



# Projet 6: Classifiez automatiquement des biens de consommation

---

QUENTIN STEPNIEWSKI

**OPENCLASSROOMS**

06 SEPT 2020

# Sommaire

---

1. Introduction – Présentation de la problématique
2. Présentation des données utilisées
3. Classification par descriptions
  - Pre-processing
  - Modèles
4. Classification par images
  - Pre-processing
  - Modèles
5. Modèle final (Images + Texte)
6. Conclusion sur la faisabilité et mise en perspective

# 1. Introduction – Présentation de la problématique

---

Data Scientist au sein de l'entreprise "Place de marché", qui souhaite lancer un marketplace e-commerce.

## Problématique principale

- Automatiser l'attribution d'une catégorie à un produit proposé par un vendeur sur la plateforme

## Objectifs de l'étude

- Proposer une étude de faisabilité de classification des produits basée sur :
  - ☐ une description du produit
  - ☐ une photo du produit

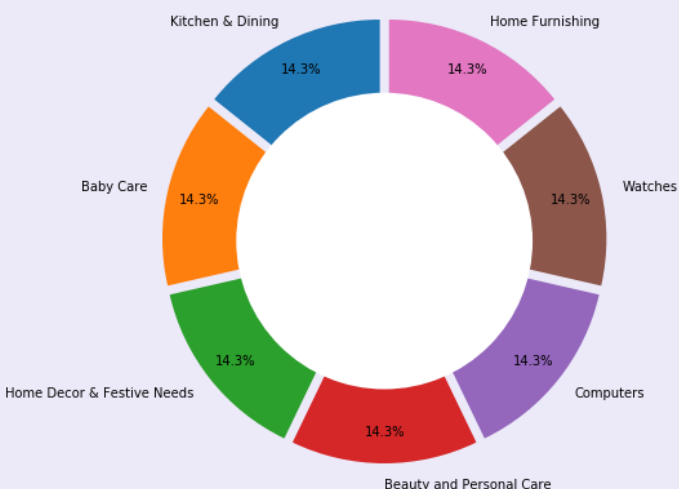



## 2. Présentation des données utilisées

Base de données composée de 1050 articles (15 colonnes par article) :

♦ Id, prix, prix soldé, marque, note...

3 features nous intéressent :

Target	Features
Product Category	Product Description
<p>Categories distribution</p>  <p>Parfaite répartition des classes</p>	<pre>train.iloc[0,1]</pre> <p>'Flippd FD040149 Casual Analog Watch - For Women, Girls - Buy Flippd FD040149 Casual Analog Watch - For Women, Girls FD040149 Online at Rs.999 in India Only at Flipkart.com. - Great Discounts, Only Genuine Products, 30 Day Replacement Guarantee, Free Shipping. Cash On Delivery!'</p> <pre>train.iloc[18,1]</pre> <p>'Buy Gift Studios Buddha Stone Showpiece - 17.6 cm for Rs.699 online. Gift Studios Buddha Stone Showpiece - 17.6 cm at best prices with FREE shipping &amp; cash on delivery. Only Genuine Products. 30 Day Replacement Guarantee.'</p>
Product Image	
	



# 3. Classification par descriptions – Pre-processing

Etapas de pre-processing :

Input : texte initial

A => a

Lower Case

you are  
=> [you, are]

Tokenizer

Suppression des  
mots inutiles :  
« of, the, in, ... »

Stop Words

created => create  
products => product

Lemmatizer  
/Stemming

Output : liste de tokens utilisables

Original Shape

Specifications of Eternity Handcrafted unique Mosaic Glass Table Lamp (35 cm, Blue) In The Box Sales Package 1 Table Lamp Number of Contents in Sales Package Pack of 1 General Type Buffet, Desk Color Blue Lamp Body Material Glass Lamp Base Material Metal Bulb Type 1x15 LED, CFLS ,Bulb Light Color Blue Assembly Required No Model Name Handcrafted unique mosaic glass model number eternity001007 Power Features Power Requirement 110-240V, 50/60Hz Power Source Ac Additional Features Handcrafted lamps Dimensions Width 15 cm Height 35 cm Weight 1000 g

Lower Case

specifications of eternity handcrafted unique mosaic glass table lamp (35 cm, blue) in the box sales package 1 table lamp number of contents in sales package pack of 1 general type buffet, desk color blue lamp body material glass lamp base material metal bulb type 1x15 led, cfls ,bulb light color blue assembly required no model name handcrafted unique mosaic glass model number eternity001007 power features power requirement 110-240v, 50/60hz power source ac additional features handcrafted lamps dimensions width 15 cm height 35 cm weight 1000 g

Tokenizer

```
['specifications', 'of', 'eternity', 'handcrafted', 'unique', 'mosaic', 'glass', 'table', 'lamp', '35', 'cm', 'blue', 'in', 'the', 'box', 'sales', 'of', 'package', '1', 'table', 'lamp', 'number', 'of', 'contents', 'in', 'sales', 'package', 'pack', 'of', '1', 'general', 'type', 'buffet', 'desk', 'color', 'blue', 'lamp', 'body', 'material', 'glass', 'lamp', 'base', 'material', 'metal', 'bulb', 'type', '1x15', 'led', 'cfls', 'bulb', 'light', 'color', 'blue', 'assembly', 'required', 'no', 'model', 'name', 'handcrafted', 'unique', 'mosaic', 'glass', 'model', 'number', 'eternity001007', 'power', 'features', 'power', 'requirement', '110', '240v', '50', '60hz', 'power', 'source', 'ac', 'additional', 'features', 'handcrafted', 'lamps', 'dimensions', 'width', '15', 'cm', 'height', '35', 'cm', 'weight', '1000', 'g']
```

Stop\_Words

```
['specifications', 'eternity', 'handcrafted', 'unique', 'mosaic', 'glass', 'table', 'lamp', '35', 'cm', 'blue', 'box', 'sales', 'package', '1', 'table', 'lamp', 'number', 'contents', 'sales', 'package', 'pack', '1', 'general', 'type', 'buffet', 'desk', 'color', 'blue', 'lamp', 'body', 'material', 'glass', 'lamp', 'base', 'material', 'metal', 'bulb', 'type', '1x15', 'led', 'cfls', 'bulb', 'light', 'color', 'blue', 'assembly', 'required', 'model', 'name', 'handcrafted', 'unique', 'mosaic', 'glass', 'model', 'number', 'eternity001007', 'power', 'features', 'power', 'requirement', '110', '240v', '50', '60hz', 'power', 'source', 'ac', 'additional', 'features', 'handcrafted', 'lamps', 'dimensions', 'width', '15', 'cm', 'height', '35', 'cm', 'weight', '1000', 'g']
```

Lemmatizer

```
['specification', 'eternity', 'handcraft', 'unique', 'mosaic', 'glass', 'table', 'lamp', '35', 'cm', 'blue', 'box', 'sale', 'package', '1', 'table', 'lamp', 'number', 'content', 'sale', 'package', 'pack', '1', 'general', 'type', 'buffet', 'desk', 'color', 'blue', 'lamp', 'body', 'material', 'glass', 'lamp', 'base', 'material', 'metal', 'bulb', 'type', '1x15', 'lead', 'cfls', 'bulb', 'light', 'color', 'blue', 'assembly', 'require', 'model', 'name', 'handcraft', 'unique', 'mosaic', 'glass', 'model', 'number', 'eternity001007', 'power', 'feature', 'power', 'requirement', '110', '240v', '50', '60hz', 'power', 'source', 'ac', 'additional', 'feature', 'handcraft', 'lamp', 'dimension', 'width', '15', 'cm', 'height', '35', 'cm', 'weight', '1000', 'g']
```

### 3. Classification par descriptions – Modèles

Modèles supervisés :

Bag of words et TF-IDF (avec une classification RandomForest)

#### Bag of Words

Chaque token présent dans l'ensemble du corpus devient une feature (ici 4465 tokens au total).

On compte le nombre d'occurrences de chaque feature au sein de la description étudiée (exemple : lamp 5 occurrences).

	word	value
32	lamp	5
30	handcrafted	3
29	glass	3
19	cm	3
43	power	3
13	blue	3
42	package	2
26	feature	2
35	material	2
37	model	2
38	mosaic	2
20	color	2
17	bulb	2
40	number	2
0	1	2
46	sale	2
6	35	2
49	table	2

#### TF-IDF

« Term Frequency – Inverse Document Frequency »

Même principe que pour Bag Of Word, sauf qu'ici on va confronter la fréquence d'apparition d'un mot à sa fréquence d'apparition dans le corpus.

Si un mot apparait trop souvent (exemple : le mot « produit »), on lui donnera un poids moins important.

	word	value
32	lamp	0.480430
30	handcrafted	0.277812
29	glass	0.255150
38	mosaic	0.230299
43	power	0.219067
6	35	0.212571
13	blue	0.195958
17	bulb	0.182188
51	unique	0.172204
49	table	0.161152
19	cm	0.138441
8	60hz	0.122185
16	buffet	0.122185
18	cfsi	0.122185
4	1x15	0.122185
5	240v	0.122185
24	eternity	0.122185
25	eternity001007	0.122185

# RandomForest avec GridSearchCV

- Hyper-paramètres de preprocessing (lemming ou stemming, différents jeux de stop words)
- Hyper-paramètres de RandomForest (n-estimators, max-depth, max\_features, ...)

Bag of Words

Validation

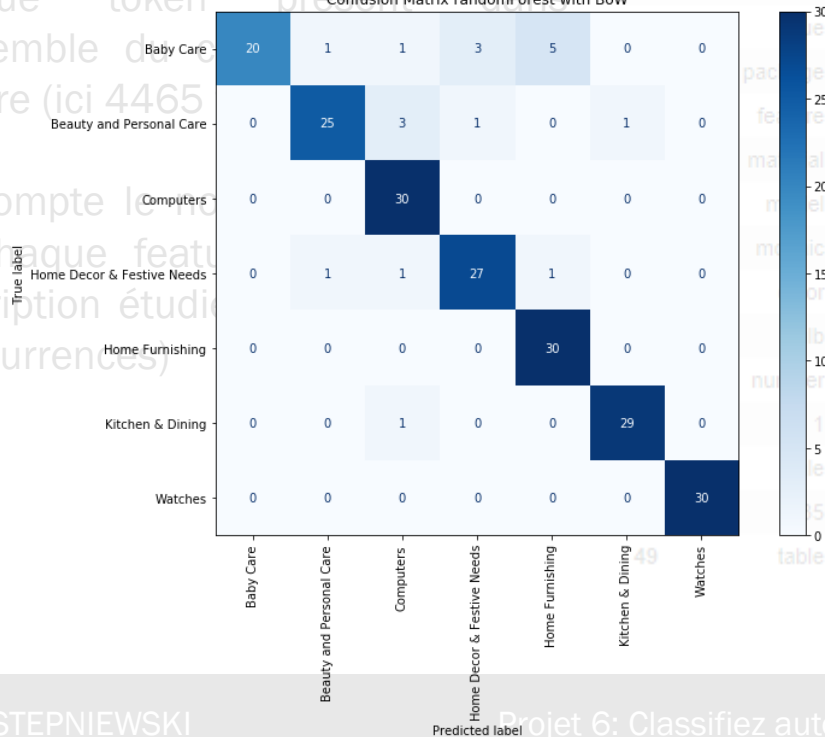
Accuracy Score

0,933

Test

0,909

Confusion Matrix randomForest with BoW



TF-IDF

Validation

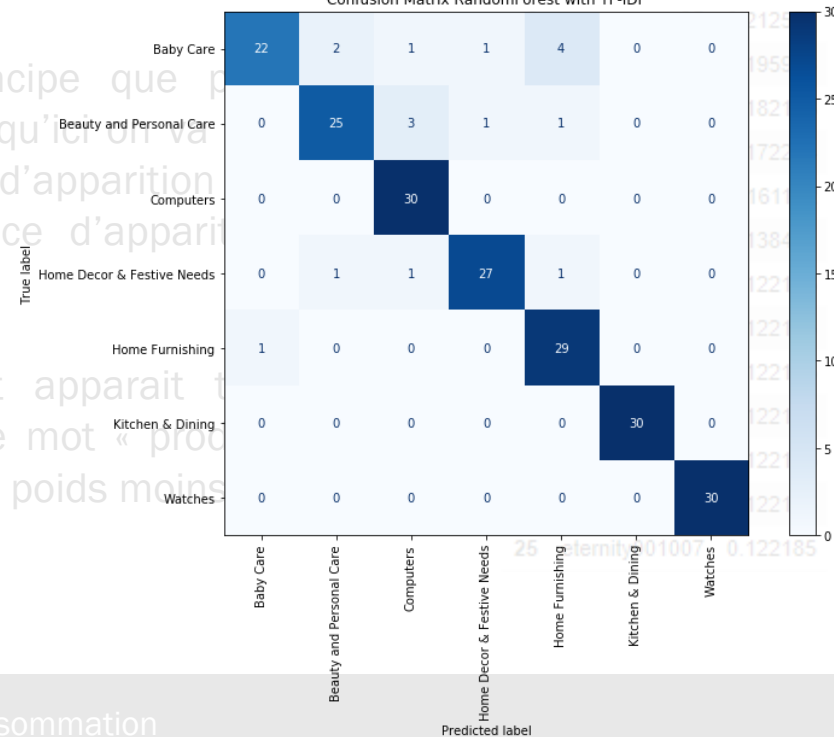
Accuracy Score

0,936

Test

0,919

Confusion Matrix RandomForest with TF-IDF



# 3. Classification par descriptions – Modèles

Modèles non-supervisés :

Topic modeling : LDA et NMF

Associer un thème à chaque document et trouver les mots les plus associés à chaque thème

LDA

Topic 0:  
warranty adapter laptop battery quality replacement product power charger vgn

Topic 1:  
cm 1 feature specification color price pack baby r cotton

Topic 2:  
r product free replacement cash delivery genuine buy day ship

Topic 3:  
buy genuine cash delivery ship free r product flipkart com

Topic 4:  
bowl sing scissor steel dessert dish shadow kitchen fork stainless

Topic 5:  
mug coffee gift ceramic perfect make one tea love design

Topic 6:  
usb light power lead portable flexible use phone otg android

NFM

Topic 0:  
com flipkart genuine cash ship delivery buy free product guarantee

Topic 1:  
rockmantra 5 mug ceramic porcelain permanent thrill stay start dishwasher

Topic 2:  
baby girl detail fabric cotton dress sleeve boy neck shirt

Topic 3:  
watch analog men woman discount india great dial maximum strap

Topic 4:  
mug coffee ceramic perfect tea printland gift one love get

Topic 5:  
cm showpiece cover cushion best design 5 1 inch pack

Topic 6:  
laptop skin shape mouse warranty pad print 6 inch battery



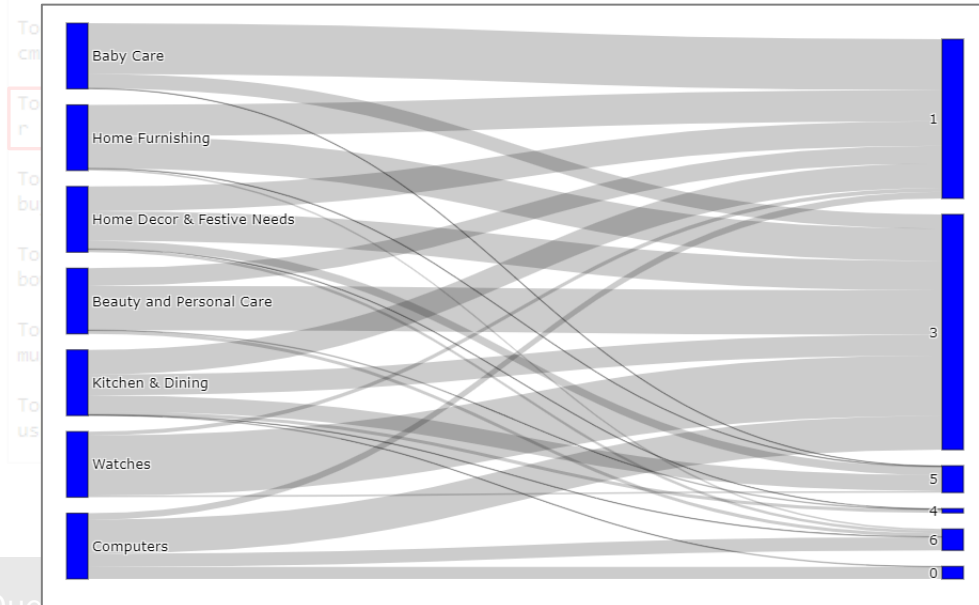
# 3. Classification par descriptions – Modèles

LDA

	0	1	2	3	4	5	6	category	real_cat	real_cat_id
0	0.001725	0.618707	0.001722	0.001722	0.001723	0.276581	0.097820	1	Home Decor & Festive Needs	0
1	0.001643	0.990144	0.001642	0.001643	0.001642	0.001643	0.001643	1	Baby Care	1
2	0.002470	0.985195	0.002464	0.002475	0.002464	0.002465	0.002466	1	Beauty and Personal Care	2
3	0.005318	0.005300	0.005292	0.968199	0.005292	0.005294	0.005305	3	Baby Care	1
4	0.007530	0.007552	0.007521	0.954785	0.007522	0.007562	0.007527	3	Home Furnishing	3
...	...	...	...	...	...	...	...	...	...	...
835	0.007149	0.298204	0.007145	0.666018	0.007145	0.007193	0.007147	3	Baby Care	1
836	0.004930	0.004933	0.004927	0.567729	0.004927	0.407626	0.004928	3	Kitchen & Dining	6
837	0.004770	0.004797	0.004764	0.971376	0.004785	0.004764	0.004764	3	Beauty and Personal Care	2
838	0.004611	0.272508	0.004609	0.704425	0.004609	0.004628	0.004610	3	Home Decor & Festive Needs	0
839	0.003664	0.003665	0.003663	0.978018	0.003663	0.003663	0.003664	3	Watches	5

840 rows x 10 columns

Topic 0:  
warranty adapter laptop battery quality replacement product power charger vgn

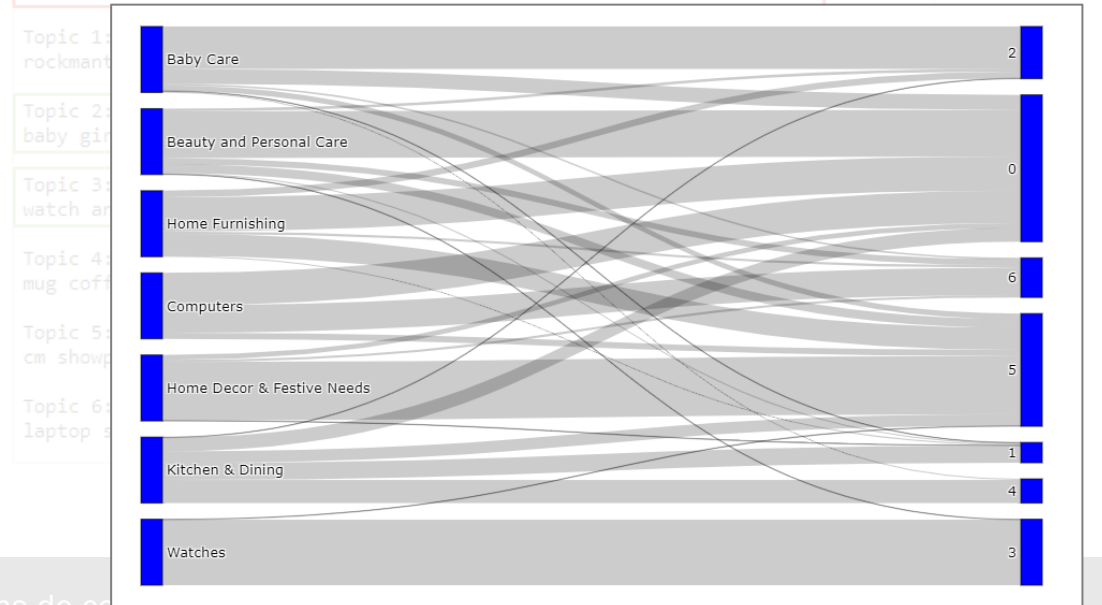


NMF

	0	1	2	3	4	5	6	category	real_cat	real_cat_id
0	0.000000	0.000000	0.012608	0.000000	0.000000	0.113504	0.002861	5	Home Decor & Festive Needs	0
1	0.000000	0.000000	0.225273	0.000000	0.000000	0.000000	0.000000	2	Baby Care	1
2	0.000000	0.000000	0.004264	0.003222	0.000000	0.008267	0.000000	5	Beauty and Personal Care	2
3	0.043016	0.000000	0.000000	0.000010	0.000000	0.002402	0.000000	0	Baby Care	1
4	0.112547	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	Home Furnishing	3
...	...	...	...	...	...	...	...	...	...	...
835	0.028747	0.000000	0.000000	0.007299	0.000000	0.000000	0.000000	0	Baby Care	1
836	0.062744	0.030902	0.000000	0.000000	0.149503	0.010903	0.000000	4	Kitchen & Dining	6
837	0.102116	0.005987	0.000000	0.000000	0.000000	0.000000	0.000000	0	Beauty and Personal Care	2
838	0.036471	0.000000	0.000000	0.000000	0.000000	0.111063	0.000000	5	Home Decor & Festive Needs	0
839	0.008186	0.000000	0.000000	0.203405	0.000000	0.000000	0.000000	3	Watches	5

840 rows x 10 columns

Topic 0:  
com flipkart genuine cash ship delivery buy free product guarantee



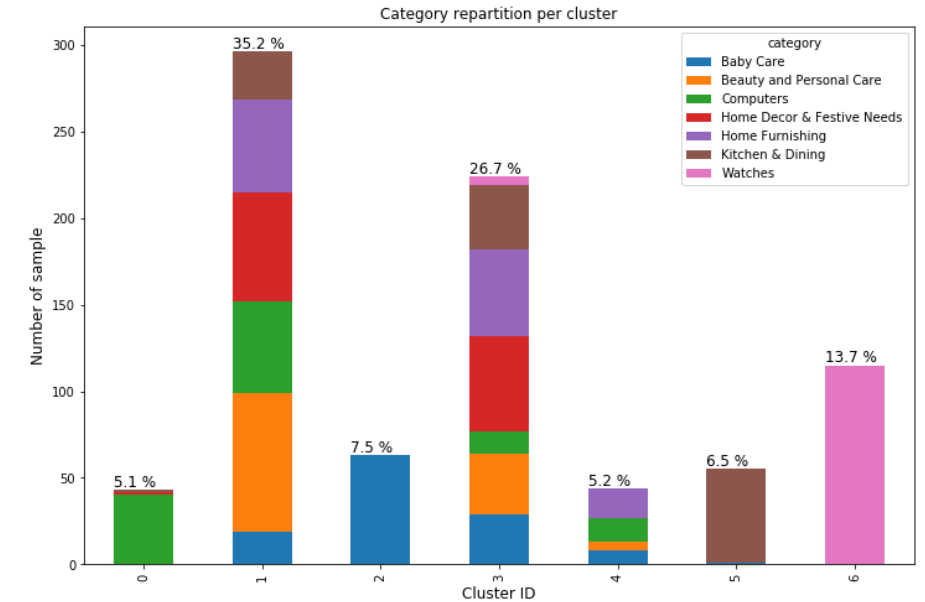
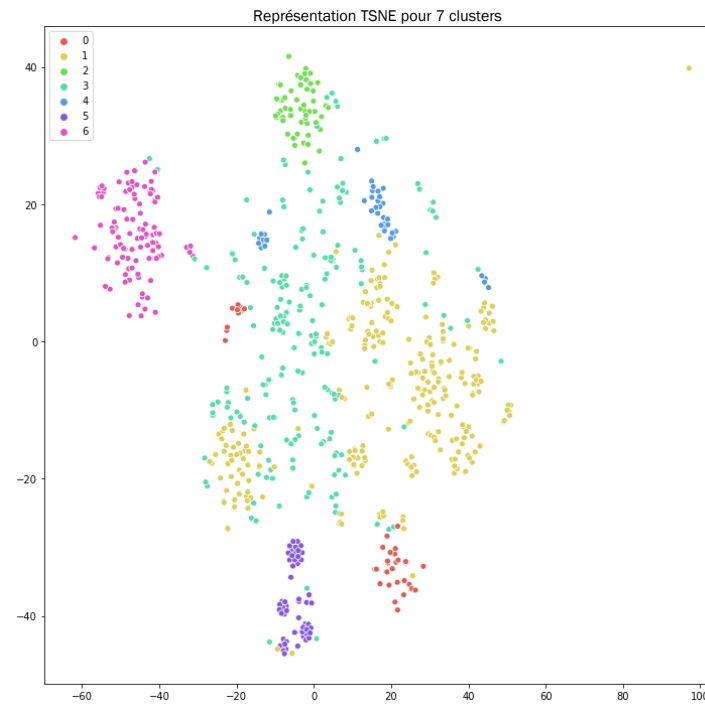
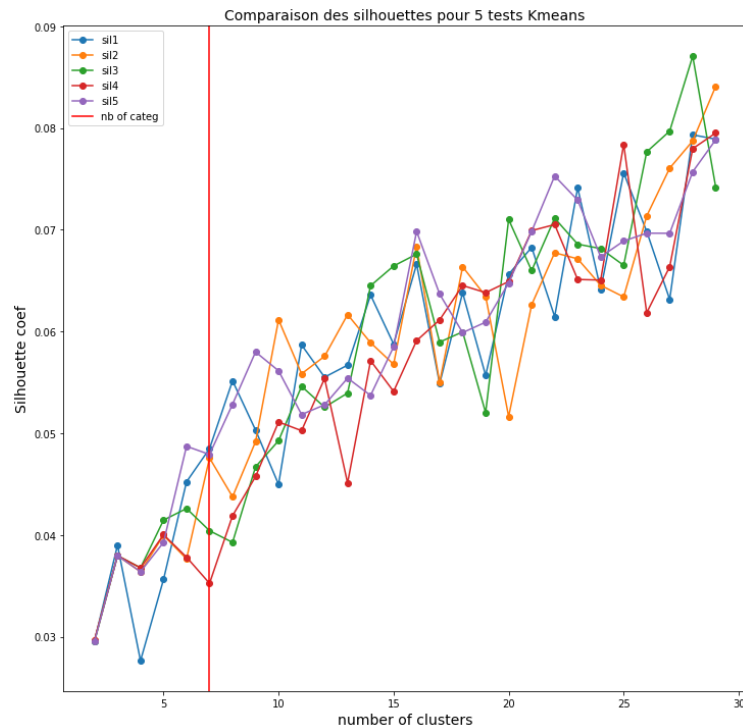
# 3. Classification par descriptions – Modèles

Modèles non-supervisés :

Algorithme de clustering : Kmeans

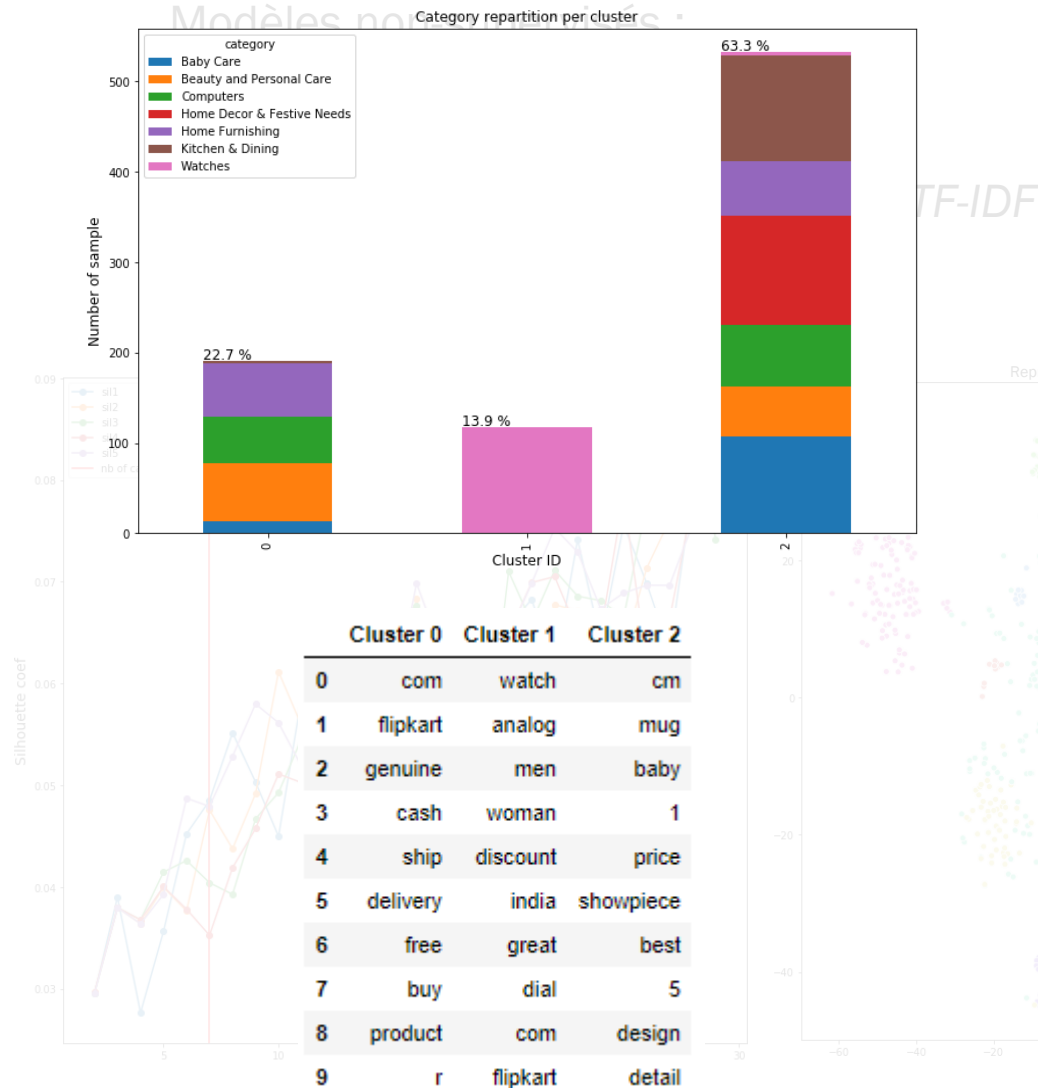
Utilisation des données en sortie de TF-IDF

Observation pour 7 clusters

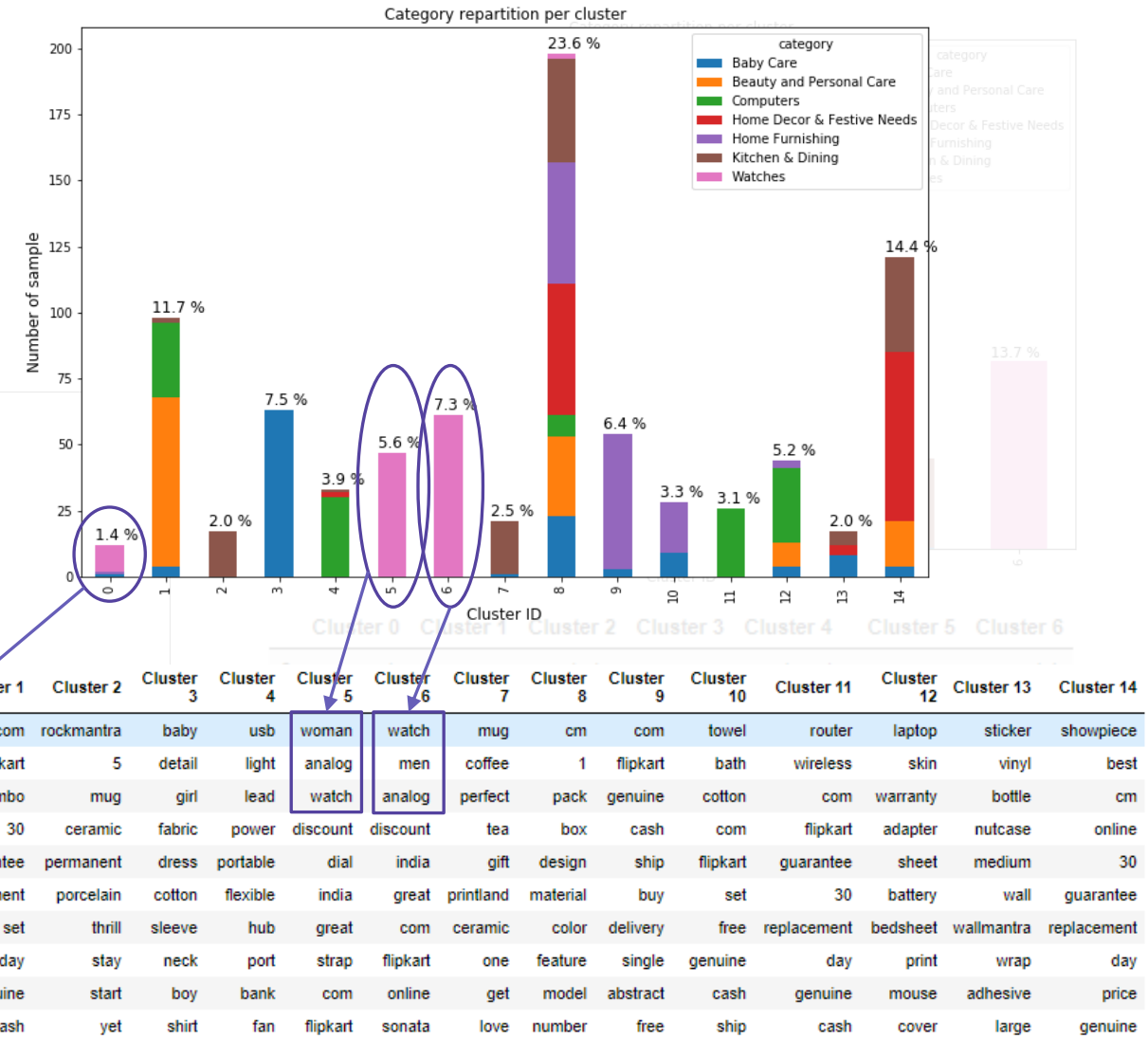


	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
0	usb	com	baby	cm	towel	mug	watch
1	light	flipkart	detail	1	skin	ceramic	analog
2	power	cash	girl	pack	bath	coffee	men
3	lead	genuine	fabric	box	cotton	perfect	woman
4	adapter	ship	dress	design	laptop	gift	discount
5	warranty	delivery	cotton	color	battery	rockmantra	india
6	portable	free	sleeve	material	set	love	great
7	flexible	buy	neck	feature	soft	one	com
8	laptop	product	boy	inch	mouse	5	dial
9	charger	30	shirt	cover	print	design	flipkart

## Observation pour 3 clusters



## Observation pour 15 clusters



### 3. Classification par descriptions – Modèles

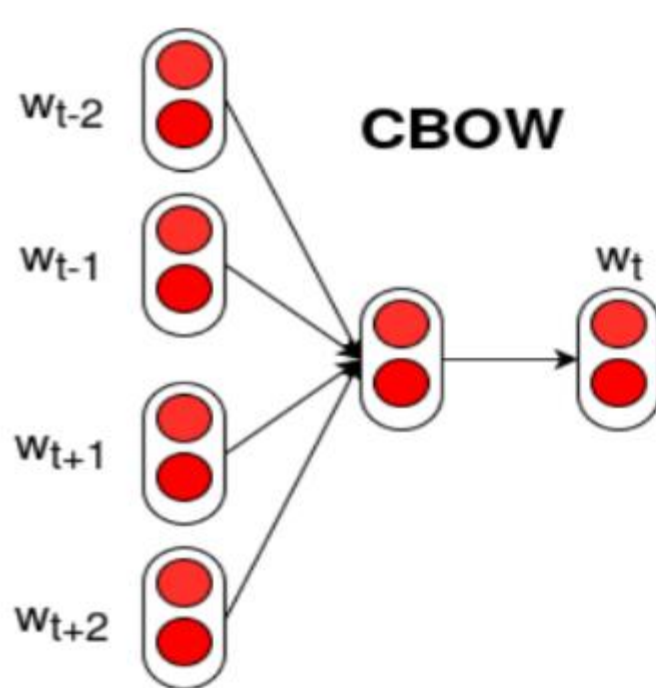
#### Word Embedding

Principe : représenter chaque mot par un vecteur en se basant sur l'hypothèse que des mots apparaissant dans un contexte similaire ont des vecteurs correspondants relativement proches

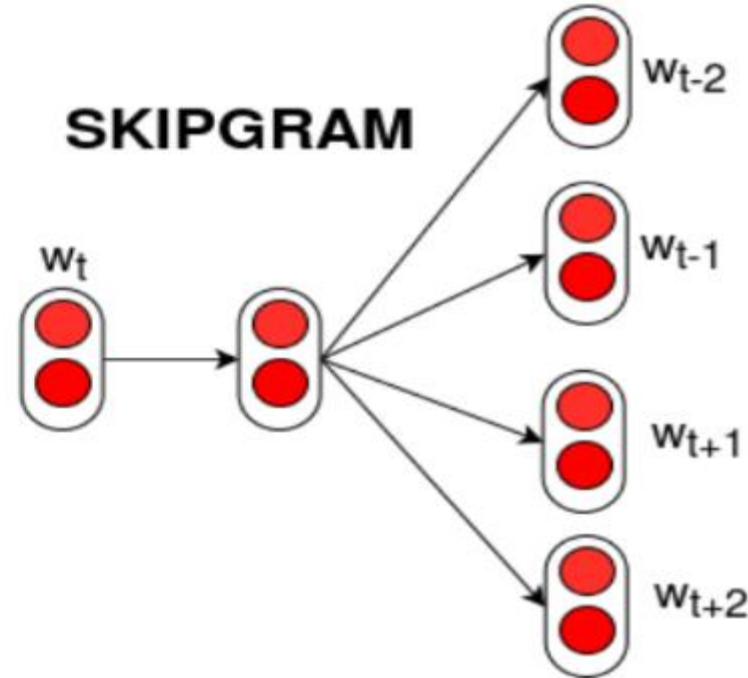
*Word2Vec : 2 algorithmes*

#### CBOW :

Retrouver le mot ciblé à partir de son contexte



#### **SKIPGRAM**



#### Skip-Gram :

A partir d'un mot, retrouver son contexte

Source Image : Wikipedia

### 3. Classification par descriptions – Modèles

#### Word Embedding

#### Doc2Vec : 2 Algorithmes

*Doc2Vec se base sur les mêmes principes que Word2Vec, à la différence qu'on cherche ici à rapprocher des documents entiers par leur contexte.*

#### PV-DM (basé sur CBOW) :

Trouver via un Paragraphe et des mots contexte, le mot ciblé

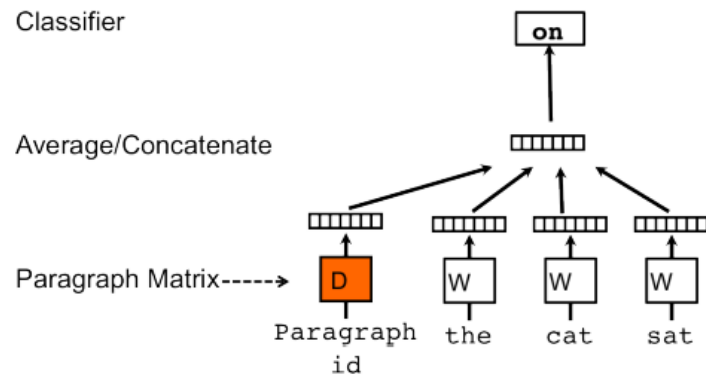


fig 3: PV-DM model

#### PV-DBOW (basé sur Skip-Gram) :

Trouver via un Paragraphe, les mots contextes

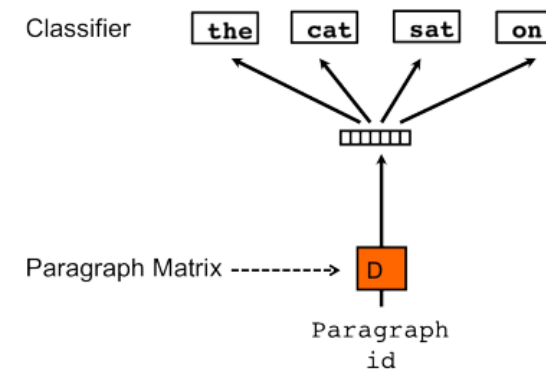


fig 4: PV-DBOW model



### 3. Classification par descriptions – Modèles

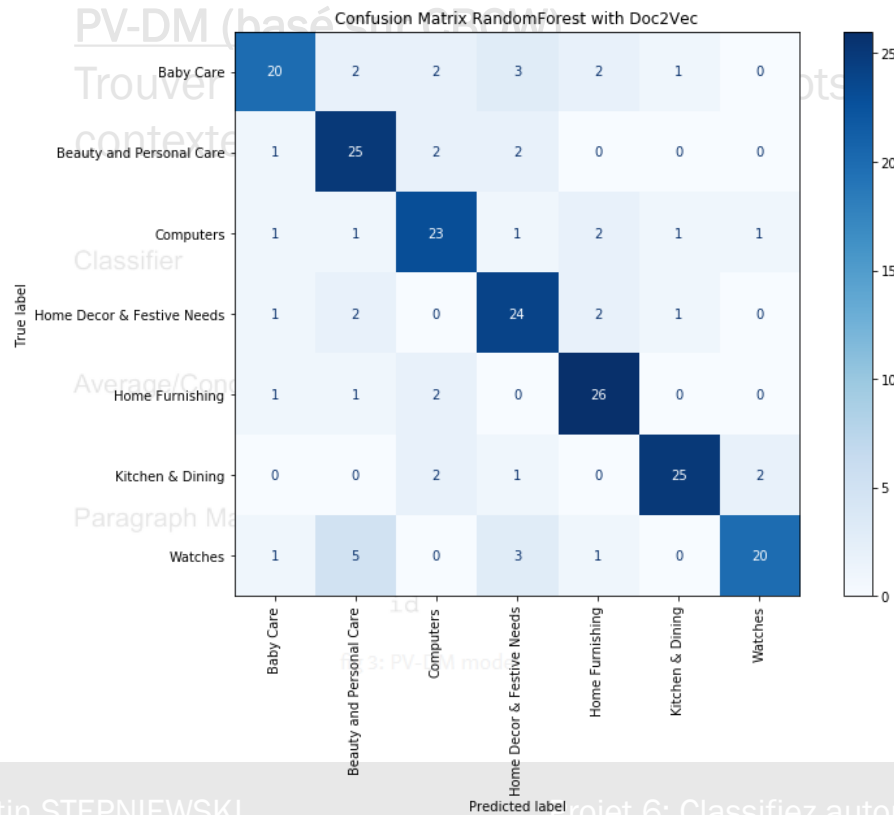
PV-DM (associé à une  
Random Forest)

Word

Doc2Vec : 2 Algorithmes

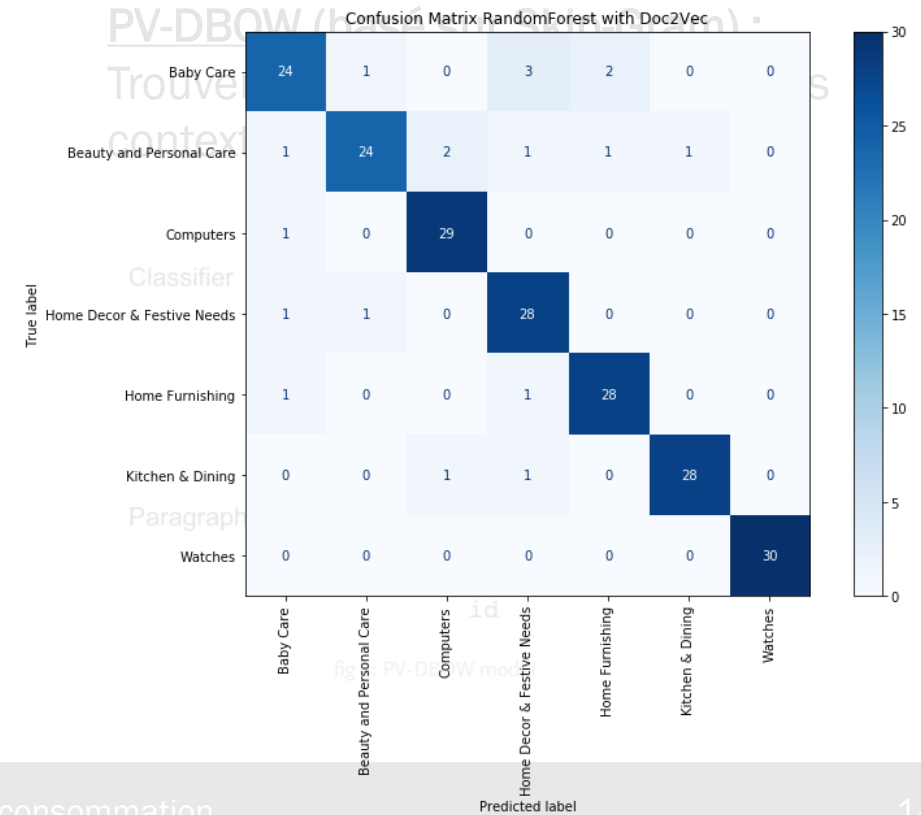
Doc2Vec se base sur les mêmes principes que Word2Vec, à la différence qu'on considère les documents entiers par leur contexte.

	Accuracy Score
Validation	0,936
Test	0,776



PV-DBOW (associé à une  
Random Forest)

	Accuracy Score
Validation	0,989
Test	0,923

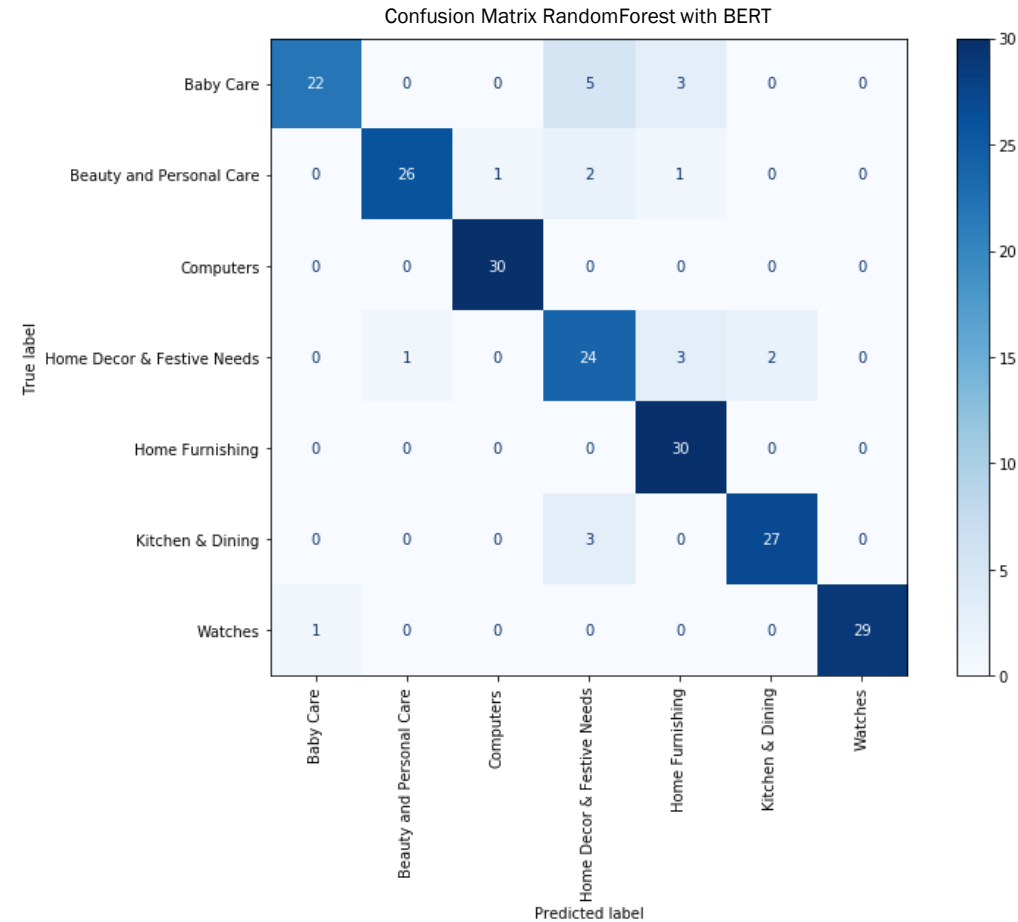


### 3. Classification par descriptions – Modèles

#### Transfer Learning

Modèle Bert, réseau de neurones pré-entraîné sur le corpus anglophone de l'encyclopédie en ligne Wikipédia

	Accuracy Score
Validation	0,901
Test	0,895



### 3. Classification par descriptions – Modèles

---

Résultats sur la partie NLP :

Ici, le meilleur modèle semble donc être Doc2Vec se basant sur l'algorithme PV-DBOW

	BOW	TF-IDF	LDA	NMF	Kmeans	Doc2Vec PV-DM	Doc2Vec PV-DBOW	BERT
Validation	0,933	0,936	Topics peu pertinents	Meilleurs résultats non-supervisés	Bons résultats uniquement sur les montres	0,936	0,989	0,901
Test	0,909	0,919				0,776	0,923	0,895

# 4. Classification par images – Pre-processing

Etapes de pre-processing :

- 1) Passage au noir et blanc
- 2) Réduction du bruit (ici flou gaussien)
- 3) Egaliseur
- 4) Redimensionnement

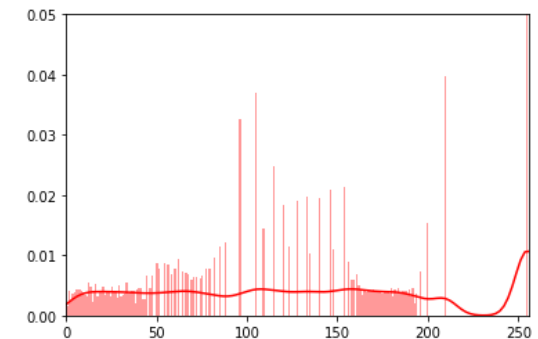
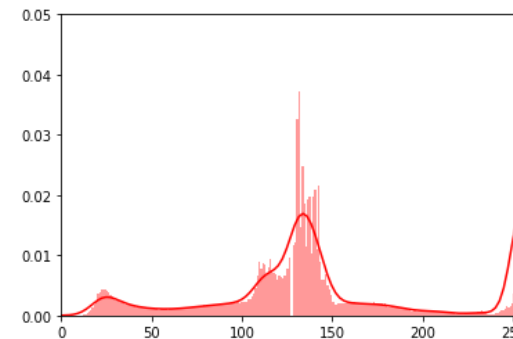


1)

2)

3)

*Ce pre-processing « basique » sera à optimiser par la suite (observation des performances de prédiction en faisant varier les étapes/paramètres de pre-processing)*

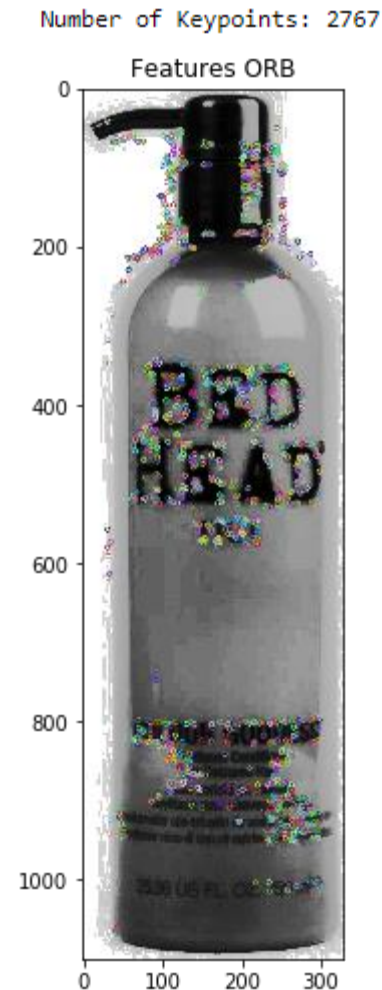


3)

# 4. Classification par images – Modèles

## Extraction de features : Méthode ORB

- Trouver des points d'intérêt sur chaque image
- Chaque point d'intérêt aura n descripteurs (ici 32 avec l'algorithme ORB)
- On aura donc pour chaque image une matrice (n\_keypoints X y\_descripteurs)
- On voudra créer des « catégories de keypoints » (faire tourner un algorithme de clustering sur l'ensemble des keypoints pour créer un « Bag of Visual Words » pour chaque image)
- Lancer un algorithme supervisé sur notre dataset de BoVW)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	32	26	34	11	7	8	9	15	43	20	29	29	15	58	28	29	33	28	18	28
1	17	7	39	6	37	12	0	42	20	55	16	39	46	13	43	18	13	11	2	64
2	27	2	20	6	31	4	4	33	28	70	27	5	39	17	42	14	17	49	6	59
3	11	7	27	10	13	13	2	23	17	50	25	24	34	25	38	26	17	48	15	75
4	35	11	37	8	17	1	4	26	20	19	18	18	42	76	20	56	26	58	0	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
835	39	1	11	6	43	11	14	40	32	37	21	7	52	21	21	25	28	28	2	35
836	27	83	11	39	8	15	12	31	35	21	30	28	34	7	39	8	24	17	3	28
837	24	5	31	25	37	12	10	25	23	32	25	13	16	23	24	29	49	41	5	51
838	49	9	29	10	22	9	6	40	24	18	64	19	20	25	34	24	34	30	9	25
839	26	40	23	6	19	11	10	15	34	21	34	23	67	48	31	19	5	26	6	36

840 rows × 20 columns

Observation pour 20 clusters de keypoints



## RandomForest avec GridSearchCV

- Hyper-paramètres de pre-processing (Gaussian ou Median Blur, taille du kernel pour le pre-processing, nombre de keypoints par image, nombre de VisualWords...)
- Hyper-paramètres de RandomForest (n-estimators, max-depth, max\_features, ...)

- Trouver des points d'intérêt sur chaque image

- Chaque point d'intérêt aura n descripteurs (ici

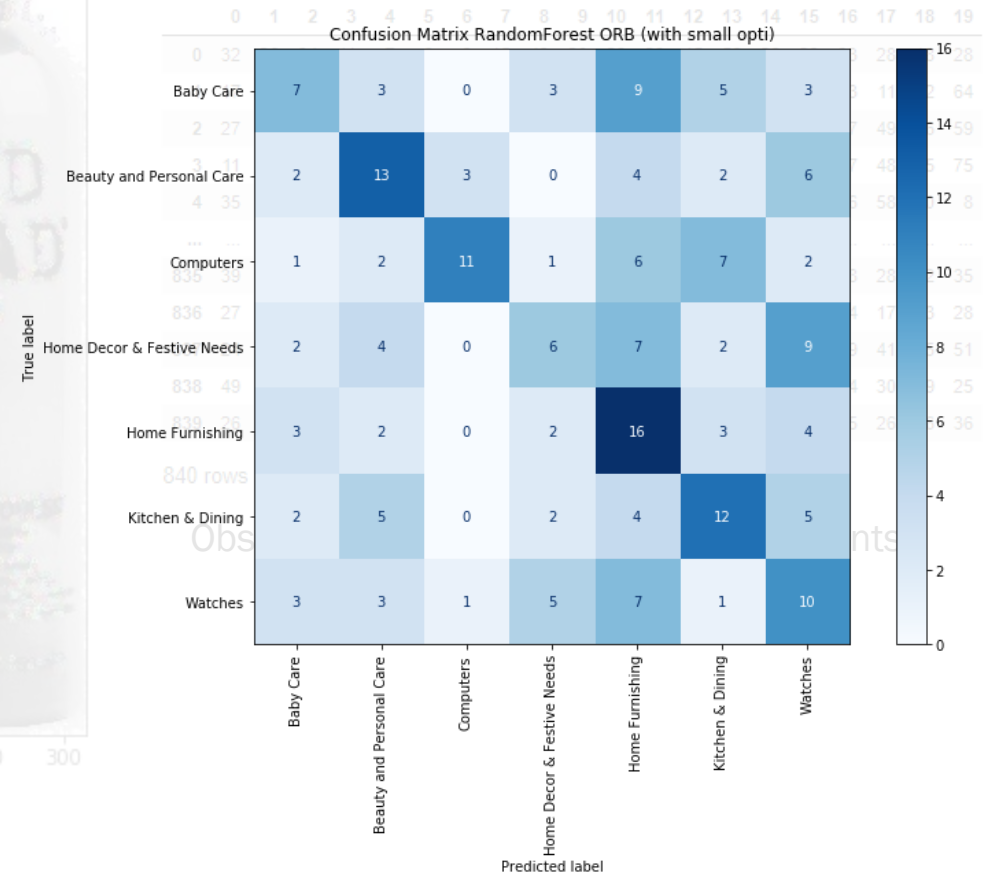
Validation Score (accuracy): 0.3928571428571428

Best Params: {'imagepreprocessor\_gaussormedian': False, 'imagepreprocessor\_kernel': 3, 'imagestobovw\_nb\_keypoints': 800, 'imagestobovw\_nb\_visual\_words': 30}

Test Score (accuracy): 0.35714285714285715

Résultats dans l'ensemble assez décevants pour cette méthode, on pourra cependant probablement améliorer ce score en ajoutant de la complexité aux paramètres pouvant être optimisés dans la GridSearchCV :

- Tester les nombreux filtres de pre-processing disponibles sur OpenCV ou d'autres modules
- Augmenter le nombre de keypoints / Visual Words
- Augmenter le nombre d'images à l'entraînement



## 4. Classification par images – Modèles

---

Convolutional Neural Network (CNN) :

Un CNN est différent d'un réseau de neurones classique :

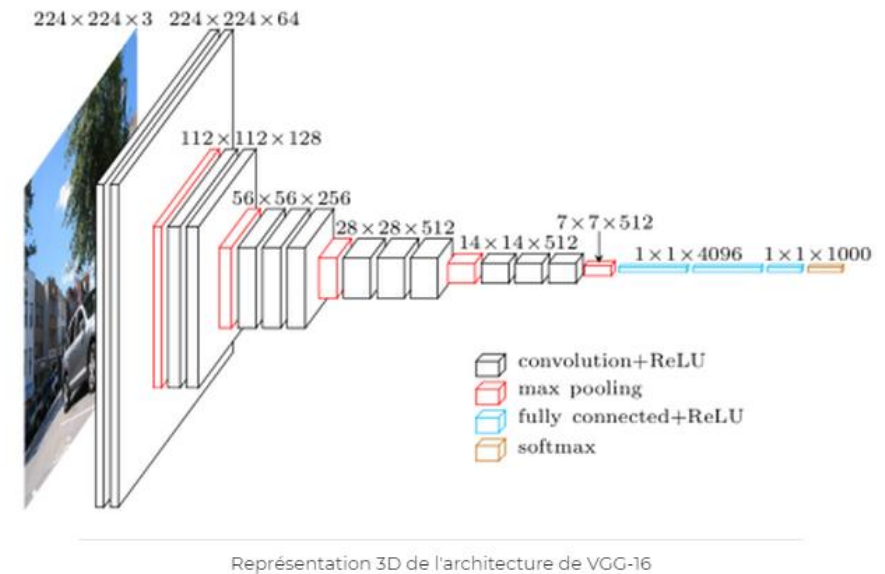
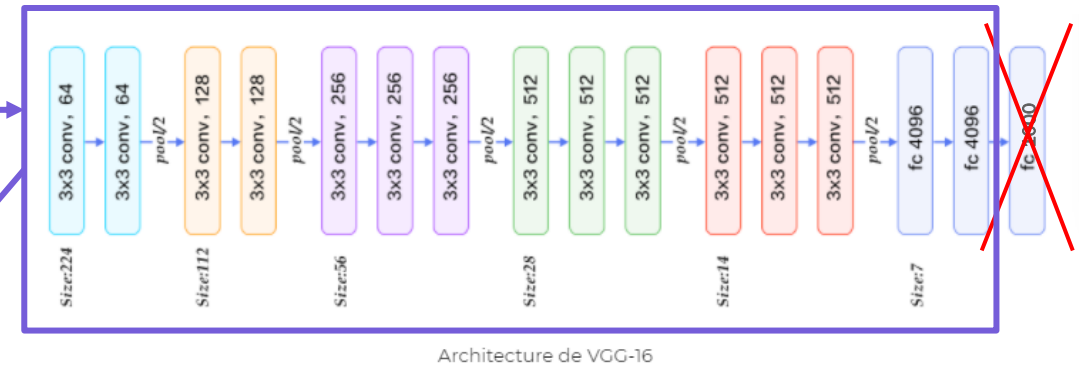
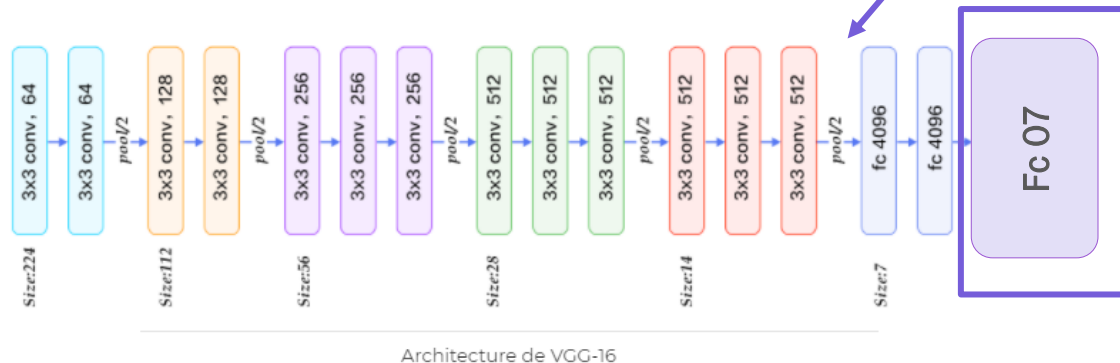
1. Reçoit en entrée des images et aura pour but dans un premier temps **d'extraire des features** grâce à plusieurs couches spécifiques :
  - Couches de Convolution : filtrage des images par convolution pour détecter des features. Ces features sont définies par un noyau de convolution qui va nous permettre de créer une features map en le passant sur chaque image
  - Couche d'activation Relu : transforme les valeurs négatives en zéro (fonction d'activation)
  - Couche de Pooling : Divise une feature map en petites fenêtres de même tailles qui peuvent se recouvrir, et conserve le maximum local pour ces fenêtres. L'idée étant de créer une features map plus petite
2. Dans un second temps, comme pour un réseau de neurones classique, on cherchera à utiliser ces features pour **effectuer une classification**, grâce aux couches fully connected

## 4. Classification par images – Modèles

### Keras CNN VGG16 (Transfer Learning) :

Modèle CNN entraîné sur 14 millions d'images provenant du site ImageNet pour une classification à 1000 classes.

Pour du transfer learning, on supprime uniquement la dernière couche propre à la problématique initiale pour en recréer une adaptée à la nôtre



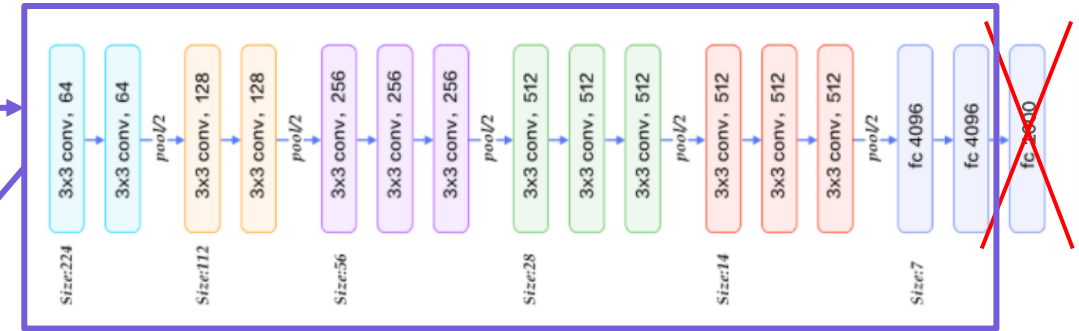
# 4. Classification par images – Modèles

Keras CNN VGG16 (Transfer Learning) :

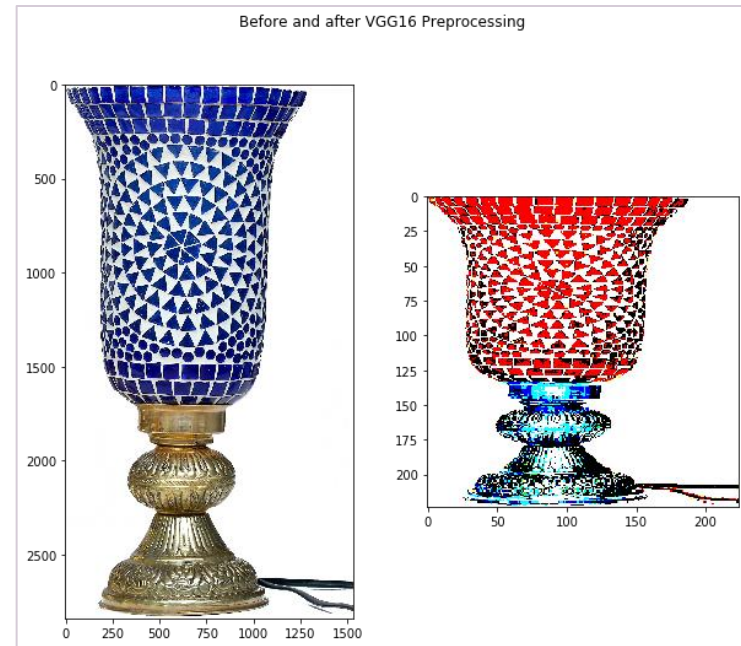
Modèle CNN entraîné sur 14 millions d'images provenant du site ImageNet pour une classification à 1000 classes.

Pour du transfer learning, on supprime uniquement la dernière couche propre à la problématique initiale pour en recréer une adaptée à la nôtre

Il faudra **réentraîner** la dernière couche avec les images pré-processées



Architecture de VGG-16



Before and after VGG16 Preprocessing



Architecture de VGG-16

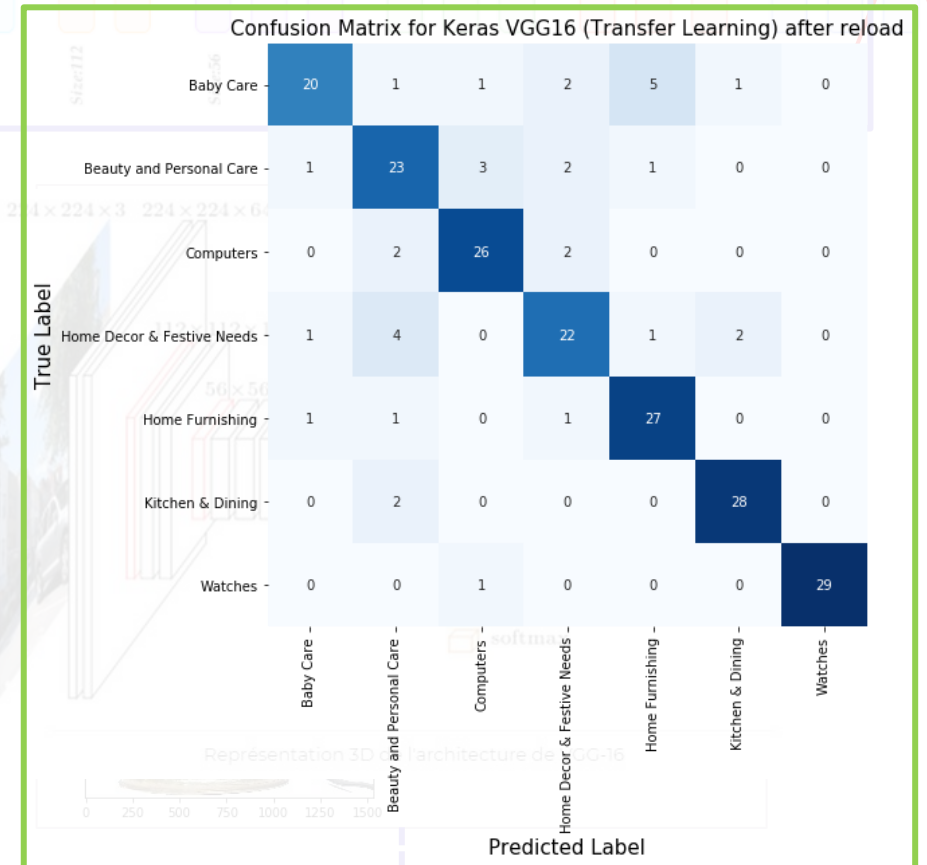
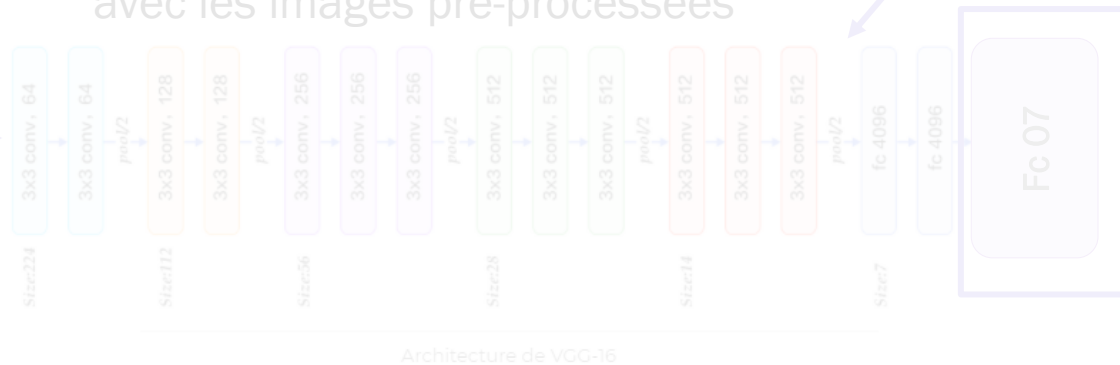
## Entraînement CNN

- Division de notre train set en train set (80%) et validation set (20%) pour EarlyStopping (éviter d'overfitter notre modèle)
- Evaluation de l'Accuracy pour différentes batch sizes

Modèle CNN entraîné sur 14 million d'images provenant du site ImageNet pour une classification à 1000 Classes.

	Batch Size: 50	Batch Size: 60	Batch Size: 30	Batch Size: 20	Batch Size: 10
Validation	0,863	0,857	0,863	0,875	0,881
Test	0,833	0,833	0,852	0,833	0,833

Il faudra reentraîner la dernière couche avec les images pre-processées



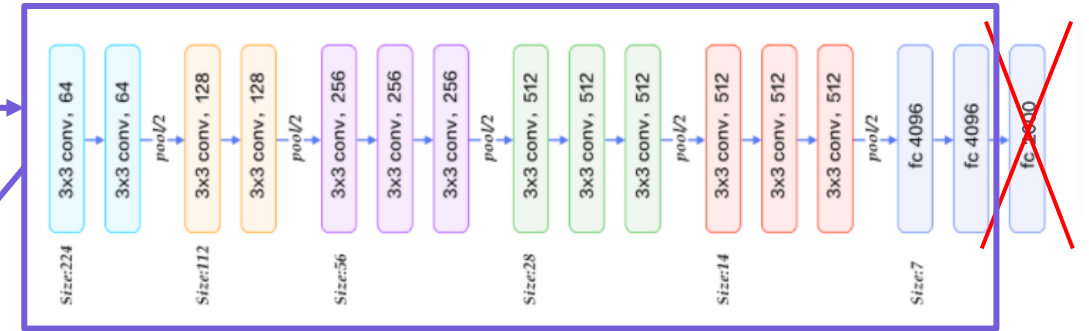


# 4. Classification par images – Modèles

## Keras CNN VGG16 (Features Extraction) :

Pour du features extraction, on supprime également la dernière couche et on aura juste à prédire les 4096 features de sorties associées à chaque image de notre base de données

Ici, j'ai choisi de mettre en place une classification non-supervisée à partir de ces features

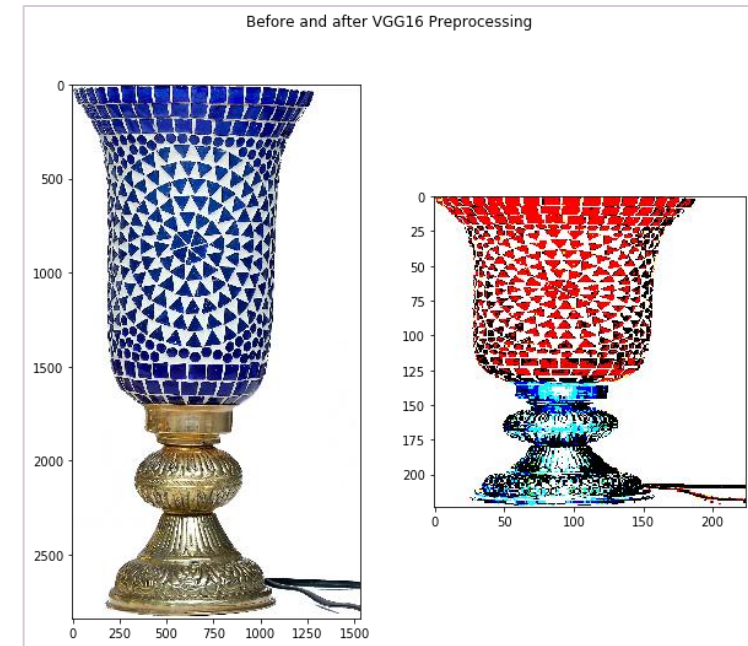


Architecture de VGG-16



Architecture de VGG-16

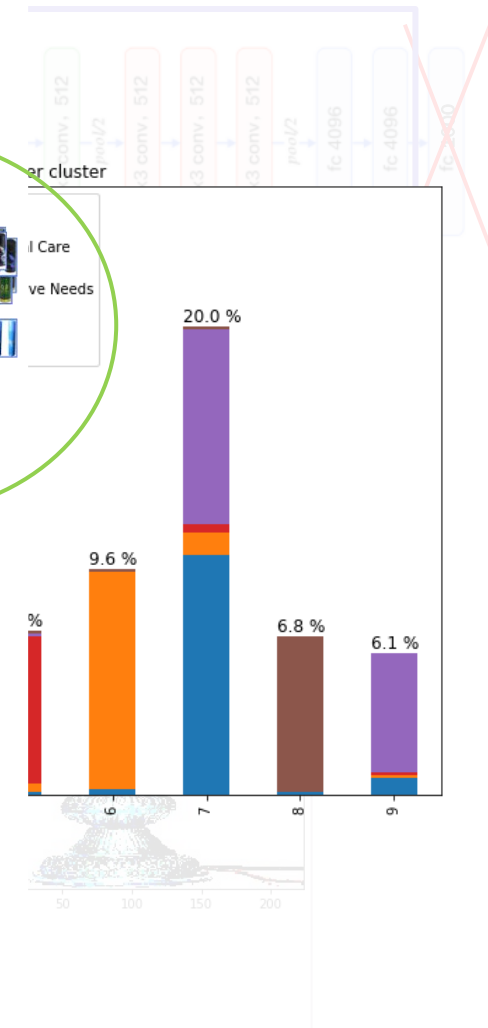
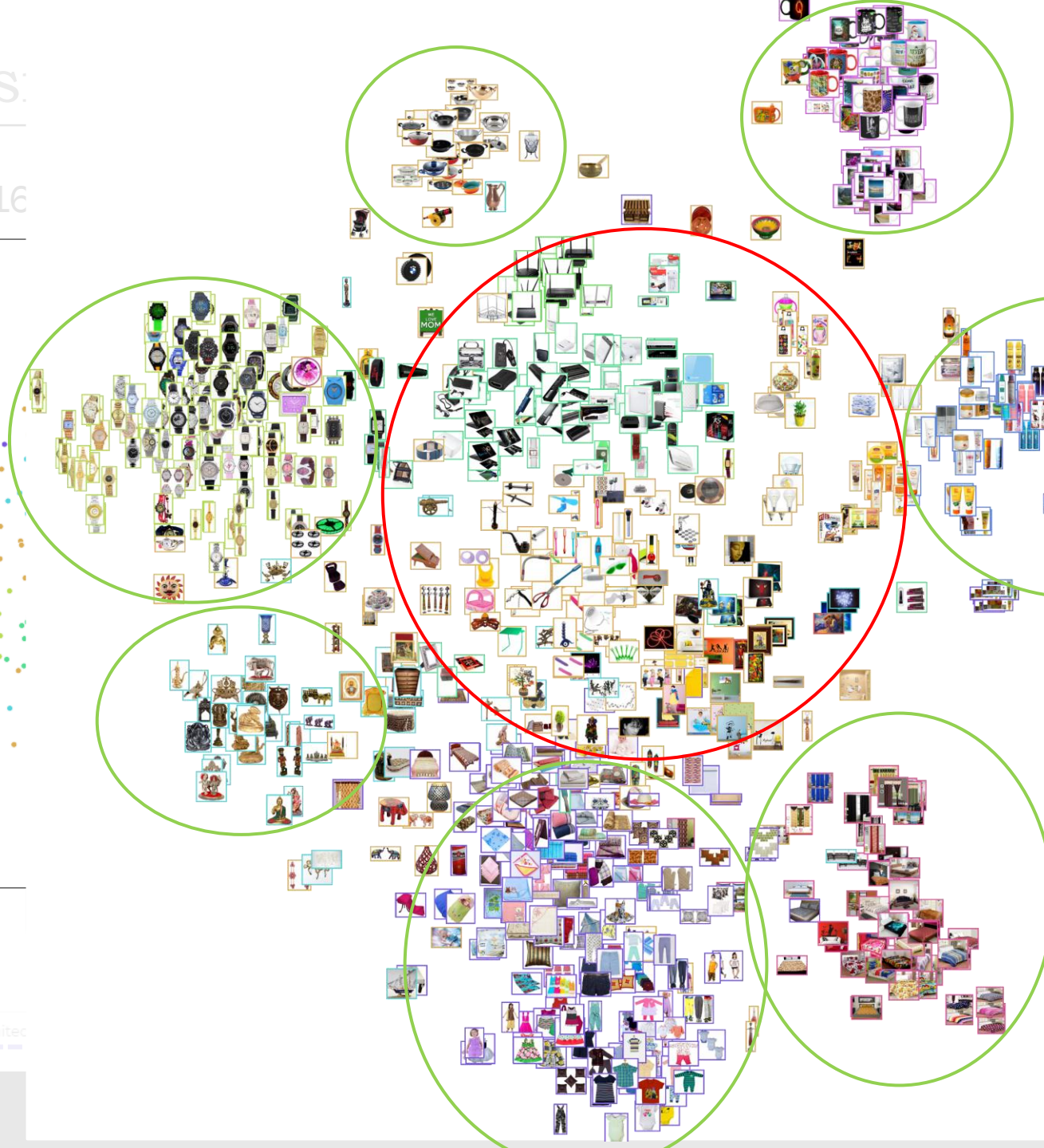
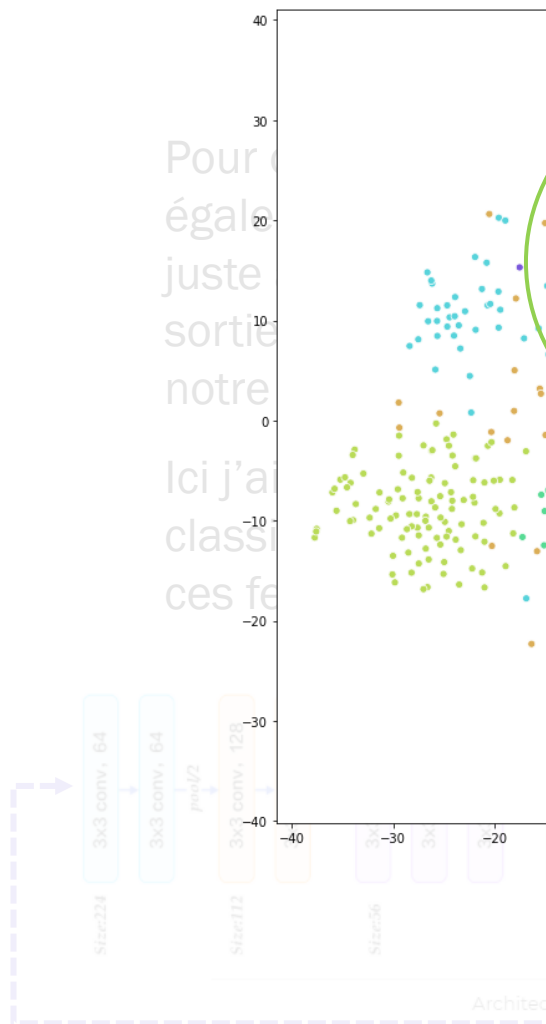
4096 features en sortie pour chaque image



# 4. Class:

Keras CNN VGG16

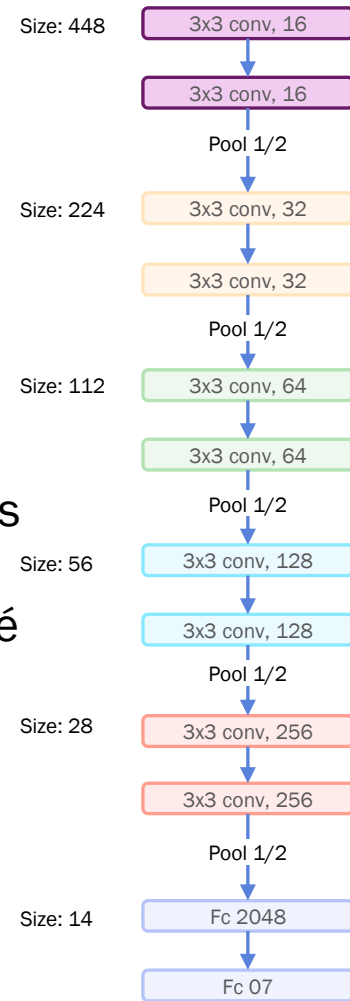
Pour éga juste sortie notre Ici j'a classi ces fe



# 4. Classification par images – Modèles

Keras CNN, construction de zéro :

- 1) Création des différentes couches
- 2) Input images 448x448 pixels, avec le même pre-processing que VGG16
- 3) Entraînement du modèle
- 4) Test pour plusieurs configurations de modèles (notamment sur la dernière couche fully-connected qui représente une grosse quantité de paramètres)



Model: "sequential\_1"

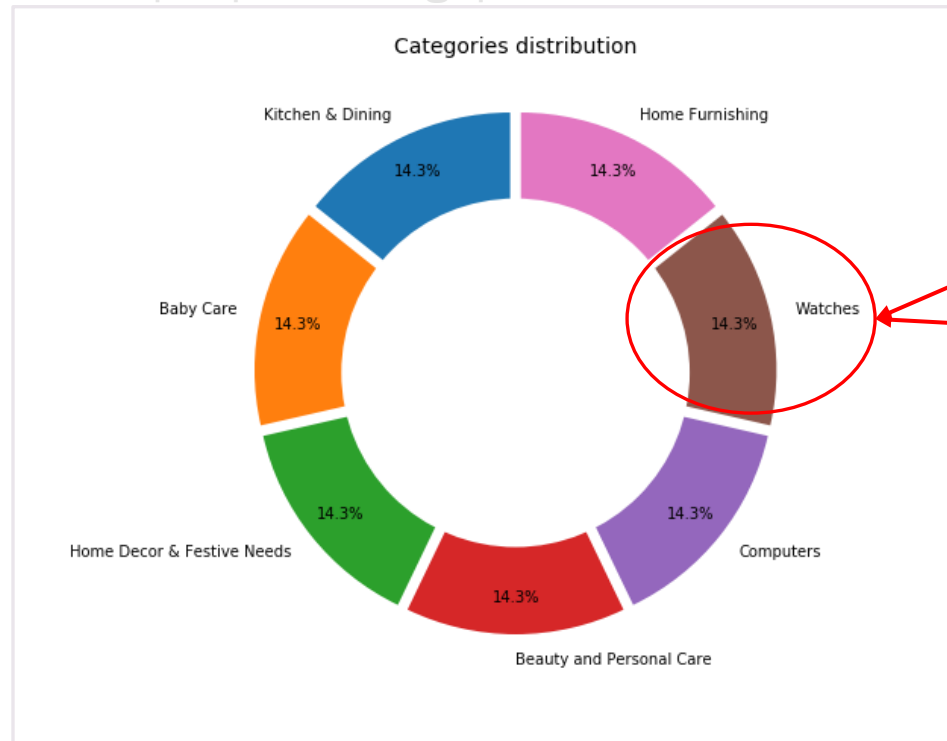
Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 448, 448, 16)	448
conv2d_11 (Conv2D)	(None, 448, 448, 16)	2320
max_pooling2d_5 (MaxPooling2D)	(None, 224, 224, 16)	0
conv2d_12 (Conv2D)	(None, 224, 224, 32)	4640
conv2d_13 (Conv2D)	(None, 224, 224, 32)	9248
max_pooling2d_6 (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_14 (Conv2D)	(None, 112, 112, 64)	18496
conv2d_15 (Conv2D)	(None, 112, 112, 64)	36928
max_pooling2d_7 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_16 (Conv2D)	(None, 56, 56, 128)	73856
conv2d_17 (Conv2D)	(None, 56, 56, 128)	147584
max_pooling2d_8 (MaxPooling2D)	(None, 28, 28, 128)	0
conv2d_18 (Conv2D)	(None, 28, 28, 256)	295168
conv2d_19 (Conv2D)	(None, 28, 28, 256)	590080
max_pooling2d_9 (MaxPooling2D)	(None, 14, 14, 256)	0
flatten_13 (Flatten)	(None, 50176)	0
dense_13 (Dense)	(None, 2048)	102762496
dense_14 (Dense)	(None, 7)	14343
Total params: 103,955,607		
Trainable params: 103,955,607		
Non-trainable params: 0		

Les résultats sont les mêmes peu importe la configuration choisie pour les CNN :

- Accuracy maximum : ~14 % (répondre la même catégorie pour chaque image)
- Il semble nécessaire d'obtenir un dataset beaucoup plus conséquent pour obtenir des résultats intéressants avec cette méthode

```
Epoch 1/30
68/68 - 328s - loss: 1.9475 - accuracy: 0.1369 - val_loss: 1.9459 - val_accuracy: 0.1481
Epoch 2/30
68/68 - 331s - loss: 1.9467 - accuracy: 0.1176 - val_loss: 1.9458 - val_accuracy: 0.1481
Epoch 3/30
68/68 - 347s - loss: 1.9468 - accuracy: 0.1310 - val_loss: 1.9458 - val_accuracy: 0.1481
Epoch 4/30
68/68 - 356s - loss: 1.9465 - accuracy: 0.1295 - val_loss: 1.9458 - val_accuracy: 0.1481
Epoch 5/30
68/68 - 340s - loss: 1.9464 - accuracy: 0.1429 - val_loss: 1.9458 - val_accuracy: 0.1481
Epoch 6/30
Restoring model weights from the end of the best epoch.
68/68 - 338s - loss: 1.9466 - accuracy: 0.1235 - val_loss: 1.9458 - val_accuracy: 0.1481
Epoch 00006: early stopping
```

pre-processing que VGG16



Size: 112

Size: 56

Size: 28

Size: 14

Confusion Matrix for Keras VGG16 (Transfer Learning)

True Label \ Predicted Label	Baby Care	Beauty and Personal Care	Computers	Home Decor & Festive Needs	Home Furnishing	Kitchen & Dining	Watches
Baby Care	0	0	0	0	30	0	0
Beauty and Personal Care	0	0	0	0	30	0	0
Computers	0	0	0	0	30	0	0
Home Decor & Festive Needs	0	0	0	0	30	0	0
Home Furnishing	0	0	0	0	30	0	0
Kitchen & Dining	0	0	0	0	30	0	0
Watches	0	0	0	0	30	0	0

## 4. Classification par images – Modèles

---

Résultats sur la partie NLP :

Ici, le meilleur modèle semble donc être Doc2Vec se basant sur l'algorithme PV-DBOW.

	ORB	VGG16 Transfer Learning	VGG16 Features extraction	CNN from scratch
Validation	0,392	0,881	Des topics intéressant mais manque de précision	0,143
Test	0,357	0,833		0,143



## 5. Modèle final (Images + Texte)

---

Modèle final :

- Modèle se basant à la fois sur les images et le texte

2 possibilités envisagées :

Fusion des features texte + image

Features à assembler :

- Doc2vec (1050 lignes x 50 colonnes)
- VGG16 (1050 lignes x 4096 colonnes)
- Fort déséquilibre entre image et texte, on effectuera également une réduction de dimension côté image via PCA
- Une fois les features assemblées, on fera tourner un algorithme de classification (ici RandomForest)

Stacking des modèles

Principe du stacking :

- 1) Entraîner une première fois plusieurs modèles différents sur notre dataset
- 2) Récupérer les probabilités en sortie de nos modèles afin de les réutiliser en tant que features en entrées d'un nouveau modèle de classification

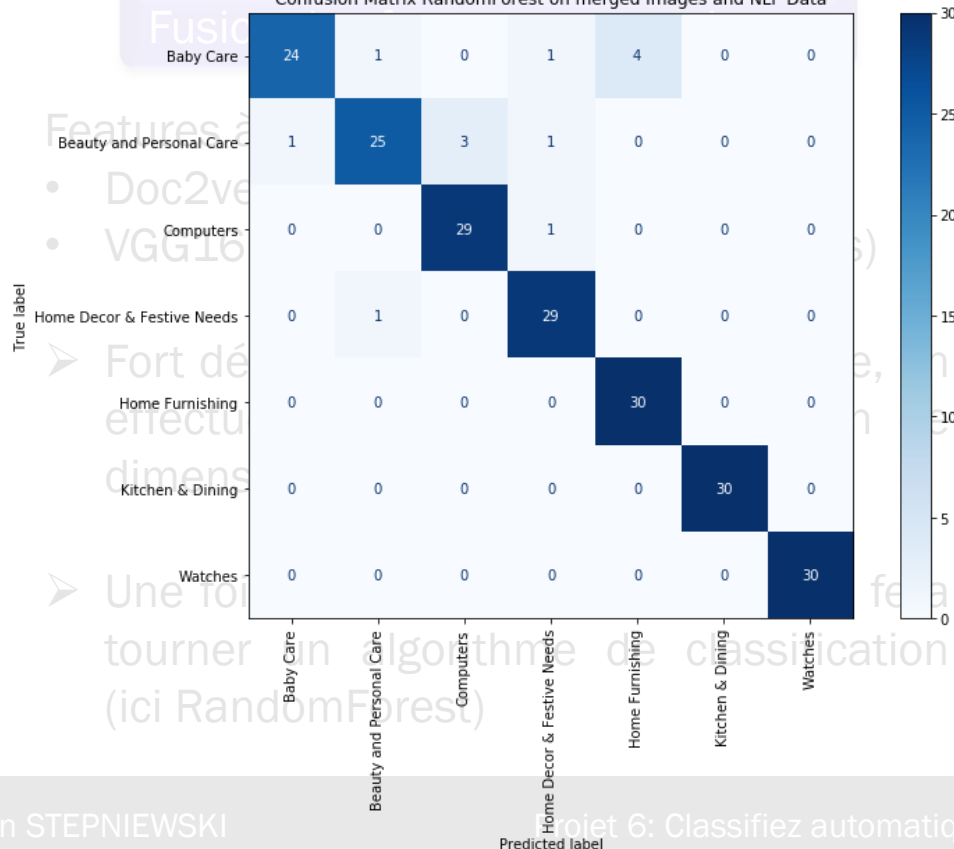
## Fusion des features texte + image

### 3.2 NLP: 50 Features / Images : 134 Features

#### Accuracy Score

Validation	0,981
Test	0,938

Confusion Matrix RandomForest on merged Images and NLP Data



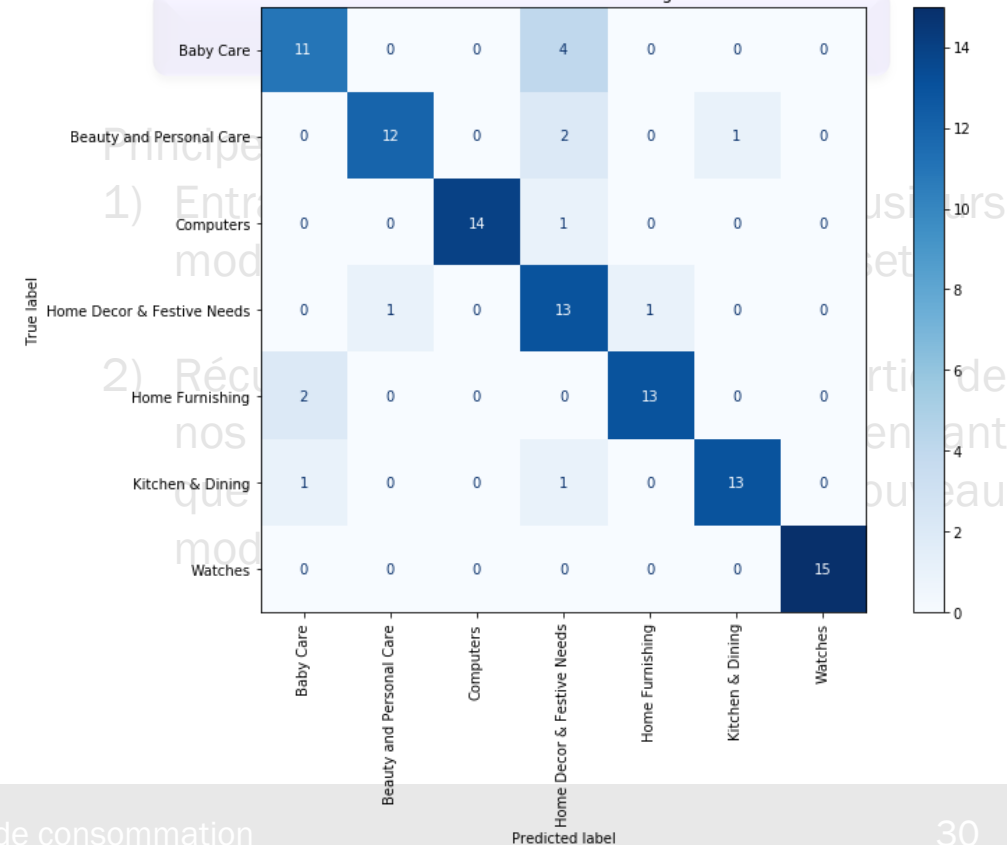
## Stacking des modèles

#### Accuracy Score

Validation	0,952
Test	0,866

Test set final plus petit pour pouvoir entraîner le modèle de stacking

Confusion Matrix Stacking



## 6. Conclusion sur la faisabilité et mise en perspective

- Des résultats très encourageants en termes de précision:

	Classification par description	Classification par image	Classification mixte
Validation	0,989	0,881	0,981
Test	0,923	0,833	0,938

➤ On peut donc ici être vraiment confiant quant à la faisabilité du projet

- De nombreuses pistes d'améliorations sont à explorer :
  - ❑ Il semble vraiment très important d'augmenter la base de donnée à disposition afin d'améliorer les performances de nos algorithmes (notamment côté images)
  - ❑ On pourra également essayer de faire varier les algorithmes de classification (autres que RandomForest) afin de choisir le plus performant sur notre problématique
  - ❑ Enfin on pourra songer à mettre en place un modèle d'ensemble plus complexe qui pourrait éventuellement augmenter encore un peu la précision de notre modèle

## 6. Conclusion sur la faisabilité et mise en perspective

- Des rés

### KDD Cup 2015 Solution



\*Jeong-Yoon Lee, Winning Data Science Competitions

Merci de votre attention







# Slides Bonus

## LDA

	0	1	2	3	4	5	6	category	real_categ	real_categ_id
0	0.001725	0.618707	0.001722	0.001722	0.001723	0.276581	0.097820	1	Home Decor & Festive Needs	0
1	0.001643	0.990144	0.001642	0.001643	0.001642	0.001643	0.001643	1	Baby Care	1
2	0.002470	0.985195	0.002464	0.002475	0.002464	0.002465	0.002466	1	Beauty and Personal Care	2
3	0.005318	0.005300	0.005292	0.968199	0.005292	0.005294	0.005305	3	Baby Care	1
4	0.007530	0.007552	0.007521	0.954785	0.007522	0.007562	0.007527	3	Home Furnishing	3
...	...	...	...	...	...	...	...	...	...	...
835	0.007149	0.298204	0.007145	0.666018	0.007145	0.007193	0.007147	3	Baby Care	1
836	0.004930	0.004933	0.004927	0.567729	0.004927	0.407626	0.004928	3	Kitchen & Dining	6
837	0.004770	0.004797	0.004764	0.971376	0.004765	0.004764	0.004764	3	Beauty and Personal Care	2
838	0.004611	0.272508	0.004609	0.704425	0.004609	0.004628	0.004610	3	Home Decor & Festive Needs	0
839	0.003664	0.003665	0.003663	0.978018	0.003663	0.003663	0.003664	3	Watches	5

840 rows × 10 columns

```
Topic 0:
warranty adapter laptop battery quality replacement product power charger vgn

Topic 1:
cm 1 feature specification color price pack baby r cotton

Topic 2:
r product free replacement cash delivery genuine buy day ship

Topic 3:
buy genuine cash delivery ship free r product flipkart com

Topic 4:
bowl sing scissor steel dessert dish shadow kitchen fork stainless

Topic 5:
mug coffee gift ceramic perfect make one tea love design

Topic 6:
usb light power lead portable flexible use phone otg android
```

## NFM

	0	1	2	3	4	5	6	category	real_categ	real_categ_id
0	0.000000	0.000000	0.012608	0.000000	0.000000	0.113504	0.002861	5	Home Decor & Festive Needs	0
1	0.000000	0.000000	0.225273	0.000000	0.000000	0.000000	0.000000	2	Baby Care	1
2	0.000000	0.000000	0.004264	0.003222	0.000000	0.008267	0.000000	5	Beauty and Personal Care	2
3	0.043016	0.000000	0.000000	0.000010	0.000000	0.002402	0.000000	0	Baby Care	1
4	0.112547	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	Home Furnishing	3
...	...	...	...	...	...	...	...	...	...	...
835	0.028747	0.000000	0.000000	0.007299	0.000000	0.000000	0.000000	0	Baby Care	1
836	0.062744	0.030902	0.000000	0.000000	0.149503	0.010903	0.000000	4	Kitchen & Dining	6
837	0.102116	0.005987	0.000000	0.000000	0.000000	0.000000	0.000000	0	Beauty and Personal Care	2
838	0.036471	0.000000	0.000000	0.000000	0.000000	0.111063	0.000000	5	Home Decor & Festive Needs	0
839	0.008186	0.000000	0.000000	0.203405	0.000000	0.000000	0.000000	3	Watches	5

840 rows × 10 columns

```
Topic 0:
com flipkart genuine cash ship delivery buy free product guarantee

Topic 1:
rockmantra 5 mug ceramic porcelain permanent thrill stay start dishwasher

Topic 2:
baby girl detail fabric cotton dress sleeve boy neck shirt

Topic 3:
watch analog men woman discount india great dial maximum strap

Topic 4:
mug coffee ceramic perfect tea printland gift one love get

Topic 5:
cm showpiece cover cushion best design 5 1 inch pack

Topic 6:
laptop skin shape mouse warranty pad print 6 inch battery
```

# Slides Bonus

```
class ImagePreprocessor(TransformerMixin, BaseEstimator):
    #Class Constructor
    def __init__( self, kernel=3, width=500, height=500, gaussormedian=True ):
        self.kernel = kernel
        self.width = width
        self.height = height
        self.gaussormedian = gaussormedian

    #Return self nothing else to do here
    def fit( self, X, y = None ):
        return self

    #Method that describes what we need this transformer to do
    def transform( self, X, y = None ):

        if self.gaussormedian == True:
            _X = [cv2.GaussianBlur(image,(self.kernel,self.kernel),0) for image in X]
        else :
            _X = [cv2.medianBlur(image,self.kernel) for image in X]

        _X = [cv2.equalizeHist(image) for image in _X]

        _X = [cv2.resize(image, dsize=(self.width,self.height)) for image in _X]

        return _X
```

```
parameters = {"imagepreprocessor__kernel": [3,5], #kernel to use
              "imagepreprocessor__gaussormedian": [True,False], #GaussianBlur or MedianBlur
              "imagestobovw__nb_keypoints": [500,800], #nb max of keypoints per image
              "imagestobovw__nb_visual_words": [20,30]} #nb of visual words to use for classification
#
# "randomforestclassifier__n_estimators": [100,150,200], #Number of tree to build
# "randomforestclassifier__max_depth": np.linspace(10, 110, num = 4), # Max depth of the tree
# "randomforestclassifier__min_samples_leaf": [1,3,5], #min amount of sample to have both left and right to a node
# "randomforestclassifier__max_features": ['auto', 'sqrt'] # additionnal source of random (max feature available
#                                     # when Looking at feature for best split)
#
# }

image_rfr_pipe = make_pipeline(ImagePreprocessor(width=500,
                                                height=500),
                              ImagesToBovw(),
                              RandomForestClassifier(n_estimators=150,
                                                    max_depth=50,
                                                    min_samples_leaf=1,
                                                    max_features='auto'))

image_rfr_grid = GridSearchCV(estimator = image_rfr_pipe,
                              param_grid = parameters,
                              scoring = 'accuracy' ,
                              cv=5,
                              verbose=1)

image_rfr_grid.fit(load_images(train['image']), train.category)

image_classifier = image_rfr_grid.best_estimator_
```

# Slides Bonus

1	-1	-1
-1	1	-1
-1	-1	1

$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9} = .55$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	-1
1	1	1
-1	1	1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

# Slides Bonus

## POOLING

Let's perform pooling with a window size 2 and a stride 2

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1			

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33		

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

# Slides Bonus

---

	Feature_name	Coef_feature
1	1	0.048518
0	0	0.039584
147	nlp_13	0.037804
3	3	0.031642
140	nlp_6	0.030465
169	nlp_35	0.028817
157	nlp_23	0.026933
141	nlp_7	0.026833
151	nlp_17	0.025324
181	nlp_47	0.024278
146	nlp_12	0.023659
2	2	0.021947
166	nlp_32	0.021480
170	nlp_36	0.020336
134	nlp_0	0.018634
174	nlp_40	0.018047
162	nlp_28	0.015868
178	nlp_44	0.015426
156	nlp_22	0.015299
177	nlp_43	0.015183
137	nlp_3	0.015080
4	4	0.014941
154	nlp_20	0.014690
138	nlp_4	0.014634
155	nlp_21	0.014589
173	nlp_39	0.014337
153	nlp_19	0.013295
167	nlp_33	0.013255
135	nlp_1	0.013156
136	nlp_2	0.012219
176	nlp_42	0.011861
179	nlp_45	0.011555
168	nlp_34	0.011450
161	nlp_27	0.011189