Classifiez automatiquement des biens de consommation

PROJET 06/ Openclassrooms

Gulsum Kapanoglu

Dans ce Project..

- ✓Problématique
- ✓Faisabilité de classification – Texte
- ✓Prétraitement des données Texte
- ✓Faisabilité de classification – Text Embedding
- ✓Prétraitement des données Images
- ✓Faisabilité de classification automatique d'images via SIFT, ORB
- ✓Faisabilité de classification automatique d'images via CNN
- ✓Conclusion

Problématique

Problématique

"**Place de marché**”, qui souhaite lancer une marketplace e-commerce.

Sur la place de marché, des vendeurs proposent des articles à des acheteurs en postant une photo et une description.

Pour l'instant, l'attribution de la catégorie d'un article est effectuée manuellement par les vendeurs, et est donc peu fiable. De plus, le volume des articles est pour l’instant très petit.

L’objectif

- Pour rendre l’expérience utilisateur des vendeurs (faciliter la mise en ligne de nouveaux articles) et des acheteurs (faciliter la recherche de produits) la plus fluide possible, et dans l'optique d'un passage à l'échelle, **il devient nécessaire d'automatiser cette tâche**.

- **Objectif est** d'étudier la faisabilité d'un **moteur de classification** des articles en différentes catégories, avec un niveau de précision suffisant.

Faisabilité de classification – Texte

Prétraitement des données Texte

Prétraitement de texte

Nettoyage de Texte

In [231]:
```
%time
import contractions

# Test
test_text = """
            Y'all can't expand contractions I'd think. I'd like to know how I'd done that!
            We're going to the zoo and I don't think I'll be home for dinner.
            Theyre going to the zoo and she'll be home for dinner.
            We should've do it in here but we shouldn't've eat it
            """
print("Test: ", contractions.fix(test_text))
```

```
CPU times: total: 0 ns
Wall time: 0 ns
Test:
            You all cannot expand contractions I would think. I would like to know how I would done that!
            We are going to the zoo and I do not think I will be home for dinner.
            They Are going to the zoo and she will be home for dinner.
            We should have do it in here but we should not have eat it
```

Prétraitement de texte

TOKENIZATION

In [241]:
```
# check
print(data['text'][31])
print(data['sentence_bow'][31])
```

```
Lenco Bdblue Tango Analog Watch  - For Men, BoysLenco Bdblue Tango Analog Watch  - For Men, Boys - Buy Lenco Bdblue Tango Analo
g Watch  - For Men, Boys  Bdblue Online at Rs.599 in India Only at Flipkart.com. Sports Watch, Casual Watch, Big Size Dial, Col
ourful Strap - Great Discounts, Only Genuine Products, 30 Day Replacement Guarantee, Free Shipping. Cash On Delivery!
['Lenco', 'Bdblue', 'Tango', 'Analog', 'Watch', 'For', 'Men', ',', 'BoysLenco', 'Bdblue', 'Tango', 'Analog', 'Watch', 'For', 'M
en', ',', 'Boys', 'Buy', 'Lenco', 'Bdblue', 'Tango', 'Analog', 'Watch', 'For', 'Men', ',', 'Boys', 'Bdblue', 'Online', 'at', 'R
s.599', 'in', 'India', 'Only', 'at', 'Flipkart.com', '.', 'Sports', 'Watch', ',', 'Casual', 'Watch', ',', 'Big', 'Size', 'Dia
l', ',', 'Colourful', 'Strap', 'Great', 'Discounts', ',', 'Only', 'Genuine', 'Products', ',', '30', 'Day', 'Replacement', 'Guar
antee', ',', 'Free', 'Shipping', '.', 'Cash', 'On', 'Delivery', '!']
```

Prétraitement de texte

STOPWORDS


```
data['sentence_bow'] = data['text'].apply(lambda x : stop_word_filter_fct(x))
```

```
# check
print(data['text'][31])
print(data['sentence_bow'][31])
```

```
Lenco Bdblue Tango Analog Watch  - For Men, BoysLenco Bdblue Tango Analog Watch  - For Men, Boys - Buy Lenco Bdblue Tango Analo
g Watch  - For Men, Boys  Bdblue Online at Rs.599 in India Only at Flipkart.com. Sports Watch, Casual Watch, Big Size Dial, Col
ourful Strap - Great Discounts, Only Genuine Products, 30 Day Replacement Guarantee, Free Shipping. Cash On Delivery!
[]
```

Prétraitement de texte

STOP WORDS AND LOWER


```
: # stop_words_filter_fct + lower_fct
all_texts = list() # a list of all (non unique) words in descriptions

texts = data['text'].apply(tokenizer_fct)
texts = texts.apply(stop_word_filter_fct)
texts = texts.apply(lower_start_fct)

for i in texts:
  all_texts.extend(i)

freq = FreqDist(all_texts)

# plot wordcloud
plt.figure(figsize=(7,7))
wordcloud = WordCloud(width=300, height=300).generate_from_frequencies(freq)
plt.imshow(wordcloud)
```

```
: <matplotlib.image.AxesImage at 0x1eb3e6b8f10>
```

Prétraitement de texte

- **Préparation du texte pour le bag of words (Countvectorizer et Tf_idf, Word2Vec):**
  i. **Appliqué; tokenizer_fct, stop_word,lower case**

In [258]:
```
# check
print(data['text'][31])
print(data['sentence_bow'][31])
```

```
Lenco Bdblue Tango Analog Watch  - For Men, BoysLenco Bdblue Tango Analog Watch  - For Men, Boys - Buy Lenco Bdblue Tango Analo
g Watch  - For Men, Boys  Bdblue Online at Rs.599 in India Only at Flipkart.com. Sports Watch, Casual Watch, Big Size Dial, Col
ourful Strap - Great Discounts, Only Genuine Products, 30 Day Replacement Guarantee, Free Shipping. Cash On Delivery!
lenco bdblue tango analog watch for men boyslenco bdblue tango analog watch for men boys buy lenco bdblue tango analog watch fo
r men boys bdblue online rs.599 india only flipkart.com sports watch casual watch big size dial colourful strap great discounts
only genuine products day replacement guarantee free shipping cash delivery
```

- **Préparation du texte pour le Bag of Word avec Lemmatisation**
  i. **Appliqué; tokenizer_fct, stop_word,lower case, lemma_fct**

In [262]:
```
# check
print(data['text'][31])

print(data['sentence_bow'][31])

print(data['sentence_bow_lem'][31])
```

```
Lenco Bdblue Tango Analog Watch  - For Men, BoysLenco Bdblue Tango Analog Watch  - For Men, Boys - Buy Lenco Bdblue Tango Analo
g Watch  - For Men, Boys  Bdblue Online at Rs.599 in India Only at Flipkart.com. Sports Watch, Casual Watch, Big Size Dial, Col
ourful Strap - Great Discounts, Only Genuine Products, 30 Day Replacement Guarantee, Free Shipping. Cash On Delivery!
lenco bdblue tango analog watch for men boyslenco bdblue tango analog watch for men boys buy lenco bdblue tango analog watch fo
r men boys bdblue online rs.599 india only flipkart.com sports watch casual watch big size dial colourful strap great discounts
only genuine products day replacement guarantee free shipping cash delivery
lenco bdblue tango analog watch for men boyslenco bdblue tango analog watch for men boy buy lenco bdblue tango analog watch for
men boy bdblue online rs.599 india only flipkart.com sport watch casual watch big size dial colourful strap great discount only
genuine product day replacement guarantee free shipping cash delivery
```

Préparation commune des traitements

7 Catégories

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

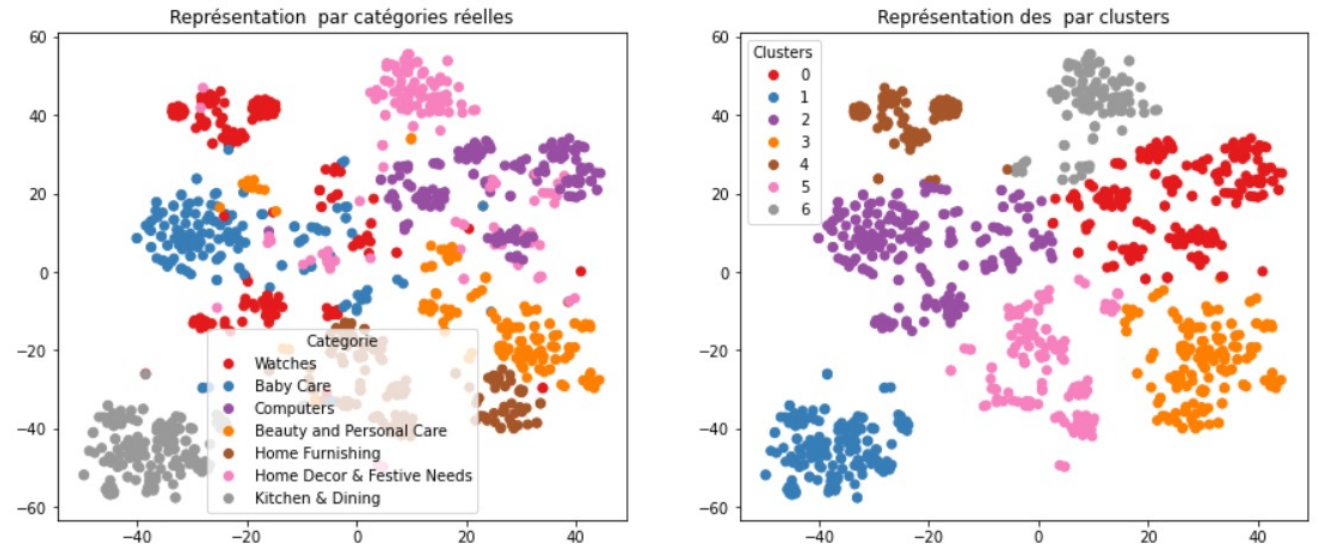
Bag of word - Tf-idf

```
print("CountVectorizer : ")
print("-----------------")
ARI, X_tsne, labels = ARI_fct(cv_transform)
print()
print("Tf-idf : ")
print("--------")
ARI, X_tsne, labels = ARI_fct(ctf_transform)
```

```
CountVectorizer :
-----------------
ARI :  0.4627 time :  11.0

Tf-idf :
--------
ARI :  0.5417 time :  9.0
```

Graphiques

In [274]:
```
TSNE_visu_fct(X_tsne, y_cat_num, labels, ARI)
```




```
ARI :  0.5417
```

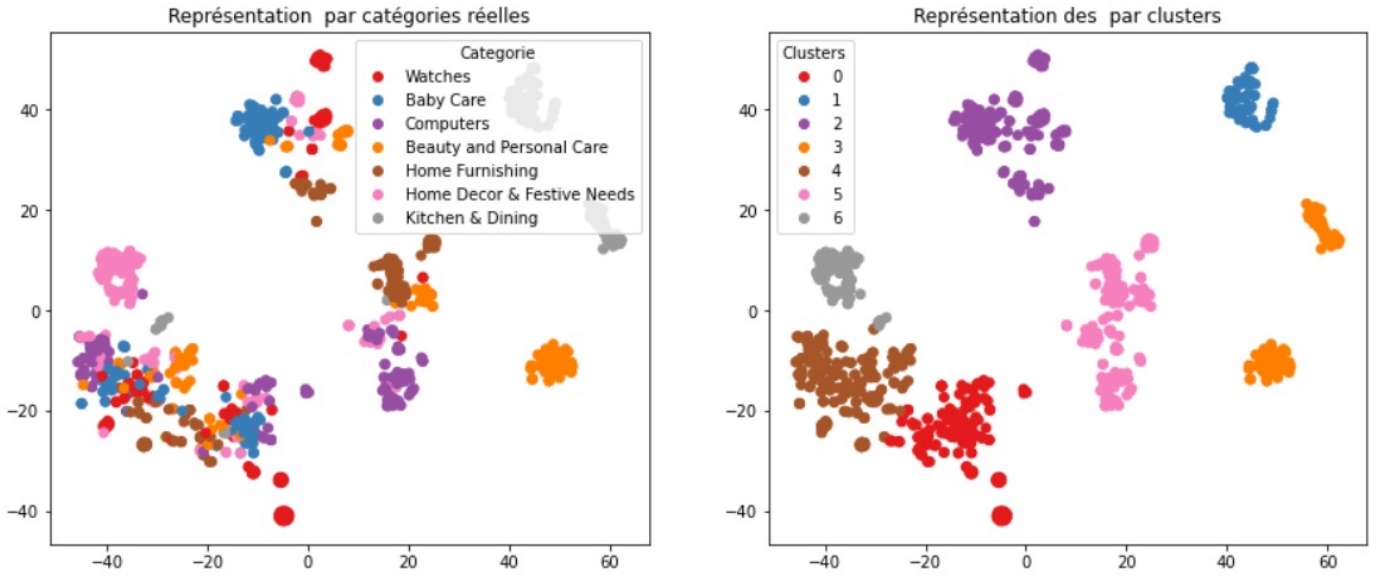
Faisabilité de classification – Text Embedding

Word2Vec

```
ARI, X_tsne, labels = ARI_fct(embeddings)
```

```
ARI :  0.1996 time :  8.0
```

```
TSNE_visu_fct(X_tsne, y_cat_num, labels, ARI)
```



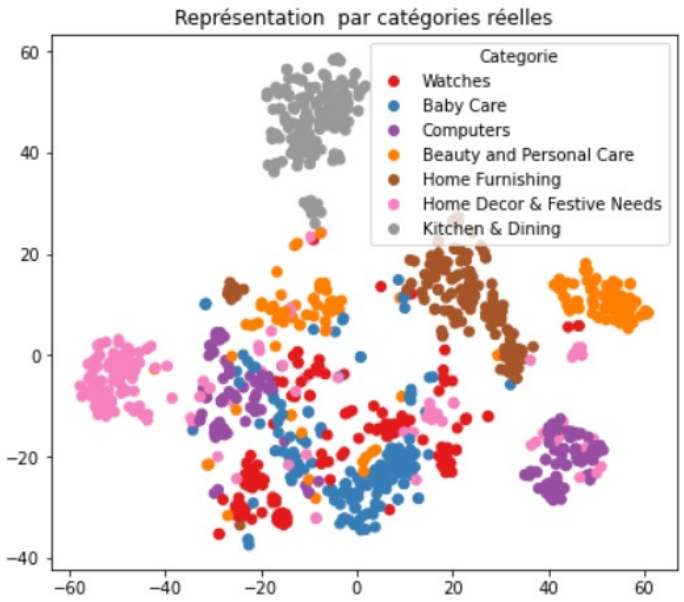

```
ARI :  0.1996
```

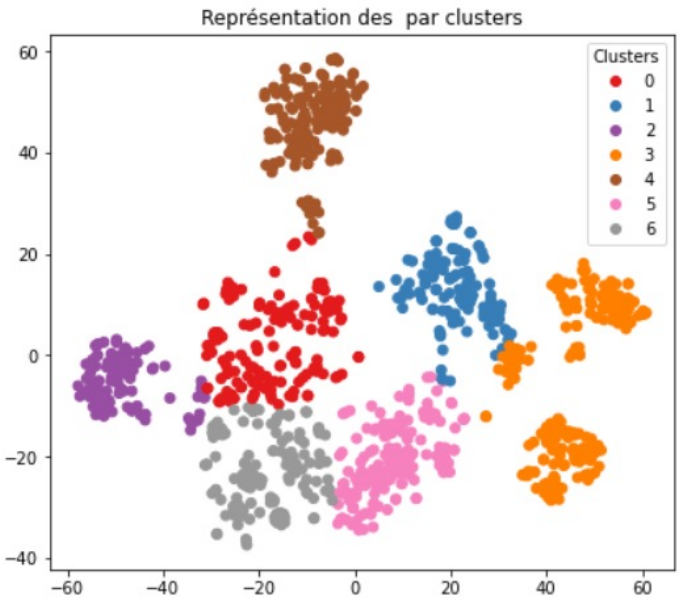
BERT

```
ARI, X_tsne, labels = ARI_fct(features_bert)
```

```
ARI :  0.4013 time :  9.0
```

```
TSNE_visu_fct(X_tsne, y_cat_num, labels, ARI)
```





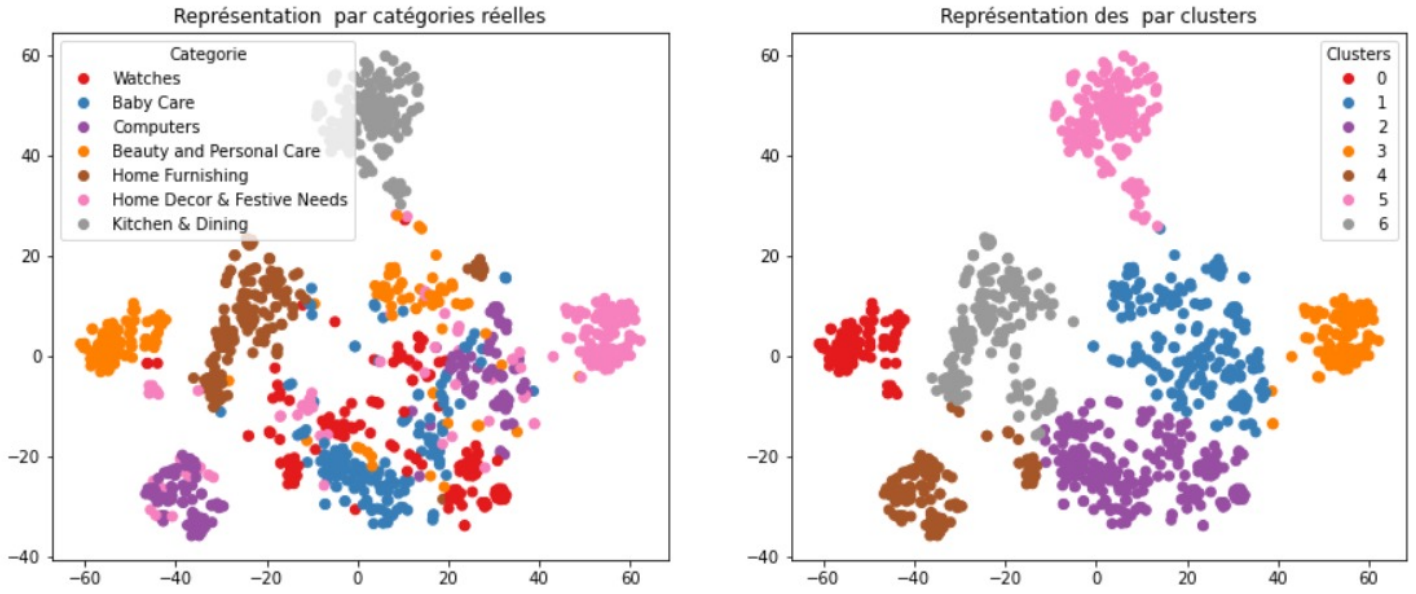

```
ARI :  0.4013
```

BERT hub Tensorflow

```
ARI, X_tsne, labels = ARI_fct(features_bert)
```

```
ARI :  0.4723 time :  9.0
```

```
TSNE_visu_fct(X_tsne, y_cat_num, labels, ARI)
```



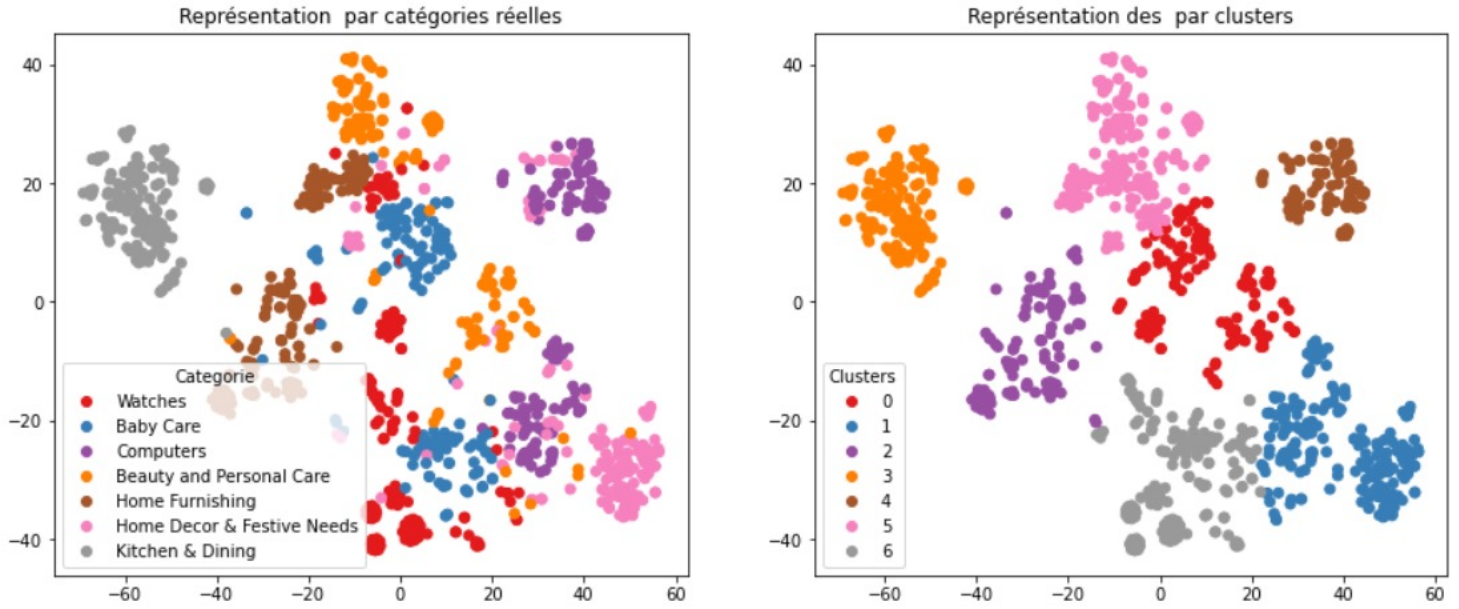

```
ARI :  0.4723
```

USE - Universal Sentence Encoder

```
ARI, X_tsne, labels = ARI_fct(features_USE)
```

```
ARI :  0.4453 time :  10.0
```

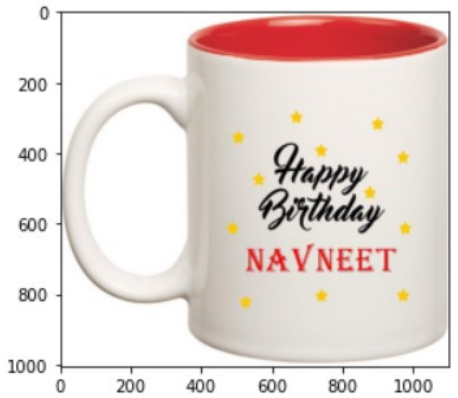
```
TSNE_visu_fct(X_tsne, y_cat_num, labels, ARI)
```



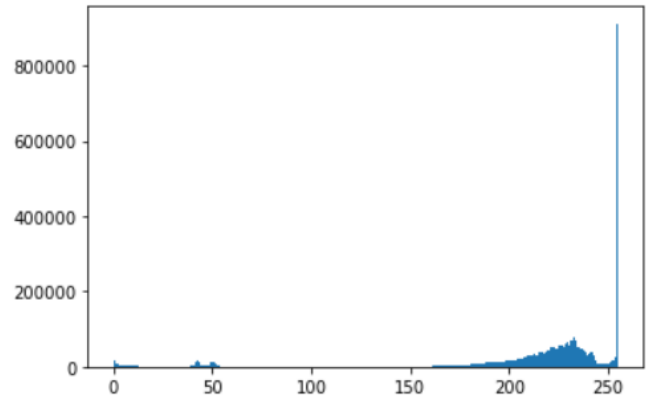

```
ARI :  0.4453
```

Prétraitement des données Images

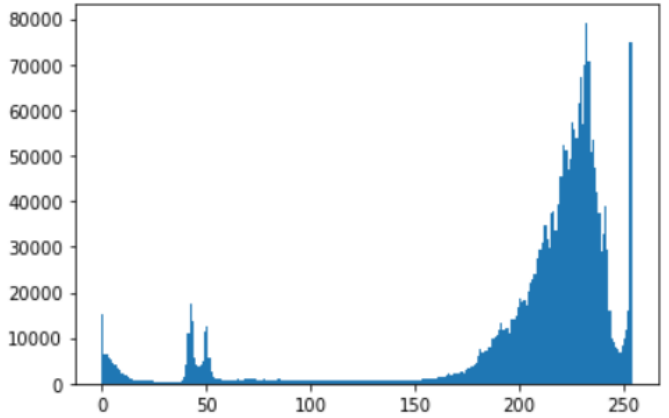
```
from PIL import Image
# Visualiser une image
image = Image.open(path+random.choice(list_photos))
plt.imshow(image)
plt.show()
```




```
import matplotlib.pyplot as plt
img = np.array(image)
n, bins, patches = plt.hist(img.flatten(), bins=range(256))
plt.show()
```

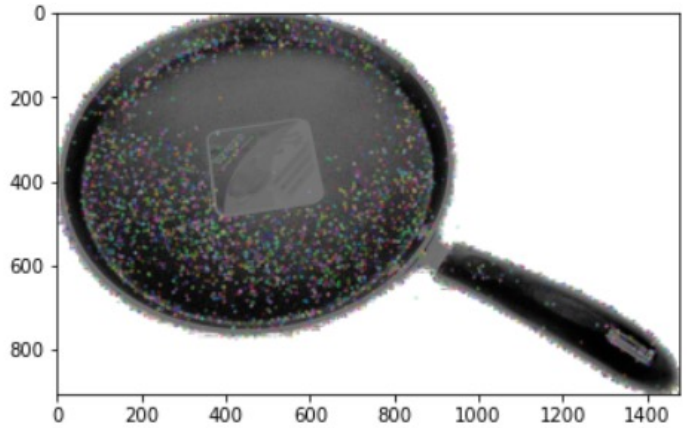


```
img = np.array(image)

n, bins, patches = plt.hist(img.flatten(), bins=range(255))
plt.show()
```



Faisabilité de classification automatique d'images via SIFT, ORB

Détermination et affichage des descripteurs SIFT




```
Descripteurs :  (4656, 128)

[[ 4.  2.  6. ...  5.  0.  0.]
 [96. 20.  0. ...  9.  2.  0.]
 [55. 16.  0. ...  0.  1.  0.]
 ...
 [12.  0.  0. ...  0.  0.  0.]
 [67. 19.  5. ...  0. 34. 13.]
 [ 0.  5. 27. ...  0.  0.  0.]]
```

Prétraitement des images via SIFT

Créations des descripteurs de chaque image

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

```
0
100
200
300
400
500
600
700
800
900
1000

Nombre de descripteurs :  (517351, 128)
temps de traitement SIFT descriptor :           489.44 secondes
```

Création des clusters de descripteurs

- Utilisation de MiniBatchKMeans pour obtenir des temps de traitement raisonnables

```
Nombre de clusters estimés :  719
Création de 719 clusters de descripteurs ...
temps de traitement kmeans :            16.47 secondes
```

- Création des features des images

Pour chaque image :

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

```
0
100
200
300
400
500
600
700
800
900
1000
temps de création histogrammes :          187.26 secondes
```

Réductions de dimension

- **Réduction de dimension PCA**

La réduction PCA permet de créer des features décorrélées entre elles, et de diminuer leur dimension, tout en gardant un niveau de variance expliquée élevé (99%)

L'impact est une meilleure séparation des données via le T-SNE et une réduction du temps de traitement du T-SNE

```
print("Dimensions dataset avant réduction PCA : ", im_features.shape)
pca = decomposition.PCA(n_components=0.99)
feat_pca= pca.fit_transform(im_features)
print("Dimensions dataset après réduction PCA : ", feat_pca.shape)
```

```
Dimensions dataset avant réduction PCA :  (1050, 719)
Dimensions dataset après réduction PCA :  (1050, 498)
```

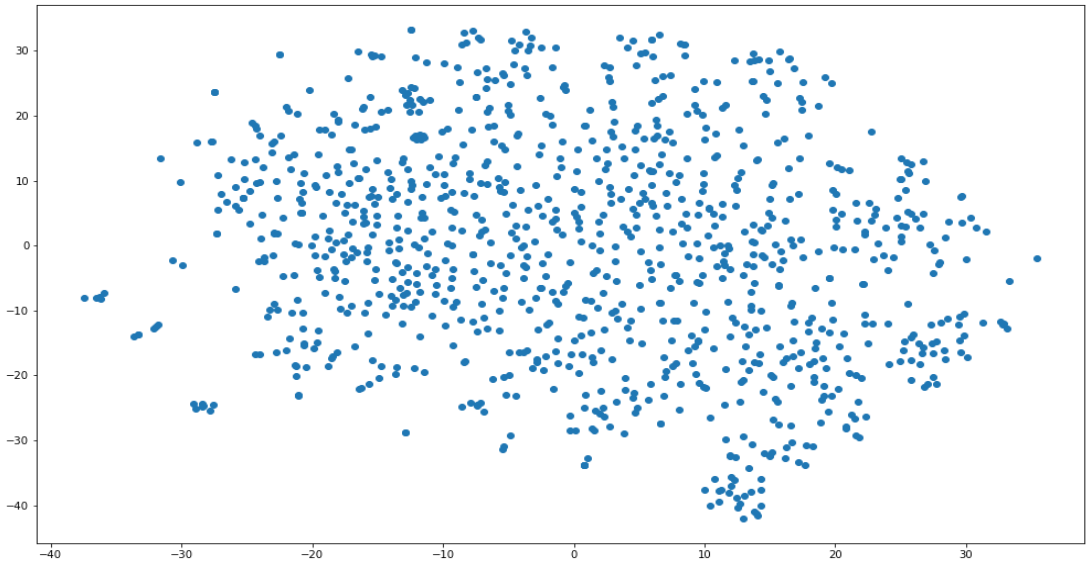
Réductions de dimension

- **Réduction de dimension T-SNE**

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

```
from sklearn.manifold import TSNE
X = im_features
X_embedded = TSNE(n_components=2).fit_transform(X)
X_embedded.shape
```

```
(1050, 2)
```



Détermination et affichage des descripteurs ORB


```
Descripteurs :  (500, 32)

[[162 161 111 ... 130  98 169]
 [163 177  96 ... 142 222 223]
 [ 70  50 173 ...  37 181 115]
 ...
 [ 80  22 254 ... 192  15 123]
 [161 157  25 ...   8 199 248]
 [ 16 169  59 ...  10 215 138]]
```

Chaque image contient 500 descripteurs. 32 répresente la longueur de vecteur de chaque descripteur.

Pré-traitement des images via ORB

Créations des descripteurs de chaque image

- ✓ Pour chaque image passage en gris et equalisation
- ✓ Création d'une liste de descripteurs par image ("orb_keypoints_by_img") qui sera utilisée pour réaliser les histogrammes par image
- ✓ Création d'une liste de descripteurs pour l'ensemble des images ("orb_keypoints_all") qui sera utilisé pour créer les clusters de descripteurs

```
0
100
200
300
400
500
600
700
800
900
1000

Nombre de descripteurs :  (520145, 32)
temps de traitement ORB descriptor :           91.96 secondes
```

Création des clusters de descripteurs

- Utilisation de MiniBatchKMeans pour obtenir des temps de traitement raisonnables

```
Nombre de clusters estimés :  721
Création de 721 clusters de descripteurs ...
temps de traitement kmeans :            27.43 secondes
```

- Création des features des images

Pour chaque image :

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

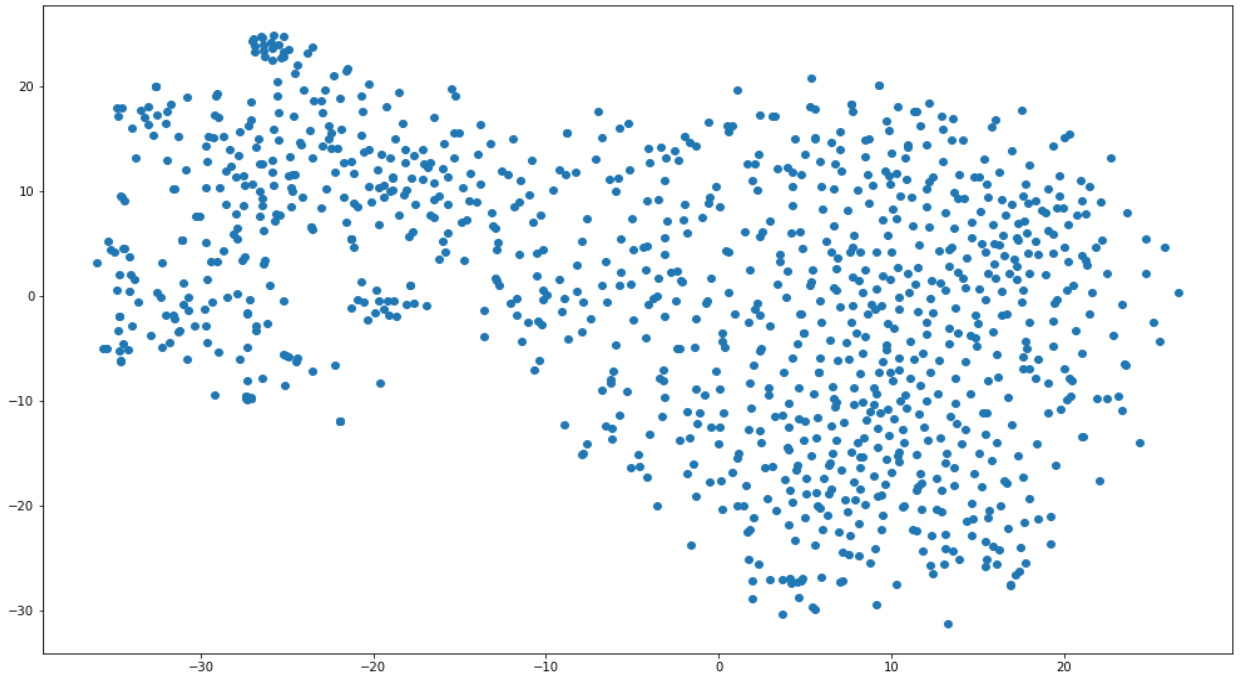
```
0
100
200
300
400
500
600
700
800
900
1000

[[0.    0.    0.004 ... 0.    0.    0.002]
 [0.    0.014 0.    ... 0.032 0.    0.   ]
 [0.    0.    0.    ... 0.    0.002 0.   ]
 ...
 [0.    0.    0.002 ... 0.01  0.    0.   ]
 [0.01  0.    0.    ... 0.    0.    0.002]
 [0.    0.014 0.    ... 0.006 0.012 0.002]]

temps de création histogrammes :          318.69 secondes
```

Réduction de dimension T-SNE

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

```
from sklearn.manifold import TSNE
X = im_features
X_embedded = TSNE(n_components=2).fit_transform(X)
X_embedded.shape
```

```
(1050, 2)
```



Faisabilité de classification automatique d'images via CNN

Transfer Learning

Transfer Learning implémenté dans Keras sur VGG16

- Création des features des images

```
0
100
200
300
400
500
600
700
800
900
1000

temps de création histogrammes :           147.51 secondes

[[0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

Réductions de dimension

- **Réduction de dimension PCA**

```
from sklearn import manifold, decomposition

print("Dimensions dataset avant réduction PCA : ", im_features.shape)
pca = decomposition.PCA(n_components=0.99)
feat_pca= pca.fit_transform(im_features)
print("Dimensions dataset après réduction PCA : ", feat_pca.shape)
```

```
Dimensions dataset avant réduction PCA :  (1050, 32)
Dimensions dataset après réduction PCA :  (1050, 26)
```

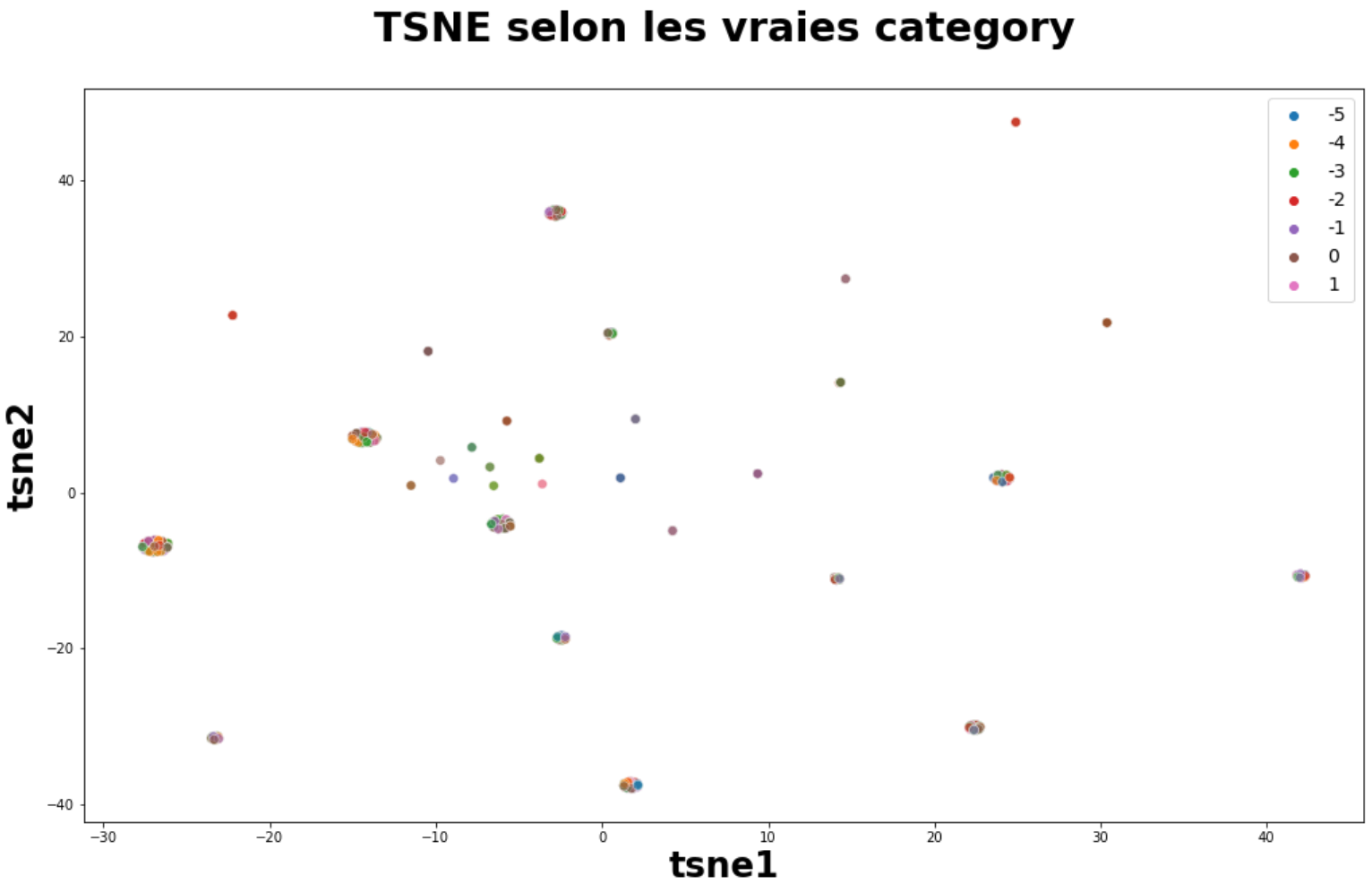
- **Réduction de dimension T-SNE**

```
X = im_features
X_embedded = TSNE(n_components=2).fit_transform(X)
X_embedded.shape
```

```
(1050, 2)
```

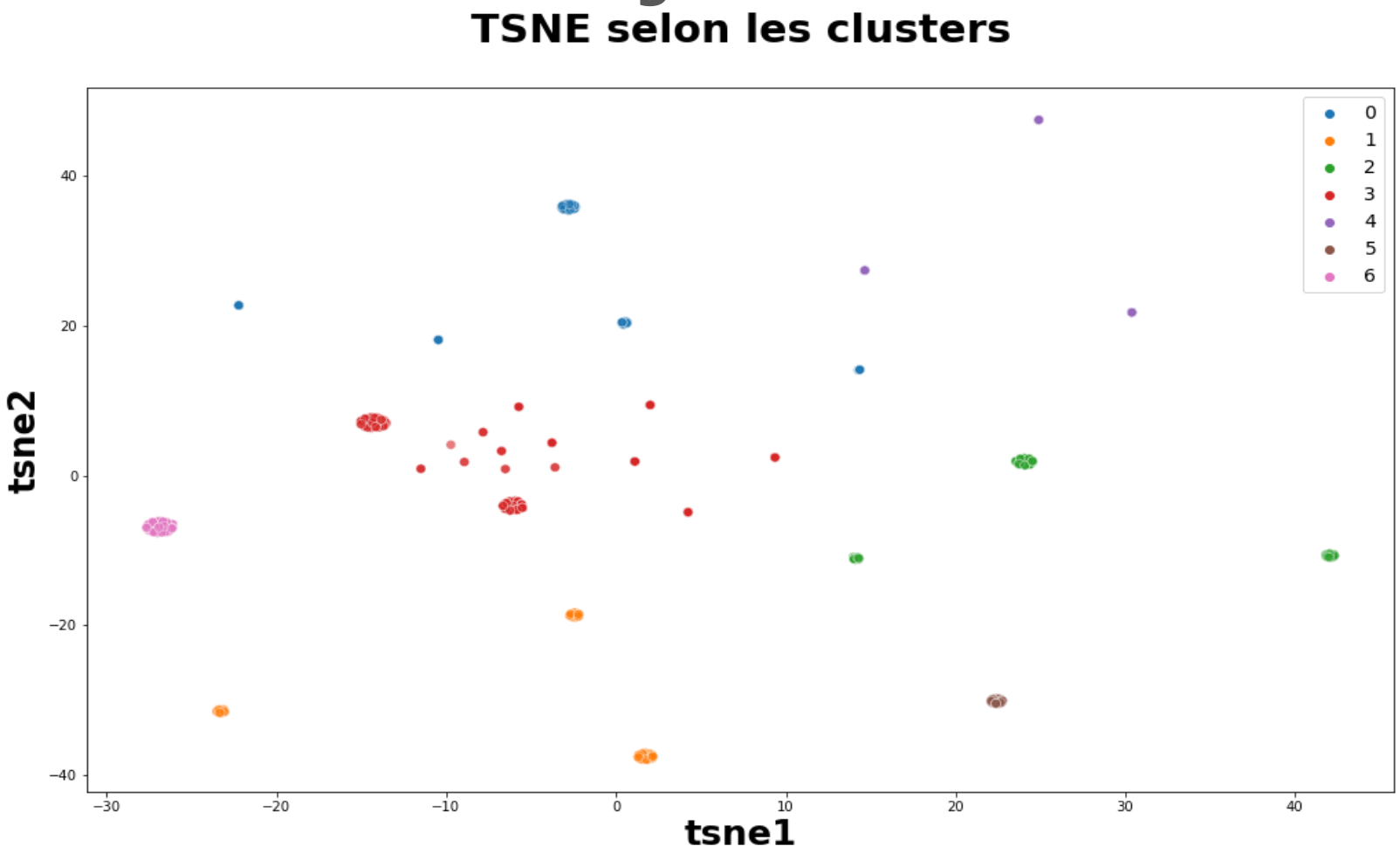
Analyse visuelle

- Affichage T-SNE selon catégories d'images



Analyse mesures

Similarité entre catégories et clusters



Analyse par classes

```
df_tsne.groupby("cluster").count()["category"]
```

```
cluster
0    204
1    168
2    166
3    265
4     99
5     57
6     91
Name: category, dtype: int64
```

7 cluster et 7 category

Conclusion

Conclusion

- Ce projet pour réaliser des algorithmes de classification supervisée, mais de préparer les données de types texte et image, afin de permettre ces traitements.
- Pour les données de types texte ; On obtenir une score(ARI) :0.44. Donc c'est confirme que la faisabilité de classer automatiquement les produits est possible.
- Pour les données de images; On peut pas de résultat probant. il n'est pas possible de distinguer les groupes. Mais la classification automatique peut être effectuée en appliquant des modèles pré-entraînés.

Merci!