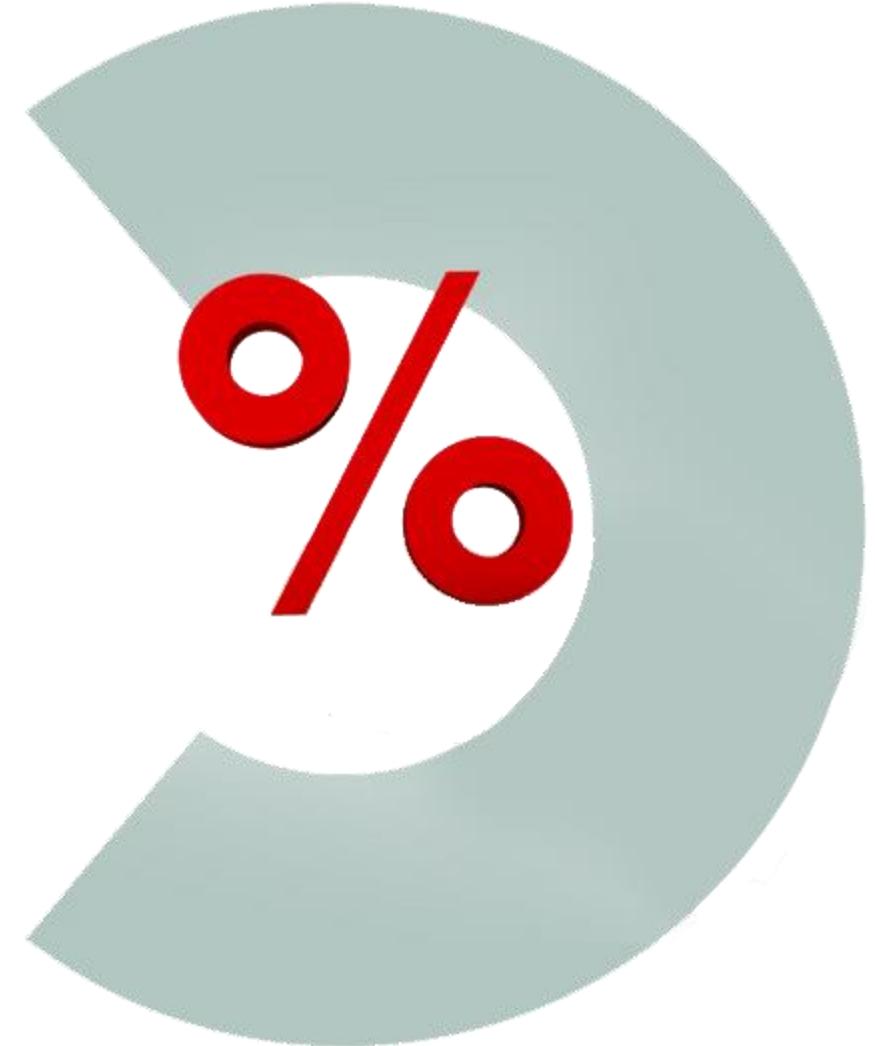


OPENCLASSROOMS

Parcours Data Scientist

Projet 6 :
Classification automatique de
biens de consommation



SOMMAIRE

Découverte des données

Partie 1, Travail sur le texte

I, Mise en place d'un modèle de Machine learning

A, Préparation des données et création d'une Matrice tf-idf

B, Recherche de sujet par Latent Dirichlet Allocation

C, Recherche de sujets par Non Negative Matrix Factorization

II, Mise en place d'un modèle de Deep learning

A, Préparation des données

B, Recurrent Neural Network

C, RNN bidimensionnel

Partie 2, Travail sur les images

I, Mise en place d'un classificateur basé sur les features

A, Sélection des features

B, Création des clusters pour les features

C, Réduction de dimensions et visualisation

D, Prédiction

II, Mise en place d'un modèle de Deep learning

A, Convolutional neural network

B, CNN avec data augmentation

C, CNN avec transfer learning

Conclusion : mise en place d'un modèle

DÉCOUVERTE DES DONNÉES

```
▶ data.shape
```

```
(1050, 14)
```

```
▶ data.isna().sum()
```

```
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
```

```
▶ data.cat_1.value_counts()
```

Home Furnishing	150
Baby Care	150
Watches	150
Home Decor & Festive Needs	150
Kitchen & Dining	150
Beauty and Personal Care	150
Computers	150

```
Name: cat_1, dtype: int64
```

Dataset se compose de 1050 lignes et 14 colonnes.

Pour notre projet 3 colonnes vont nous intéresser :

- product_category_tree que nous allons travailler pour me faire remonter que les catégories principales au nombre de 7
- description, qui correspond aux descriptions laissées par les vendeurs et sur lesquelles nous allons nous baser pour la partie 1
- image, qui correspond au nom de l'image correspond au produit et donc nous allons nous servir dans la partie 2

PARTIE 1 TRAVAIL SUR LE TEXTE

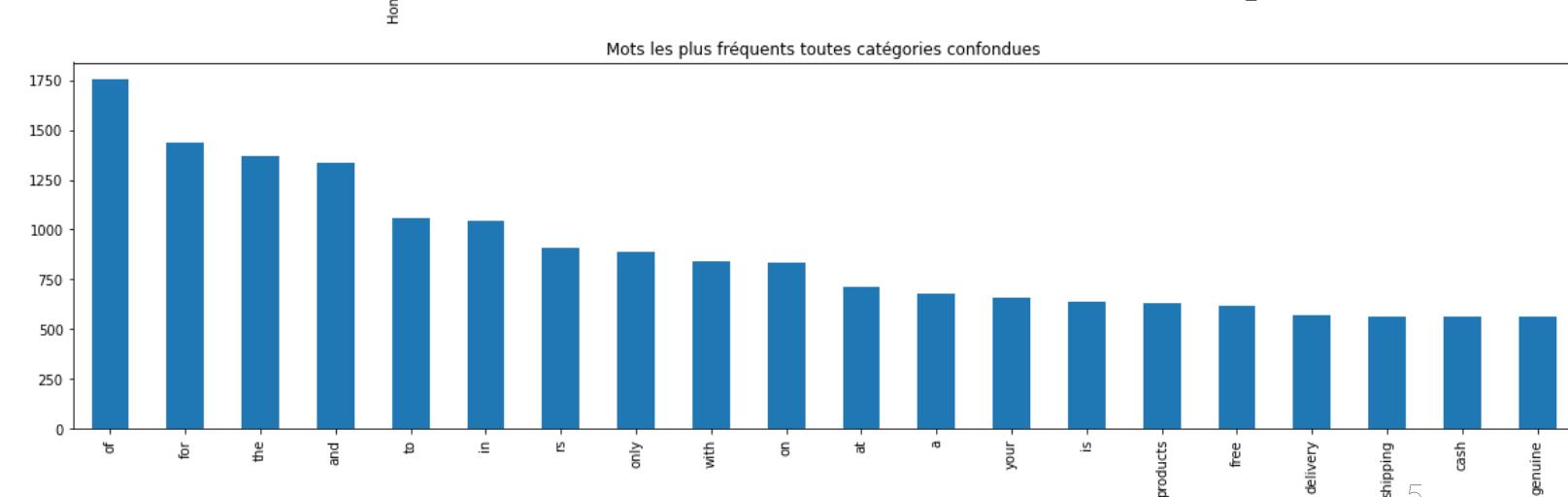
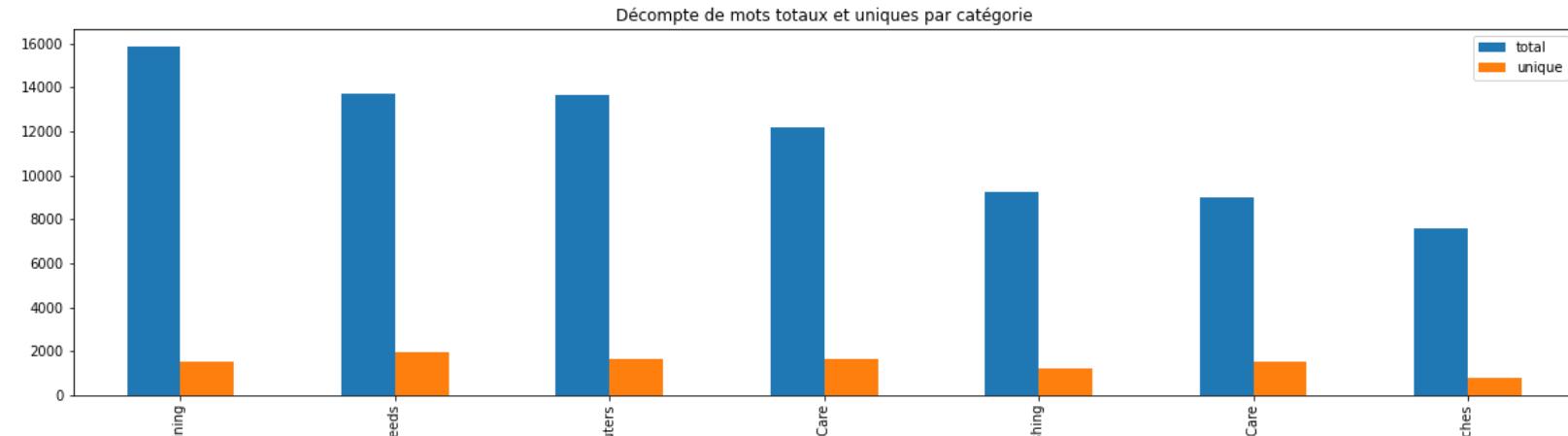
MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

A, Préparation des données et création d'une matrice TF-IDF

Mise en place d'un **dictionnaire** par catégorie.

Tokenization (séparation de tous les mots).

Décompte des mots (totaux et uniques)

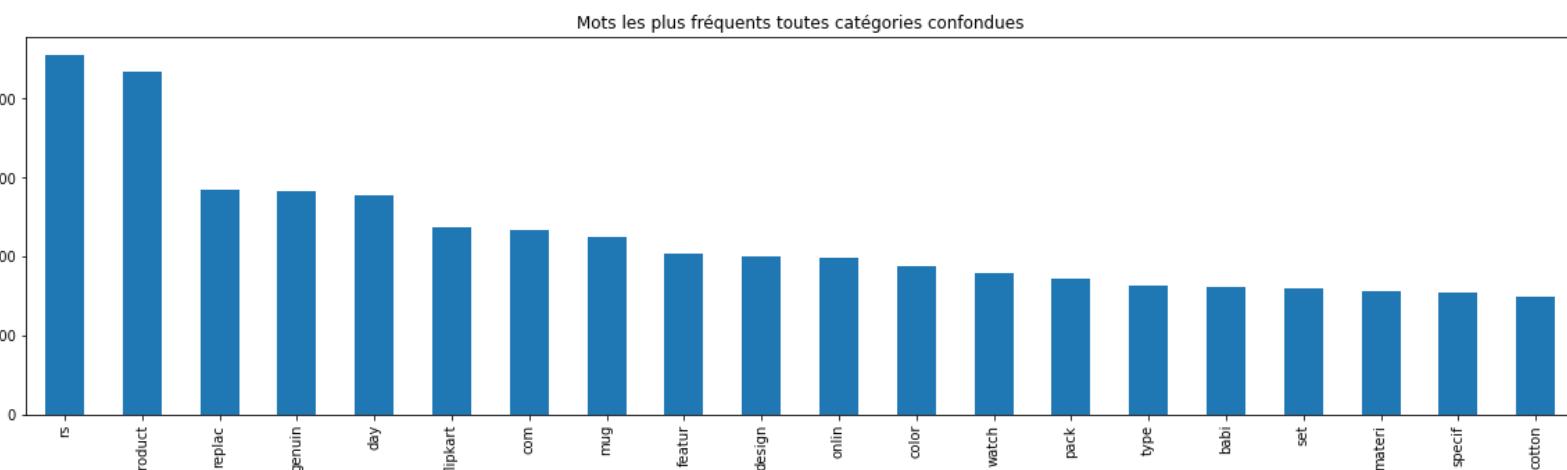
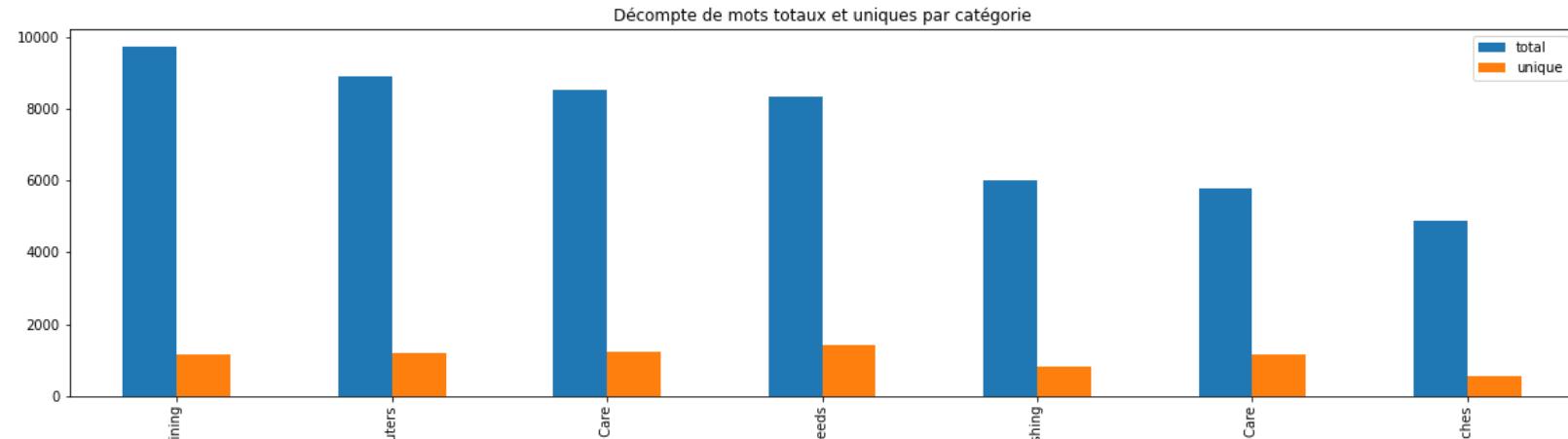


MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

A, Préparation des données et création d'une matrice TF-IDF

Retrait des '**stop words**',
mots de liaison donnant peu
d'information sur le sens.

Stemming, réduction des
mots à leur racine afin de les
regroupés.



I, MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

A, Préparation des données et création d'une matrice TF-IDF

Mise en place d'un
bag of word

(sac de mots dresse
la liste exhaustive
des mots utilisés et
leur **fréquence** par
catégorie).

	01433cmgi	01727lpln	01741lpln	01784bmli	03918cmli	04615cmgi	05712lmli	05tg	06362cmgi	07034lmli	...	zikrak	zinc	zingalala	zip	zipexterior
Baby Care	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0
Beauty and Personal Care	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
Computers	0	0	0	0	0	0	0	0	0	0	...	0	0	1	1	0
Home Decor & Festive Needs	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0
Home Furnishing	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0
Kitchen & Dining	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
Watches	1	1	1	1	1	1	1	0	1	1	...	0	0	0	0	0

7 rows x 4511 columns

I, MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

A, Préparation des données et création d'une matrice TF-IDF

Création de la matrice TF-IDF

(term frequency – inverse document frequency) qui met en avant l'importance des mots dans chaque catégories et entre les catégories.

Ici on prends en compte les **monogrammes** et les **digrammes**

(2 éléments adjacents pertinents)

Matrice finale de 7 lignes et **10333 colonnes**.

Nous **vectorisons** les data à travers le prisme de cette matrice.

```
tfidvectorizer = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), max_df=0.8, stop_words='english')

tfidf_wm = tfidvectorizer.fit_transform(corpus_count)

tfidf_tokens = tfidvectorizer.get_feature_names()

df_tfidfvect = pd.DataFrame(data=tfidf_wm.toarray(), index = df5.index, columns = tfidf_tokens)
```

	01433cmgi	01433cmgi 77036sm02j	01727lpin	01727lpin bezel	01741lpin	01741lpin 7052ym07	01784bml	01784bml 7092si01	03918cml	03918cml maxima	...	zipper silicon	zone	zone expos	z h
Baby Care	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.000000	0.000000	0.00000	0.00000
Beauty and Personal Care	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.000000	0.020096	0.02421	0.000
Computers	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.023214	0.000000	0.00000	0.00000
Home Decor & Festive Needs	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.000000	0.000000	0.00000	0.00000
Home Furnishing	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.000000	0.024354	0.00000	0.029
Kitchen & Dining	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	...	0.000000	0.000000	0.00000	0.00000
Watches	0.03398	0.03398	0.03398	0.03398	0.03398	0.03398	0.03398	0.03398	0.03398	0.03398	...	0.000000	0.000000	0.00000	0.00000

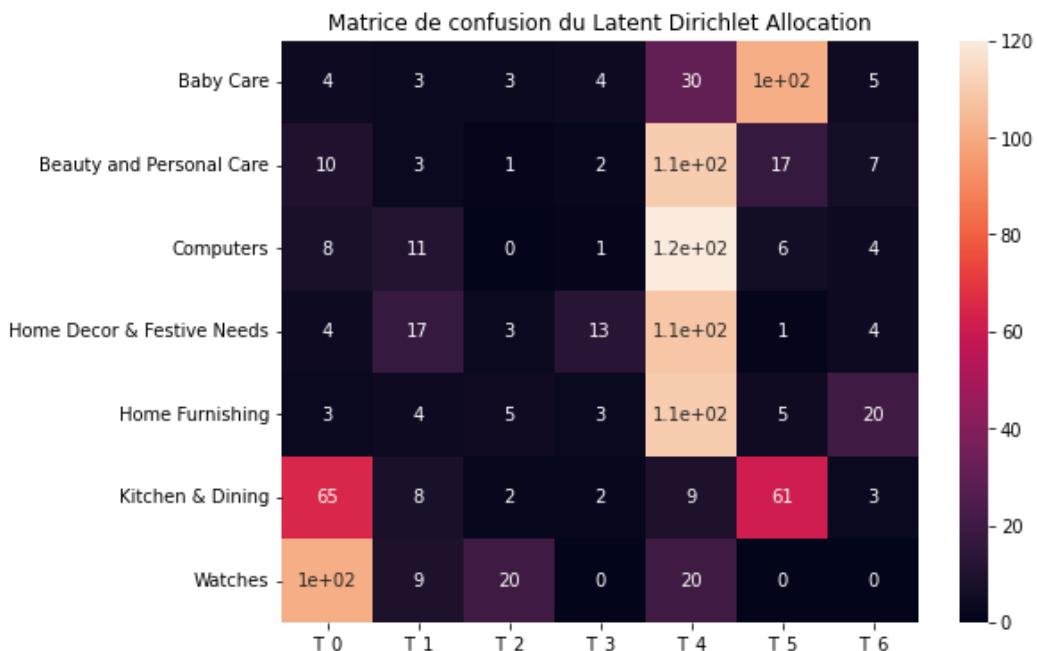
7 rows × 10333 columns

I, MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

B, Recherche de sujet par Latent Dirichlet Allocation

LDA est un modèle génératif probabiliste permettant d'expliquer des ensembles d'observations par le moyen de groupes non observés.

L'occurrence de chaque mot est révélateur d'un thème et fait pencher le document vers un thème.



Principaux mots de chaque thème trouvé par le LDA

```
Topic 0 :  
parti anthil 799lpp02 fu203 suitabl better rectangl urban lumen damag avoid solid basic exposur accid hotel usespec  
if scratch clock small height swivel fix navneet 350ml stainless eman sky high fashionin  
Topic 1 :  
wed smart deep prabhavali magnific paper statubuy regent leaf hue beautifi mobil audio wkhs0164 beyouti copper buff  
et to socialis eye dreamt pos0081 intrferenti glassspecif fantast yeskey goddess grecrafto music  
Topic 2 :  
entwin vanilla beat defect site oz equinox eb 8098ym01 makeup charm treasur propylen innov creamlotti kokum ident m  
odish entir light intrferenti sued sand profil mussel unisex iso artpiec yeskey memor mmull  
Topic 3 :  
stich ftbuy month neckflipkart eas katori prsmd barbi innov charm jain buddhist dimmabl visual media 23x16x33 cmsbu  
y neckflipkart com memori line floral fork tine sport north match femal larg addit tip handl 12011ppgw 333tms333 co  
lour vitamin  
Topic 4 :  
tonneau bendabl siliconebuy cut heavi reliabl conform sam cutterstep4d pc pleas 3951 dv6170ea lenovo 1003 stucco  
ncret hip 272017a 2in1 3key king 1x2gb sdram display mention reader cut function electr nice sf vc_011 dv6171cl ace  
r  
Topic 5 :  
leav air ensur avail glovecotonex nagar mandhania onlinemaniya shag bathmat_ri zero home polyfil thrade surg mn acc  
ent string aci20160340 allur quick option authent thing mb990hn mb99011 zipper flap  
Topic 6 :  
meet intern appli wish damag everlast pmr1918 pmba1872 catarrh clip observ preval youth craft jack macintosh faux a  
merica europ returnspecif brillar mohanjodero woodspecif jumpsuit uft tsw softest safest suitabl winter pillow thre  
ad abil
```

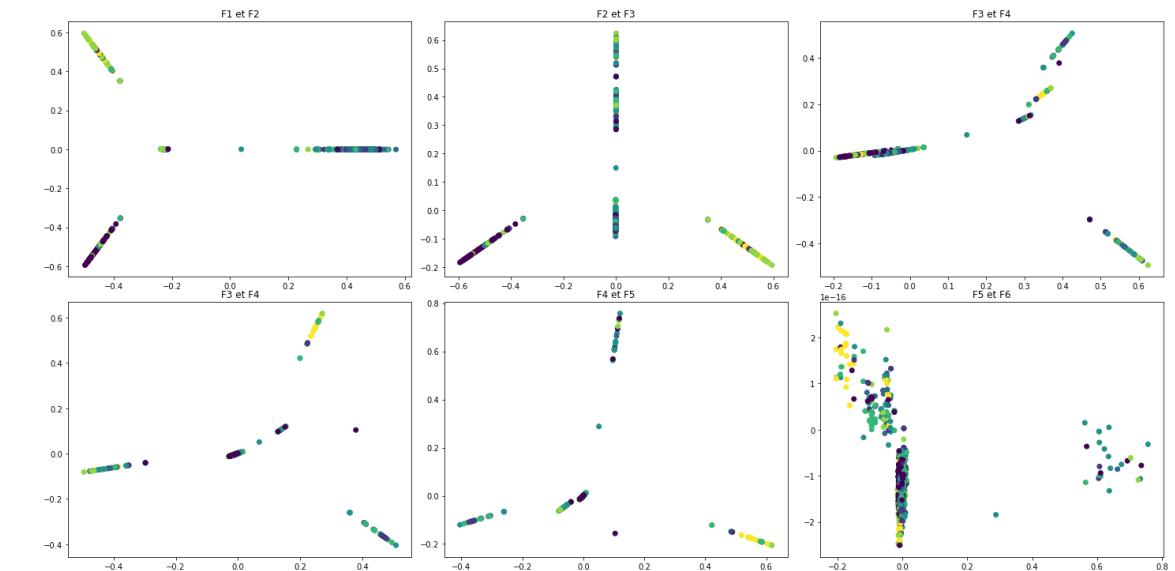
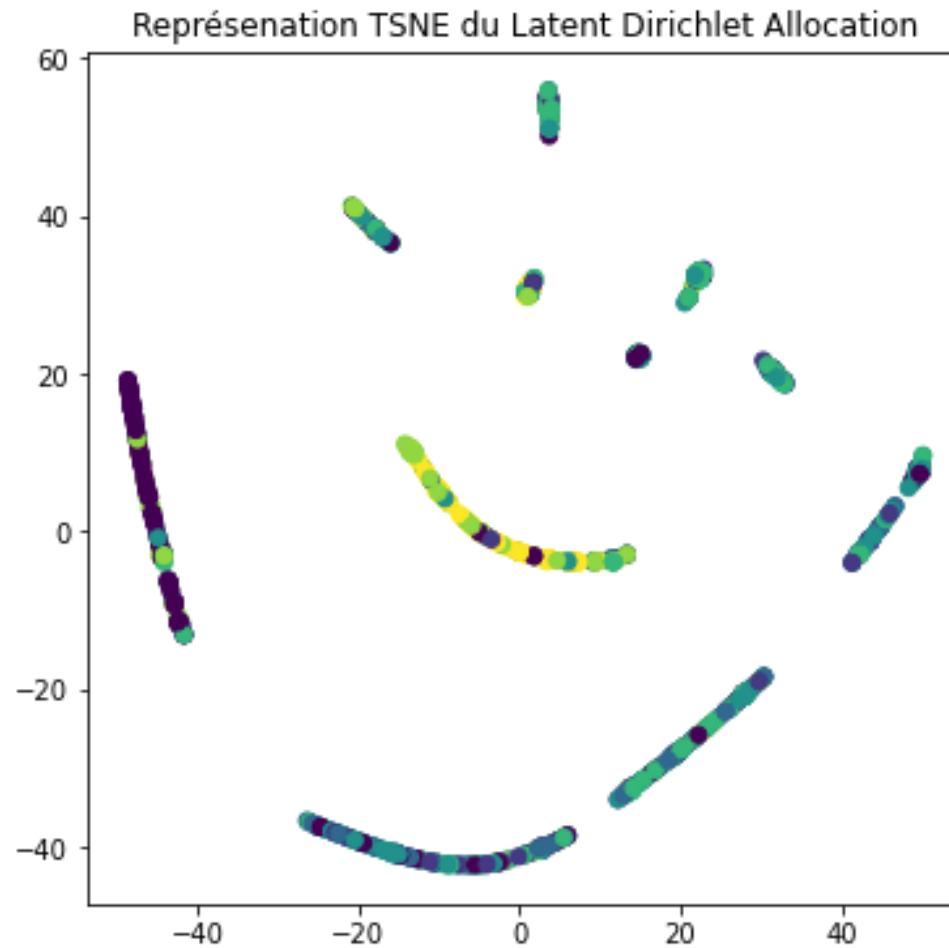
Coefficient ARI : 0,17

Mesure de similarité entre deux groupes.

Cela nous montre que le **LDA est un échec** et que les groupes identifiés ne sont **pas assez distinct** les uns des autres.

IMISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

B, Recherche de sujet par Latent Dirichlet Allocation

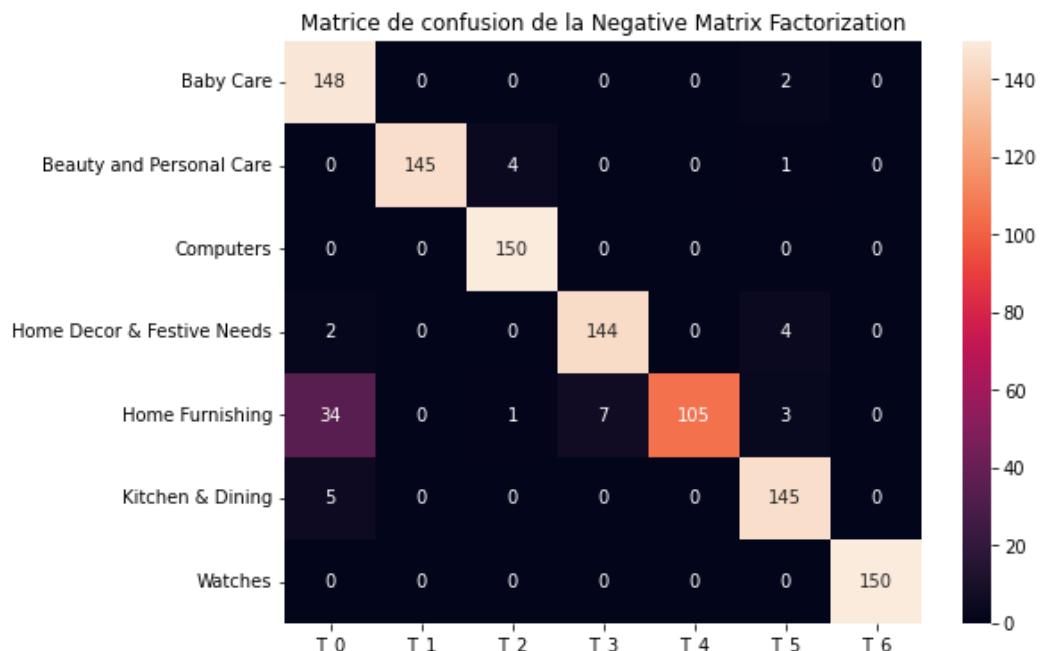


Les représentations graphiques confirment la matrice de confusion et le coefficient ARI,
le LDA n'arrive pas à distinguer les groupes.

MISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

C, Recherche de sujets par Non Negative Matrix Factorization

NMF est un groupe d'algorithme où une matrice (ici TF-IDF) est factorisée en plusieurs matrices afin de trouver les espaces moins denses qui délimitent des groupes.



Principaux mots de chaque thème trouvé par la NMF

```
Topic 0 :  
pile yarn bilstri strblue coverbuy gold aroma afternoon siesta trident hnh1 trident trick horizont ferrari  
ferrari 75x150cm solid short goe greedi goe trendz softright trendz pamper origin nkp blstri kolaveri  
  
Topic 1 :  
01433cmgi 7963pp02 dean 8925ym06 8925ym06 flippd 8944s103 8944s103 317tms317 8959y101 8959y101 yuva 8974pp01 8974pp  
01 n1192 romex priceless mundan mundan timepiec romex 906_blk 906_blk flaunt cnl cnl rg fabul event rked perucci  
  
Topic 2 :  
zyxel 3g mani anti mb990hn mb99011 mb990hn mb13311 60w mb13311 mb003ta respect mb003ta mb003ja mb003ja maxi  
mum keyston max peacock max mathemat num mathemat match femal mari max mari manualkey smartpro mb99011  
  
Topic 3 :  
purpos popular reusabl soften chandan multivitamin rheumat arthriti rheumat reviv benefit reviv charg surg returnsp  
ecif brillar read bedroom returnspecif returnkey stay returnkey return fidel cheek cheek super richfeel richfeel su  
nshield chandan chamomil angelica  
  
Topic 4 :  
bo slicer ski winner fantast vintag skip skip evil boil picnic boil skyblue02 skyblue02 lunch magic fogg bo 550ml s  
lice slice siver slicer sam bone mad abstrct mad slow slow ga  
  
Topic 5 :  
compon plaquet sleepsuit sj tulip skey kandyfloss faq trait skin enrich bornbabbykid munchkin bornbabbykid skirt  
skirt feet born shown boreal rectangular boreal booti age sleepsuit booti shrikant trader sleepsuitkey sleepsuitkey  
3kfatori sleeve  
  
Topic 6 :  
arsalan awesom b117 student perform occupi strike studio mirror occupi occas corpor sturdi ambien occas style ch  
eck occas style dirt hd awesom guest obstacl patron subtl act sub sub thriller obstacl awaken titl
```

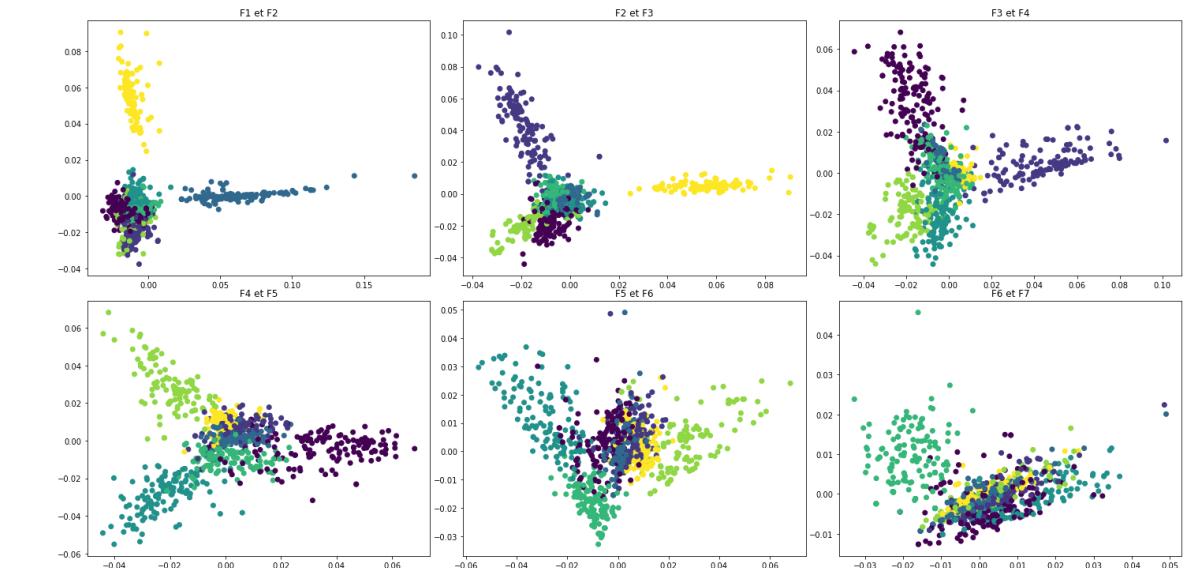
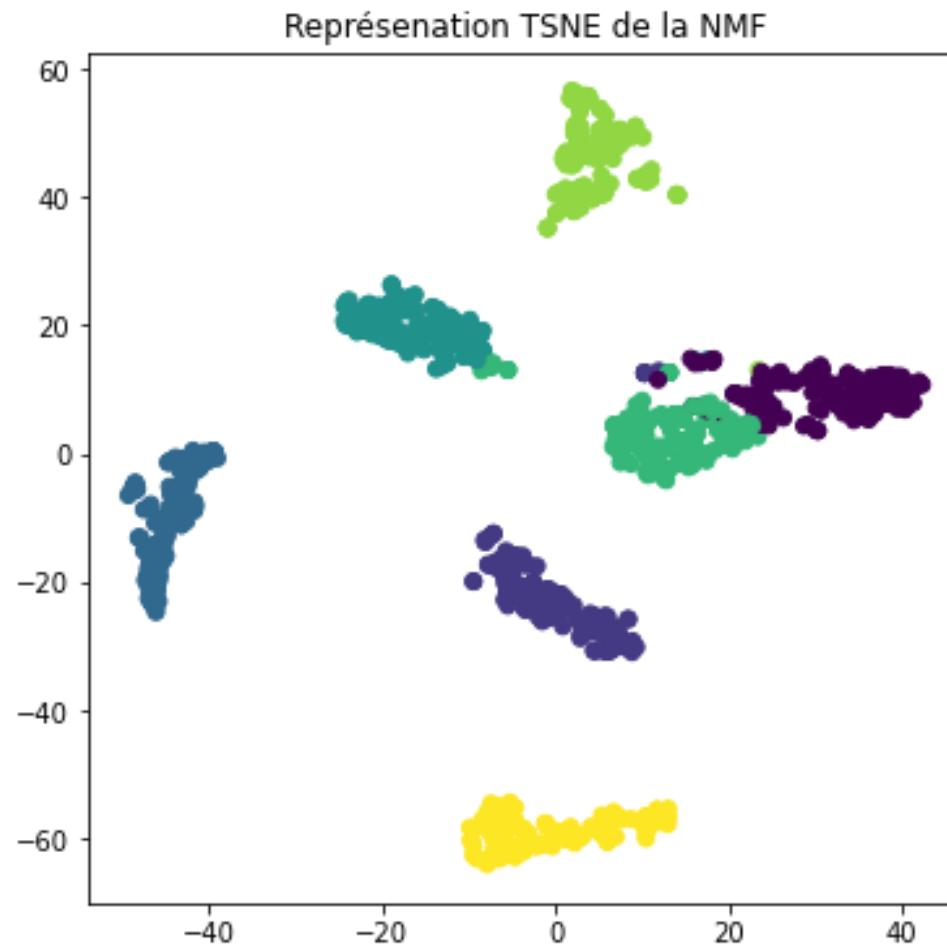
Coefficient ARI : 0,87

Mesure de similarité entre deux groupes.

Cela nous montre que la **NMF** a réussi à délimiter des **groupes très différents** les uns des autres.
La matrice de confusion le montre avec une belle **diagonale..**

IMISE EN PLACE D'UN MODÈLE DE MACHINE LEARNING

C, Recherche de sujets par Non Negative Matrix Factorization



Les représentations graphiques montrent cette distinction des groupes, particulièrement le **TSNE**. On voit néanmoins deux groupes qui se **touchent**, la matrice nous l'avait déjà montré.

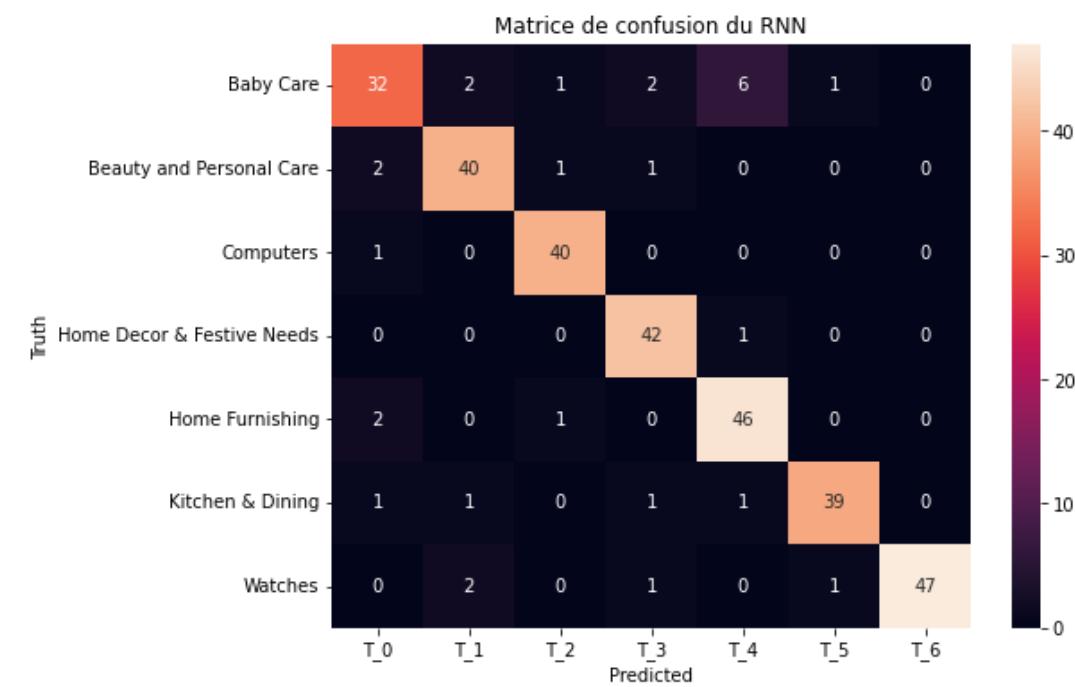
II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, Mise en place d'un Recurrent Neural Network (RNN)

```
rnn = keras.models.Sequential()
rnn.add(keras.layers.Embedding(vocab_size, 80, input_length=max_lengh))
rnn.add(keras.layers.Flatten())
rnn.add(keras.layers.Dense(7, activation='softmax'))

rnn.compile(optimizer='Adam', loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
```

Temps d'entraînement du modèle : 18,12 secondes



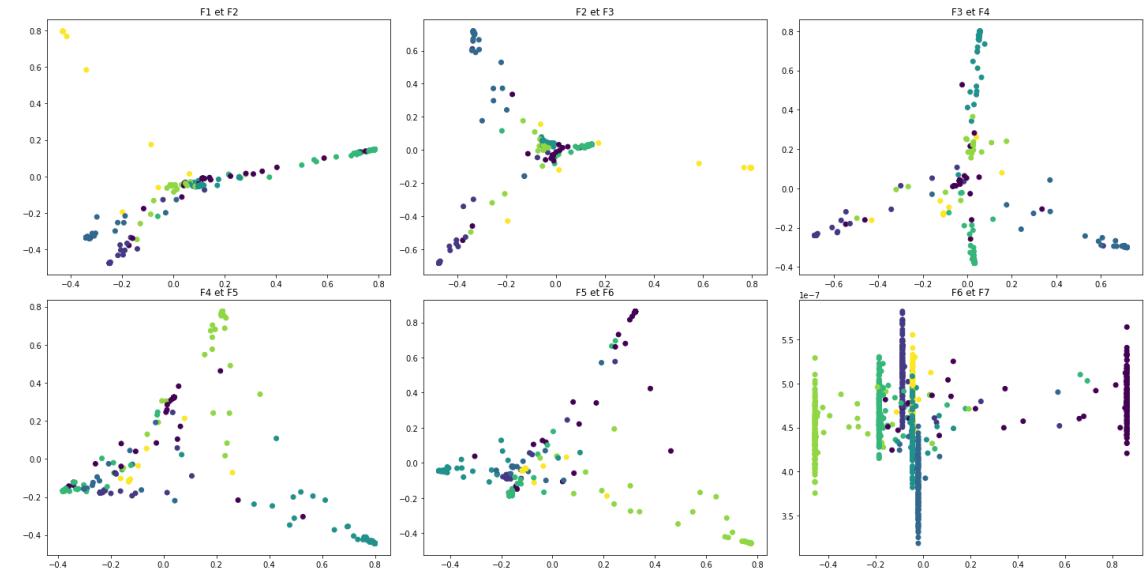
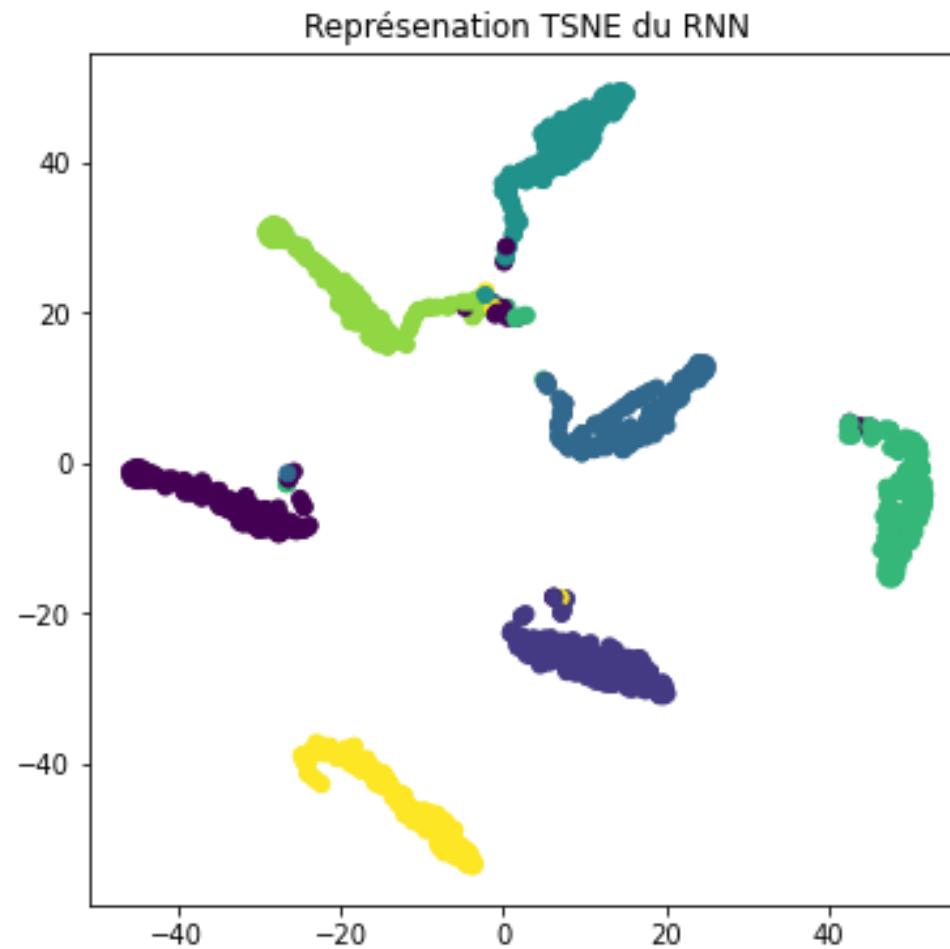
Coefficient ARI : 0,80

Mesure de similarité entre deux groupes.

Le score ARI est moins bon que pour la NMF, cependant, la matrice de confusion est plus équilibrée.
Tous les groupes sont bien distincts.

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, Mise en place d'un Recurrent Neural Network (RNN)



Le TSNE montre très bien les groupes distincts. Contrairement à la NMF, tous les groupes sont distincts et seuls quelques descriptions sont mal interprétées par le modèle.

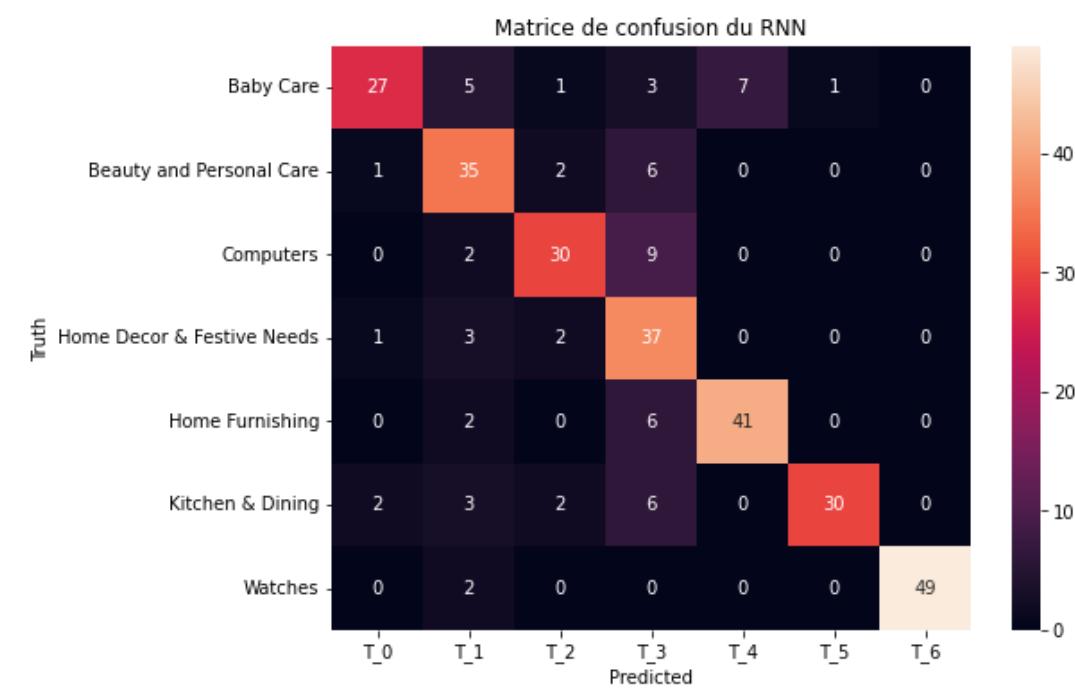
II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, Mise en place d'un RNN bidirectionnel

```
rnn_bi = keras.models.Sequential()
rnn_bi.add(keras.layers.Embedding(vocab_size, output_dim=80, input_length=max_lengh, name='embedding'))
rnn_bi.add(keras.layers.Bidirectional(keras.layers.LSTM(160)))
rnn_bi.add(keras.layers.Dense(7, activation='softmax'))

rnn_bi.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Temps d'entraînement du modèle : **360,41** secondes



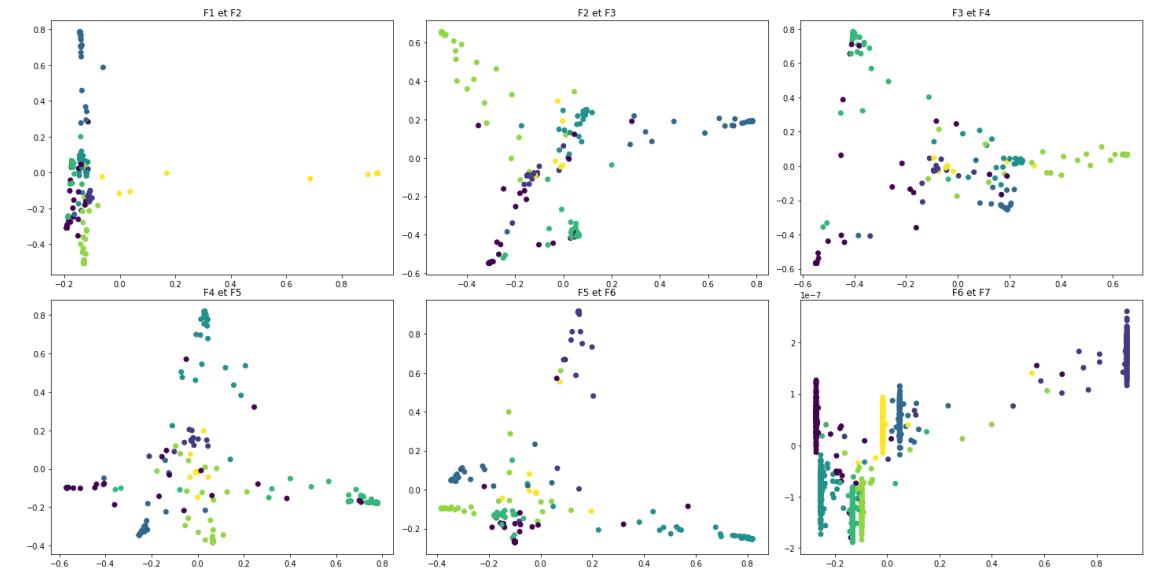
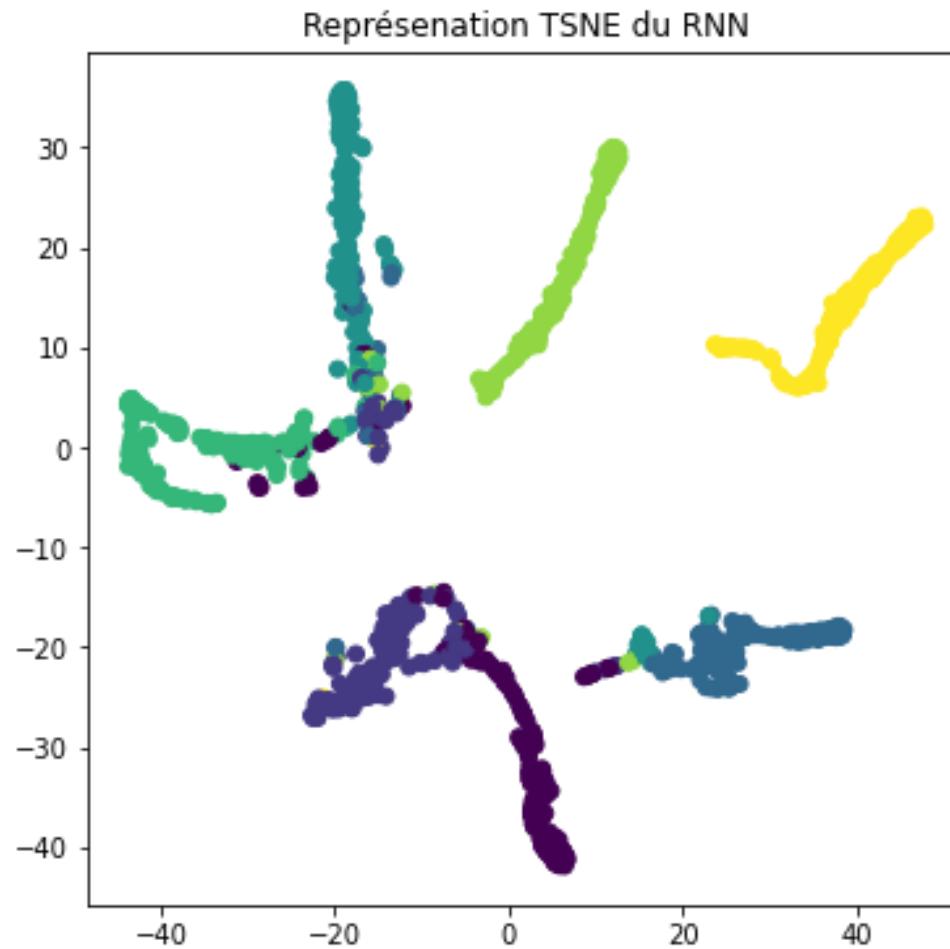
Coefficient ARI : **0,57**

Mesure de similarité entre deux groupes.

Le RNN bidimensionnel fait **moins bien** que le RNN standard et en **20x** plus de temps.

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, Mise en place d'un Recurrent Neural Network (RNN)



Les graphiques confirment que les limites entre les groupes sont **moins bien définies**.

Ce modèle n'est **pas pertinent** pour notre problématique.

PARTIE 1 , CONCLUSION

De tous les modèles essayés, celui qui nous donne le plus de satisfaction est le **RNN simple**.

La NMF a certes obtenu un meilleur score ARI mais la matrice de confusion du réseau de neurone était plus **équilibrée**.

De plus, les RNN a obtenu **90%** de prédictions correctes sur un jeu de donnée test. Ce score nous invite à penser qu'un moteur de classification est **faisable** en se basant sur ce modèle.



PARTIE 2 TRAVAIL SUR LES IMAGES

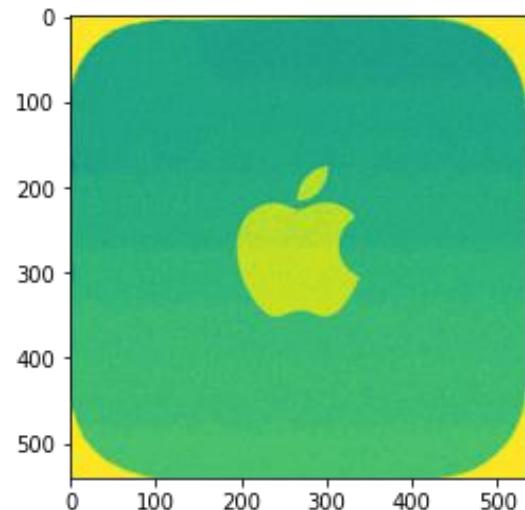


I, MISE EN PLACE D'UN CLASSIFICATEUR BASÉ SUR LES FEATURES

A, Sélection des features

```
#!/usr/bin/python3  
#  
sift_keypoints = []  
img_keypoints = []  
tempst1 = time.time()  
  
sift = cv.SIFT_create(500)  
  
for img in images2:  
    kp, des = sift.detectAndCompute(img, None)  
    sift_keypoints.append(des)  
    #img_keypoints.append(cv.drawKeypoints(img, kp, None, flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS))  
print("Duration :", time.time()-tempst1)
```

Duration : 416.30388951301575



Méthode SIFT

(scale invariant feature transform)

limitation manuelle à **500** features par image.

L'algorithme va chercher des **points d'intérêt** invariants à différentes échelles et rotations.



I, MISE EN PLACE D'UN CLASSIFICATEUR BASÉ SUR LES FEATURES

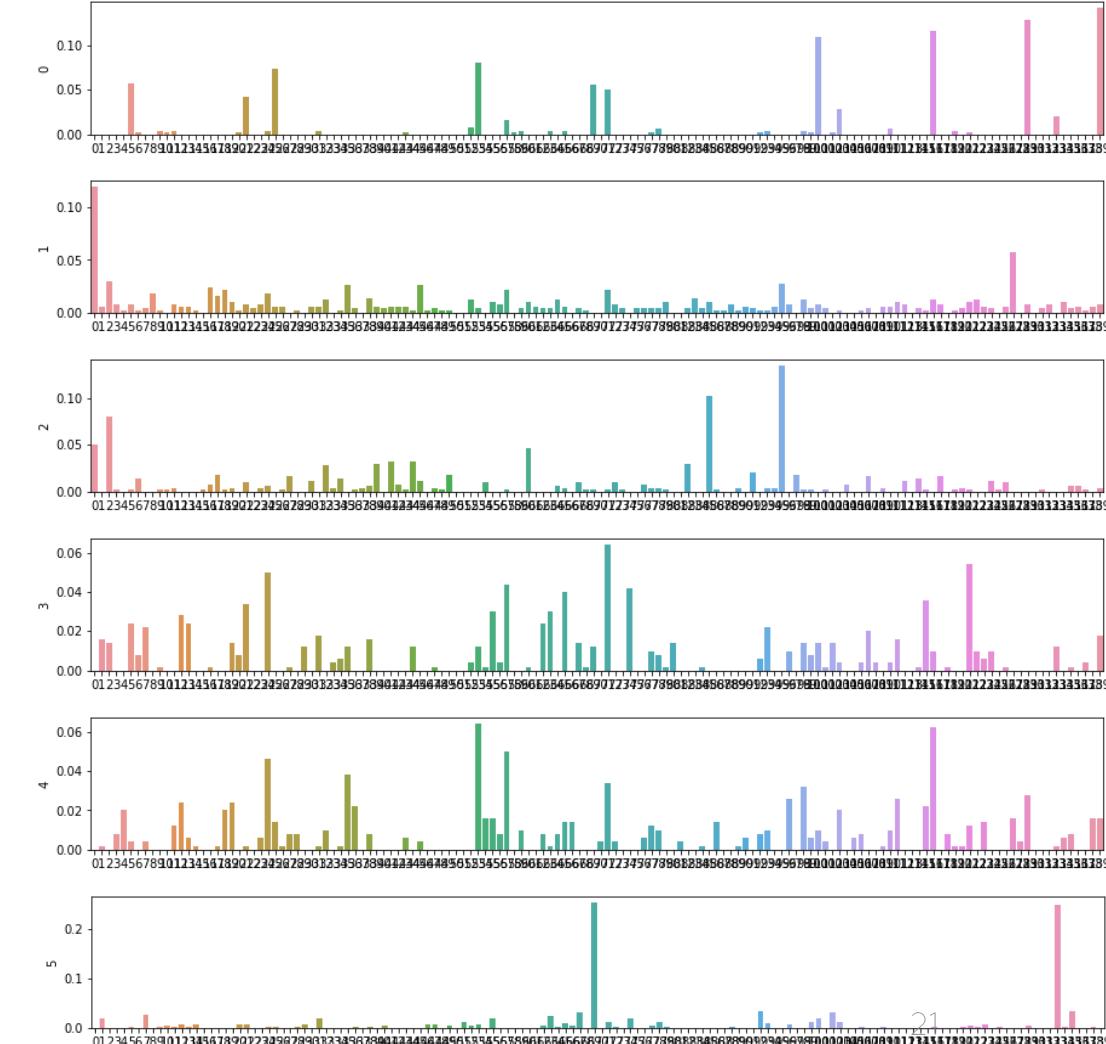
B, Création des clusters pour les features

```
k = 140  
  
print("Nombre de K : ", k)  
  
kmeans = MiniBatchKMeans(n_clusters=k, batch_size=20000)  
kmeans.fit(sift_keypoints_all)  
print("Duration :", time.time() - temps1)
```

Nombre de K : 140
Duration : 18.678999662399292

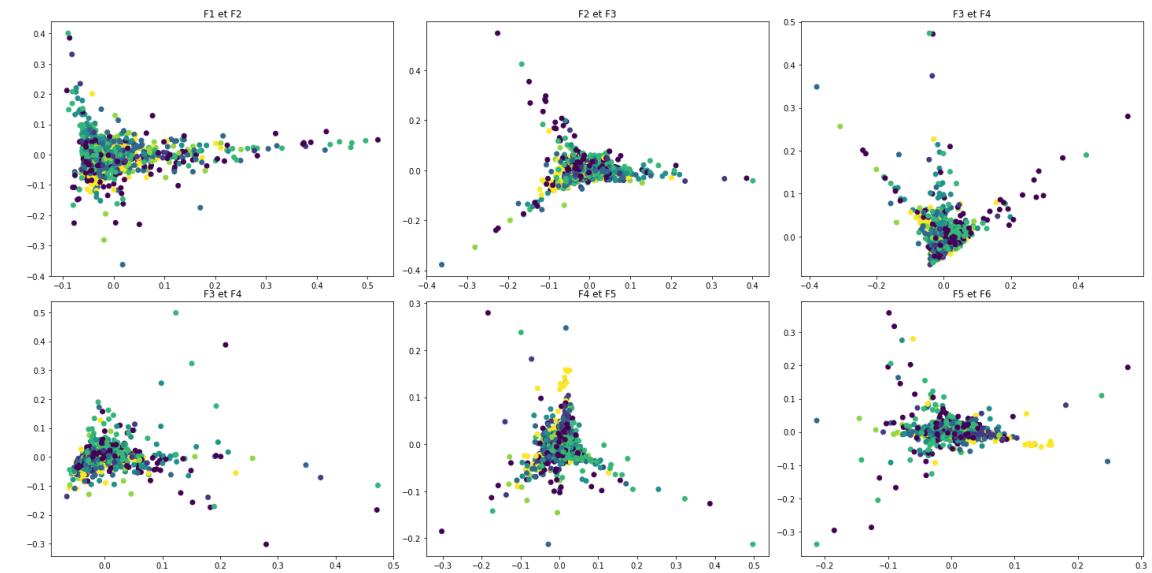
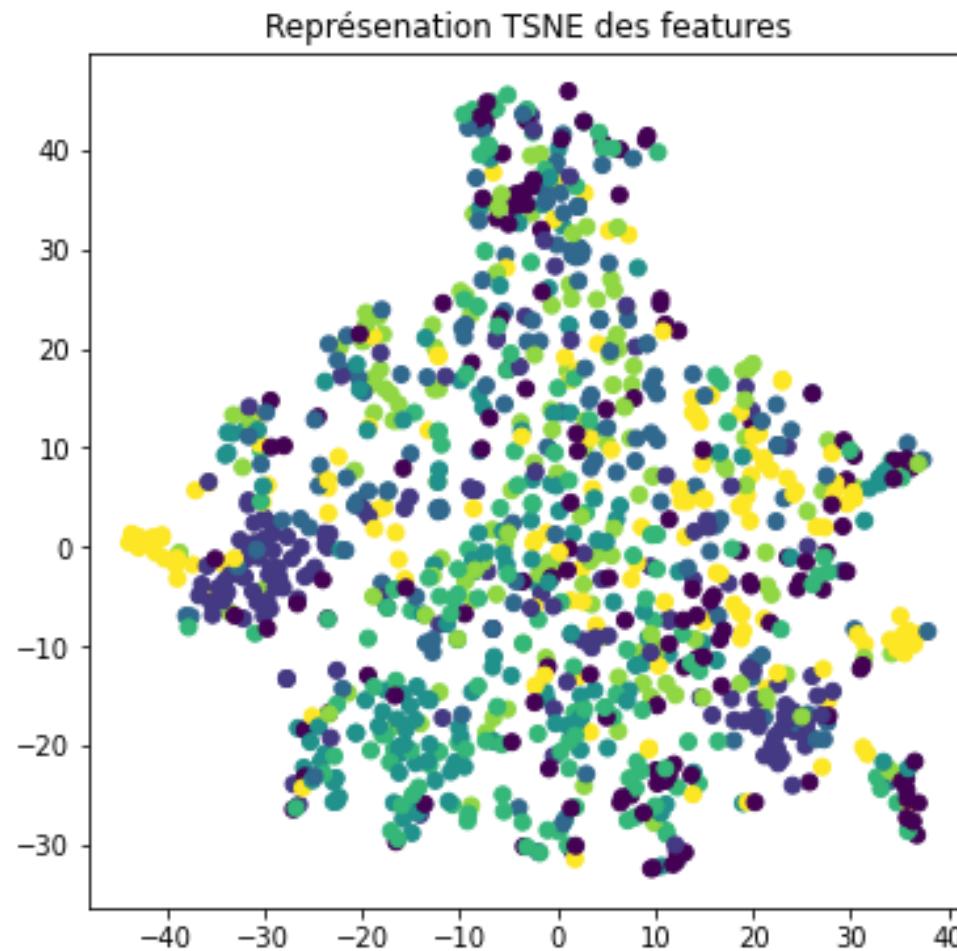
Clusterisation avec mini batch Kmeans pour le **regroupement** des features entre 140 groupes.

Création d'**histogrammes** représentant les différents features.



I, MISE EN PLACE D'UN CLASSIFICATEUR BASÉ SUR LES FEATURES

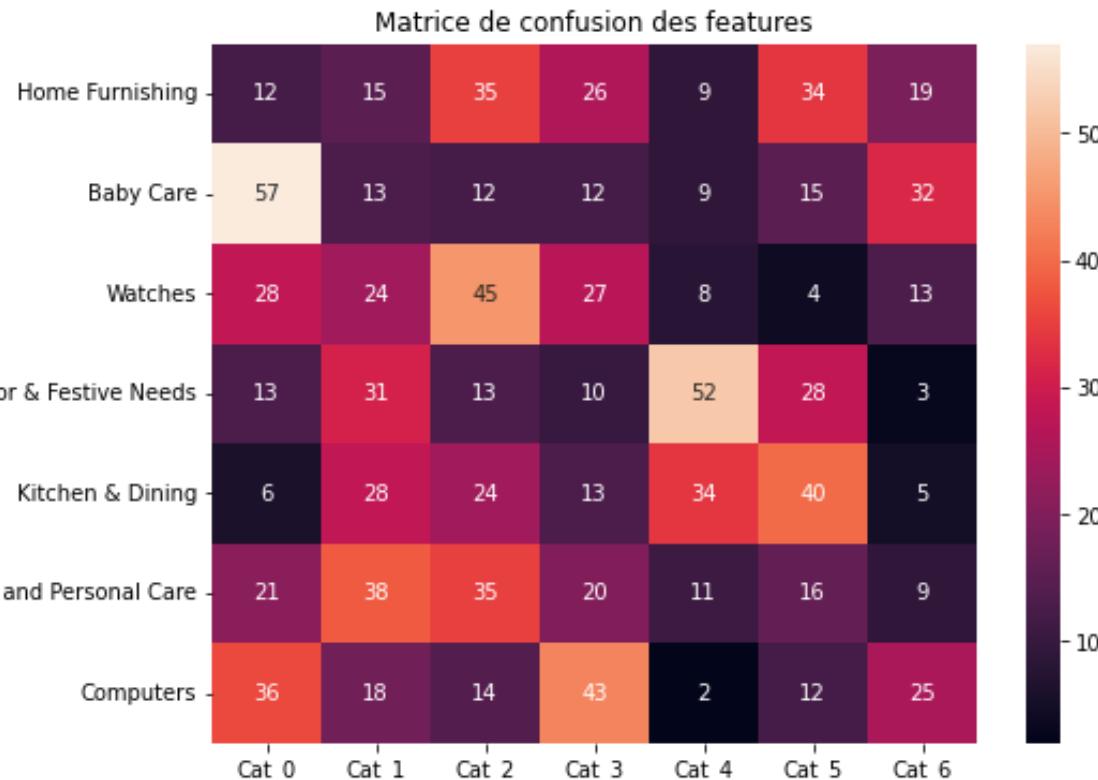
C, Réduction de dimensions et visualisation



Les réductions de dimensions successives ne permettent pas au modèle de distinguer

I, MISE EN PLACE D'UN CLASSIFICATEUR BASÉ SUR LES FEATURES

D, Prédiction



Coefficient ARI : 0,05

Mesure de similarité entre deux groupes.

La matrice et le coefficient ARI nous montrent que le modèle est **incapable d'identifier des groupes distincts**.

En utilisant cette méthode, nous ne pouvons mettre en place un classificateur automatique se basant sur les images.

Afin d'essayer de trouver une solution, nous allons nous tourner vers les **réseaux de neurones**.

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

A, Convolutional neural network

Préparation des données

Redimensionnement des images
selon les dimensions des plus
petites du jeu de donnés.

Largeur : 160 px
Hauteur : 145 px



II,MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

A, Convolutional neural network

```
cnn.summary()

Model: "sequential"

Layer (type)          Output Shape       Param #
=====
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 145, 160, 16)	448
average_pooling2d (AveragePo	(None, 72, 80, 16)	0
conv2d_1 (Conv2D)	(None, 72, 80, 32)	4640
average_pooling2d_1 (Average	(None, 36, 40, 32)	0
conv2d_2 (Conv2D)	(None, 34, 38, 64)	18496
average_pooling2d_2 (Average	(None, 17, 19, 64)	0
flatten (Flatten)	(None, 20672)	0
dense (Dense)	(None, 128)	2646144
dense_1 (Dense)	(None, 7)	903

```
=====
Total params: 2,670,631
Trainable params: 2,670,631
Non-trainable params: 0
```

```
Epoch 19/20
27/27 [=====] - 1s 36ms/step - loss: 0.0637 - accuracy: 0.9798
Epoch 20/20
27/27 [=====] - 1s 35ms/step - loss: 0.0339 - accuracy: 0.9952
```

Réseau de neurone convolutif avec **3 couches convolutives.**

(qui filtrent l'image
 $(m-f+1) \times (n-f+1)$ pour une image $m \times n$ et un filtre $f \times f$
ensuite regroupement avec moyenne (pooling))

99,5% taux de précision à
l'entraînement ce qui fait présager un
bon score pour les prédictions.

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

A, Convolutional neural network

Coefficient ARI : 0,19

Mesure de similarité entre deux groupes.

52% de résultats corrects sur le jeu de test.

Cela montre un sérieux **overfitting** du CNN.

Le CNN fait **bien mieux** que le modèle traditionnel mis en place précédemment.

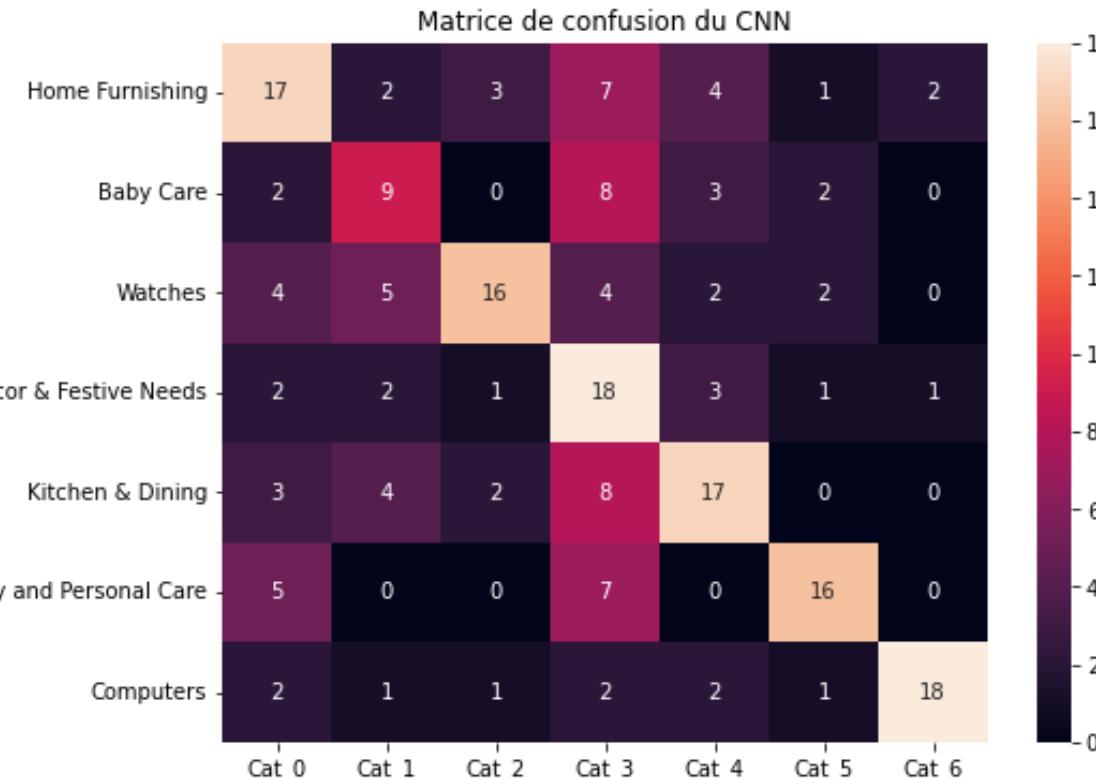
La matrice de confusion laisse apparaître une

diagonale et certaines catégories se distinguent relativement bien.

Cependant, la précision des prédictions n'est **pas encore suffisante** pour construire un modèle.

```
▶ cnn.evaluate(X_test, y_test)
```

```
7/7 [=====] - 0s 39ms/step - loss: 5.1208 - accuracy: 0.5286
```



II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, CNN avec data augmentation

```
cnn_da.summary()  
Model: "sequential_2"  


| Layer (type)                  | Output Shape         | Param # |
|-------------------------------|----------------------|---------|
| sequential_1 (Sequential)     | (None, 145, 160, 3)  | 0       |
| conv2d_3 (Conv2D)             | (None, 145, 160, 16) | 448     |
| average_pooling2d_3 (Average) | (None, 72, 80, 16)   | 0       |
| conv2d_4 (Conv2D)             | (None, 72, 80, 32)   | 4640    |
| average_pooling2d_4 (Average) | (None, 36, 40, 32)   | 0       |
| conv2d_5 (Conv2D)             | (None, 34, 38, 64)   | 18496   |
| average_pooling2d_5 (Average) | (None, 17, 19, 64)   | 0       |
| dropout (Dropout)             | (None, 17, 19, 64)   | 0       |
| flatten_1 (Flatten)           | (None, 20672)        | 0       |
| dense_2 (Dense)               | (None, 128)          | 2646144 |
| dense_3 (Dense)               | (None, 7)            | 903     |

  
Total params: 2,670,631  
Trainable params: 2,670,631  
Non-trainable params: 0
```

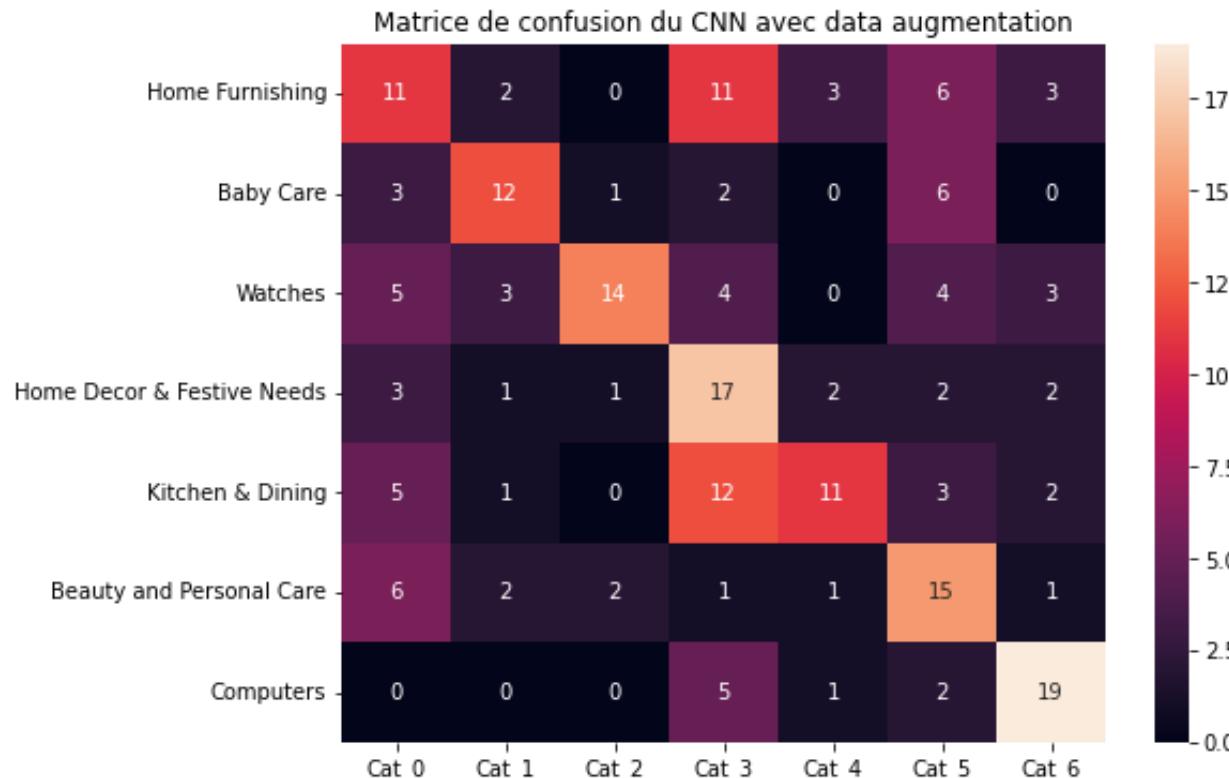
```
data_augmentation = keras.Sequential([  
    layers.experimental.preprocessing.RandomFlip('horizontal', input_shape=(min_height, min_width, 3)),  
    layers.experimental.preprocessing.RandomRotation(0.1),  
    layers.experimental.preprocessing.RandomZoom(0.1),  
])
```

Solutions pour limiter l'overfitting :

- une couche de **transformation** des images
(retournement, rotation et zoom)
- une couche de **décharge** afin de supprimer de l'entraînement certaines images aléatoirement

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

B, CNN avec data augmentation



Coefficient ARI : 0,15

Mesure de similarité entre deux groupes.

47% de résultats corrects sur le jeu de test.

L'optimisation de notre réseau de neurone avec de l'ajout de data augmentation n'a pas permis d'amélioration des résultats.

Il nous reste une dernière approche : le **transfer learning**

```
cnn_da.evaluate(X_test, y_test)
```

```
7/7 [=====] - 0s 17ms/step - loss: 2.6146 - accuracy: 0.4714
```

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

C, CNN avec transfer learning

L'idée est de se servir de réseaux de neurones déjà entraînés (avec des poids et biais établis) et d'adapter leur architecture à notre problématique.

Ici, nous utilisons le VGG16 auquel nous retirons la dernière couche de neurones pour la remplacer par la notre.

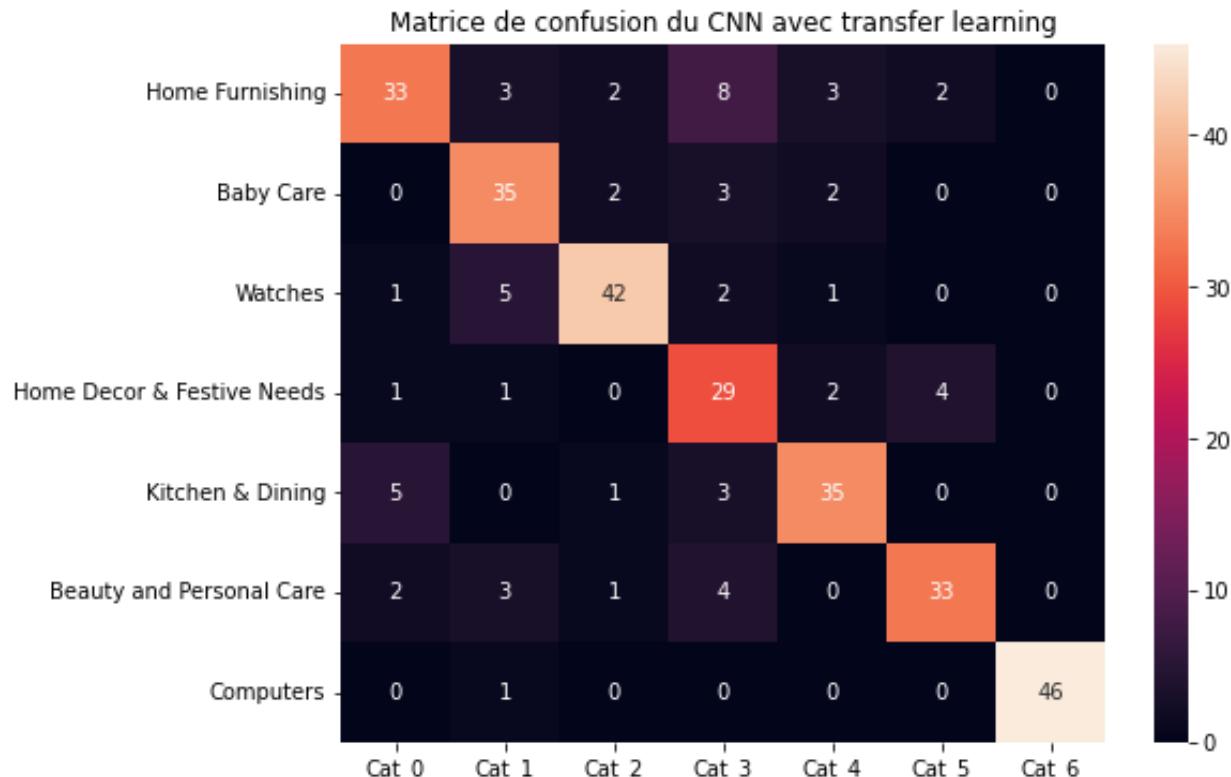
Ainsi dans notre modèle définitif, nous avons **14 millions** de poids non modifiables (issus de VGG16) et **175000** poids entraînables.

Ce modèle est beaucoup plus **complexe** que les précédant et son entraînement n'aurait pas été possible en **local**.

Model: "model"		
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_2 (Flatten)	(None, 25088)	0
dense_4 (Dense)	(None, 7)	175623
<hr/>		
Total params: 14,890,311		
Trainable params: 175,623		
Non-trainable params: 14,714,688		

II, MISE EN PLACE D'UN MODÈLE DE DEEP LEARNING

C, CNN avec transfer learning



Coefficient ARI : 0,61

Mesure de similarité entre deux groupes.

80% de résultats corrects sur le jeu de test.

La matrice de confusion fait apparaître une **diagonale nette**.

Les essais sur le jeu de test et le coefficient ARI nous montrent que le modèle **distingue clairement les groupes** et est capable de faire des **prédictions pertinentes**.

```
cnn_t1.evaluate(x_test, y_test)
```

```
10/10 [=====] - 9s 904ms/step - loss: 0.6335 - accuracy: 0.8032
```

CONCLUSION : MISE EN PLACE D'UN MODÈLE

Nous gardons les modèles les plus pertinent de chaque partie :

- RNN simple pour le texte
- CNN transfer learning pour les images

Nous les cumulons en faisant la moyenne des probabilités proposées par chaque réseau et en ne gardant que la plus importante.

0,99 coefficient ARI du modèle élaboré.

2 erreurs sur un jeu de test de **315** produits.

Le modèle apparaît suffisamment fiable pour être utilisé.

