

*Rapport Technique d’évaluation*

Classification des produits e-commerce (Rakuten)



Promotion : Bootcamp Datascientist janvier 2021

Participants :

Aylin KAYA

Damien LE DIRACH

Manh Hoa NGUYEN

Julien JUHEL

# Table des matières

[**Contexte**](#_wsf9ynu47mqy) **2**

[**Data**](#_jmb8lwuq6oua) **3**

[Les datasets](#_2zma4v561xpz) 3

[Nettoyage des données](#_qbuaxoiuwgcg) 4

[Premières analyses](#_jer4iaabmlbu) 4

[Nombre d’articles par classe pour y\_train](#_73gqwb9zr8ox) 4

[Echantillons d’images par classe](#_7vf4s2sbxcol) 5

[Identification des mots clés par classe](#_d2iwa61l29nl) 7

[**Projet**](#_q0io2ic0jqe7) **9**

[Modèles basés sur l’analyse lexical:](#_d1s3n48l8wq0) 9

[Best TF-IDF](#_bw3r28yq8ut8) 9

[Word2vec](#_644gpgo3nwlk) 9

[Modèles de machine learning](#_ezx6oi8uq9dz) 9

[Features engineering](#_4kagk9bkzi5q) 9

[Modèles ML appliqués](#_4ylfv8do6k17) 10

[Modèles de deep learning](#_q6oz4i651g0r) 10

[Architecture CNN pour l’analyse des images:](#_erknzbpjfrj0) 10

[Architecture DNN pour l’analyse du texte:](#_m1c51oxchsjg) 11

[Modèle de prédiction par vote](#_i36dl5nwhvml) 11

[**Difficultés rencontrées lors du projet**](#_ve70lxybcyq9) **12**

[**Bilan**](#_p70wkx2ll1b4) **13**

[Analyse des articles mal classés](#_qtt45rsjsm61) 13

[Paires de classes avec les confusions relatives les plus importantes (en %)](#_gn7rh9o3wks4) 13

[Paires de classes avec les confusions absolues les plus importantes](#_idkko8c51995) 13

[Top 20 des classes (réelles) générant le plus d’erreur](#_pv8bhm7jcf2w) 13

[Top 20 des classes (prédites) induisant le plus d’erreur](#_ud3rzxixdnlq) 14

[Préconisations et perspectives](#_pzyrsvxyzk8) 14

[Résultat au challenge RAKUTEN:](#_xny5efodfact) 15

# 

# Contexte

Ce projet porte sur le thème de la classification multimodale (texte et image) des codes de type de produit à grande échelle, et son objectif est de prédire le code de type de chaque produit tel que défini dans le catalogue de Rakuten France.

La mise en catalogue des listes de produits via la catégorisation des titres et des images, est un problème fondamental pour tout marché de commerce électronique, avec des applications allant de la recherche et recommandations personnalisées, à la compréhension des requêtes. Les approches de catégorisation manuelles et basées sur des règles ne sont pas évolutives, car les produits commerciaux sont organisés en de nombreuses classes. Le déploiement d'approches multimodales serait une technique utile pour les entreprises de commerce électronique car elles ont du mal à classer les produits en fonction des images et des étiquettes des commerçants, et à éviter la duplication, en particulier lors de la vente de produits neufs et d'occasion auprès de commerçants professionnels et non professionnels. Les progrès dans ce domaine de recherche ont été limités en raison du manque de données réelles provenant de catalogues commerciaux réels. Le projet présente plusieurs aspects de recherche intéressants en raison de la nature bruyante intrinsèque des étiquettes et des images des produits, de la taille des catalogues de commerce électronique modernes et de la distribution des données déséquilibrée typique.

# 

# Data

## Les datasets

Pour ce projet, Rakuten France a publié une liste d’environ 99K produits au format CSV, comprenant l’ensemble d’entraînement (84 916) et l'ensemble de test (13 812).

Les données sont divisées en deux critères, formant trois ensembles distincts: formation ou test, entrée ou sortie.

* X\_train.csv: fichier d'entrée d'entraînement
* Y\_train.csv: fichier de sortie d'entraînement
* X\_test.csv: fichier d'entrée de test

En outre, le fichier images.zip fourni contient toutes les images. La décompression de ce fichier fournira un dossier nommé “/images” avec deux sous-dossiers nommés image\_training et image\_test, contenant respectivement des images d'entraînement et de test.

La première ligne des fichiers d'entrée contient l'en-tête et les colonnes sont séparées par des virgules (","). Les colonnes sont:

* **identifiant** du produit - Cet ID est utilisé pour associer le produit à son code de type de produit correspondant.
* **désignation** - Le titre du produit, un court texte résumant le produit.
* **description** - Un texte plus détaillé décrivant le produit. Tous les marchands n'utilisent pas ce champ, donc pour conserver l'originalité des données, le champ de description peut contenir une valeur NaN pour de nombreux produits.
* **productid** - Un identifiant unique pour le produit.
* **imageid** - Un identifiant unique pour l'image associée au produit.

Les champs imageid et productid sont utilisés pour récupérer les images du dossier d'image respectif. Pour un produit particulier, le nom du fichier image est image\_imageid\_product\_productid.jpg.

## Nettoyage des données

Les données ont nécessité un nettoyage au niveau de la partie texte:

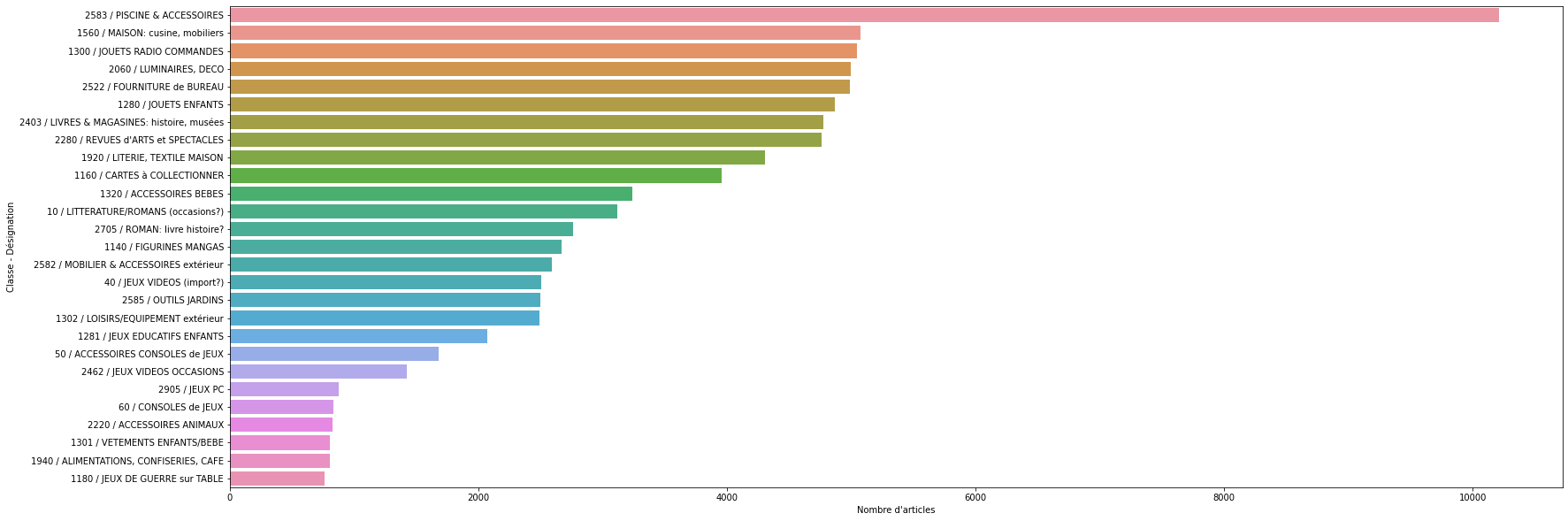
* présence de **balises html** (supprimées)
* présence de **caractères spéciaux** (remplacés)
* présence de **mots en langues étrangères** (pris en compte dans le traitement des stop\_words)
* 35% des articles n’ont pas de description (seulement une désignation et une image).

## Premières analyses

La cible se compose de 27 classes. Chaque article doit être affecté à une de ces classes.

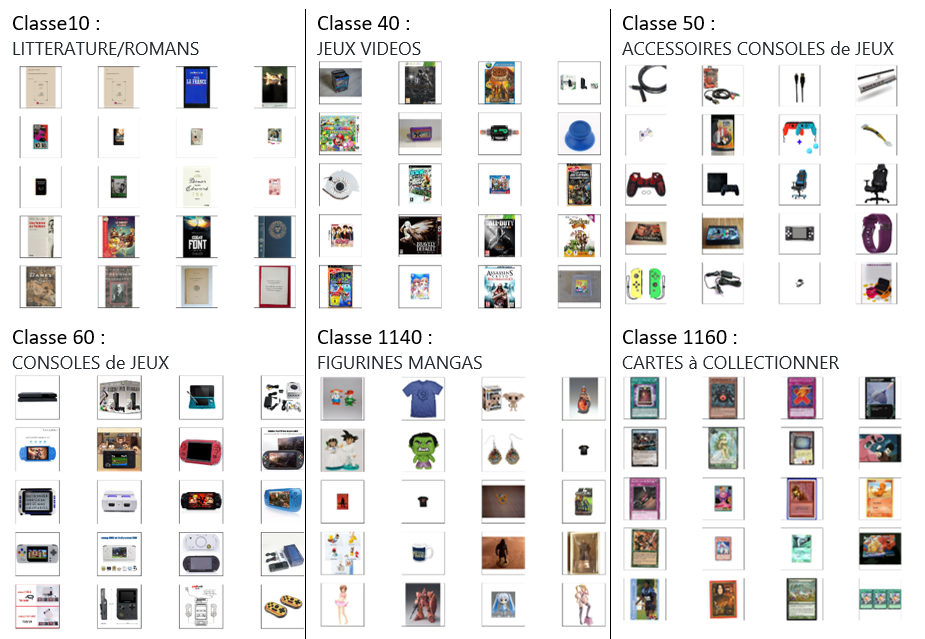
Afin de mieux appréhender le jeu de données, nous avons réalisé une série d’analyses et de représentations visuelles.

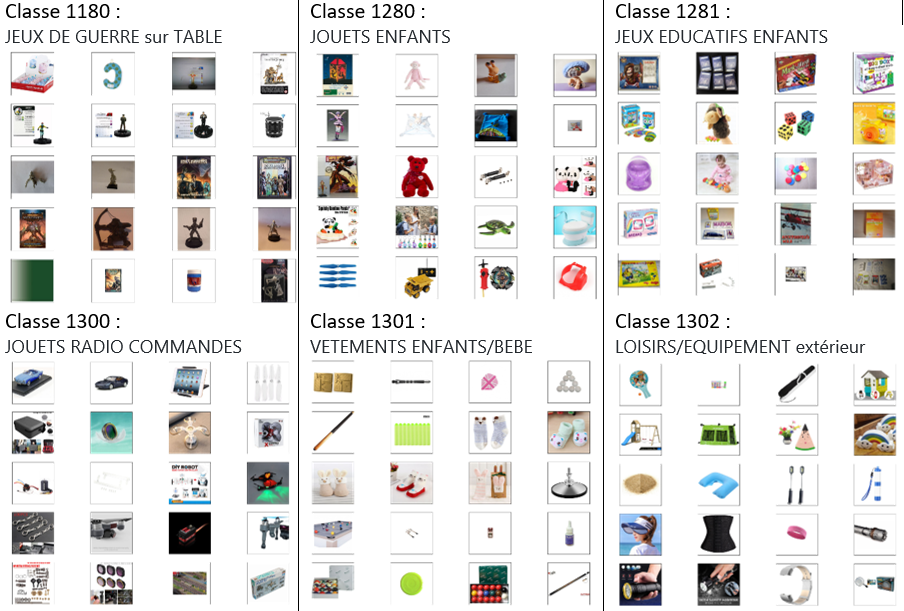
### Nombre d’articles par classe pour y\_train

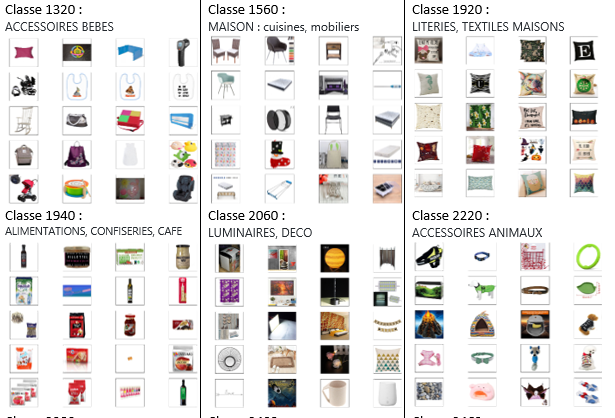


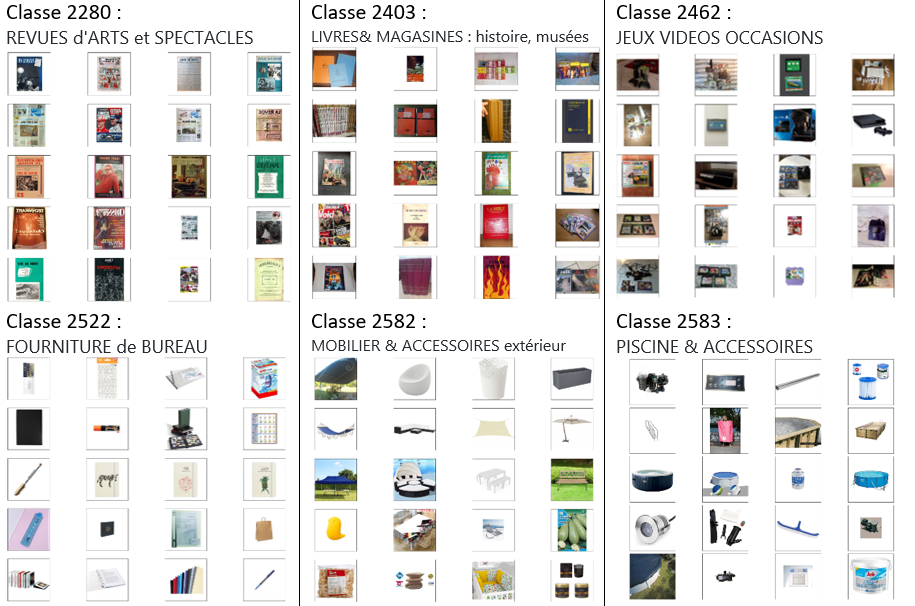
### 

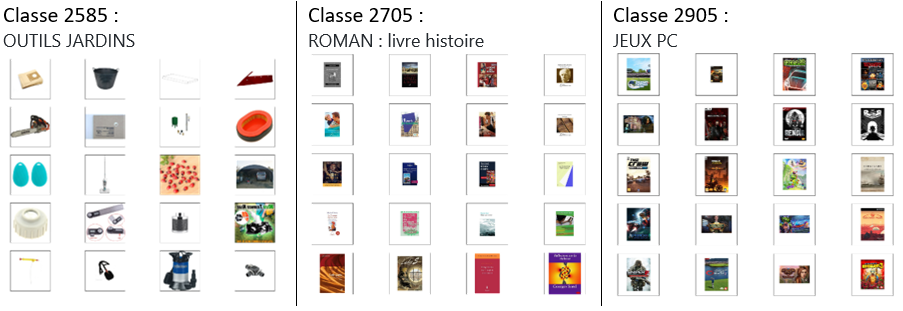
### Echantillons d’images par classe



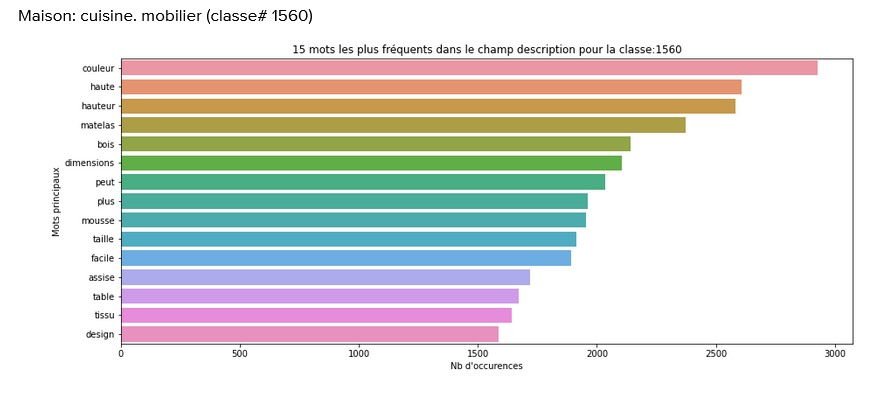


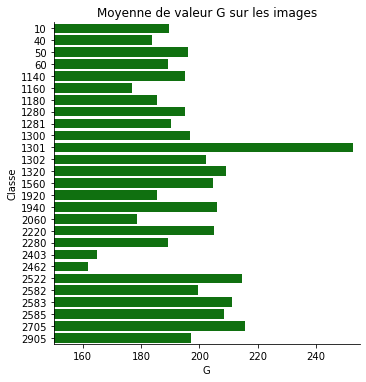
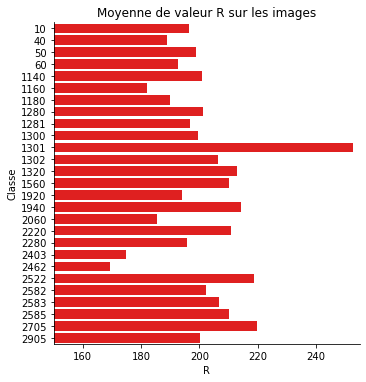
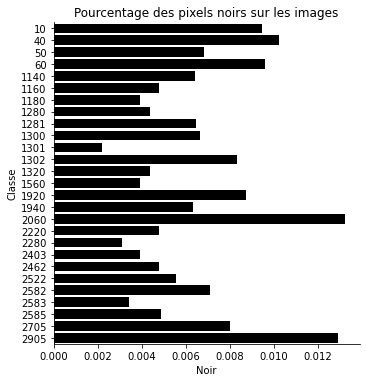
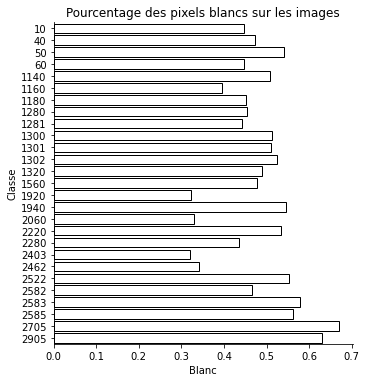


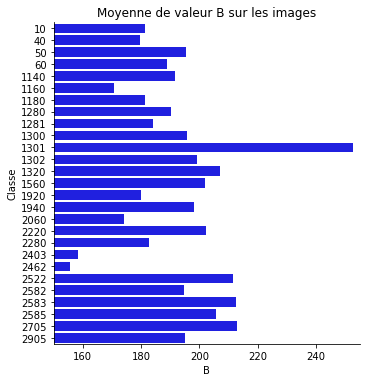




### Identification des mots clés par classe

*Identification des mots les plus présents dans la désignation et description des produits*

*Identification des caractéristiques principales des images par classe*



Grâce à ces différentes données, nous avons pu définir une première stratégie de prédiction basée sur des création de variables (features engineering) et l’application de modèles simples. Lors de la partie suivante, nous allons pouvoir détailler notre démarche, en présentant l'ensemble des approches développées, en partant des modèles les plus simples jusqu’aux méthodes de prédiction utilisant des réseaux de neurones.

# Projet

Pour réaliser ce projet nous avons réalisé et testé plusieurs modèles différents:

## Modèles basés sur l’analyse lexical :

### Best TF-IDF

Le premier modèle appliqué est un modèle simple basé sur l’analyse de la partie textuelle uniquement. L’algorithme développé est le suivant:

1. définition des 15 mots les plus représentatifs (et leur scores associés) de chaque classe en utilisant la méthode TF-IDF.
2. pour tous les articles testés, l’algorithme recherche dans le champ désignation si il y des mots présents dans les listes constituées à l’étape 1 et quand des mots sont trouvés additionne leur score pour la classe correspondante.
3. Une fois le champ désignation analysé, l’article est classé dans la classe ayant le score le plus important (best TF-IDF)

Ce modèle a permis d’obtenir une **accuracy de 0.48** (à comparer à l’aléatoire qui a une accuracy théorique de 0.037).

### Word2vec

Le deuxième modèle d’analyse lexical utilise l’approche Word2vec. Après une étape de création de features basée sur la moyenne des vecteurs représentant les mots d’après word2vec (300 dimensions).

Compte tenu du nombre important de variables (300 dimensions x 2 champs) Nous avons appliqué une ACP (95%) et ensuite nous avons entraîné un Random Forest (sans paramètre) qui permet d’atteindre une **accuracy de 55%**.

## Modèles de machine learning

### Features engineering

Après une étapes de création de features construites sur

* + la partie texte:
    - dénombrement de certaines REGEX
    - Scoring TF-IDF de chaque classe
  + la partie image:
    - moyenne des pixels en canal de couleur R
    - moyenne des pixels en canal de couleur G
    - moyenne des pixels en canal de couleur B
    - pourcentage de pixel Noir
    - pourcentage de pixel Blanc

### Modèles ML appliqués

Nous avons testé sur ces données les modèles et paramètres ci-dessous:

|  |  |  |
| --- | --- | --- |
| **Modèle** | **Paramètres** | **Accuracy obtenue** |
| GradientBoosting |  | **0.72** |
| SVM |  | **0.68** |
| Rég. Log. | liblinear | **0.55** |
| Random Forest |  | **0.77** |
| KNN | k = 3 | **0.75** |
| Voting - Soft | rf, lr, knn | **0.77** |
| Voting - Hard | rf, lr, knn | **0.65** |

## Modèles de deep learning

### Architecture CNN pour l’analyse des images:

Dans un premier temps, nous avons entraîné un modèle de ***LeNet*** qui comprend deux couches de convolution, deux couches de pooling, ainsi que des couches de dropout: le résultat atteint était une **validation accuracy de 42%**.

Dans un second temps, nous avons entraîné différents modèles de ***transfer learning*** avec le modèle de base VGG16:

* + - Un modèle de départ avec le VGG16 suivi par une couche Global Average Pooling, 3 couches de Dense et 2 couches de dropout
    - Un modèle avec le VGG16 en excluant sa couche
    - Un modèle avec le VGG16 en débloquant ses 4 dernières couches

Finalement, nous avons maintenu le modèle CNN image avec l'accuracy la plus élevée qui était construit par l'architecture ci-dessous:

* + - Un train générateur pour prendre les images de train avec la taille (224,224), class\_mode='sparse' et zoom\_range=**[0.5,1.0]**: nous avons défini ce zoom\_range en vue d'éliminer le cadre blanc qui peut aller jusqu'à ~70% d'image et qui existe sur plusieurs images dans le dataset
    - Un test générateur pour prendre les images de test avec la taille (224,224), class\_mode='sparse'et sans autre modification
    - Modèle de base VGG16 avec les poids imagenet
    - 1 couche de ***Global Average Pooling 2D***
    - 1 couche de ***Dense*** avec 1024 neurons, et activation 'relu'
    - 1 couche de ***Dropout*** avec le taux 0.2
    - 1 couche de ***Dense*** avec 512 neurons, et activation 'relu'
    - 1 couche de ***Dropout*** avec le taux 0.2
    - 1 couche de ***Dense*** avec 27 neurons, correspondantes au nombre de classes, et activation 'softmax'

L’étape de compilation & fit a été appliquée avec les paramètres et les callbacks ci-dessous:

*Paramètres de compilation:*

* + - * optimiseur: Adam
      * fonction de perte : sparse\_categorical\_crossentropy
      * métriques : accuracy

*Callbacks:*

* EarlyStopping qui suit le val\_loss pour le mode min
* ModelCheckPoint qui sauvegarde le modèle à la suite de chaque epoch accompli

En appliquant ce modèle on obtient une **validation accuracy de 58%.**

### Architecture DNN pour l’analyse du texte:

Un modèle DNN d’analyse du texte basé sur l’architecture ci-dessous:

* + 1 couche ***Embedding*** de 100 neurones
  + 1 couche de ***Flatten***
  + 1 couche ***Dense*** de 100 neurones avec une fonction d’activation ‘relu’
  + 1 couche de ***Dropout*** appliquant un ratio de 0.5
  + En sortie: 1 couche ***Dense*** de 27 **neurones** avec une fonction d’activation ‘softmax’

*Paramètres de compilation:*

* + - optimiseur: Adam
    - taux d’apprentissage: 0.001
    - fonction de perte : categorical\_crossentropy
    - métriques : accuracy

En appliquant ce modèle on obtient une **validation accuracy de 82.5%**.

## Modèle de prédiction par vote

Afin de traiter à la fois la partie texte et la partie image nous avons décidé d’exploiter au mieux les spécificités des différents modèles définis ci-dessus, par le biais d’un système de vote.

Pour cela nous avons calculé pour tous les articles, les moyennes pondérées (par la validation accuracy de chacun) des probabilités de chaque classe issues du predict\_proba des 3 modèles principaux (RF, CNN, DNN). Puis sur l’ensemble de ces probabibilités l’algorithme sélectionne la classe obtenant la probabilité moyenne la plus élevée.

En appliquant ce modèle nous avons atteint une **validation accuracy de** **84%**.

# 

# Difficultés rencontrées lors du projet

Les difficultés que nous avons rencontrées pendant le projet et notre perspective sur comment nous aurions pu les éviter:

* Durée d'entraînement des modèles avec beaucoup de features (word2vec)
* Mise en “production” de modèles développés dans des notebook indépendants: *rendre le code plus modulable, moins utiliser de notebook (développer sur des fichiers en .py)*
* Gestion des versions et partages des codes: *utiliser github dès le début du développement*
* Organisation non optimisée des dataset: r*egrouper tous les dataset (X\_train, y\_train et X\_test) dans une seule BdD*
* Taille de dataset image: nous avons travaillés sur *84 916* images de taille (500,500) qui faisait que le temps de téléchargement pour les analyser et le temps d'entraînement de modèle était assez importante
  + *travailler par classe des images était une solution que nous avons adoptée pour pouvoir réaliser les analyses et la datavisualisation*
  + *utiliser les images générateurs (flow\_from\_dataframe) nous a permis de faire passer cette taille des images par notre modèle sans les télécharger*
  + *par ailleurs, nous pourrions travailler sur des images ajustées par leur taille, couleur etc. pour permettre plus d'efficacité à leur traitement et à l'entraînement*
  + *aussi, utiliser les fonctions de TensorFlow qui sont plus rapide par leur fonctionnement parallèle pour télécharger les images*
* Problème de prédictions par le modèle CNN image: *garder le même astype des classes (surtout garder int si c'est des classes en int) pendant l'entraînement pour éviter que les classes se réorganisent alphabétiquement après la prédiction*

# 

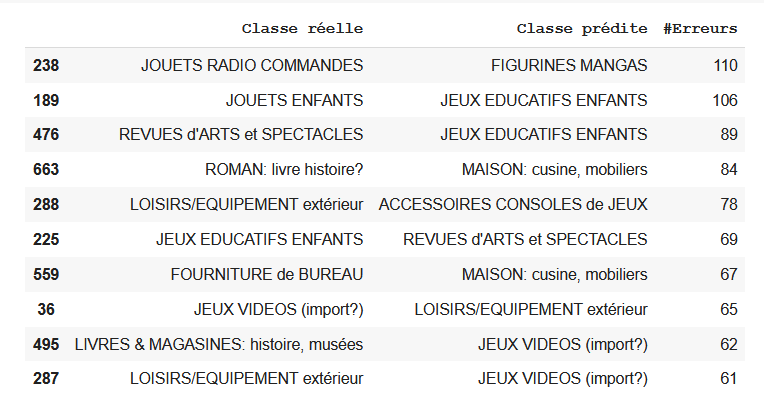
# Bilan

## Analyse des articles mal classés

### Paires de classes avec les confusions relatives les plus importantes (en %)



### Paires de classes avec les confusions absolues les plus importantes



### Top 20 des classes (réelles) générant le plus d’erreur

**

### Top 20 des classes (prédites) induisant le plus d’erreur



## Préconisations et perspectives

*Leviers métiers:*

* + *veiller à renseigner une description pour chaque article*
  + *veiller à bien traduire les descriptions et les désignations*
  + *Possibilité d’analyser la pertinence des classes générant beaucoup de confusion (classe avec TF-IDF identiques)*
  + *veiller à utiliser de bonnes images des articles qui sont à la fois efficaces pour leur classification aux catégories ainsi que leur présentation sur une site e-commerce*

*Le projet pourrait être poursuivi par l'entreprise qui sert comme un point de départ pour leur classification de produit en ligne, et peut être renforcé par les tests et contrôles pour s'assurer la correction de mal-prédictions. Notre perspective pour une potentielle amélioration comprend:*

* + *Possibilité d’affiner l'algorithme de voting pour une meilleure précision de prédiction*
  + *Analyse des articles mal classés pour un pré-traitement des données d'image et de texte adapté à chaque classe, qui améliorerait sûrement la performance des modèles*
  + *Produire un travail de pré-traitement d’images avant l’entraînement du modèle en Deep learning pour ressortir plus de features/caractéristiques liés à certaines classes d’images (Fond blanc, objets multiples, arrière-fond d’un objet) afin d’améliorer les performance du modèle de prédiction d’image.*

## 

## Résultat au challenge RAKUTEN:

Dans le cadre du projet nous avons participé au challenge data proposé par RAKUTEN par l'intermédiaire de l’ENS et du Collège de France.

Notre architecture de modèle nous a permis d’atteindre le top 15 de ce challenge avec un score (f1\_weighted) de 0.8425.

