

# Classifiez automatiquement des biens de consommation

place de marché

## DONNÉES TEXTUELLES



## IMAGES





1 Problématique/Données



2 Traitement données textuelles



3 Traitement données images



4 Combinaison textes/images



5 Conclusions



## Problématique/Données



## Traitement données textuelles



## Traitement données images



## Combinaison textes/images



## Conclusions



# Problématique



place de marché

## Mission :

Réaliser une **étude de faisabilité** d'un **moteur de classification** d'articles basé sur une image et une description pour l'**automatisation** de l'attribution de la **catégorie** de l'article pour l'entreprise *Place de Marché*.

## Objectifs :

**Améliorer** l'expérience des utilisateurs.

**Fiabiliser** la catégorie des articles avec pertinence et précision.

# Processus



## PRÉ TRAITEMENT



Données TEXTES



Données IMAGES



Données TEXTES  
+  
Données IMAGES

## FEATURES EXTRACTION

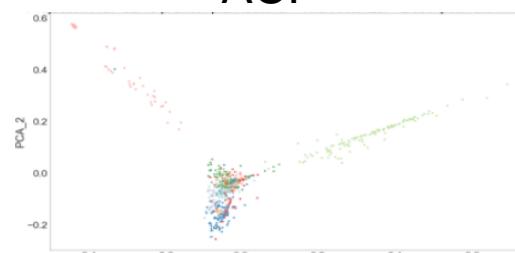
Bags Of Words  
Words Embeddings  
Topics Embeddings

Bags Of Features

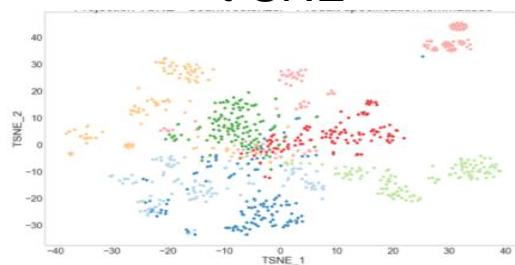
Words Embeddings  
Bags Of Features

## RÉDUCTION DIMENSION

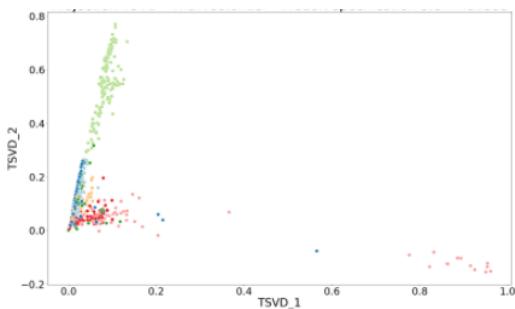
ACP



t-SNE

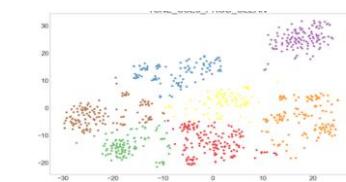
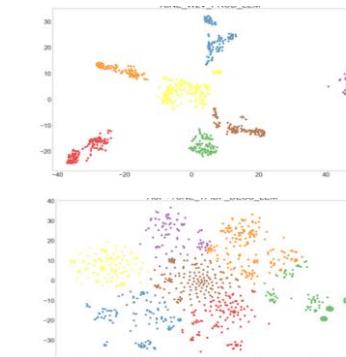


TruncatedSVD



## CLASSIFICATION

Apprentissage  
Non supervisé  
KMeans



Apprentissage  
Supervisé

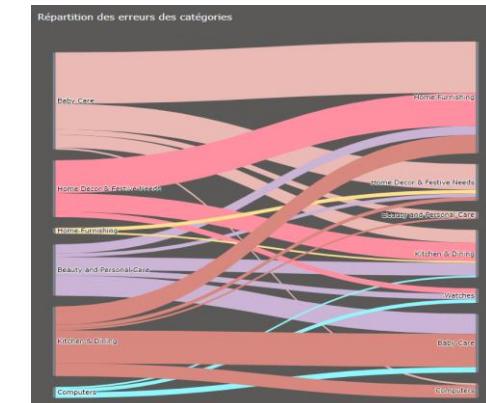
## ÉVALUATION



TSNE\_USE5\_PROD\_CLEAN

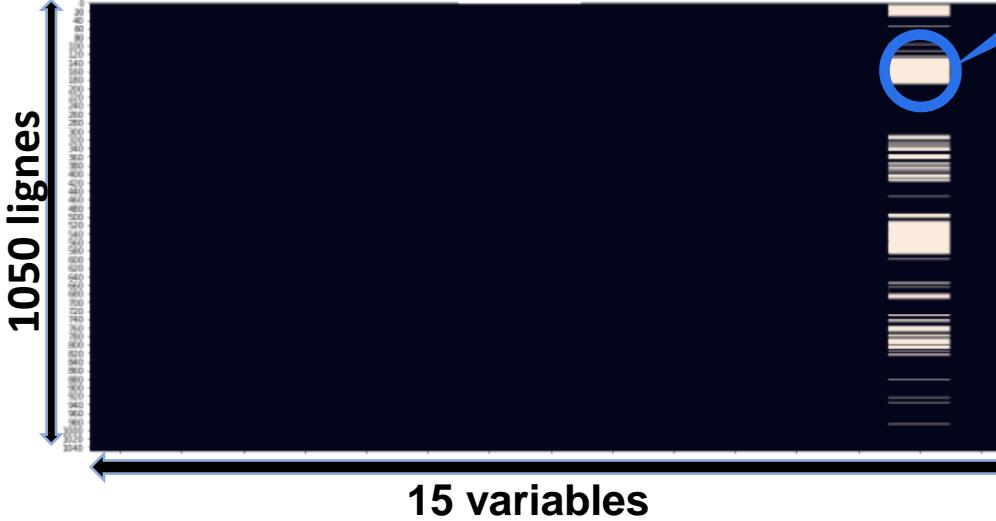
	0	1	2	3	4	5	6	7	8	9
Baby Care	93	11	3	3	1	2	0	0	0	0
Beauty and Personal Care	4	128	0	0	0	0	9	0	0	0
Computers	2	3	145	0	0	0	23	0	0	0
Home Decor & Festive Needs	14	3	0	118	1	18	0	0	0	0
Home Furnishing	32	2	0	1	144	10	0	0	0	0
Kitchen & Dining	5	0	0	25	4	87	0	0	0	0
Watches	0	3	2	3	0	1	150	0	0	0
	0	1	2	3	4	5	6	7	8	9

'Précision: 82.38%'





# Données



Jeu de données  
(e-commerce Indien Flipkart)

Moins de 2% de valeurs manquantes

## Textes

**product\_name**  
Sathiya Cotton Bath Towel

**description**

Specifications of Sathiya Cotton Bath Towel (3 Bath Towel, Red, Yellow, Blue)  
Bath Towel Features Machine Washable Yes Material ...

**product\_specifications**

```
{"product_specification":>[{"key":>"Brand", "value":>"Elegance"},  
 {"key":>"Designed For", "value":>"Door"},...]
```

**brand**

Sathiya

## Image

**image**



## Cible

**product\_category\_tree**

"Baby Care >> Baby Bath & Skin >> Baby Bath Towels >> Sathiya Baby Bath Towels >> Sathiya Cotton Bath Towel (3 Bath Towel, Red, Y..."



# Données – Niveaux des catégories

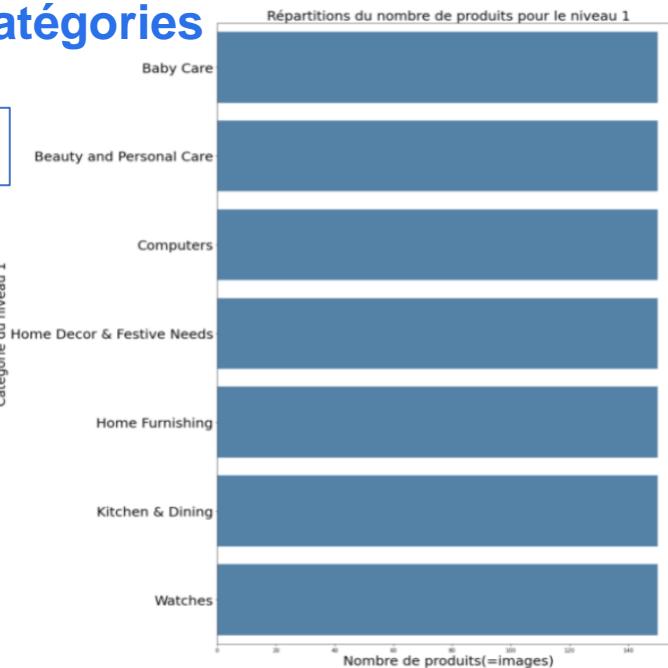


7 catégories



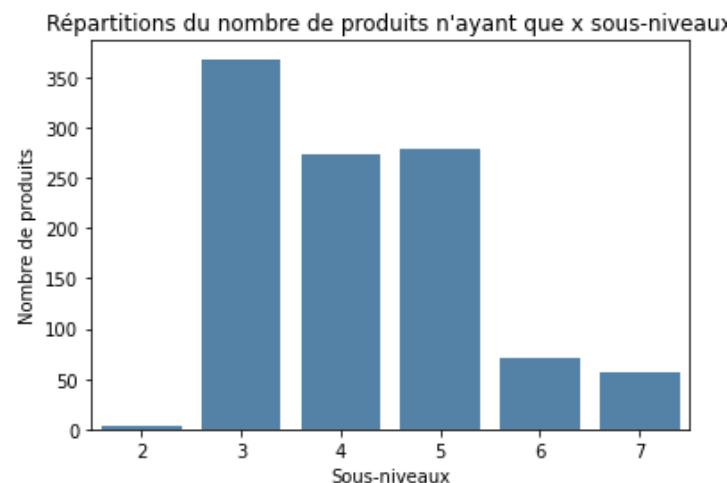
Niveau 1

150 articles  
ou images  
par catégorie



Sous-niveaux

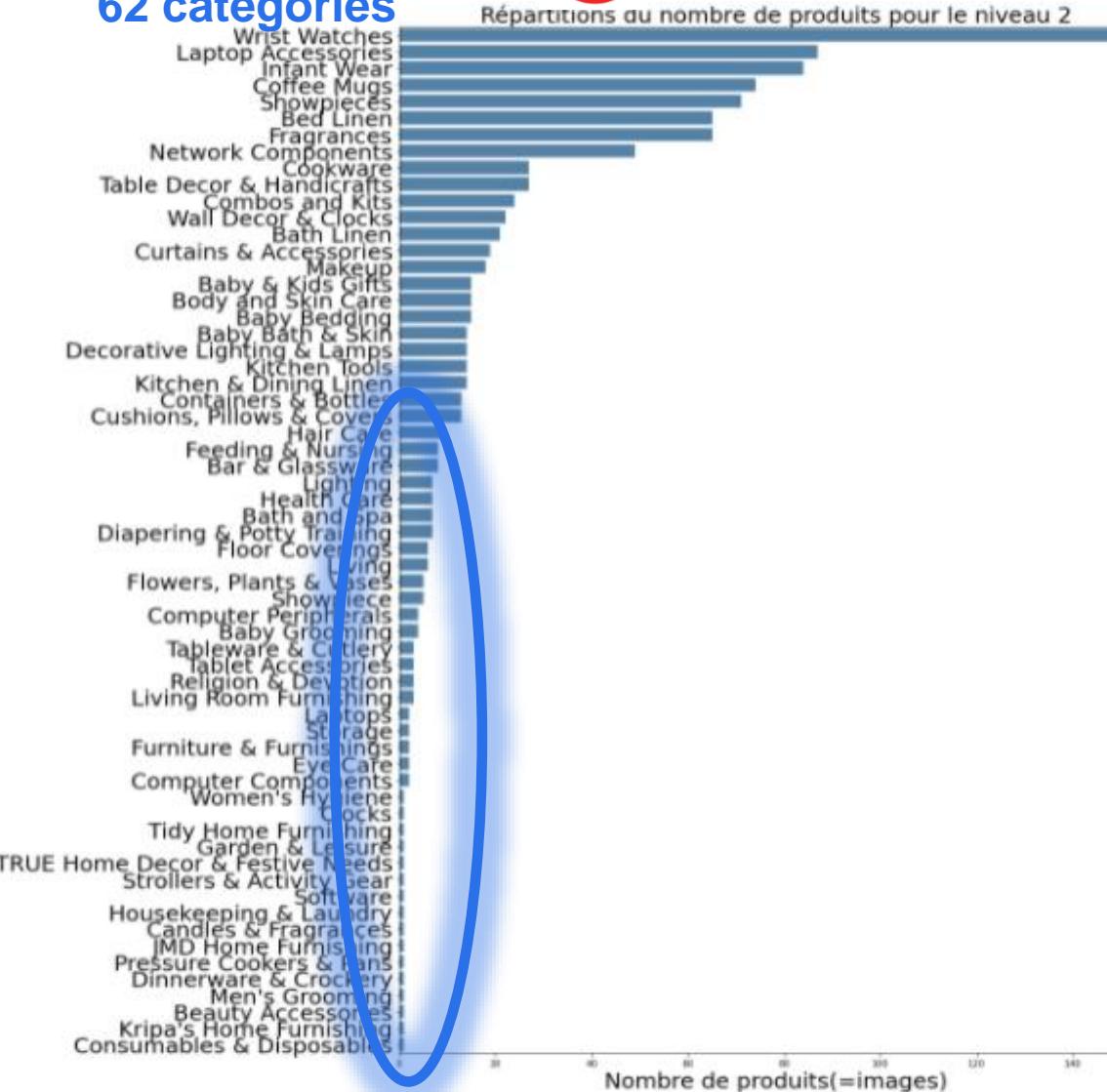
Niveau	Nb catégories
2	62
3	241
4	349
5	297
6	117
7	57



62 catégories



Niveau 2





Problématique/Données



Traitement données textuelles



Traitement données images



Combinaison textes/images



Conclusions

## PRÉ TRAITEMENT

### Données TEXTES

description  
product\_name  
product\_specifications  
brand  
Combinaisons :  
Produits  
Toutes les variables

Tokenisation

Normalisation

Racinement

Lemmatisation

## FEATURES EXTRACTION

Bags Of Words  
CountVectorizer  
TfidfVectorizer  
HashingVectorizer

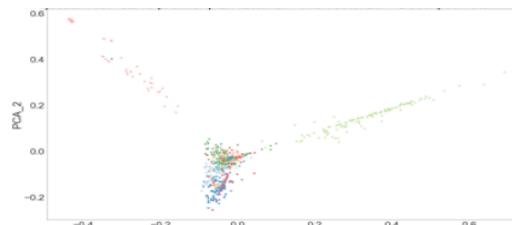
Words Embeddings  
Word2Vec  
Doc2Vec

Sentence-  
Transformers  
roberta-large

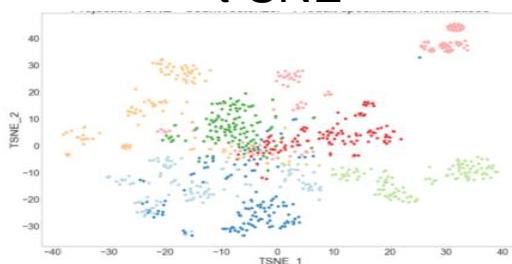
distilbert-base-uncased  
distilbert-base-uncased-  
finetuned-sst-2-english  
xlm-roberta-large-  
finetuned-conll03-english  
google/electra-large-  
discriminator...

## RÉDUCTION DIMENSION

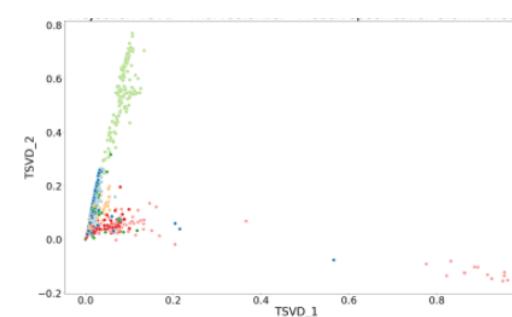
### ACP



### ACP suivi de t-SNE t-SNE

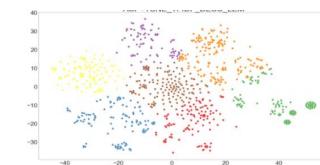


### TruncatedSVD



## CLASSIFICATION

Topic modelling  
LDA  
NMF



Apprentissage  
Non supervisé  
KMeans

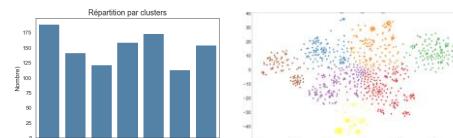
Apprentissage  
Supervisé  
Pycaret

## ÉVALUATION CATÉGORIE

Évaluation  
ARI  
Accuracy

	TIME_USED_PROD_CLEAN	Baby Care	Beauty and Personal Care	Computers	Home Decor & Festive Needs	Home Furnishing	Kitchen & Dining	Watches	
Baby Care	11	11	3	3	1	2	0	0	0
Beauty and Personal Care	4	103	0	0	0	0	0	0	0
Computers	2	3	165	0	0	0	29	0	0
Home Decor & Festive Needs	14	3	0	158	1	18	0	0	0
Home Furnishing	32	2	0	1	144	10	0	0	0
Kitchen & Dining	5	0	0	25	4	17	0	0	0
Watches	0	3	2	3	0	1	0	100	0
	0	1	2	4	5	5	0	0	0
Précision: 82.38%									

Interprétation clusters



Erreurs





# NLP – Pré-traitement



TOKENISATION

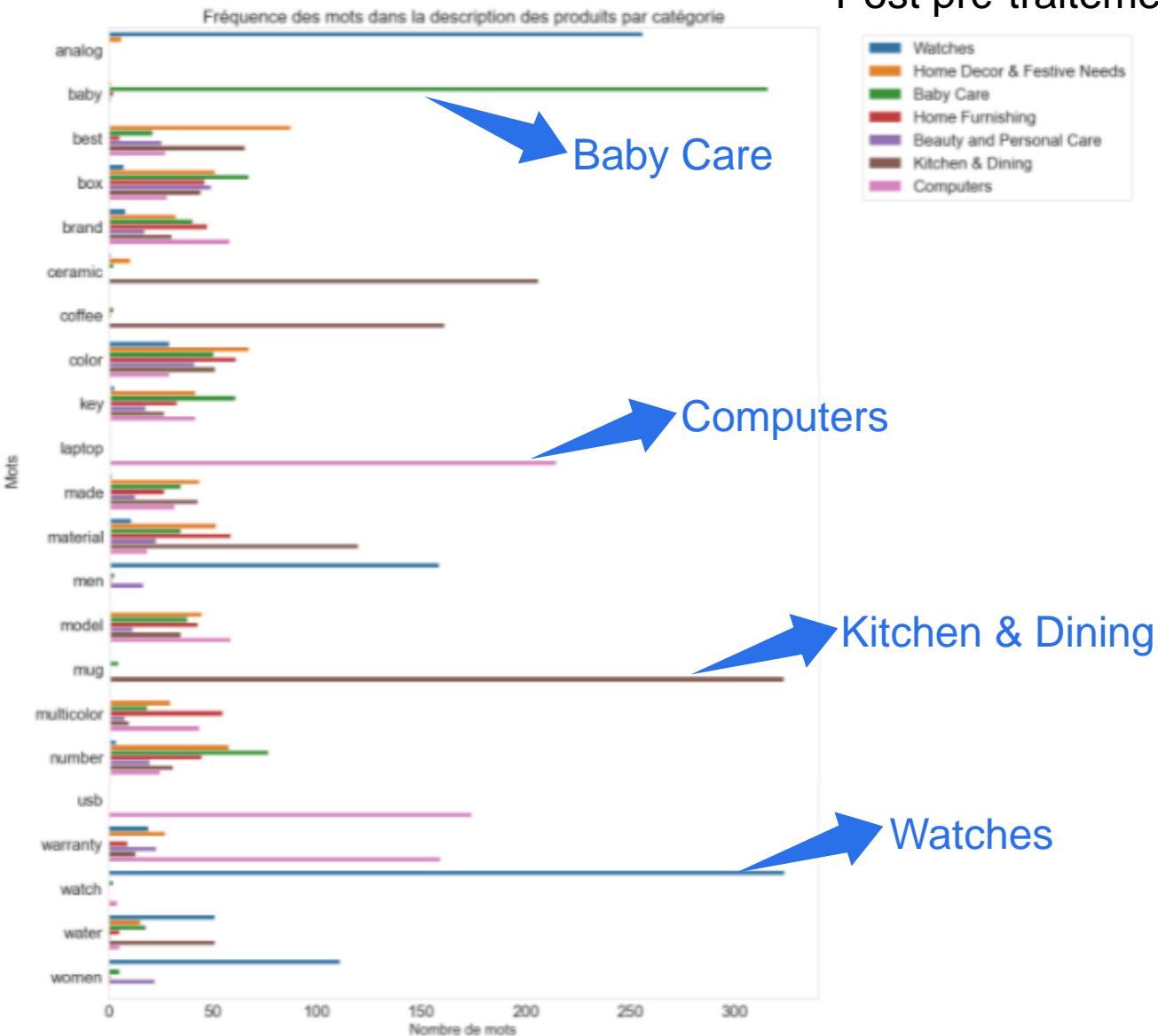
NORMALISATION

3

4

Texte original	Wallmantra Large Vinyl Sticker Sticker (Pack of 1) Price: Rs. 1,896 Bring home this exclusive Piece of Wall Art to give your home a refreshing look it deserves ! Wall Decals are the latest trend, sweeping the world of interior design, as a quick and easy way to personalise and transform your home.
Tokenisation NLTK 1	['Wallmantra', 'Large', 'Vinyl', 'Sticker', 'Sticker', '(', 'Pack', 'of', '1', ')', 'Price', ':', 'Rs', '.', '1,896', 'Bring', 'home', 'this', 'exclusive', 'Piece', 'of', 'Wall', 'Art', 'to', 'give', 'your', 'home', 'a', 'refreshing', 'look', 'it', 'deserves', '!', 'Wall', 'Decals', 'are', 'the', 'latest', 'trend', ',', 'sweeping', 'the', 'world', 'of', 'interior', 'design', ',', 'as', 'a', 'quick', 'and', 'easy', 'way', 'to', 'personalise', 'and', 'transform', 'your', 'home', '.']
Tokens en 1 phrase	Wallmantra Large Vinyl Sticker Sticker ( Pack of 1 ) Price : Rs . 1,896 Bring home this exclusive Piece of Wall Art to give your home a refreshing look it deserves ! Wall Decals are the latest trend , sweeping the world of interior design , as a quick and easy way to personalise and transform your home .
TextHero minuscules 2	wallmantra large vinyl sticker sticker ( pack of 1 ) price : rs . 1,896 bring home this exclusive piece of wall art to give your home a refreshing look it deserves ! wall decals are the latest trend , sweeping the world of interior design , as a quick and easy way to personalise and transform your home .
TextHero suppr chiffres	wallmantra large vinyl sticker sticker ( pack of ) price : rs . bring home this exclusive piece of wall art to give your home a refreshing look it deserves ! wall decals are the latest trend sweeping the world of interior design as a quick and easy way to personalise and transform your home .
TextHero suppr ponctuation	wallmantra large vinyl sticker sticker pack of price rs bring home this exclusive piece of wall art to give your home a refreshing look it deserves wall decals are the latest trend sweeping the world of interior design as a quick and easy way to personalise and transform your home
TextHero suppr espaces blancs	wallmantra large vinyl sticker sticker pack of price rs bring home this exclusive piece of wall art to give your home a refreshing look it deserves wall decals are the latest trend sweeping the world of interior design as a quick and easy way to personalise and transform your home
TextHero + NLTK stop_words	wallmantra large vinyl sticker sticker pack price rs bring home exclusive piece wall art give home refreshing look deserves wall decals latest trend sweeping world interior design quick easy way personalise transform home
Suppr mots NLTK fréquents-rares	wallmantra large vinyl sticker sticker pack bring home exclusive piece wall art give home refreshing look wall decals latest trend world interior design quick easy way transform home
Racinisation (radical)	wallmantra larg vinyl sticker sticker pack bring home exclus piec wall art give home refresh look wall decal latest trend world interior design quick easi way transform home
Lemmatisation (forme canonique)	wallmantra large vinyl sticker sticker pack bring home exclusive piece wall art give home refreshing look wall decal latest trend world interior design quick easy way transform home

# NLP – Fréquence mots : description



## Post pré-traitement

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

material  
pack  
**cover**  
polyester  
set  
comfort  
inch  
design  
cotton  
package  
sales  
floral  
multicolor

## Home Decor & Festive Needs

color  
beautiful  
wooden  
prices  
paper  
ganesh  
made  
showpiece  
material  
height  
gift  
width  
type  
box  
home  
inch  
wall  
best  
room  
pack  
sales  
general  
dimensions  
brass  
number  
dimensions  
art  
model

## Kitchen & Dining

product  
steel  
quality  
dishwasher  
pack  
design  
coffee  
perfect  
mugs  
ceramic  
mug  
safe  
kitchen  
best  
one  
safer  
microwave  
water  
rockmantrapizza  
tea  
mug  
coffee  
ceramic  
mug  
material  
fresh  
type

## Baby Care

casual  
girl  
box  
printed  
color  
type  
number  
details  
fabric  
contents  
set  
dress  
ideal  
key  
details  
**baby**  
cotton  
blue  
boy  
wash  
shirt  
length  
neck  
pack  
sleeve  
pattern  
general

## Beauty and Personal Care

product  
cream  
beauty  
jewellery  
prices  
warranty  
body  
type  
**Combo**  
set  
skin  
box  
oil  
hair  
pack  
lip  
ideal  
best  
color  
traits

## Computers

product  
key  
flexible  
charge  
general  
print  
skin  
**laptop**  
adapter  
battery  
pad  
please  
light  
brand  
warranty  
usb  
power  
mouse  
size  
model  
vaio  
led  
quality  
shapes

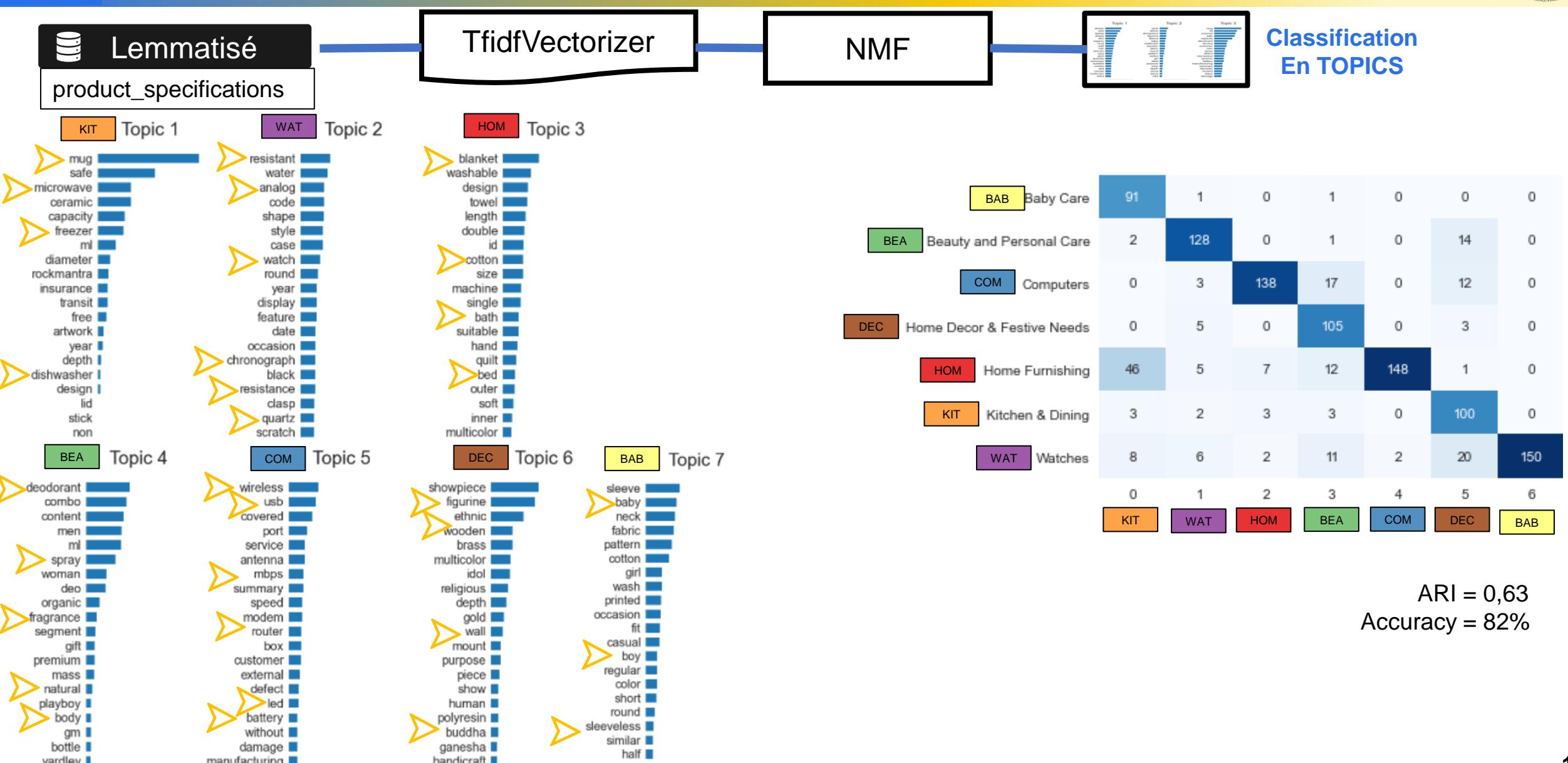


## Traitement données textuelles

# **NLP – TOPICS MODELLING**



# NLP – Classification NMF





## Traitement données textuelles

# NLP – MÉTHODES CLASSIQUES



# NLP – Pipeline Méthodes classiques



## PRÉ TRAITEMENT

Données TEXTES

Lemmatisé  
product\_specifications

Stemmatisé  
product\_specifications

## FEATURES EXTRACTION

Bag Of Words

TfidfVectorizer

CountVectorizer

HashingVectorizer

## RÉDUCTION DIMENSION

ACP +  
T-SNE

T-SNE

## CLASSIFICATION

Non Supervisé  
KMeans

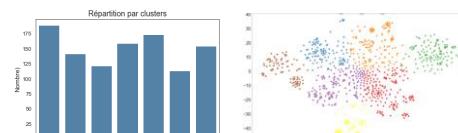
Supervisé  
Pycaret  
ExtraTreesClassifier  
CatboostClassifier  
KNeighborsClassifier

## ÉVALUATION CATÉGORIE

Évaluation  
ARI  
Accuracy

	TSNE_UPDS_PROD_CLEAN						
	0	1	2	3	4	5	6
Baby Care	35	10	2	1	2	0	0
Beauty and Personal Care	4	10	0	0	0	0	0
Computers	2	2	0	0	0	0	0
Home Decor & Festive Needs	14	3	0	0	1	0	0
Home Furnishing	32	2	0	1	144	10	0
Kitchen & Dining	5	0	0	26	4	0	0
Vases	0	3	2	3	0	1	100
	0	1	2	3	4	5	6

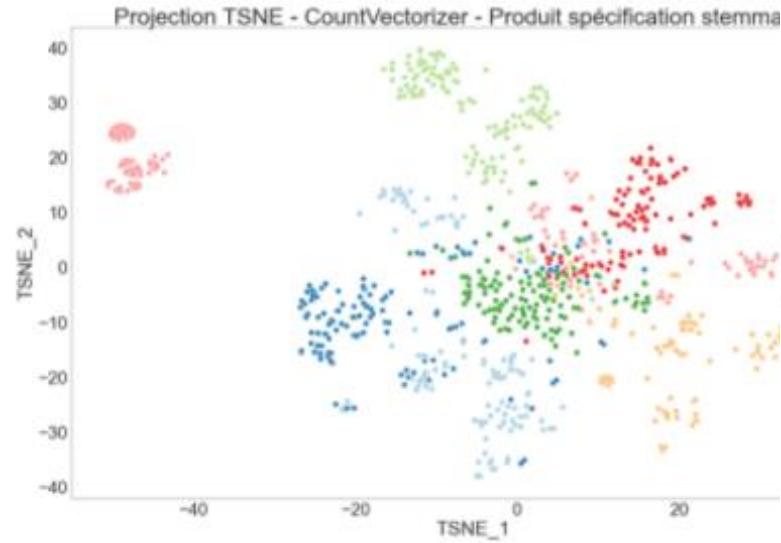
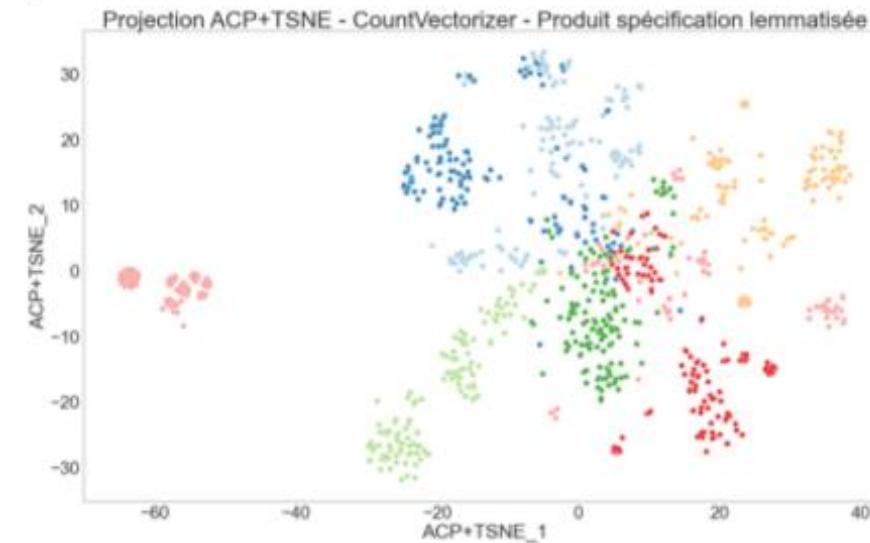
Interprétation clusters



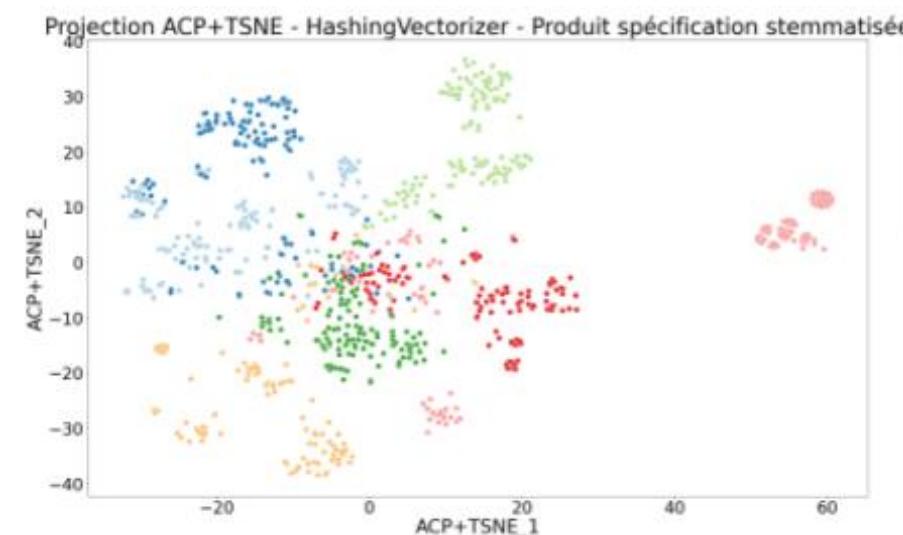
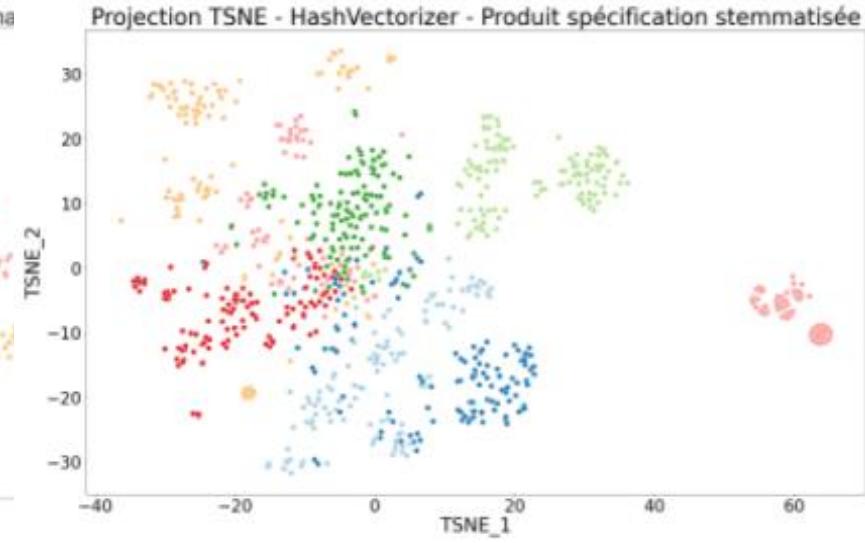
# NLP – Réduction de dimension



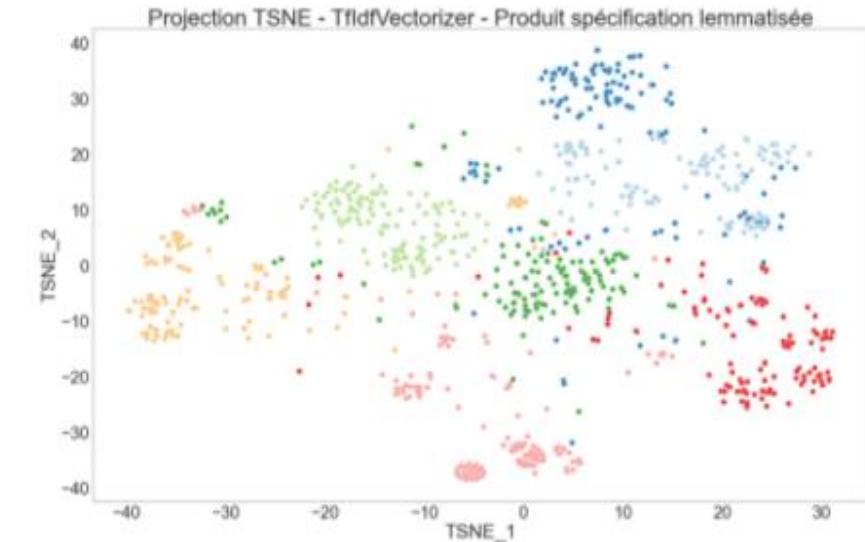
ACP + T-SNE



T-SNE



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

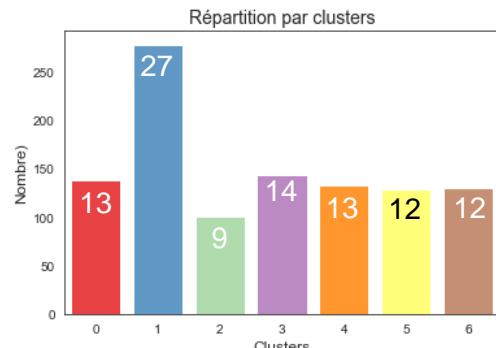




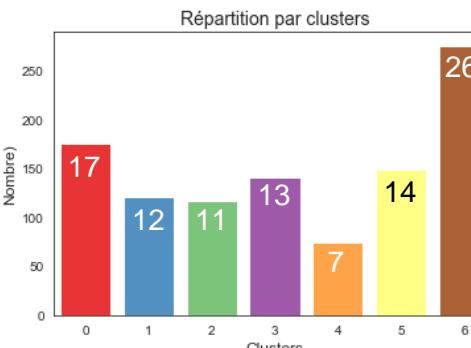
# NLP – Classification Non Supervisée KMeans



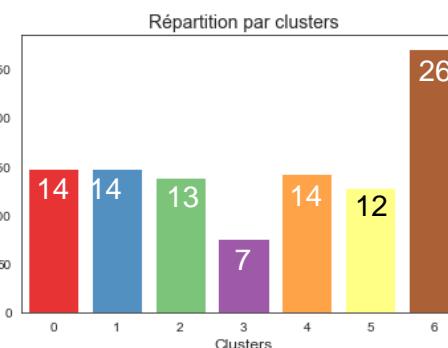
ACP + T-SNE CV PROD LEM



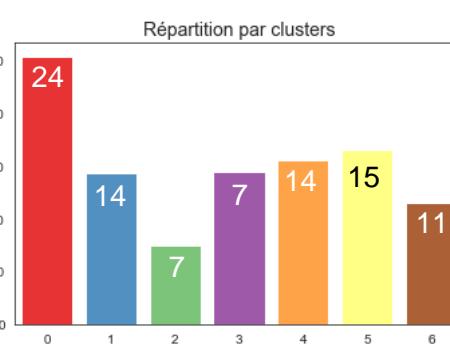
ACP + T-SNE HASH PROD STEM



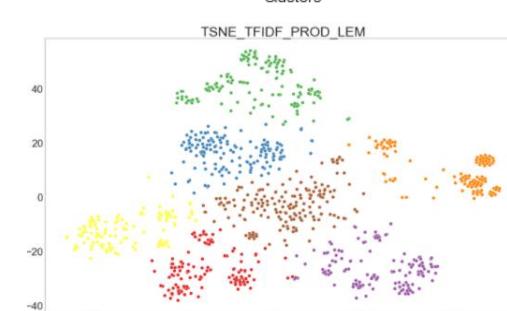
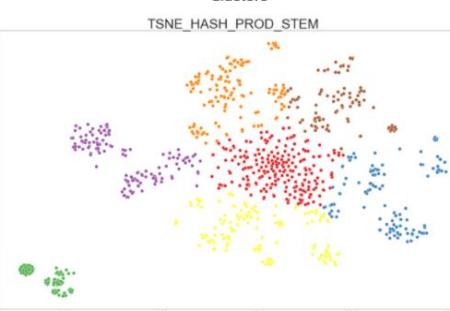
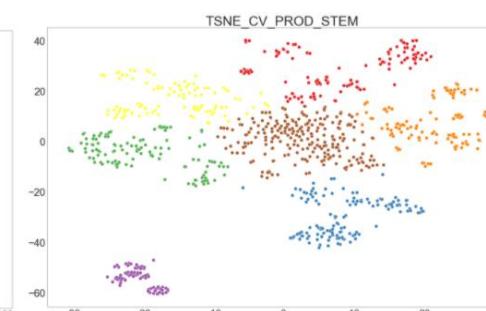
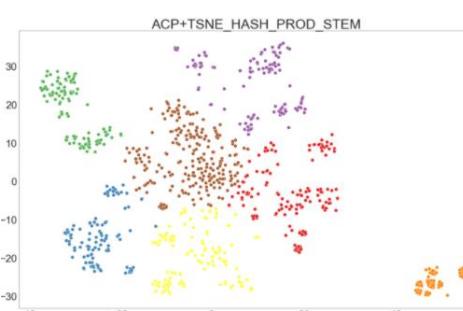
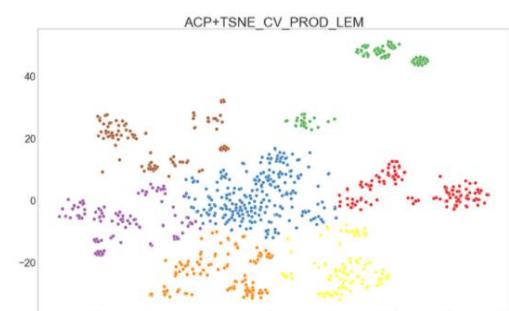
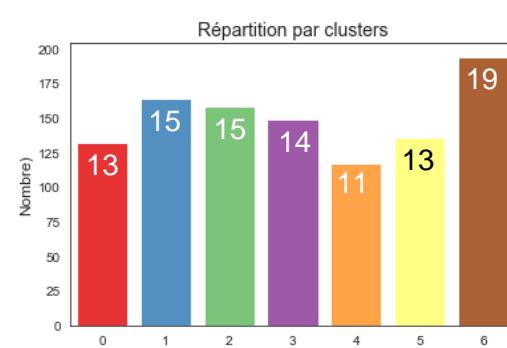
T-SNE CV PROD STEM



T-SNE HASH PROD STEM



T-SNE TFIDF PROD LEM



ACP+TSNE_CV_PROD_LEM						
BAB	93	0	0	0	35	0
BEA	2	106	2	13	0	20
COM	0	0	130	0	0	0
DEC	23	40	18	137	18	30
HOM	32	4	0	0	97	0
KIT	0	0	0	0	100	0
WAT	0	0	0	0	0	138

ACP+TSNE_HASH_PROD_STEM						
BAB	89	0	0	0	32	0
BEA	4	107	5	4	0	55
COM	0	0	117	17	0	7
DEC	18	38	28	124	19	14
HOM	39	5	0	5	99	0
KIT	0	0	0	0	0	74
WAT	0	0	0	0	0	116

TSNE_CV_PROD_STEM						
BAB	92	0	0	0	46	0
BEA	1	110	0	1	0	31
COM	0	0	128	12	0	7
DEC	33	40	16	133	2	38
HOM	23	0	6	0	99	0
KIT	1	0	0	0	0	74
WAT	0	0	0	4	3	0

TSNE_HASH_PROD_STEM						
BAB	107	0	0	5	43	0
BEA	2	107	2	19	0	35
COM	0	0	120	16	0	7
DEC	31	43	17	100	19	34
HOM	10	0	11	6	88	0
KIT	0	0	0	0	0	74
WAT	0	0	0	4	0	140

TSNE_TFIDF_PROD_LEM						
BAB	80	1	0	1	42	0
BEA	6	127	0	3	0	13
COM	0	3	132	14	0	9
DEC	21	14	14	119	10	16
HOM	30	4	0	0	98	0
KIT	4	0	1	2	0	110
WAT	0	1	3	11	0	2

ARI = 0,53  
Accuracy = 76%

ARI = 0,43  
Accuracy = 69%

ARI = 0,56  
Accuracy = 76%

ARI = 0,45  
Accuracy = 73%

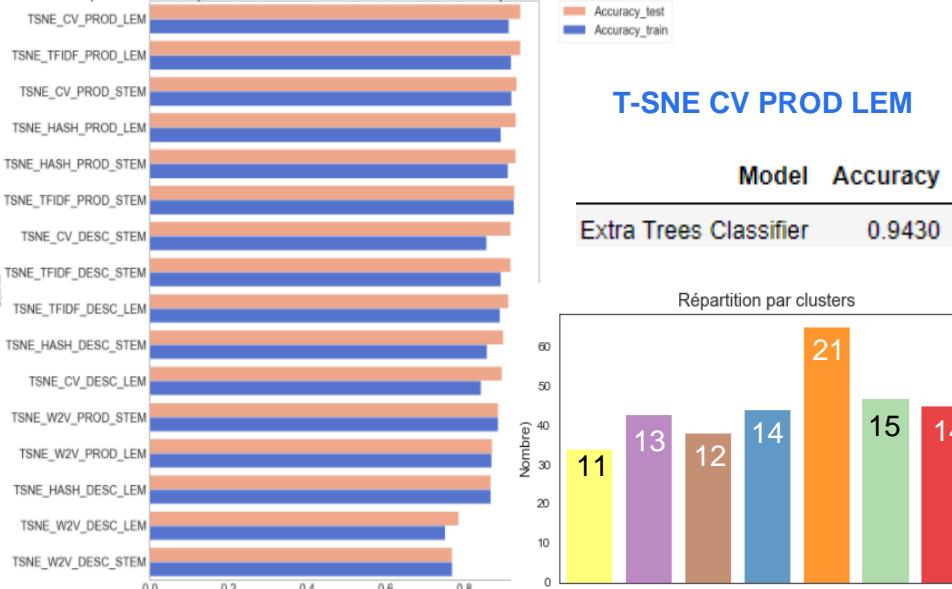
ARI = 0,6  
Accuracy = 78%  



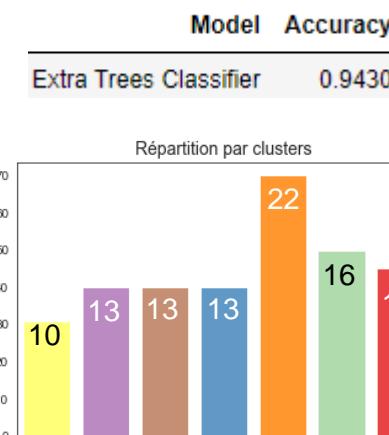
# NLP – Classification Supervisée



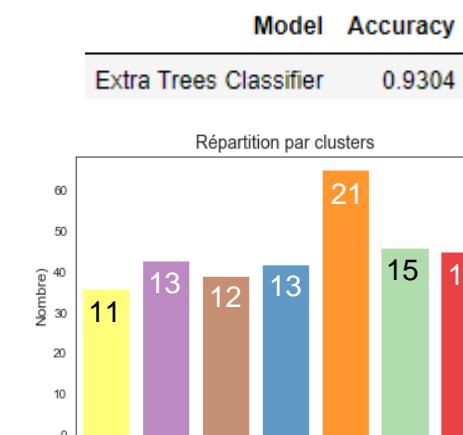
Comparaison de la précision de la classification - NLP Classiques du TEST SET



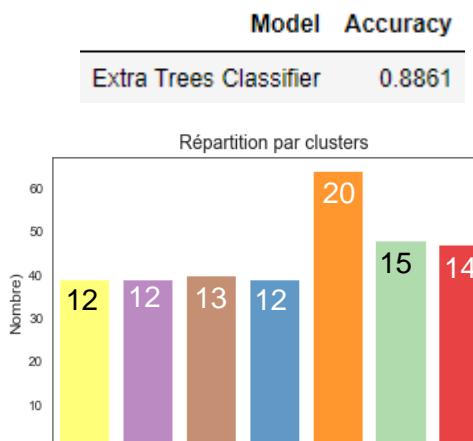
**T-SNE TFIDF PROD LEM**



**T-SNE HASH PROD LEM**



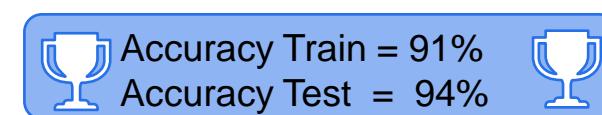
**T-SNE W2V PROD STEM**



BAB	Baby Care
BEA	Beauty and Personal Care
COM	Computers
DEC	Home Decor & Festive Needs
HOM	Home Furnishing
KIT	Kitchen & Dining
WAT	Watches

TSNE_CV_PROD_LEM							
BAB	0	0	0	2	3	1	
BEA	0	41	0	1	0	1	0
COM	2	1	38	1	0	1	0
DEC	0	1	0	38	0	0	0
HOM	2	0	0	0	62	0	0
KIT	0	0	0	2	0	44	0
WAT	0	0	0	0	0	0	45

Accuracy Train = 91%  
Accuracy Test = 94%



Accuracy Train = 91%  
Accuracy Test = 94%

Accuracy Train = 91%  
Accuracy Test = 94%

Accuracy Train = 91%  
Accuracy Test = 94%

Accuracy Train = 89%  
Accuracy Test = 93%

Accuracy Train = 88%  
Accuracy Test = 88%

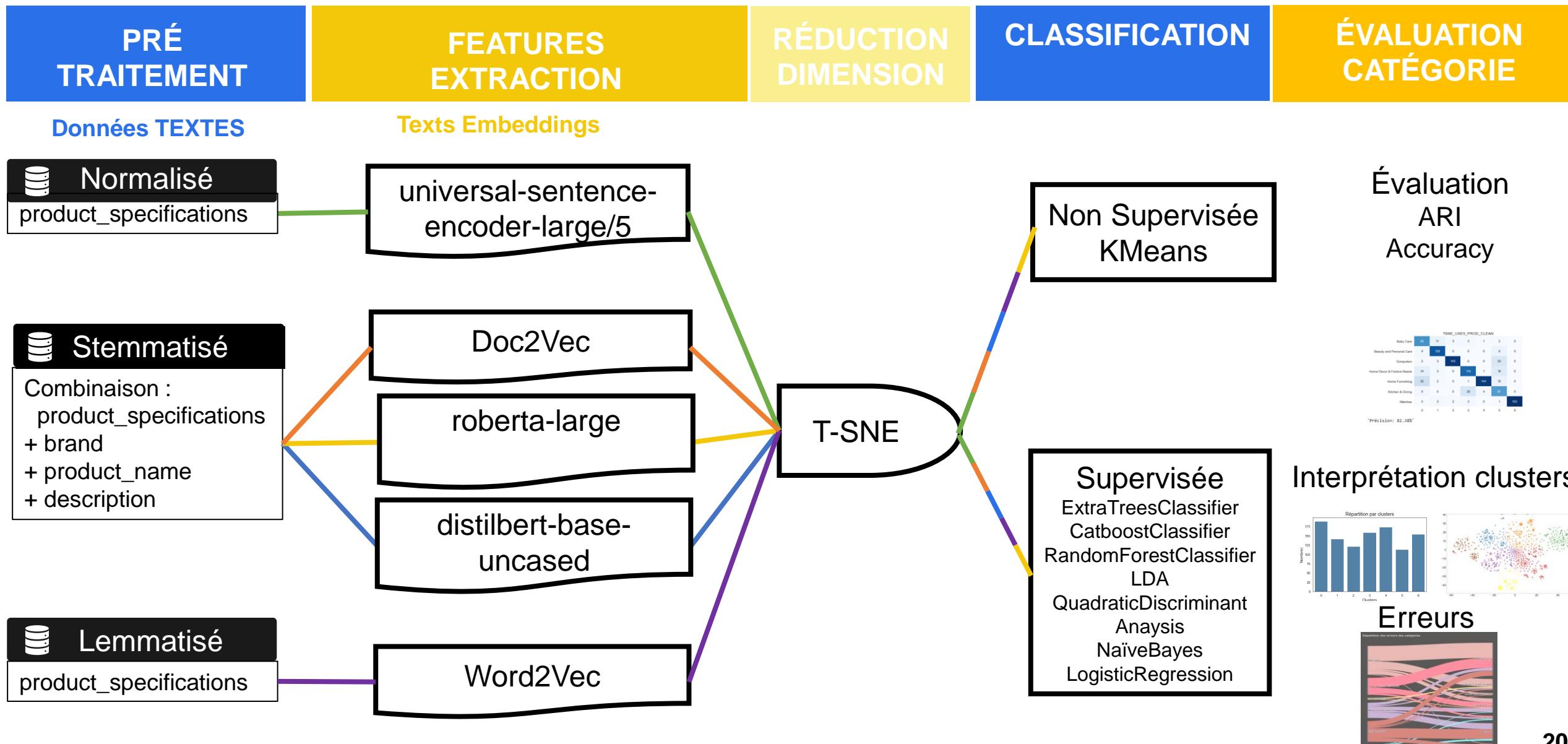


## Traitement données textuelles

# NLP – TEXTS EMBEDDINGS



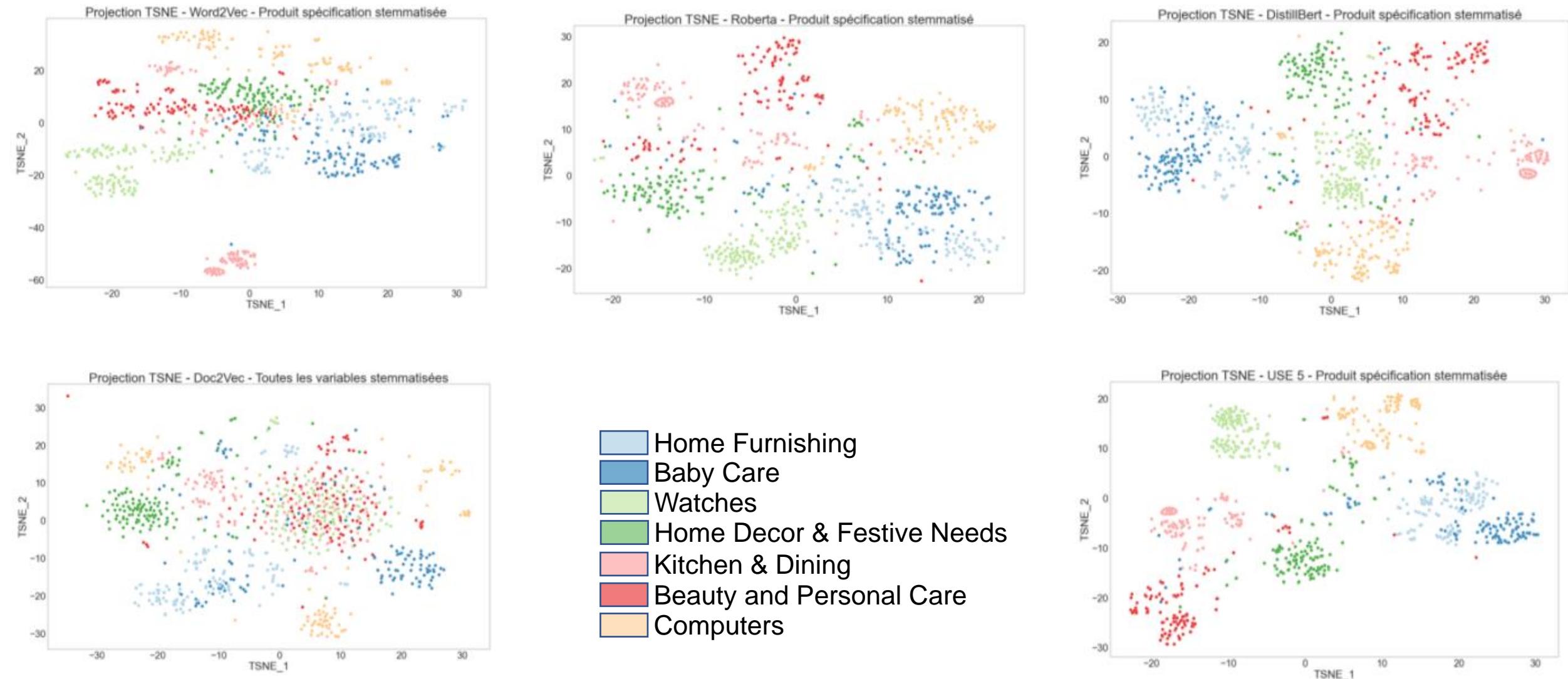
# NLP – Pipeline Texts Embeddings





# NLP – Réduction de dimension

place de marché

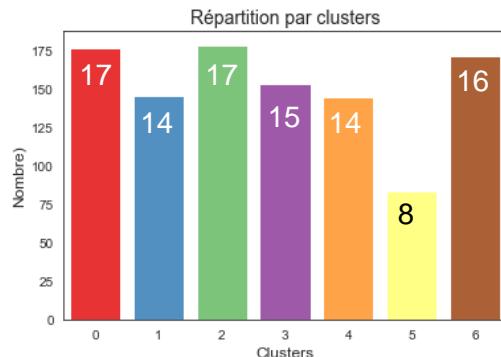




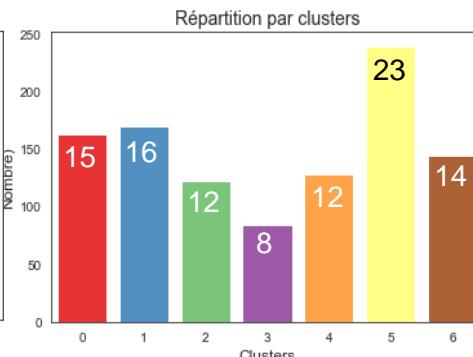
# NLP – Classification Non Supervisée KMeans



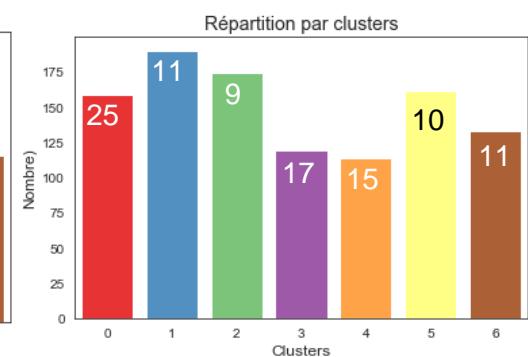
T-SNE DOC2VEC ALL STEM



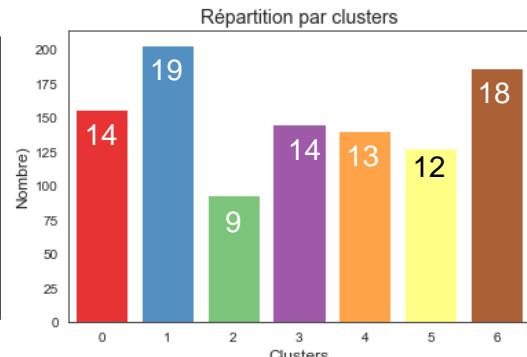
T-SNE WORD2VEC PROD LEM



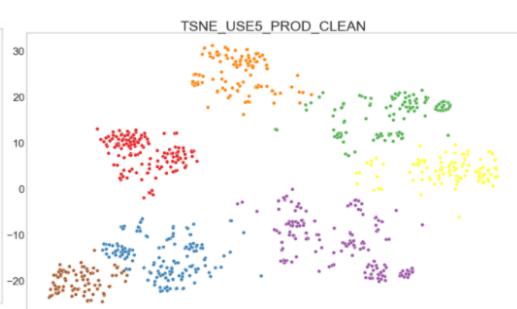
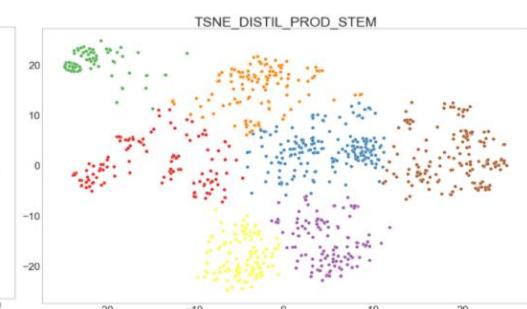
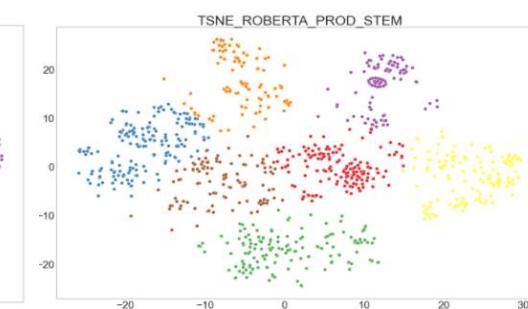
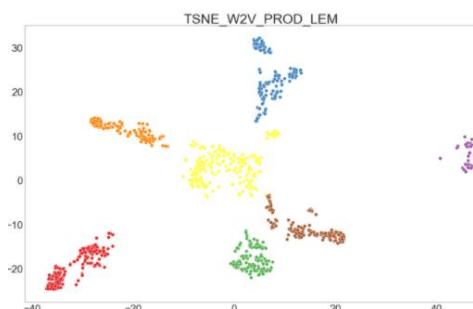
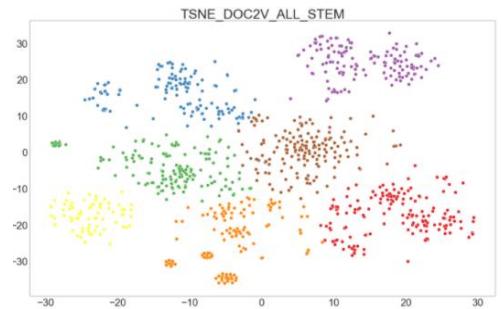
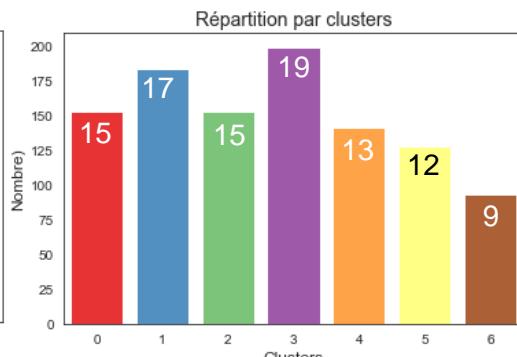
T-SNE ROBERTA PROD STEM



T-SNE DISTILLBERT PROD STEM



T-SNE USE PROD CLEAN



TSNE_DOC2V_ALL_STEM						
BAB	83	0	0	0	0	0
BEA	7	130	16	3	0	22
COM	0	2	131	4	0	8
DEC	14	15	2	136	1	3
HOM	36	3	0	0	137	0
KIT	10	0	1	4	12	117
WAT	0	0	0	3	0	0
	0	1	2	3	4	5
	6	7	8	9	10	11

TSNE_W2V_PROD_LEM						
BAB	88	0	0	1	33	0
BEA	2	117	0	1	0	8
COM	0	4	132	22	0	12
DEC	24	26	14	117	8	50
HOM	35	1	0	0	108	0
KIT	1	2	3	2	0	76
WAT	0	0	1	7	1	4
	0	1	2	3	4	5
	6	7	8	9	10	11

TSNE_ROBERTA_PROD_STEM						
BAB	112	0	0	1	77	0
BEA	3	102	0	1	7	1
COM	2	4	131	14	0	9
DEC	3	39	5	113	0	8
HOM	22	2	14	13	65	1
KIT	5	0	0	0	114	0
WAT	3	3	0	8	1	127
	0	1	2	3	4	5
	6	7	8	9	10	11

TSNE_DISTIL_PROD_STEM						
BAB	104	1	0	0	22	0
BEA	7	126	3	6	0	14
COM	0	5	145	20	4	9
DEC	0	7	1	107	0	19
HOM	19	1	0	2	123	0
KIT	1	1	1	2	0	88
WAT	19	9	0	13	1	20
	0	1	2	3	4	5
	6	7	8	9	10	11

TSNE_USE5_PROD_CLEAN						
BAB	92	0	0	1	0	0
BEA	5	131	0	3	0	2
COM	11	4	146	26	2	10
DEC	4	8	3	111	0	2
HOM	28	2	1	5	147	0
KIT	9	5	0	3	0	136
WAT	1	0	0	2	0	0
	0	1	2	3	4	5
	6	7	8	9	10	11

ARI = 0,68  
Accuracy = 84%

ARI = 0,53  
Accuracy = 75%

ARI = 0,56  
Accuracy = 72%

ARI = 0,59  
Accuracy = 79,5%

ARI = 0,73  
Accuracy = 87%  
22

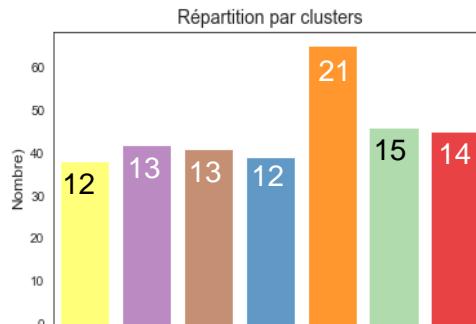


# NLP – Classification Supervisée



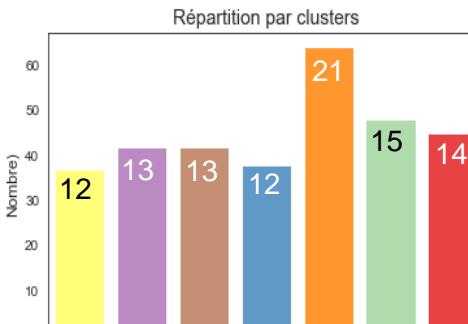
T-SNE DOC2VEC ALL CLEAN

Model	Accuracy
Extra Trees Classifier	0.9652



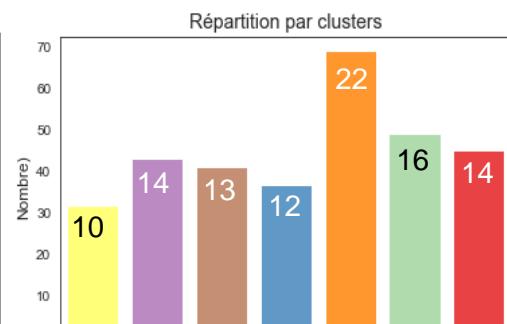
T-SNE USE5 PROD CLEAN

Model	Accuracy
Extra Trees Classifier	0.9620



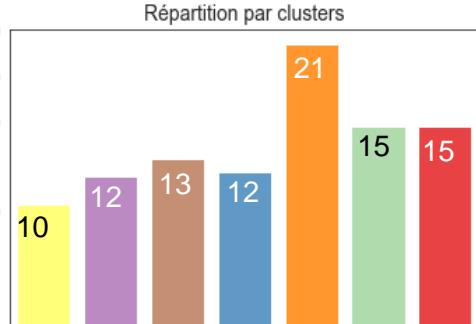
T-SNE ROBERTA PROD LEM

Model	Accuracy
Random Forest Classifier	0.9367



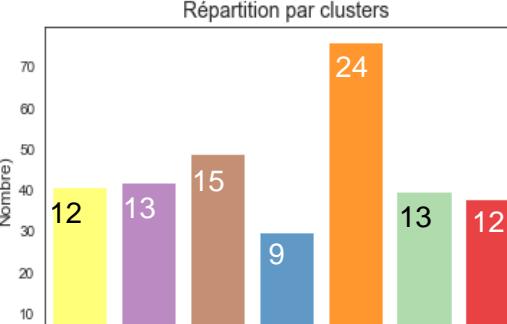
T-SNE DISTILLBERT PROD STEM

Model	Accuracy
Extra Trees Classifier	0.9241



T-SNE ELECTRA PROD CLEAN

Model	Accuracy
Extra Trees Classifier	0.7595



TSNE\_DOC2V\_ALL\_CLEAN

BAB	0	0	0	0	3	0	0
BAB	33	0	0	0	0	0	0
BEA	1	41	0	0	0	1	0
COM	1	0	41	1	0	0	0
DEC	1	0	0	38	0	0	0
HOM	2	0	0	0	62	0	0
KIT	0	1	0	0	0	45	0
WAT	0	0	0	0	0	0	45
	0	1	2	3	4	5	6

TSNE\_USE5\_PROD\_CLEAN

BAB	0	0	0	1	2	1	0
BAB	32	0	0	1	2	1	0
BEA	0	42	0	0	1	0	0
COM	0	0	42	0	1	0	0
DEC	1	0	0	37	0	1	0
HOM	4	0	0	0	60	0	0
KIT	0	0	0	0	0	46	0
WAT	0	0	0	0	0	0	45
	0	1	2	3	4	5	6

TSNE\_ROBERTA\_PROD\_LEM

BAB	0	1	0	1	4	0	0
BAB	30	1	0	1	4	0	0
BEA	0	40	1	0	1	1	0
COM	0	0	40	1	2	0	0
DEC	0	0	0	35	0	2	2
HOM	2	0	0	0	62	0	0
KIT	0	0	0	0	0	46	0
WAT	0	2	0	0	0	0	43
	0	1	2	3	4	5	6

TSNE\_DISTIL\_PROD\_STEM

BAB	0	1	0	1	4	1	0
BAB	29	1	0	1	4	1	0
BEA	1	36	1	2	1	1	1
COM	0	0	41	0	1	1	0
DEC	0	1	0	35	0	1	2
HOM	2	0	0	0	61	0	1
KIT	0	0	0	1	0	45	0
WAT	0	0	0	0	0	0	45
	0	1	2	3	4	5	6

TSNE\_ELEC\_PROD\_CLEAN

BAB	0	3	2	0	5	2	0
BAB	24	3	2	0	5	2	0
BEA	2	32	3	2	3	0	1
COM	0	1	36	1	4	1	0
DEC	4	2	4	23	5	1	0
HOM	5	2	1	2	54	0	0
KIT	2	1	2	2	3	35	1
WAT	4	1	1	0	2	1	36
	0	1	2	3	4	5	6

Accuracy Train = 92%  
Accuracy Test = 96,5%

Accuracy Train = 94,5%  
Accuracy Test = 96%

Accuracy Train = 90%  
Accuracy Test = 94%

Accuracy Train = 91%  
Accuracy Test = 92,5%

Accuracy Train = 71%  
Accuracy Test = 76%

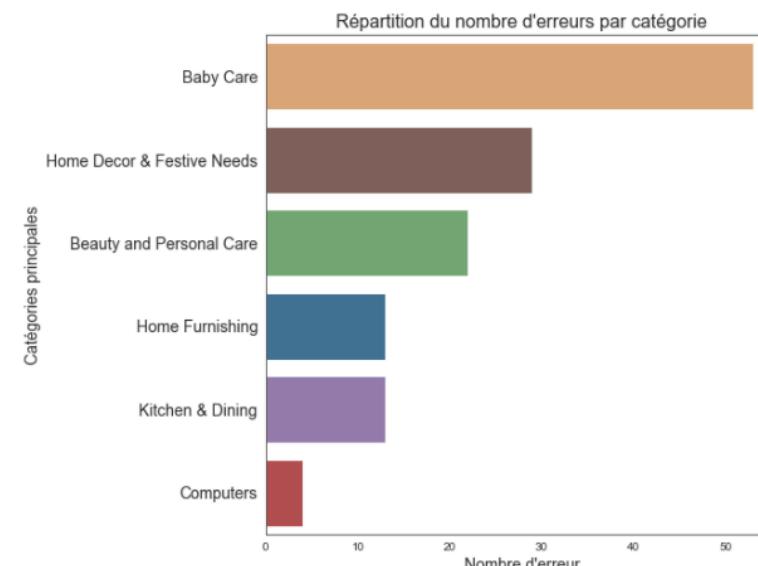
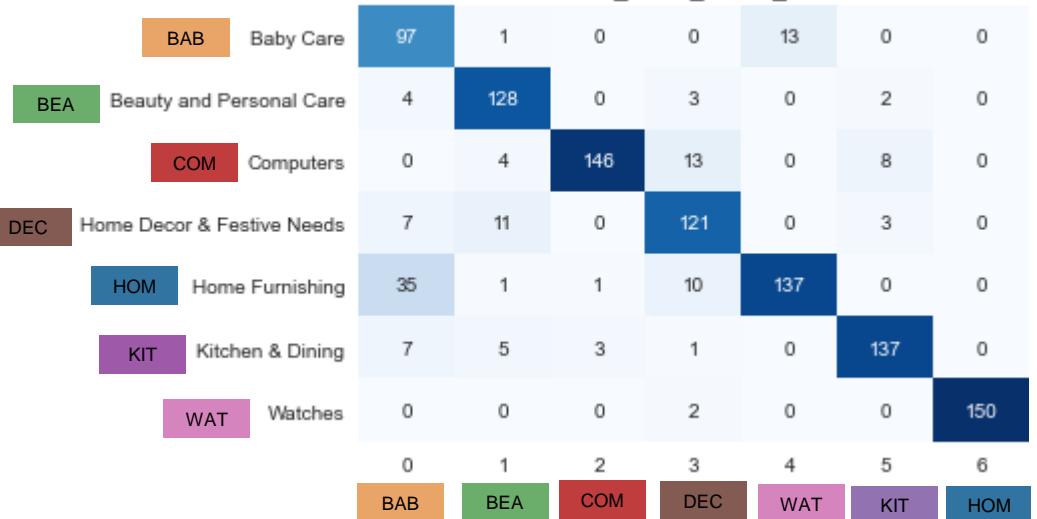


# NLP – USE 5 Erreurs – Non supervisé

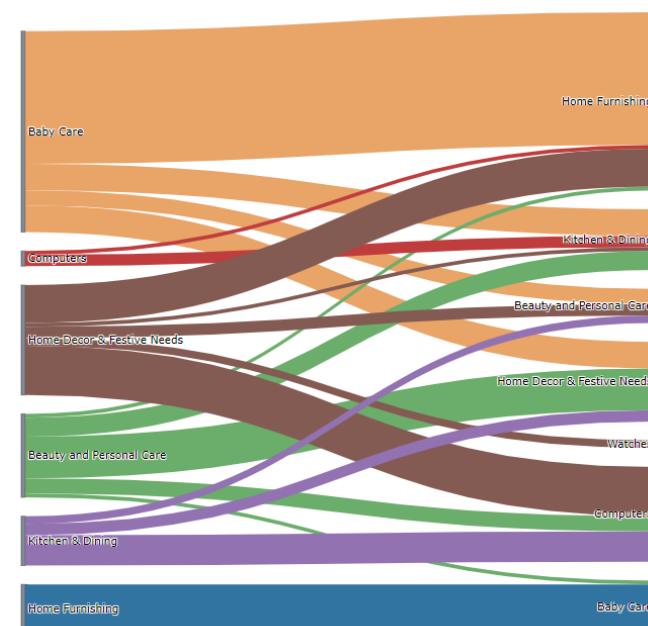
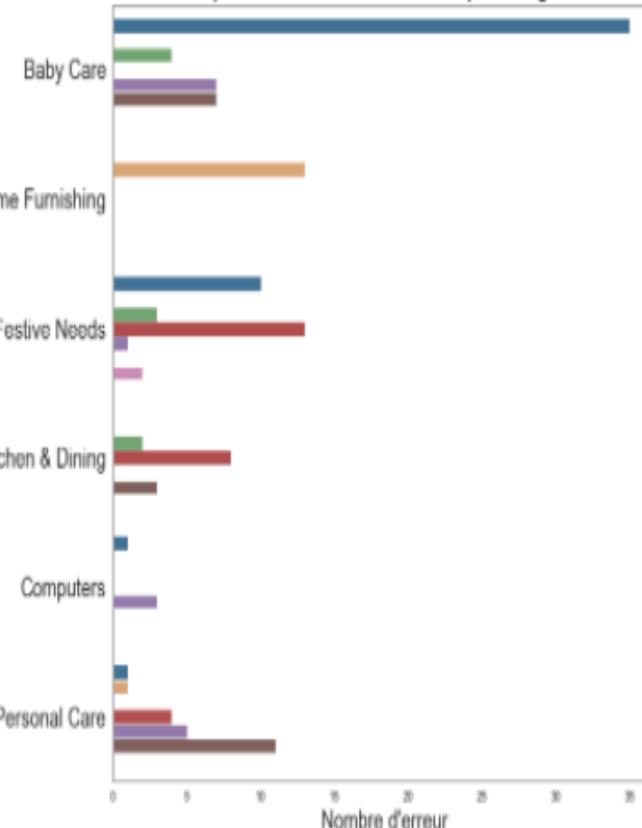


Type_données	ARI	Homogénéité	Complétude	V-measure	Accuracy(%)
TSNE_USE5_PROD_LEM	0.73	0.76	0.76	0.76	87

TSNE\_USE5\_PROD\_LEM

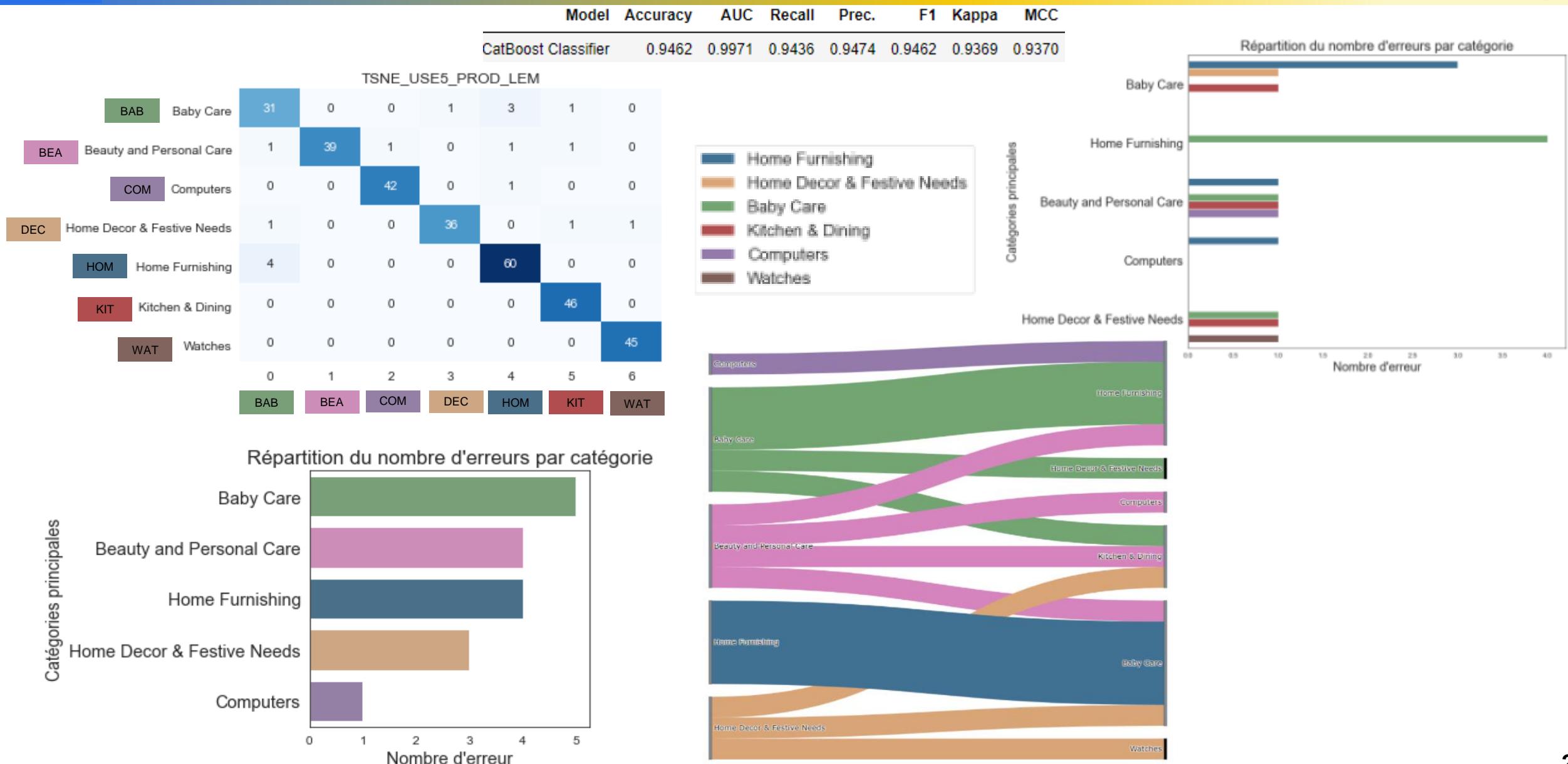


Répartition du nombre d'erreurs par catégorie





# NLP – USE 5 Erreurs – Supervisé





Problématique/Données



Traitement données textuelles



Traitement données images



Combinaison textes/images

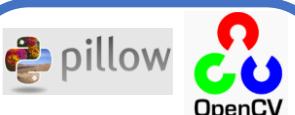


Conclusions



## PRÉ TRAITEMENT

Données IMAGES



Originale

Correction exposition

Correction contraste

Convertir niveaux de gris

Redimensionnement

Originales

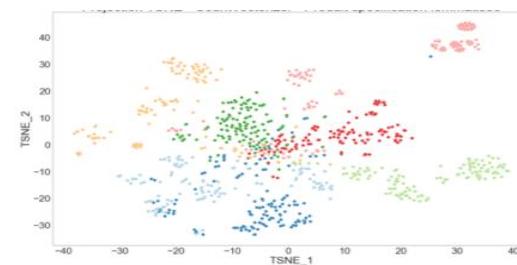
## FEATURES EXTRACTION

Bags Of Visual Words  
SIFT  
ORB

Keras CNN Transfert  
VGG16 VGG19  
Xception  
ResNet152  
ResNet152V2  
InceptionV3  
InceptionResNetV2  
MobileNetV2  
DenseNet201  
NASNetLarge  
EfficientNetB7...

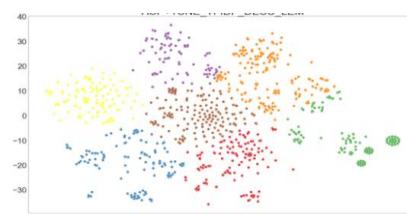
## RÉDUCTION DIMENSION

t-SNE



## CLASSIFICATION

Apprentissage Non supervisé KMeans



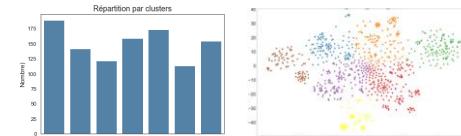
## ÉVALUATION CATÉGORIE

Évaluation ARI Accuracy

	TIME_USED_PROD_CLEAN
Baby Care	11 11 3 3 1 2 0
Beauty and Personal Care	4 10 0 0 0 0 0
Computers	2 3 16 0 0 29 0
Home Decor & Festive Needs	14 3 0 0 18 1 0
Home Furnishing	32 2 0 0 144 10 0
Kitchen & Dining	5 0 0 25 4 17 0
Watches	0 3 2 3 0 1 100
	0 1 2 3 4 5 6

\*Précision: 82.38%

Interprétation clusters



Apprentissage Supervisé Pycaret

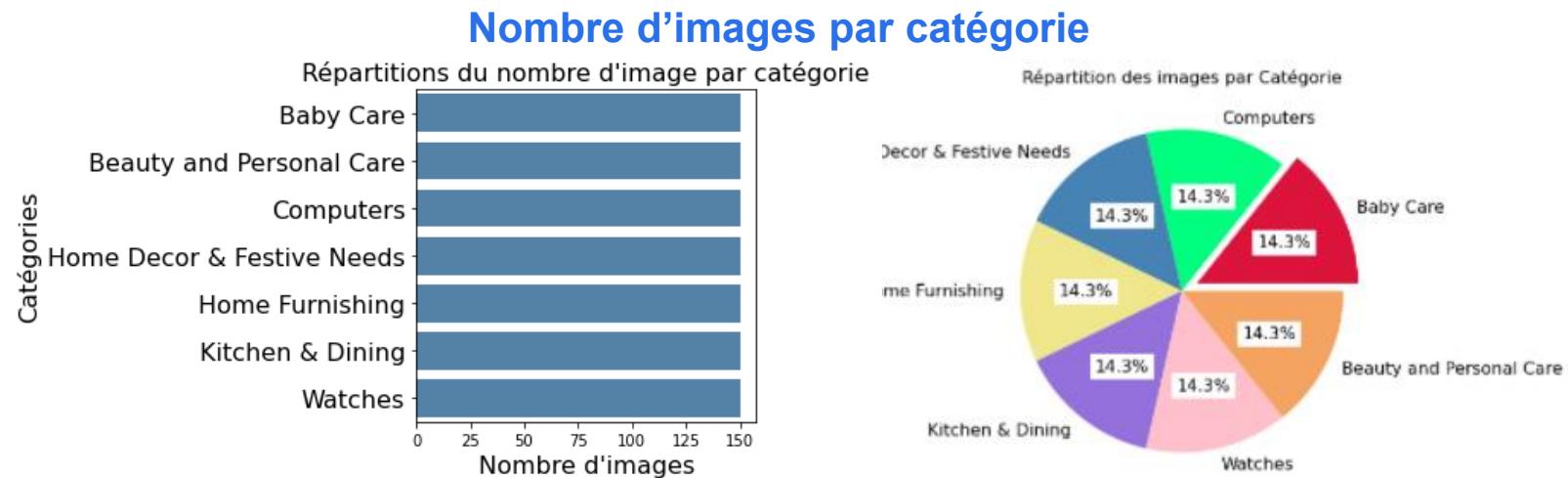
Erreurs



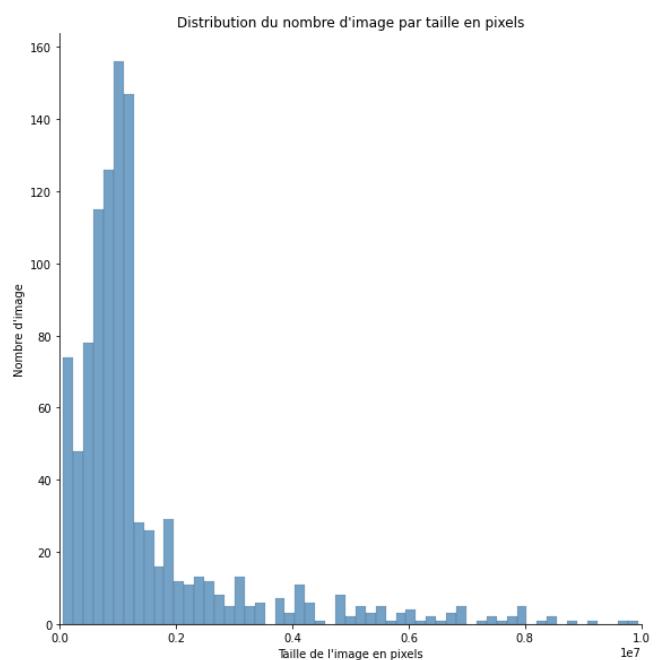
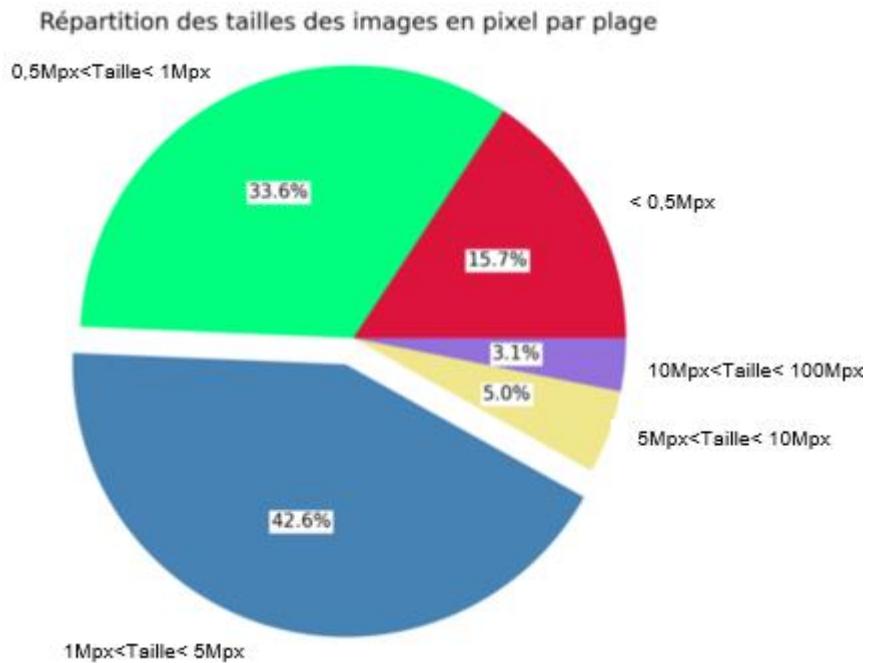
# Images – Analyse exploratoire



## Quelques Exemples



## Taille des images





## Traitement données images

# IMAGES – MÉTHODES CLASSIQUES



# 3 Images - Pipeline Méthodes classiques



PRÉ  
TRAITEMENT

FEATURES  
EXTRACTION

RÉDUCTION  
DIMENSION

CLASSIFICATION

ÉVALUATION  
CATÉGORIE

Données IMAGES

Bag Of Words

Non Supervisé  
KMeans

Évaluation  
ARI  
Accuracy



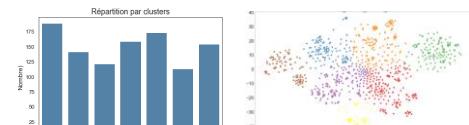
SIFT

ORB

T-SNE

Supervisé  
Pycaret  
Ridge Classifier  
CatBoost Classifier

Interprétation clusters



Erreurs

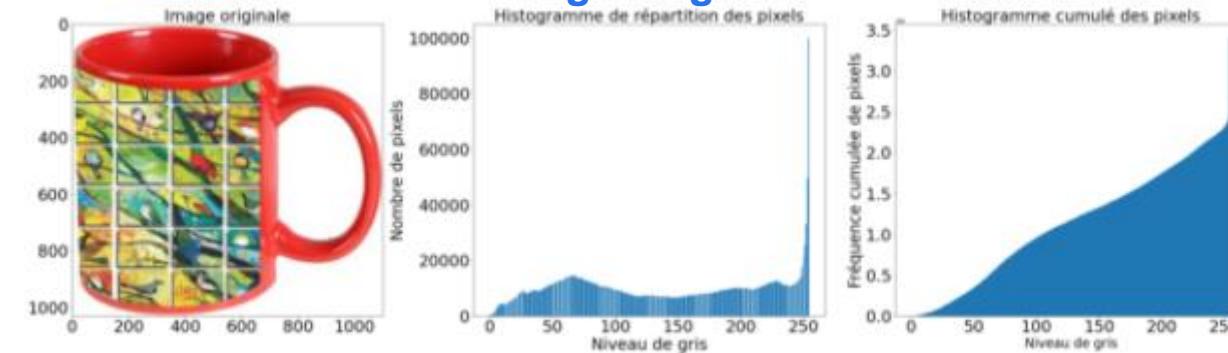




# 3 Images – Pré-traitement



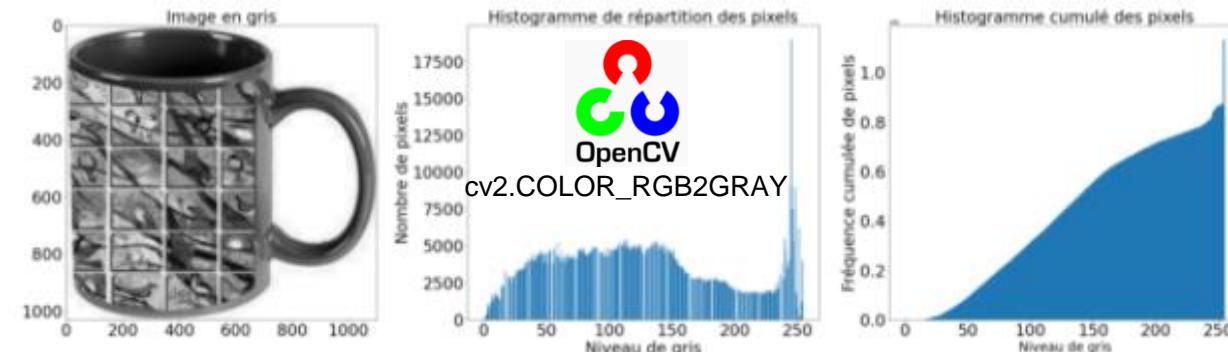
## 0. Image Originale



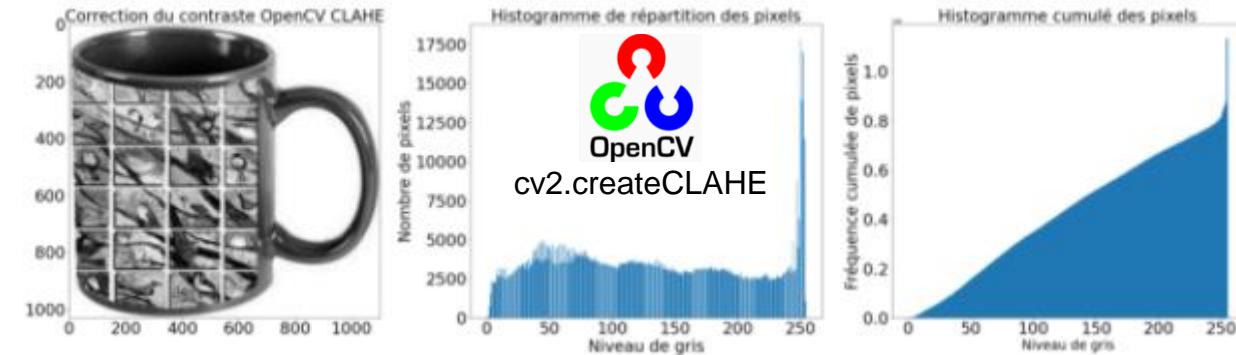
## 1. Correction de l'exposition (étirement d'histogramme)



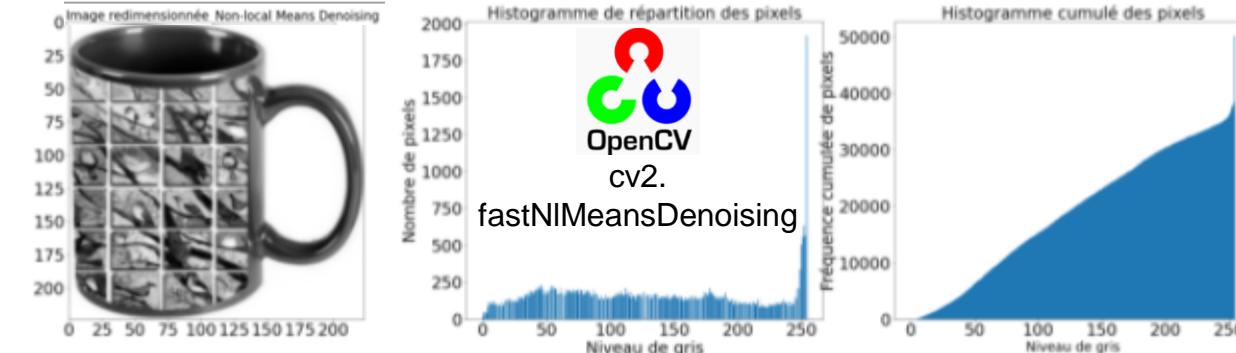
## 2. Conversion de l'image en niveau de gris



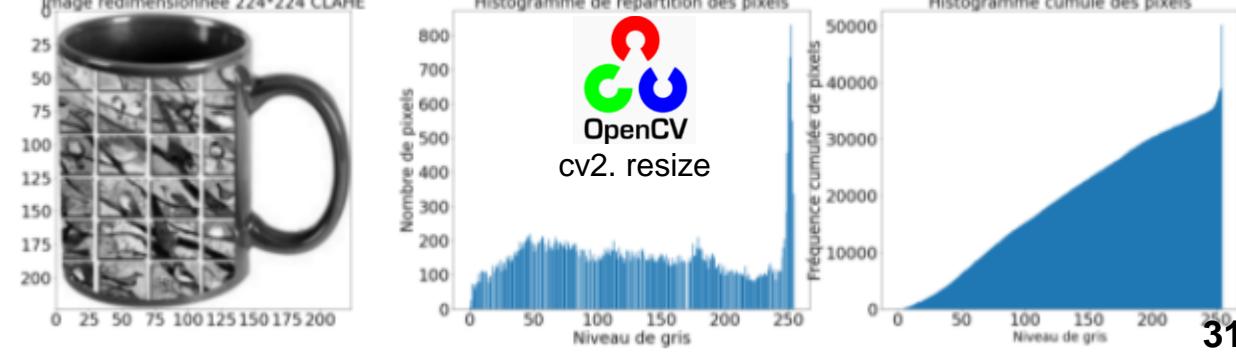
## 3. Correction du contraste (égalisation d'histogramme)



## 4. Réduction du bruit Algo. Non-local Means Denoising



## 5. Redimensionnement en 224 \* 224

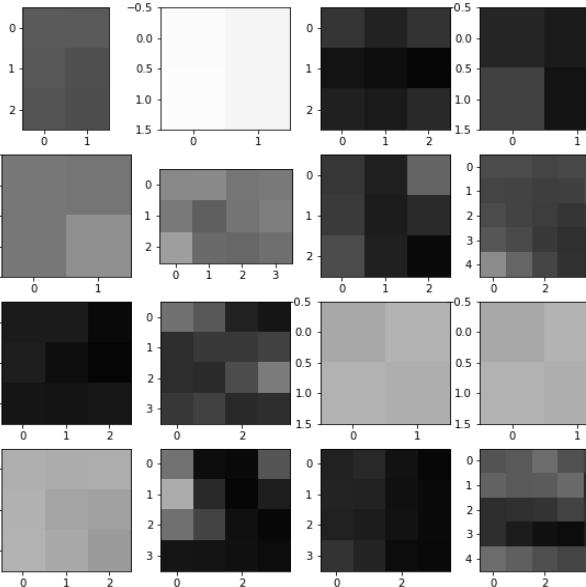
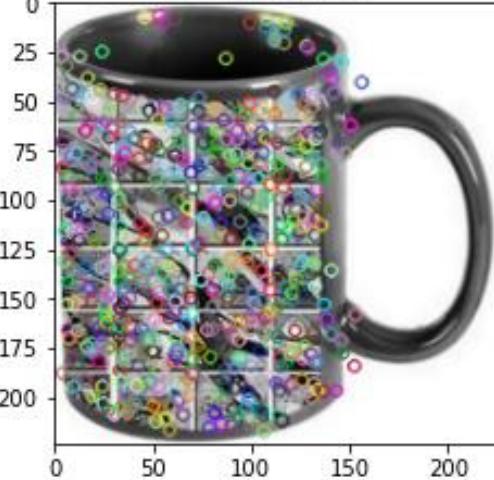


# Images – SIFT & ORB Visual Words

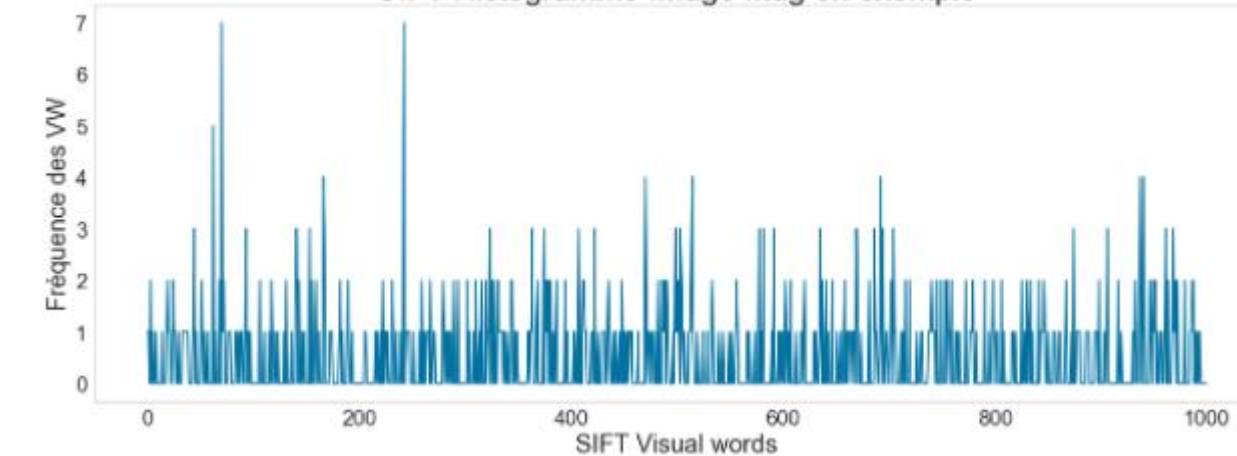


## SIFT

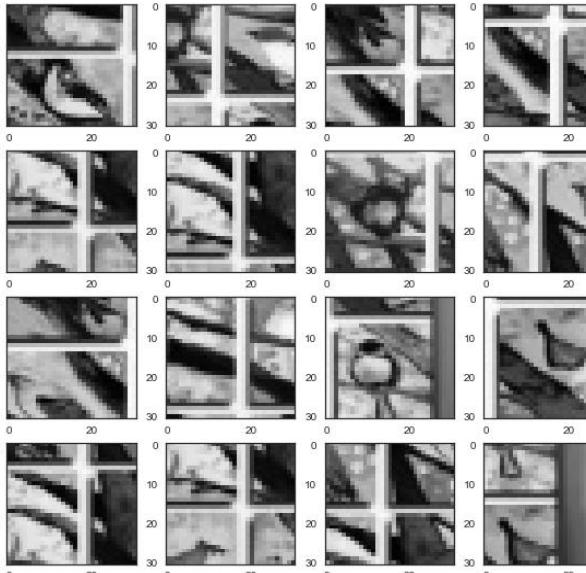
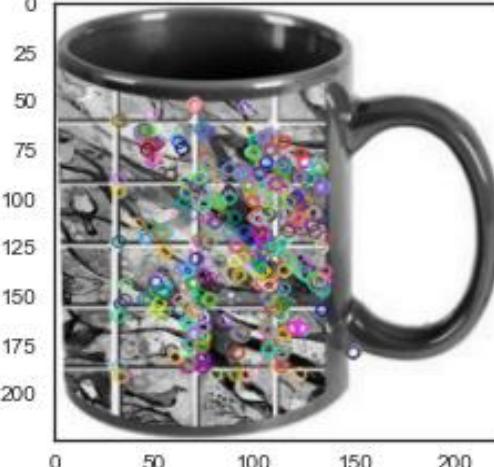
SIFT Points clés



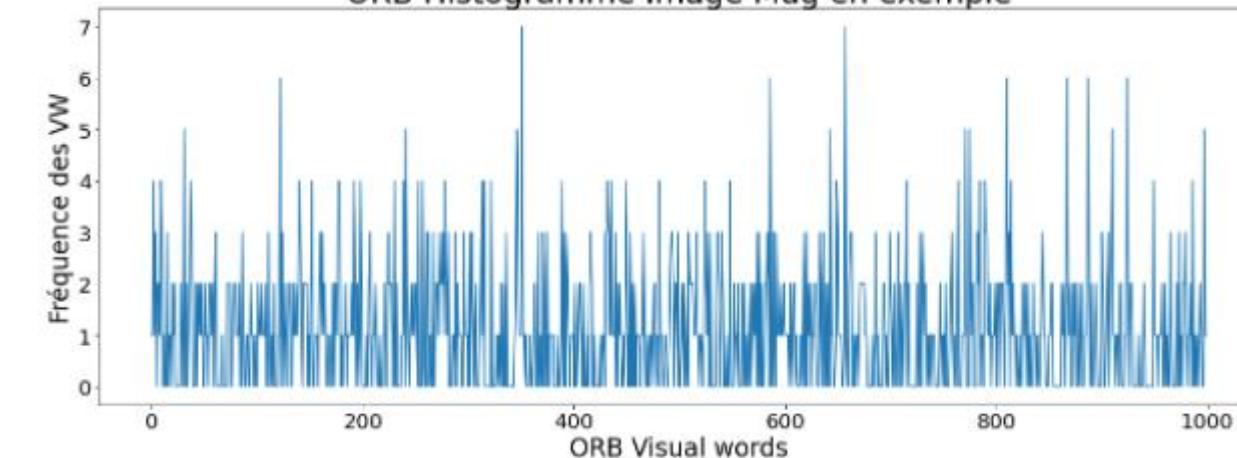
SIFT Histogramme Image Mug en exemple



ORB Points clés



ORB Histogramme Image Mug en exemple



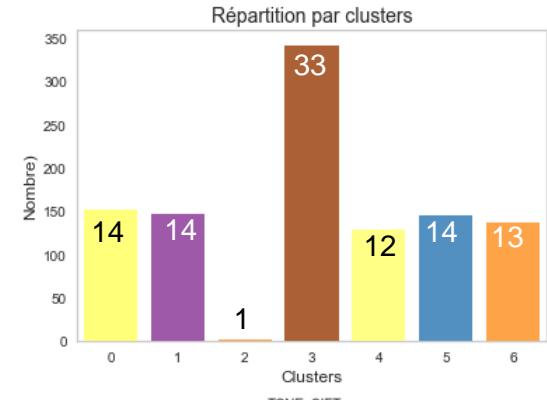
## ORB



# 3f Images - Classification

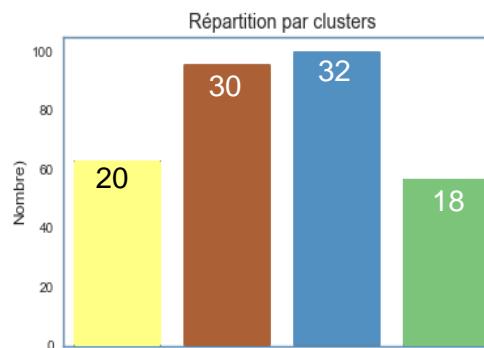


Non supervisée KMeans



SIFT

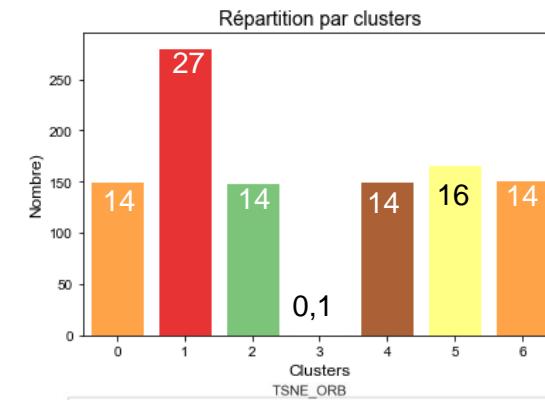
Supervisée



Model Accuracy

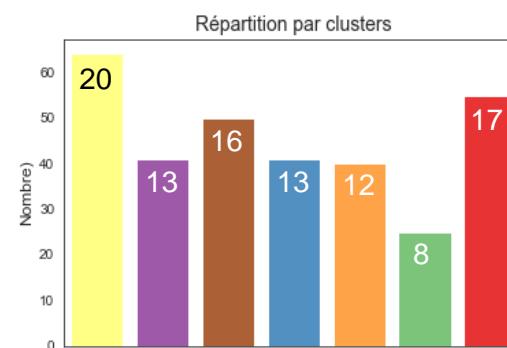
Ridge Classifier 0.1139

Non supervisée KMeans



ORB

Supervisée



Model Accuracy

CatBoost Classifier 0.2975

BAB	73	30	34	33	42	34	34
BEA	8	33	20	23	20	16	26
COM	22	51	71	40	24	70	64
DEC	29	12	0	49	32	18	5
HOM	18	24	25	5	32	12	21
KIT	0	0	0	0	0	0	0
WAT	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
BAB	4	0	12	12	0	8	0
BEA	8	0	9	19	0	7	0
COM	10	0	14	14	0	5	0
DEC	10	0	13	10	0	6	0
HOM	12	0	21	18	0	13	0
KIT	11	0	12	15	0	8	0
WAT	8	0	15	12	0	10	0

ARI = 0,035  
Accuracy = 24,5%

Accuracy Train = 17%  
Accuracy Test = 11%

ARI = 0,029  
Accuracy = 25%

Accuracy Train = 26%  
Accuracy Test = 30%

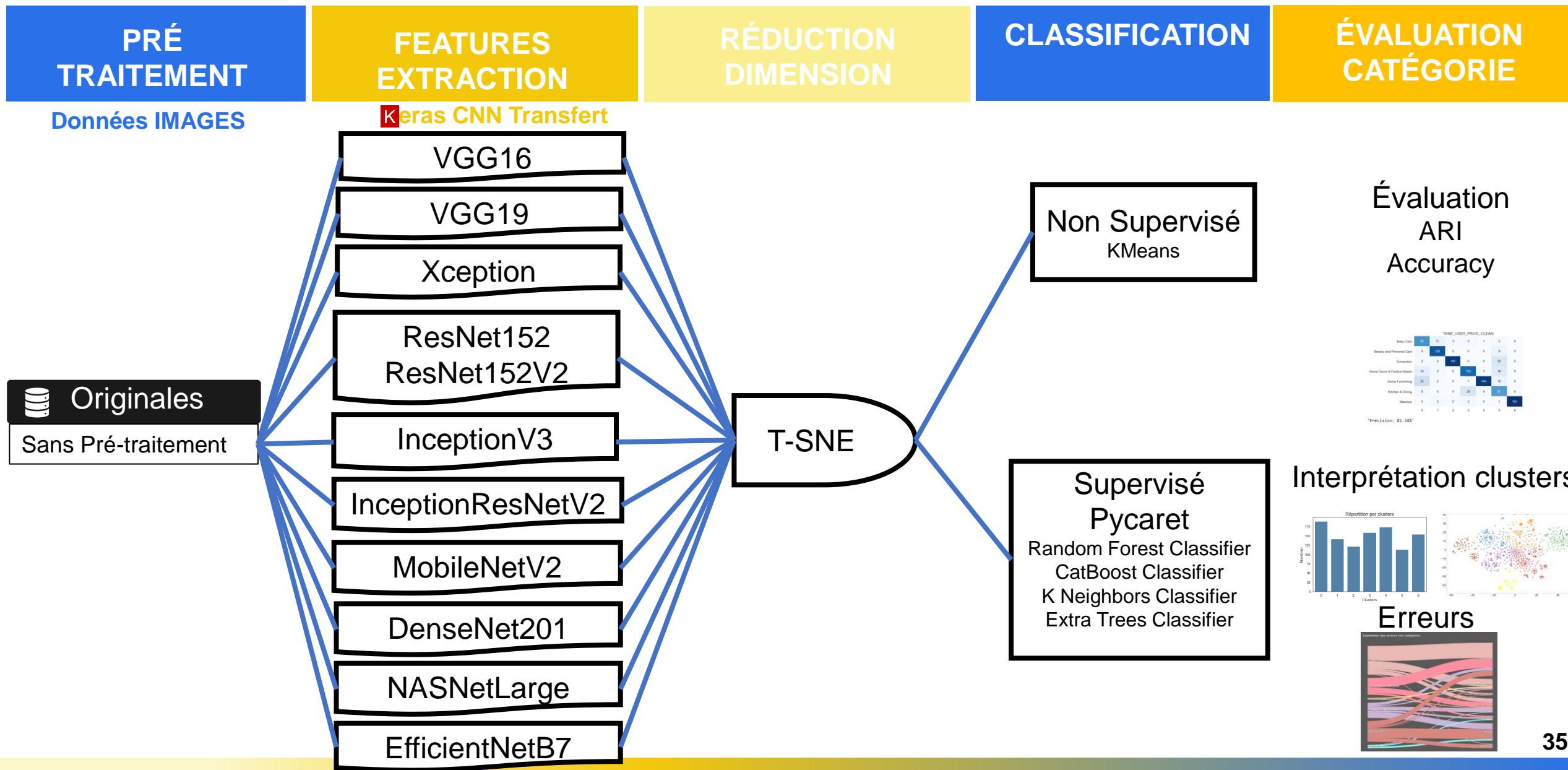


## Traitement données images

# **IMAGES – CNN Transfert Learning**



# 3 Images - Pipeline Réseaux de neurones CNN

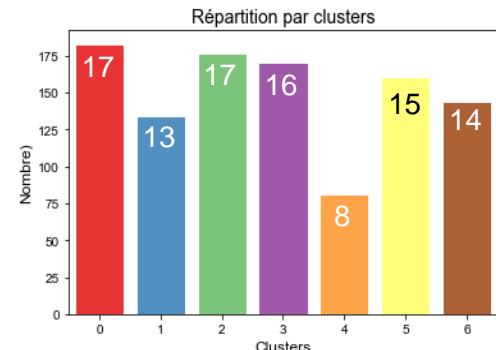




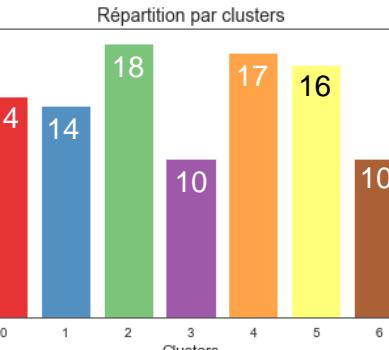
# Images Classification Non Supervisée KMeans



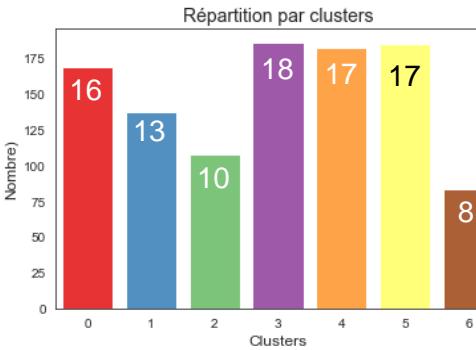
TSNE XCEPTION



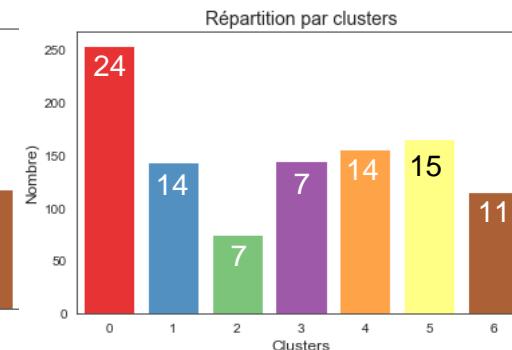
TSNE INCEPTIONV3



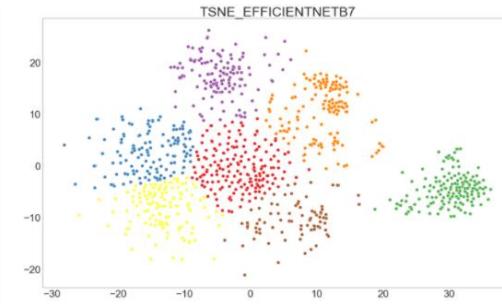
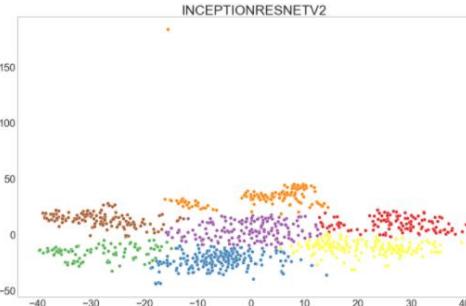
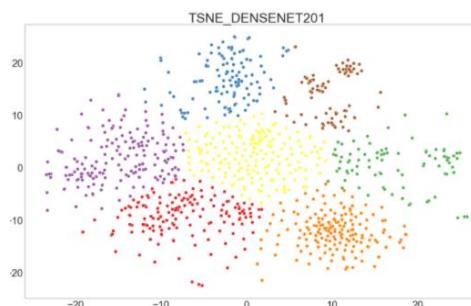
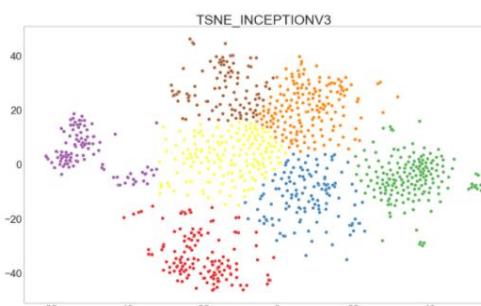
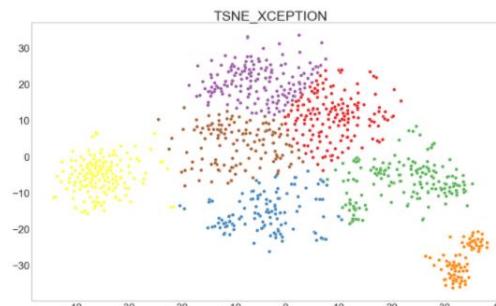
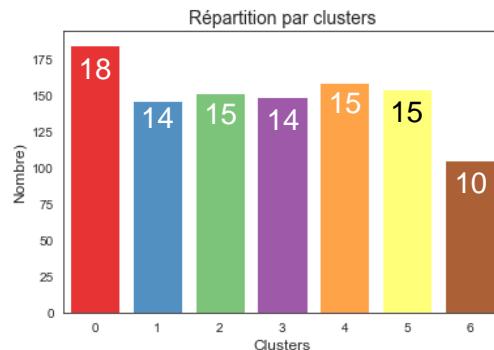
T-SNE DENSENET201



T-SNE INCEPTIONRESNETV2



T-SNE EFFICIENTB7



TSNE_XCEPTION							
BAB	112	5	2	18	32	1	0
BEA	5	119	2	9	1	41	0
COM	2	5	103	10	0	14	0
DEC	8	9	35	75	3	12	2
HOM	22	10	7	27	114	3	0
KIT	1	0	0	1	0	79	0
WAT	0	2	1	10	0	0	148

TSNE_INCEPTIONV3							
BAB	112	5	4	18	39	2	0
BEA	4	124	4	3	0	16	0
COM	1	1	115	19	0	8	0
DEC	23	13	9	89	20	16	2
HOM	9	1	1	6	90	1	0
KIT	1	0	0	5	0	102	0
WAT	0	6	17	10	1	5	148

TSNE_DENSENET201							
BAB	91	1	5	60	10	2	0
BEA	4	116	7	6	0	4	0
COM	0	4	93	1	0	7	3
DEC	16	13	35	60	2	56	3
HOM	35	6	1	9	135	0	0
KIT	0	3	0	3	1	76	0
WAT	4	7	9	11	2	5	144

INCEPTIONRESNETV2							
BAB	87	1	4	13	29	0	0
BEA	3	107	5	3	0	20	0
COM	1	2	95	1	0	1	4
DEC	18	24	16	86	12	14	0
HOM	33	7	3	18	109	2	0
KIT	3	1	1	16	0	108	0
WAT	5	8	26	13	0	5	146

TSNE_EFFICIENTNETB7							
BAB	0	0	0	0	0	0	0
BEA	10	112	12	4	0	11	0
COM	2	3	92	2	0	2	4
DEC	22	9	28	101	8	16	1
HOM	108	10	11	24	141	6	0
KIT	7	16	7	14	1	114	0
WAT	1	0	0	5	0	1	145

ARI = 0,48  
Accuracy = 71,5%

ARI = 0,51  
Accuracy = 74%

ARI = 0,47  
Accuracy = 68%

ARI = 0,45  
Accuracy = 70%

ARI = 0,45  
Accuracy = 67%

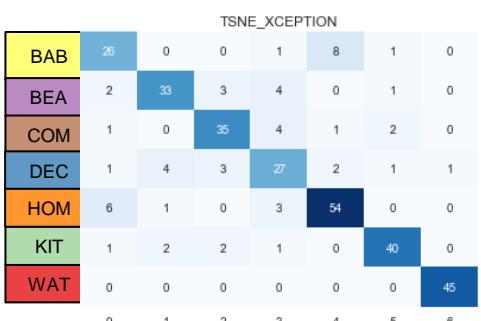
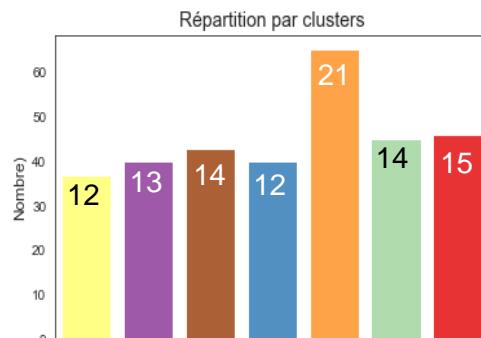


# Images - Classification Supervisée



**TSNE XCEPTION**

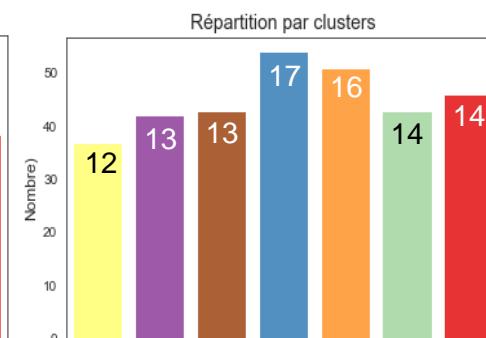
Model	Accuracy
K Neighbors Classifier	0.8228



Accuracy Train = 83%  
Accuracy Test = 82%

**TSNE INCEPTIONV3**

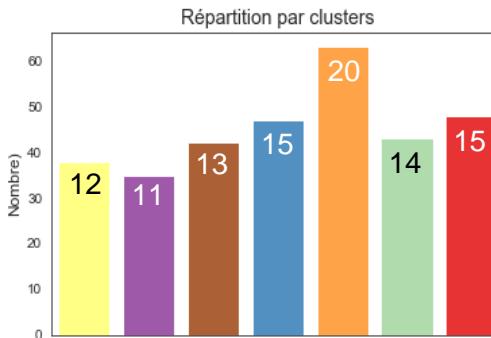
Model	Accuracy
CatBoost Classifier	0.7785



Accuracy Train = 80%  
Accuracy Test = 78%

**T-SNE DENSENET201**

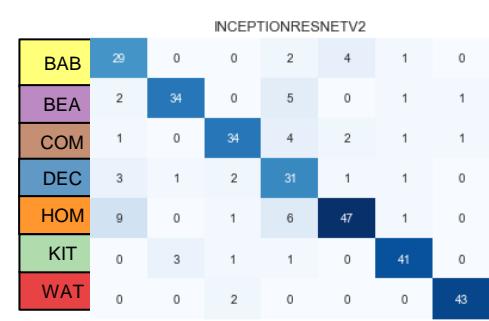
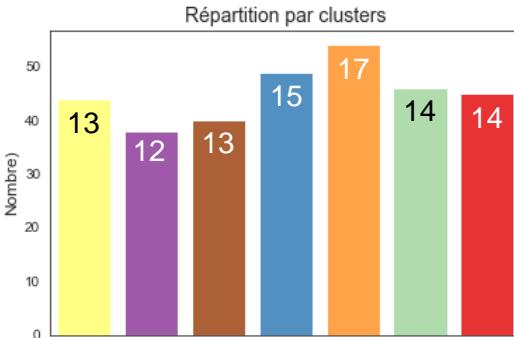
Model	Accuracy
Extra Trees Classifier	0.8639



Accuracy Train = 81%  
Accuracy Test = 86%

**T-SNE INCEPTIONRESNET2**

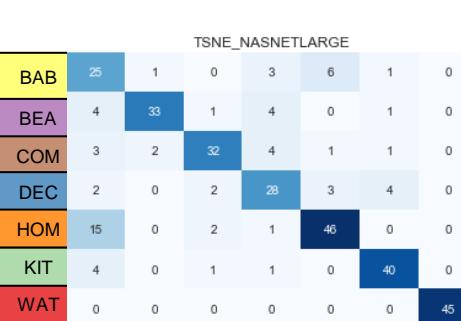
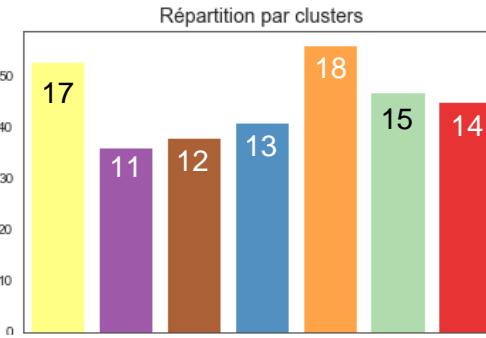
Model	Accuracy
Random Forest Classifier	0.8196



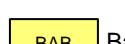
Accuracy Train = 81%  
Accuracy Test = 82%

**T-SNE EFFICIENTB7**

Model	Accuracy
CatBoost Classifier	0.7880



Accuracy Train = 78%  
Accuracy Test = 79%



BAB Baby Care

BEA Beauty and Personal Care

COM Computers

DEC Home Decor & Festive Needs

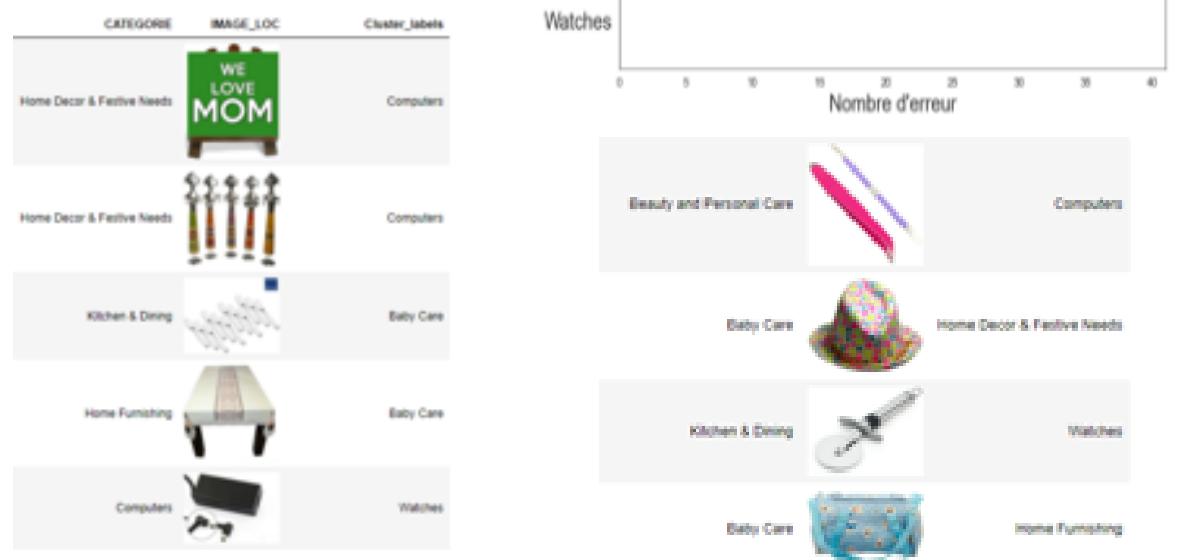
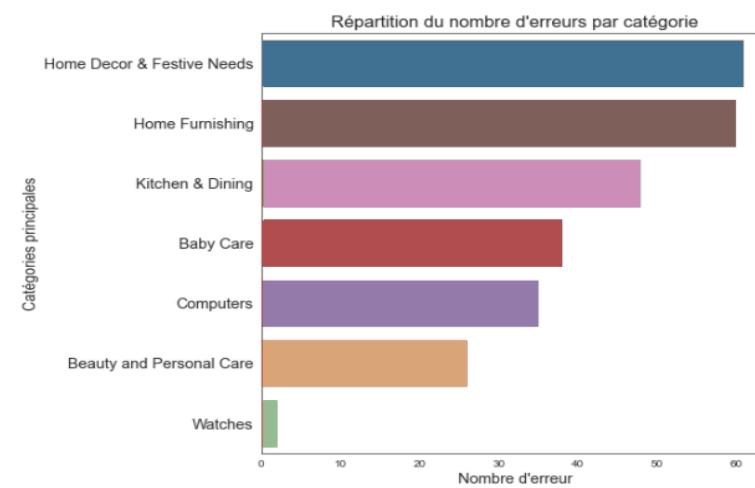
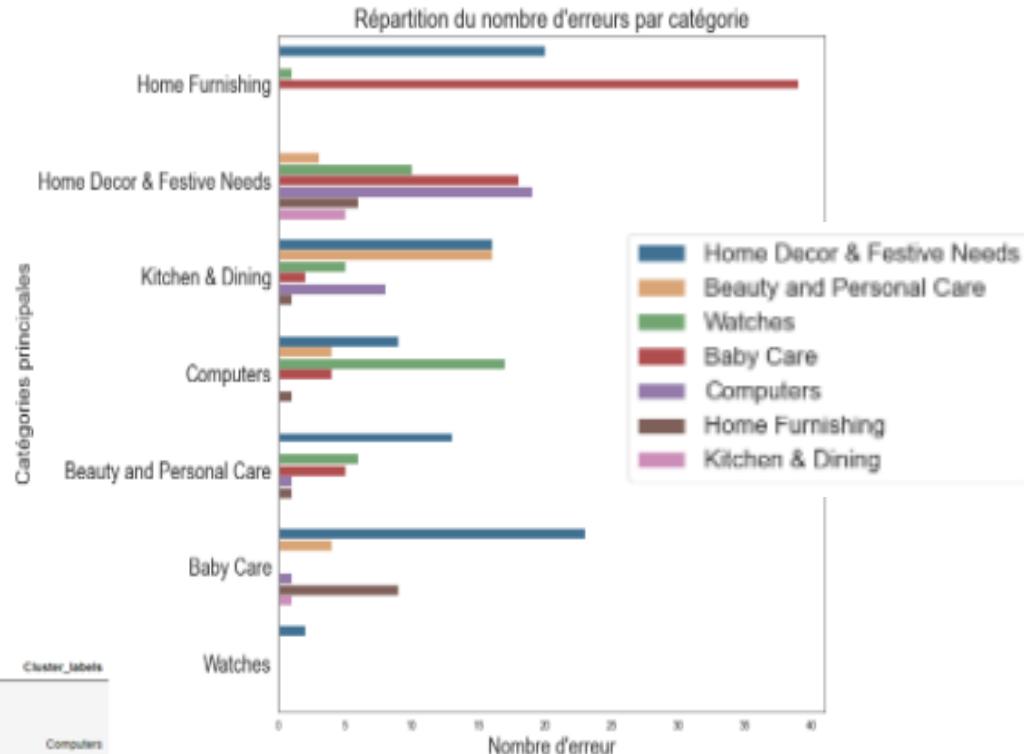
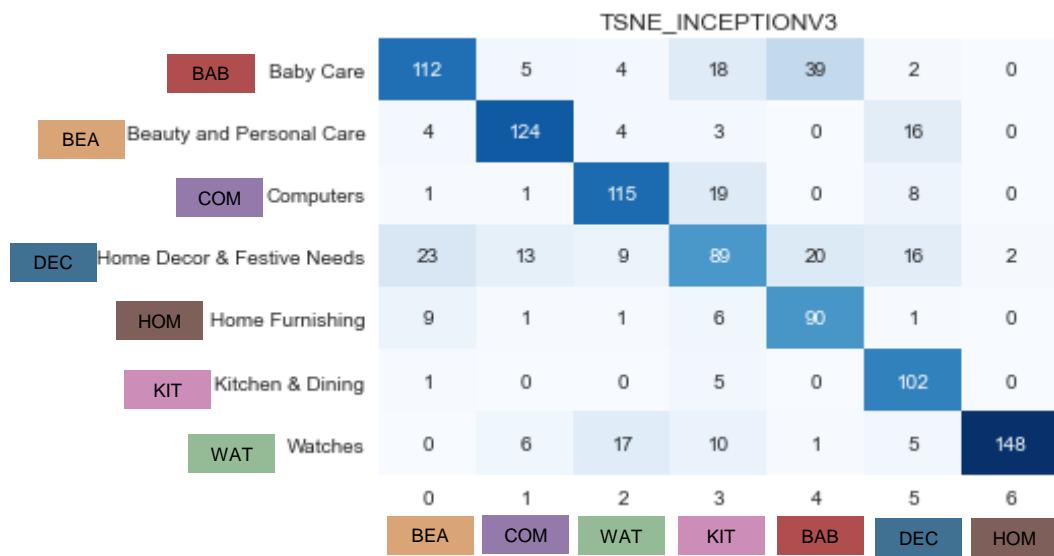
HOM Home Furnishing

KIT Kitchen & Dining

WAT Watches



# 3 Images – InceptionV3 Erreurs – Non supervisé





Problématique/Données



Traitement données textuelles



Traitement données images



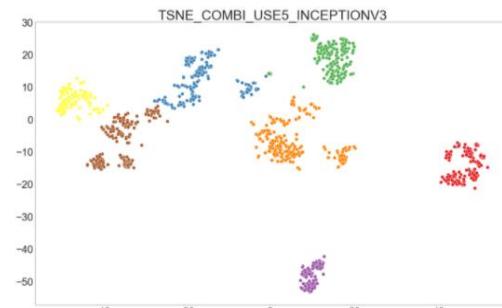
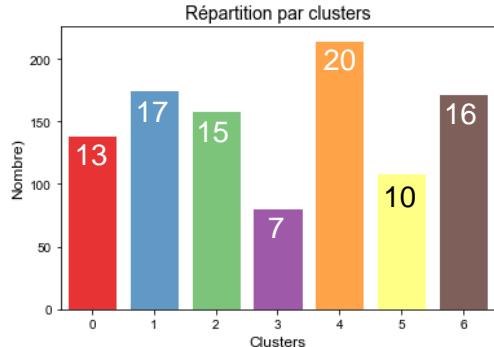
Combinaison textes/images



Conclusions

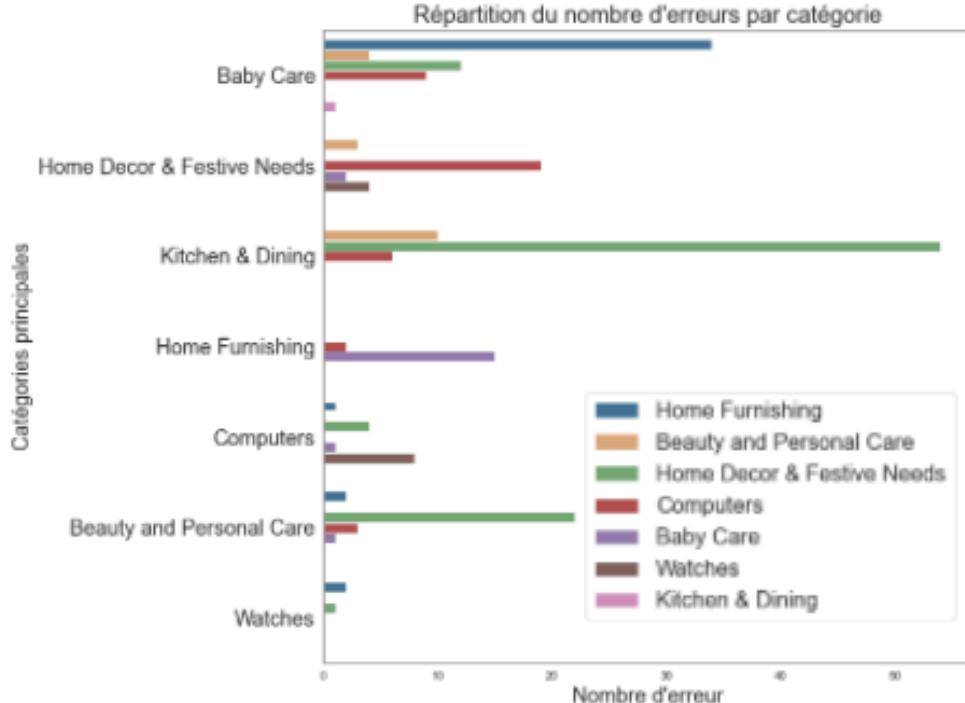
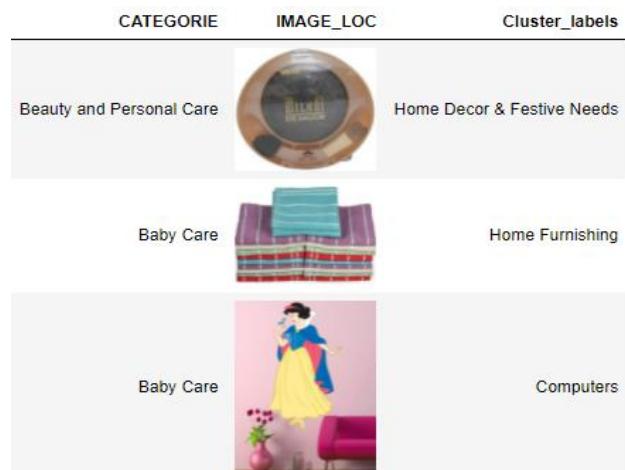
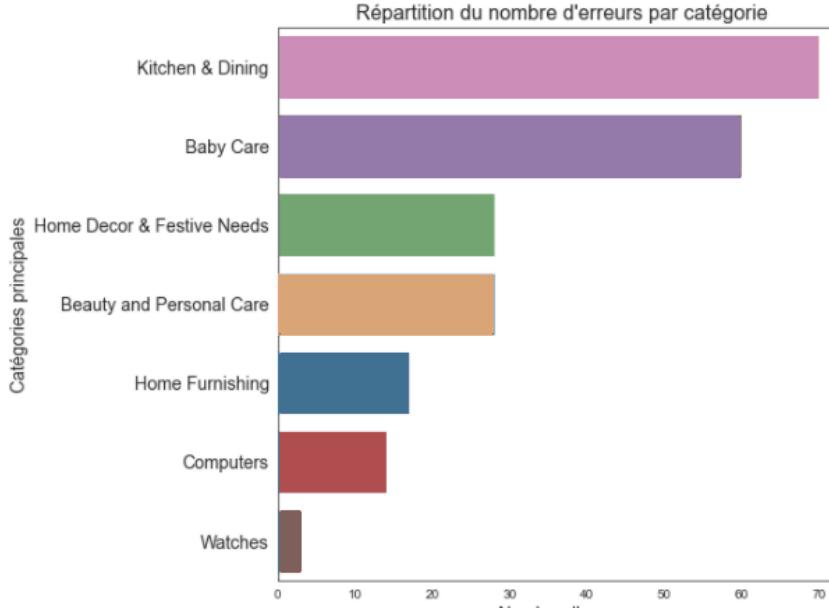
# Combinaison USE 5 textes & InceptionV3 img

**Use** (version 5, textes, ARI=73, Accuracy= 0.87) + **InceptionV3** (Images, ARI=0.51, Accuracy= 0.74)



	BAB	1	1	2	15	0	0
BAB	90	1	1	2	15	0	0
BEA	4	122	0	3	0	10	0
COM	9	3	136	19	2	6	0
DEC	12	22	4	122	0	54	1
HOM	34	2	1	0	133	0	2
KIT	1	0	0	0	0	80	0
WAT	0	0	8	4	0	0	147

ARI = 0,6  
Accuracy = 79%





Problématique/Données



Traitement données textuelles



Traitement données images



Combinaison textes/images

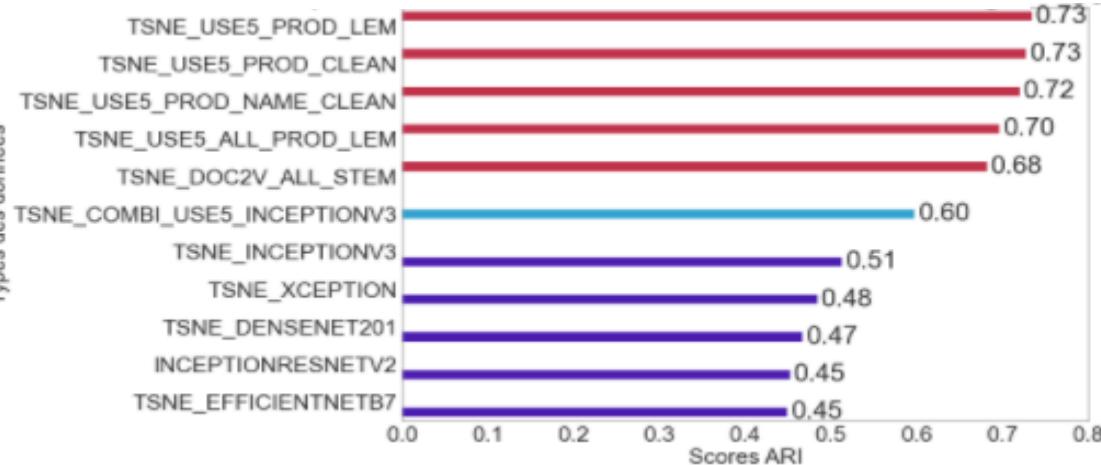


Conclusions

# Conclusions – Bilan final

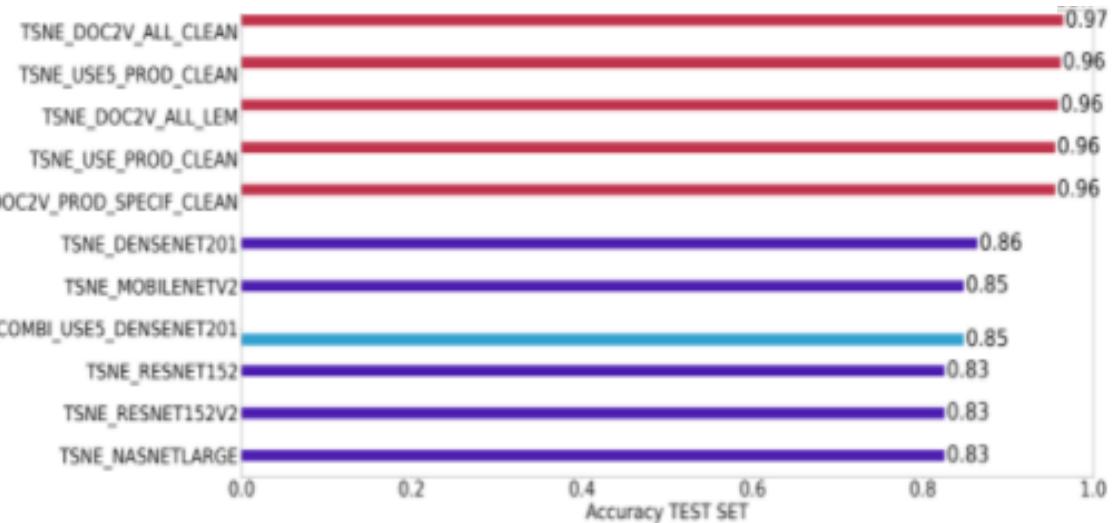
## Apprentissage non supervisé ARI

Types des données



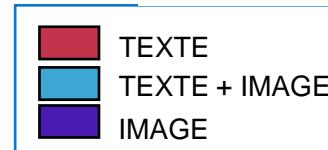
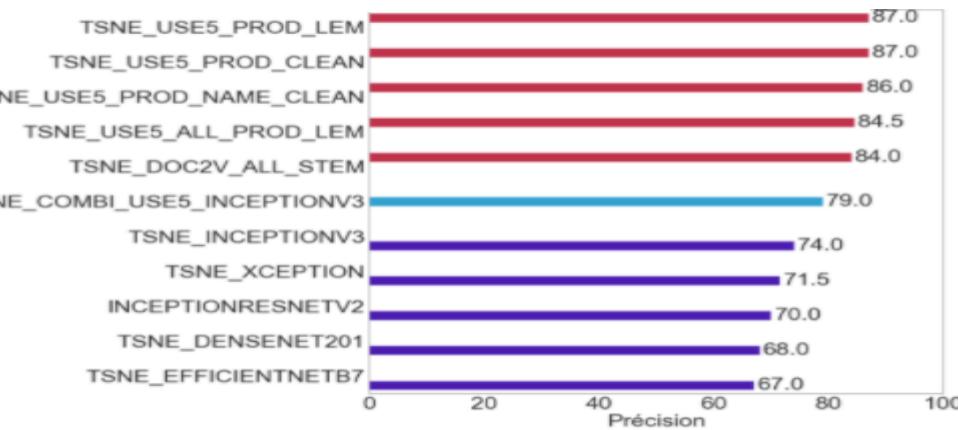
## Apprentissage supervisé Accuracy Test Set

Types des données



## Accuracy

Type des données



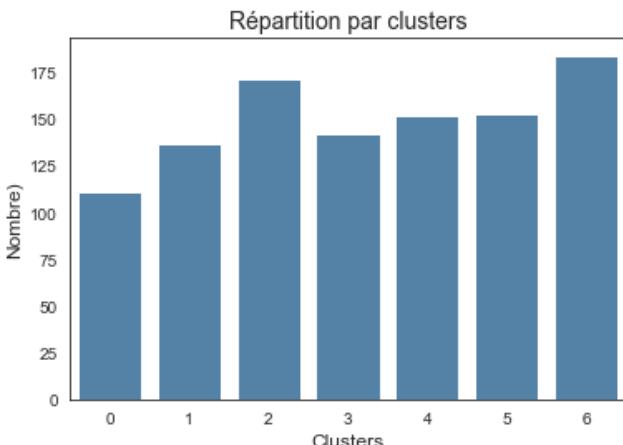
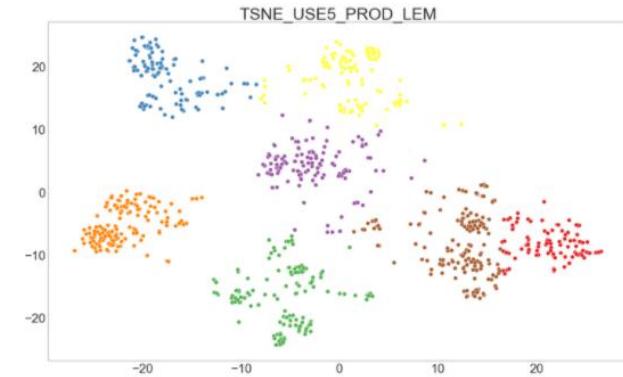
Données  
Textuelles





Meilleur	Données textuelles	Données images	Combinaison textuelles + images
<b>Apprentissage NON SUPERVISÉ</b>			
Modèle	<b>USE 5</b>	InceptionV3	USE 5 + InceptionV3
ARI	<b>0,73</b>	0,51	0,6
Accuracy	<b>87%</b>	74%	79%

Apprentissage SUPERVISÉ			
Modèle	USE 5	DenseNet201	USE 5 + DenseNet201
Accuracy Test	<b>96%</b>	86%	85%
Accuracy Train	<b>94%</b>	81%	81%



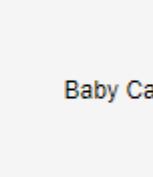
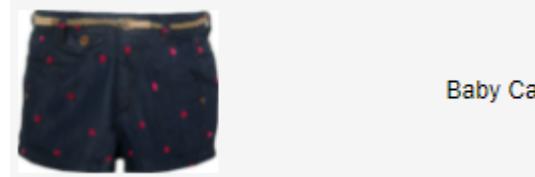
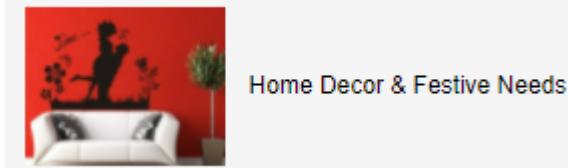
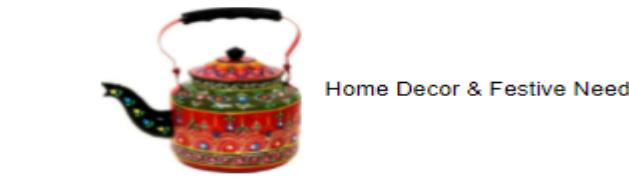
TSNE_USE5_PROD_LEM								
Baby Care	97	1	0	0	13	0	0	
Beauty and Personal Care	4		128	0	3	0	2	0
Computers	0	4		146	13	0	8	0
Home Decor & Festive Needs	7	11	0		121	0	3	0
Home Furnishing	35	1	1	10		137	0	0
Kitchen & Dining	7	5	3	1	0		137	0
Watches	0	0	0	2	0	0		150
	0	1	2	3	4	5	6	

# 5 Conclusions - Prolongements



Jeu de données images :  
ajouter des images (150 seulement par catégorie).

Étiquettes des catégories :  
nombreuses erreurs entre les catégories Home Furnishing et Home Decor & Festive Needs.  
vérifier les étiquettes avant d'envisager d'utiliser l'apprentissage supervisé



Baby Care





# Annexes





# Annexe – NLP – FE - CountVectorizer



Librairie scikit-learn.

*document1*

*document2*

*document3*

document = [ “One Geek helps Two Geeks”, “Two Geeks help Four Geeks”, “Each Geek helps many other Geeks at GeeksforGeeks.” ]

Convertit une collection de documents texte en une matrice de comptage de tokens :

	at	each	four	geek	geeks	geeksforgeeks	help	helps	many	one	other	two
document[0]	0	0	0	1	1	0	0	1	0	0	0	1
document[1]	0	0	1	0	2	0	1	0	0	0	0	1
document[2]	1	1	0	1	1	1	0	1	1	0	1	0

Cette implémentation produit une représentation éparse, creuse du nombre de tokens :

La fonction "count vectorizer" fournit un index à chaque mot et génère un vecteur qui contient le nombre d'apparitions de chaque mot dans un morceau de texte.

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	0	0	1	0	0	0	1
0	0	1	0	2	0	1	0	0	0	0	1
1	1	0	1	1	1	1	0	1	1	0	1

fit permet d'apprendre le vocabulaire d'un ensemble de textes.

Vocabulary: {‘one’: 9, ‘geek’: 3, ‘helps’: 7, ‘two’: 11, ‘geeks’: 4, ‘help’: 6, ‘four’: 2, ‘each’: 1, ‘many’: 8, ‘other’: 10, ‘at’: 0, ‘geeksforgeeks’: 5}

transform permet la transformation en un cadre de données pouvant être utilisé pour construire des modèles de machine learning.

```
[ [0 0 0 1 1 0 0 1 0 1 0 1]
  [0 0 1 0 2 0 1 0 0 0 0 1]
  [1 1 0 1 1 1 0 1 1 0 1 0] ]
```



# Annexe – NLP – FE - TfidfVectorizer



tf-idf (term frequency-inverse document frequency)

$$tfidf = \frac{\text{occurrence du mot dans le document}}{\text{nombre de mots du document}} * \log\left(\frac{\text{nombre de documents dans le corpus}}{\text{nombre de documents dans lequel le terme apparaît}}\right)$$

## 0. Documents Originaux

$d_1$ : "The sky is blue."

$d_2$ : "The sun is bright today."

$d_3$ : "The sun in the sky is bright."

$d_4$ : "We can see the shining sun, the bright sun."

## 1. Pré-traitement

$d_1$ : "sky blue"

$d_2$ : "sun bright today"

$d_3$ : "sun sky bright"

$d_4$ : "can see shining sun bright sun"

Notes :

**TF** : plus le mot est présent dans le document, plus il est important.

Son TFIDF sera fort.

**IDF** : plus le mot est présent dans un corpus, moins il est important.

Son TFIDF sera faible

$$TFIDF_{t,d,D} = \underbrace{TF_{t,d}}_{\text{Importance d'un terme } t \text{ dans un document } d} \times \underbrace{IDF_{t,D}}_{\substack{\text{Fréquence d'un terme } t \text{ dans un document } d \\ \text{Importance du terme } t \text{ dans l'ensemble des documents } D}}$$

2. Pour calculer le **TF**, Fréquence du mot  $t$  dans le document  $d$ . Nous trouvons donc la matrice document-mot, puis nous normalisons les lignes pour que leur somme soit égale à 1.

3. Calculer l'**IDF** : Trouvez le nombre de documents dans lesquels chaque mot apparaît, puis calculez la formule suivante

4. Calculer **TF-IDF** : Multipliez les scores TF et IDF.

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1	1	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0
n.t	1	3	1	1	1	2	3	1

$\times$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.301	0.125	0.602	
	0.602	0.125	0.602	0.602	0.301	0.125	0.602	

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1	1	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0
n.t	1	3	1	1	1	2	3	1

	blue	bright	can	see	shining	sky	sun	today
1	0.301	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	0.201
3	0	0.0417	0	0	0	0	0.100	0.0417
4	0	0.0209	0.100	0.100	0.100	0	0.0417	0
n.t	1	3	1	1	1	2	3	1

Chaque token correspond directement à une position de colonne dans une matrice, dont la taille est prédéfinie.

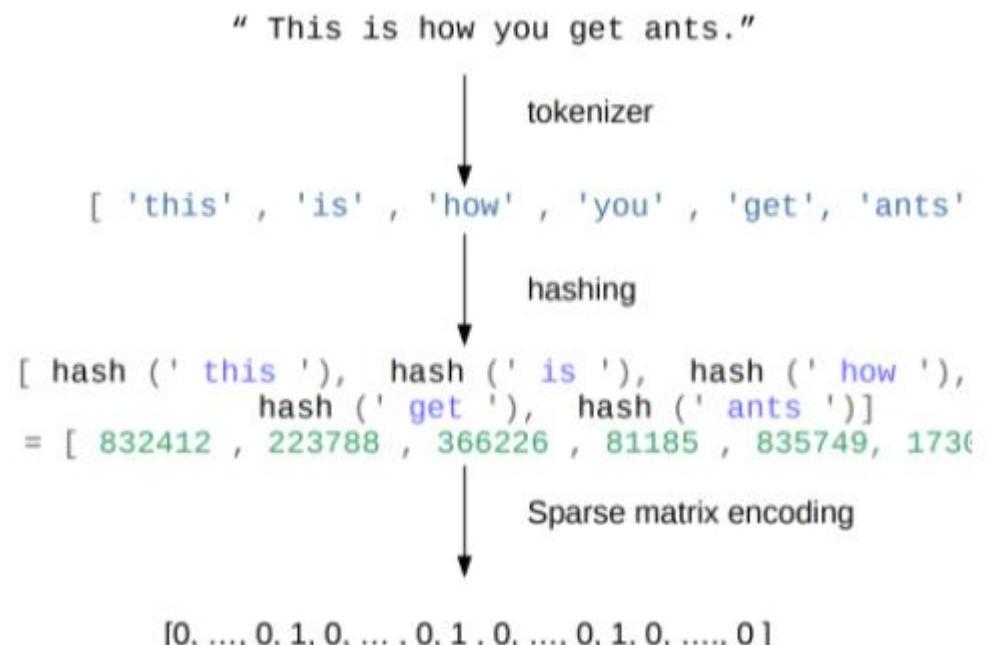
Par exemple, si vous avez 10 000 colonnes dans votre matrice, chaque jeton correspond à une des 10 000 colonnes.

Cette correspondance se fait par hachage pour trouver la correspondance entre le nom de la chaîne de caractères et l'index entier de la caractéristique.

La fonction de hachage utilisée est appelée Murmurhash3.

Comme il n'est pas nécessaire de stocker le dictionnaire de vocabulaire dans la mémoire, pour les grands ensembles de données, il est très peu extensible en mémoire.

Cependant, il n'y a pas de transformation inverse (du vecteur au texte), il peut y avoir des collisions et il n'y a pas de pondération inverse de la fréquence des documents.





# Annexe – NLP – FE - Word2Vec



Il s'agit d'un ensemble de **modèles de réseaux de neurones** développés par google qui ont pour but de représenter les mots dans l'espace vectoriel.

Ces modèles sont très efficaces et performants pour comprendre le contexte et la relation entre les mots.

Les mots similaires sont placés près les uns des autres dans l'espace vectoriel, tandis que les mots dissemblables sont placés loin les uns des autres.

Il existe deux modèles dans cette classe (un réseau de neurones à 3 couches : 1 couche d'entrée, 1 couche cachée, 1 couche de sortie) :

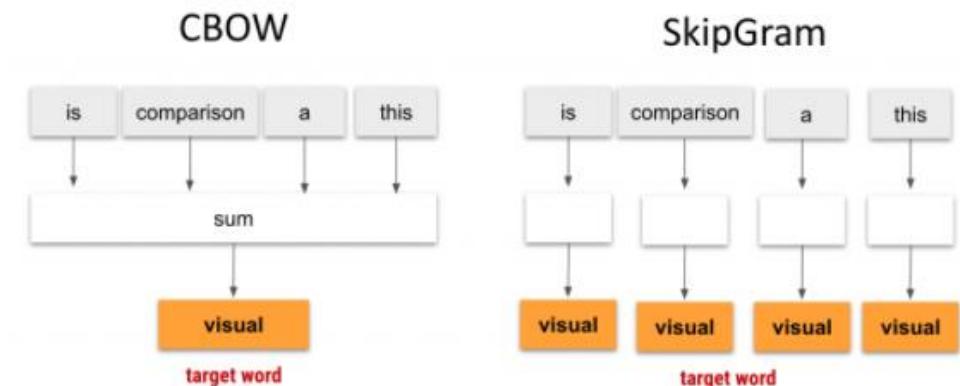
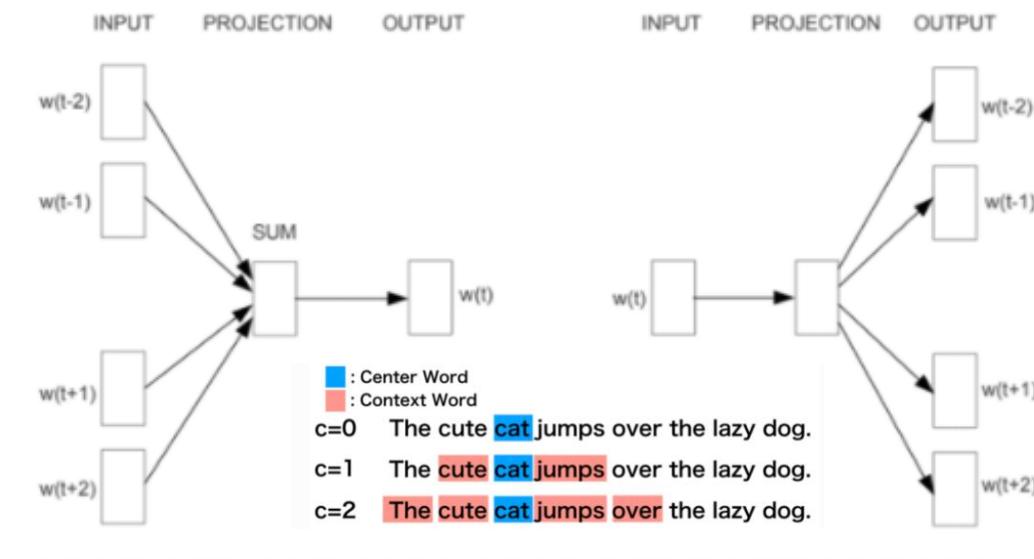
**CBOW (Continuous Bag of Words)** : Le réseau neuronal examine les mots environnants (disons 2 à gauche et 2 à droite) et prédit le mot qui se trouve entre les deux.

- 1. La couche Embedding va transformer chaque mot du contexte en vecteur d'embedding. La matrice W de l'embedding sera apprise au fur et à mesure que le modèle s'entraîne. Les dimensions résultantes sont : (lot, context\_size, embedding).
- 2. Ensuite, la couche GlobalAveragePooling1D permet de sommer les différents embedding pour avoir une dimension en sortie (batch\_size, embedding).
- 3. Enfin, La couche Dense de taille « voc\_size » permet de prédire le mot cible.

**Skip-grams** : Le réseau neuronal prend un mot et essaie ensuite de prédire les mots environnants.

**CBOW** : Le modèle est nourri par le contexte, et prédit le mot cible. Le résultat de la couche cachée est la nouvelle représentation du mot ( $h_1, \dots, h_N$ ).

**Skip Gram** : Le modèle est nourri par le mot cible, et prédit les mots du contexte. Le résultat de la couche cachée est la nouvelle représentation du mot ( $h_1, \dots, h_N$ ).



By: Kavita Ganeshan

This is a visual comparison

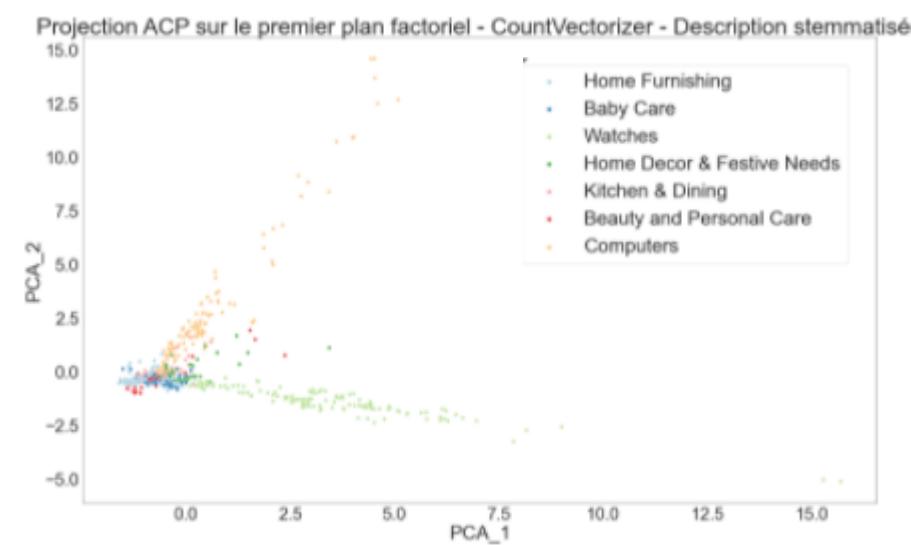
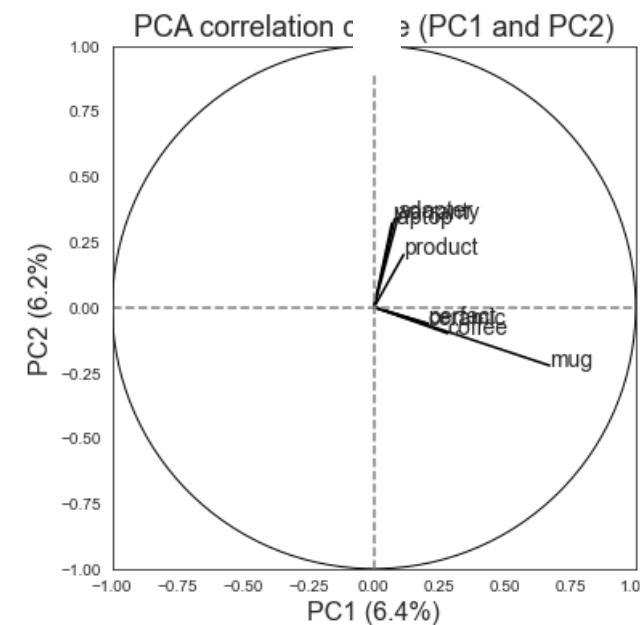
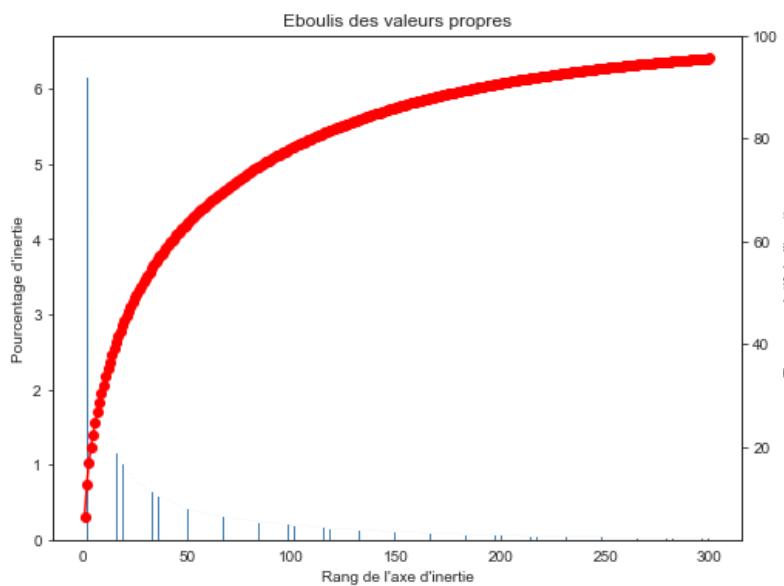
L'ACP est une technique permettant de **réduire le nombre de dimensions** d'un ensemble de données tout en conservant la plupart des informations.

Elle utilise la corrélation entre certaines dimensions et tente de fournir un nombre minimum de variables qui conserve le maximum de variation ou d'informations sur la façon dont les données originales sont distribuées.

Pour ce faire, il ne s'agit pas de deviner, mais de faire appel à des mathématiques dures et d'utiliser ce que l'on appelle les valeurs propres et les vecteurs propres de la matrice de données.

Ces vecteurs propres de la matrice de covariance ont la propriété de pointer dans les principales directions de variation des données. Ce sont les directions de variatio

naximale dans un ensemble de données.



# Annexe – Réduction – t-SNE

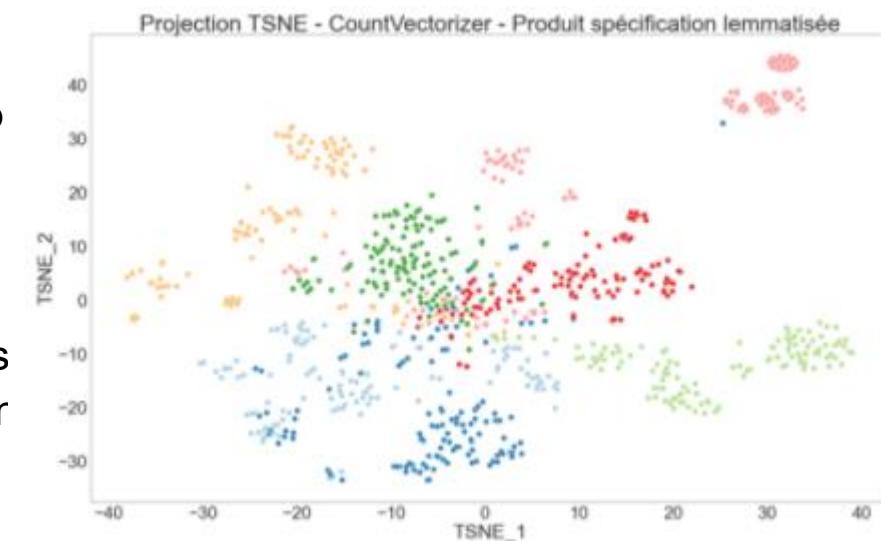
L'algorithme de réduction de dimensionnalité appelé **t-distributed stochastic neighbor embedding (t-SNE)** est un algorithme d'apprentissage non supervisé.

Développé par Laurens van der Maaten et Geoffrey Hinton, il permet d'analyser des données décrites dans des espaces à forte dimensionnalité (via un grand nombre de descripteurs) pour les représenter dans des espaces à deux ou trois dimensions.

Cet algorithme est très utilisé car il facilite la visualisation de données ayant beaucoup de descripteurs.

t-SNE est un **algorithme non-linéaire de “feature extraction”** qui construit une nouvelle représentation des données de telle sorte que les données proches dans l'espace original aient une probabilité élevée d'avoir des représentations proches dans le nouvel espace. A l'inverse, les données qui sont éloignées dans l'espace original, ont une probabilité faible d'avoir des représentations proches dans le nouvel espace.

En pratique la similarité entre chaque paire de données, dans les deux espaces, est mesurée par le biais de calculs probabilistes basés sur des hypothèses de distribution. Et les nouvelles représentations se construisent de telle sorte à minimiser<sup>1</sup> la différence<sup>2</sup> entre les distributions de probabilités mesurées dans l'espace original et celles du nouvel espace.



# Annexe – Réduction – TruncatedSVD

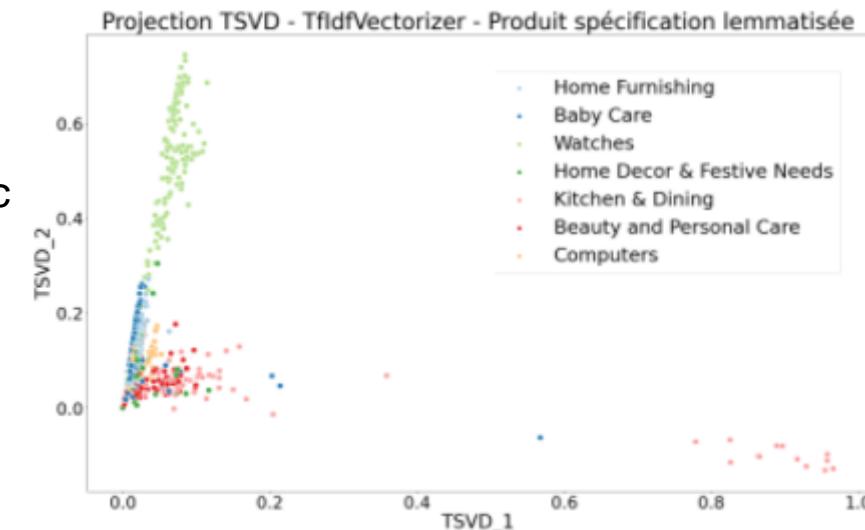
Réduction de la dimensionnalité au moyen de la SVD tronquée.

Ce transformateur effectue une réduction linéaire de la dimensionnalité au moyen de la décomposition de la valeur singulière (SVD) tronquée.

Contrairement à l'ACP, cet estimateur ne centre pas les données avant de calculer la décomposition en valeurs singulières. Cela signifie qu'il peut travailler efficacement avec des matrices éparses.

$$\underline{\text{SVD}}: M_{Q \times N}^{\text{LIM}} = T_{Q \times n} \times S_{n \times n} \times (D_{N \times n})^T$$

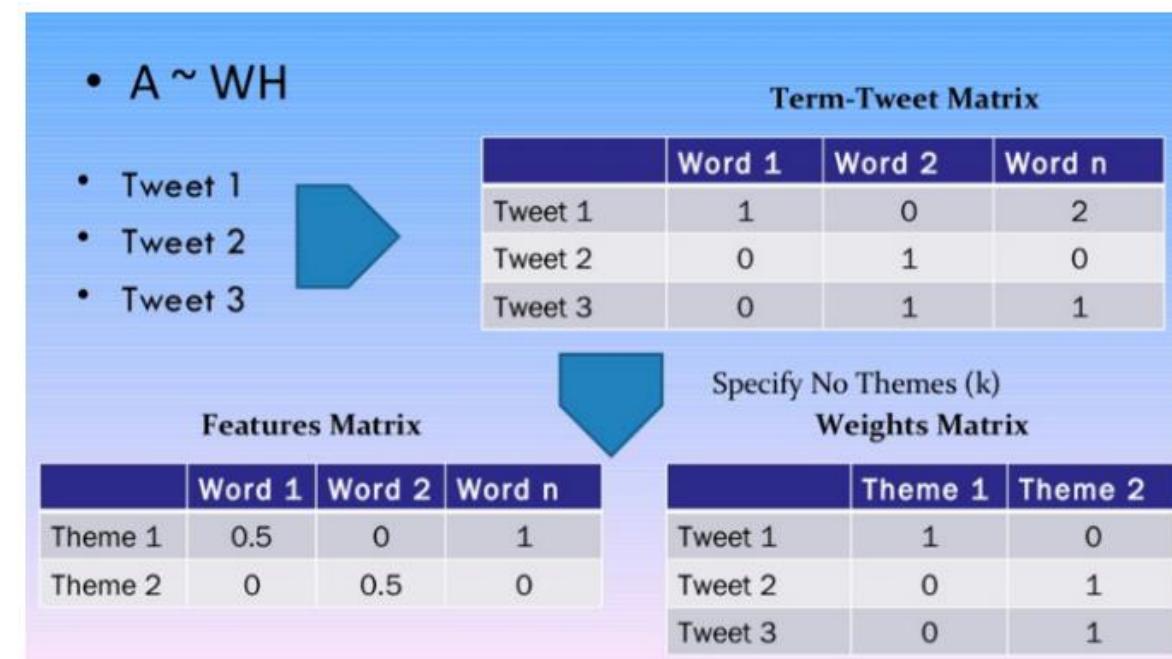
$$\underline{\text{Truncated SVD}}: \hat{M}_{Q \times N}^{\text{LIM}} = T_{Q \times \rho} \times S_{\rho \times \rho} \times (D_{N \times \rho})^T$$



LDA - Latent Dirichlet Allocation : est un algorithme d'apprentissage non supervisé utilisé pour découvrir les thèmes présents dans un corpus.

LDA est basé sur la modélisation graphique probabiliste.

L'algorithme prend en entrée une matrice de sac de mots (c'est-à-dire que chaque document est représenté par une ligne, chaque colonne contenant le nombre de mots dans le corpus). L'objectif est ensuite de produire deux matrices plus petites, une matrice document/sujet et une matrice mot/sujet qui, une fois multipliées ensemble, reproduisent la matrice du sac de mots avec l'erreur la plus faible.



Algorithme :

On fixe un nombre K de thèmes et on cherche à apprendre les thèmes représentés dans chaque document et les mots associés à ces thèmes.

*Initialisation :*

On attribue un thème à chaque mot de chaque document, selon une distribution de Dirichlet sur un ensemble de K thèmes. Ceci génère un premier « modèle de sujet » : des thèmes présents dans les documents et les mots définissant les thèmes. Ce modèle de sujet est très peu vraisemblable car généré aléatoirement.

*Apprentissage :*

On cherche à améliorer le modèle de sujet généré aléatoirement en initialisation. Pour cela, dans chaque document, on prend chaque mot et on met à jour le thème auquel il est lié. Ce nouveau thème est celui qui aurait la plus forte probabilité de le générer dans ce document. On fait donc l'hypothèse que tous les thèmes sont corrects, sauf pour le mot en question.

Plus précisément : pour chaque mot (w) de chaque document (d), on calcule deux quantités pour chaque thème (t) :

$p(t|d)$  : la probabilité que le document d soit assigné au thème t

$p(w|t)$  : la probabilité que le thème t dans le corpus soit assigné au mot w.

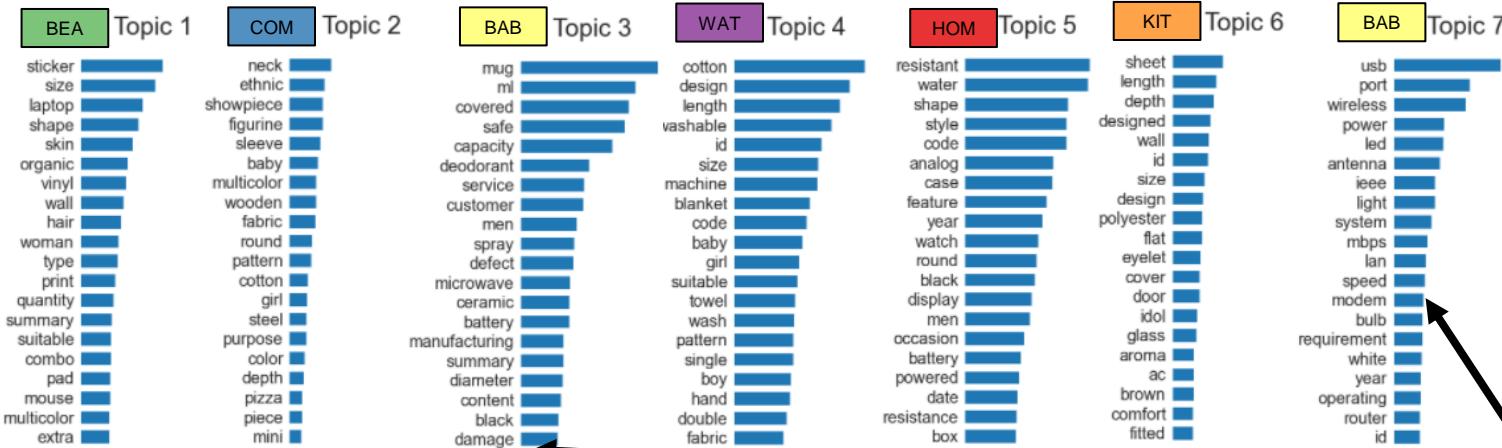
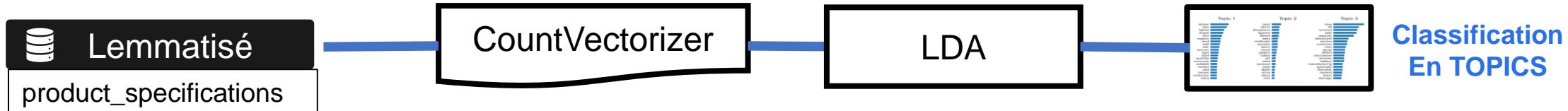
On choisit alors le nouveau thème t avec la probabilité  $p(t|d) \times p(w|t)$ .

Ceci correspond à la probabilité que le thème t génère le mot w dans le document d.

En répétant les étapes précédentes un grand nombre de fois, les assignations se stabilisent. On obtient le mélange de thème présent dans chaque document en comptant chaque représentation d'un thème (assigné aux mots du document).

On obtient les mots associés à chaque thème en comptant les mots qui y sont associés dans le corpus.

# NLP – Classification LDA



ARI = 0,50  
Accuracy = 68%

LDA_CV_PROD_STEM								
	BEA	BAB	Baby Care	107	2	10	8	2
BEA	Beauty and Personal Care			3	94	0	9	0
COM	Computers			0	2	125	12	0
DEC	Home Decor & Festive Needs			0	0	0	0	0
HOM	Home Furnishing			35	12	15	117	147
KIT	Kitchen & Dining			5	38	0	0	0
WAT	Watches			0	2	0	4	1
				0	1	2	3	4
				5	6	7	8	9
	BEA	COM	BAB	WAT	HOM	KIT	BAB	



Seules 6 catégories sur 7 sont trouvées.  
'Home Decor & Festive Needs' manque.



55



# Annexe - NMF

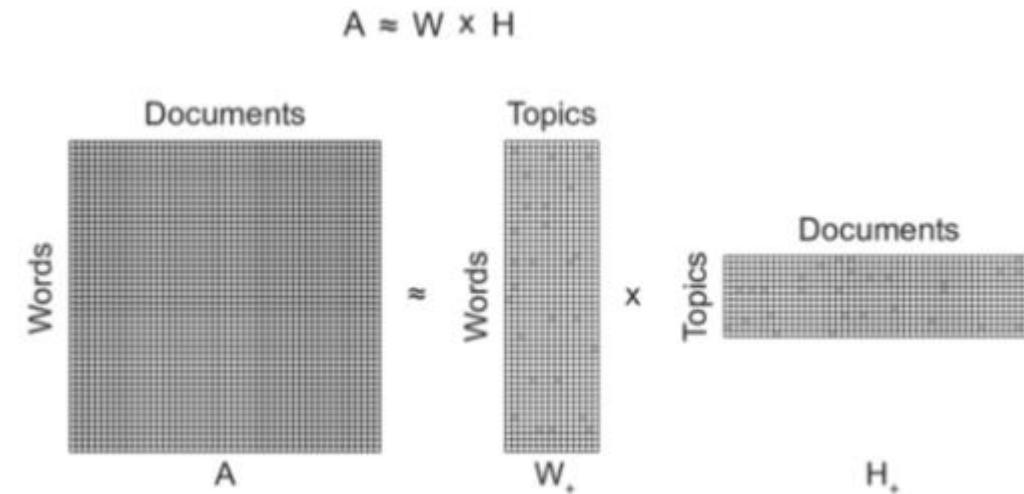


NMF - Non-negative Matrix Factorization :

Famille d'algorithmes d'algèbre linéaire permettant d'identifier la structure latente dans des données représentées par une matrice non négative.

La NMF peut être appliquée à la modélisation de sujets, où l'entrée est une matrice terme-document, généralement normalisée par TF-IDF.

- Entrée : Matrice terme-document, nombre de sujets.
- Sortie : Deux matrices non négatives des  $n$  mots originaux par  $k$  sujets et ces mêmes  $k$  sujets par les  $m$  documents originaux.



La conception de doc2vec (2014) est basée sur word2vec et utilise une approche d'apprentissage non supervisé pour apprendre la représentation du document.

Le nombre de textes (c'est-à-dire de mots) par document peut varier, tandis que la sortie est constituée de vecteurs de longueur fixe.

Les vecteurs de paragraphes et de mots sont initialisés. Le vecteur paragraphe est unique pour tous les documents, tandis que les vecteurs mots sont partagés entre tous les documents, de sorte que le vecteur mot peut être appris à partir de différents documents.

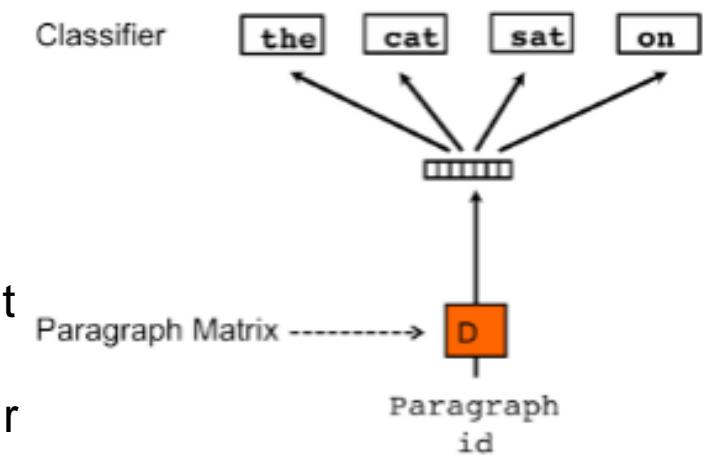
Pendant la phase de formation, les vecteurs de mots seront formés tandis que les vecteurs de paragraphes seront ensuite éliminés.

Pendant la phase de prédiction (c'est-à-dire la prédiction en ligne), le vecteur paragraphe sera initialisé de manière aléatoire et calculé par les vecteurs mots.

#### Algorithme : PV-DBOW (Distributed Bag of Words version of Paragraph Vector)

Au lieu de prédire le mot suivant, il utilise un vecteur de paragraphe pour classer tous les mots du document. Pendant l'apprentissage, une liste de mots est échantillonnée, puis un classificateur est formé pour déterminer si le mot appartient au document, de sorte que les vecteurs de mots puissent être appris. Ce modèle ne tient pas compte des mots contextuels en entrée, mais force le modèle à prévoir des mots échantillonés arbitrairement dans le paragraphe en sortie.

Cette approche est similaire à l'approche skip-gram de word2vec.



Architecture of PV-DBOW (Mikolov et al., 2014)

Le modèle 'universal sentence encoder model' (2018 Google) encode les données textuelles (phrases, paragraphes courts, expressions) dans des vecteurs de haute dimension appelés embeddings qui sont des représentations numériques des données textuelles.

Le modèle Transformer est capable de prendre un mot ou une phrase en entrée et de générer des embeddings pour celui-ci.

Le flux de base est le suivant :

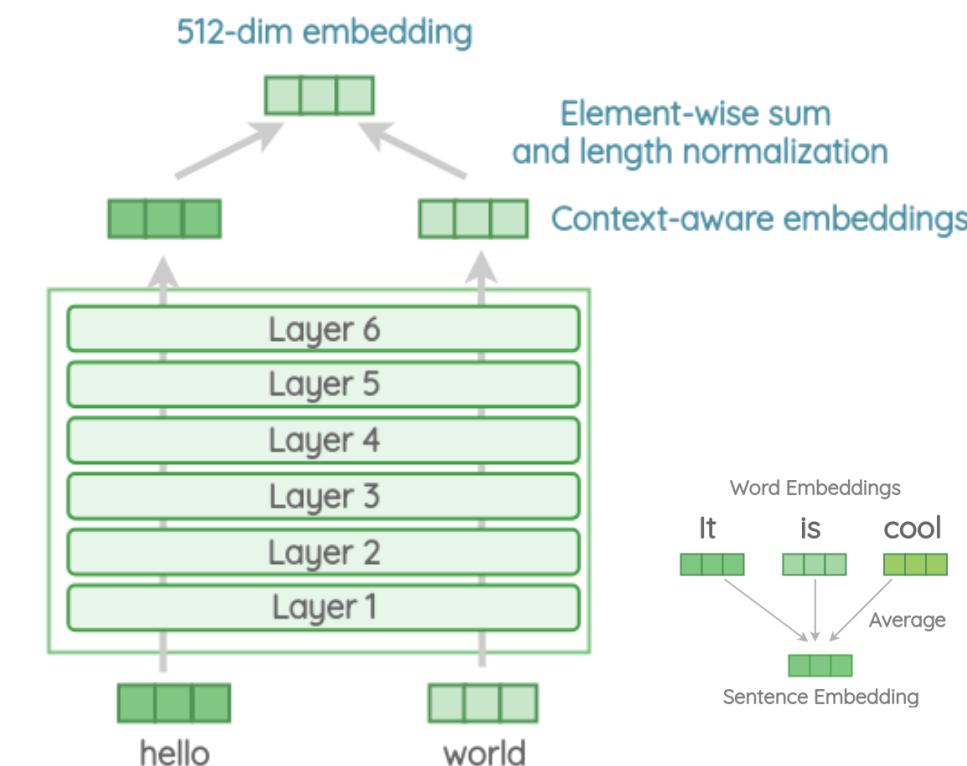
Tokeniser les phrases après les avoir converties en minuscules.

Selon le type d'encodeur, la phrase est convertie en un vecteur à 512 dimensions.

Universal Sentence Encoder - Large et Universal Sentence Encoder – Lite :

Sont des modèles Tensorflow pré-entraînés sur des textes en anglais qui renvoient un encodage sémantique pour des entrées de texte de longueur variable.

Les encodages peuvent être utilisés pour mesurer la similarité sémantique, la parenté, la classification ou le regroupement de textes en langage naturel.





# Annexe - NLP – Roberta



RoBERTa est un modèle de transformateur pré-entraîné sur un large corpus de données anglaises de manière auto-supervisée. Cela signifie qu'il a été pré-entraîné sur les textes bruts uniquement, sans aucun étiquetage humain (c'est pourquoi il peut utiliser de nombreuses données publiques) avec un processus automatique pour générer des entrées et des étiquettes à partir de ces textes.

Plus précisément, il a été pré-entraîné avec l'objectif de modélisation du langage masqué (MLM). À partir d'une phrase, le modèle masque de manière aléatoire 15 % des mots en entrée, puis fait passer l'ensemble de la phrase masquée dans le modèle et doit prédire les mots masqués. Cette méthode est différente des réseaux neuronaux récurrents traditionnels (RNN) qui voient généralement les mots les uns après les autres, ou des modèles autorégressifs comme le GPT qui masquent en interne les futurs tokens. Il permet au modèle d'apprendre une représentation bidirectionnelle de la phrase.

Introduite chez Facebook, Robustly optimized BERT, RoBERTa Approche, est une nouvelle formation de BERT avec une méthodologie de formation améliorée, 1000% de données et de puissance de calcul en plus.

Pour améliorer la procédure de formation, RoBERTa supprime la tâche de prédiction de la phrase suivante (NSP) de la préformation de BERT et introduit un masquage dynamique de sorte que le token masqué change pendant les époques de formation. Des lots de formation plus grands se sont également avérés plus utiles dans la procédure de formation.

Il est important de noter que RoBERTa utilise 160 Go de texte pour le pré-entraînement, dont 16 Go de Books Corpus et de Wikipedia en anglais utilisés dans BERT. Les données supplémentaires comprennent le jeu de données CommonCrawl News (63 millions d'articles, 76 Go), le corpus de textes Web (38 Go) et les Stories from Common Crawl (31 Go). Ces données, couplées à l'utilisation de 1024 GPU V100 Tesla pendant une journée, ont permis le pré-entraînement de RoBERTa.

SentenceBERT a été introduit en 2018 et a immédiatement pris la pole position pour les Sentence Embeddings. Au cœur de ce modèle basé sur BERT, il y a 4 concepts clés :

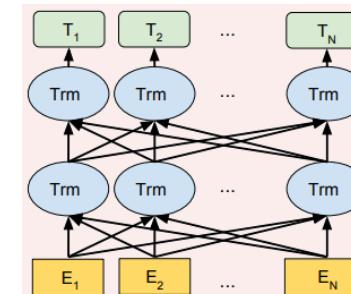
Attention

Transformateurs

BERT (**Bidirectional Encoder Representations from Transformers**,

Transformateur **bidirectionnel** pré-entraîné : chaque mot est contextualisé en utilisant les mots à sa gauche et à sa droite.)

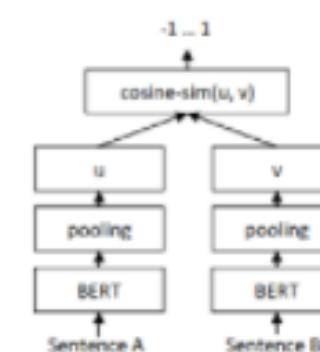
Réseau siamois



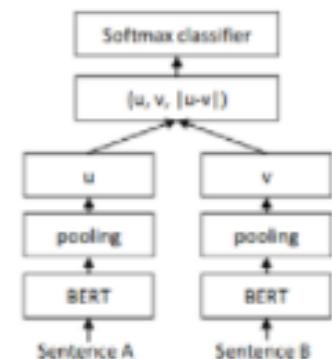
Sentence-BERT utilise une architecture de type réseau siamois pour fournir deux phrases en entrée.

Ces deux phrases sont ensuite transmises aux modèles de BERT et à une couche de mise en commun pour générer leurs encastrements.

Ensuite, les embeddings de la paire de phrases sont utilisés comme entrées pour calculer la similarité en cosinus.



Sentence-BERT for sentence similarity(Regression Task)

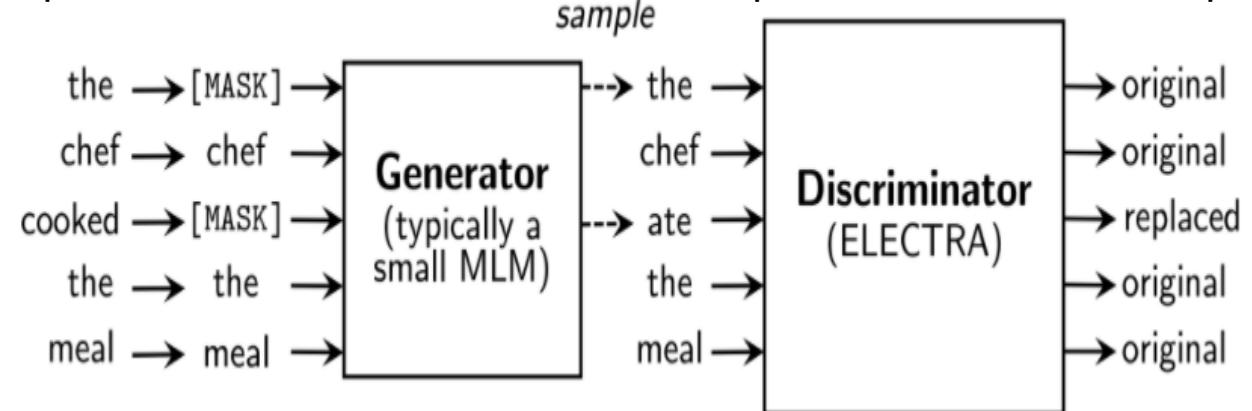


Sentence-BERT for Classification task

ELECTRA est une nouvelle approche de pré-entraînement qui entraîne deux modèles transformateurs : le générateur et le discriminateur.

Le rôle du générateur est de remplacer les tokens dans une séquence, et il est donc entraîné comme un modèle de langage masqué.

Le discriminateur, qui est le modèle qui nous intéresse, essaie d'identifier quels tokens ont été remplacés par le générateur dans la séquence.



Processus :

Pour une séquence d'entrée donnée, on remplace aléatoirement certains tokens par un token [MASK].

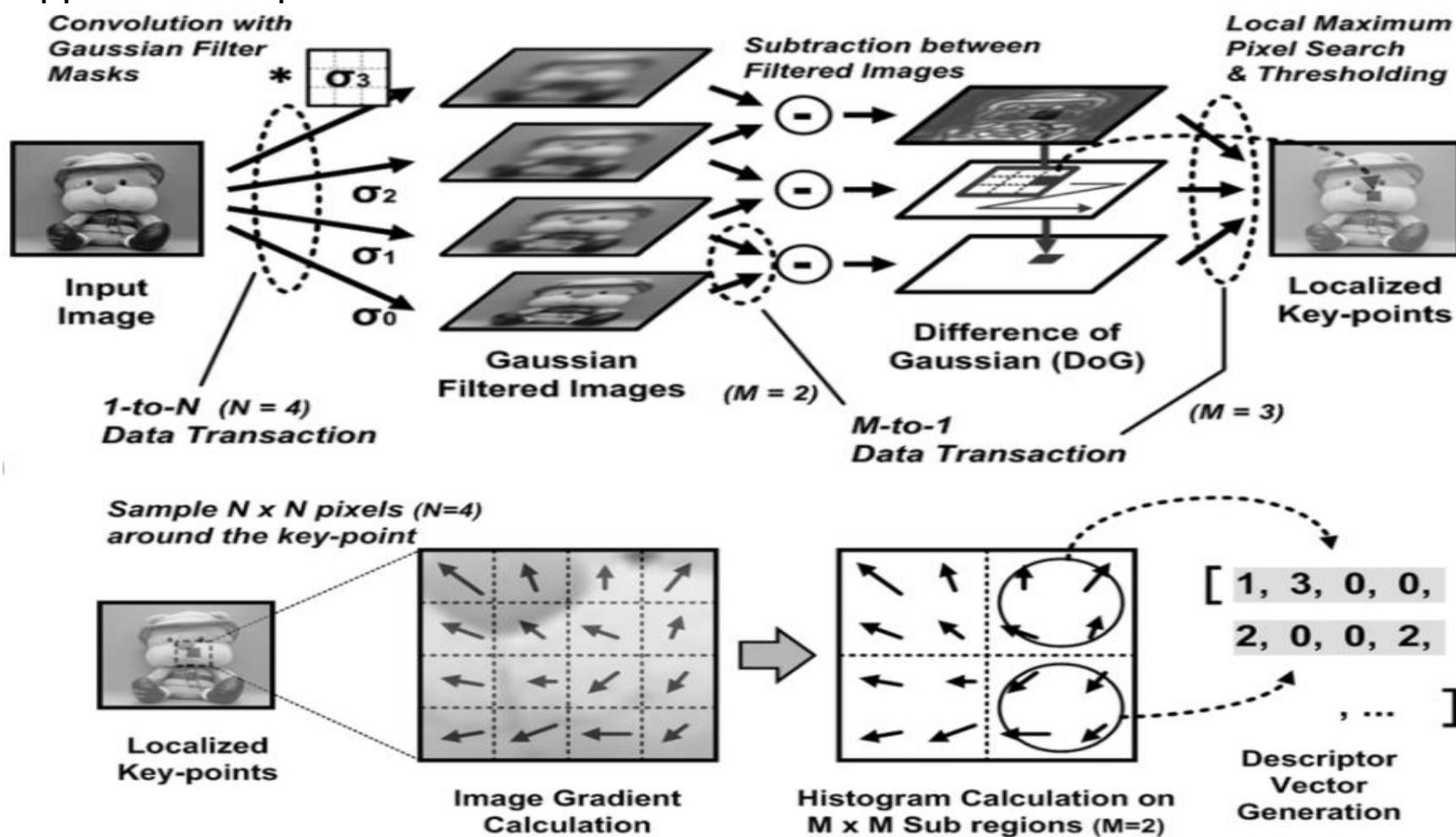
Le générateur prédit les tokens originaux pour tous les tokens masqués.

La séquence d'entrée du discriminateur est construite en remplaçant les jetons [MASK] par les prédictions du générateur.

Pour chaque jeton de la séquence, le discriminateur prédit s'il s'agit d'un original ou s'il a été remplacé par le générateur.

Le modèle du générateur est entraîné à prédire les jetons originaux pour les jetons masqués, tandis que le modèle du discriminateur est entraîné à prédire quels jetons ont été remplacés dans une séquence corrompue. Cela signifie que la perte du discriminateur peut être calculée sur tous les jetons d'entrée car il effectue une prédiction sur chaque jeton. Avec MLM, la perte du modèle n'est calculée que sur les tokens masqués. Il s'agit d'une différence clé entre les deux approches et la raison principale de la plus grande efficacité d'ELECTRA.

La scale-invariant feature transform (SIFT), que l'on peut traduire par « transformation de caractéristiques visuelles invariante à l'échelle », est un algorithme utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques (éléments de paysages, objets, personnes, etc.). Il a été développé en 1999 par le chercheur David Lowe.



## 1. Localisation des points clés

stables, invariants d'échelle et invariants de rotation, partiellement invariants à l'éclairage ou à la luminosité des images

## 2. Génération des descripteurs

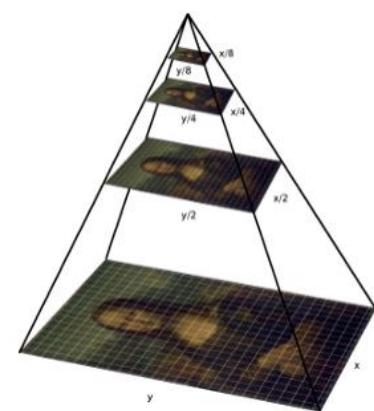
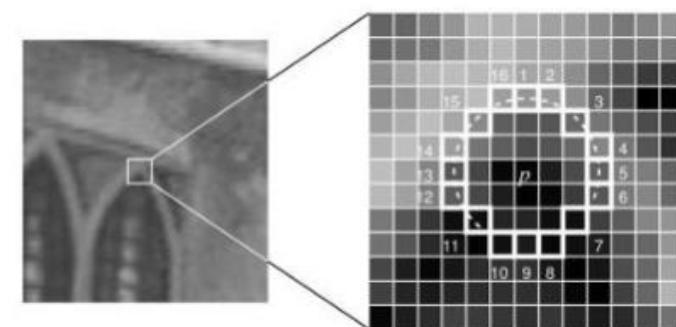
**ORB (Oriented FAST and Rotated BRIEF)** : un détecteur de caractéristiques locales rapide et robuste, présenté pour la première fois par Ethan Rublee et al. en 2011, qui peut être utilisé dans des tâches de vision par ordinateur comme la reconnaissance d'objets ou la reconstruction 3D.

ORB est une fusion du détecteur de points clés FAST et du descripteur BRIEF avec quelques caractéristiques ajoutées pour améliorer les performances.

FAST (Features from Accelerated Segment Test) est utilisé pour détecter les caractéristiques de l'image fournie. Il utilise également une pyramide pour produire des caractéristiques multiscalaires. Mais il ne calcule pas l'orientation et les descripteurs des caractéristiques, c'est là que BRIEF entre en jeu.

ORB utilise les descripteurs BRIEF mais comme le BRIEF est peu performant avec la rotation. Ainsi, ORB fait pivoter le BRIEF en fonction de l'orientation des points clés. En utilisant l'orientation du patch, sa matrice de rotation est trouvée et fait tourner la BRIEF pour obtenir la version tournée.

ORB est une alternative efficace aux algorithmes SIFT ou SURF utilisés pour l'extraction de caractéristiques, en termes de coût de calcul, de performance de correspondance, et surtout de brevets SIFT et SURF sont brevetés et vous êtes censés les payer pour leur utilisation. Mais ORB n'est pas breveté.





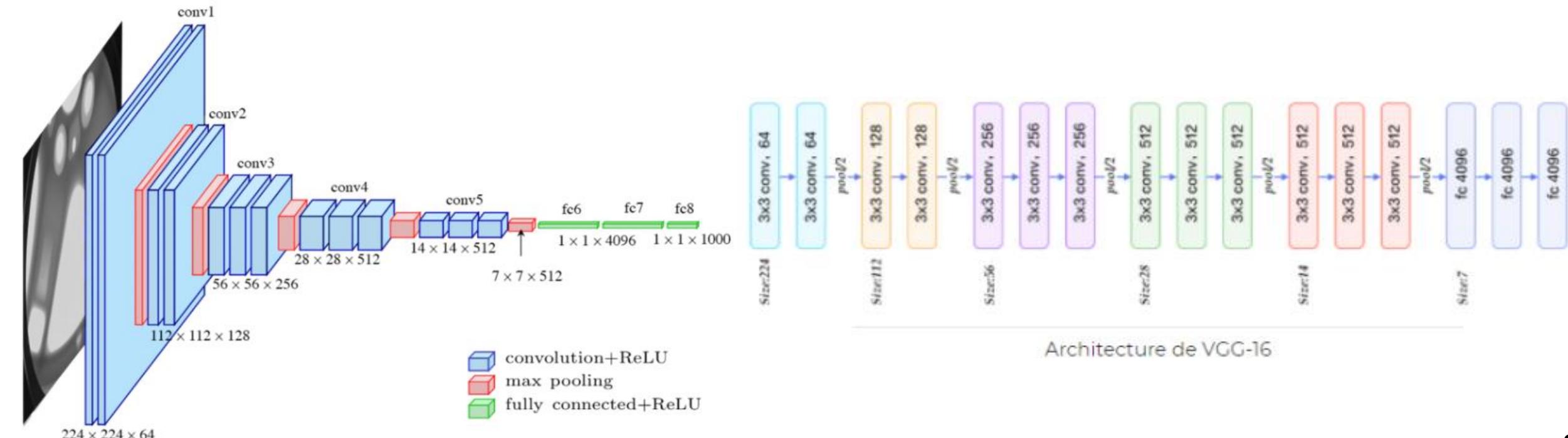
# Annexe – Images – VGG16



VGG-16 est une version du réseau de neurones convolutif VGG-Net (inventé par Simonyan et Zisserman du Visual Geometry Group (VGG) de l'université d'Oxford en 2014).

VGG-16 est constitué de plusieurs couches, dont 13 couches de convolution et 3 fully-connected. Il doit donc apprendre les poids de 16 couches.

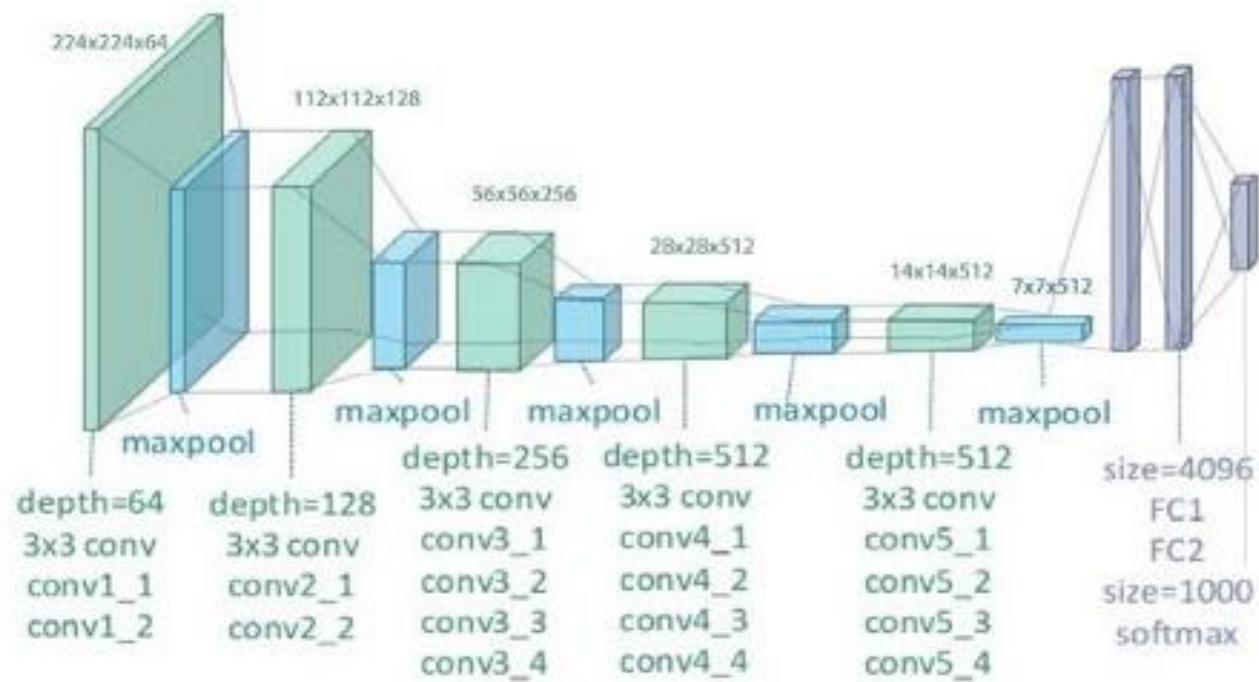
Il prend en entrée une image en couleurs de taille  $224 \times 224$  px et la classifie dans une des 1000 classes. Il renvoie donc un vecteur de taille 1000, qui contient les probabilités d'appartenance à chacune des classes.



Le modèle VGG19 est un réseau neuronal convolutif dont la profondeur est de 19 couches. Il a été construit et entraîné par K. Simonyan et A. Zisserman à l'université d'Oxford en 2014.

Le réseau VGG-19 est entraîné en utilisant plus d'un million d'images de la base de données ImageNet. Il a été entraîné sur des images colorées de 224x224 pixels. Naturellement, vous pouvez importer le modèle avec les poids formés par ImageNet.

Ce réseau pré-entraîné peut classifier jusqu'à 1000 objets.

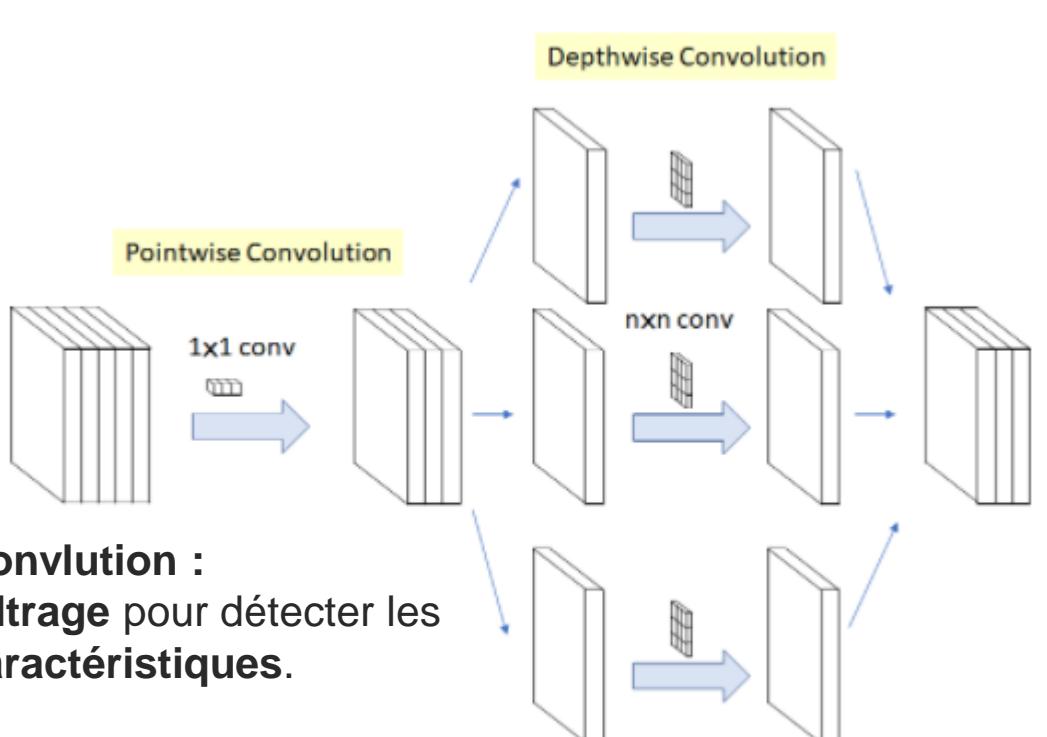


# Annexe – Images –Xception



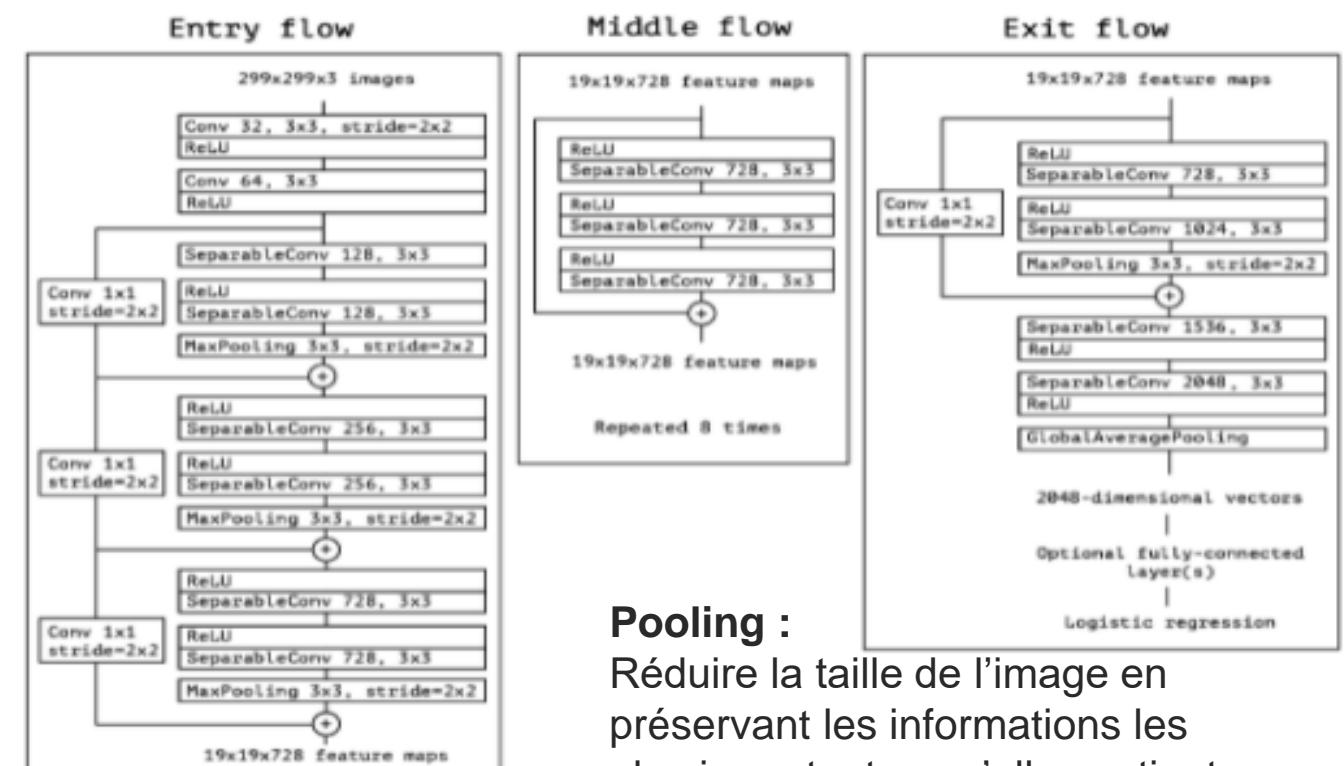
**Xception** créé par Google en 2017 signifie 'version extrême d'Inception' et constitue une modification de Inception.

Avec une convolution séparable en profondeur modifiée, elle est même meilleure qu'Inception-v3 (également par Google, 1er prix à l'ILSVRC 2015) pour les jeux de données ImageNet ILSVRC et JFT.



**Convolution :**  
**Filtrage** pour détecter les caractéristiques.

The Modified Depthwise Separable Convolution used as an Inception Module in Xception, so called "extreme" version of Inception module ( $n=3$  here)



**Pooling :**  
Réduire la taille de l'image en préservant les informations les plus importantes qu'elle contient.

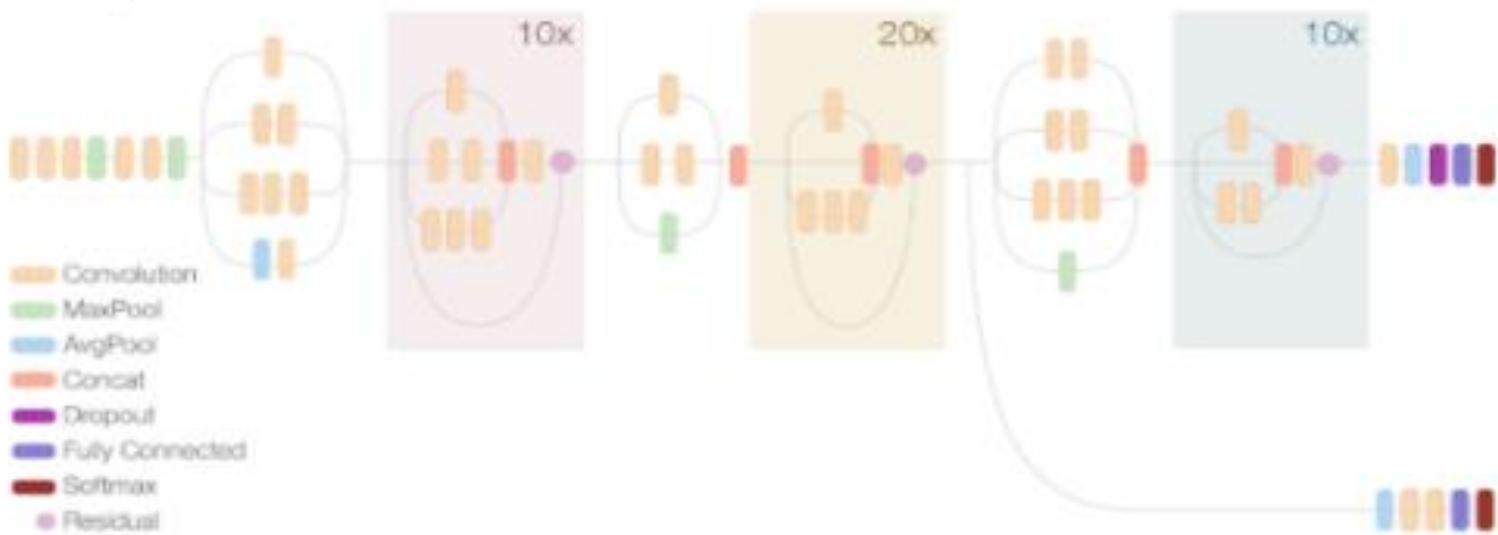
Overall Architecture of Xception (Entry Flow > Middle Flow > Exit Flow)

# Annexe – Images – InceptionResNetV2

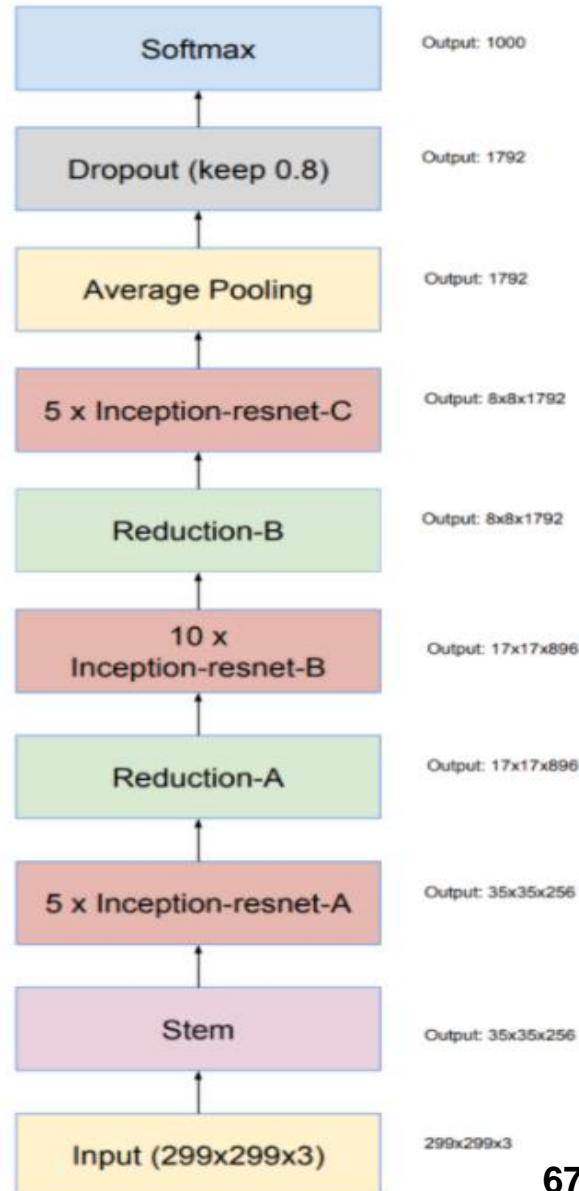


**Inception-ResNet-v2** est un réseau de neurones convolutif entraîné sur plus d'un million d'images de la base de données ImageNet (publié par Microsoft Research et Facebook AI research).

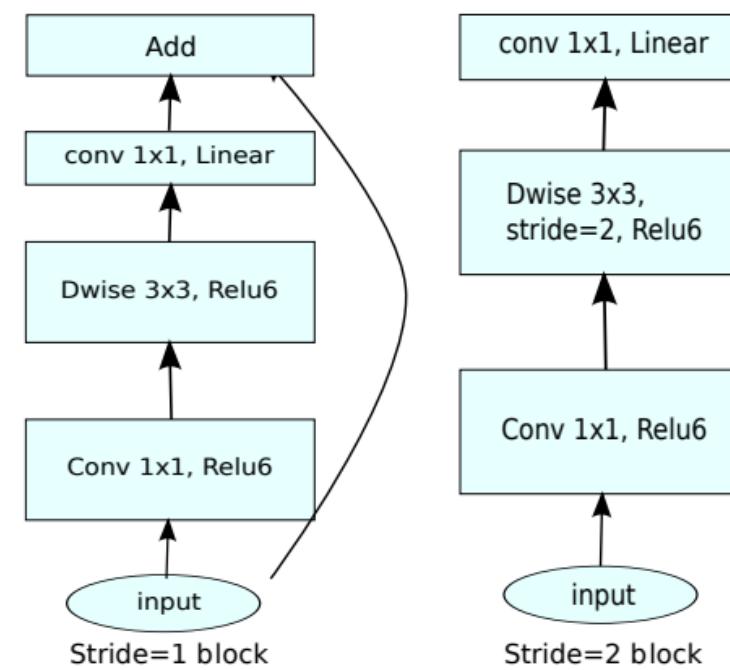
Le réseau a une profondeur de 164 couches et peut classer les images dans 1000 catégories d'objets, comme le clavier, la souris, le crayon et de nombreux animaux. Par conséquent, le réseau a appris des représentations de caractéristiques riches pour un large éventail d'images. Le réseau a une taille d'entrée d'image de 299 par 299.



Schematic diagram of Inception-ResNet-v2



**MobileNetV2** (Google 2017) est une architecture de réseau neuronal convolutif qui cherche à être performante sur les appareils mobiles. Il est basé sur une structure résiduelle inversée où les connexions résiduelles se trouvent entre les couches d'étranglement. La couche d'expansion intermédiaire utilise des convolutions légères en profondeur pour filtrer les caractéristiques comme source de non-linéarité. Dans l'ensemble, l'architecture de MobileNetV2 contient la couche initiale à convolution complète avec 32 filtres, suivie de 19 couches résiduelles à goulot d'étranglement.

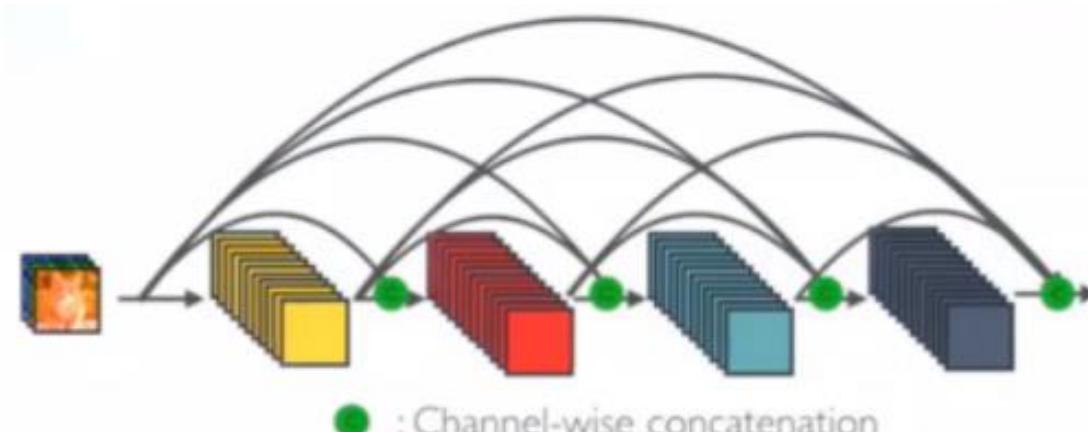


(d) Mobilenet V2

DenseNet (Densely Connected Convolutional Networks) est l'une des nouvelles découvertes en matière de réseaux neuronaux pour la reconnaissance visuelle des objets.

DenseNet est assez similaire à ResNet avec quelques différences fondamentales. ResNet utilise une méthode additive (+) qui fusionne la couche précédente (identité) avec la couche future, alors que DenseNet concatène la sortie de la couche précédente avec la couche future.

Dans DenseNet, chaque couche obtient des entrées supplémentaires de toutes les couches précédentes et transmet ses propres cartes de caractéristiques à toutes les couches suivantes. La concaténation est utilisée. Chaque couche reçoit des "connaissances collectives" de toutes les couches précédentes.

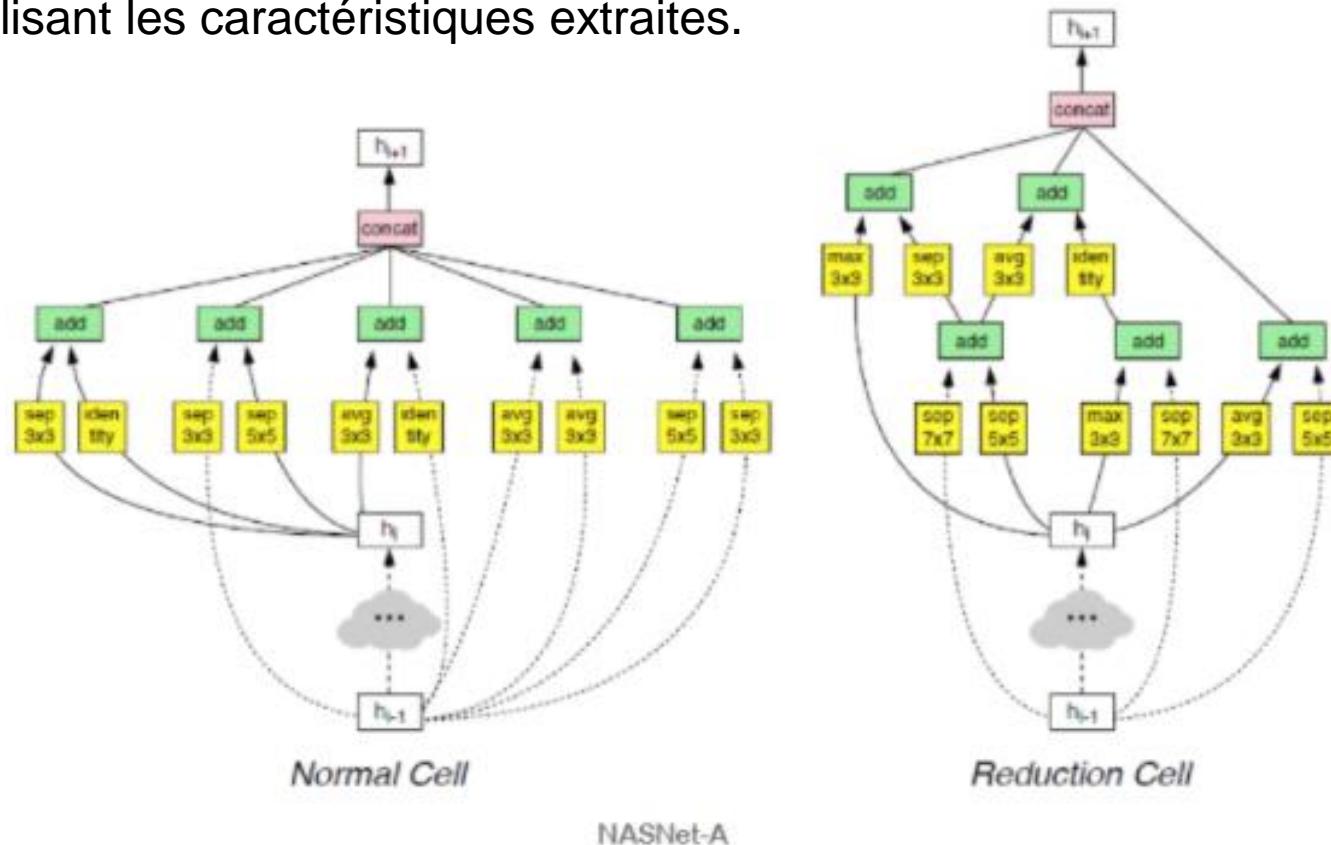


One Dense Block in DenseNet

# Annexe – Images –NaSNetLarge

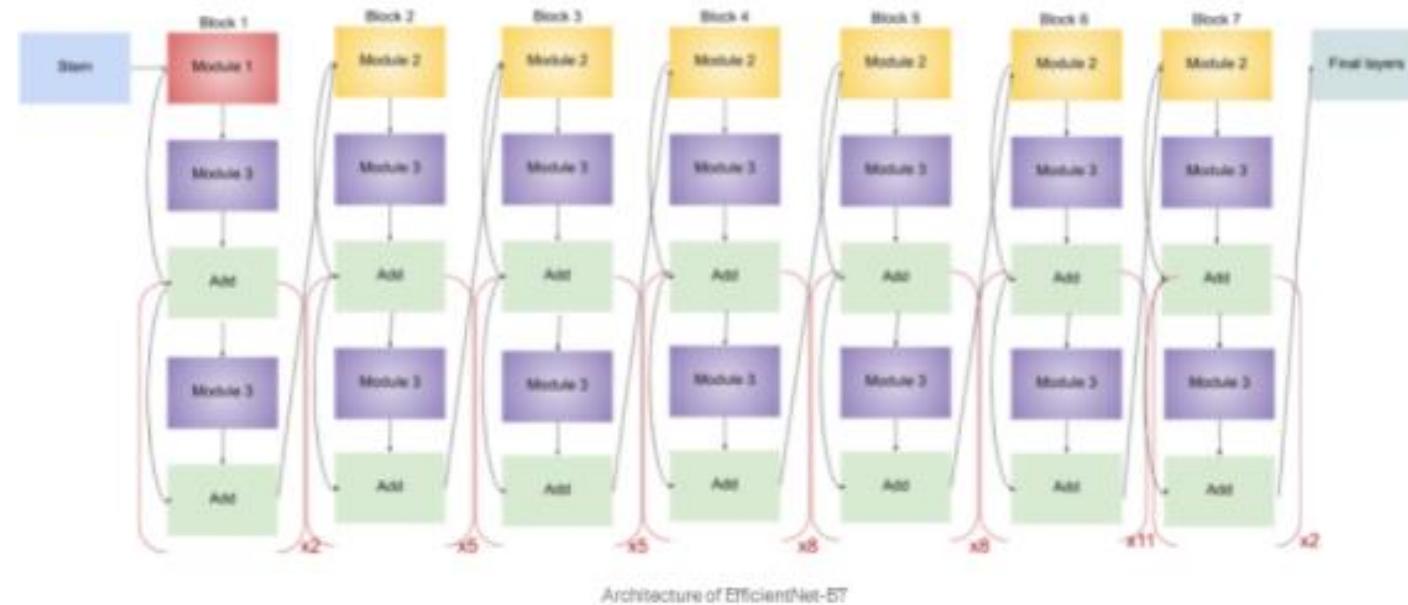
**NASNetLarge** (Neural Architecture Search Network), développé purement par l'environnement d'apprentissage par renforcement de Google (NAS- Neural Architecture Search) sans intervention humaine, est l'une des architectures les plus populaires.

La partie réseau neuronal à convolution de l'architecture est appelée la base, et la partie réseau neuronal artificiel (avec des couches denses) est appelée la tête. La base extrait les caractéristiques des images d'entrée. La tête effectue la classification en utilisant les caractéristiques extraites.



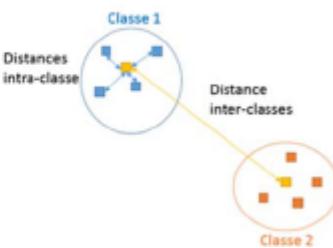
NASNet-A

Les **EfficientNets** sont une famille d'architectures de réseaux neuronaux publiées par Google en 2019 qui ont été conçues par une procédure d'optimisation qui maximise la précision pour un coût de calcul donné.



## Clustering

- Définir une distance, une similarité
- Distance intra-cluster / classe (minimiser)
- Distance inter-cluster / classe (maximiser)



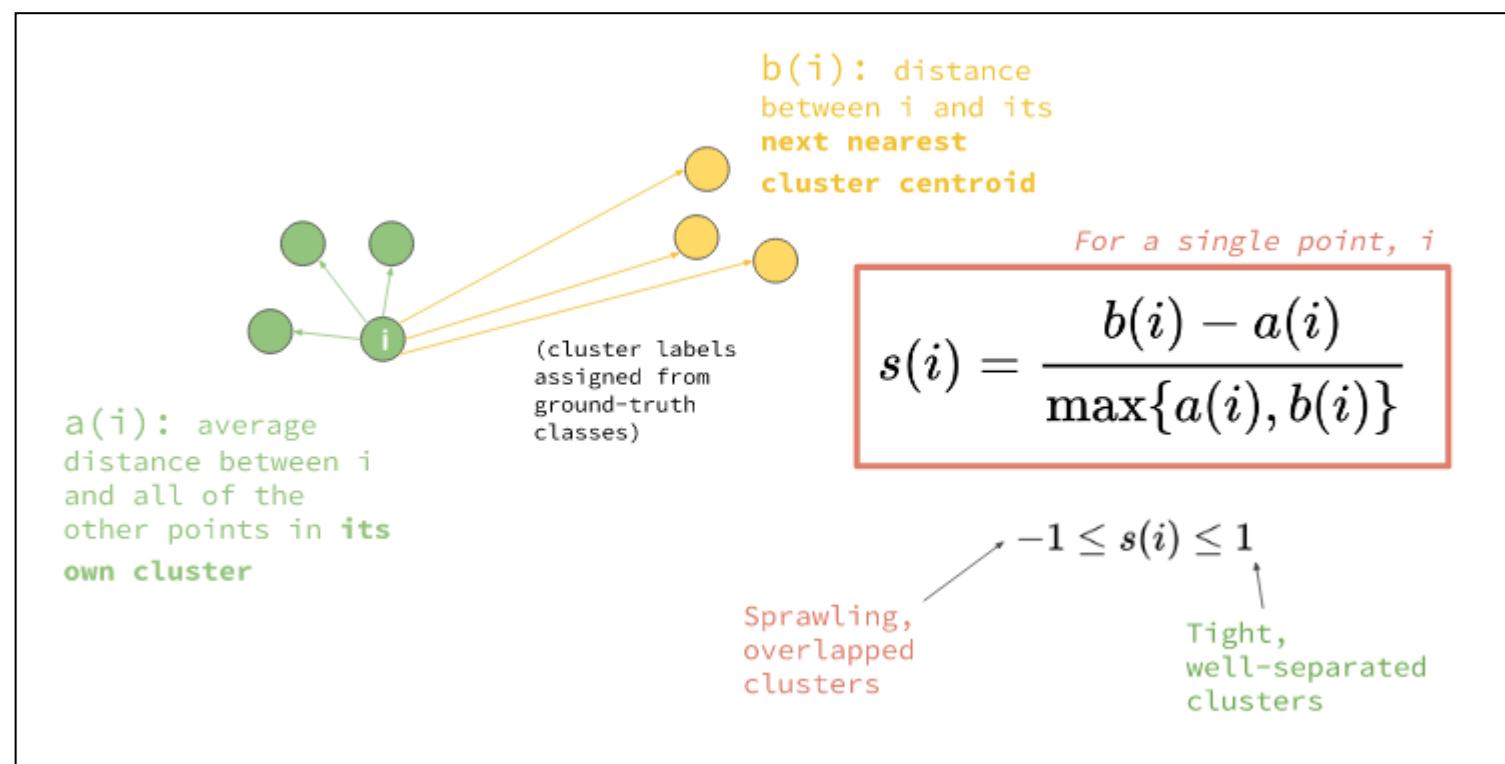
## Silhouette : autre évaluation pour homogénéité et séparation

- Pour chaque point  $x$ : est-ce qu'il appartient au « bon » cluster :
  - Est-il proche des points du cluster auquel il appartient ?
    - distance moyenne aux autres points du même cluster
    - $a(x) = \frac{1}{n_k-1} \sum_{y \in C_k, y \neq x} d(x, y)$
  - Est-il loin des points des autres clusters ?
    - Distance moyenne minimale si  $x$  affecté dans un autre cluster?
    - $b(x) = \min_{l \neq k} \frac{1}{n_l} \sum_{y \in C_l} d(x, y)$
  - Si le point  $x$  est dans le bon cluster :  $a(x) < b(x)$
- Silhouette : combinaison des deux scores :  $s(x) = \frac{b(x)-a(x)}{\max(a(x), b(x))}$ 
  - compris dans  $[-1, 1]$
- Pour tous les points :  $S = \frac{1}{n} \sum s(x)$
- Aide pour déterminer le nombre de clusters  $K$  (minimiser  $S$ )

Au moins 2 points et 2 clusters ...

Silhouette score  
(à maximiser)

C'est la différence entre les distances intra-cluster et les distances au cluster extérieur le plus proche (rapportée à la plus grande des deux)  $\Rightarrow$  entre 0 (pire) et 1 (meilleur)





# Annexe – Clustering score – Dispersion



La moyenne de la somme des carrés des distances au centroïde le plus proche.

Par conséquent, plus l'inertie est faible, plus l'amas est dense (tous les points sont plus proches les uns des autres).

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

- Combinaison des deux critères (homogénéité et séparation)

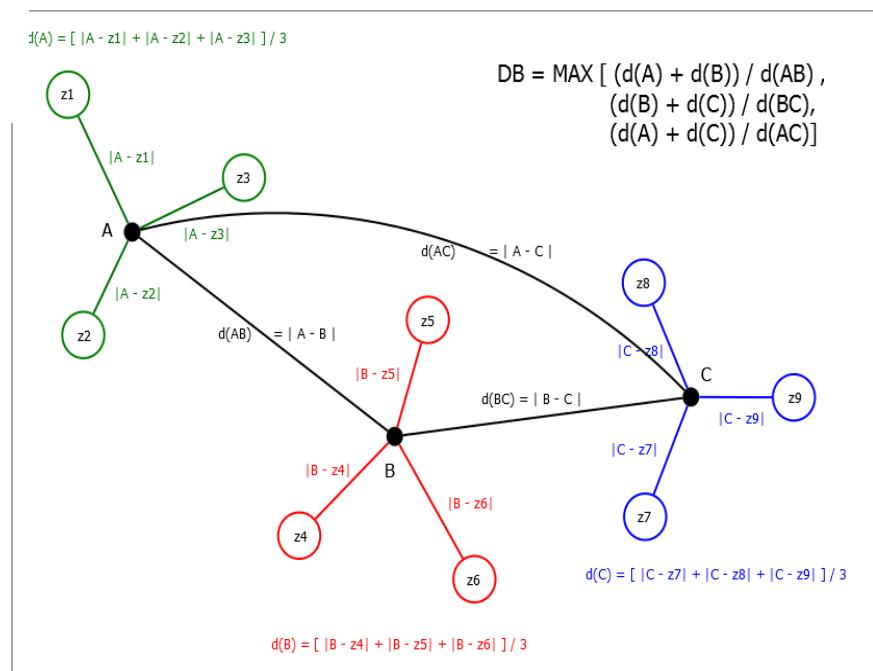
- Indice de Davies Bouldin

- pour un cluster  $k$  :
  - Combiner les valeurs des tightness (intra) et distances entre les barycentres (inter):
  - $DB_k = \max_{l \neq k} \left( \frac{T_k + T_l}{d(C_k, C_l)} \right)$
- Valeur faible si les clusters sont homogènes (numérateur petit) et s'ils sont bien séparés (dénominateur grand)
- pour tous les clusters :  $DB = \frac{1}{K} \sum_{k=1}^K DB_k$

- Aide pour déterminer le nombre de clusters  $K$  (minimiser  $DB$ )

## Davies-Bouldin score (à minimiser)

C'est la moyenne du rapport maximal entre la distance des points à leur centroïde et la distance entre deux centroïdes (deux à deux)  
 $\Rightarrow$  entre 0 (meilleur) et  $+\infty$  (pire)



distance moyennes  
des points à leur centroïde

$$\bar{\delta}_k = \frac{1}{|I_k|} \sum_{i \in I_k} d(x^i, \mu_k)$$

$$S_{DB} = \frac{1}{K} \sum_{k=1}^K \max_{k' \neq k} \left( \frac{\bar{\delta}_k + \bar{\delta}_{k'}}{d(\mu_k, \mu_{k'})} \right)$$

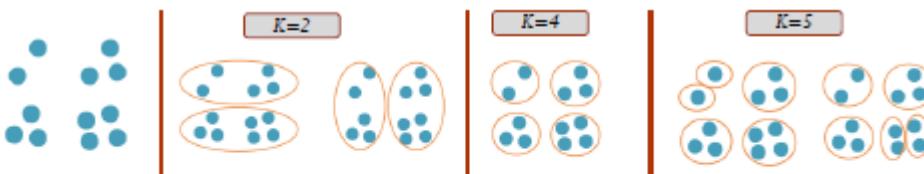
distance entre deux centroïdes

Note: formules de l'illustration avec la distance Manhattan

## Stabilité d'un clustering

- Stabilité

- Nombreux algorithmes non déterministes : résultats différents lors d'exécutions différentes de l'algorithme
  - lancer l'algorithme plusieurs fois sur les mêmes données (éventuellement bruitées), avec initialisation différente, avec des sous-ensembles différents,
- Est-ce que les points sont regroupés de manière similaire ?



- Mesure pour évaluer cette similarité .... (sans dépendre de la numérotation des clusters) ... Voir Indice de Rand



Indice de Rand ajusté pour le hasard.

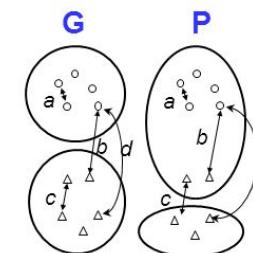
L'indice de Rand calcule une mesure de similarité entre deux clusters en considérant toutes les paires d'échantillons et en comptant les paires qui sont assignées dans les mêmes ou différents clusters dans les clusters prédicts et réels.

Le score brut RI est ensuite "ajusté pour le hasard" dans le score ARI en utilisant le schéma suivant :

$$\text{ARI} = (\text{RI} - \text{Expected\_RI}) / (\text{max(RI)} - \text{Expected\_RI}).$$

L'indice de Rand ajusté est ainsi assuré d'avoir une valeur proche de 0,0 pour un étiquetage aléatoire indépendamment du nombre de clusters et d'échantillons et exactement 1,0 lorsque les clusters sont identiques (jusqu'à une permutation).

## ARI score (à maximiser)



Agreement:  $a, d$   
Disagreement:  $b, c$

$$\text{RI}(P, G) = \frac{a+d}{a+b+c+d}$$

L'*Adjusted Rand Index* (ARI) est la normalisation de l'indice de Rand (RI) qui permet de comparer deux partitions de nombres de classes différentes.

$$\text{ARI} = \frac{\text{RI} - \text{E(RI)}}{\text{max(RI)} - \text{E(RI)}}$$

- RI : indice de Rand, proportion de paires de points qui sont groupés de la même façon dans les deux partitions.
- E(RI) : espérance de l'indice de Rand (pour une partition aléatoire)
- max(RI) : indice de Rand maximal qui pourrait être obtenu étant donné le nombre de classes distincts

(à maximiser)

L'homogénéité mesure à quel point les échantillons d'une grappe sont similaires.

$$h = 1 - \frac{H(C|K)}{H(C)}$$

Elle est définie à l'aide de l'entropie de Shannon.

$$H(C|K) = - \sum_{c,k} \frac{n_{ck}}{N} \log \left( \frac{n_{ck}}{n_k} \right)$$

$n_{ck} / n_k$  représente le rapport entre le nombre d'échantillons étiquetés c dans la grappe k et le nombre total d'échantillons dans la grappe k.

Lorsque tous les échantillons de la grappe k ont la même étiquette c, l'homogénéité est égale à 1.  
Notez qu'il y a une somme sur c et k, peu importe quel est le cluster qui contient une étiquette particulière, il suffit qu'il y en ait au moins un.  
Ceci est utile lors de l'exécution de méthodes non supervisées dont la sortie n'a rien à voir avec les étiquettes.

(à maximiser)

La complétude, bien qu'elle mesure le degré de regroupement des échantillons similaires par l'algorithme de clustering.

$$c = 1 - \frac{H(K|C)}{H(K)}$$

Elle est définie à l'aide de l'entropie de Shannon.  $H(K|C) = - \sum_{k,c} \frac{n_{ck}}{N} \log \left( \frac{n_{kc}}{n_c} \right)$

Si l'on regarde le terme  $H(K|C)$ , il contient  $n_{kc} / n_c$  qui représente le rapport entre le nombre d'échantillons étiquetés c dans le cluster k et le nombre total d'échantillons étiquetés c.

Lorsque tous les échantillons de type c ont été affectés au même cluster k, la complétude est égale à 1.

(à maximiser)

V-measure (ou Normalised Mutual Information).

Ce score peut être interprété comme la moyenne harmonique de deux autres mesures : l'homogénéité (h) et la complétude (c).

$$NMI = 2 * \frac{h * c}{h + c}$$

Ce score est une mesure entre 0 et 1 qui quantifie en fait la qualité de la partition de clustering.

En fait, il exige que l'homogénéité h et la complétude c soient maximisées (NMI est égal à 1 lorsque h et c sont tous deux égaux à 1).

De plus, si le clustering ne remplit aucune de ces deux conditions, la NMI sera nulle.

Il existe de nombreuses mesures de ce type, mais grâce à sa factorisation en deux mesures, l'homogénéité et la complétude, son interprétation est claire et intuitive.