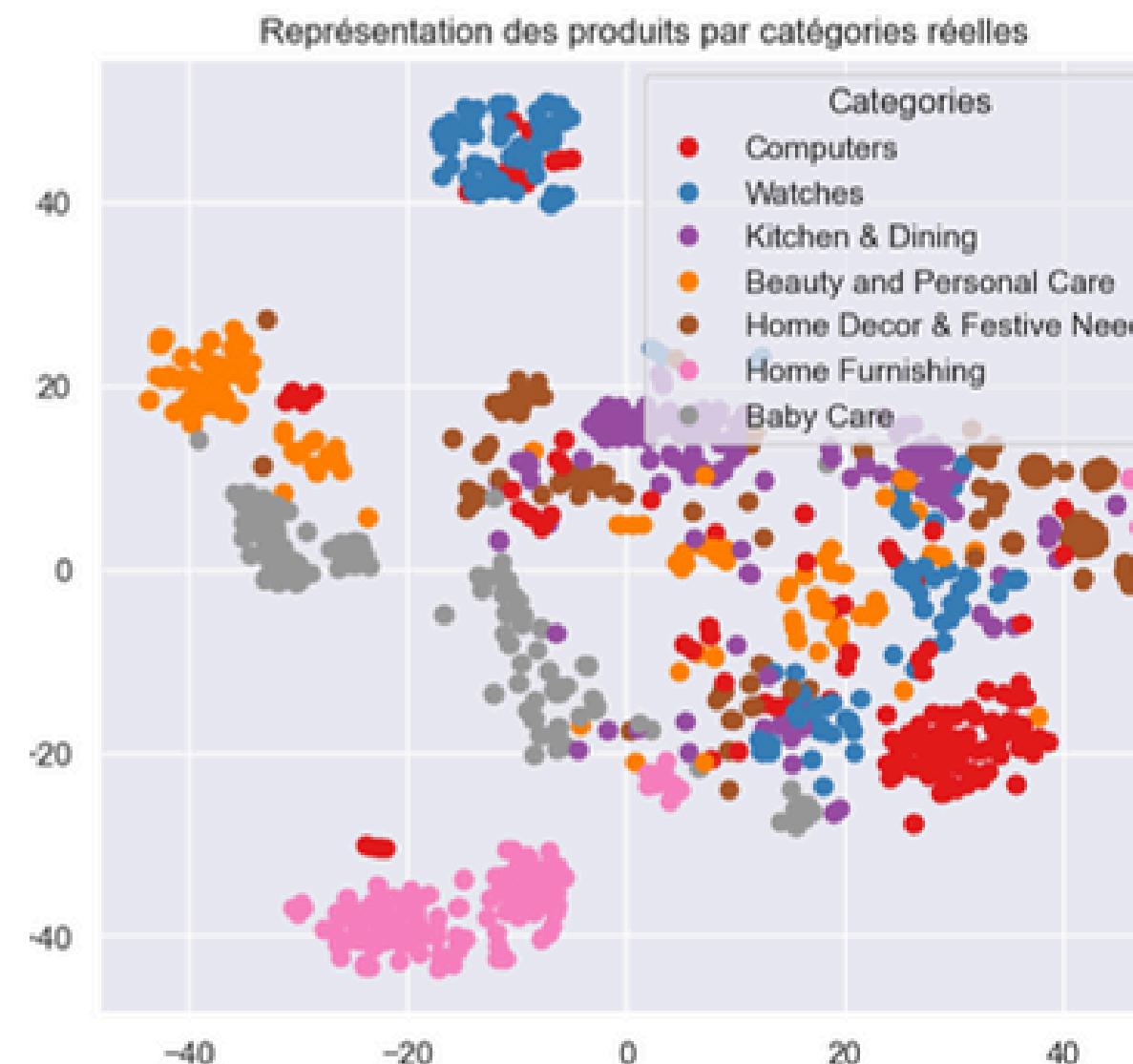
Score ARI : 0.38

L'analyse graphique montre visuellement qu'il est réalisable de séparer automatiquement les produits selon leurs descriptions.

Faisabilité classification à partir de la description



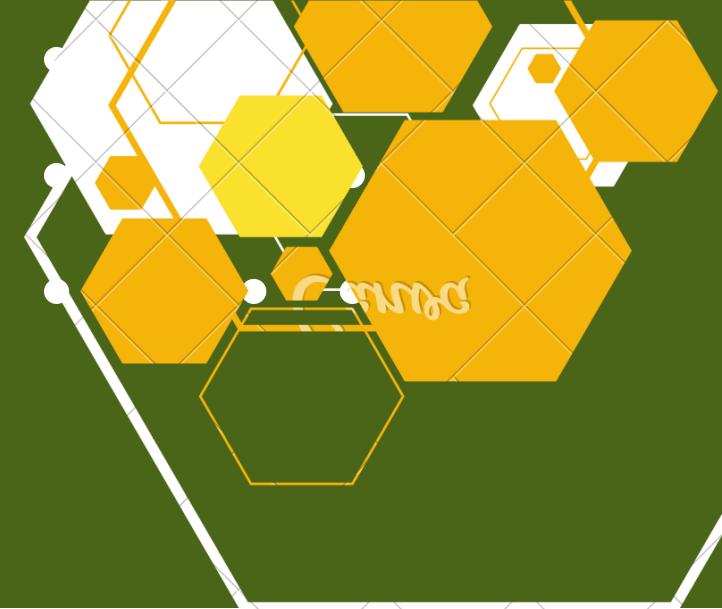
- **Visualisation après mise en place du Transformer BERT**



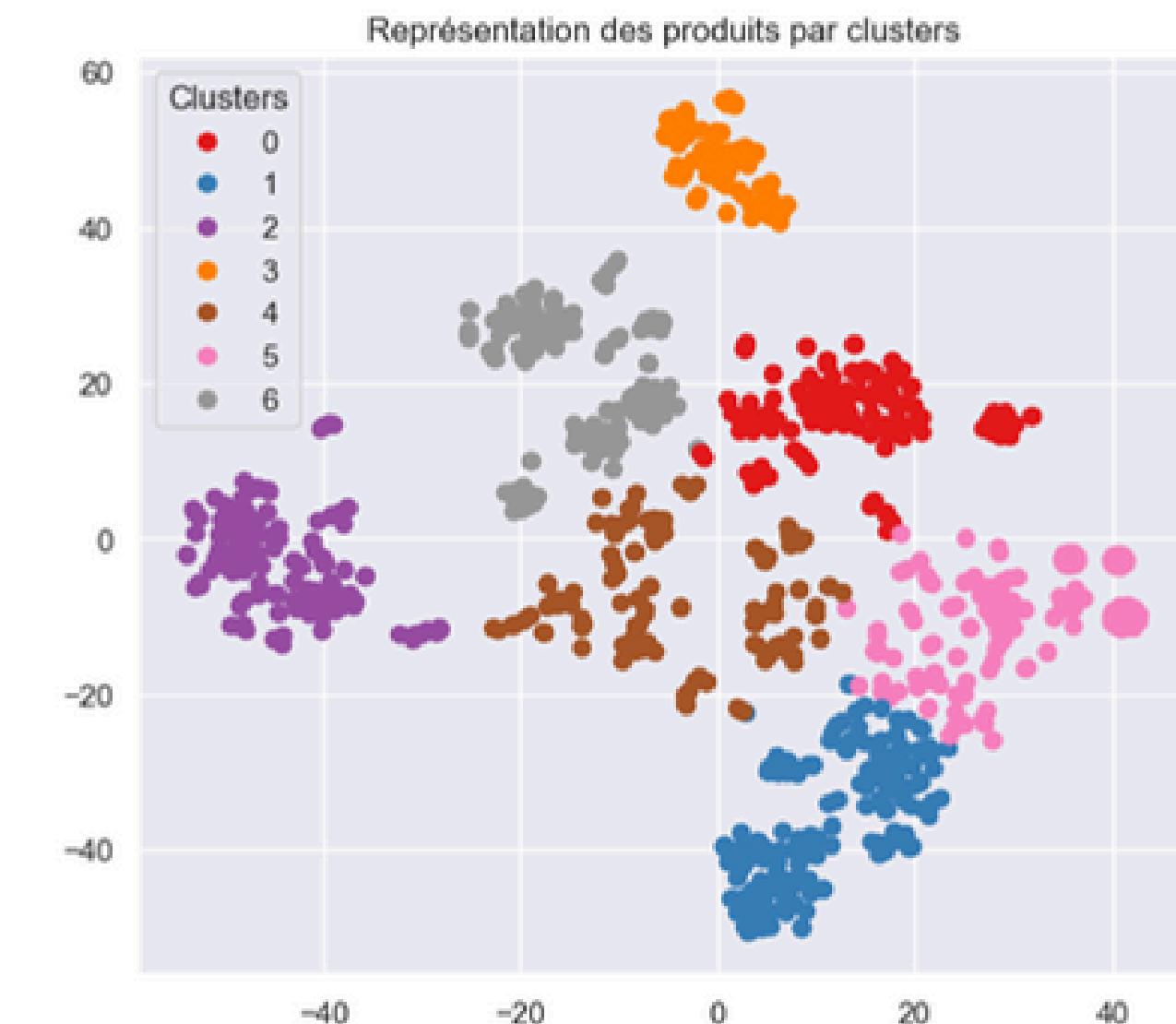
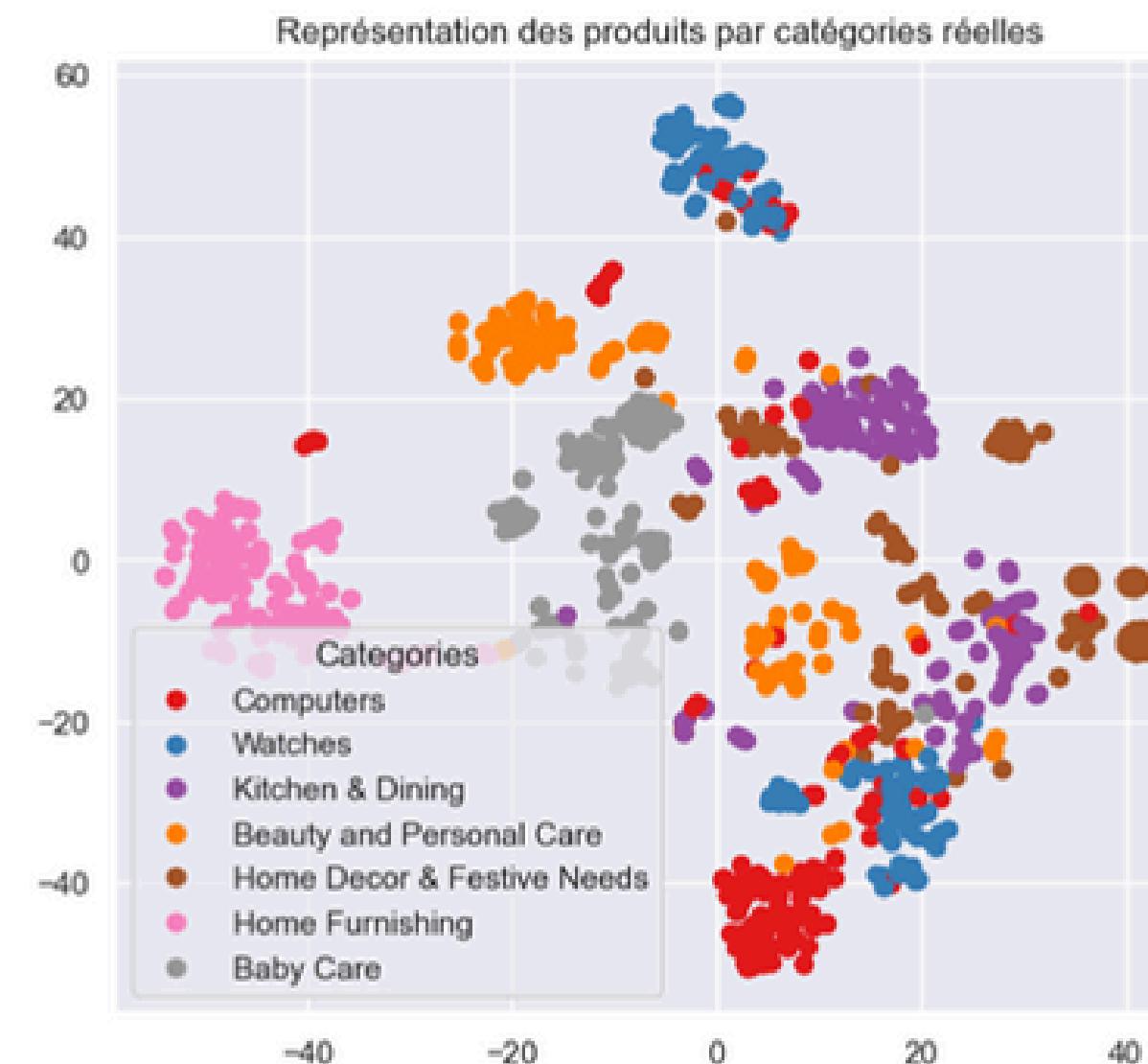
Score ARI : 0.33

L'analyse graphique montre visuellement qu'il est réalisable de séparer automatiquement les produits selon leurs descriptions.

Faisabilité classification à partir de la description



- **Visualisation après mise en place de l'architecture profonde USE**

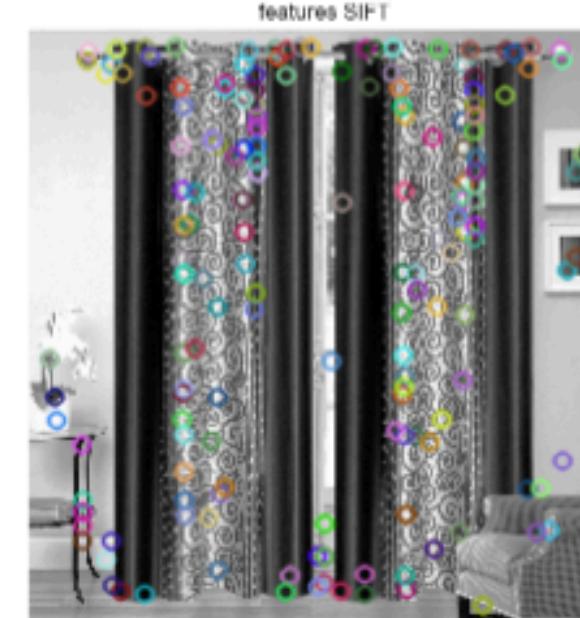


L'analyse graphique montre visuellement qu'il est réalisable de séparer automatiquement les produits selon leurs descriptions. Cette faisabilité est encore plus évidente via la mise en place d'une architecture profonde telle que USE

Faisabilité classification à partir des images



- **Prétraitement des images**



- **Création des descripteurs de chaque image via SIFT**

Nombre de descripteurs : (517306, 128)
Temps de traitement SIFT descriptor : 1934.25 secondes

- **Création des clusters de descripteurs**

Nombre de clusters estimés : 719
Création de 719 clusters de descripteurs ...
Temps de traitement kmeans : 27.02 secondes

- **Création des features des images**

Dimensions dataset avant réduction PCA : (1050, 719)
Temps de création histogrammes : 162.78 secondes

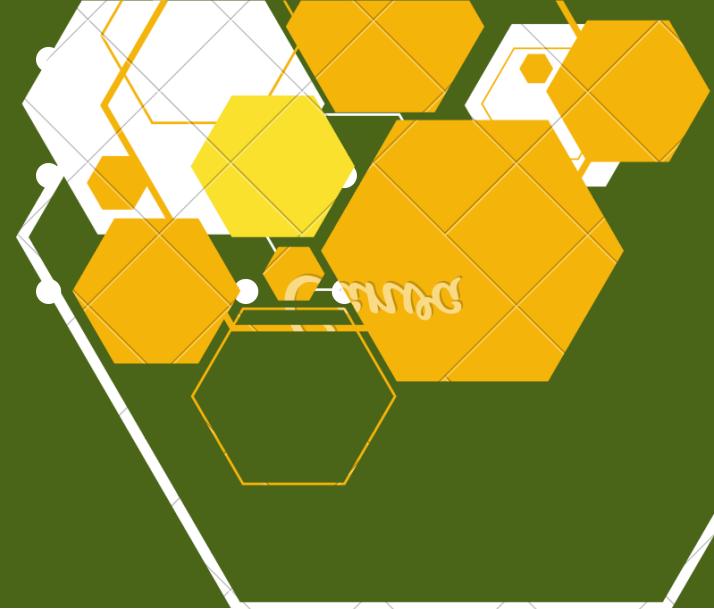
- **Réduction de dimensions via PCA**

Dimensions dataset après réduction PCA : (1050, 494)

- **Réduction de dimensions via T-SNE**

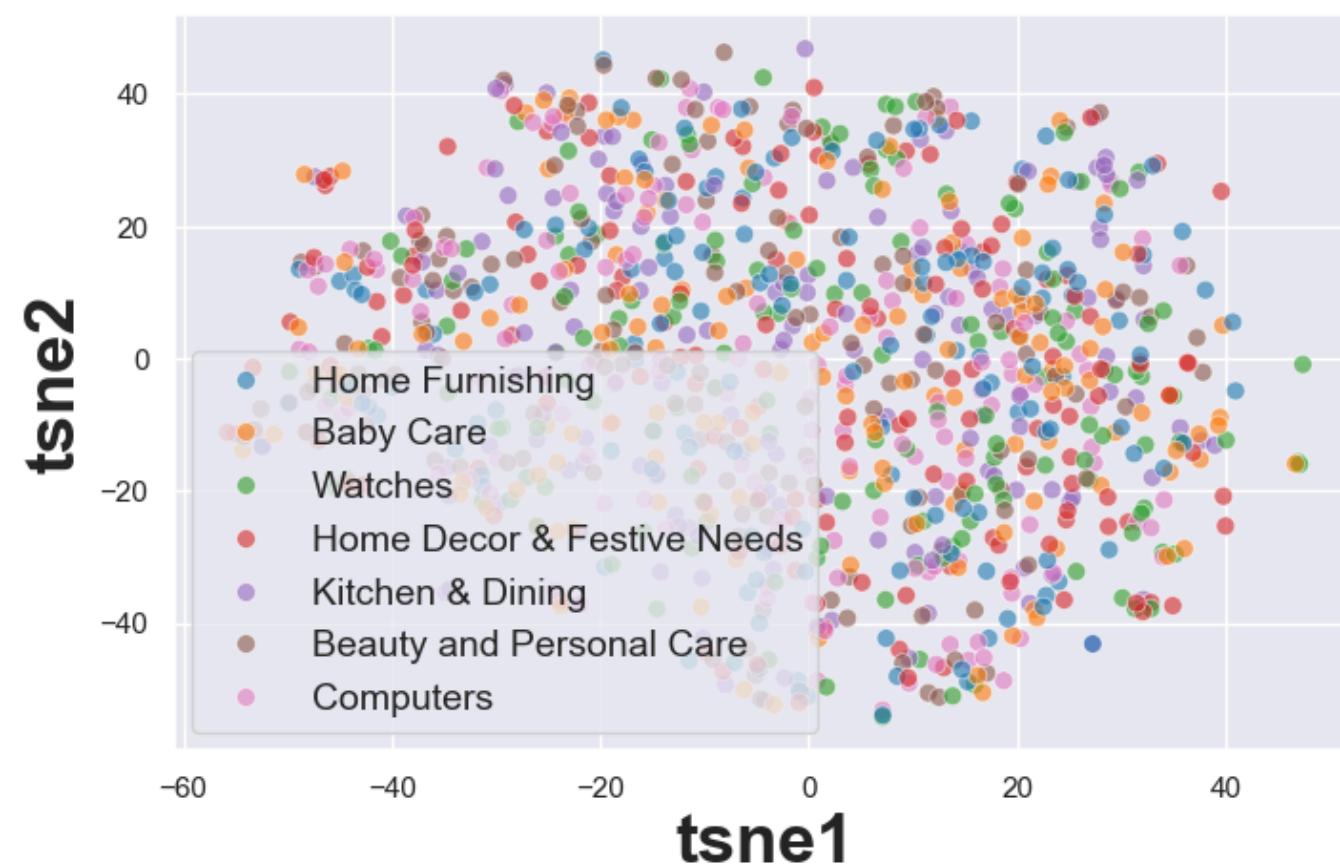
```
tsne = manifold.TSNE(n_components=2, perplexity=20, learning_rate=400,
n_iter=2000, init='random', random_state=6)
```

Faisabilité classification à partir des images



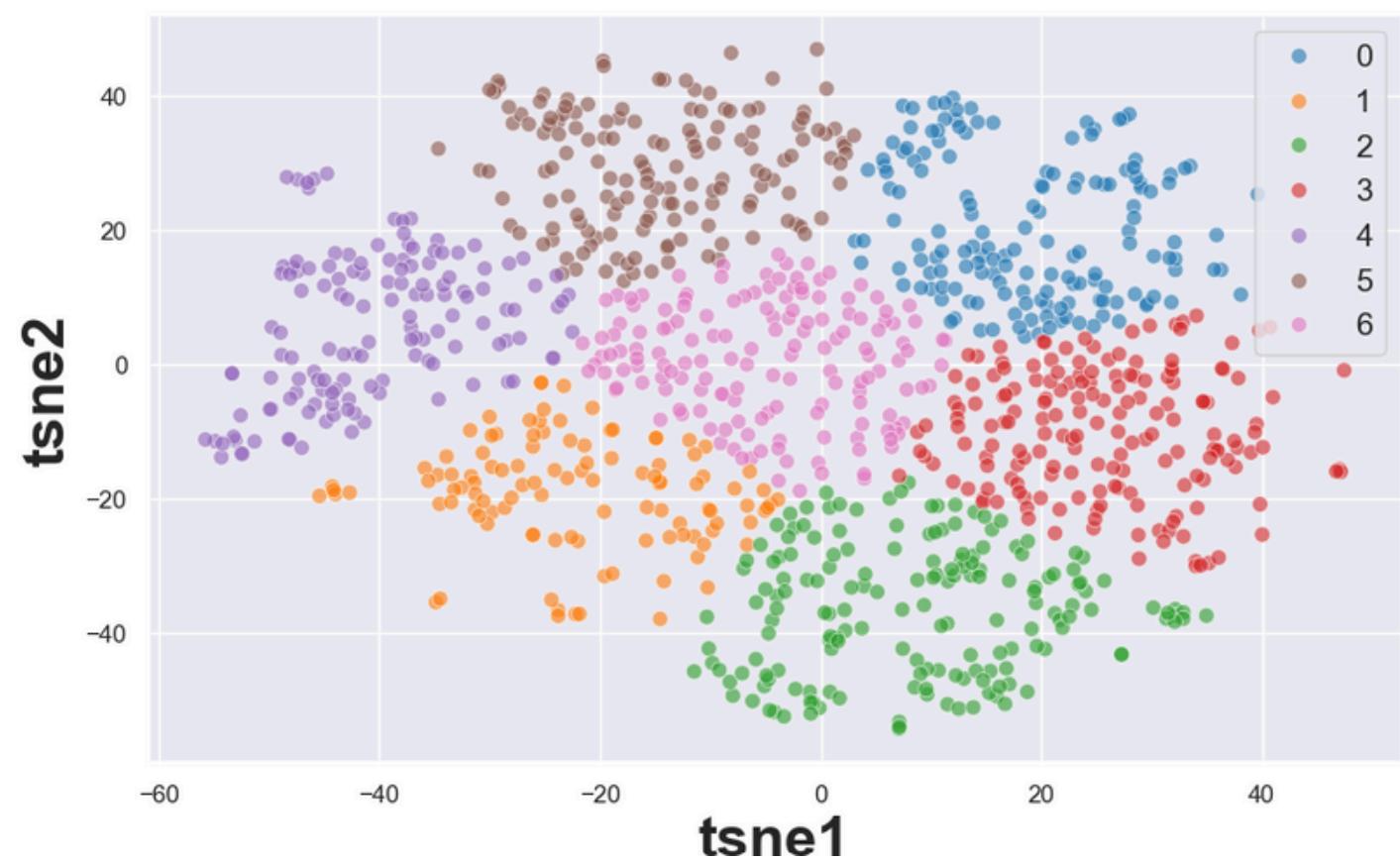
- Analyse visuelle et conclusion

TSNE selon les vraies classes



Score ARI : 0.0032

TSNE selon les clusters



Visuellement, la séparation entre les différents clusters n'est pas nette. Le très faible score ARI vient confirmer cette observation visuelle. En s'appuyant uniquement sur cette méthode d'embedding via les descripteurs SIFT, la faisabilité d'une classification des produits en fonction de leurs images n'est pas admise.

Faisabilité classification à partir des images



- **Faisabilité d'une classification automatique via VGG16**

==> Utilisation du modèle VGG 16 sans la couche de prédiction)

==> Extraction des caractéristiques principales des images

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312

Total params: 134,260,544 (512.16 MB)

Trainable params: 134,260,544 (512.16 MB)

Non-trainable params: 0 (0.00 B)

- **Identification des points clés et descripteurs associés**

Longueur embedding VGG16 : (1050, 4096)

temps de traitement VGG embedding : 489.34 secondes

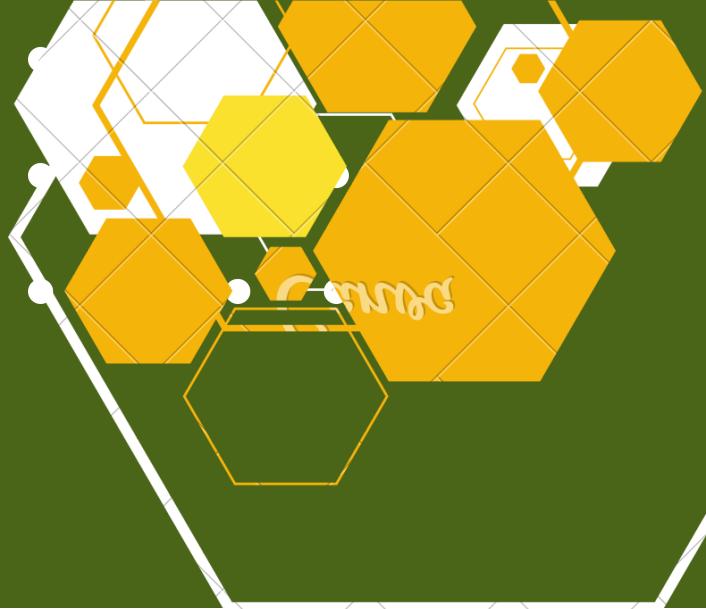
- **Réduction de dimensions via PCA**

Dimensions avant réduction PCA : (1050, 4096)

Dimensions après réduction PCA : (1050, 793)

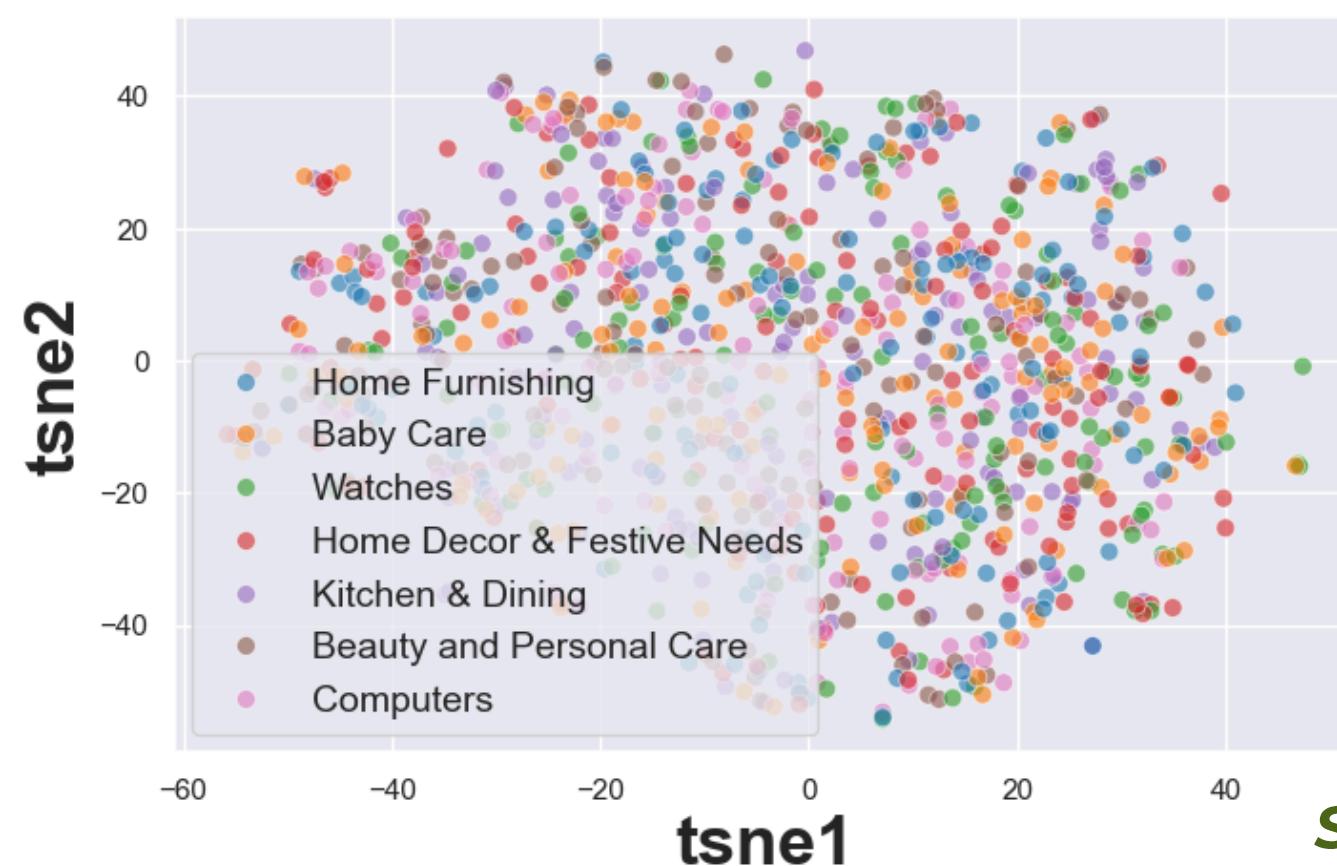
- **Réduction de dimensions via T-SNE**

Faisabilité classification à partir des images



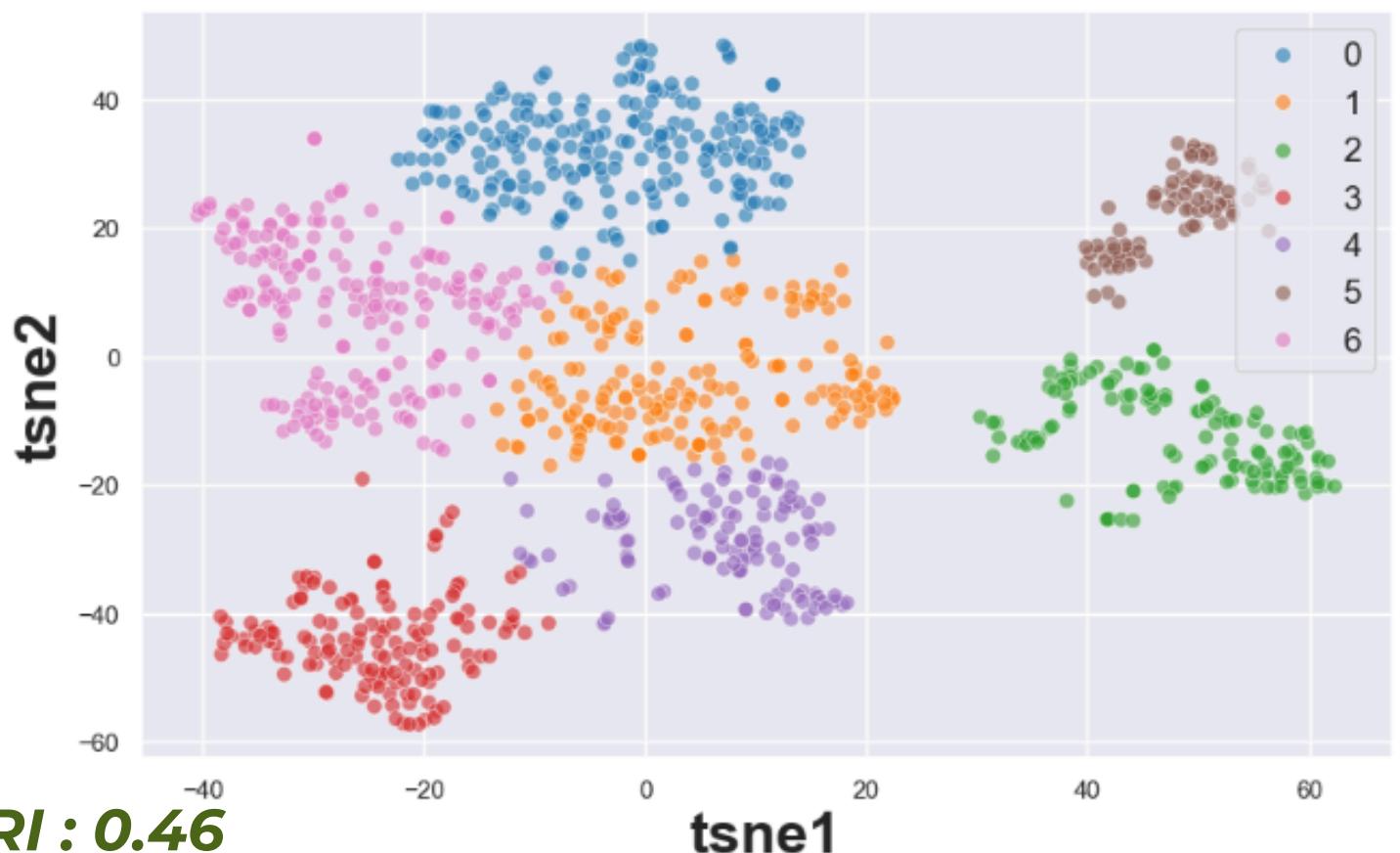
- Analyse visuelle et conclusion

TSNE selon les vraies classes



Score ARI : 0.46

TSNE selon les clusters



- L'analyse graphique montre visuellement qu'il est réalisable de 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.

Classification supervisée à partir des images



NB : Nous prenons pour cette classification l'approche ImageDatagenerator avec data augmentation

- **Initialisation d'une base convolutionnelle pré entraînée VGG 16**

```
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

- **Split des données (train 80% / validation 20%)**

Taille de l'ensemble d'entraînement : 840

Taille de l'ensemble de validation : 210

- **Préparation des générateurs de données d'entraînement**

==> normalisation des pixels

==> rotation aléatoire

==> décalage horizontal et vertical

==> Cisaillement aléatoire

==> zoom aléatoire

==> retournement horizontal

==> niveau de gris

- **Préparation des générateurs de données de validation**

==> normalisation des pixels

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total params: 14,714,688 (56.13 MB)

Trainable params: 14,714,688 (56.13 MB)

Non-trainable params: 0 (0.00 B)

Classification supervisée à partir des images



- **Fine tuning du modèle de base**

==> Nous gélons toutes les couches sauf le block5

```
Total params: 14,714,688 (56.13 MB)
```

```
Trainable params: 7,079,424 (27.01 MB)
```

```
Non-trainable params: 7,635,264 (29.13 MB)
```

==> Ajout des couches de classification

```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(7, activation='softmax'))
```

==> Compilation du modèle

```
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(learning_rate=2e-5),
              metrics=['accuracy'])
```

- **==> Synthèse du modèle obtenu**

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6,422,784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 7)	1,799

Total params: 21,139,271 (80.64 MB)

Trainable params: 13,504,007 (51.51 MB)

Non-trainable params: 7,635,264 (29.13 MB)

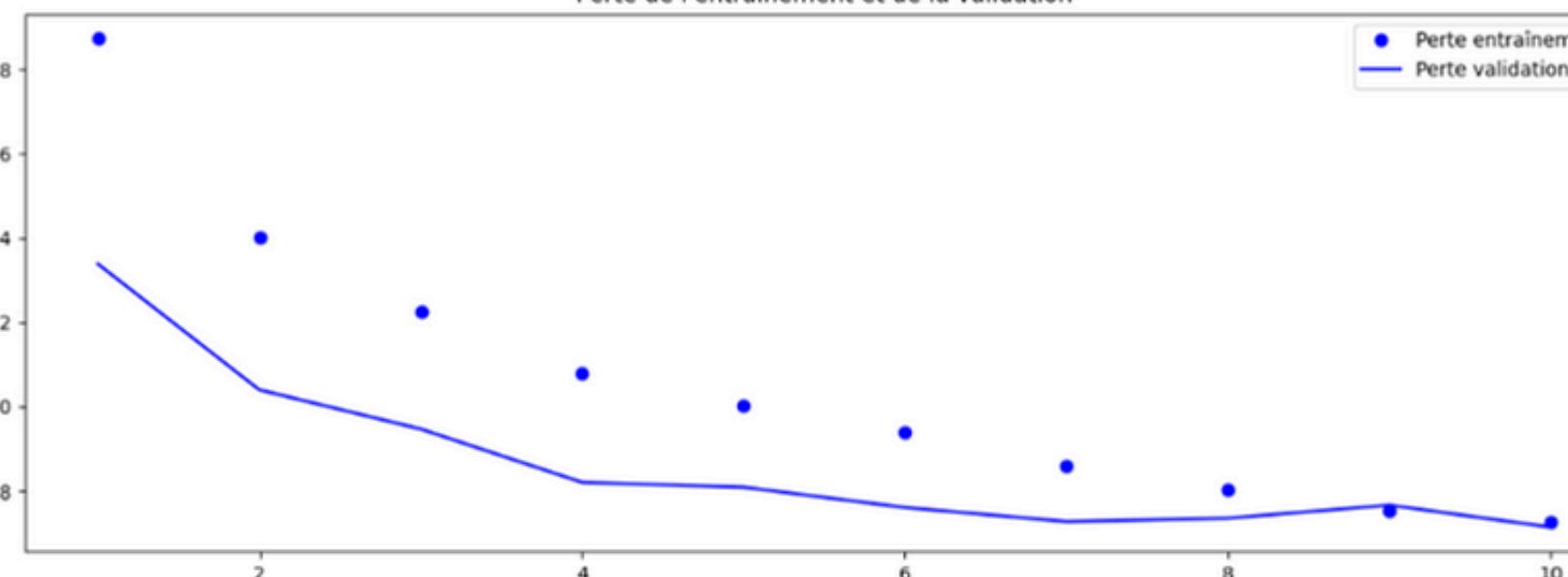
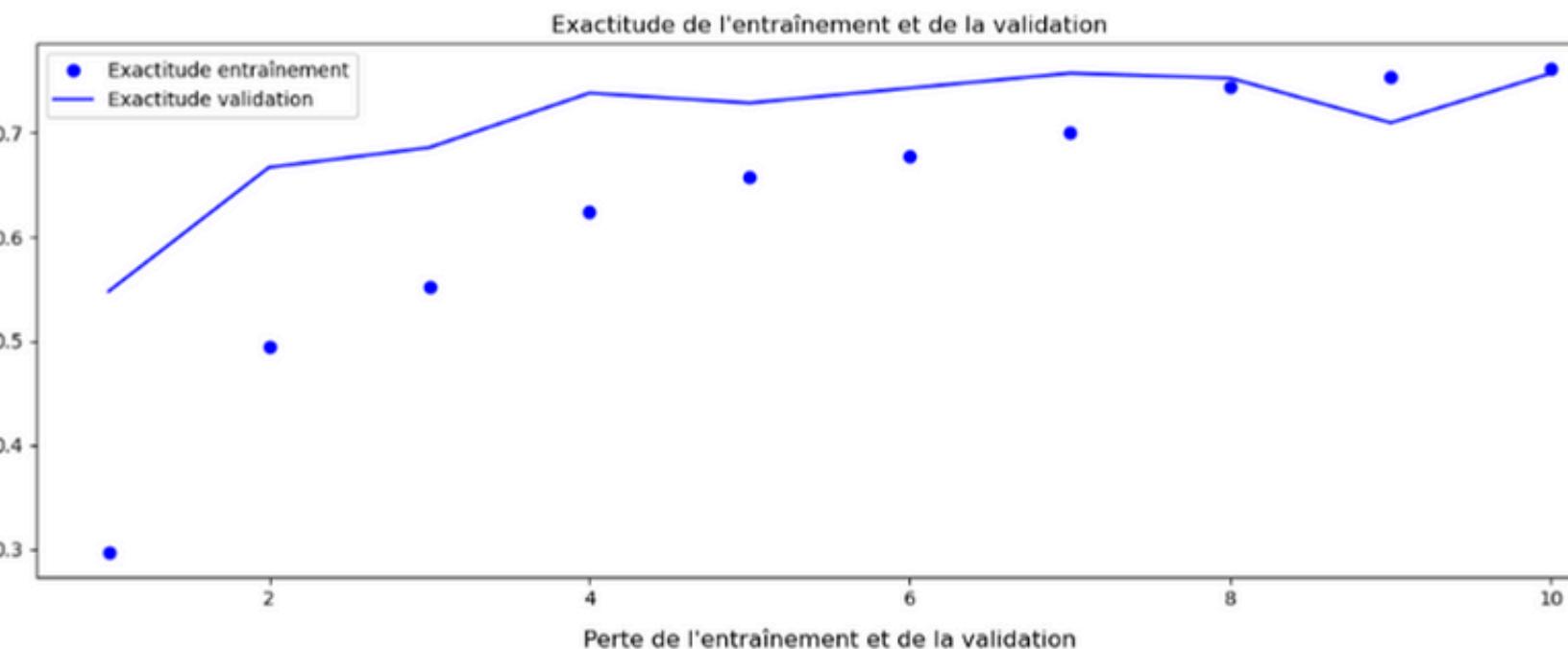
- **Entraînement du modèle**

```
history = model.fit(
    train_generator,
    steps_per_epoch=50,
    epochs=10,
    validation_data=validation_generator)
```

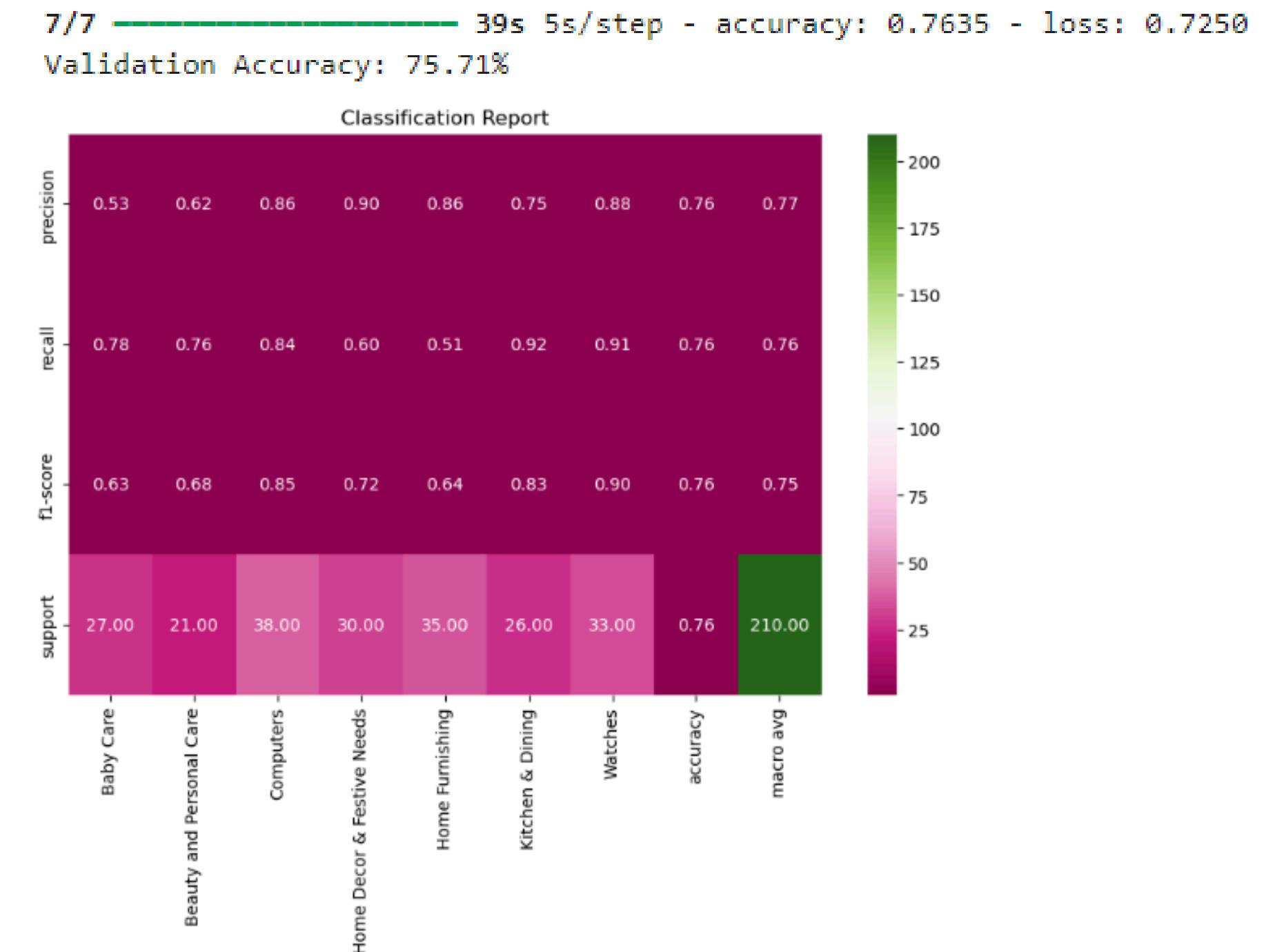
Classification supervisée à partir des images



- Evaluation du modèle**



- Evaluation sur l'ensemble de validation**



Classification supervisée à partir des images



- **Fine tuning d'un deuxième modèle**

==> Nous gêlons toutes les couches du modèle de base VGG 16

```
Total params: 14,714,688 (56.13 MB)
```

```
Trainable params: 0 (0.00 B)
```

```
Non-trainable params: 14,714,688 (56.13 MB)
```

==> Ajout des couches de classification

```
model1 = models.Sequential()
model1.add(conv_base)
model1.add(layers.Flatten())
model1.add(layers.Dense(256, activation='relu'))
model1.add(layers.Dropout(0.5))
model1.add(layers.Dense(64, activation='relu'))
model1.add(layers.Dense(64, activation='relu'))
model1.add(layers.Dense(7, activation='softmax'))
```

==> Compilation du modèle

```
model1.compile(loss='categorical_crossentropy',
                optimizer=optimizers.RMSprop(learning_rate=2e-5),
                metrics=['accuracy'])
```

==> Synthèse du modèle obtenu

Model: "sequential_1"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 256)	6,422,784
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 64)	16,448
dense_4 (Dense)	(None, 64)	4,160
dense_5 (Dense)	(None, 7)	455

```
Total params: 21,158,535 (80.71 MB)
```

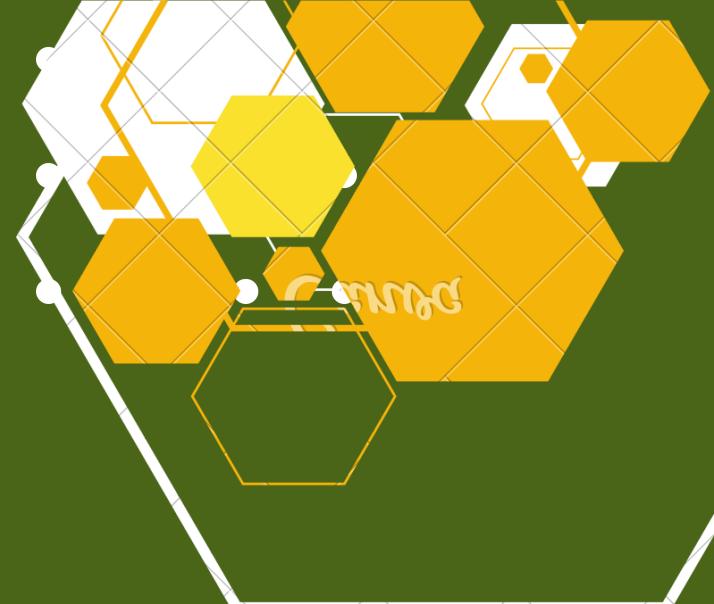
```
Trainable params: 13,523,271 (51.59 MB)
```

```
Non-trainable params: 7,635,264 (29.13 MB)
```

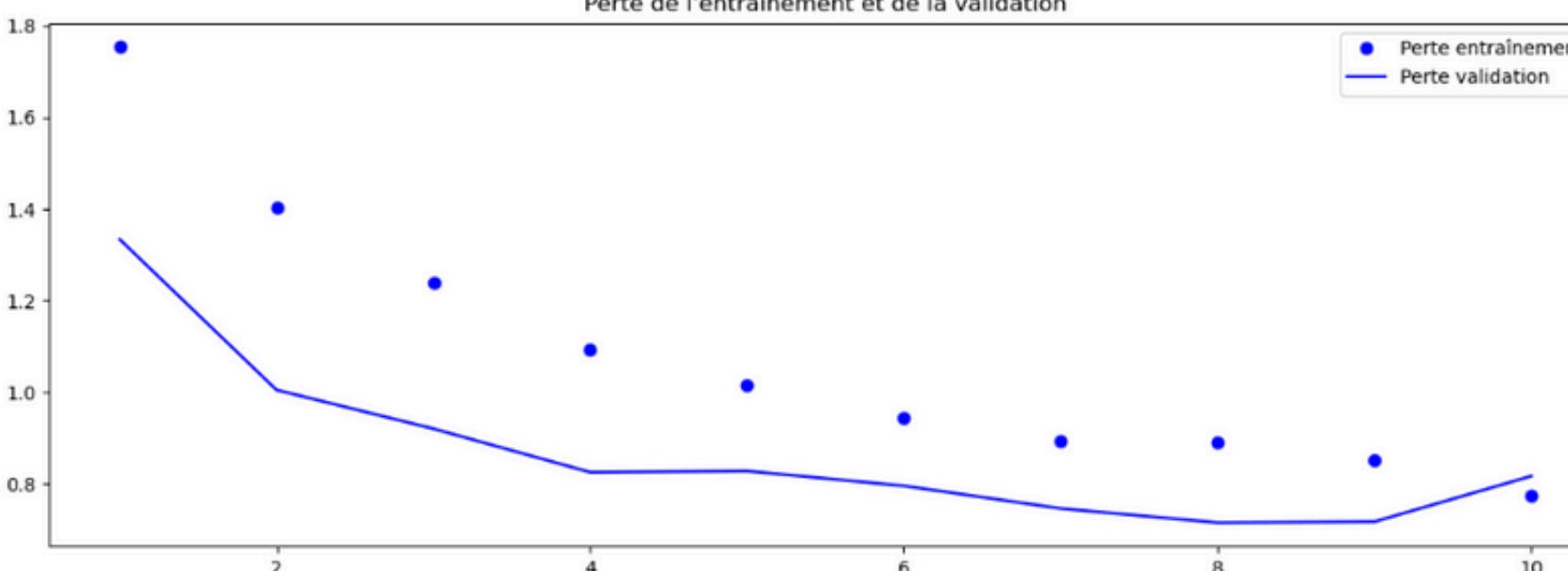
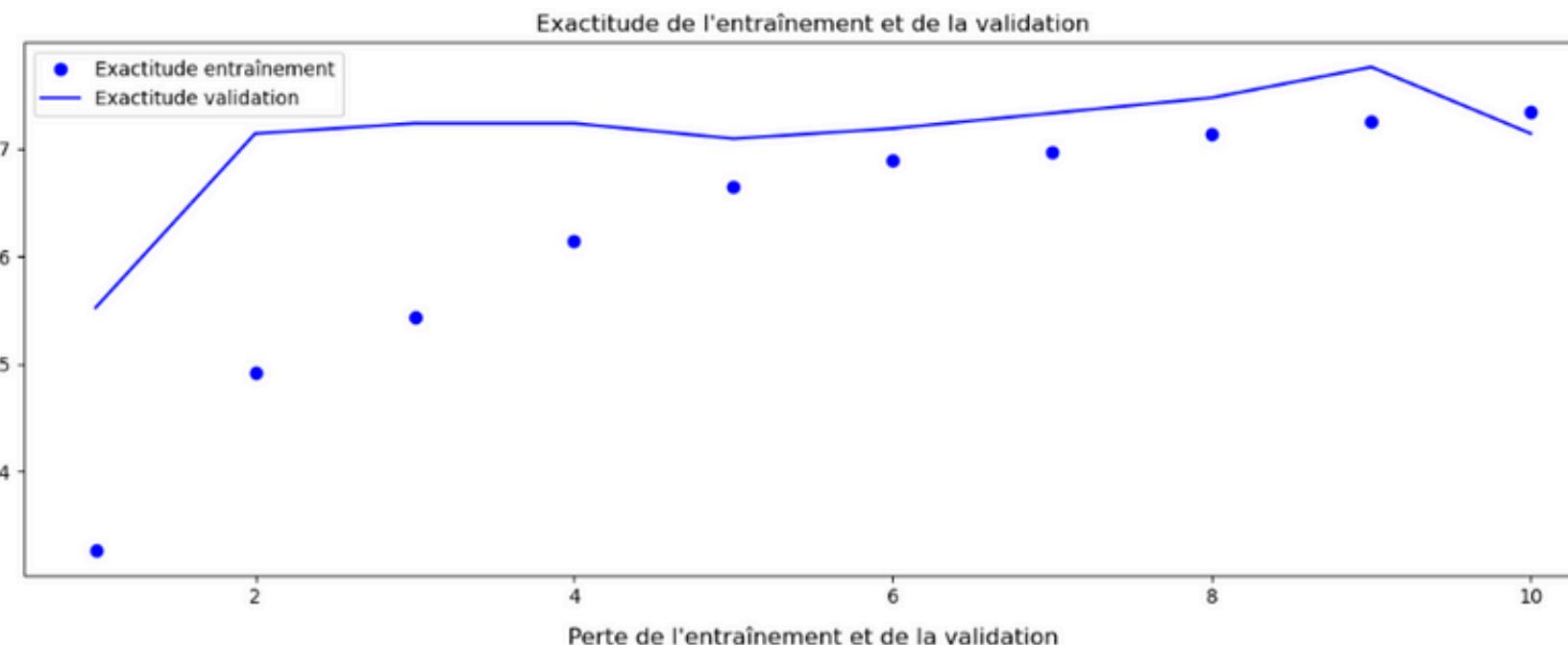
- **Entraînement du modèle**

```
history = model1.fit(
    train_generator,
    steps_per_epoch=50,
    epochs=10,
    validation_data=validation_generator)
```

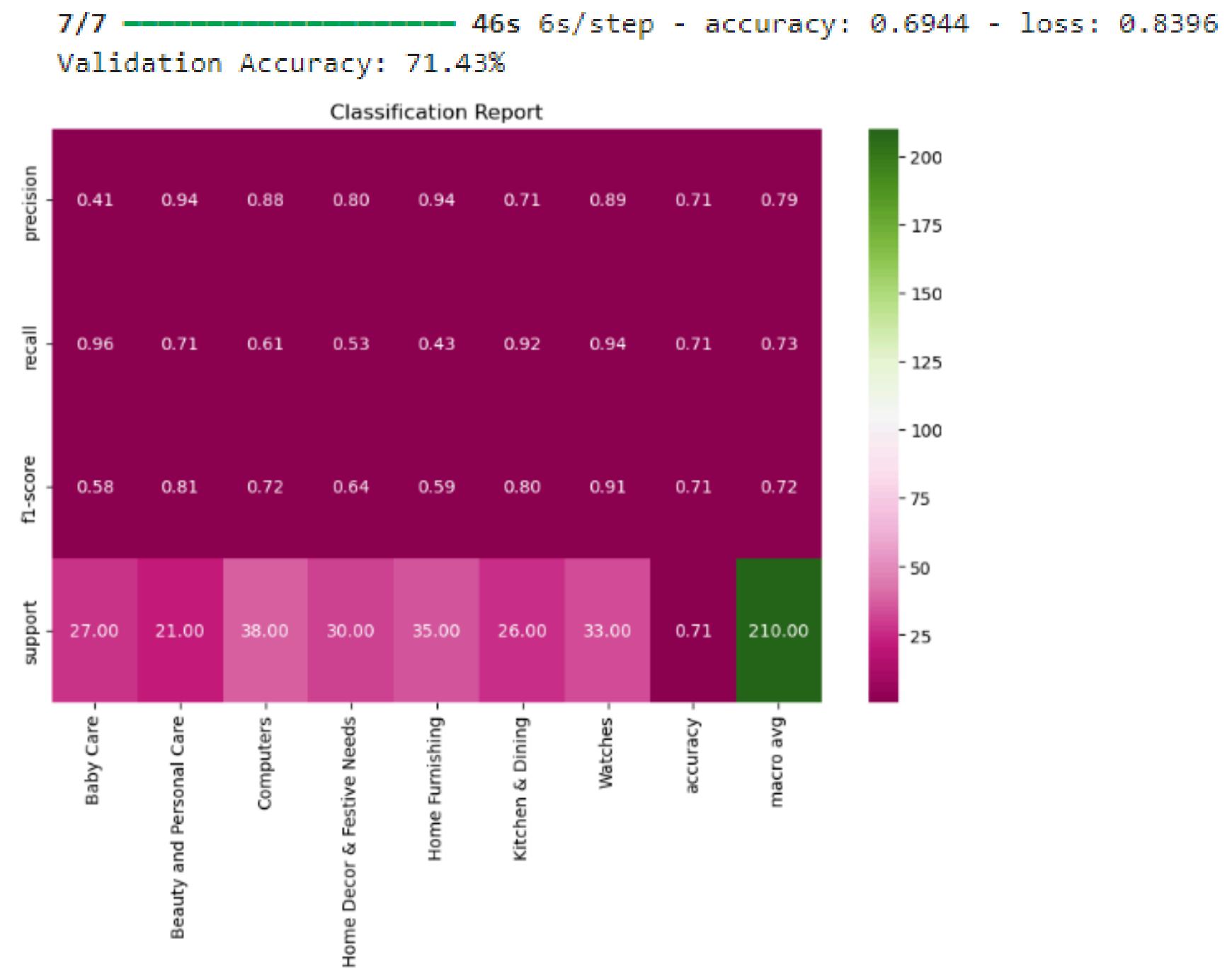
Classification supervisée à partir des images



- Evaluation du modèle**



- Evaluation sur l'ensemble de validation**





- **Accès à l'API Edamam**

==> obtention d'un identifiant **app_id** et d'une clé **app_key**

- **Construction des requêtes**

==> indication de l'URL dédié pour les aliments

<https://api.edamam.com/api/food-database/v2/parser>

==> filtre sur les ingrédients “**ingr**”

==> construction de la liste des résultats sur les seuls champs nécessaires

- **Résultat des produits à base de champagne**

Nombre total de produits à base de champagne trouvés : 4

ID: food_9886ek2e9dqb2ad1am0tbe1hdss, Label: Champagne, Category: Generic foods, Contents: , Image: <https://www.edamam.com/food-img/a71/a718cf3c52ad0522128929f1f324d2ab.jpg>

ID: food_bncple4a2uagu1b4ho72buz2vs, Label: Champagne Grape, Category: Generic foods, Contents: , Image: <https://www.edamam.com/food-img/ca5/ca55ac74deb991d159942c65777115df.jpg>

ID: food_bxdqxxka2vgmpbv8e4ygb4zfnkn, Label: Champagne Vinegar, Category: Generic foods, Contents: , Image: <https://www.edamam.com/food-img/5f6/5f69b84c399d778c4728e9ab4f8065a2.jpg>

ID: food_hrtdd7beak7p11lastchh1aduyz1q, Label: Sherry or Champagne Vinegar, Category: Generic foods, Contents: , Image: <https://www.edamam.com/food-img/5f6/5f69b84c399d778c4728e9ab4f8065a2.jpg>

- **Résultat des 10 premiers produits ayant les champs (foodId, label, category, foodContentsLabel, image)**

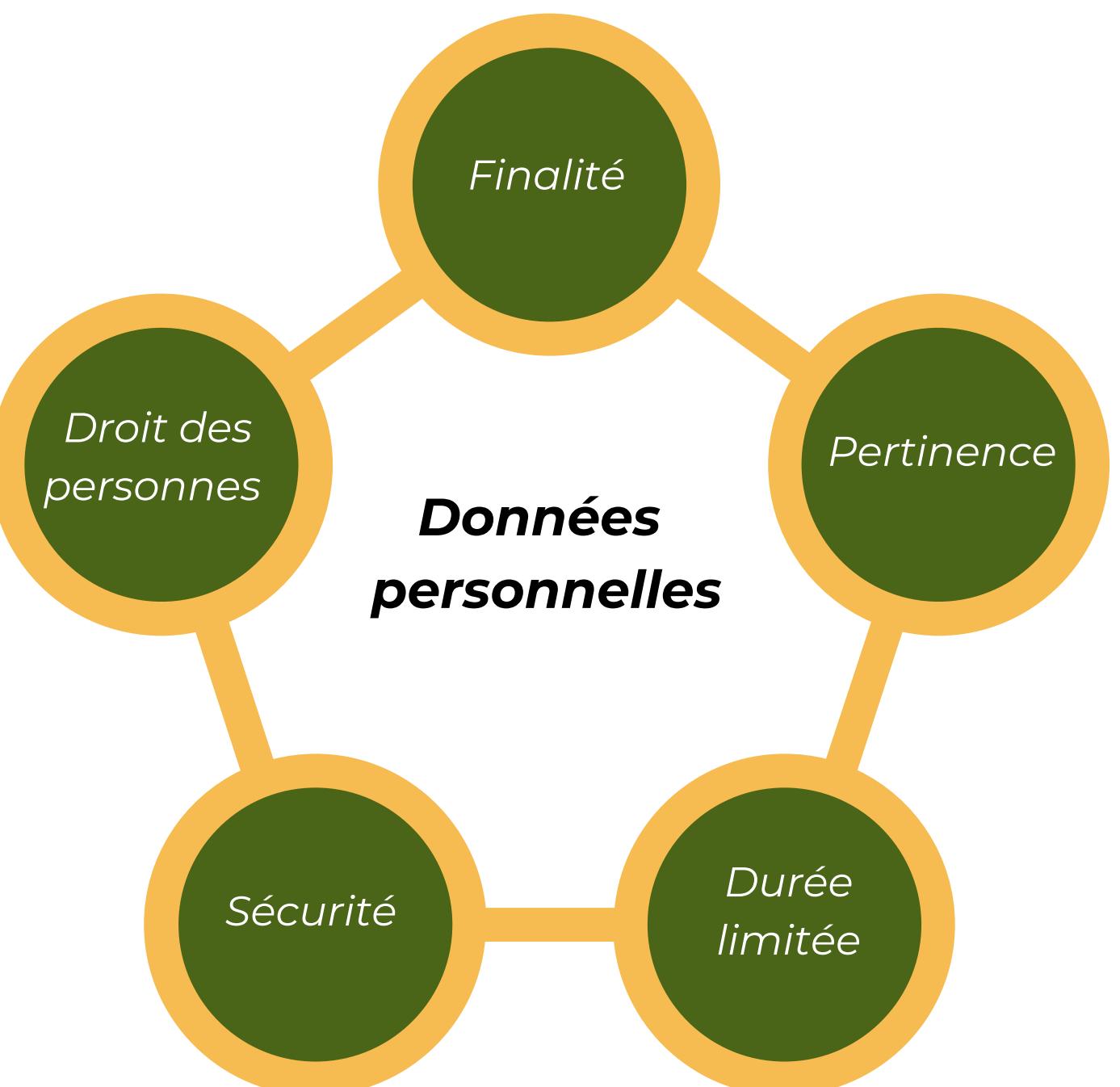
==> le fichier.csv **first_10_food_products.csv** des résultats de cette requête a été fourni



Le RGPD ou Règlement Général sur la Protection des Données

- Les données personnelles appartiennent à chaque citoyen. Chaque citoyen doit pouvoir exercer un contrôle.
- Les organisations ne sont que des dépositaires temporaires des données.
- Des responsables doivent être désignés pour l'utilisation des données, leur sécurité et leur confidentialité.

Principes fondamentaux du RGPD



Le RGPD : Comment avons nous respecté les principes du RGPD ?

La sécurité : connexion par identification et clé d'accès

Pertinence / Finalité : collecte des seules données utiles pour le projet

Durée limité : Les données collectées ne seront conservées que pendant la durée du projet.

20/20

Merci

