OpenClassRooms

Data Scientist

P6 Classification of consumer goods

Developped on a Notebook Jupyter Colaboratory

Pictures used for educational purpose only

Benoît DELORME
Creation : 28/06/2021
Update : 12/01/2022

Summary

I. Introduction

II. Image processing

III. Text processing

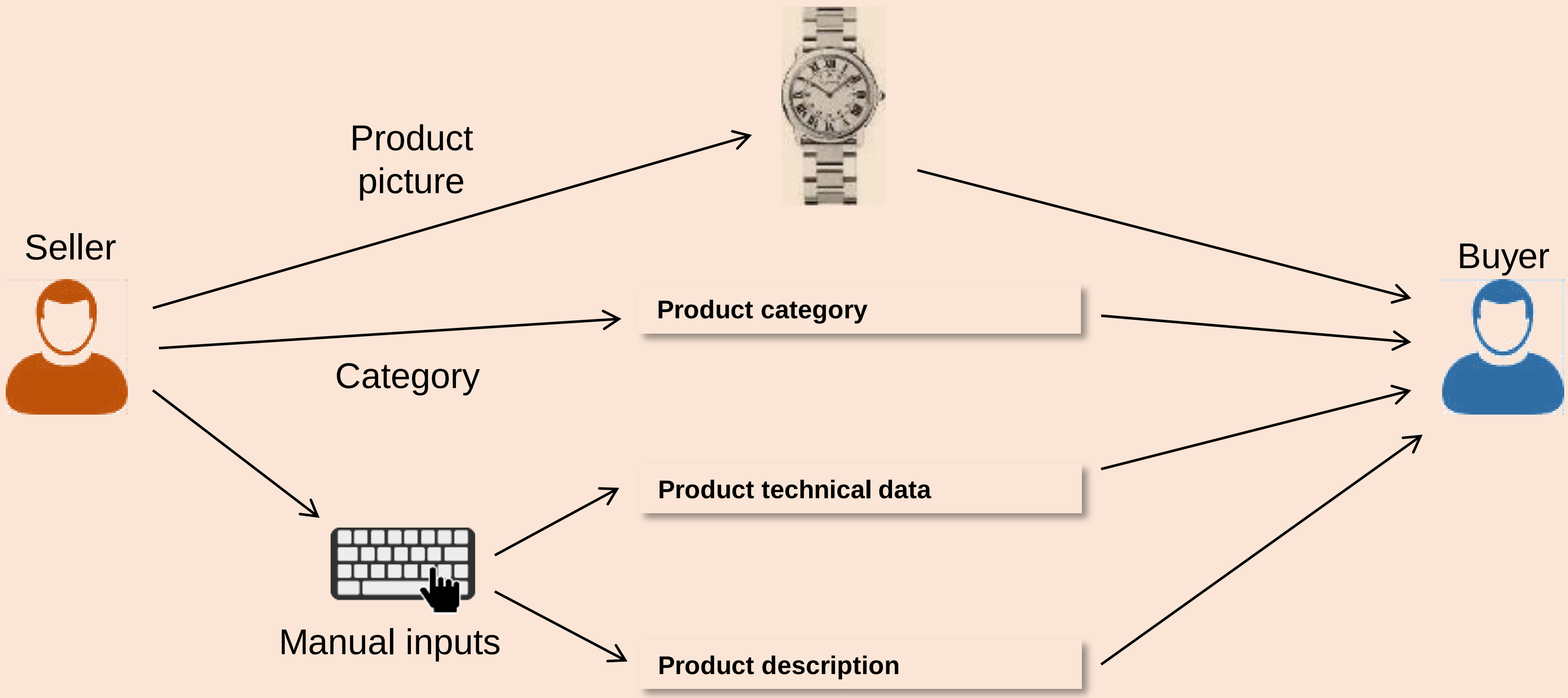
IV. Clustering

V. Conclusion

I. Introduction

1. The company and its needs

Company activity: an online marketplace



Problematic

Difficulties to guarantee the **manual inputs liabilities**

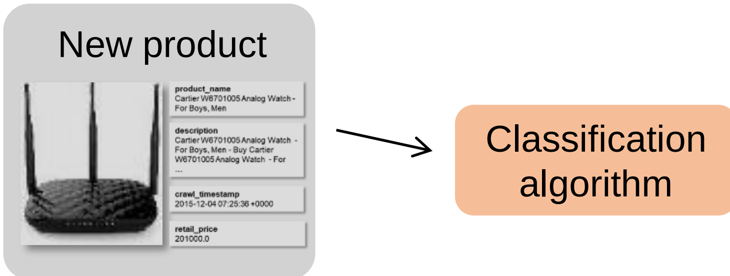
Project

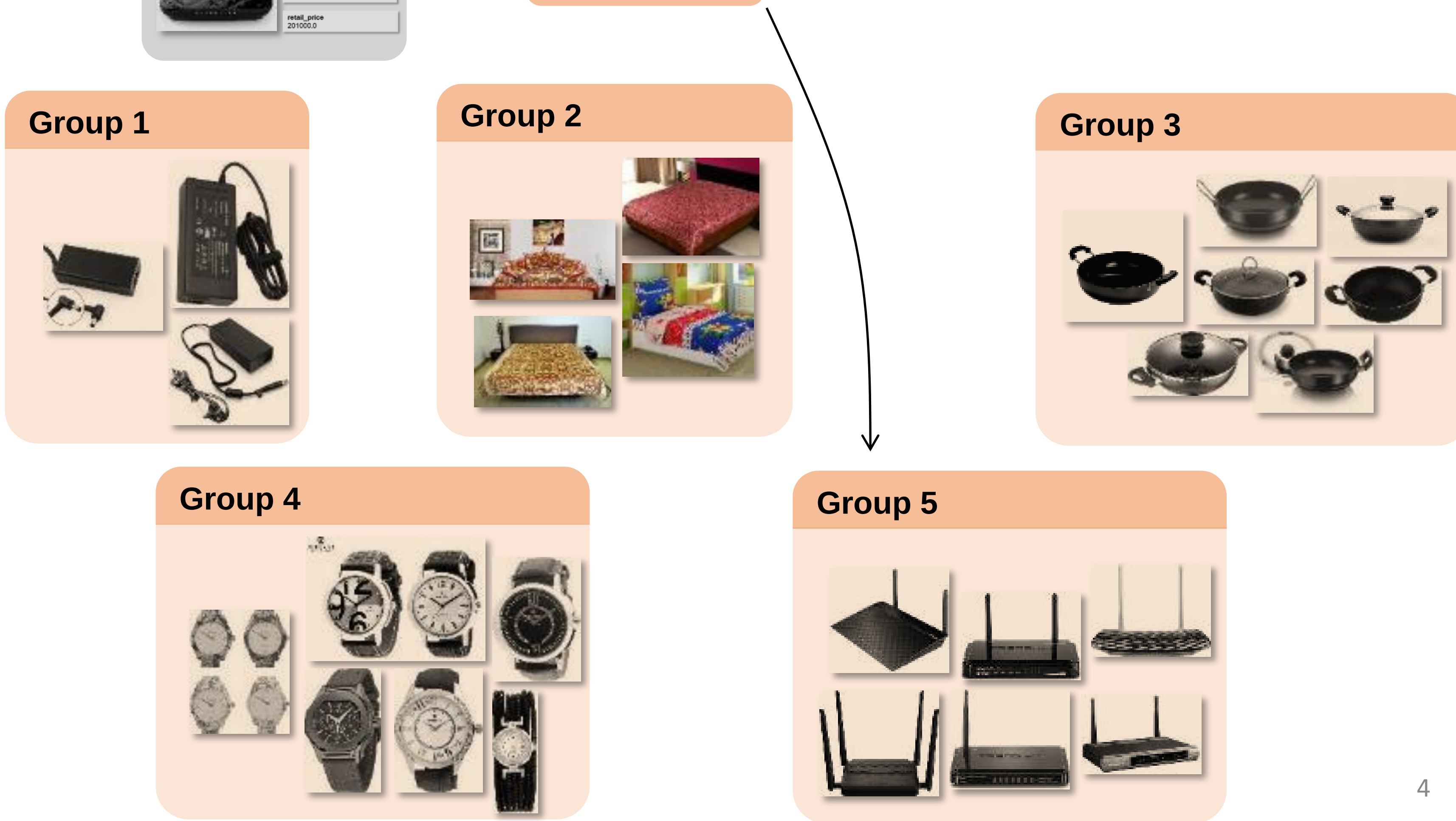
Feasibilty study of a product **classification**, with a sufficient accuracy level

I. Introduction

2. The deliverable

« *with a sufficient accuracy level* » :
→ set a **limit** to the number of **classification errors**



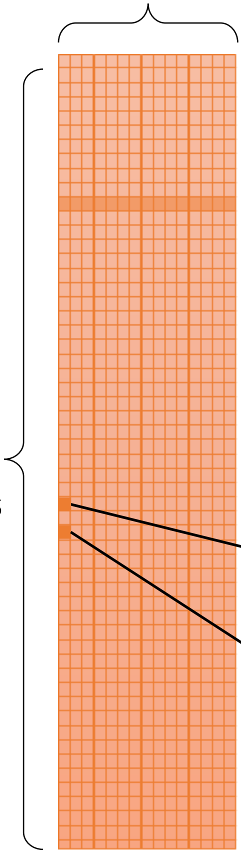


I. Introduction

3. The data set

15 features

1050 products



Main table

Dates, prices, descriptions, identifiers, brands, …

	crawl_timestamp	product_name	product_category_tree	retail_price	discounted_price	description	brand
0	2016-04-30 03:22:56 +0000	Elegance Polyester Multicolor Abstract Eyelet ...	["Home Furnishing >> Curtains & Accessories >>...	1899.0	899.0	Key Features of Elegance Polyester Multicolor ...	Elegance

Images

I. Introduction

4. Different types of data

Text data

product_name

description

product_category_tree
["Watches >> Wrist Watches >> Cartier Wrist Watches"]

product_specifications
{"product_specification"=>[{"key"=>"Chronograph", "value"=>"No"}, {"key"=>"Altimeter", "value"=>"No"}, ….

Numerical data

retail_price

Dates

crawl_timestamp

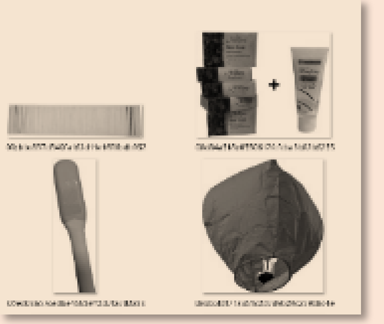
Boolean data

is_FK_Advantage_product

Identifiers

uniq_id

Images



Some text data have an internal structure, they can contain:

- A tree structure with branches
- A dictionnary, with specific keys for each product (similar to .json).

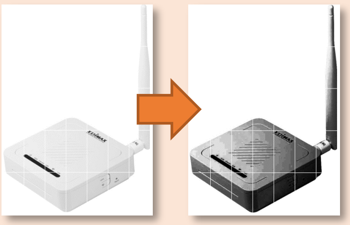
After a specific processing, these subelements will provide with new features.

I. Introduction

5. Available tools

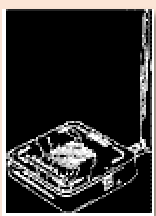
Image processing

Equalization

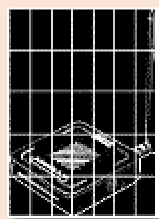


Gaussian filter

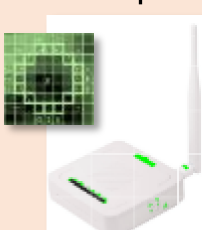
Canny filter



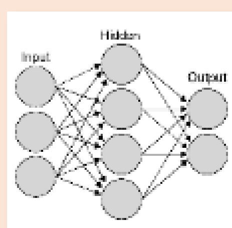
Gradient



Descriptors

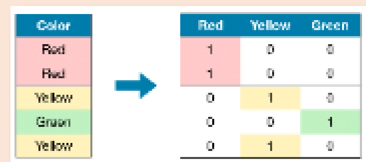


Neural network



Text processing

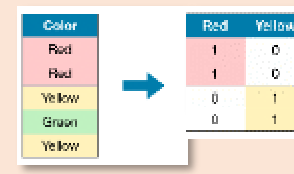
OneHotEncoding



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

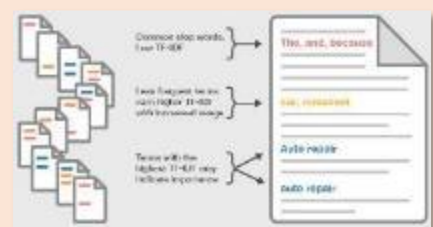
Specific encoding



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow
1	0
1	0
0	1
0	1

tf-idf weights



Objective: feed the classification algorithms k-means & DBSCAN

II. Images processing

1. Steps

Equalize

Principle : spread the distribution of grey levels.
Goal : be able to compare images.

Filter

Gauss | Canny | Gradient

Principle : soften the details and mitigate noise.
Goal : do not perturbate algorithms with noise.

Spot the keypoints

Principle : spot the points specific to the image.
Goal : transform the images to numerical data.

Classify

Descriptors | Coordinates

Principle : apply a classification algorithm to images.
Goal : classify the keypoints.

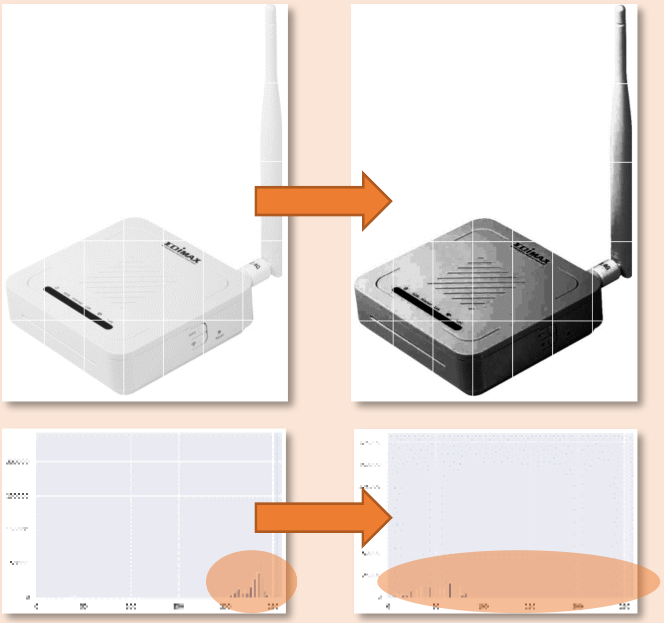
Verify

Principle : visually check the classification relevance.
Goal : refine the preceding steps, or conclude.

II. Images processing

2. Pre-processing

1. Equalizer



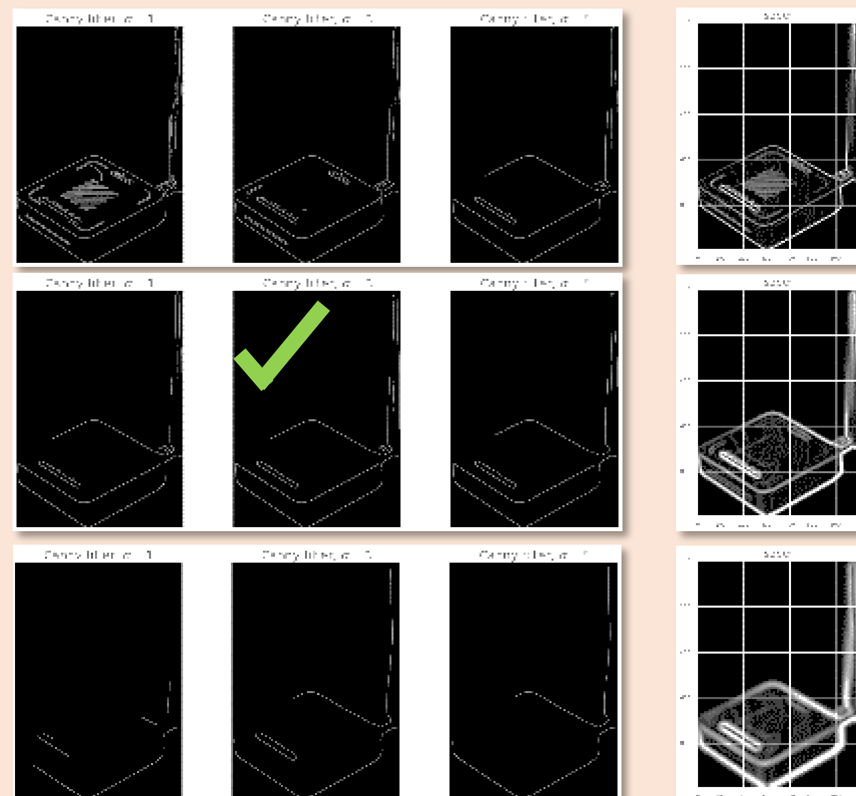
2. Filter

Gauss | Canny | Gradient

2

5

10



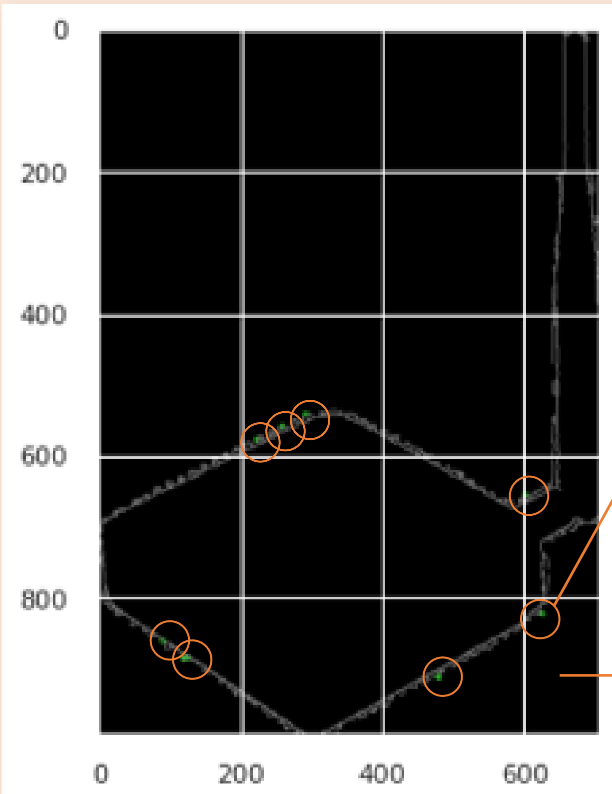
On several choosen images, results are compared accross different **Gaussian filters** and **Canny filters** →

→ A 5 Gaussian filter and a 3 Canny filter give the best compromise. This configuration will be used for the **whole data set**.

II. Images processing

3. Keypoints

Specific points spotted by ORB



Number of points is changeable

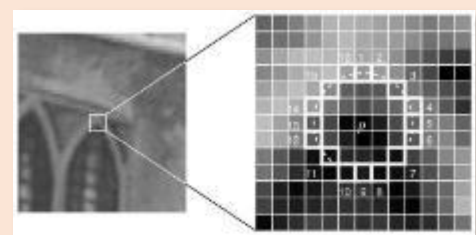
Keypoint coordinates

```
array([[318.9032 , 591.22504]], dtype=float32)
```

Keypoint **abscissa**   Keypoint **ordinate**

Keypoint descriptor

```
array([[  6,  50, 143, 130,  70, 115, 184,  72, 174, 245, 147, 255,  76,
        238, 127,  10, 190, 133, 235, 183, 149, 243,  28, 192,  24,  20,
        203, 199, 205,  39,  45, 212]], dtype=uint8)
```



Source
https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

→ Who provide with the best results: coordinates or descriptors?

II. Images processing

4. Classification

→ Around 22 groups identified visually

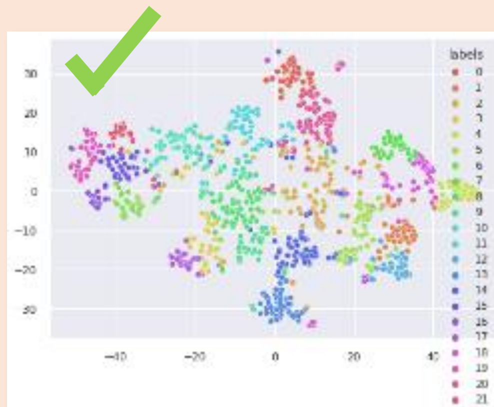
Classify the keypoints in 22 groups with k-mean

Represent in t-SNE plan

Count the number of labels per image

With coordinates

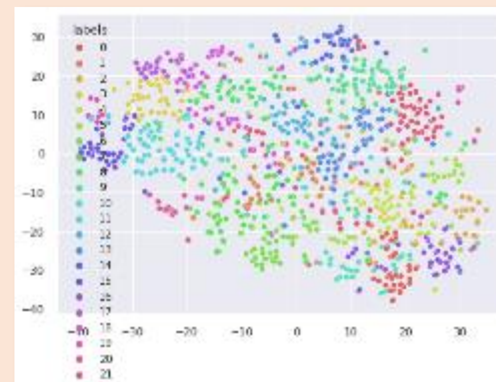
uniq_id	descriptor_id	keypoints_x	keypoints_y	labels
280	1.0	310.00000	519.00000	14
280	2.0	519.60004	304.80002	13
280	3.0	302.40002	613.44000	14
280	4.0	172.80002	559.87207	14
86	1.0	889.00000	182.00000	5
86	2.0	172.80000	687.60004	14
86	3.0	732.96002	694.08002	15
86	4.0	412.99203	765.50409	21



index	0	1	2	3	4	...	20	21
820	2.0	0.0	0.0	8.0	0.0	...	0.0	0.0
1003	0.0	0.0	0.0	8.0	0.0	...	0.0	0.0
162	0.0	0.0	0.0	7.0	0.0	...	1.0	0.0
200	1.0	0.0	0.0	5.0	0.0	...	0.0	0.0
137	0.0	0.0	0.0	4.0	0.0	...	2.0	0.0

With descriptors

uniq_id	descriptor_id	0	1	2	3	4	.	labels
280	1.0	165	252	156	97	56	.	17
280	2.0	88	32	19	32	1	.	11
280	3.0	43	173	2	186	89	.	20
280	4.0	96	48	40	104	40	.	19
86	1.0	64	50	128	101	96	.	10
86	2.0	48	32	26	48	9	.	11
86	3.0	65	20	20	69	48	.	13
86	4.0	36	96	177	64	128	.	6



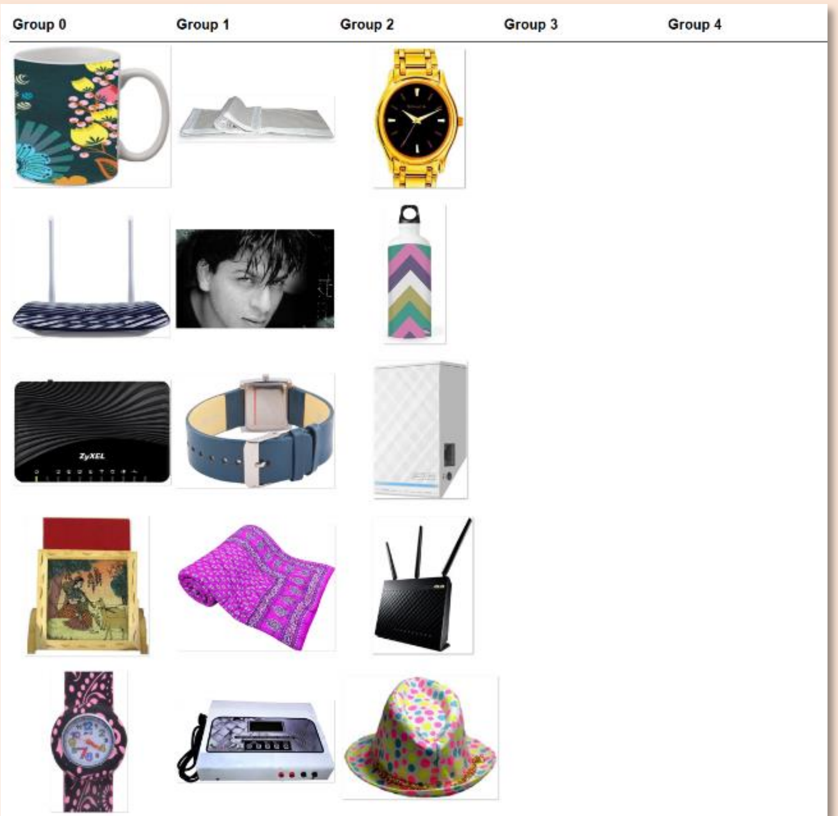
index	0	1	2	3	4	...	20	21
78	0.0	0.0	0.0	3.0	0.0	...	0.0	1.0
471	1.0	1.0	0.0	2.0	0.0	...	0.0	3.0
244	0.0	0.0	1.0	1.0	1.0	...	0.0	0.0
955	2.0	0.0	0.0	1.0	0.0	...	2.0	1.0
137	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0

Keypoints are better distinguished in t-SNE plan with regard to their **coordinates**

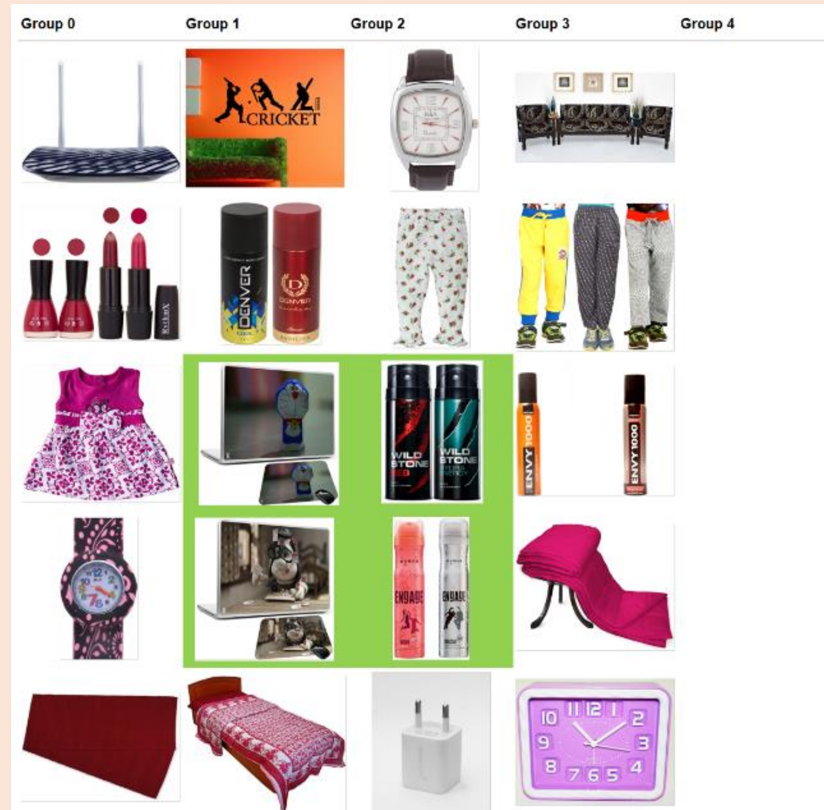
II. Images processing

5. Overlook of groups found

1. With coordinates



2. With descriptors



→ Groups quite inconsistent

III. Text processing

1. Steps

Extract

Principle: extract text features from raw data.
Goal: obtain information from obscure data types (json-like, multi-dictionnary, tree structure, …)

Clean

Principle: eradicate useless words, replace useful words by their roots
Goal: do not perturbate algorithms with noise.

Encode

Principle: transform textual data into numerical data.
Goal: be able to run an algorithm on a numerical table.

Verify

Principle: verify visually the results consistency.
Goal: refine preceding steps, or conclude.

III. Text processing

2. Extract

→ Tree structure

product_category_tree
["Home Furnishing >> Curtains & Accessories >> Curtains >> …

2 first branches

cat_tree_depth	cat_tree_1st	cat_tree_2nd
4	Home Furnishing	Curtains & Accessories
5	Baby Care	Baby Bath & Skin

→ Dictionary (similar to json)

product_specifications
{"product_specification"=>[{"key"=>"Brand", "value"=>"Elegance"}, ….
{"key"=>"Type", "value"=>"Eyelet"},
….

> 2% of dataset

spec_type	spec_brand	spec_sales_package
Eyelet	Elegance	2 Curtains
Bath Towel	Sathiyas	3 Bath Towel

Clean

description
Cartier W6701005 Analog Watch - For Boys, Men - Buy Cartier W6701005 Analog Watch - For
…

product_name
Cartier W6701005 Analog Watch - For Boys, Men

Brand
Cartier

Delete (too many unique values, representing noise for algorithms)

III. Text processing

3. Clean

Raw sentence	XYZ Designed For Computers, Laptops, Mobiles Model Number SF-VC_011 Model Name Vacuum Cleaner In the Box Sales Package 1 vacuum cleaner
Without **numbers**	XYZ Designed For Computers, Laptops, Mobiles Model Number Model Name Vacuum Cleaner In the Box Sales Package vacuum cleaner
Without **punctuation**	XYZ Designed For Computers Laptops Mobiles Model Number Model Name Vacuum Cleaner In the Box Sales Package vacuum cleaner
Without **stopwords**	XYZ Designed Computers Laptops Mobiles Model Number Model Name Vacuum Cleaner Box Sales Package vacuum cleaner
Without words of **1 ou 2 letters**	XYZ Designed Computers Laptops Mobiles Model Number Model Name Vacuum Cleaner Box Sales Package vacuum cleaner
Roots only	xyz design comput laptop mobil model number model name vacuum cleaner box sale packag vacuum cleaner

III. Text processing

4. Encoder

OneHotEncoding

→ One **category** per cell

cat_tree_1st
home furnish
babi care

↓

	cat1_home_furnish	cat1_babi_care
	0.0	0.0
	1.0	0.0

Specific encoding

→ **Several** distinct words per cell

spec_type	spec_brand	spec_sales_package
eyelet	eleg	curtain
bath towel	sathiya	bath towel
flat	jaipur print	bed sheet pillow cover

↓

spec_type_eyelet	spec_type_towel	spec_type_bath
1	0	0
0	1	1

tf-idf weights

→ A **significant number** of distinct words per cell

description
key featur eleg polyest multicolor abstract ey...
specif sathiya cotton bath towel bath towel re...

product_name
eleg polyest multicolor abstract eyelet door c...
sathiya cotton bath towel

↓

desc_key	desc_featur	desc_eleg
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

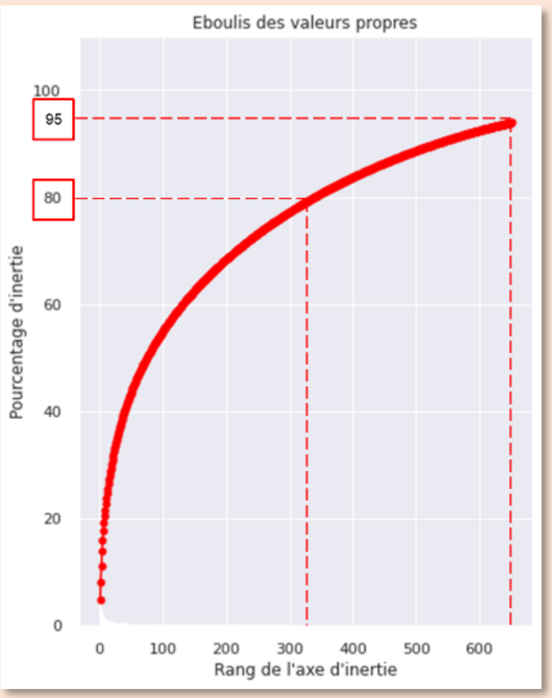
name_eleg	name_polyest	name_multicolor
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

IV. Classification

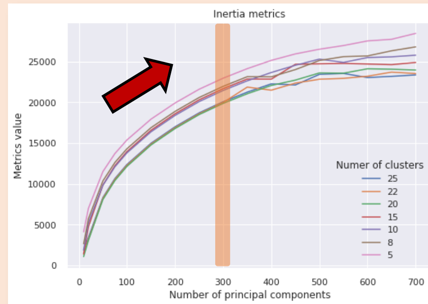
1. Dimension reduction

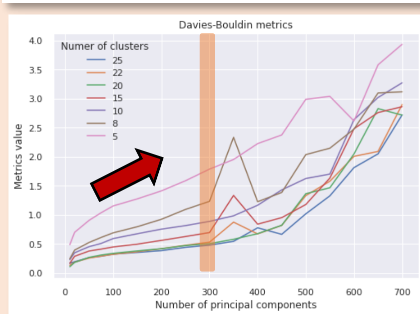
Usual approach

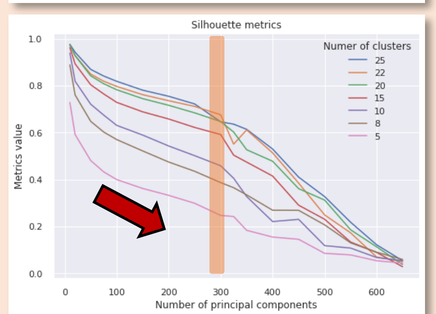
Keep 80% or 95% of the most important components:



k-means

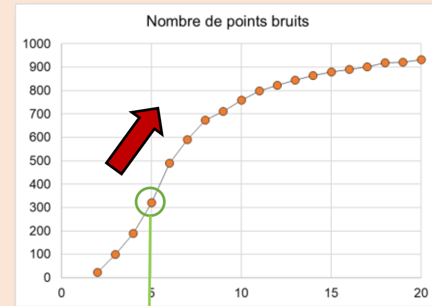




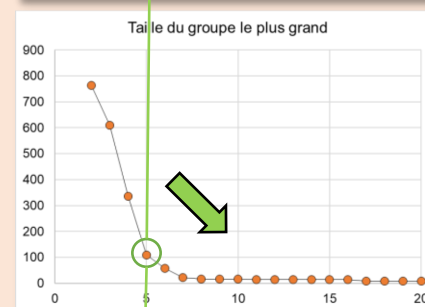


DBSCAN

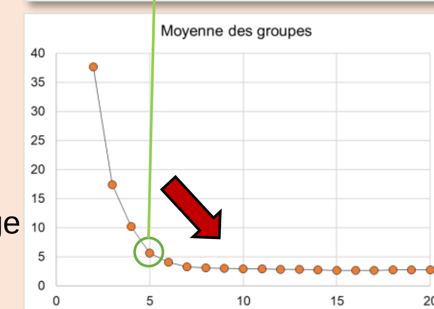
Noise points



Bigger group size



Group size average

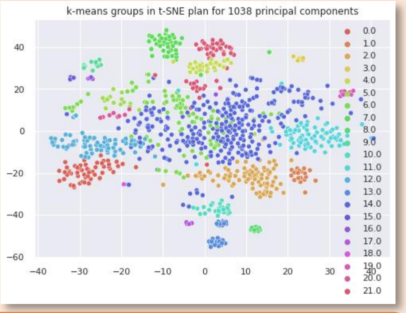
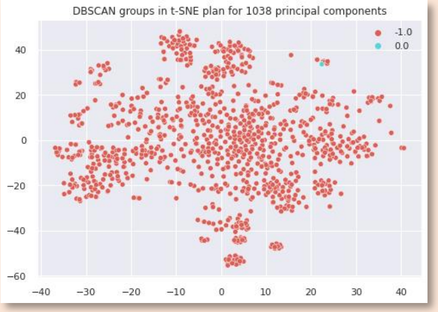
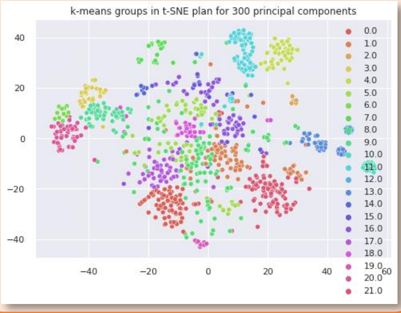
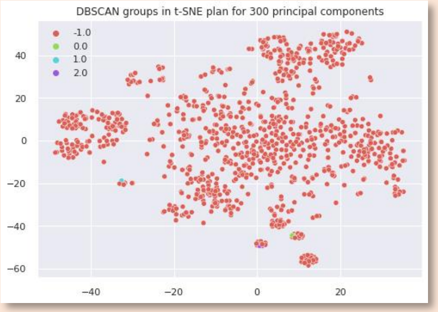
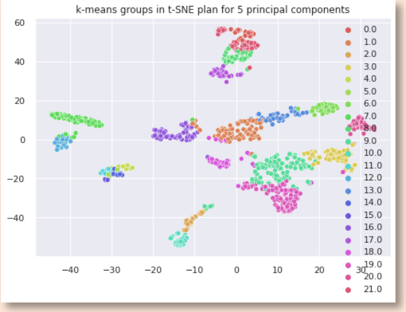
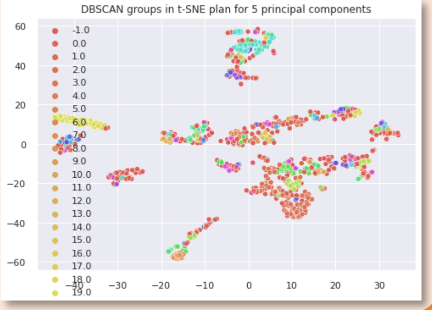


→ For k-means: best is to keep 300 principal components;

→ For DBSCAN, best compromise is to keep 5 principal components.

IV. Classification

2. Visualisation in t-SNE plan

	k-means	DBSCAN
1038 components (total number of products)		
300 components		
5 components		

→ Consistency between t-SNE and k-means.

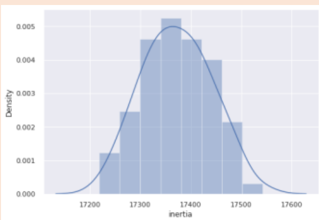
→ DBSCAN best performances are insufficient (≈ 100 groups)

IV. Classification

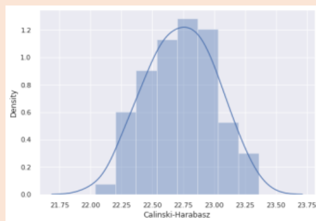
3. Optimization

k-means

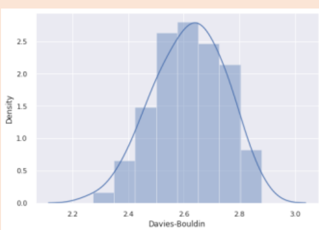
```python
# Hyperparameters
n_init_list = [10, 15, 20, 30]
max_iter_list = [300, 400, 500, 600]
tol_list = [0.0001, 0.0003, 0.0005, 0.0008, 0.001]
```



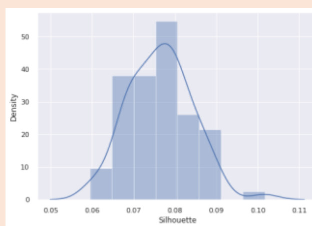
Inertie



Calinski-Harabasz



Davies-Bouldin



Silhouette

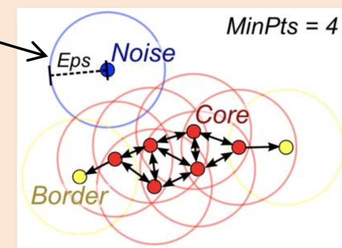
→ Similar metrics whatever the combination.

→ No combination gives significantly better results than the others.

DBSCAN

```python
# Hyperparameters
eps_list = [0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
min_samples_list = [2, 3, 4, 5]
leaf_size_list = [10, 20, 30]
```

→ Best scores for *eps* = **0,1.**



Reference value for eps is **0,2**.

An eps = 0,1 will make the groups more **compact**.

But more points will be considered as **noise.**

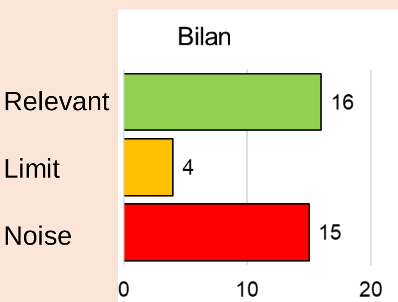
→ reference values for k-means
→ eps = 0,1 for DBSCAN

IV. Classification

4. The deliverable

→ Study of 7 groups, with 5 images per group

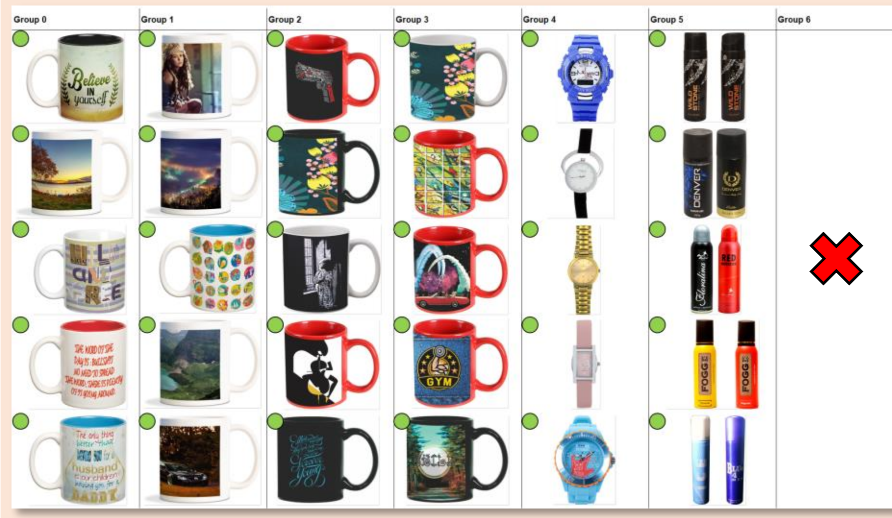
k-means

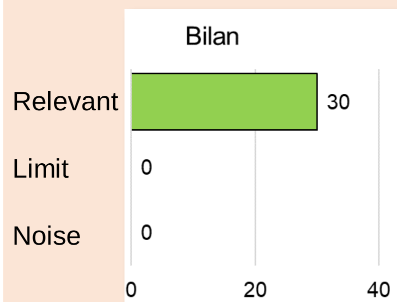


→ Accuracy : 46%

→ Similarities between groups (i.e. group 0 and group 6)

DBSCAN





→ Accuracy : 100%

But :

1. 95,6 % of dataset considered as noise, thus unusable.
2. Identical groups.

→ k-means, although not perfect, is more adapted to this dataset.

V. Conclusion

Dataset

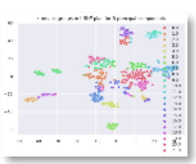
• **Not much data** (1050 products), but on the whole, the features are workable.
• Dataset is small and prevents algorithms to give good performances.

Images

• For some images, ORB algorithme finds **relevant keypoints.**
• For other images, ORB seems to be perturbated by **noise, hardly softened by the filters.**

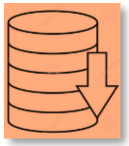
Texts

• Text data are workable**.**
• The result after cleaning is relevant.



Classification

• **Dimensional reduction** differs from an algorithm to the other.
• **k-means** provides with the best results, with **46%** accuracy on a sample of images.
• **DBSCAN** does not seem to be adapted to this dataset.

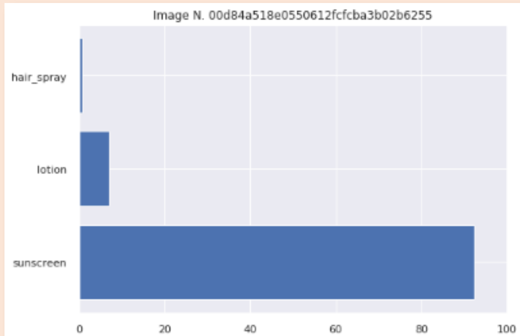


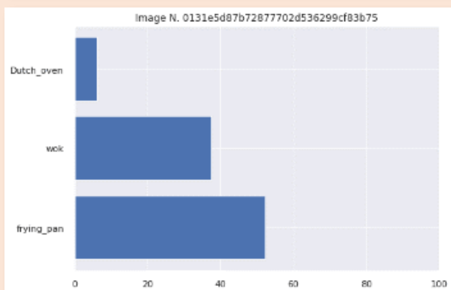
Outlooks

• **Enrich** the dataset with other products.
• **SIFT** could not be used, but seems to be more accessible in its paid version (opencv_contrib).

Improvements

Neural network VGG16 (Keras)





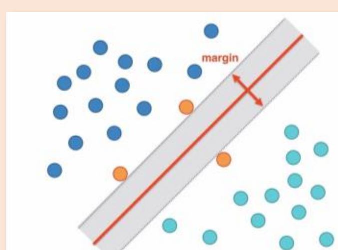
Specific processing for difficult images

Abundance of details

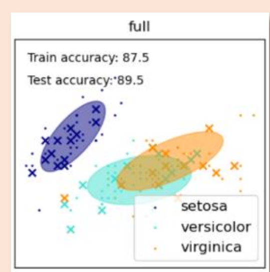
Floral patterns

Shiny objects

Other classification algorithms



SVM



Gaussian Mixture

End of the presentation

Thank you for your attention