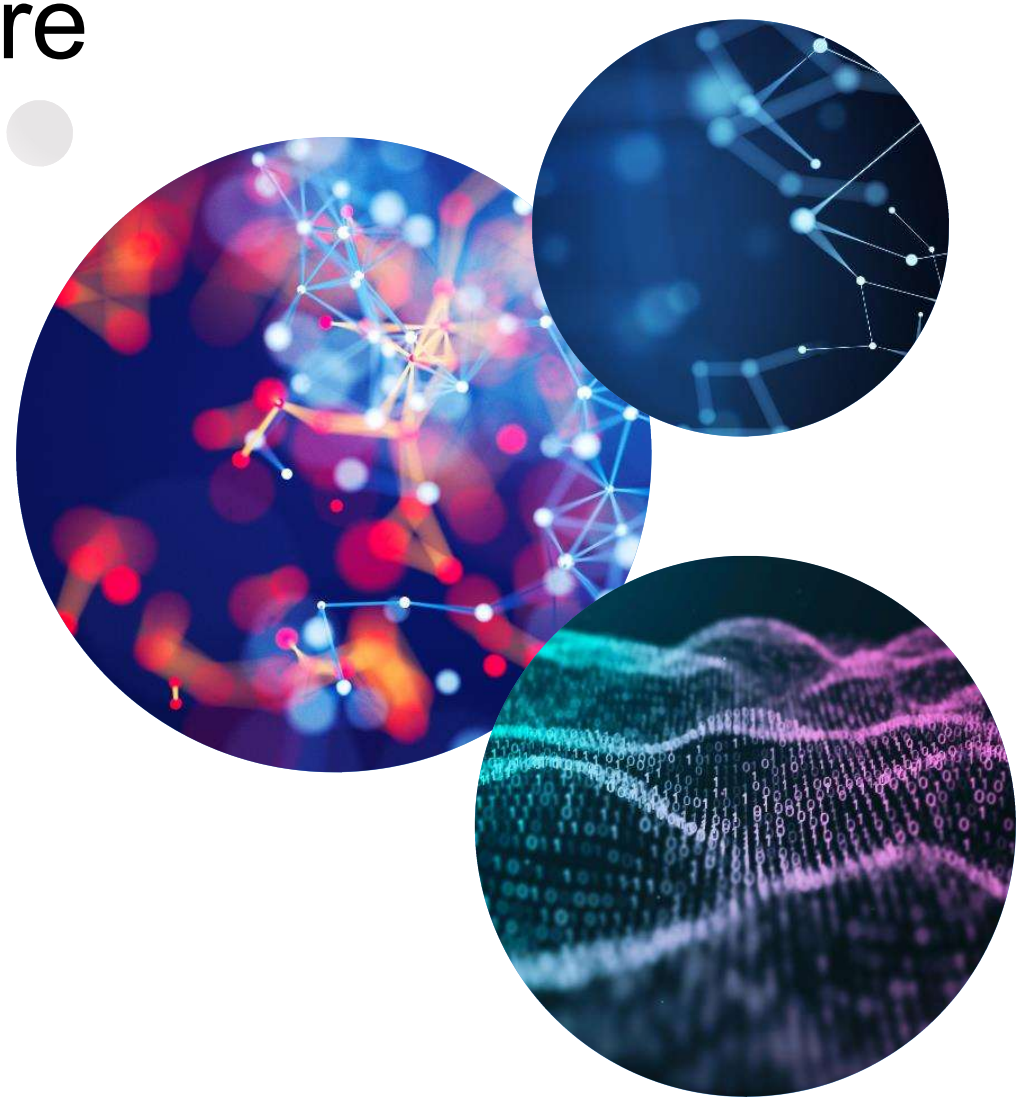
Classifiez automatiquement des biens de consommation

Armand FAUGERE LinkedIn
armand-faugere@live.fr

Sommaire



I) Cadrage du projet et données d'entrée

I) Cadrage du projet et données d'entrée

❑ **Contexte :**

- Projet de lancement d'un site de E-commerce pour la société « Place de marché ».

Attribution de la catégorie des articles manuelle par les vendeurs

➔ peu fiable, pas adapté à un volume d'articles important

❑ **But :**

- Etudier la faisabilité d'un moteur de classification des articles par catégories, à partir du texte et de l'image

❑ **Objectifs :**

- Etudier la faisabilité d'un moteur de classification avec le texte
- Etudier la faisabilité d'un moteur de classification avec les images
- Réaliser une classification supervisée à partir des images
- Tester la collecte de produits via une API

❑ **Le jeu de données ➔ 1050 produits**

- flipkart_com-ecommerce_sample_1050.csv

- 1050 images au format jpg

❑ **7 catégories avec répartition homogène :**

- Home Furnishing
- Baby Care
- Watches
- Home Decor & Festive Needs
- Kitchen & Dining
- Beauty and Personal Care
- Computers

❑ **Principes de protection des données** (finalité, proportionnalité et pertinence, durée de conservation limitée, sécurité et confidentialité, droits des personnes) www.cnil.fr

II) Faisabilité classification données textuelles

2) Faisabilité classification données textuelles

1 Exploration Data Analysis

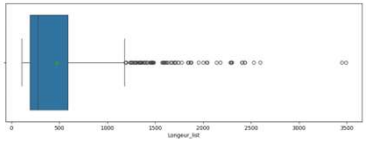
Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, re, nltk

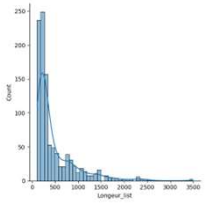
Sélection des colonnes pertinentes → Extraction des catégories → Analyse de la partie corpus

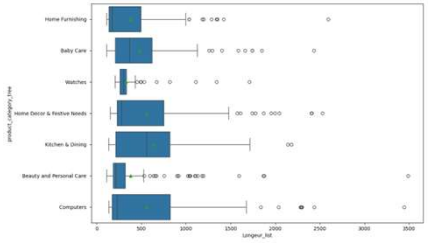
- ❑ Analyse info sur datset (nb de lignes, colonnes…)
- ❑ Analyse valeurs nulles et doublons
- ❑ Sélection : ‘product_name’, 'image’, 'product_category_tree’, 'description'

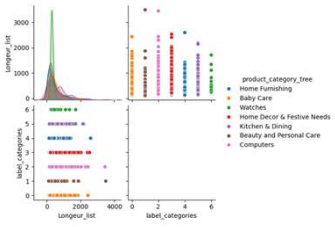
- ❑ Fonctions lambda de traitement ➔ *product_category_tree*
- ❑ Vérification distribution
- ❑ Verification valeurs nulles et doublons

- ❑ Création d’une colonne longueur_List
- ➔(min, max, mean, std)
- ➔ box plot & distribution au global
- ➔ box plot catégories
- ➔ pair_plot & corrélations (longueur list et catégories)









2) Faisabilité classification données textuelles

2 Text Processing

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, re, nltk, worldcloud, gensim, keras, tensorflow

Nettoyage et Normalisation → Approche Bags of Words → Approche word/sentence embedding → **Conclusion Faisabilité ARI**

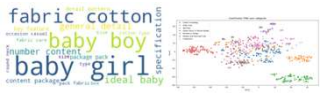
PCA ➔ TSNE ➔ Clusters ➔ ARI

❑ Préparation de la fonction :
➔ Nettoyage et Normalisation (lower, tokenizer, stop_words, alpha, lemmatizer, english, mots spécifiques)

❑ Application de la fonction

❑ Fréquences des mots
➔ WorldCloud par catégories
➔ CountVectorizer

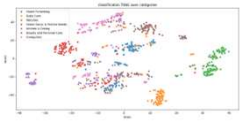
❑ TF-IDF des mots
➔ TfidfVectorizer

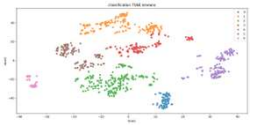


❑ WORD2VEC cbow
➔ Modèle + table vocabulaire + training (vecteurs + clefs)
➔ Matrice embedding
➔ Modèle embedding (Input, embedding, features)

❑ BERT Hugging Face
➔ Préparation des sentences
➔ Modification dernière couche réseau de neurone
➔ features

❑ USE
➔ Création du modèle
➔ features





2) Faisabilité classification données textuelles

Approche Bags of Words

bow_baby_care

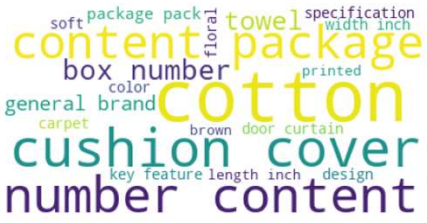
bow_Beauty_and_Personal_Care

bow_Home_Decor_Festive_Needs

WordCloud par catégories (20 mots plus fréquents)

bow_Computers

bow_Home_Furnishing



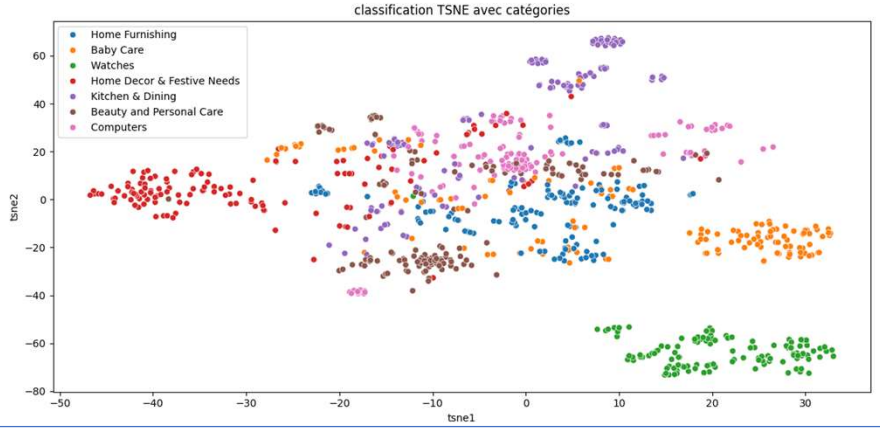
bow_Kitchen_Dining

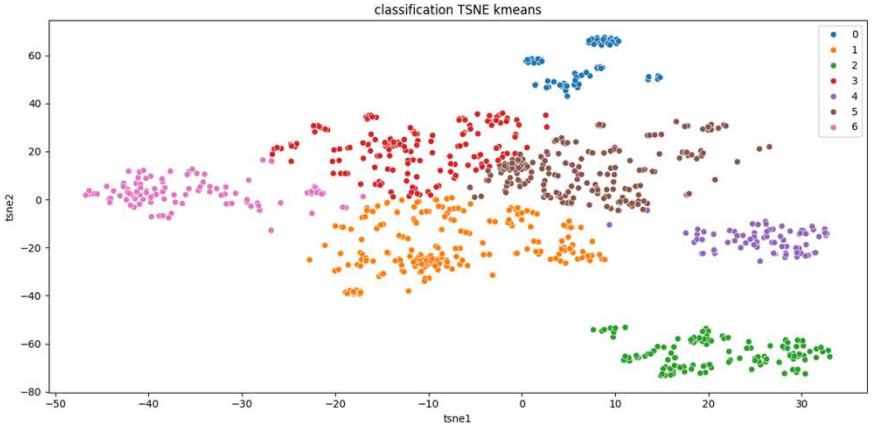
bow_Watches

2) Faisabilité classification données textuelles

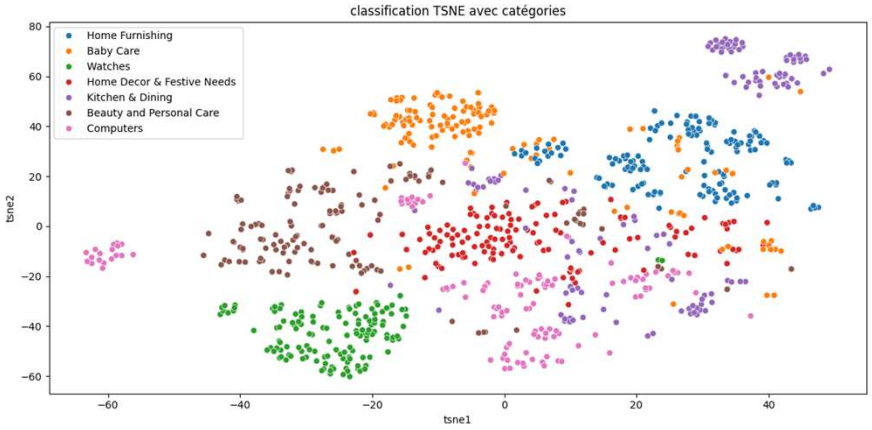
Approche Bags of Words

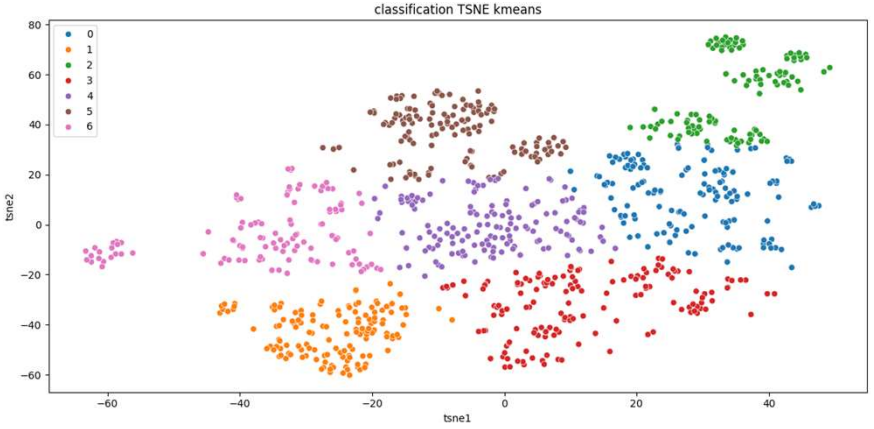
Fréquence
➔ ARI = 0,366





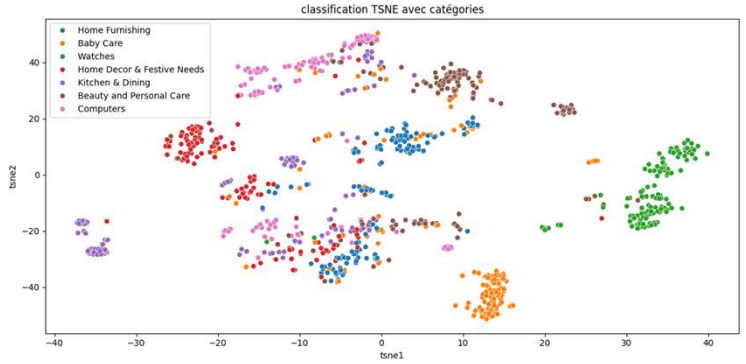
Tf-idf
➔ ARI = 0,477

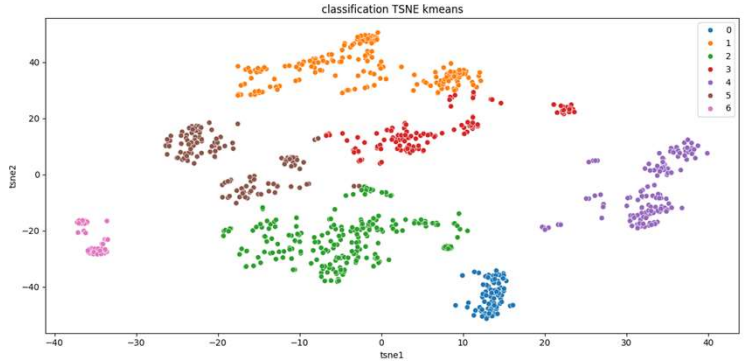




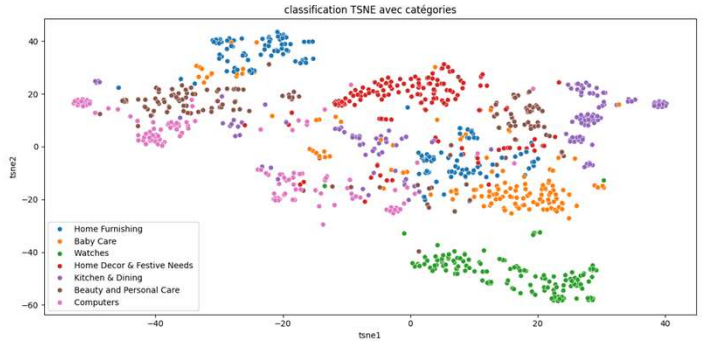
2) Faisabilité classification données textuelles

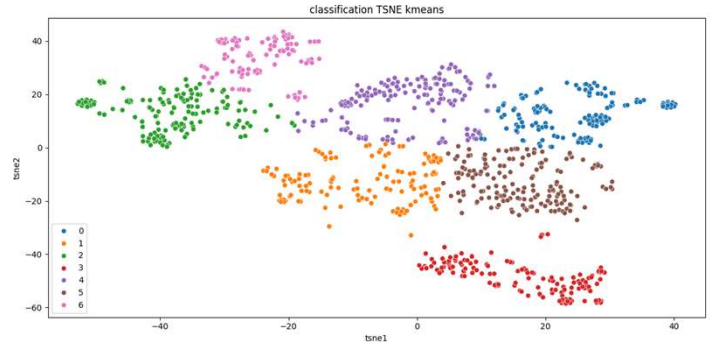
Approche word/sentence embedding



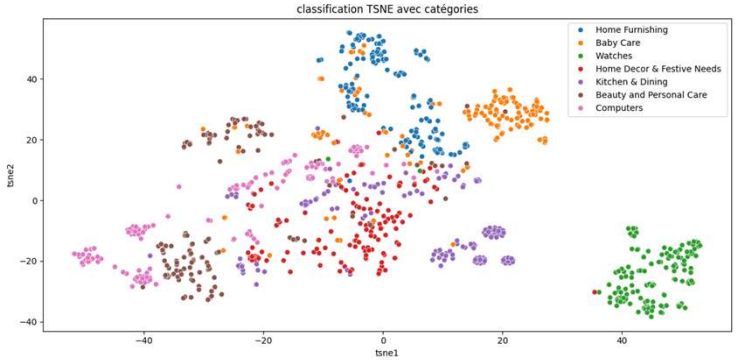


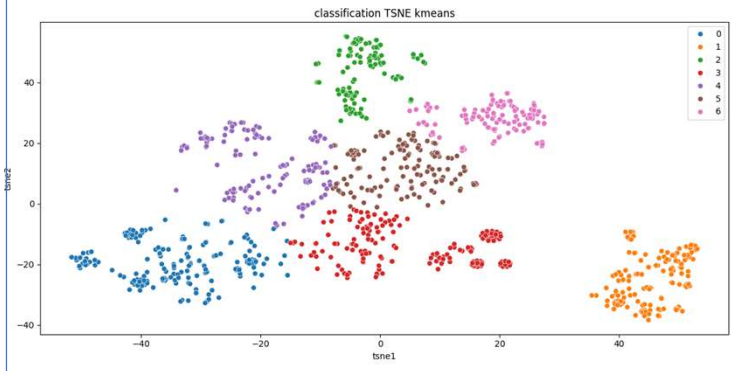
Word2vec ➔ ARI = 0,338





BERT ➔ ARI = 0,401





USE ➔ ARI = 0,383

2) Faisabilité classification données textuelles

Conclusion Faisabilité

Classement	Méthode	SCORE ARI
1	TF-IDF	0,477
2	BERT	0,401
3	USE	0,383
4	Fréquence	0,366
5	Word2vec	0,338

Les scores ARI sont largement supérieurs à 0

➔ La faisabilité de classification par les données textuelles est démontrée

III) Faisabilité classification données images

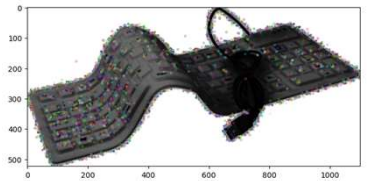
3) Faisabilité classification données images

3 Image Processing SIFT

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

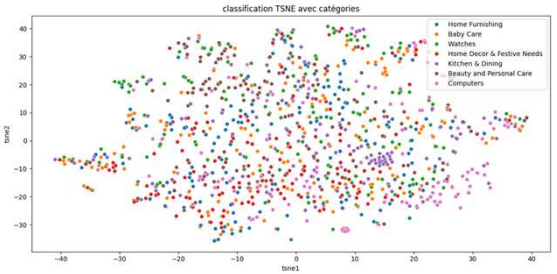
Descripteurs SIFT → Features images → **Conclusion Faisabilité ARI**

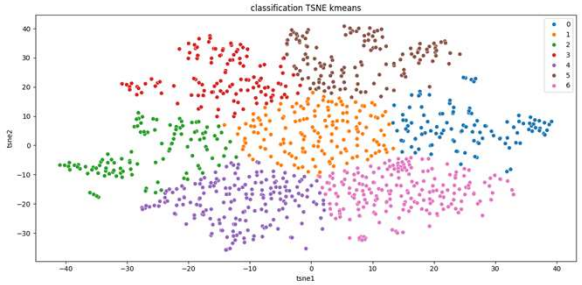
- ❑ Dataframe focus image
- ❑ Fonction d'affichage des images par catégories
- ❑ Descripteurs SIFT
- ❑ Descripteurs SIFT par image



PCA ➔ TSNE ➔ Clusters ➔ ARI

- ❑ Clusters de descripteurs ➔ cluster.MiniBatchKmeans (k = 325)
- ❑ Création des features images





3) Faisabilité classification données images

4 Image Processing CNN transfert learning

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

Création du modèle pré-entrainé → Features images → **Conclusion Faisabilité ARI**

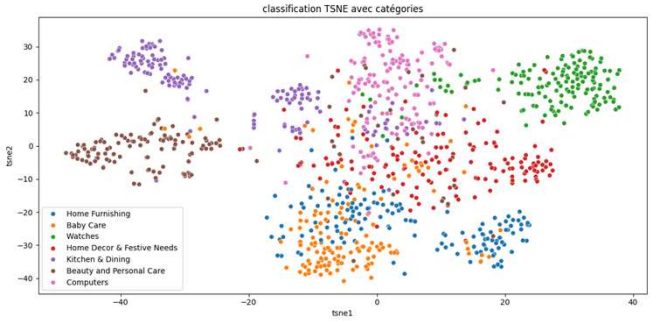
PCA ➔ TSNE ➔ Clusters ➔ ARI

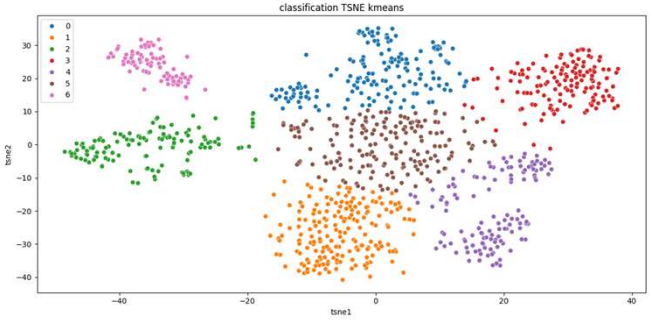
- ❑ Importation VGG16
  ➔ Suppression couche de sortie

- ❑ Préparation des images pour rentrer dans le modèle
- ❑ Prédictions du modèle

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312

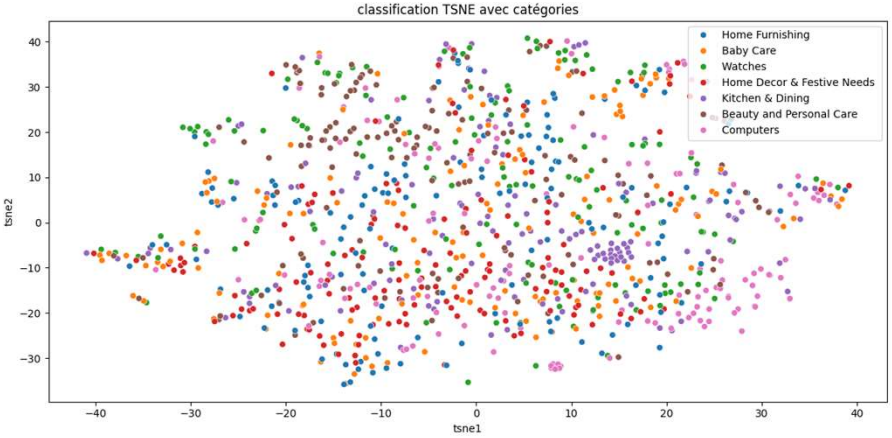
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0

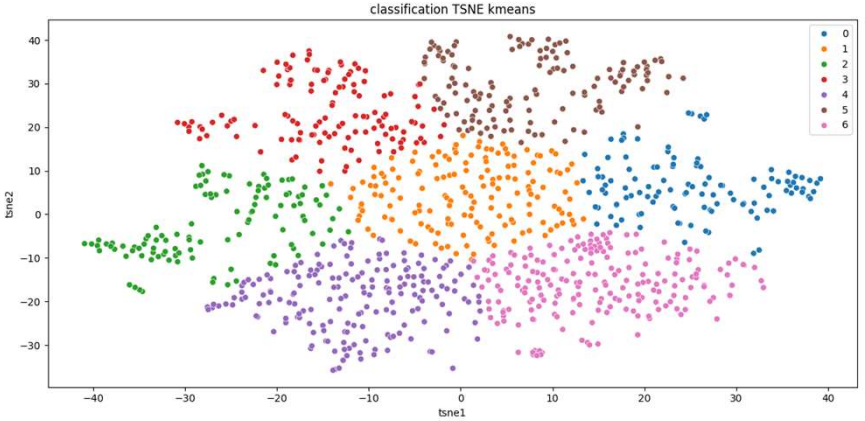




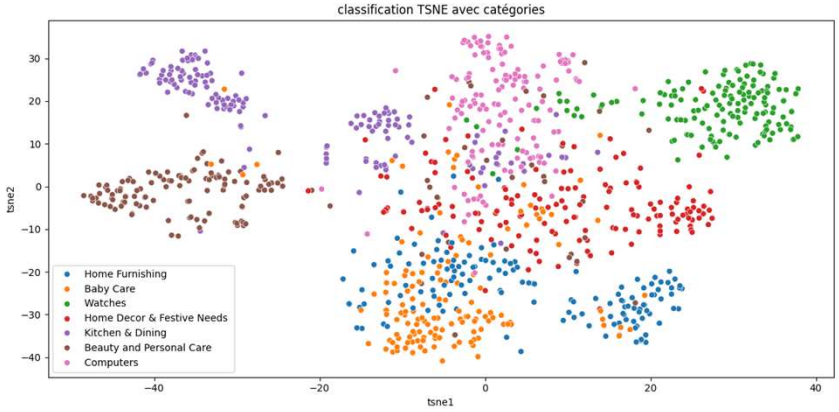
3) Faisabilité classification données images

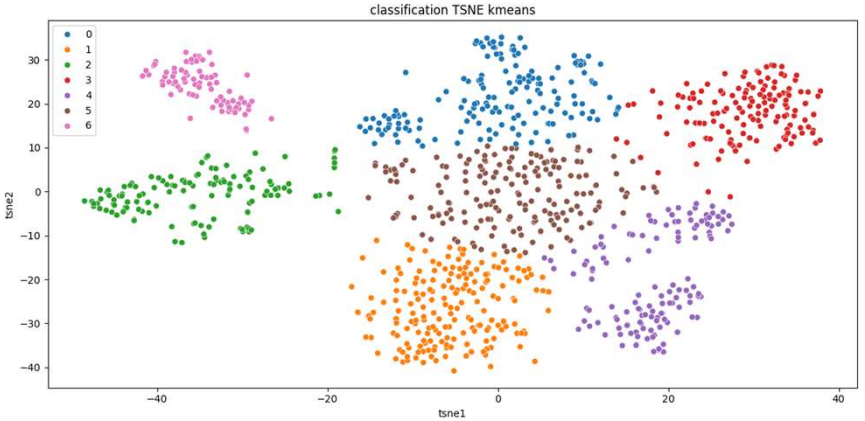
SIFT
➔ ARI = 0,04





CNN transfert Learning
➔ ARI = 0,449





3) Faisabilité classification données images

Conclusion Faisabilité

Classement	Méthode	SCORE ARI
1	CNN Transfert learning	0,449
2	SIFT	0,04

Le score ARI est largement supérieur à 0 pour le CNN Transfert learning

➔ La faisabilité de classification par images avec une approche par réseau de neurones est démontrée

IV) Classification supervisée

4) Classification supervisée

5 Classification supervisée

VGG16 (sans data augmentation)

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

Préparation des images

- ❑ Fonction de préparation images
  - ➔ Boucle pour chaque image
  - ➔ Chargement et transformation pour entrer dans VGG16 (array, reshape, preprocess)

Création du modèle

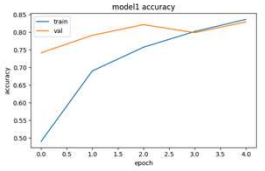
- ❑ Fonction de création de modèle
  - ➔ Récupération VGG16
  - ➔ Adaptation (global averagepooling2D, dense, dropout, dense (7))
- ❑ Création callback (save best model itération)

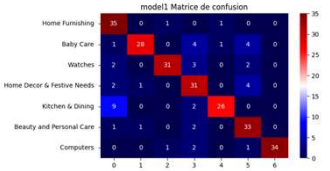
Entrainement & Test

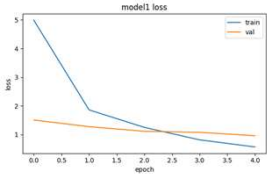
- ❑ Entrainement du modèle
  - ➔ epoch = 5
  - ➔ batch_size = 64
  - ➔ callbacks = callbacks_list

Evaluation

- ❑ Accuracy
- ❑ Loss
- ❑ Confusion matrix

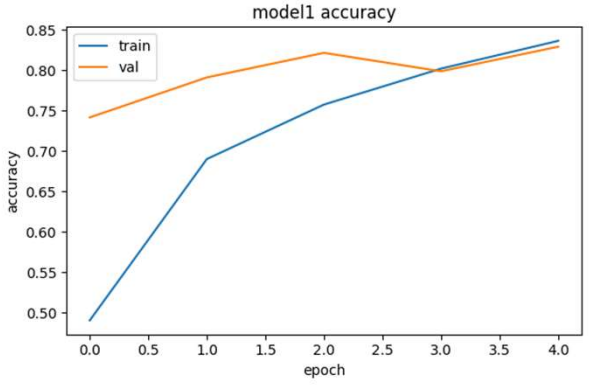


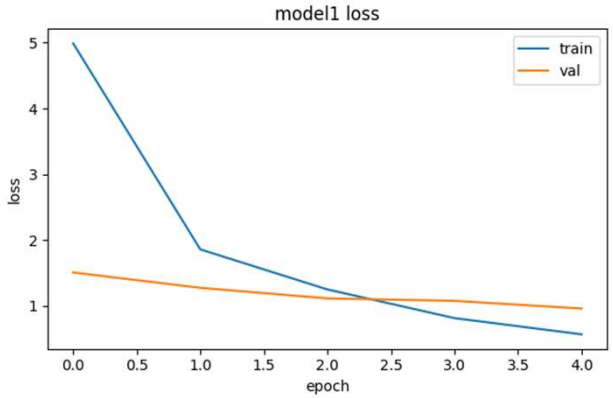




4) Classification supervisée

VGG16 sans data augmentation



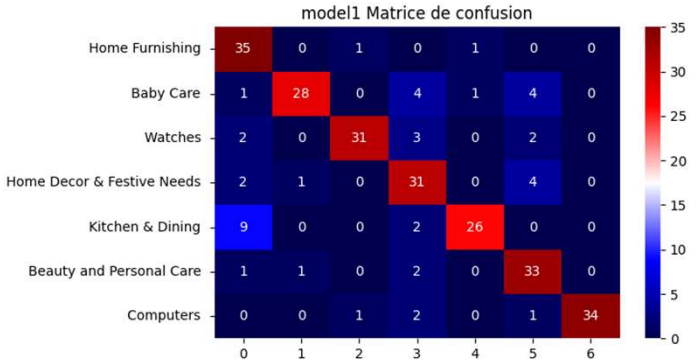


- L'accuracy s'améliore au fur et à mesure des epochs et l'erreur diminue

- Score best epoch :

Accuracy train ➔ **0,94**
Accuracy validation ➔ **0,83**

- Temps total = 683 s
- Temps best epoch = 135 s



- La catégorie Kitchen & Dining est moins bien prédite

- La catégorie computers est très bien prédite

4) Classification supervisée

6 Classification supervisée

VGG16 (avec data augmentation dans modèle)

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

Préparation des images

- ❑ Fonction de préparation images
  - ➔ Boucle pour chaque image
  - ➔ Chargement et transformation pour entrer dans VGG16 (array, reshape, preprocess)

Création du modèle + data augmentation

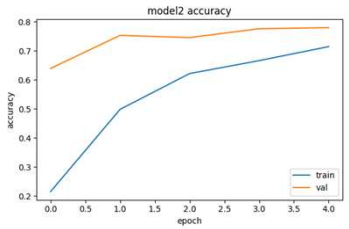
- ❑ Fonction de création de modèle
  - ➔ Récupération VGG16
  - ➔ Adaptation (global averagepooling2D, dense, dropout, dense (7))
  - ➔ Data augmentation (randomflip, randomrotation, randomzoom, rescaling)
- ❑ Création callback (save best model itération)

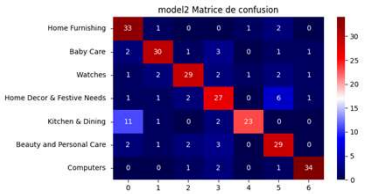
Entrainement & Test

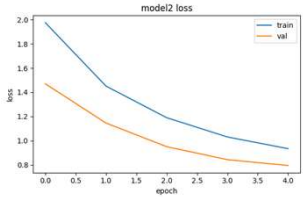
- ❑ Entrainement du modèle
  - ➔ epoch = 5
  - ➔ batch_size = 64
  - ➔ callbacks = callbacks_list

Evaluation

- ❑ Accuracy
- ❑ Loss
- ❑ Confusion matrix

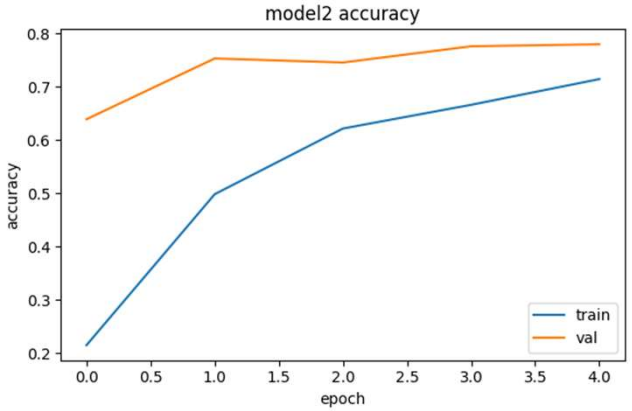


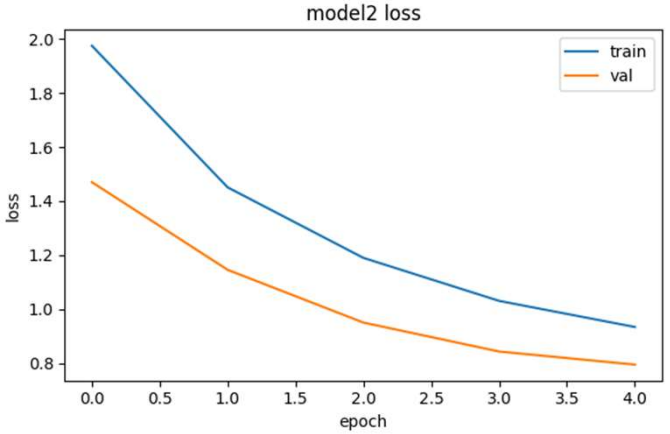




4) Classification supervisée

VGG16 avec data augmentation dans modèle





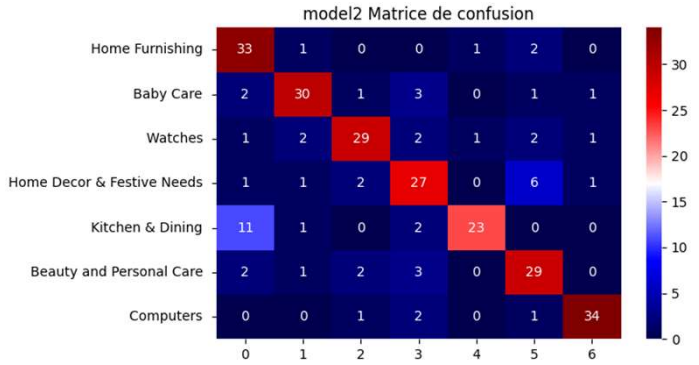
- L'accuracy s'améliore au fur et à mesure des epochs et l'erreur diminue

- Score best epoch :

Accuracy train ➔ 0,81
Accuracy validation ➔ 0,78

Le modèle aurait pu encore largement s'améliorer avec un nb d'epoch plus important

- Temps total = 698 s
- Temps best epoch = 135 s



- La catégorie Kitchen & Dining est moins bien prédite
- La catégorie computers est très bien prédite

4) Classification supervisée

7 Classification supervisée

VGG16 (avec data augmentation Image Data Generator)

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

Préparation extraction à partir du dataframe

- ❑ rajout du path dans dataframe
- ❑ Fonction de préparation du flow_from_dataframe
  - ➔ X_col = « image_path »
  - ➔ Y_col = « category »
  - ➔ target_size
- ❑ ImageDataGenerator
  - ➔ Génération du set de train data augmenté
  - ➔ découpage train et validation

Création du modèle + data augmentation

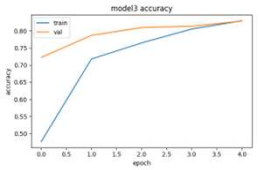
- ❑ Fonction de création de modèle
  - ➔ Récupération VGG16
  - ➔ Adaptation (global averagepooling2D, dense, dropout, dense (7))
- ❑ Création callback (save best model itération)

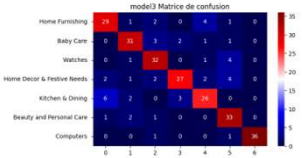
Entrainement & Test

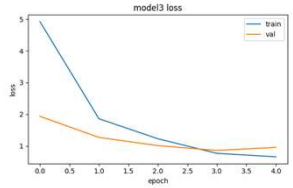
- ❑ Entrainement du modèle
  - ➔ epoch = 5
  - ➔ batch_size = 64
  - ➔ callbacks = callbacks_list

Evaluation

- ❑ Accuracy
- ❑ Loss
- ❑ Confusion matrix

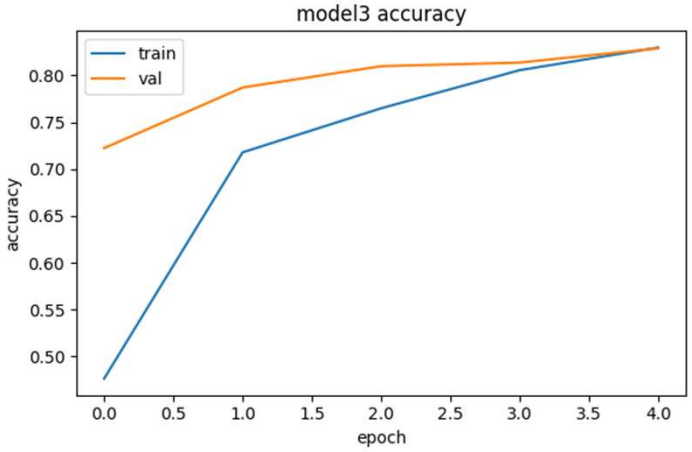






4) Classification supervisée

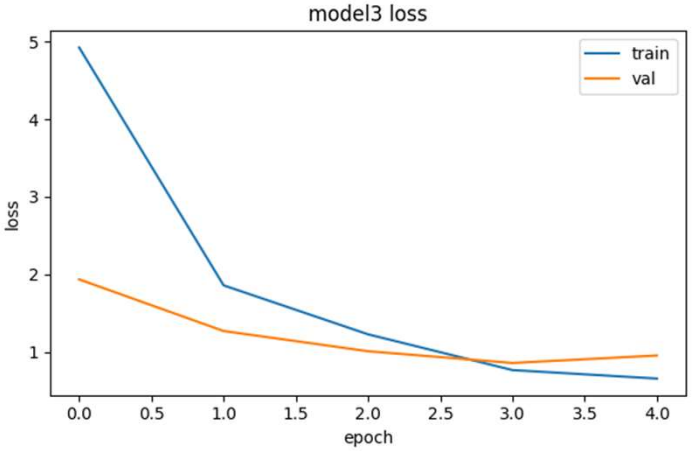
VGG16 avec data augmentation ImageDataGenerator



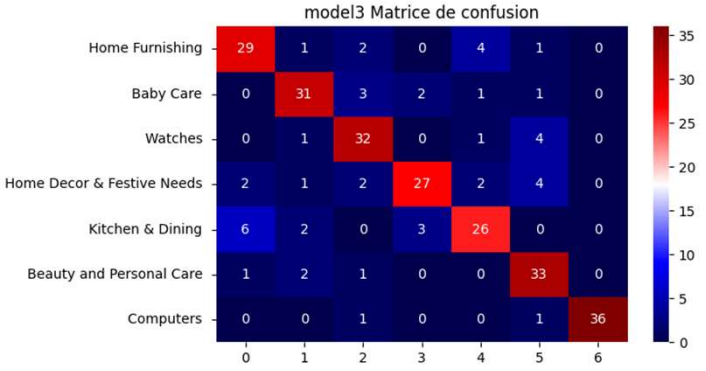
- L'accuracy s'améliore au fur et à mesure des epochs et l'erreur diminue

- Score best epoch :

Accuracy train ➔ 0,94
Accuracy validation ➔ 0,81



- Temps total = 686 s
- Temps best epoch = 136 s



- La catégorie Kitchen & Dining est moins bien prédite

- La catégorie computers est très bien prédite

4) Classification supervisée

8 Classification supervisée

RESNET50 (avec data augmentation Image Data Generator)

Jupiter Notebook, Python, Pandas, Numpy, Matplotlib, Seaborn, sklearn, os, imread, keras, tensorflow

Préparation extraction à partir du dataframe

- ❑ rajout du path dans dataframe
- ❑ Fonction de préparation du flow_from_dataframe
  - ➔ X_col = « image_path »
  - ➔ Y_col = « category »
  - ➔ target_size
- ❑ ImageDataGenerator
  - ➔ Génération du set de train data augmenté
  - ➔ découpage train et validation

Création du modèle + data augmentation

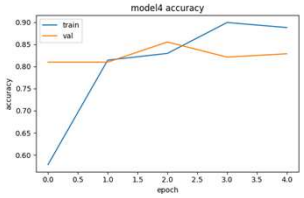
- ❑ Fonction de création de modèle
  - ➔ Récupération ResNet50
  - ➔ Adaptation (global averagepooling2D, dense, dropout, dense (7))
- ❑ Création callback (save best model itération)

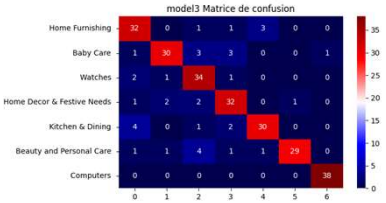
Entrainement & Test

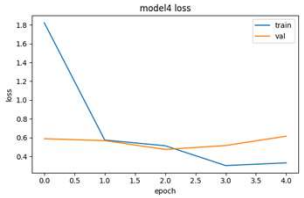
- ❑ Entrainement du modèle
  - ➔ epoch = 5
  - ➔ batch_size = 64
  - ➔ callbacks = callbacks_list

Evaluation

- ❑ Accuracy
- ❑ Loss
- ❑ Confusion matrix

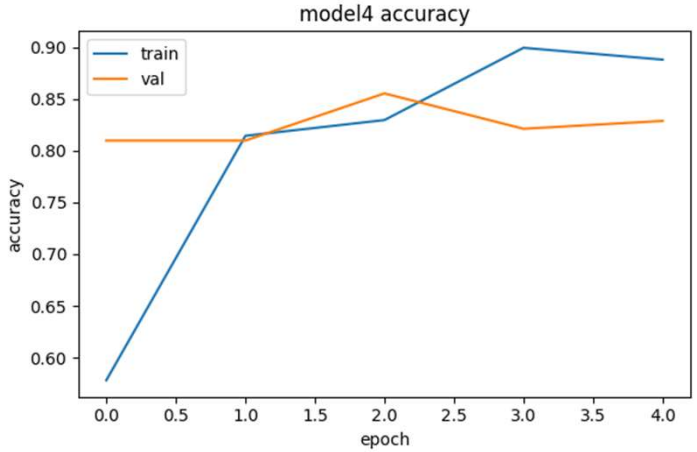


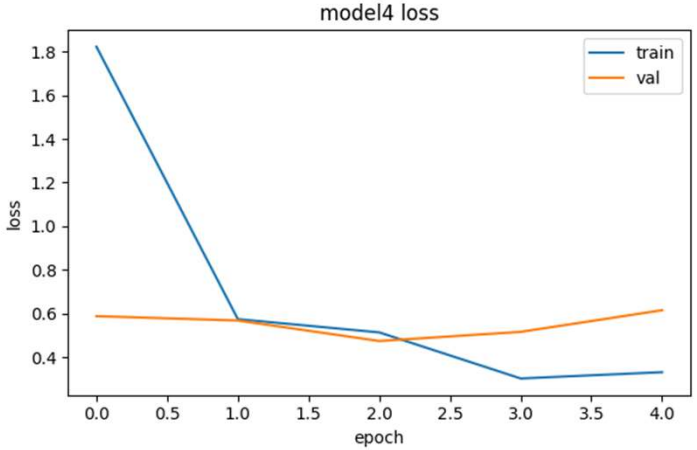




4) Classification supervisée

RESNET50 avec data augmentation ImageDataGenrator



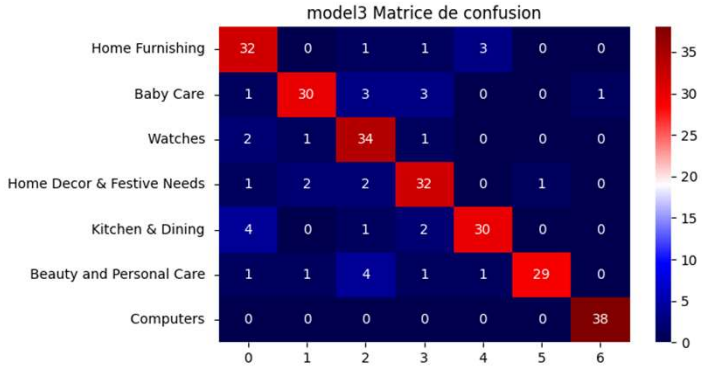


- L'accuracy s'améliore au fur et à mesure des epochs et l'erreur diminue

- Score best epoch :

Accuracy train ➔ 0,94
Accuracy validation ➔ 0,85

- Temps total = 271 s
- Temps best epoch = 55 s



- La catégorie Beauty & personnal care est moins bien prédite

- La catégorie computers est très bien prédite

4) Classification supervisée

Conclusion classification supervisée

METHODE	BEST EPOCH SCORE ACCURACY TRAIN	BEST EPOCH SCORE ACCURACY VALIDATION	TEMPS BEST EPOCH	TEMPS TOTAL	COMMENTAIRES
VGG16 (sans data augmentation)	0,94	0,83	135 s	683 s	+ Modèle performant - Temps de traitement très long
VGG16 (avec data augmentation dans modèle)	0,81	0,78	135 s	695 s	- Modèle qui n'a pas convergé (nb epoch insuffisants) - Temps de traitement très long
VGG16 (avec data augmentation Image Data Generator)	0,94	0,81	138 s	686 s	+ Modèle performant - Temps de traitement très long
RESNET50 (avec data augmentation Image Data Generator)	0,94	0,85	55 s	271 s	+ Modèle performant +Temps de traitement correct

V) Test API

5) Test API

9
Test API

Jupiter Notebook, Python, Pandas, Numpy, time, json

Création d'une fonction d'importation de produit → Test sur « vin » & « champagne » → **Sauvegarde des 10 premiers produits**

```
def edamam_use(produit) :
# importation URL
   url = "https://edamam-food-and-grocery-database.p.rapidapi.com/api/food-database/v2/parser"
# requête
   querystring = {"ingr":produit}
# idenfifiant + lien vers base
   headers = {"X-RapidAPI-Key": os.getenv("clé"),
   "X-RapidAPI-Host": "edamam-food-and-grocery-database.p.rapidapi.com"}
# réponse requête
   response = requests.get(url, headers=headers, params=querystring)
   json_doc = response.json()
# création dataframe produits
   df_data = pd.json_normalize(pd.DataFrame(json_doc['hints'])["food"].to_list())
   df_data = df_data[["foodId","label","category","foodContentsLabel","image"]]
   return df_data
```

5) Test API

Extraction dataframe

	foodId	label	category	foodContentsLabel	image
0	food_a656mk2a5dmqb2adiamu6beihduu	Champagne	Generic foods	NaN	https://www.edamam.com/food-img/a71/a718cf3c52add522128929f1f324d2ab.jpg
1	food_b753ithamdb8psbt0w2k9aquo06c	Champagne Vinaigrette, Champagne	Packaged foods	OLIVE OIL; BALSAMIC VINEGAR; CHAMPAGNE VINEGAR; GARLIC; DIJON MUSTARD; SEA SALT.	NaN
2	food_b3dyababjo54xobm6r8jzbghjgqe	Champagne Vinaigrette, Champagne	Packaged foods	INGREDIENTS: WATER; CANOLA OIL; CHAMPAGNE VINEGAR; SUGAR; OLIVE OIL; SALT; DRIED GARLIC; DRED SHALLOTS; BLACK PEPPER; XANTHAN GUM; SPICE	https://www.edamam.com/food-img/d88/d88b64d97349ed062368972113124e35.jpg

VI) Conclusion

6) Conclusion

Etudier la faisabilité d'un moteur de classification des articles par catégories, à partir du texte et de l'image	
Etudier la faisabilité d'un moteur de classification avec le texte	Démonstration réalisée avec étude de faisabilité : • avec approches fréquence, TF-IDF, Word2Vec, BERT, USE • réduction de dimensions PCA + TSNE • Comparaison vraies catégories et catégories Kmeans (ARI) ➔ Approches concluantes
Etudier la faisabilité d'un moteur de classification avec les images	Démonstration réalisée avec étude de faisabilité : • avec approches SIFT, CNN • réduction de dimensions PCA + TSNE • Comparaison vraies catégories et catégories Kmeans (ARI) ➔ CNN concluant
Réaliser une classification supervisée à partir des images	Classification supervisée : • VGG16 : Sans data augmentation, data augmentation dans le modèle, data augmentation à partir du fichier image • RESNET50 : data augmentation à partir du fichier image ➔ Approches concluantes
Tester la collecte de produits via une API	Elaboration d'une fonction de récupération de produits à partie de l'API edamam-food-and-grocery-database.p.rapidapi.com

Merci

- Armand FAUGERE
- armand-faugere@live.fr