CLASSIFICATION AUTO DE BIENS DE CONSOMMATION

INTRODUCTION

_PLAN

1. Introduction:

- Objectifs
- Données

2. Analyse des images:

- Pré-traitement
- Clustering
- CNN
- Transfert Learning

3. Analyse du texte:

- Pré-traitement
- Bag of words
- TFIDF
- Word2Vect

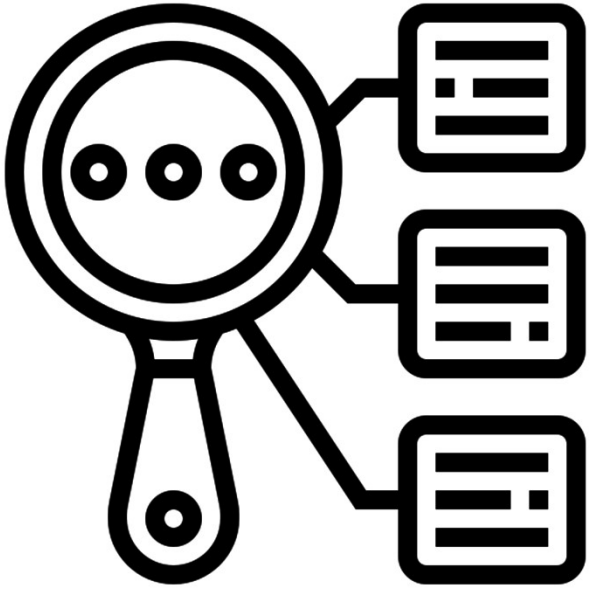
4. Conclusion:

- Voting Classifier
- Faisabilité

INTRODUCTION

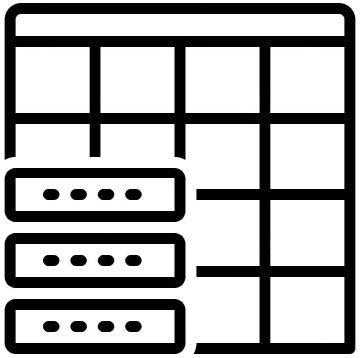
_OBJECTIFS

- Automatiser l'attribution de catégories d'articles
- Étude de faisabilité d'un moteur de classification
- Analyse et traitement sur un jeu d'images
- Analyse et traitement sur un corpus de descriptions



INTRODUCTION

_DONNÉES

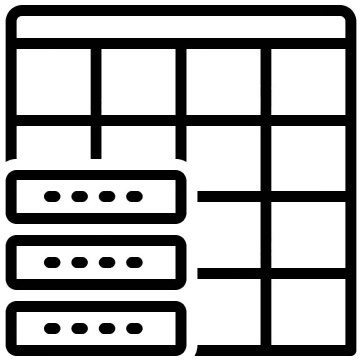


Pas de NaN
Url d'images
Descriptions
Catégories

- Images: ≠ Dimensions – Dossier – Objets
  -> Création d'une liste d'images
- Catégories: – Category tree
  -> Séparation des catégories

INTRODUCTION

_DONNÉES



- **Images**

```
path = "C:/Users/Damien/Desktop/Data Scientist/P6/Dataset/Flipkart/Images512
list_photos = [file for file in listdir(path)]
print(len(list_photos))
```

- **Catégories**

```
df['cat_lvl_1'].unique()
```

```
array(['Home Furnishing', 'Baby Care', 'Watches',
       'Home Decor & Festive Needs', 'Kitchen & Dining',
       'Beauty and Personal Care', 'Computers'], dtype=object)
```

ANALYSE IMAGES

_ PRÉ-TRAITEMENT

IMAGES

Nombre: 1050
Catégories: 7

- **Redimensionnement**
- **Passage au gris**
- **Equalization**
- **Reduction du bruit**

ANALYSE IMAGES

_ CLUSTERING

CLUSTERING

- **Création liste descripteurs : 1000 sift_keypoints**

```
sift_keypoints = []
temps1=time.time()
sift = cv2.SIFT_create(1000)
```

```
Nombre de descripteurs :  (783051, 128)
temps de traitement SIFT descriptor :            29.37 secondes
```

- **Features extraction: Kmeans(n_clusters=√ n_descripteurs)**

```
Nombre de clusters estimés :  885
Création de 885 clusters de descripteurs ...
temps de traitement kmeans :            55.23 secondes
```

ANALYSE IMAGES

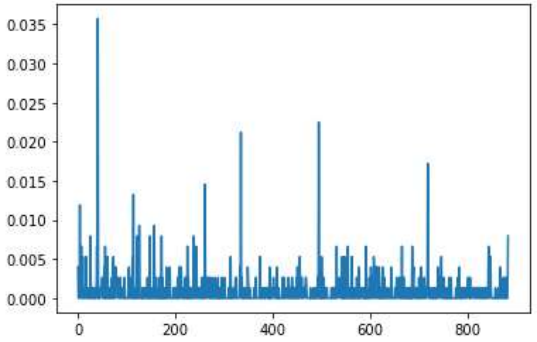
_ CLUSTERING

CLUSTERING

- **Histogramme: vectorisé**

```
1 plt.plot(im_features[90])
```

```
[<matplotlib.lines.Line2D at 0x1fc9fecd730>]
```




```
1 Image(path+list_photos[90], width=200)
```

- **Reduction de dimension: PCA (n_comp=0.99)**

```
Dimensions dataset avant réduction PCA :  (1050, 885)
Dimensions dataset après réduction PCA :  (1050, 609)
```

- **Reduction de dimension: T-SNE(PCA)**

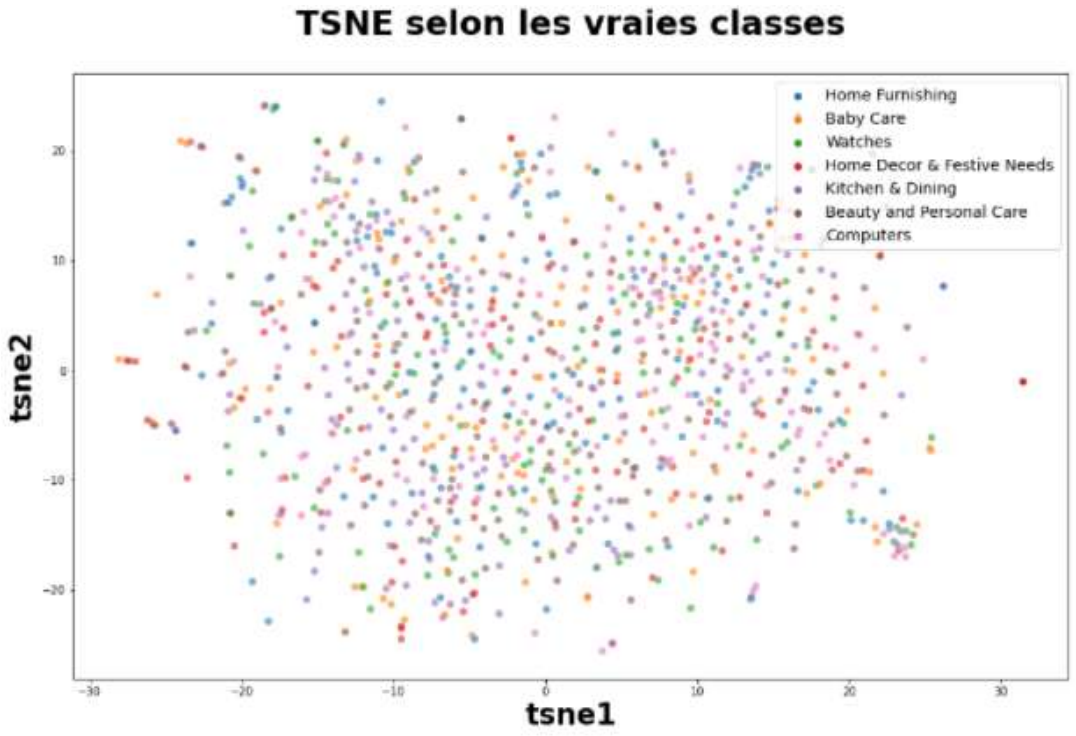
```
(1050, 3)
```

ANALYSE IMAGES

_ CLUSTERING

CLUSTERING

- **Clustering:**



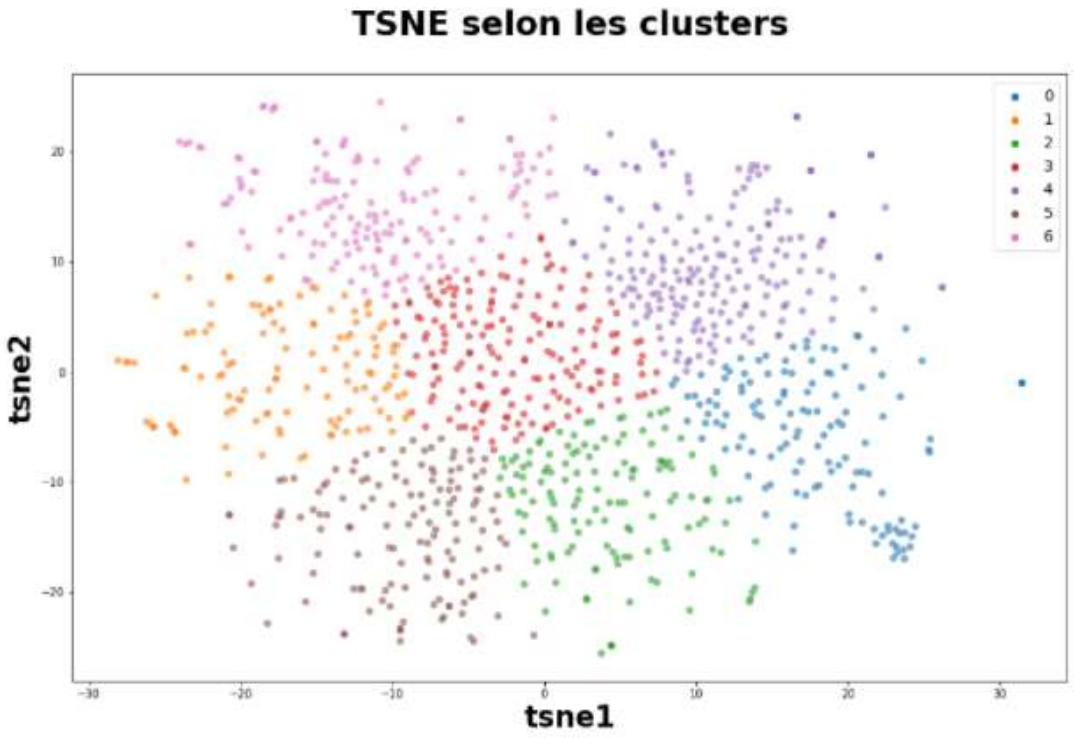
place de marché

ANALYSE IMAGES

_ CLUSTERING

CLUSTERING

- **Clustering:**



ARI : -0.0013680726996754358

ANALYSE IMAGES

_ CLUSTERING

CLUSTERING

- **Clustering:**

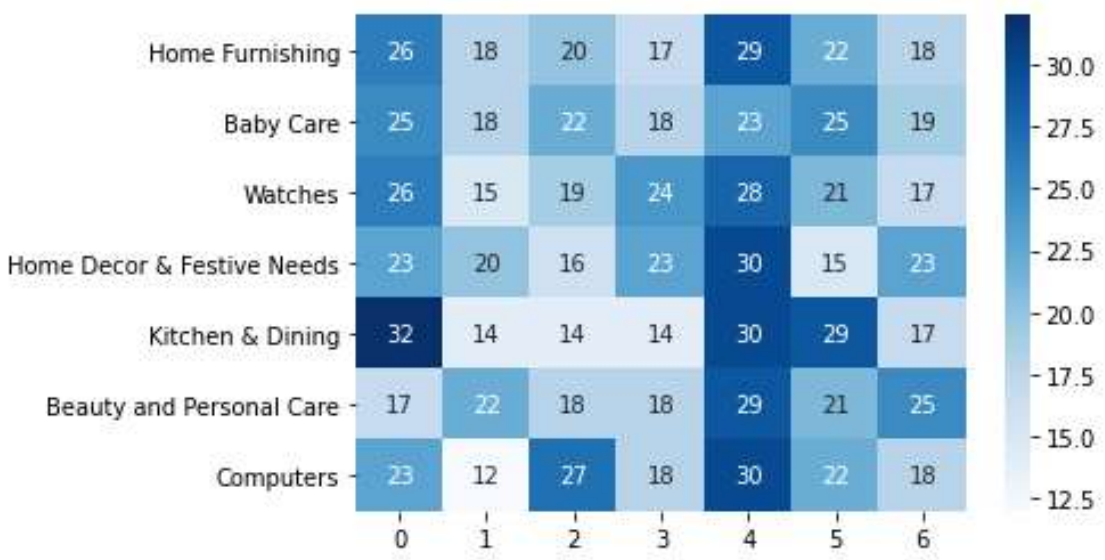
	precision	recall	f1-score	support
0	0.15	0.17	0.16	150
1	0.15	0.12	0.13	150
2	0.14	0.13	0.13	150
3	0.17	0.15	0.16	150
4	0.15	0.20	0.17	150
5	0.14	0.14	0.14	150
6	0.13	0.12	0.13	150
accuracy			0.15	1050
macro avg	0.15	0.15	0.15	1050
weighted avg	0.15	0.15	0.15	1050

ANALYSE IMAGES

_ CLUSTERING

CLUSTERING

- **Clustering:**



ANALYSE IMAGES

_ CNN

PREPARATION

- Données: Train(0.75) / Test(0.25)
- Redimensionnement: 128x128
- Normalisation: Pixels/255
- Transformer: LabelEncoder pour les categories
- Stack

ANALYSE IMAGES

_ CNN

MODEL

- **Keras: Sequentiel**
- **Couches:**

```
model.add(keras.layers.Conv2D(8, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Conv2D(16, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(100, activation='relu'))
model.add(keras.layers.Dropout(0.2))
```

- **Classification: Softmax sur 7 categories**

```

model.add(keras.layers.Dense(num_classes, activation='softmax'))

```

- **Compilation:**

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

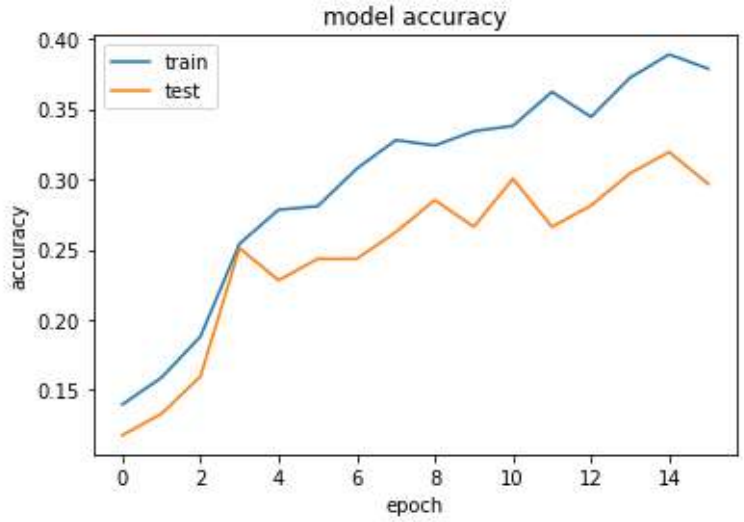
ANALYSE IMAGES

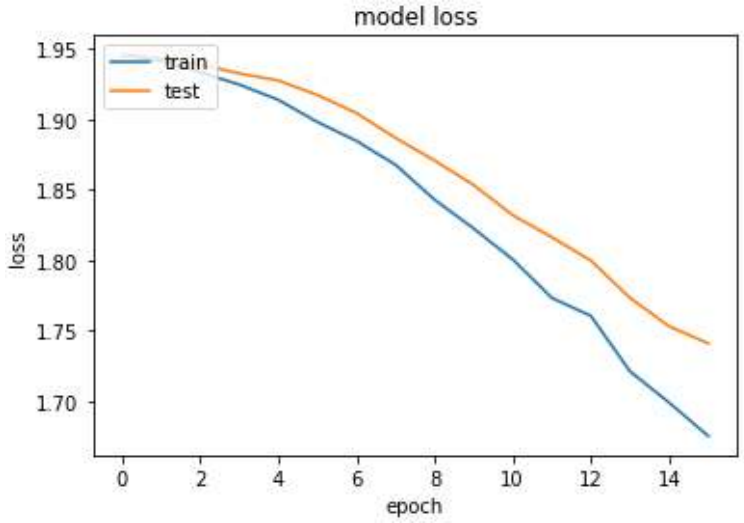
_ CNN

PERFORMANCES

- **Accuracy/Loss:**

```
Test loss     : 1.7412
Test accuracy : 0.2966
```



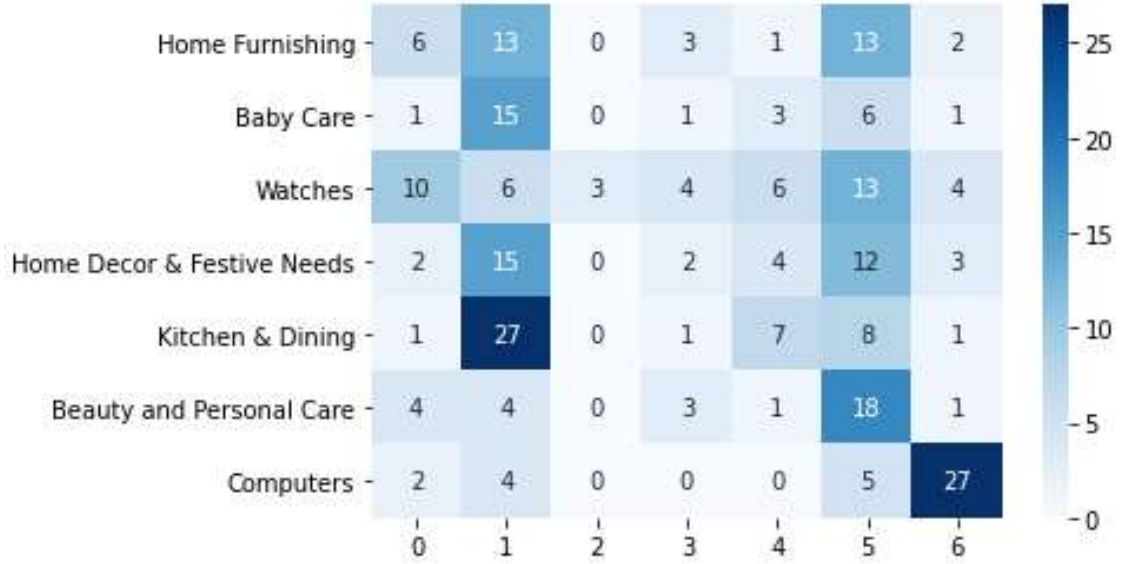


ANALYSE IMAGES

_ CNN

PERFORMANCES

- **Matrice confusion:**



ANALYSE IMAGES

_ TRANSFERT LEARNING

FEATURES EXTRACTOR

- **Model: VGG16(imagenet)**
- **Couches: Sans classification**

```
# Charger VGG16 sans couches de classification (include_top=False)
VGG_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
```

- **Sans entrainement**

```
for layer in VGG_model.layers:
    layer.trainable = False
```

```
=================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________
```

ANALYSE IMAGES

_ TRANSFERT LEARNING

MODEL

- **Model: RandomForest**
- **GridSearch: n_estimator, min_samples_leaf, max_features**

```
# Charger VGG16 sans couches de classification (include_top=False)
VGG_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
```

- **Accuracy**

```
Accuracy = 0.7984790874524715
```

ANALYSE IMAGES

_ TRANSFERT LEARNING

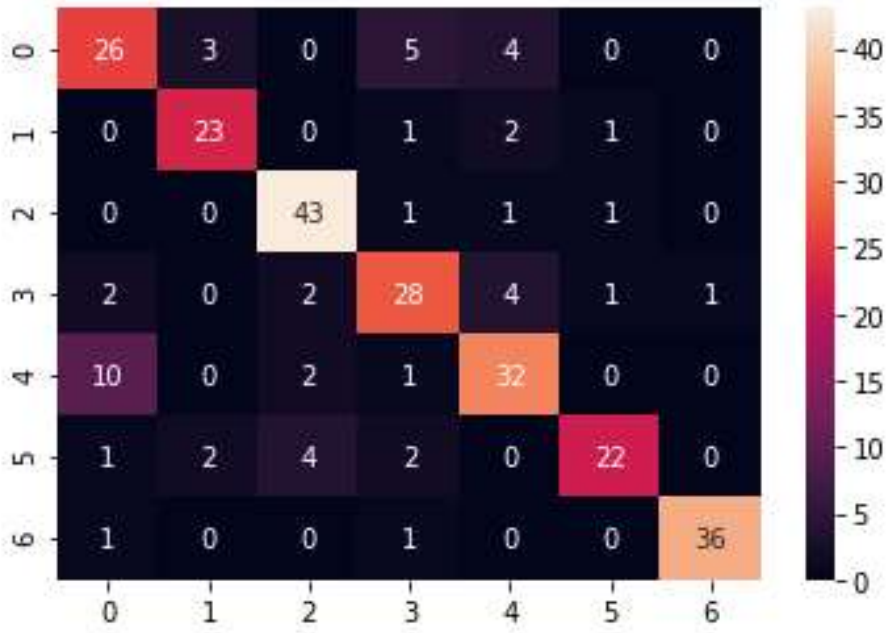
PERFORMANCES

	precision	recall	f1-score	support
Baby Care	0.65	0.68	0.67	38
Beauty and Personal Care	0.82	0.85	0.84	27
Computers	0.84	0.93	0.89	46
Home Decor & Festive Needs	0.72	0.74	0.73	38
Home Furnishing	0.74	0.71	0.73	45
Kitchen & Dining	0.88	0.71	0.79	31
Watches	0.97	0.95	0.96	38
accuracy			0.80	263
macro avg	0.80	0.80	0.80	263
weighted avg	0.80	0.80	0.80	263

ANALYSE IMAGES

_ TRANSFERT LEARNING

PERFORMANCES



ANALYSE DU TEXTE
_PRÉ-TRAITEMENT

TEXTE

Phrases: 1050
Mots: 80013

- **Lowercase**
- **Tokenizer**
- **Stopwords**
- **Lemmatisation**

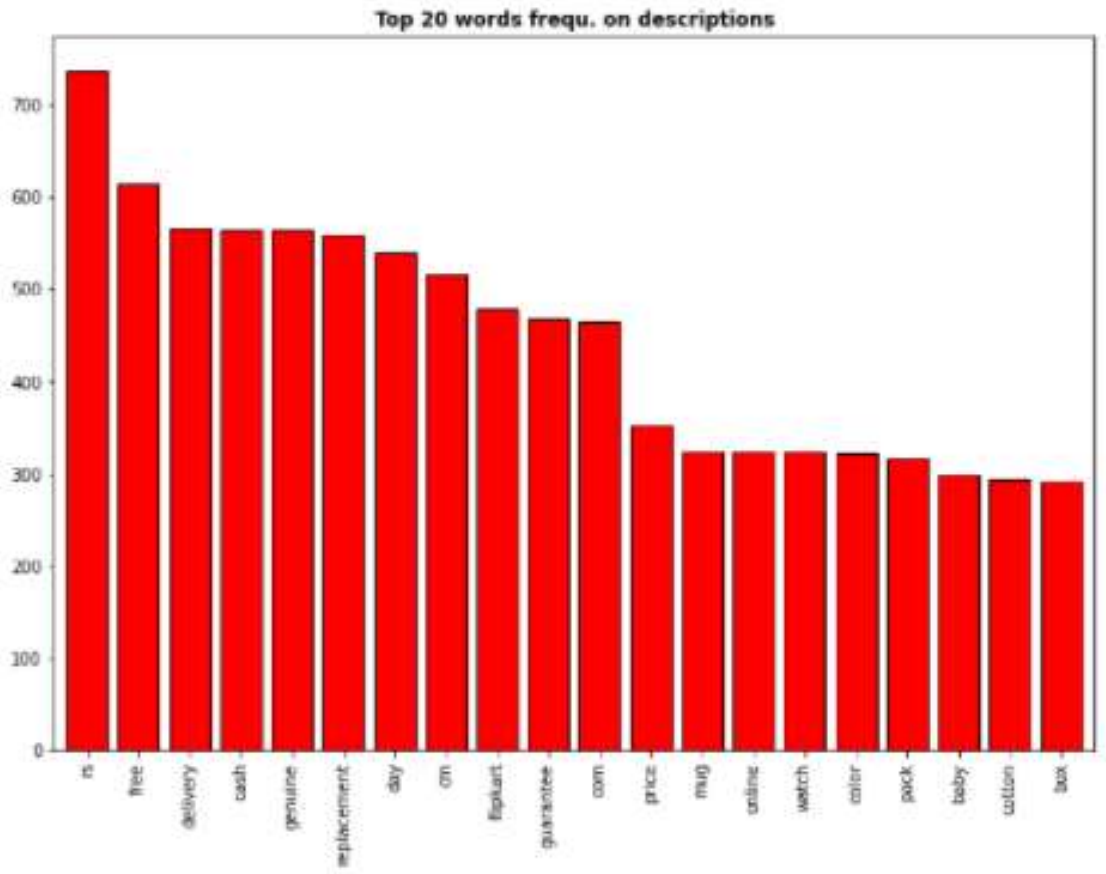
ANALYSE DU TEXTE
_PRÉ-TRAITEMENT

TEXTE

```
======== PRE TRAITEMENT ========
Key Features of Mom and Kid Baby Girl's Printed Green Top & Pyjama Set Fabric: Cotton Brand Color: Green,Mom and Kid Baby Gir
l's Printed Green Top & Pyjama Set Price: Rs. 309 Girls Pyjama set,Specifications of Mom and Kid Baby Girl's Printed Green Top
& Pyjama Set General Details Pattern Printed Ideal For Baby Girl's Night Suit Details Fabric Cotton Type Top & Pyjama Set Neck
Round Neck In the Box 1 Top & Pyjama Set

======== LOWERCASE ========
key features of mom and kid baby girl's printed green top & pyjama set fabric: cotton brand color: green,mom and kid baby gir
l's printed green top & pyjama set price: rs. 309 girls pyjama set,specifications of mom and kid baby girl's printed green top
& pyjama set general details pattern printed ideal for baby girl's night suit details fabric cotton type top & pyjama set neck
round neck in the box 1 top & pyjama set

======== TOKENIZER ========
['key', 'features', 'of', 'mom', 'and', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', '&', 'pyjama', 'set', 'fabric',
':', 'cotton', 'brand', 'color', ':', 'green', ',', 'mom', 'and', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', '&',
'pyjama', 'set', 'price', ':', 'rs', '.', '309', 'girls', 'pyjama', 'set', ',', 'specifications', 'of', 'mom', 'and', 'kid', 'b
aby', 'girl', "'s", 'printed', 'green', 'top', '&', 'pyjama', 'set', 'general', 'details', 'pattern', 'printed', 'ideal', 'fo
r', 'baby', 'girl', "'s", 'night', 'suit', 'details', 'fabric', 'cotton', 'type', 'top', '&', 'pyjama', 'set', 'neck', 'round',
'neck', 'in', 'the', 'box', '1', 'top', '&', 'pyjama', 'set']

======== STOPWORDS ========
['key', 'features', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'fabric', 'cotton', 'bran
d', 'color', 'green', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'price', 'rs', '309', 'gi
rls', 'pyjama', 'set', 'specifications', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'gener
al', 'details', 'pattern', 'printed', 'ideal', 'baby', 'girl', "'s", 'night', 'suit', 'details', 'fabric', 'cotton', 'type', 't
op', 'pyjama', 'set', 'neck', 'round', 'neck', 'box', '1', 'top', 'pyjama', 'set']

======== LEMMATISATION ========
['key', 'feature', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'fabric', 'cotton', 'brand',
'color', 'green', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'price', 'r', '309', 'girl',
'pyjama', 'set', 'specification', 'mom', 'kid', 'baby', 'girl', "'s", 'printed', 'green', 'top', 'pyjama', 'set', 'general', 'd
etail', 'pattern', 'printed', 'ideal', 'baby', 'girl', "'s", 'night', 'suit', 'detail', 'fabric', 'cotton', 'type', 'top', 'pyj
ama', 'set', 'neck', 'round', 'neck', 'box', '1', 'top', 'pyjama', 'set']
```

ANALYSE DU TEXTE
_PRÉ-TRAITEMENT

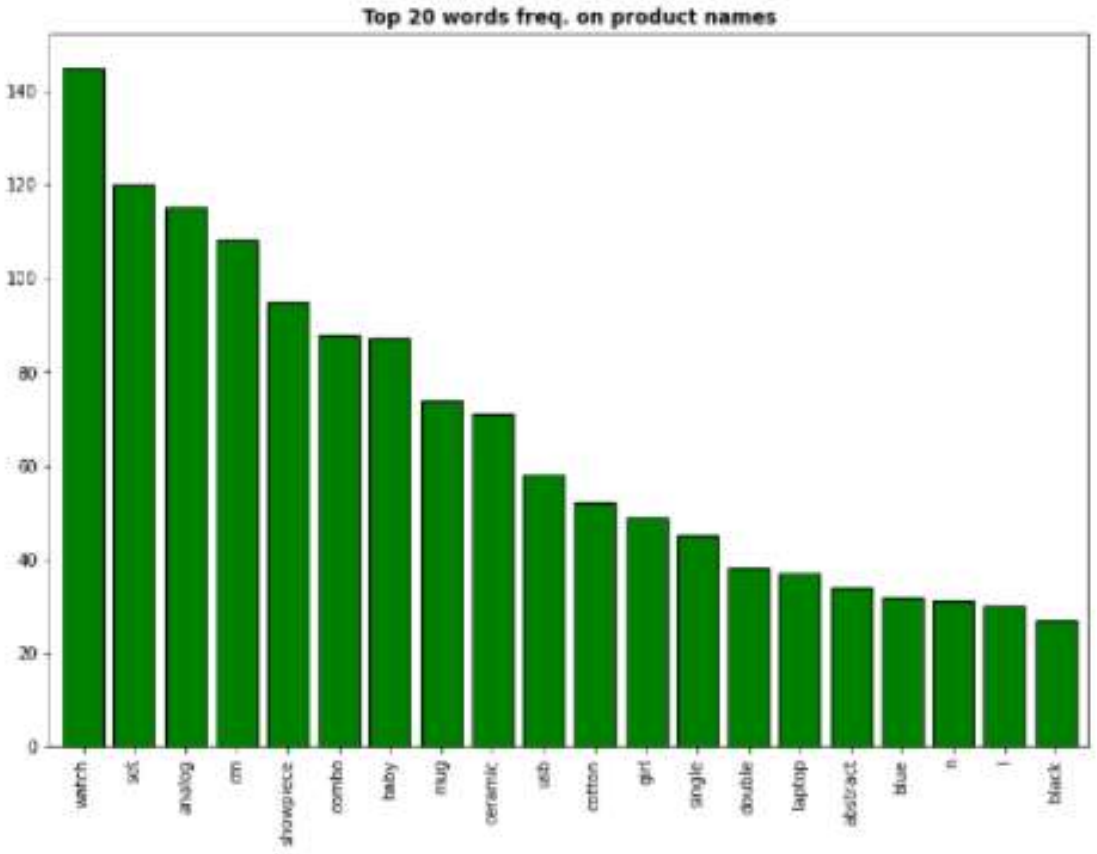
TEXTE



ANALYSE DU TEXTE

_PRÉ-TRAITEMENT

TEXTE



ANALYSE DU TEXTE

_PRÉ-TRAITEMENT

TEXTE

ANALYSE DU TEXTE
_BAG OF WORDS

BAG OF WORDS

- **Feature Extractor: CountVectorizer**

```
<1050x3475 sparse matrix of type '<class 'numpy.int64'>'
        with 25485 stored elements in Compressed Sparse Row format>
```

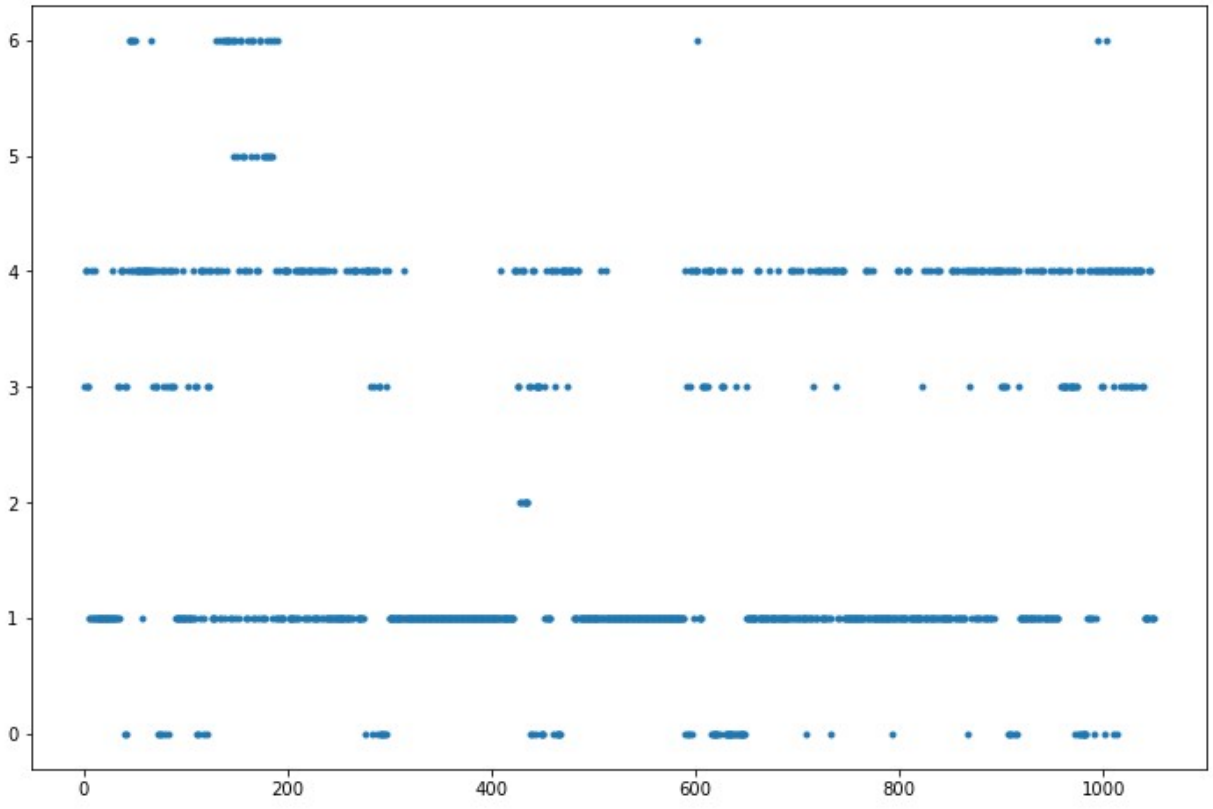
- **Reduction de dimension: TruncatedSVD (LSA)**

```
Dimensions dataset avant réduction LSA :  (1050, 3475)
Dimensions dataset après réduction LSA :  (1050, 100)
```

ANALYSE DU TEXTE
_BAG OF WORDS

BAG OF WORDS

- **Clustering:**



ARI : 0.027677005335788898

ANALYSE DU TEXTE

_BAG OF WORDS

BAG OF WORDS

- **Clustering: Top termes par clusters**

Cluster 0:	Cluster 1:	Cluster 2:	Cluster 3:
cm	adapter	rs	price
design	vgn	free	laptop
color	replacement	cash	quality
product	cr	genuine	box
material	vaio	delivery	color
pack	warranty	day	warranty
width	smartpro	flipkart	pack
inch	e	replacement	skin
number	power	guarantee	type
model	product	com	general

Cluster 4:	Cluster 5:	Cluster 6:
baby	mug	vacuum
girl	coffee	computer
fabric	ceramic	attachment
cotton	perfect	keyboard
dress	tea	power
ideal	love	usb
general	ml	brush
neck	quality	port
pack	material	dust
shirt	printland	laptop

0: Baby Care
1: Beauty & Personal
2: Computers
3: Home Decor
4: Home Furnishing
5: Watches
6: Kitchen Dining

ANALYSE DU TEXTE
_BAG OF WORDS

BAG OF WORDS

- **Clustering:**

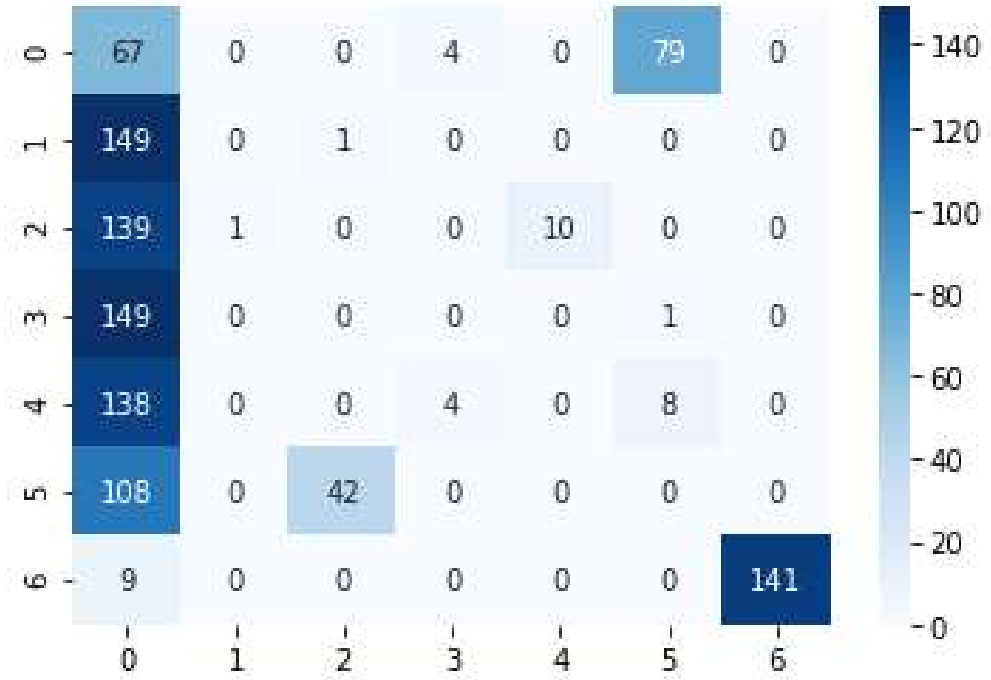
	precision	recall	f1-score	support
0	0.09	0.45	0.15	150
1	0.00	0.00	0.00	150
2	0.00	0.00	0.00	150
3	0.00	0.00	0.00	150
4	0.00	0.00	0.00	150
5	0.00	0.00	0.00	150
6	1.00	0.94	0.97	150
accuracy			0.20	1050
macro avg	0.16	0.20	0.16	1050
weighted avg	0.16	0.20	0.16	1050

ARI : 0.027677005335788898

ANALYSE DU TEXTE
_BAG OF WORDS

BAG OF WORDS

- **Clustering:**



ARI : 0.13624830849931518

ANALYSE DU TEXTE

_TFIDF

PREPARATION

- Données: Train(0.75) / Test(0.25)
- Pré-traitement: Idem
- Transformer: LabelEncoder pour les categories

ANALYSE DU TEXTE

_TFIDF

MODEL

- **Feature extractor: TFIDF**

```
pipe_svd = make_pipeline(CountVectorizer(), TruncatedSVD(n_components=300))
pipe_svd.fit(train_NLP['desc_clean'])
feat_train_svd = pipe_svd.transform(train_NLP['desc_clean'])
feat_train_svd.shape
```

```
(787, 300)
```

- **Model: RandomForest – LogisticRegression – MultinomialNB**
- **Reduction dimension: TruncatedSVD (LSA)**
- **GridSearch**

ANALYSE DU TEXTE

_TFIDF

PERFORMANCES

- **Accuracy**

-> RandomForest 0.9543726235741445

-> RandomForest (LSA) 0.9163498098859315

-> LogisticRegression 0.9467680608365019

-> LogisticRegression (LSA) 0.9505703422053232

-> MultinomialNB 0.8935361216730038

ANALYSE DU TEXTE
_WORD2VECT

PREPARATION

- Données: Train(0.75) / Test(0.25)
- Pré-traitement: Idem
- Transformer: Idem

ANALYSE DU TEXTE
_WORD2VECT

MODEL

- **Feature extractor: Word2Vect**

```
model_W2V = Word2Vec(sentance, vector_size=300, window=20,
                     min_count=2, workers=1)
model_W2V.corpus_count
```

```
263
```

- **Model: RandomForest – LogisticRegression**

ANALYSE DU TEXTE
_WORD2VECT

PERFORMANCES

- **Accuracy**

-> RandomForest `0.49049429657794674`

-> LogisticRegression `0.4296577946768061`

CONCLUSION

_VOTING CLASSIFIER

- **Features Extractor:**
  - Image: VGG16
  - NLP: TFIDF
- **Concatenation:**
- **Pipeline: ColumnExtractor**
  - Image: Appliquer modèle RandomForest (Acc 0.77)
  - NLP: Appliquer modèle LinearRegression (Acc 0.94)
- **VotingClassifier:**
  - Soft: Acc 0.93
  - Hard: Acc 0.81

CONCLUSION

_FAISABILITÉ

MOTEUR DE CLASSIFICATION

- **Bonnes performances sur images et textes:**
  - Accuracy > 0.80 et 0.90 respectivement
- **Modèle pré-entrainer et fréquentiels sont bien meilleurs pour les images et textes:**
  - VGG16
  - TFIDF
- **Modèle de clustering mauvais:**
  - TFIDF
- **Axes d'ameliorations:**
  - Affiner performances sur certaines categories de produits
  - Tester la faisabilité sur niveaux 2 et 3 des catégories