Enhancing E-commerce: “Multimodal Product Data Classification”

Data Exploration & Modeling Report

August 2023

Ramiro H. Rodriguez

linkedin.com/in/ramiro-hernan-rodriguez-b0a1191bb

June23_bds

DataScientest

Project description

Effectively categorizing products based on a multitude of data is pivotal within the dynamic e-commerce domain. It plays a multifaceted role, from refining product recommendations and enabling personalized exploration to assisting customers in pinpointing precisely what aligns with their preferences. At the core of this project lies the challenge to craft a robust model capable of classifying products into 27 distinct categories. This is achieved by meticulously analyzing not only product images but also the accompanying textual descriptions, merging the visual and semantic facets of product understanding into a unified classification framework.

Data source

The datasets have been sourced from the Rakuten France Multimodal Product Data Classification challenge, accessible at https://challengedata.ens.fr/challenges/35.

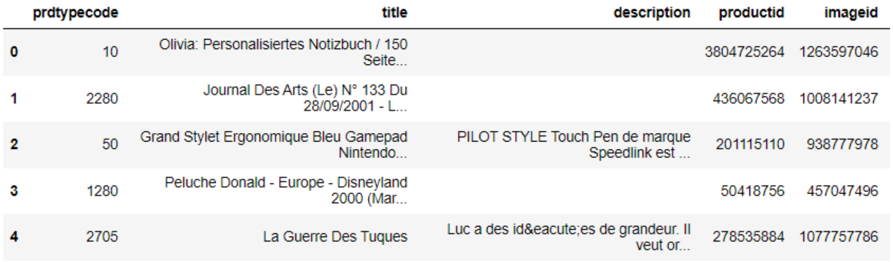
This dataset comprises a CSV file documenting around 99,000 products, offering "text data" for each item. This textual information includes a product 'title' (or 'designation') and a corresponding 'description'. Each of the 27 categories is denoted by a numerical 'product type code'. Additionally, the dataset encompasses a set of images, one for each product, mirroring the visuals displayed on the e-commerce platform alongside the product description.

In total, the dataset encompasses approximately **60 MB of text data** and **2.2 GB of image data**.

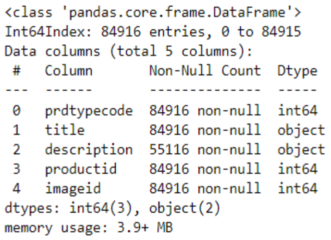
Text data exploration

Dataset description

The text data is consolidated in a dataframe listing **84916** products (rows) for which 5 variables (columns) are provided, as illustrated in the left table below, showcasing five exemplary instances. On the right, the accompanying table offers a comprehensive overview of this data, providing details on the data type of each column and the extent of missing values.



	prdtypecode	title	description	productid	imageid
0	10	Olivia: Personalisiertes Notizbuch / 150 Seite...		3804725264	1263597046
1	2280	Journal Des Arts (Le) N° 133 Du 28/09/2001 - L...		436067568	1008141237
2	50	Grand Stylet Ergonomique Bleu Gamepad Nintendo...	PILOT STYLE Touch Pen de marque Speedlink est ...	201115110	938777978
3	1280	Peluche Donald - Europe - Disneyland 2000 (Mar...		50418756	457047496
4	2705	La Guerre Des Tuques	Luc a des id&eacute;es de grandeur. Il veut or...	278535884	1077757786



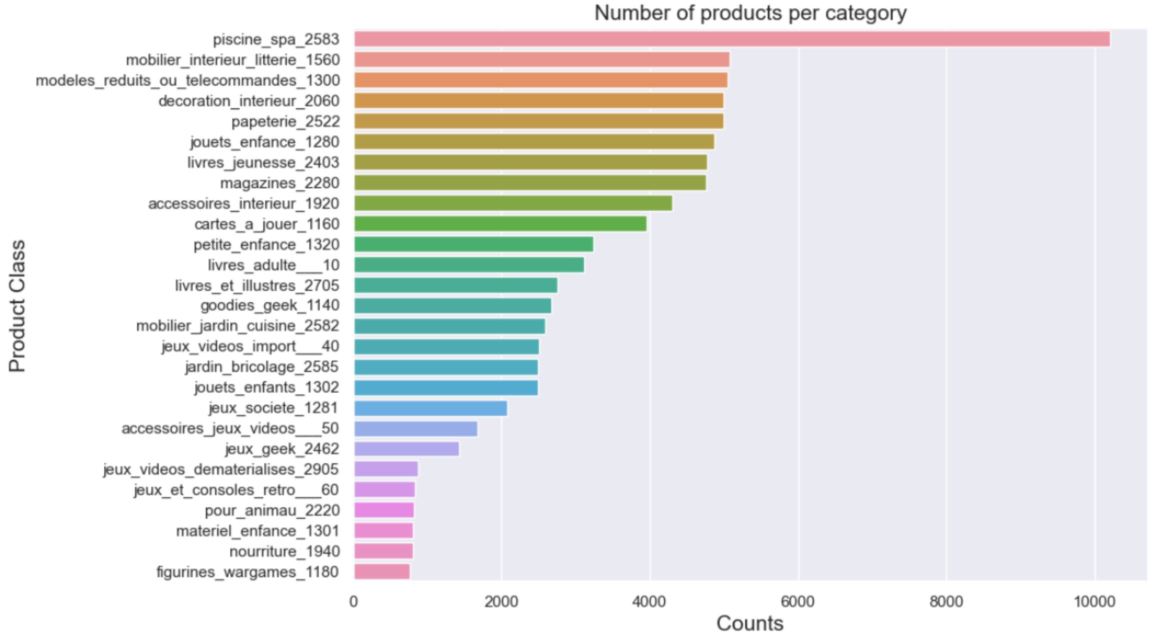
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84916 entries, 0 to 84915
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   prdtypecode  84916 non-null  int64 
 1   title        84916 non-null  object
 2   description  55116 non-null  object
 3   productid    84916 non-null  int64 
 4   imageid      84916 non-null  int64 
dtypes: int64(3), object(2)
memory usage: 3.9+ MB
```

The columns represents:

- **prdtypecode** - The product type code signifying the respective product class.
- **title** - A concise text that encapsulates the essence of the product description.
- **description** - A more detailed textual description of the product and its characteristics.
- **productid** - A unique identifier designated to each product.
- **imageid** - A unique identifier for the product’s associated image.

The fields *imageid* and *productid* are important for retrieving the images from the directory containing the entire image collection. To be precise, the image file name for a given product follows the format: ‘*image_imageid_product_productid.jpg*’.

There is a total of 27 classes of product, each one assigned to a value of ‘*prdtypecode*’. This is the target variable that the model aims to predict based on the textual and image data. It is important to note that all the classes are not equally represented. The bar plot below shows the number of products in each category. Notably, the most predominant class being the category ‘*piscine_spa*’ (code number ‘2583’), with 10.2k items, which accounts for almost **12%** of the dataset.



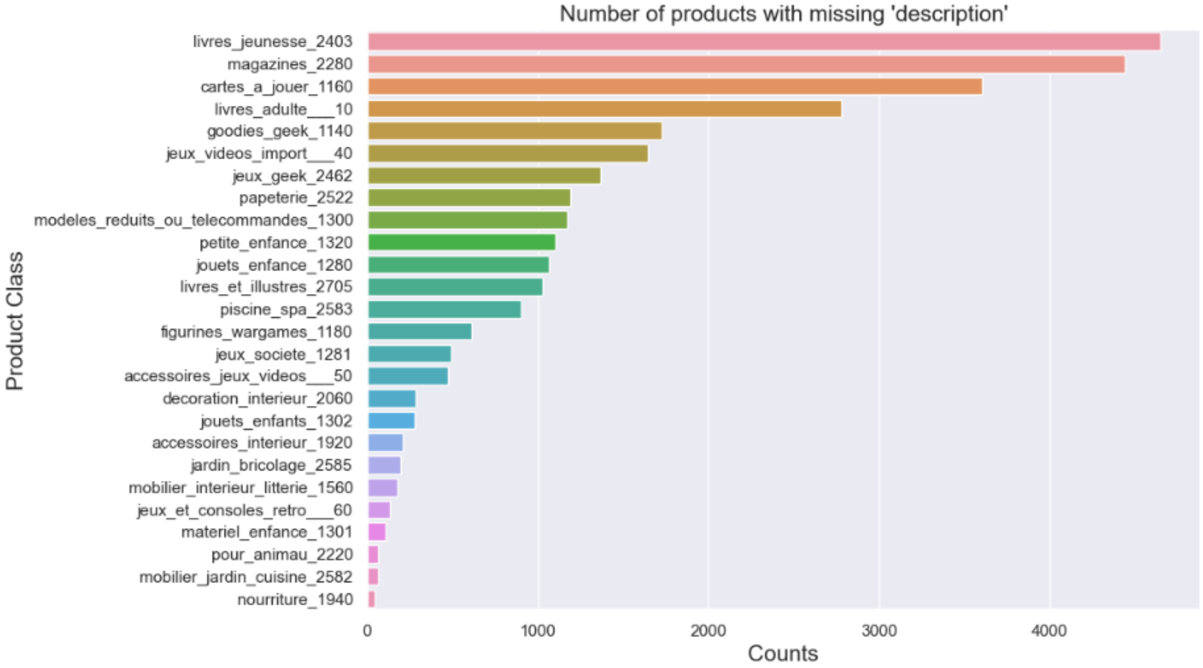
Missing values

The only feature exhibiting instances of missing values is the ‘description’ attribute. Within the dataset, a total of **29800** items lacks a 'description', constituting **35%** of the whole catalogue. This isn't entirely surprising since not all vendors utilize this feature. To retain originality of the data, these absent descriptions have been replaced by NaN values.

To visualize this aspect, the graph below shows the distribution of product classes affected by missing descriptions. The y-axis corresponds to the product classes, while the x-axis displays the count of products lacking a description within each respective category.

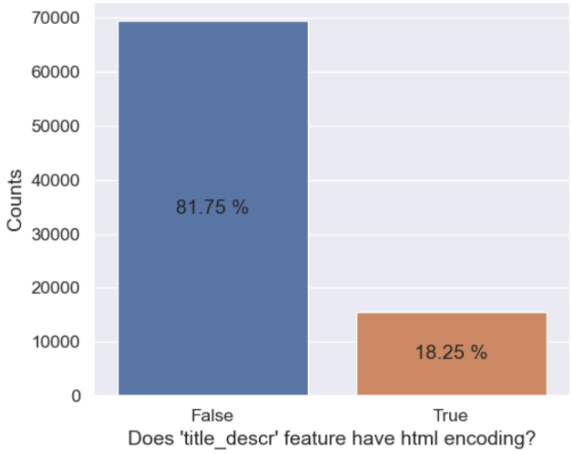
Feature engineering: title + description:

Addressing the potential absence of values in the 'description' field, a strategic approach is employed through the creation of a new variable named '*title_descr*'. This variable merges the textual content from both the 'title' and 'description' variables by concatenation. Any instances of NaN values are seamlessly substituted with an empty string, ' '.



HTML parsing

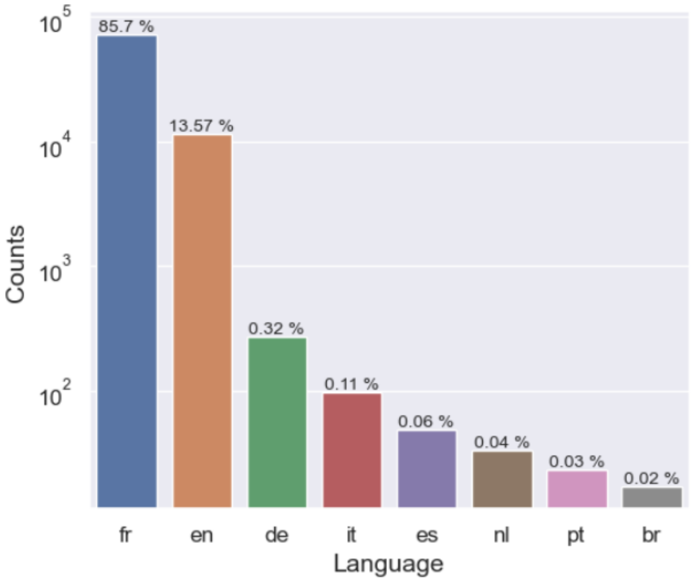
A closer examination of some values in the variable '*title_descr*' shows that in some cases the text is HTML encoded, thus containing some of the most common tags such as '`<br>`' and '`<h1>`' and similar text formatting elements, along with encoded representations of distinct French characters (e.g., '`&eacute;`' for '**é**', '`&ecirc;`' for '**ê**', among others). The figure below shows that more than **18%** of the dataset's items exhibit some degree of html encoding. This observation appears reasonable since some of the text data might have been obtained through web-scraping the e-commerce platform.



Considering this observation, the '**title_descr**' variable undergoes an essential transformation process. This involves employing commonly used libraries like *BeautifulSoup* to decode the HTML, thereby enhancing the text data's readability and usability.

Language detection

Most of the content within the 'title_descr' variable is composed in the French language, but there exists a portion of items with descriptions in English. To be more quantitative, the bar-plot below displays the distribution of languages detected using the language identifier from the *langid* library. Note that the y-axis is in logarithmic scale, hence 99.28% of the dataset is either in French or in English. Nevertheless, other languages were also present as shown in the plot. Importantly, only detection cases with a confidence larger than 0.9999 were retained in a first detection, while low confidence detections were reclassified in a second iteration either as French or English for simplicity. A total of ~7476 cases of low confidence detection, were reclassified as French, while ~6498 cases were reclassified as English.

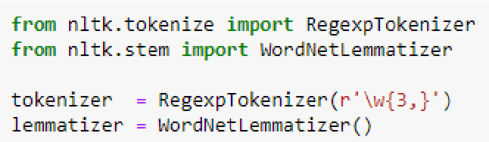


Language code ISO 639-1

fr	French
en	English
de	German
it	Italian
es	Spanish
nl	Dutch
lb	Luxembourgish
pt	Portuguese

Lemmatization and Tokenization

Once all the available textual data for each product, gathered in a single variable 'title_descr', has been properly html-decoded and its language has been correctly detected, it can be further analysed using common techniques of the natural language processing (NLP) fields. Lemmatization and Tokenization are two techniques that allows to transform raw text into structured, normalized, and semantically meaningful representations. They enable to increase the information density of a textual data and to extract useful additional features helping machine learning algorithms learn its patterns and make accurate predictions from the textual data. These processes are achieved by calling the respective objects from the ***nltk*** library:



```
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer

tokenizer  = RegexpTokenizer(r'\w{3,}')
lemmatizer = WordNetLemmatizer()
```

Text Segmentation

For each product in the catalogue, the variable '*title_descr*' is first tokenized: the text data is broken down into individual words (tokens) and only tokens of three or more consecutive word characters are retained. Short tokens like "le", "et", "un", "is", "an" are discarded.

Text Normalization: Improved semantics

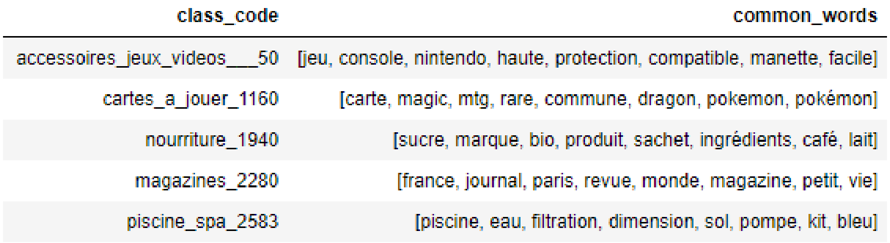
Lemmatization allows to reduce tokens to their base or root form (lemma) making it easier to identify the core meaning of different inflections of the word. By this one can reduce the dimensionality of the data and control the size of the feature space. It will also help models to generalize better, as they can learn patterns from a broader context of related words rather than memorizing different inflections of the same words.

Stop Words Removal

Stop words like "the" & "and" in English, or "avec" & "des" in French are removed from the lemmatized token lists, since they don't carry significant meaning and can introduce noise into the analysis. Since the language has been detected for each product's textual information, one can remove the stop words belonging to the corresponding language.

Most common words

After tokenization, lemmatization and stop words removal, the obtained structured text for each product is stored in a new '*lemma_tokens*' variable. From here we can extract the most common words per category, as displayed in the table below for 5 distinct categories. Notice that the set of most common words of one category are in some degree closely related from one another but they are much less related to the common words of other categories ('nourriture' vs 'magazines' for example). This observation is indicating that certain tokens might be key in determining a product category from the textual data.



class_code	common_words
accessoires_jeux_videos___50	[jeu, console, nintendo, haute, protection, compatible, manette, facile]
cartes_a_jouer_1160	[carte, magic, mtg, rare, commune, dragon, pokemon, pokémon]
nourriture_1940	[sucre, marque, bio, produit, sachet, ingrédients, café, lait]
magazines_2280	[france, journal, paris, revue, monde, magazine, petit, vie]
piscine_spa_2583	[piscine, eau, filtration, dimension, sol, pompe, kit, bleu]

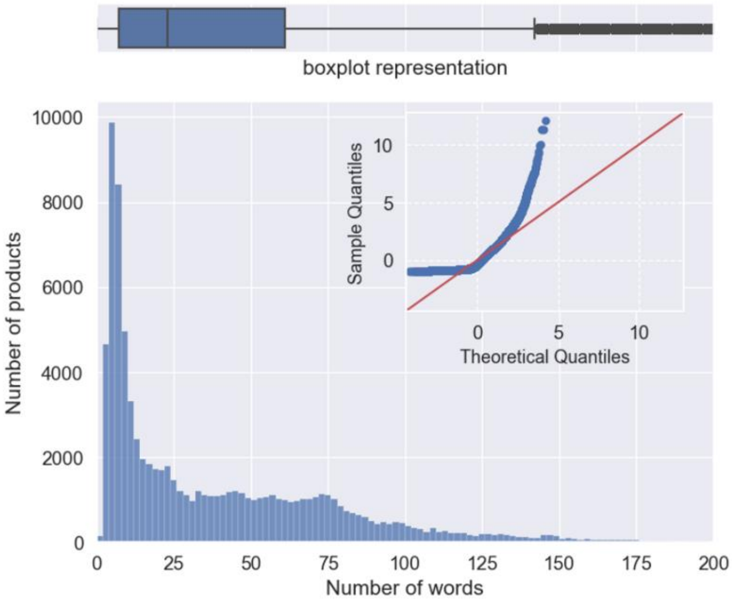
Feature extraction: token-length:

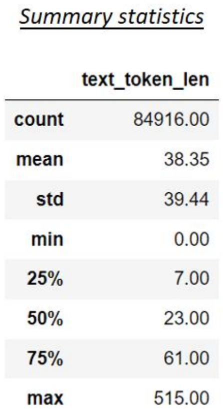
The number of tokens in the description of a product is also an interesting feature to look because certain types of products are naturally easier to describe than other types. For example, it seems reasonable to think that electronic products from the category '*modeles_reduits_ou_telecommandes*' (scale_models_and_remote_controls) have longer descriptions because they usually come with various features, technical specifications, and functionalities that need to be adequately described. On the other hand, products from the category '*papeterie*' (stationery) such as pens, eraser and rulers probably have a short description since it generally focuses on basic features like brand, colour and size.

With that in mind the total number of tokens in the ‘lemma_tokens’ list is stored in a new variable ‘*text_token_len*’. Note that the html parsing also helps to reduce the dimensionality of the data before the tokenization since tag-words have been removed and French words with special characters and accents are now correctly identified.

The figure below displays the statistical distribution of the variable ‘*text_token_len*’. The main plot is the histogram of this variable, i.e., the number of catalogue items having x number of tokens on their textual data. The table to the right lists the main statistical indicators, some of which are displayed in the bar-plot representation (upper plot window). Notice that the x-axis of the main plot and the bar-plot has been limited at 200 words, since extreme values are observed with products having up to 515 tokens in their description. Nevertheless, most of the products tends to have a description composed of less than 75 lemmatized tokens.

The histogram shows that the distribution is asymmetric, with the *mean* being 1.67 times the *median.* Although there is no reason a priori to expect the variable ‘text_token_len’ to be normally distributed, a numerical confirmation can be obtained by displaying the Quantile-Quantile plot (inset plot), on which if the distribution is normal, then it should not deviate from the red diagonal line.



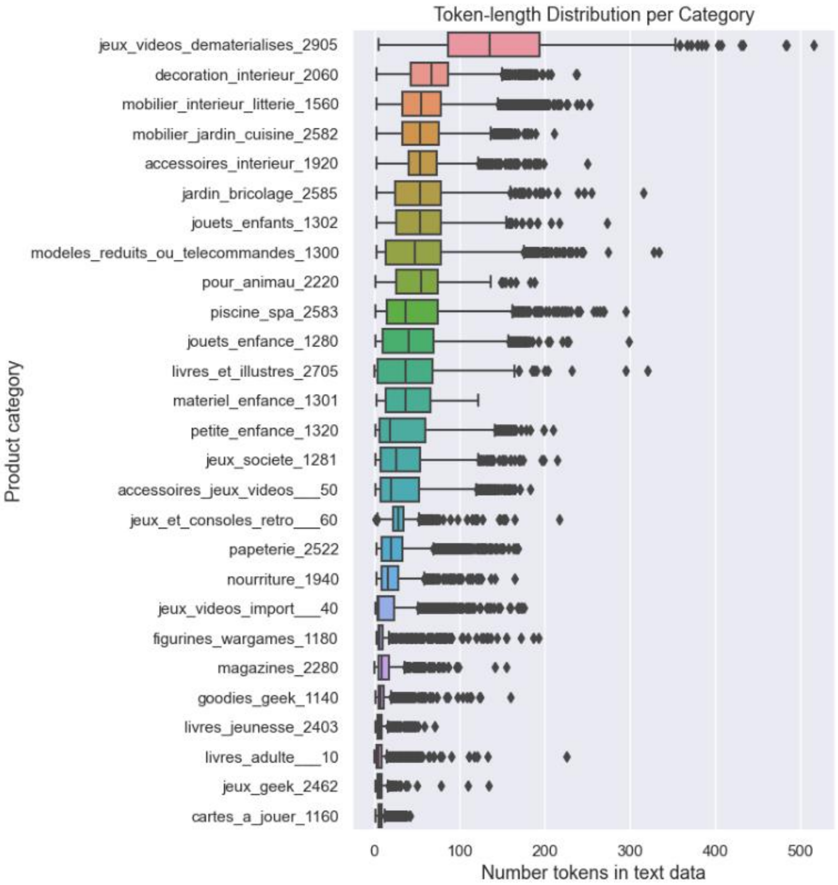

Summary statistics

	text_token_len
count	84916.00
mean	38.35
std	39.44
min	0.00
25%	7.00
50%	23.00
75%	61.00
max	515.00

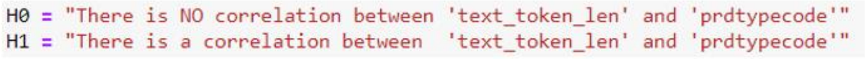
Correlations

As mentioned, we could expect to find some degree of correlation between the number of words in the text information describing a product and the class of product.

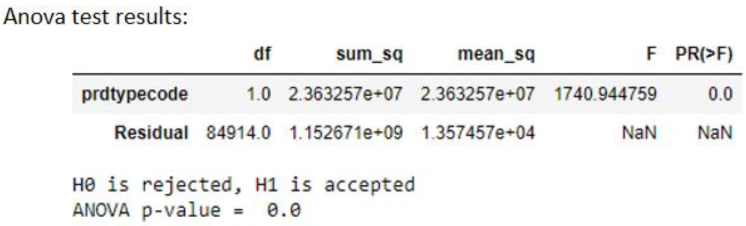
Below is a first visualization of the ‘*text_token_len*’ (the number of words) distributions ***per product category*** (*prdtypecode*), ordered by decreasing *mean* value. The box plot shows that each category has a somewhat ‘unique’ distribution. For example, the category ‘*jeux_videos_dematerialises*’ has a distribution characterized by a very unique set of statistical indicators **[min, q1, q2, mean, q3, max]** which could potentially help in determining the probability for a product of belonging to one category or another.



An ANOVA test between the variables ‘text_token_len’ (numerical) and ‘prdtypecode’ (categorical) confirms the existence of a correlation:



```
H0 = "There is NO correlation between 'text_token_len' and 'prdtypecode'"
H1 = "There is a correlation between  'text_token_len' and 'prdtypecode'"
```



Anova test results:

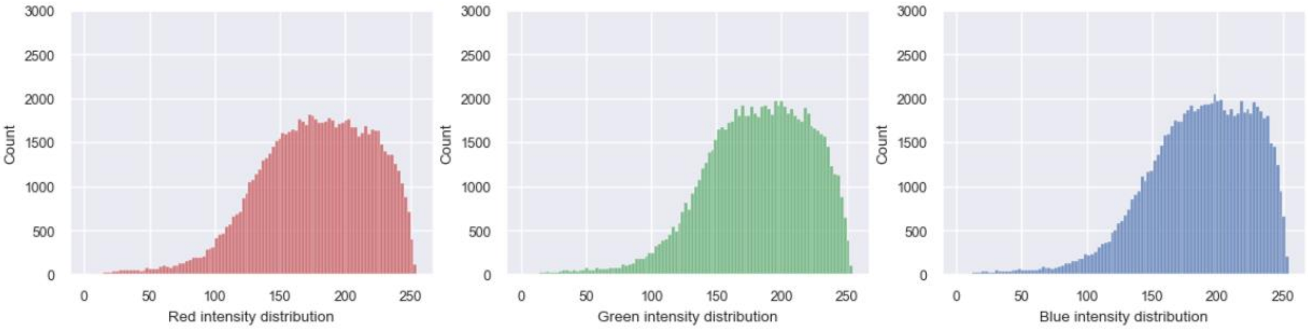
	df	sum_sq	mean_sq	F	PR(>F)
prdtypecode	1.0	2.363257e+07	2.363257e+07	1740.944759	0.0
Residual	84914.0	1.152671e+09	1.357457e+04	NaN	NaN

```
H0 is rejected, H1 is accepted
ANOVA p-value =  0.0
```

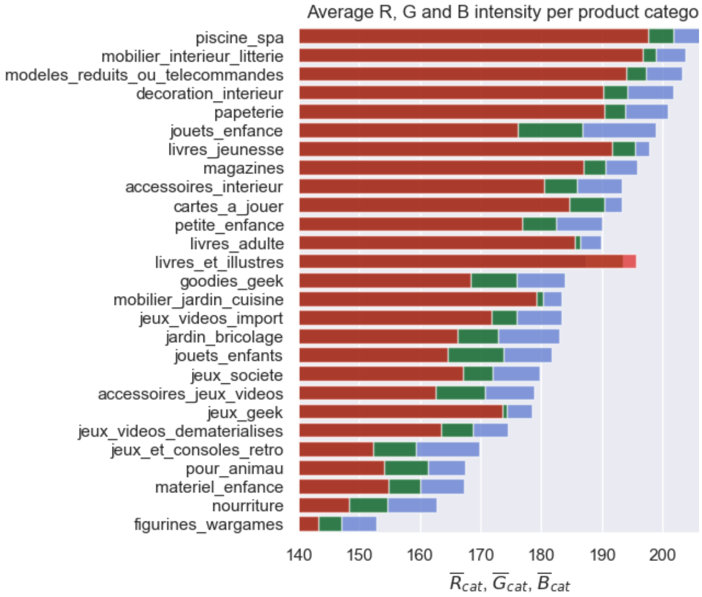
Image data exploration

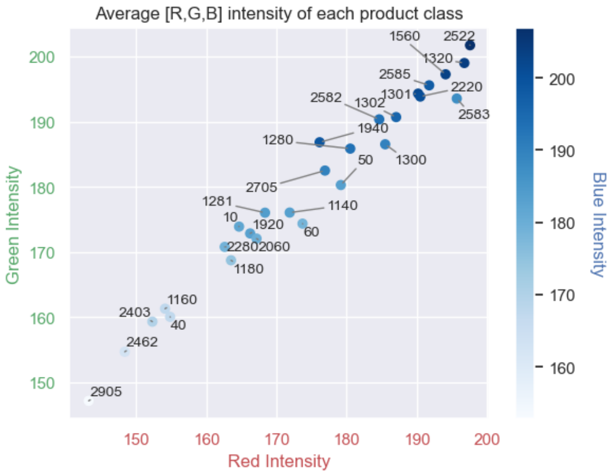
Images are provided with size 500 x 500 pixel, each pixel having the 3 usual RGB channels. After displaying some images one can notice that many items in the catalogue have a considerable amount of white background in the image as shown by the three examples below. To avoid any bias in our classification model and to increase the information density it is better to crop the images as to the smallest square that fits the entire object. This step is part of the preprocessing stage of image data.

Then, for each image we can obtain the average intensity on each channel, resulting in a vector [avgR, avgG, avgB] for each product in the catalogue. Below we can see the average intensity distribution of red (R), green (G) and blue (B) in the whole dataset.

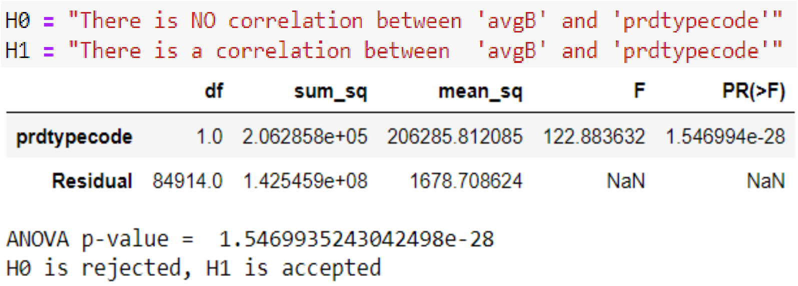


These distributions are skewed toward the maximum pixel intensity of 250 which can be interpreted either as the dataset having many ‘bright’ objects or as it still having many images with a considerable amount of white background. Although the latter can be a source of bias, by cropping the images it has been minimized. Moreover, the amount of white background on the image indirectly conveys certain properties of the objects such as shape elongation and object compactness. The variables [avgR, avgG, avgB] can be grouped by class of product and the overall mean intensity value, in each channel, can be obtained for each category. The images below show that each product class can be identified as single point in the meanRGB space.





More importantly, these plots help to visualize that the three variables are related in a linear fashion. Each point, corresponding to a product class, is labelled by the product type code. This visualization indicates that classes such as 2462 (‘*jeux_geek*’) and 2403 (‘*livres_jeunesse*’) have typically very ‘bright’ images, while product classes such as 2705 (‘*livres_et_illustres*’) and 2522 (‘*papeterie*’) have typically dark images. Running an Anova test confirms the correlation between average channel intensity and the product category.

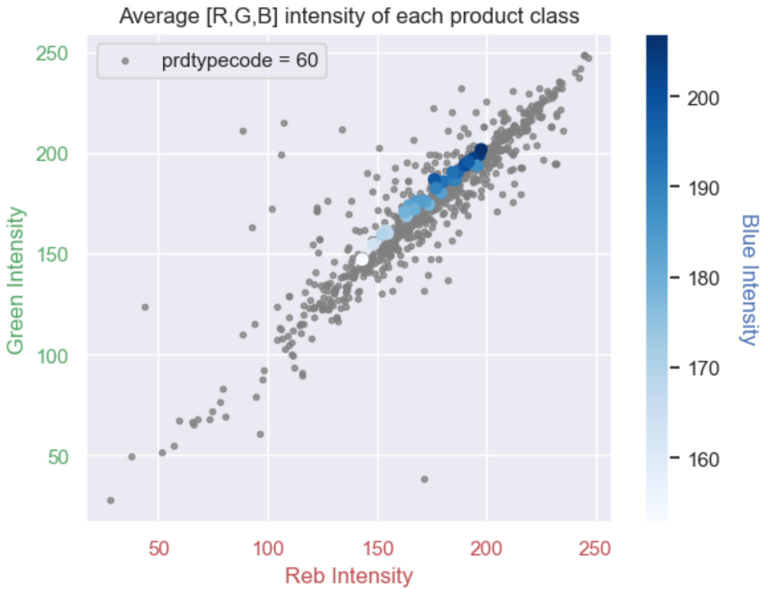


```
H0 = "There is NO correlation between 'avgB' and 'prdtypecode'"
H1 = "There is a correlation between  'avgB' and 'prdtypecode'"
```

	df	sum_sq	mean_sq	F	PR(>F)
prdtypecode	1.0	2.062858e+05	206285.812085	122.883632	1.546994e-28
Residual	84914.0	1.425459e+08	1678.708624	NaN	NaN

```
ANOVA p-value =  1.5469935243042498e-28
H0 is rejected, H1 is accepted
```

While the distinct separability among average RGB values across different categories may suggest the appeal of implementing a clustering classification algorithm, the plot below reveals a counterintuitive pattern. It illustrates that all items within a single category (grey dots) are scattered even beyond the entire range occupied by the average RGB values of individual categories. This vividly demonstrates that a clustering algorithm solely reliant on mean RGB values is inadequate for the classification task in this space.



Base models

In this section, first a series of foundational baseline models for our classification problem is discussed. These models are pivotal in shaping the project's course, offering essential insights into intrinsic data characteristics. Using simple algorithms, one can explore data patterns, important features and possible challenges, gaining insights about a baseline understanding of predictive performance while uncovering potential data limitations.
Furthermore, the development of these base models served as a guide, aiding in algorithm selection, and informing subsequent decisions. In fact, their simplicity and interpretability are instrumental in identifying suitable paths forward, from initial selection to fine-tuning allowing the development of a more advanced model that will be described in the next sections.

Train - Validation - Test split

Before training and evaluating the performance of the different models, the dataset is split in 3 sets in a stratified way, which allows to preserve the proportion of product classes within each subset. The 3 datasets are:

- Training set (64% - `54345 items`) for training the models.
- Validation set (16% - `13587 items`) for model selection and hyperparameter optimization.
- Test set (20% - `16984 items`) for model evaluation.

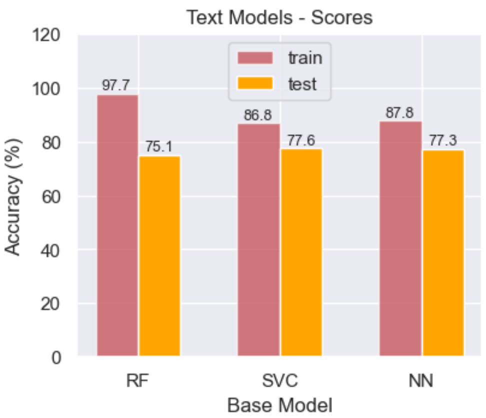
The Null Model

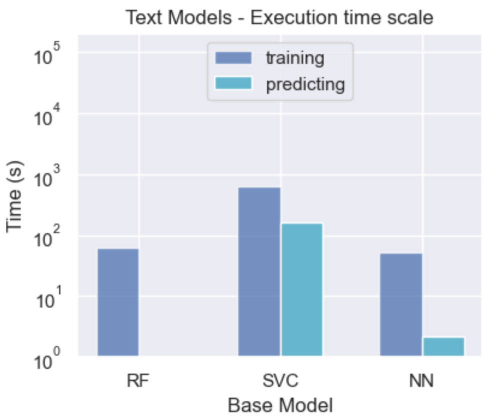
The most basic model for the prediction task will consist simply in predicting always the most common class seen on the training set, in this case the class ‘piscine_spa’ (code 2583). The test dataset contains a total of 16984 items, of which only 2042 items belong to the class ‘piscine_spa’. Therefore, the null model achieves an accuracy of only **12.02 %** on the test dataset. In fact, this is the only class that scores a precision of 1 (but with a recall of 0.12), while all other classes have precision and recall equal to 0.

Text Base Models

Three base models for classification leveraging only the textual data have been built: A random forest (RF), a support vector machine (SVC) and a dense neural network (NN). The following part will briefly describe their key advantages & drawbacks.

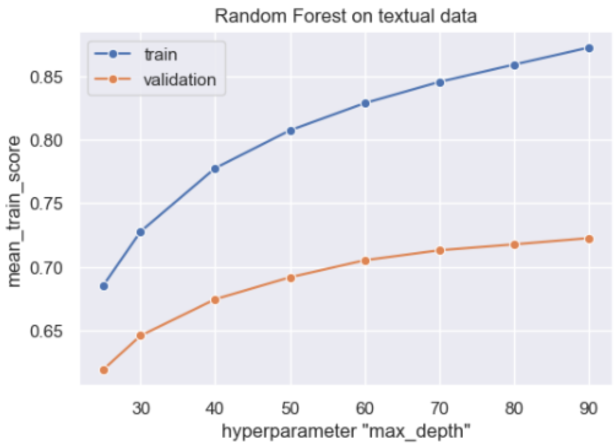
The bar plots below summarize and compares their performance. The figure in the left shows the accuracy score obtained with the trained models on the train set and more importantly on the test set. Additionally, the figure in the right shows the execution time scale involved in the whole training process as well as the execution time scale necessary to make predictions on the test set.



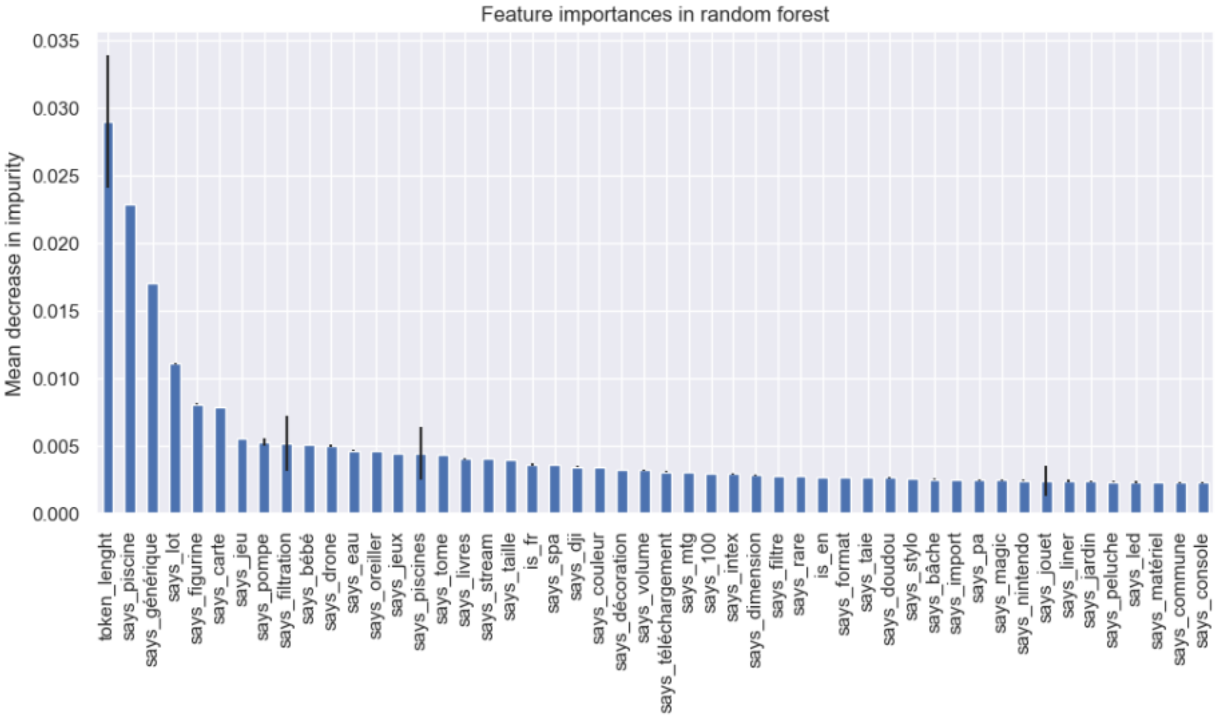


Random Forest (RF)

A basic random forest, instantiated with default parameters, achieves a test accuracy of 75.1%, which is already much better than the null model. However, the train accuracy = 97.7%, indicates that the model is overfitting. Fine tuning the model hyperparameters can help reduce overfitting. A key parameter to restrict would be the ‘max_depth’ of individual trees to keep the forest as an ensemble of ‘weak estimators’. In fact, the trained model has trees with max_depth up to 1019. However, restricting the max_depth also have a detrimental effect on the performance of the model on new data as shown in the plot below where both, train and validation accuracy, decreases when reducing the max_depth.



Despite this limitations, Random Forest can be used as an informative tool due their high degree of interpretability. In particular, we can obtain the importance of each feature based on the associate mean decrease of impurity every time a feature in used in a splitting node. This is displayed on the graph below, which shows the most important feature for this model.



Among the most important features are the token_length, some of the most common words of individual categories (‘piscine’, ‘carte’, ‘jeu’, ‘bébê’, ‘livre’, etc) and if the text is in french (is_fr) or english (is_en).

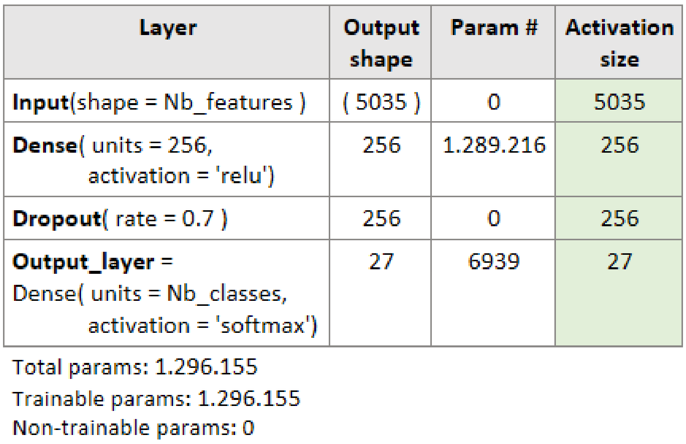
Support Vector Machines Classifier (SVC)

This type of classifier performs well in high-dimensional spaces, making them suitable for complex problems with many features as it is the case of text analysis. As a base model for this project a default SVC classifier shows less overfitting than the RF and an improved performance on the test set. Indeed, the train accuracy = 86.8%, while the test accuracy = 77.6%.

However, a major drawback of this model is that it is computationally expensive when dealing with large datasets affecting the memory usage and the computation time. As shown in the bar plot comparison of base text models, the SVC classifier takes 10 times more to train than the RF classifier, and more than 100 times longer to make predictions on the test set than the RF classifier.

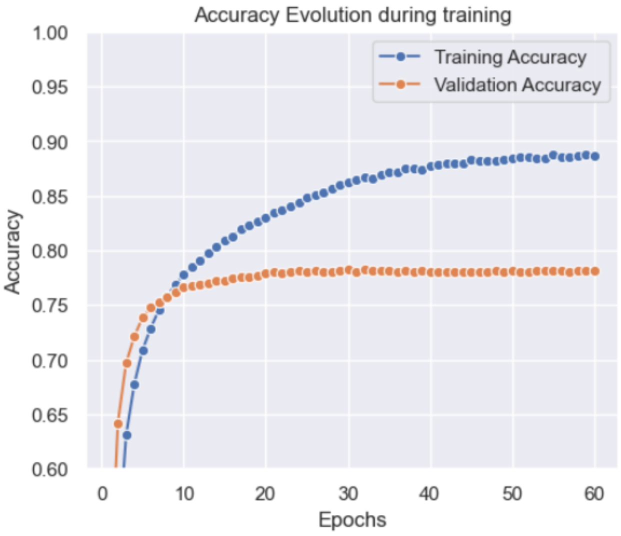
Dense Neural Network (NN)

Neural networks are powerful models that can handle at many levels the complex relationships within textual data. They can capture the non-linear relationships between words and their impact on the classification outcome. A simple neural network architecture as shown below serves a good base model. Notice that a drop out layer was included as a regularization mechanism to reduce overfitting and to improve the performance on the test set.



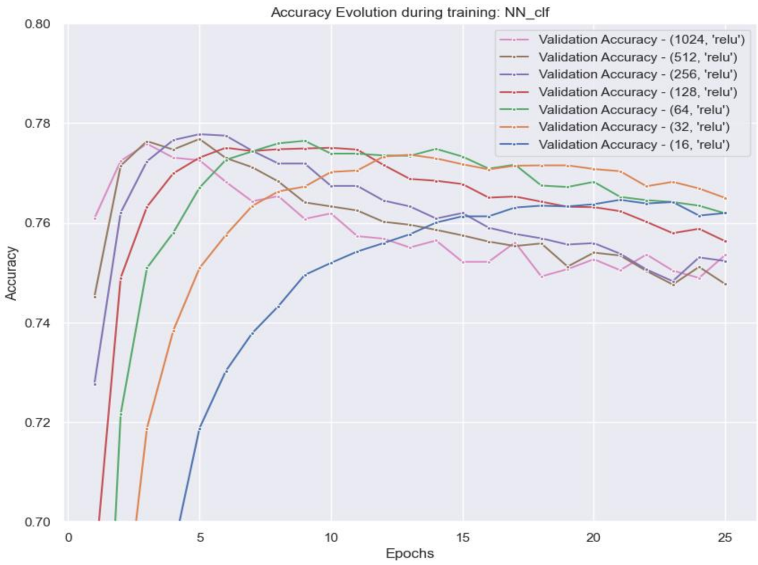
Layer	Output shape	Param #	Activation size
Input(shape = Nb_features)	(5035)	0	5035
Dense(units = 256, activation = 'relu')	256	1.289.216	256
Dropout(rate = 0.7)	256	0	256
Output_layer = Dense(units = Nb_classes, activation = 'softmax')	27	6939	27

Total params: 1.296.155
Trainable params: 1.296.155
Non-trainable params: 0

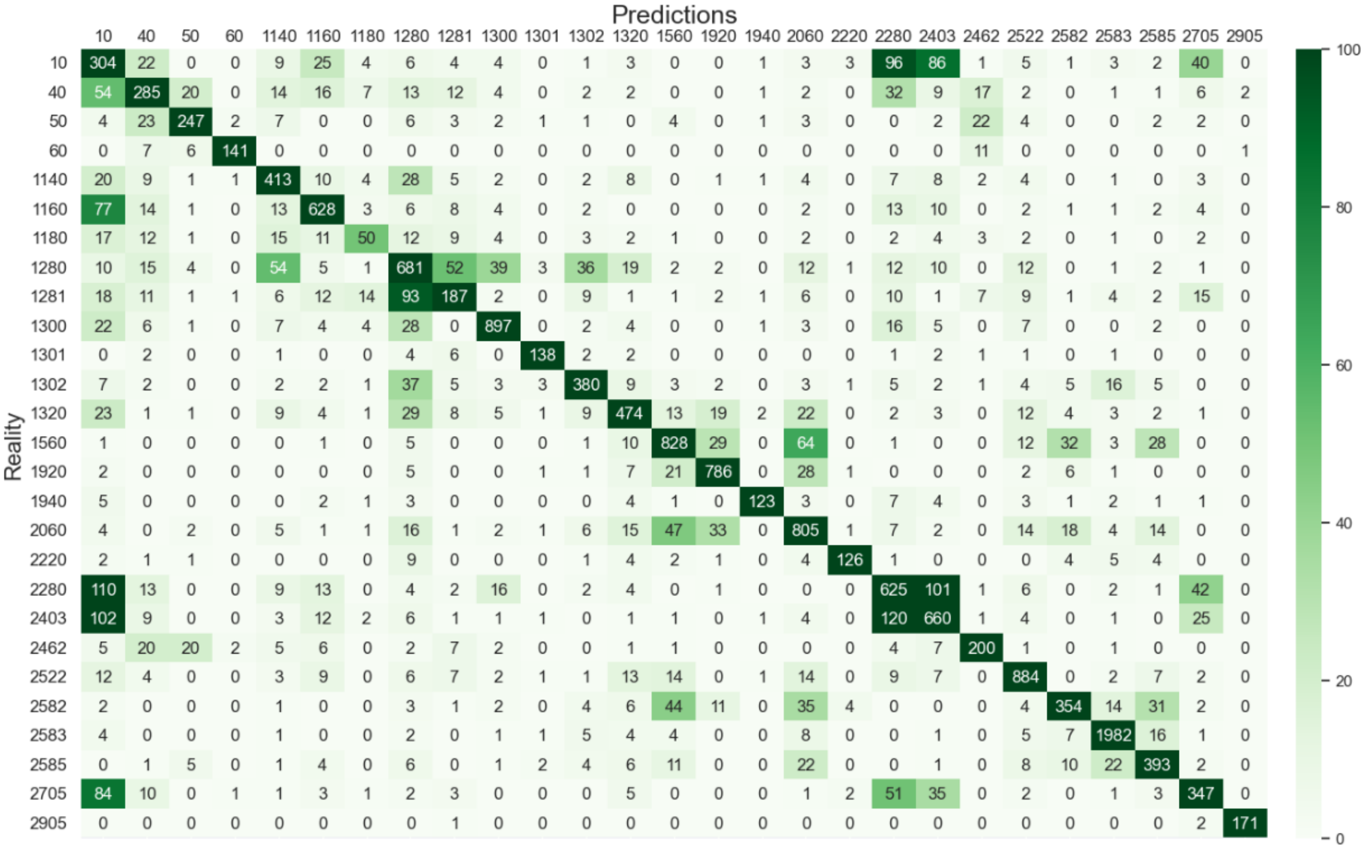


This model reaches a performance as good as the SVC classifier, but trains 10 times faster and it is also 100 times faster for making predictions on new data. The score obtained with this model were: train accuracy of 87.8 % and test accuracy 77.3 %.

The optimal number of units in the 1st dense layer was obtained by comparing the accuracy curve on the validation set during training process for different number of units. The plot below shows one of such comparisons, although in this case the initial learning rate was not optimal, one can notice that the highest peak is reached by the model with 256 units. A similar approach can be followed to obtain an optimal value for the dropout rate of **0.70**.



Due to its high performance in the test set and the low computation cost for training and predicting, the neural network (NN) model is selected as the ***best base text model*** for the classification task. The figure below displays the confusion matrix. The labels on the x and y axis are the product code for each of the 27 categories (for the respective category name, please refer to figure 1 of the report).

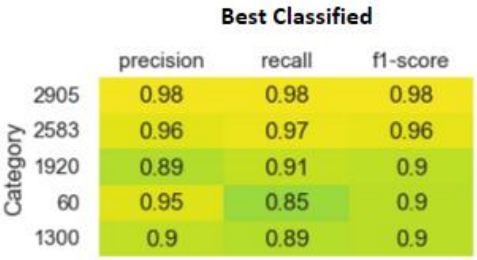


	10	40	50	60	1140	1160	1180	1280	1281	1300	1301	1302	1320	1560	1920	1940	2060	2220	2280	2403	2462	2522	2582	2583	2585	2705	2905
10	304	22	0	0	9	25	4	6	4	4	0	1	3	0	0	1	3	3	96	86	1	5	1	3	2	40	0
40	54	285	20	0	14	16	7	13	12	4	0	2	2	0	0	1	2	0	32	9	17	2	0	1	1	6	2
50	4	23	247	2	7	0	0	6	3	2	1	1	0	4	0	1	3	0	0	2	22	4	0	0	2	2	0
60	0	7	6	141	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	1
1140	20	9	1	1	413	10	4	28	5	2	0	2	8	0	1	1	4	0	7	8	2	4	0	1	0	3	0
1160	77	14	1	0	13	628	3	6	8	4	0	2	0	0	0	0	2	0	13	10	0	2	1	1	2	4	0
1180	17	12	1	0	15	11	50	12	9	4	0	3	2	1	0	0	2	0	2	4	3	2	0	1	0	2	0
1280	10	15	4	0	54	5	1	681	52	39	3	36	19	2	2	0	12	1	12	10	0	12	0	1	2	1	0
1281	18	11	1	1	6	12	14	93	187	2	0	9	1	1	2	1	6	0	10	1	7	9	1	4	2	15	0
1300	22	6	1	0	7	4	4	28	0	897	0	2	4	0	0	1	3	0	16	5	0	7	0	0	2	0	0
1301	0	2	0	0	1	0	0	4	6	0	138	2	2	0	0	0	0	0	1	2	1	1	0	1	0	0	0
1302	7	2	0	0	2	2	1	37	5	3	3	380	9	3	2	0	3	1	5	2	1	4	5	16	5	0	0
1320	23	1	1	0	9	4	1	29	8	5	1	9	474	13	19	2	22	0	2	3	0	12	4	3	2	1	0
1560	1	0	0	0	0	1	0	5	0	0	0	1	10	828	29	0	64	0	1	0	0	12	32	3	28	0	0
1920	2	0	0	0	0	0	0	5	0	0	1	1	7	21	786	0	28	1	0	0	0	2	6	1	0	0	0
1940	5	0	0	0	0	2	1	3	0	0	0	0	4	1	0	123	3	0	7	4	0	3	1	2	1	1	0
2060	4	0	2	0	5	1	1	16	1	2	1	6	15	47	33	0	805	1	7	2	0	14	18	4	14	0	0
2220	2	1	1	0	0	0	0	9	0	0	0	1	4	2	1	0	4	126	1	0	0	0	4	5	4	0	0
2280	110	13	0	0	9	13	0	4	2	16	0	2	4	0	1	0	0	0	625	101	1	6	0	2	1	42	0
2403	102	9	0	0	3	12	2	6	1	1	1	0	1	1	0	1	4	0	120	660	1	4	0	1	0	25	0
2462	5	20	20	2	5	6	0	2	7	2	0	0	1	1	0	0	0	0	4	7	200	1	0	1	0	0	0
2522	12	4	0	0	3	9	0	6	7	2	1	1	13	14	0	1	14	0	9	7	0	884	0	2	7	2	0
2582	2	0	0	0	1	0	0	3	1	2	0	4	6	44	11	0	35	4	0	0	0	4	354	14	31	2	0
2583	4	0	0	0	1	0	0	2	0	1	1	5	4	4	0	0	8	0	0	1	0	5	7	1982	16	1	0
2585	0	1	5	0	1	4	0	6	0	1	2	4	6	11	0	0	22	0	0	1	0	8	10	22	393	2	0
2705	84	10	0	1	1	3	1	2	3	0	0	0	5	0	0	0	1	2	51	35	0	2	0	1	3	347	0
2905	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	171

Non-null off-diagonal elements indicate categories that are often confused by the model. For example, this text base model can sometimes:

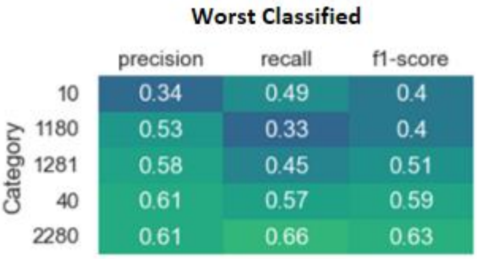
- Confuse all four categories: 10 ('livres_adulte'), 2403 ('livres_jeunesse'), 2280 ('magazines'), 2705 ('livres_et_illustres').
- Classify items from category 1280 ('jouets_enfance') as belonging to the category 1281 ('jeux_societe'), 1300 ('modeles_reduits_ou_telecomabdes') and 1302 ('jouets_enfants'); and viceversa.
- Classify items from categories 1160 ('cartes_a_jouer') and 40 ('jeux_videos_import') as belonging to category 10.

Notice that these categories are closely related, and it can be challenging to stablish a clear difference in how their type of object can be described with text in a website. A summary of the top-5 best and worst classified categories is displayed below:



Best Classified

Category	precision	recall	f1-score
2905	0.98	0.98	0.98
2583	0.96	0.97	0.96
1920	0.89	0.91	0.9
60	0.95	0.85	0.9
1300	0.9	0.89	0.9

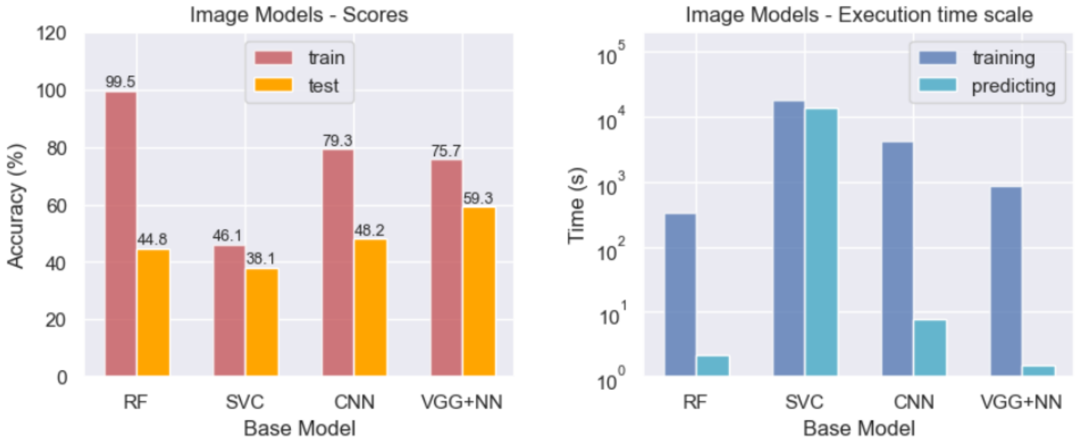


Worst Classified

Category	precision	recall	f1-score
10	0.34	0.49	0.4
1180	0.53	0.33	0.4
1281	0.58	0.45	0.51
40	0.61	0.57	0.59
2280	0.61	0.66	0.63

Image Base Models

Four image base models were trained to classify the product using only the image data: a random forest (RF), a support vector machine (SVC), a convolutional neural network (CNN) and a hybrid model that make use of convolutional pretrained layer of the VGG16 model in combination with a set of custom trainable dense layers (VGG+NN). The following bar plot compares their performance in term of the accuracy and the computation time.

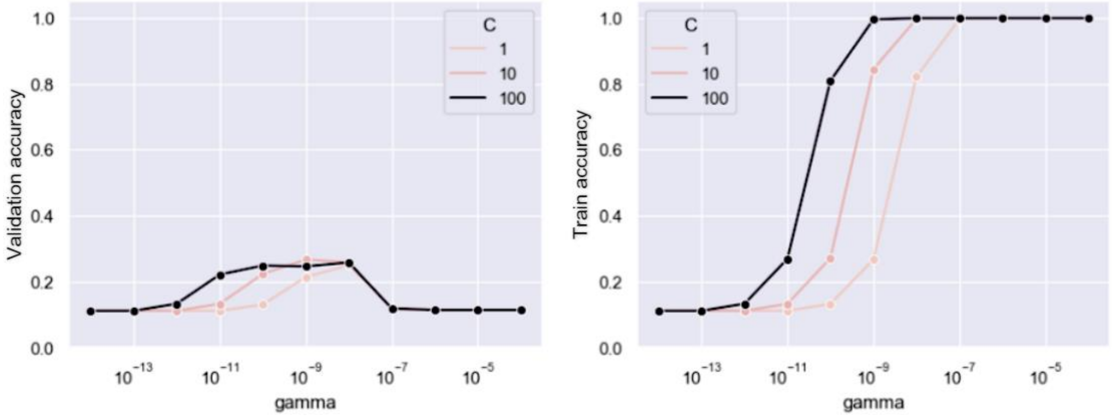


Random Forest (RF)

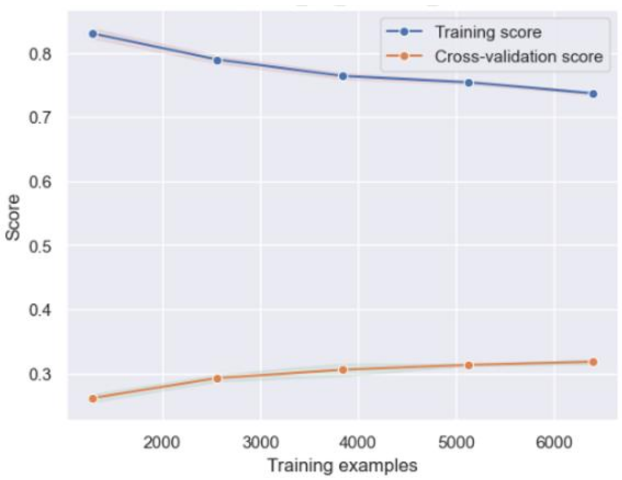
A generic random forest (default parameters) was trained on the image data, which resulted in an ensemble of trees with a maximum depth of up to 72 nodes. The model takes about 132 seconds to train, achieving a train accuracy of 99.5 %. When faced to new data, the RF model takes 12 seconds to get prediction on the test dataset, with a measured accuracy of 44.8 %. As seen for the text base models, the unrestricted random forest tends to overfit, and it does not generalize well.

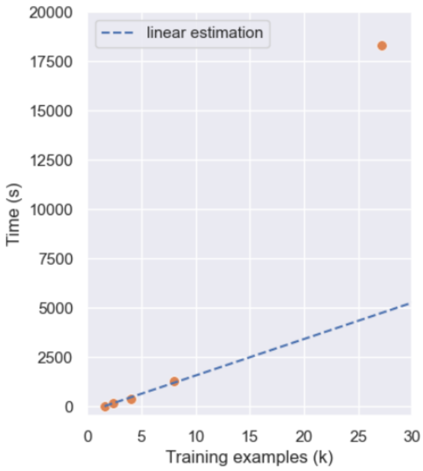
Support Vector Machine Classifier (SVC)

A default SVC classifier was instantiated and trained for about 5 hours to achieve a train accuracy of 46.1 % on the train set and an accuracy of 38.1 % on the test set. Nevertheless, the model being computationally expensive, only 50% of training dataset was possible to feed to the model at once. Improvements on this model could be achieved by tunning the hyperparameters ‘gamma’ (γ) and ‘C’ which typically have an inverse proportion relationship of the form γ * C = constant, for the optimal performance. A quick verification was performed training the model in only 3% of the image dataset. The figure below shows the performance landscape of the model in the validation set (left) and train set (right) for different sets of parameters and C. Note that the trained default classifier has C=1, and γ =1/(nb_features * Xdata.var()) ~ 5x10^{-9} for the image dataset.



Increasing the size of the dataset on which the model is training improves the validation score and reduces overfitting but, the improvements seem to slow down beyond 15% of the whole dataset as shown in the left figure below. Additionally, a linear time complexity estimation largely underestimates the observation when training on 50% of the dataset (~27k training examples) as shown in the right figure below.

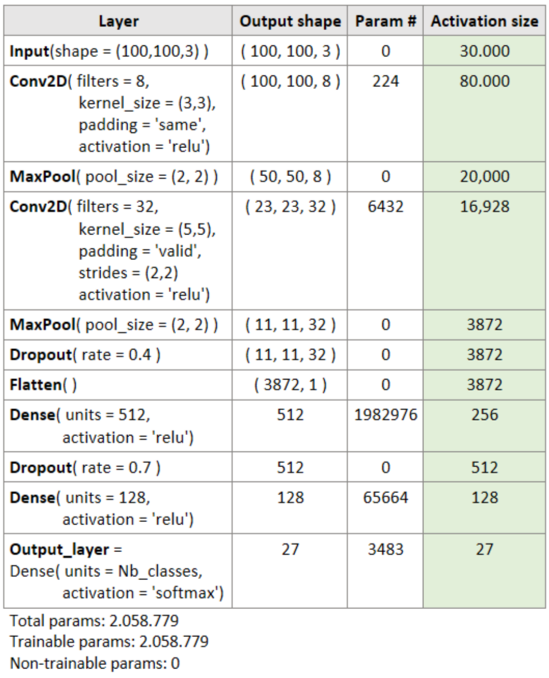




<u>Convolution Neural Network</u> (CNN)

Convolutional Neural Networks are highly effective as image classification models due to their ability to capture and exploit spatial hierarchies and patterns within images. CNNs consisting of multiple layers learn progressively complex features. Lower layers capture simple features like edges, while deeper layers learn more abstract and high-level features like complex shapes or object parts. This type of neural network takes advantage of ‘convolution layers’ having a set of small filters that convolves the images capturing spatial relationships as well as ‘Pooling layers’ that helps selecting important feature and downscale the feature space during training.

A CNN model with an architecture inspired in LeNet model was trained on the image dataset. The parameters of the different layers were chosen to keep the feature space (output shape) wide and shallow in the initial stages allowing the network to find many complex relationships, and then make it narrower and deeper towards the end, forcing the network to summarize the important findings. A good practice is to try to smoothly reduce the activation shape while progressing on the network and avoid abrupt changes. The right figure below shows the accuracy measured during training on the train and validation set.



Layer	Output shape	Param #	Activation size
Input(shape = (100,100,3))	(100, 100, 3)	0	30.000

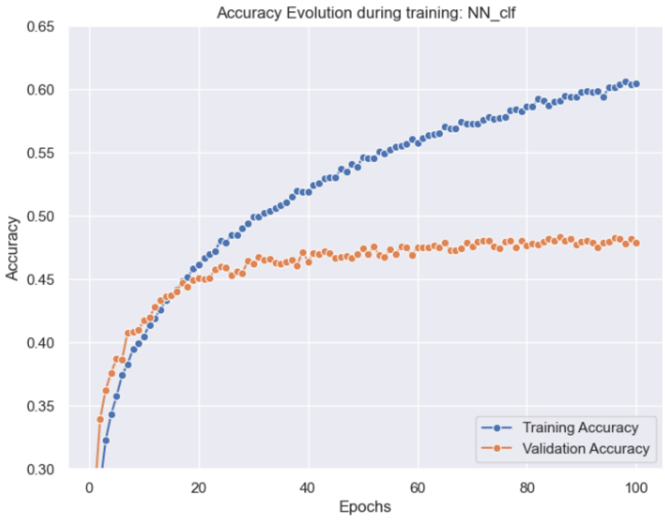
Conv2D(filters = 8,
kernel_size = (3,3),
padding = 'same',
activation = 'relu')
	(100, 100, 8)	224	80.000

MaxPool(pool_size = (2, 2))	(50, 50, 8)	0	20,000

Conv2D(filters = 32,
kernel_size = (5,5),
padding = 'valid',
strides = (2,2)
activation = 'relu')
	(23, 23, 32)	6432	16,928

MaxPool(pool_size = (2, 2))	(11, 11, 32)	0	3872
Dropout(rate = 0.4)	(11, 11, 32)	0	3872
Flatten( )	(3872, 1)	0	3872
Dense(units = 512, activation = 'relu')	512	1982976	256
Dropout(rate = 0.7)	512	0	512
Dense(units = 128, activation = 'relu')	128	65664	128
Output_layer = Dense(units = Nb_classes, activation = 'softmax')	27	3483	27

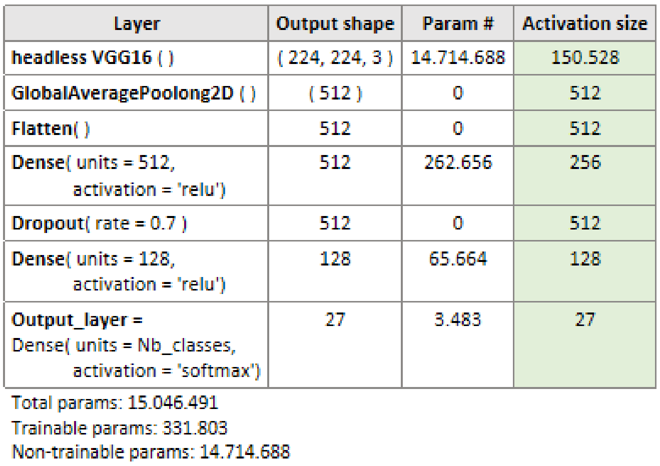
Total params: 2.058.779
Trainable params: 2.058.779
Non-trainable params: 0



As can be seen in the bar plot comparing the four image base models, the CNN model achieves a better performance than the classical machine learning algorithms (RF & CSV), resulting in a test accuracy of 48.2 %. This model has a moderate degree of overfitting thanks to the *dropout* layers which are inserted before the *dense* layers since these contains most of the model’s parameters. Concerning the computation time, the CNN model needs 1h-9min to train on the whole training subset, but it might be possible to reduce the training time by half and still keep an accuracy above 48%.

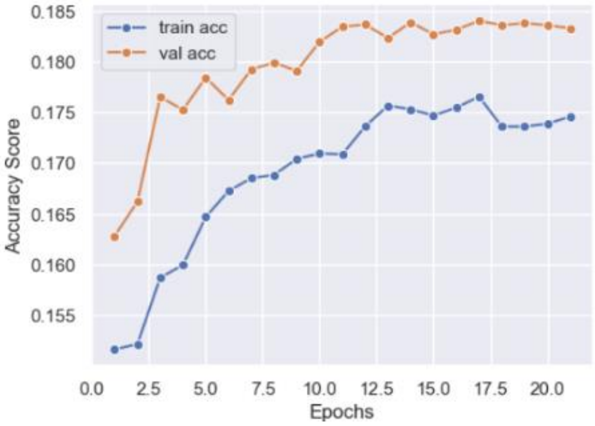
Modified VGG16 model (VGG+NN)

This model takes advantage of the transfer learning technique commonly used in deep learning fields. This image base model is built on top of a set of pretrained layers from the VGG16 model, available through the *keras* library. For this base model we keep only the convolution layers from the VGG16 and drop its native top dense layers. Then a series of custom dense layer is added on top of the headless VGG. These new dense layers will have an architecture more adapted to the classification problem of this project and contain the parameters that need to be trained on our image dataset. One can train the complete modified VGG model on the image dataset using the image generators and the *flow_from_directory()* method to be able to process images of size 224 *x* 224 pixels with limited RAM. A first try to this approach puts in evidence another difficulty: the VGG16 model is a large model, and hence training the complete modified VGG model with a moderate batch size of 128 training examples takes up to 1h30min per epoch. It practices, tuning this model becomes extremely difficult due to the time scales. The figure below shows the poor training process of this model following this approach.

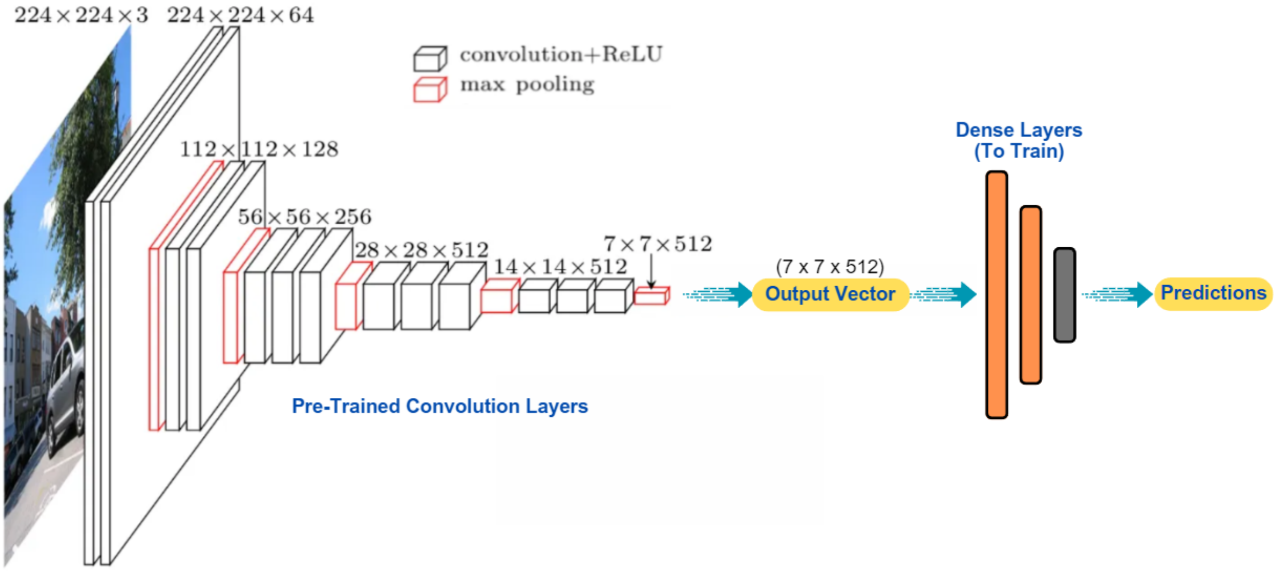


Layer	Output shape	Param #	Activation size
headless VGG16 ( )	(224, 224, 3)	14.714.688	150.528
GlobalAveragePoolong2D ( )	(512)	0	512
Flatten( )	512	0	512
Dense(units = 512, activation = 'relu')	512	262.656	256
Dropout(rate = 0.7)	512	0	512
Dense(units = 128, activation = 'relu')	128	65.664	128
Output_layer = Dense(units = Nb_classes, activation = 'softmax')	27	3.483	27

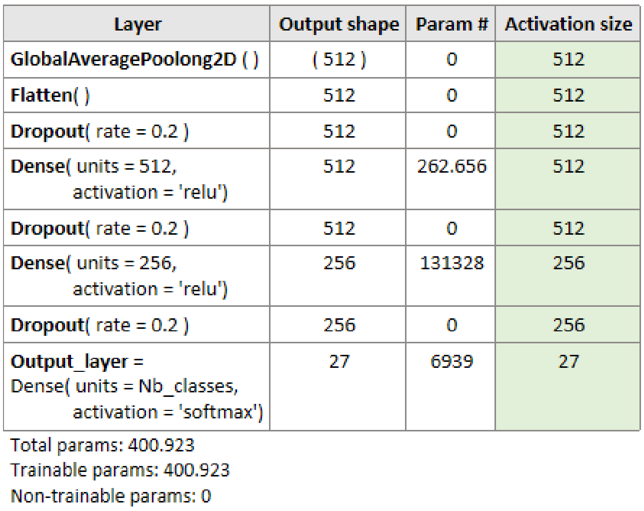
Total params: 15.046.491
Trainable params: 331.803
Non-trainable params: 14.714.688



However, another approach is possible to circumvents this issue: one can use the pretrained VGG16 as a static feature extractor, decoupled from the dense layer to train. The headless VGG16 model will be use as a preprocessing step. It will take a picture and transform it into an output vector of shape (7,7,512) which is locally stored. Once all the images have been transformed by VGG16, a simple neural network will be trained on the transformed datasets constituted by all the output vectors obtained from the images. The figure below is a schematic of this approach.

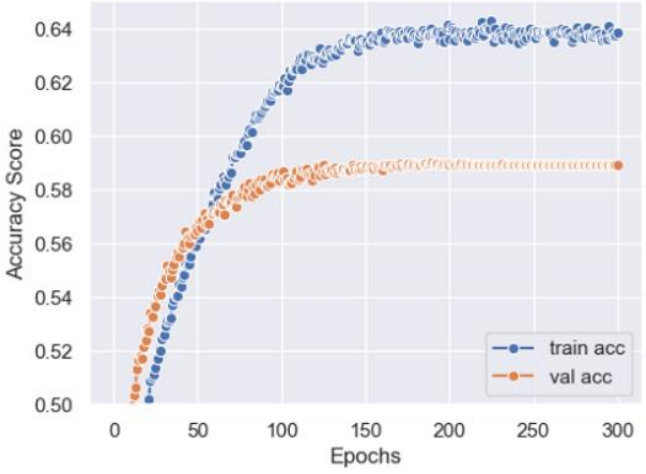


With this methodology, training the dense layer is relatively fast. Taking only 3s per epoch, the total training is achieved in 14 mins, which is a considerably improvement from the previous method. This allows to quickly tune the model to find optimal parameters. The figure below shows the final layer architecture as well as the train and validation accuracies during training process.

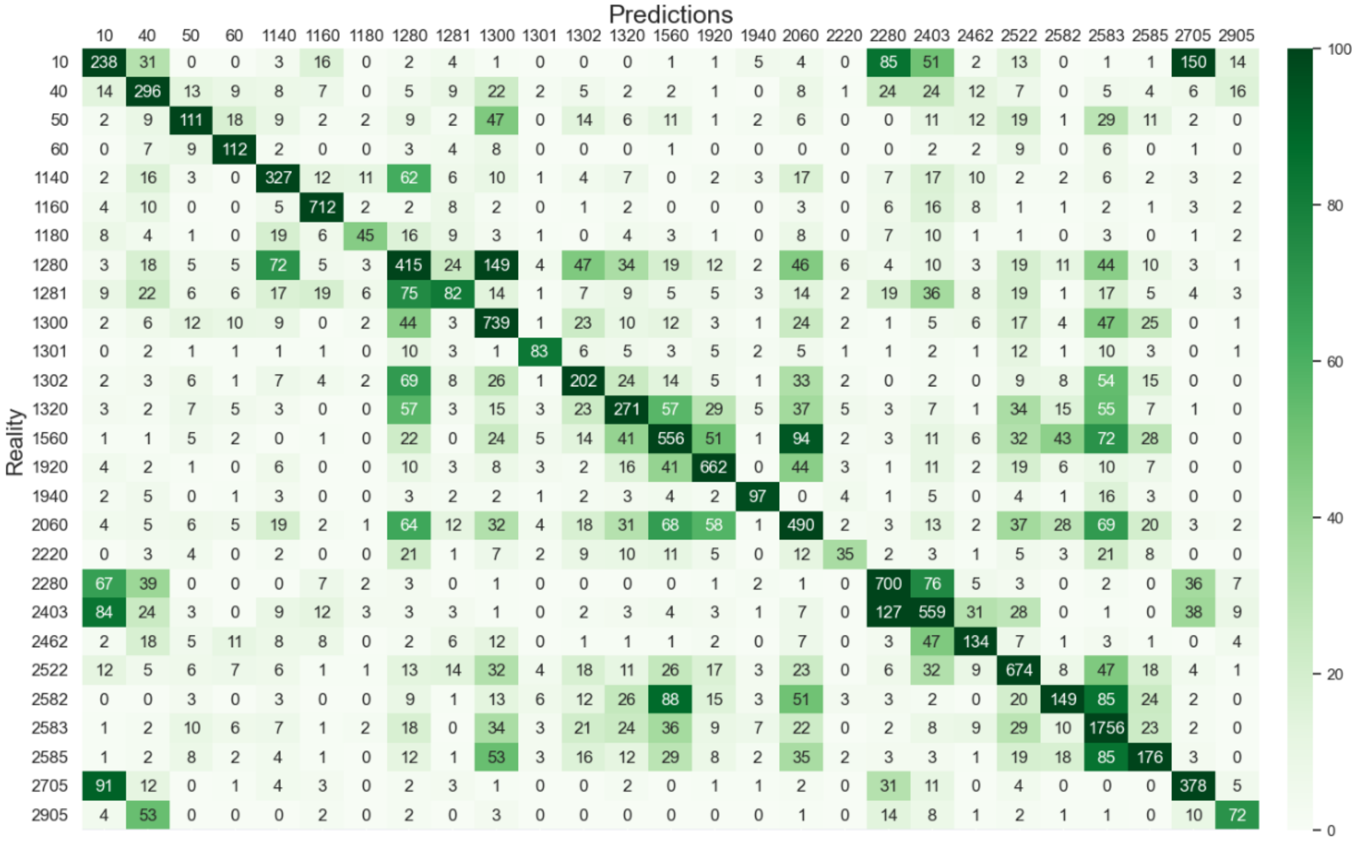


Layer	Output shape	Param #	Activation size
GlobalAveragePoolong2D ( )	(512)	0	512
Flatten( )	512	0	512
Dropout(rate = 0.2)	512	0	512
Dense(units = 512, activation = 'relu')	512	262.656	512
Dropout(rate = 0.2)	512	0	512
Dense(units = 256, activation = 'relu')	256	131328	256
Dropout(rate = 0.2)	256	0	256
Output_layer = Dense(units = Nb_classes, activation = 'softmax')	27	6939	27

Total params: 400.923
Trainable params: 400.923
Non-trainable params: 0



This model achieves the best performance among all the image base models candidates, with a test accuracy of 59.3 %, more than 10 points above the best CNN model. Additionally, it is relatively fast when performing predictions on the test set (1.5 s). This model is then chosen as the ***best image base model***. The graph below displays the confusion matrix of this model. The axis labels are the product code for each category.

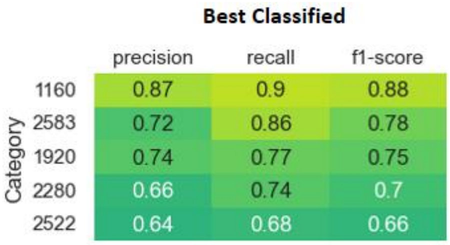


	10	40	50	60	1140	1160	1180	1280	1281	1300	1301	1302	1320	1560	1920	1940	2060	2220	2280	2403	2462	2522	2582	2583	2585	2705	2905
10	238	31	0	0	3	16	0	2	4	1	0	0	0	1	1	5	4	0	85	51	2	13	0	1	1	150	14
40	14	296	13	9	8	7	0	5	9	22	2	5	2	2	1	0	8	1	24	24	12	7	0	5	4	6	16
50	2	9	111	18	9	2	2	9	2	47	0	14	6	11	1	2	6	0	0	11	12	19	1	29	11	2	0
60	0	7	9	112	2	0	0	3	4	8	0	0	0	1	0	0	0	0	0	2	2	9	0	6	0	1	0
1140	2	16	3	0	327	12	11	62	6	10	1	4	7	0	2	3	17	0	7	17	10	2	2	6	2	3	2
1160	4	10	0	0	5	712	2	2	8	2	0	1	2	0	0	0	3	0	6	16	8	1	1	2	1	3	2
1180	8	4	1	0	19	6	45	16	9	3	1	0	4	3	1	0	8	0	7	10	1	1	0	3	0	1	2
1280	3	18	5	5	72	5	3	415	24	149	4	47	34	19	12	2	46	6	4	10	3	19	11	44	10	3	1
1281	9	22	6	6	17	19	6	75	82	14	1	7	9	5	5	3	14	2	19	36	8	19	1	17	5	4	3
1300	2	6	12	10	9	0	2	44	3	739	1	23	10	12	3	1	24	2	1	5	6	17	4	47	25	0	1
1301	0	2	1	1	1	1	0	10	3	1	83	6	5	3	5	2	5	1	1	2	1	12	1	10	3	0	1
1302	2	3	6	1	7	4	2	69	8	26	1	202	24	14	5	1	33	2	0	2	0	9	8	54	15	0	0
1320	3	2	7	5	3	0	0	57	3	15	3	23	271	57	29	5	37	5	3	7	1	34	15	55	7	1	0
1560	1	1	5	2	0	1	0	22	0	24	5	14	41	556	51	1	94	2	3	11	6	32	43	72	28	0	0
1920	4	2	1	0	6	0	0	10	3	8	3	2	16	41	662	0	44	3	1	11	2	19	6	10	7	0	0
1940	2	5	0	1	3	0	0	3	2	2	1	2	3	4	2	97	0	4	1	5	0	4	1	16	3	0	0
2060	4	5	6	5	19	2	1	64	12	32	4	18	31	68	58	1	490	2	3	13	2	37	28	69	20	3	2
2220	0	3	4	0	2	0	0	21	1	7	2	9	10	11	5	0	12	35	2	3	1	5	3	21	8	0	0
2280	67	39	0	0	0	7	2	3	0	1	0	0	0	0	1	2	1	0	700	76	5	3	0	2	0	36	7
2403	84	24	3	0	9	12	3	3	3	1	0	2	3	4	3	1	7	0	127	559	31	28	0	1	0	38	9
2462	2	18	5	11	8	8	0	2	6	12	0	1	1	1	2	0	7	0	3	47	134	7	1	3	1	0	4
2522	12	5	6	7	6	1	1	13	14	32	4	18	11	26	17	3	23	0	6	32	9	674	8	47	18	4	1
2582	0	0	3	0	3	0	0	9	1	13	6	12	26	88	15	3	51	3	3	2	0	20	149	85	24	2	0
2583	1	2	10	6	7	1	2	18	0	34	3	21	24	36	9	7	22	0	2	8	9	29	10	1756	23	2	0
2585	1	2	8	2	4	1	0	12	1	53	3	16	12	29	8	2	35	2	3	3	1	19	18	85	176	3	0
2705	91	12	0	1	4	3	0	2	3	1	0	0	2	0	1	1	2	0	31	11	0	4	0	0	0	378	5
2905	4	53	0	0	0	2	0	2	0	3	0	0	0	0	0	0	1	0	14	8	1	2	1	1	0	10	72

As always, non-null off-diagonal elements indicate categories that are often misclassified. Similar to the best text base model (NN), this image base model (VGG+NN) often confuses:

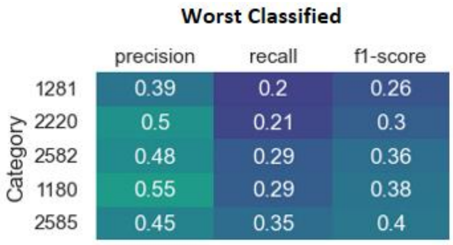
- Category 10 ('livres_adulte') with category 2403 ('livres_jeunesse') & 2280 ('magazines'), but it shows a slightly improvement. However, it confuses more often the category 10 with category 2705 ('livres_et_illustres') than the base text model.
- Category 1280 ('jouets_enfance') and Category 1300 ('modeles_reduits_ou_telecomabdes').

However, one can notice that these are categories that are closely related, and it can be difficult to distinguish even for a human being by just looking at an image. Moreover, it seems that many items are misclassified into the categories 1280, 2060, and 2583 as seen by vertical darker columns. Among the top 5 worst classified categories we find the categories 2220 and 1180 are also part of the 5 less represented categories.



Best Classified

Category	precision	recall	f1-score
1160	0.87	0.9	0.88
2583	0.72	0.86	0.78
1920	0.74	0.77	0.75
2280	0.66	0.74	0.7
2522	0.64	0.68	0.66

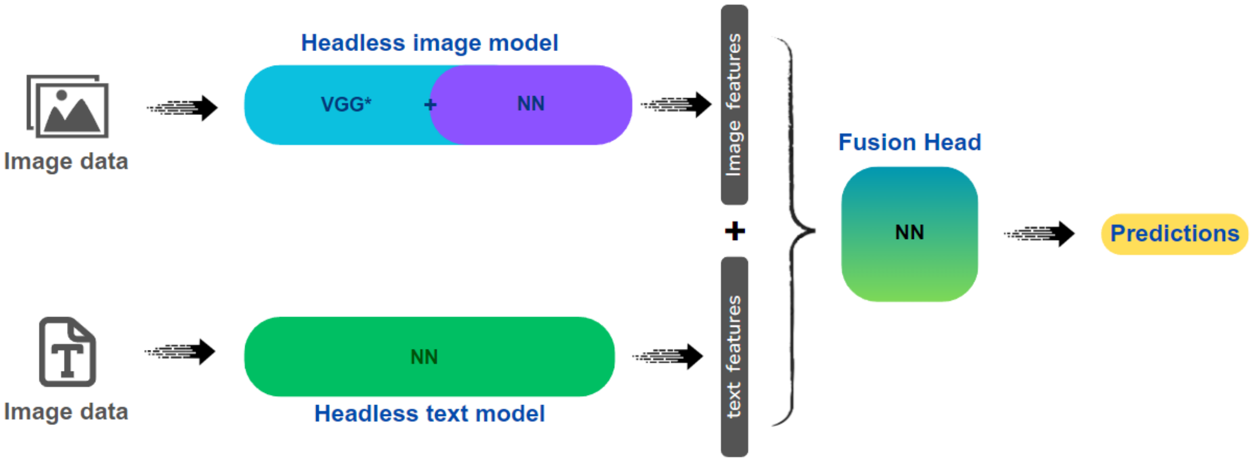


Worst Classified

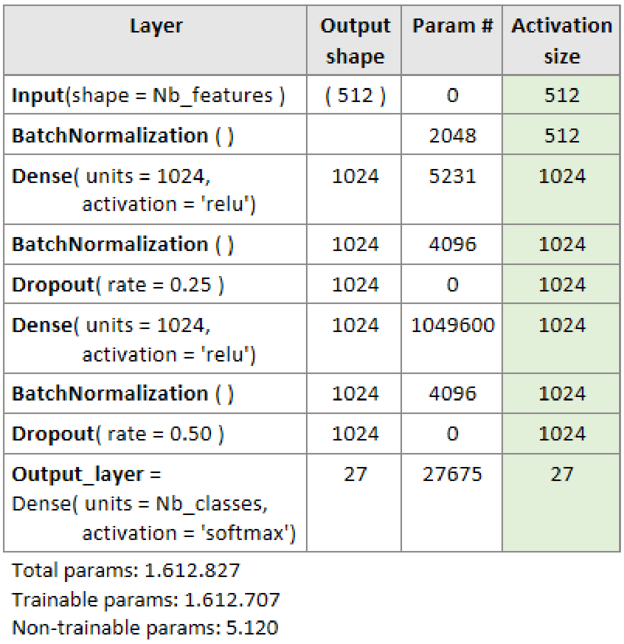
Category	precision	recall	f1-score
1281	0.39	0.2	0.26
2220	0.5	0.21	0.3
2582	0.48	0.29	0.36
1180	0.55	0.29	0.38
2585	0.45	0.35	0.4

Fusion model

A more advanced model, called the Fusion Model is developed in order to be able to use all the available data for the classification task: textual and image data, by taking advantage of the best performing base models discussed before: the NN model for text data and the VGG+NN model for image data. The fusion model will use the pretrained and optimized NN model to process the text data and extract the most relevant "features". These new features are just the output vector of the last dense layer, before the classification layer. In a similar way, the fusion model will use the pretrained and optimized VGG+NN model to process the image data and extract the set of most relevant features for our classification problem. As expected, these new image "features" are in the output vector of the last dense layer before classification. Then, the newly extracted text and image features are concatenated after proper normalization to form what it will be the input data of the fusion model head. The head model, a dense neural network, will then train in a dataset which effectively combines text and image features that are highly relevant for the classification problem. The image below shows a schematic of the fusion model.

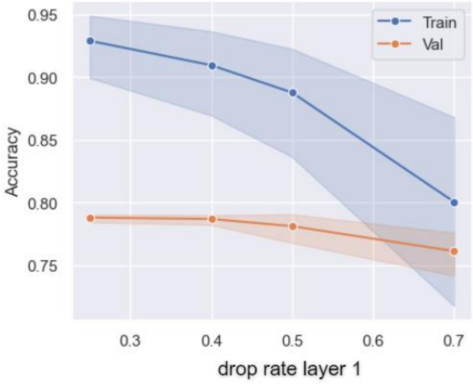


The fusion head model was trained in the concatenated feature vector dataset form by 250 text features plus 250 image features for each item in the catalogue. The architecture of the fusion head model consists of two dense layers with 1024 units each. Additionally, dropout layers have been included to prevent overfitting and favour the generalization of the model. This particular architecture and its several hyperparameters have been fine tuned in order to optimize the model by looking the performance on the validation set. The plot below shows the measurements that allows tunning the dropout rate on the first layer as an example. The final model architecture is summarized on the left table below.

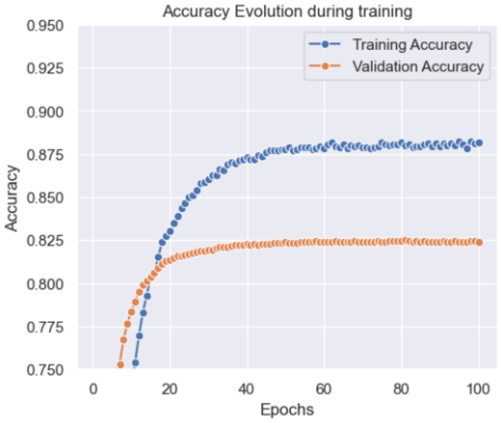


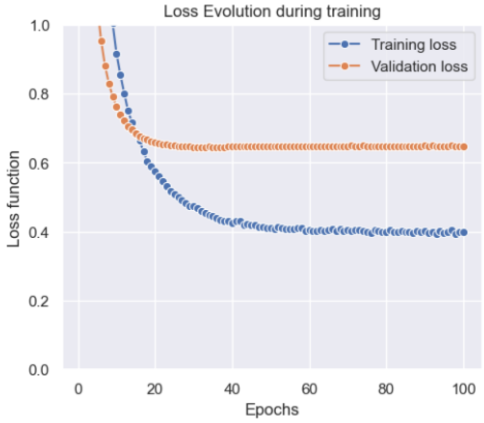
Layer	Output shape	Param #	Activation size
Input(shape = Nb_features)	(512)	0	512
BatchNormalization ( )		2048	512
Dense(units = 1024, activation = 'relu')	1024	5231	1024
BatchNormalization ( )	1024	4096	1024
Dropout(rate = 0.25)	1024	0	1024
Dense(units = 1024, activation = 'relu')	1024	1049600	1024
BatchNormalization ( )	1024	4096	1024
Dropout(rate = 0.50)	1024	0	1024
Output_layer = Dense(units = Nb_classes, activation = 'softmax')	27	27675	27

Total params: 1.612.827
Trainable params: 1.612.707
Non-trainable params: 5.120

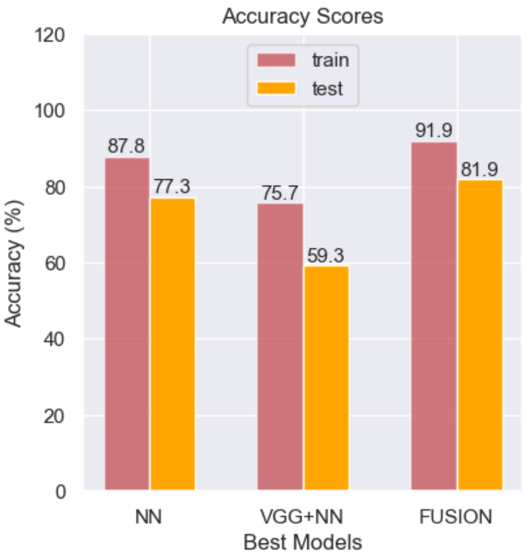


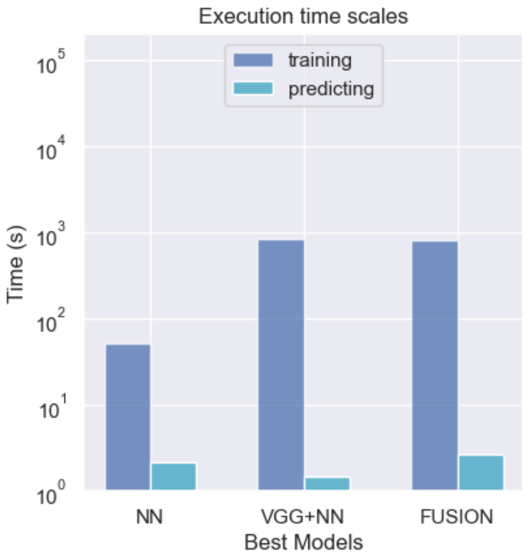
The model was trained for 100 epochs with an initial learning of rate of 2e-6, allowing it to smoothly reach a minimum in the loss function, while maximizing the accuracy in the validation set:



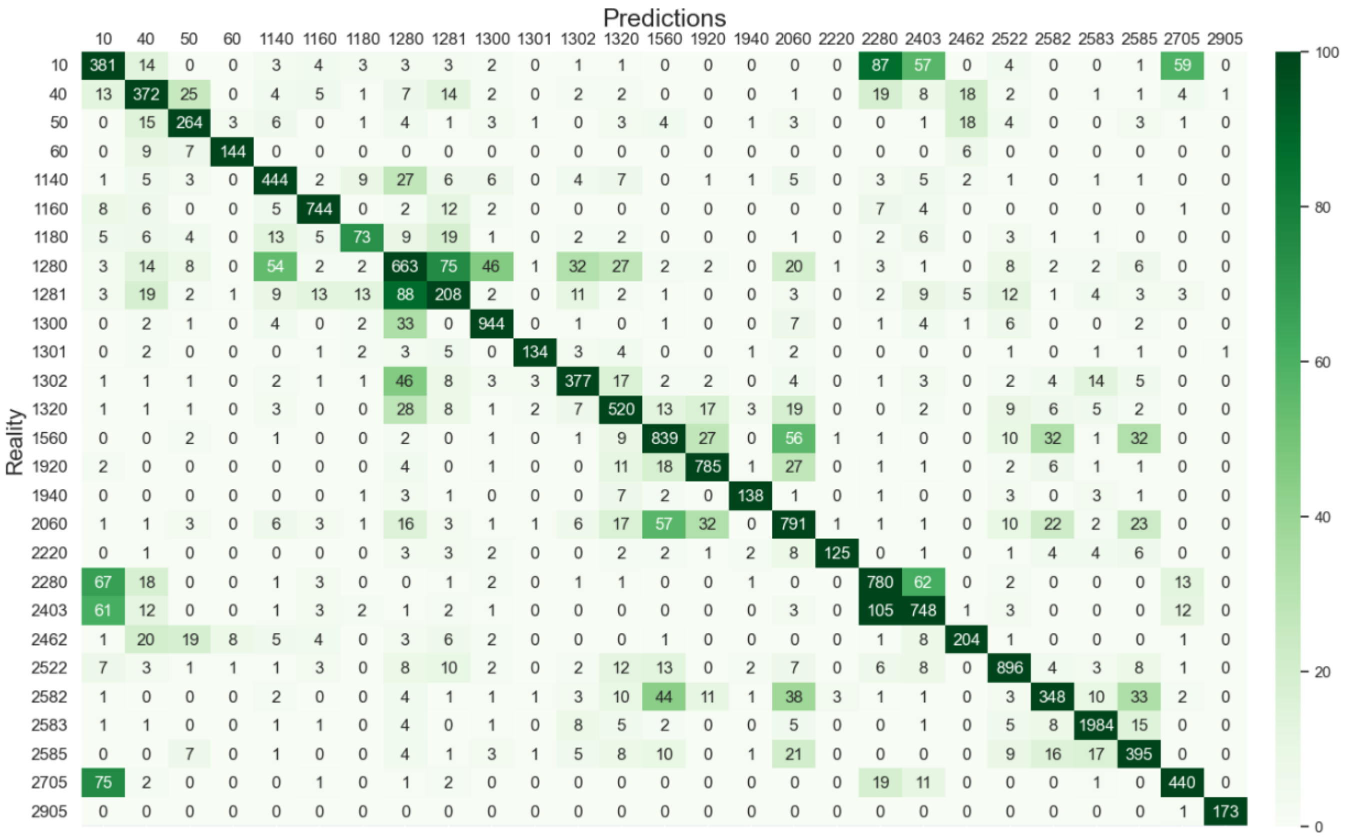


The bar plot below shows a comparison of the performance between the best text base model (NN), the best image base model (VGG+NN) and the fusion model (FM). The fusion model performs much better than the other models, achieving a **test accuracy** of **81.9 %**, which is almost 5% higher than the best base model trained on this project. Regarding the computation time, the fusion model trains in about 13 minutes and remains fast for computing predictions (2.7 seconds for the entire test set).



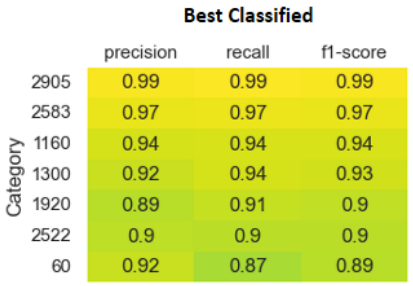


The figure below displays the confusion matrix of the fusion model. One can see that it is much cleaner than the confusion matrix of the base models, having most of the off-diagonal elements smaller than 10 units. The fusion model is better than the text model at distinguishing the categories 10 and 40 while reducing the number of categories that previously were sometimes misclassified as 10 (see for example the categories 1140, 1160, 1281, 1300, 1320 in the confusion matrix of the NN model). The fusion model also reduces the misclassification of the often-confused pair of categories 2403-2280 ('livres_jeunesse' & 'magazines') and 1280-1281 ('jouets_enfance' & 'jeux_societe').



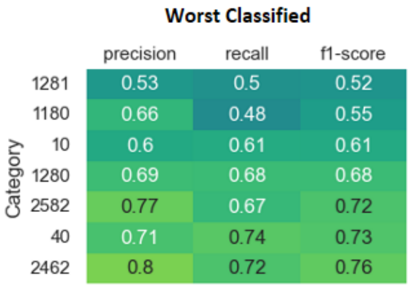
Reality \ Predictions	10	40	50	60	1140	1160	1180	1280	1281	1300	1301	1302	1320	1560	1920	1940	2060	2220	2280	2403	2462	2522	2582	2583	2585	2705	2905
10	381	14	0	0	3	4	3	3	3	2	0	1	1	0	0	0	0	0	87	57	0	4	0	0	1	59	0
40	13	372	25	0	4	5	1	7	14	2	0	2	2	0	0	0	1	0	19	8	18	2	0	1	1	4	1
50	0	15	264	3	6	0	1	4	1	3	1	0	3	4	0	1	3	0	0	1	18	4	0	0	3	1	0
60	0	9	7	144	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0
1140	1	5	3	0	444	2	9	27	6	6	0	4	7	0	1	1	5	0	3	5	2	1	0	1	1	0	0
1160	8	6	0	0	5	744	0	2	12	2	0	0	0	0	0	0	0	0	7	4	0	0	0	0	0	1	0
1180	5	6	4	0	13	5	73	9	19	1	0	2	2	0	0	0	1	0	2	6	0	3	1	1	0	0	0
1280	3	14	8	0	54	2	2	663	75	46	1	32	27	2	2	0	20	1	3	1	0	8	2	2	6	0	0
1281	3	19	2	1	9	13	13	88	208	2	0	11	2	1	0	0	3	0	2	9	5	12	1	4	3	3	0
1300	0	2	1	0	4	0	2	33	0	944	0	1	0	1	0	0	7	0	1	4	1	6	0	0	2	0	0
1301	0	2	0	0	0	1	2	3	5	0	134	3	4	0	0	1	2	0	0	0	0	1	0	1	1	0	1
1302	1	1	1	0	2	1	1	46	8	3	3	377	17	2	2	0	4	0	1	3	0	2	4	14	5	0	0
1320	1	1	1	0	3	0	0	28	8	1	2	7	520	13	17	3	19	0	0	2	0	9	6	5	2	0	0
1560	0	0	2	0	1	0	0	2	0	1	0	1	9	839	27	0	56	1	1	0	0	10	32	1	32	0	0
1920	2	0	0	0	0	0	0	4	0	1	0	0	11	18	785	1	27	0	1	1	0	2	6	1	1	0	0
1940	0	0	0	0	0	0	1	3	1	0	0	0	7	2	0	138	1	0	1	0	0	3	0	3	1	0	0
2060	1	1	3	0	6	3	1	16	3	1	1	6	17	57	32	0	791	1	1	1	0	10	22	2	23	0	0
2220	0	1	0	0	0	0	0	3	3	2	0	0	2	2	1	2	8	125	0	1	0	1	4	4	6	0	0
2280	67	18	0	0	1	3	0	0	1	2	0	1	1	0	0	1	0	0	780	62	0	2	0	0	0	13	0
2403	61	12	0	0	1	3	2	1	2	1	0	0	0	0	0	0	3	0	105	748	1	3	0	0	0	12	0
2462	1	20	19	8	5	4	0	3	6	2	0	0	0	1	0	0	0	0	1	8	204	1	0	0	0	1	0
2522	7	3	1	1	1	3	0	8	10	2	0	2	12	13	0	2	7	0	6	8	0	896	4	3	8	1	0
2582	1	0	0	0	2	0	0	4	1	1	1	3	10	44	11	1	38	3	1	1	0	3	348	10	33	2	0
2583	1	1	0	0	1	1	0	4	0	1	0	8	5	2	0	0	5	0	0	1	0	5	8	1984	15	0	0
2585	0	0	7	0	1	0	0	4	1	3	1	5	8	10	0	1	21	0	0	0	0	9	16	17	395	0	0
2705	75	2	0	0	0	1	0	1	2	0	0	0	0	0	0	0	0	0	19	11	0	0	0	1	0	440	0
2905	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	173

An extract of the classification repport shows that the top 6 best classified categories have an f1-score better than 90%, while the list of worst classified categories achieves a f1-score better than 50%:



Best Classified

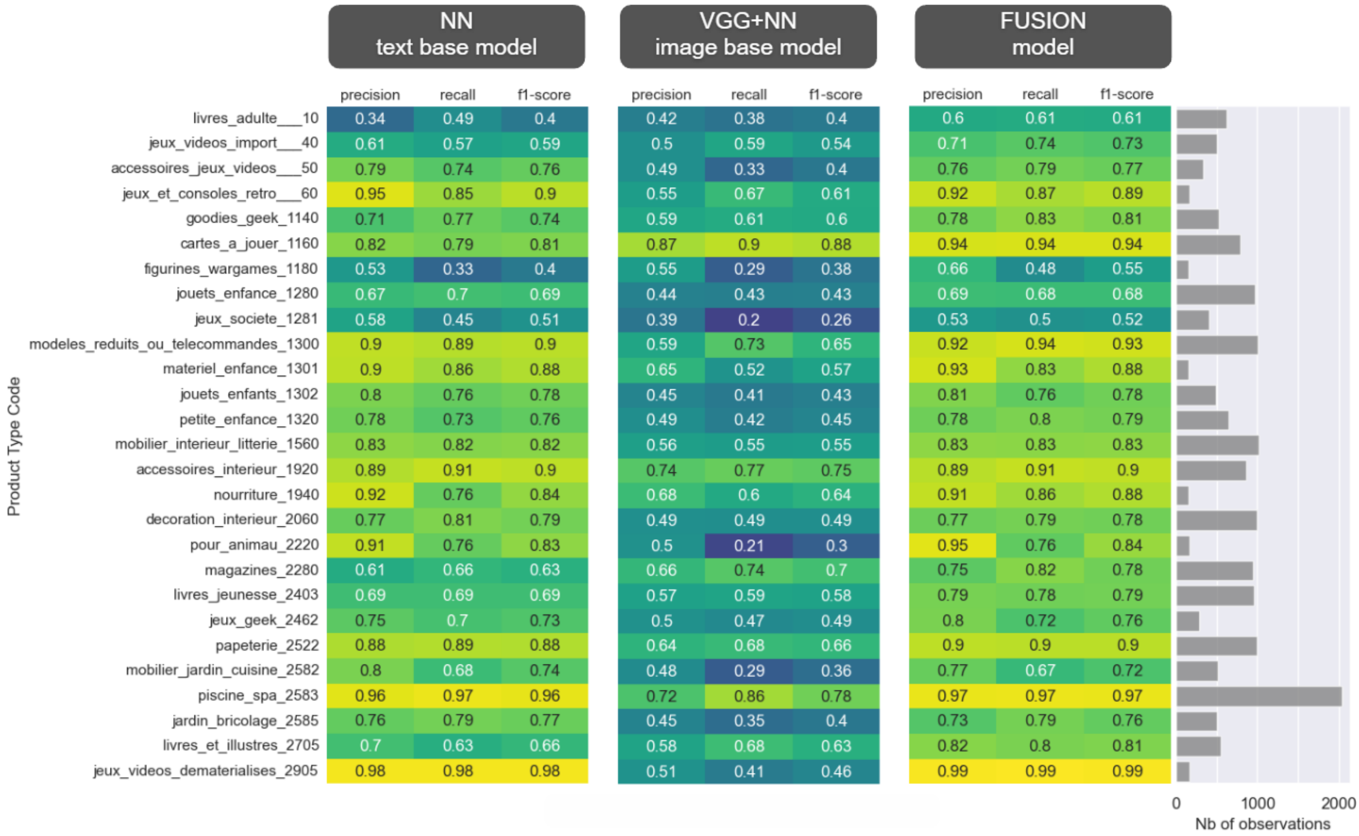
Category	precision	recall	f1-score
2905	0.99	0.99	0.99
2583	0.97	0.97	0.97
1160	0.94	0.94	0.94
1300	0.92	0.94	0.93
1920	0.89	0.91	0.9
2522	0.9	0.9	0.9
60	0.92	0.87	0.89



Worst Classified

Category	precision	recall	f1-score
1281	0.53	0.5	0.52
1180	0.66	0.48	0.55
10	0.6	0.61	0.61
1280	0.69	0.68	0.68
2582	0.77	0.67	0.72
40	0.71	0.74	0.73
2462	0.8	0.72	0.76

Finally, the figure below shows the detailed comparison of the classification report of the three best performing models. The values of the different metrcis have been color encoded such that the lighter the cell is, the better the model performs. This side-by-side comparison allows to easily spot the categories for which the fusion model has improved the classification, such as categories 10, 40, 1180, 2705, etc. There is almost no category for which the classification has been drastically deteriorated. Cases where the f1-score is deteriorated is only by a drop of about 1% on the metric value.



	NN text base model			VGG+NN image base model			FUSION model		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
livres_adulte___10	0.34	0.49	0.4	0.42	0.38	0.4	0.6	0.61	0.61
jeux_videos_import___40	0.61	0.57	0.59	0.5	0.59	0.54	0.71	0.74	0.73
accessoires_jeux_videos___50	0.79	0.74	0.76	0.49	0.33	0.4	0.76	0.79	0.77
jeux_et_consoles_retro___60	0.95	0.85	0.9	0.55	0.67	0.61	0.92	0.87	0.89
goodies_geek_1140	0.71	0.77	0.74	0.59	0.61	0.6	0.78	0.83	0.81
cartes_a_jouer_1160	0.82	0.79	0.81	0.87	0.9	0.88	0.94	0.94	0.94
figurines_wargames_1180	0.53	0.33	0.4	0.55	0.29	0.38	0.66	0.48	0.55
jouets_enfance_1280	0.67	0.7	0.69	0.44	0.43	0.43	0.69	0.68	0.68
jeux_societe_1281	0.58	0.45	0.51	0.39	0.2	0.26	0.53	0.5	0.52
modeles_reduits_ou_telecommandes_1300	0.9	0.89	0.9	0.59	0.73	0.65	0.92	0.94	0.93
materiel_enfance_1301	0.9	0.86	0.88	0.65	0.52	0.57	0.93	0.83	0.88
jouets_enfants_1302	0.8	0.76	0.78	0.45	0.41	0.43	0.81	0.76	0.78
petite_enfance_1320	0.78	0.73	0.76	0.49	0.42	0.45	0.78	0.8	0.79
mobilier_interieur_litterie_1560	0.83	0.82	0.82	0.56	0.55	0.55	0.83	0.83	0.83
accessoires_interieur_1920	0.89	0.91	0.9	0.74	0.77	0.75	0.89	0.91	0.9
nourriture_1940	0.92	0.76	0.84	0.68	0.6	0.64	0.91	0.86	0.88
decoration_interieur_2060	0.77	0.81	0.79	0.49	0.49	0.49	0.77	0.79	0.78
pour_animau_2220	0.91	0.76	0.83	0.5	0.21	0.3	0.95	0.76	0.84
magazines_2280	0.61	0.66	0.63	0.66	0.74	0.7	0.75	0.82	0.78
livres_jeunesse_2403	0.69	0.69	0.69	0.57	0.59	0.58	0.79	0.78	0.79
jeux_geek_2462	0.75	0.7	0.73	0.5	0.47	0.49	0.8	0.72	0.76
papeterie_2522	0.88	0.89	0.88	0.64	0.68	0.66	0.9	0.9	0.9
mobilier_jardin_cuisine_2582	0.8	0.68	0.74	0.48	0.29	0.36	0.77	0.67	0.72
piscine_spa_2583	0.96	0.97	0.96	0.72	0.86	0.78	0.97	0.97	0.97
jardin_bricolage_2585	0.76	0.79	0.77	0.45	0.35	0.4	0.73	0.79	0.76
livres_et_illustres_2705	0.7	0.63	0.66	0.58	0.68	0.63	0.82	0.8	0.81
jeux_videos_dematerialises_2905	0.98	0.98	0.98	0.51	0.41	0.46	0.99	0.99	0.99

Conclusions

In conclusion, tackling the Rakuten classification problem presented a unique set of challenges, as it required us to harness the power of two distinct data modalities: textual and image data. To effectively capture the complex relationships within the textual data, neural networks proved to be a formidable tool. Conversely, for the image data, which inherently exhibits spatial and hierarchical complexity, Convolutional Neural Networks (CNNs) emerged as the ideal choice.

Our fusion model effectively combines these two powerful tools. Utilizing neural networks, it carefully extracted intricate textual relationships, while simultaneously leveraging the transfer learning capabilities of CNNs to derive crucial image features. The combination was further enhanced by a head model made of dense layers optimized for multimodal data processing, resulting in a highly effective solution.

One important achievement of our model is its ability to efficiently manage memory when dealing with large sets of images, an often-daunting challenge for classical machine learning algorithms. It overcomes this hurdle, while ensuring that the model remains efficient even with significant image data.

The comprehensive metrics discussed in this report highlights the effectiveness of the fusion model as the solution to our classification problem. It outperformed its individual component models and other powerful classical machine learning algorithms, not only in terms of accuracy but also in computational efficiency and memory usage. These attributes position it as a good candidate for a potential online deployment.

Looking ahead, this multimodal classification model has the potential to play a pivotal role in more sophisticated recommendation systems within the e-commerce domain. Its versatility and robust performance make it a valuable asset for enhancing user experiences and driving personalized product recommendations. In essence, our journey in developing this model represents a significant stride towards the creation of smarter, more intuitive online shopping platforms.