Table des matières

[**Membres de l’équipe projet :** 1](#_Toc94001428)

[**Introduction** 3](#_Toc94001429)

[**La matière première : le Dataset** 3](#_Toc94001430)

[**Phase d’exploration du dataset mis à notre disposition** 4](#_Toc94001431)

[**Caractéristiques de la base de données:** 4](#_Toc94001432)

[**Données manquantes dans la database:** 8](#_Toc94001433)

[**Gestion des textes de produits écrits en langues étrangères** 8](#_Toc94001434)

[**Préprocessing du texte avec la librairie python nltk et Spacy d’autre part** 10](#_Toc94001435)

[**Tokenization et stopwords** 10](#_Toc94001436)

[**Stemming illustré ,testé mais non retenu** 10](#_Toc94001437)

[**Lemmatisation illustré et appliqué** 11](#_Toc94001438)

[**Détection et suppression des listes vides de tokens en fin du préprocessing texte :** 11](#_Toc94001439)

[**Le nom du fichier de sortie de la phase d’exploration et de préprocessing est X\_train\_rakuten\_afterEDA\_preprocessing.csv** 12](#_Toc94001440)

[**Préprocessing image** 13](#_Toc94001441)

[**Quelques cas explicites pour illustrer la difficulté de tirer de l’information différenciatrice de certaines images** 15](#_Toc94001442)

[**Tableau récapitulatif des performances f1-score pondéré (weighted) de chaque modèle évalué** 17](#_Toc94001443)

[**Présentation du modèle Machine Learning unimodal texte proposé** 19](#_Toc94001444)

[Pour la partie texte, nous avons testé trois approches : 19](#_Toc94001445)

[**1.1. Vectorisation : TfidfVectorizer vs CountVectorizer** 19](#_Toc94001446)

[**VotingClassifier** 23](#_Toc94001447)

[**1.3. LES CLASSES LES MIEUX/ MOINS BIEN PREDITES** 23](#_Toc94001448)

[VotingClassifier 27](#_Toc94001449)

[**Présentation du modèle Deep Learning multimodale (texte + image) proposé** 28](#_Toc94001450)

[**Que le début du chemin :** 33](#_Toc94001451)

# **Membres de l’équipe projet :**

Haeji YUN,

Mamadou LO,

Christophe Paquet

# **Introduction**

Ce rapport couvre les phases d’activités que notre groupe a mené sur le projet fil rouge de la formation de data scientist.

Ainsi, il est composé de l’étude préliminaire des données texte et image à exploiter, suivi de leur prétraitement.

Ce projet s’inscrivant dans le domaine du NLP qui fourmille d’innovations nous a permis de tester de nombreuses techniques particulièrement sur les méthodes d’embedding , extraction de features texte sous forme de vecteurs numériques dense.

L’approche méthodologique décomposée en une première phase baseline (réponse court terme très utile en début de projet) fut basée sur les algorithmes de Machine Learning de type différents (LR, LGB et RF) pour avoir une base intéressante de comparaison.

Note : Même si ce ne fut pas le cas ici, l’approche probabiliste avec un Naive Bayes par exemple devra toujours être considérée comme modèle avec du potentiel.

Ensuite, les expérimentations itératives, nous ont amené à développer d’une part des modèles de Deep Learning sur le framework tensorflow/keras de type séquentiel servant aux analyses unimodales puis fonctionnel pour bâtir une étude multimodale et d’autre part à utiliser la puissance de transfert learning ou d’un transformateur encodeur pour générer des vecteurs embedding sur le texte , features qui furent ensuite passées à un modèle ML de classification.

# **La matière première : le Dataset**

3 fichiers données téléchargés depuis le site du challenge **Rakuten France Multimodal Product Data Classification.**

<https://challengedata.ens.fr/challenges/35>

Les données textuelles : **~60 mb**

Données images : **~2.2 gb**

Pour l’entrainement des modèles :

X\_train\_rakuten\_update.csv et Y\_train\_rakuten\_CVw08PX.csv

Pour rendre les prédictions du modèle entrainé au jury du challenge:

X\_test\_rakuten\_update.csv

**Codes livrés sur github pour cette phase :**

<https://github.com/DataScientest-Studio/Rakuten_a_win_win_text_image_classification_in_PY>

le README.md apporte les informations utiles sur le projet ( livrables notebook et script python pour streamlit , rapports )

# **Phase d’exploration du dataset mis à notre disposition**

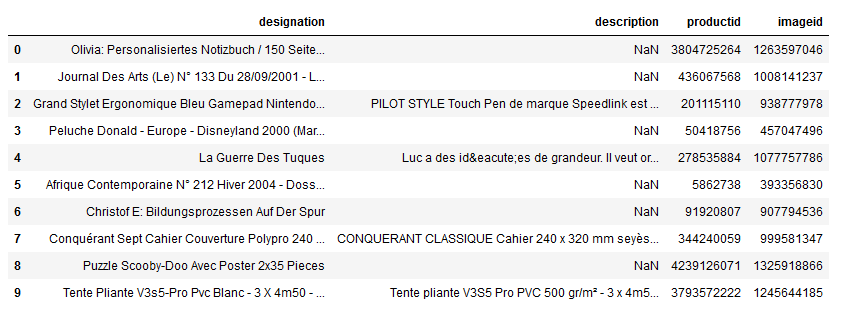
# **Caractéristiques de la base de données:**

Taille du dataset et vecteur classes d’apprentissage : (84916, 4)

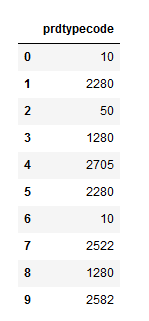
Donc 84916 échantillons et 4 features

Taille du dataset test : (13812, 4)

Début des données de la database :



Début du vecteur label cible des différentes classes dans lesquelles tout produit doit être classé :



Note : on voit de suite, qu’un nombre est certes très utile pour les algorithmes mais pas vraiment parlant pour nous, donc nous allons donner un nom représentatif à chacune de ces classes selon ce que celles-ci regroupent.

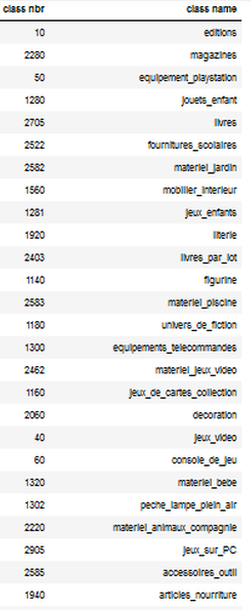
Il y a 27 classes dont voici la liste exhaustive :

[ 10 2280 50 1280 2705 2522 2582 1560 1281 1920 2403 1140 2583 1180

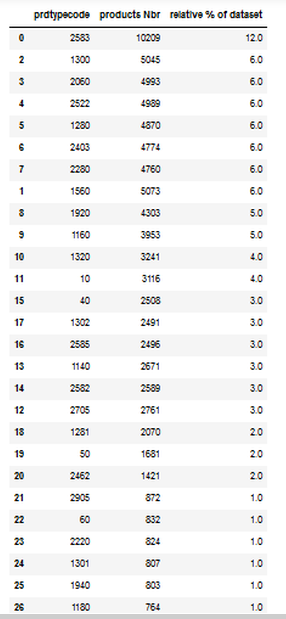
1300 2462 1160 2060 40 60 1320 1302 2220 2905 2585 1940 1301]

Note : Le nom ci-dessous a été choisi en regardant des dizaines d’images de chaque classe, le nuage de mots et l’histogramme de fréquences des tokens extrait du texte concaténé designation+description (si ce second était présent car manquant pour 35% des produits) des produits composant chaque classe.

**Note : Dans certains cas, le nommage ne fut pas aisé car il y a d’une part vraiment des similitudes, une part de recouvrement entre certaines classes (inter classe) et d’autre part, une variabilité des types de produits rangés au sein d’une meme classe. (intra classe)**



Etude de la répartition des produits parmi toutes les classes :



Note : La database est déséquilibrée avec par exemple la classe 2583 qui est sur représentée

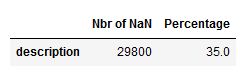
(12% de l’ensemble des articles) et 6 petites classes (1%) et 3 autres à 2%.

Cette remarque prend sa justification dans l’existence potentielle future d’un biais d’apprentissage favorable aux classes majoritaires car les algorithmes vont avoir tendance à mettre au point leurs paramètres (minimisation de la métrique d’erreur) sur la base des catégories les plus fréquentes.

Le TOP10 des codes produits (sur 27) équivaut ainsi à 64% de la database.

**Fusion des deux colonnes texte**

# **Données manquantes dans la database:**



Note : seul, le champ texte description n’est pas renseigné pour 35 % des cas de produits

Note : Les deux champs texte désignation et description seront fusionnés pour agréger le maximum d’informations sur chaque produit et optimisation la capacité de classification sur une base de **mots** **significatifs** la plus fournie possible.

Note : Au préalable, la variable description est travaillée en éliminant les balises html qui pollue le texte.

# **Gestion des textes de produits écrits en langues étrangères**

Que faire de la partie significative des textes en langue étrangère du dataset (17,6% dont 14,4 en anglais , 6% en allemand …) ?

Note : une colonne lang\_designation portant l’indication de la langue du texte va etre ajoutée au dataframe

fr 65123

en 12389

de 2783

nl 1061

ca 764

it 630

ro 338

es 317

pt 292

id 190

no 160

af 120

tl 114

da 89

et 79

sv 63

sw 61

so 45

sl 45

hr 42

cy 38

pl 35

fi 34

vi 33

tr 25

lt 11

hu 10

sq 7

sk 7

lv 6

cs 5

Name: lang\_designation, dtype: int64

Puis utilisation de google translate en restant dans les conditions d’utilisation (1000 traductions par heure et volumétrie journalière probablement) pour traduire en français les 19793 textes de leur langue étrangère (qui viennent d’être identifiés).

Idem respect de la taille maximale du texte à traduire par l’API Google qui est de 5000 caractères.

Note : présence potentielle encore d’un reliquat de mots en langue étrangère.

Ce cas possible n’a pas été investigué particulièrement car la réflexion est venue trop tardivement. La détection de la langue aurait dû être faite séparément sur les deux champs designation et description car il se peut que la fonction qui a déterminé la langue ait été leurrée en cas de mélange de plusieurs langues chacune héritée des deux champs 😊

**Exploration des données image**

Suivi de la régle de nomenclature des images fournie par le challenge Rakuten avec la création d’une nouvelle colonne nom\_img au dataframe de référence en lieu et place des deux champs productid et imageid

Note sur l’organisation du rangement des images.

Celles ci sont stockées dans un meme répertoire et non pas dans des sous répertoires par classe.(remarque significative pour le framework Deep Learning Tensorflow/Keras concernant le label de chaque échantillon qui ne sera pas déduit du sous répertoire)



**Constat global de la mise à disposition d’images de taille modeste :**

De 3ko à 105ko : images originelles téléchargées du challenge Rakuten.

Nombre d’images avec une taille inférieure à 50 ko : 45726.pour 38650 au-dessus.

Et là-dedans, on a meme 3095 images avec une taille inférieure à 10 ko.

Après le traitement des images pour supprimer la présence de nombreux cadres blanc, l’intervalle des poids des images passe de 0ko à 209 ko.

Donc, la définition des images couleurs (nombre de pixels en largeur et hauteur) est incontestablement petite.

Tout cela est bien loin de la qualité des appareils photo intégrés qui servent comme argument commercial aux nouveaux smartphones 😊

**Remarque sur les articles associés à une même image :**

En se basant sur le checksum des images, nous avons constaté un nombre important d’images identiques associées à hauteur de 7% des produits fournis. (5692 doublons)

Comme l’ampleur n’est pas négligeable, une étude spécifique a été faite sur le modèle multimodale choisi pour connaitre l’impact négatif ou pas de la présence de ces articles avec doublons image et le score f1 weighted est meilleur en gardant ceux-ci 😊

# **Préprocessing du texte avec la librairie python nltk et Spacy d’autre part**

# **Tokenization et stopwords**

La tokenisation est l’opération qui consiste à segmenter une phrase en unités « atomiques » : les tokens. Ce découpage se faisant en phrases, en mots voire meme en caractéres selon l’utilité dans le cas expérimenté.

stopword : on applique un filtre qui va supprimer les **mots outils** de la langue francaise qui ne portent pas d’informations de sens pour aider à classifier chaque produit.

# **Stemming illustré ,testé mais non retenu**

Ce processus a pour but de conserver la racine (donc principalement on tronque les suffixes et préfixes pour garder l’origine) . Il est plus simple que la technique de lemmatisation qui utilise un dictionnaire pour savoir reconnaitre , différencier nom et verbe pour appliquer un traitement différent adéquat.

['journal', 'art', 'art', 'marche', 'salon', 'art', 'asiatique', 'paris', 'jacques', 'barrere', 'francois', 'perrier', 'reforme', 'ventes', 'encheres', 'publiques', 'sna', 'fete', 'cent']

['journal', 'art', 'art', 'march', 'salon', 'art', 'asiatiqu', 'pari', 'jacqu', 'barrer', 'francoi', 'perrier', 'reform', 'vent', 'encher', 'publiqu', 'sna', 'fete', 'cent']

On voit bien sur les mots surlignés les effets de bords sur le sens des mots modifiés par cette technique (ville qui devient pari de jeu et vente qui devient un élément naturel )

# **Lemmatisation illustré et appliqué**

En surligné, on voit quelques modifications représentatives apportées par cette technique (féminin devenant masculin, pluriel en singulier, verbe conjugué en infinitif)

**On peut adapter cette transformation car peu d’effets de bords indésirables et cette technique va aider à diminuer le volume du dictionnaire BOW.**

['piscine', 'gonflable', 'pulverisation', 'forme', 'tortue', 'merveilleuse', 'piscine', 'pulverisation', 'offrira', 'divertissement', 'amis', 'familles', 'arriere', 'cour', 'faite', 'paroi', 'pvc', 'lamine', 'resistant', 'couches', 'piscine', 'gonflable', 'capacite', 'piscine', 'installee', 'maniere', 'rapide', 'suffit', 'poser', 'pataugeoire', 'surface', 'plane', 'gonfler', 'remplir', 'piscine', 'eau', 'piscine', 'raccordement', 'pompe', 'filtre', 'incluse', 'outre', 'replier', 'stockage', 'transport', "lorsqu'elle", 'utilisee', 'convient', 'enfant', 'partir', 'materiau', 'pvc', 'totales', 'diam', 'diametre', 'raccordement', 'capacite', 'paroi', 'laminee', 'couches', 'partition', 'epaisse', 'fonction', 'pulverisation', 'connectee', 'tuyau', "d'arrosage", 'installation', 'rapide', 'convient', 'enfant', 'partir']

['piscine', 'gonflable', 'pulverisation', 'forme', 'tortue', 'merveilleux', 'piscine', 'pulverisation', 'offrir', 'divertissement', 'ami', 'famille', 'arrier', 'cour', 'faire', 'paroi', 'pvc', 'lamine', 'resistant', 'couche', 'piscine', 'gonflable', 'capacite', 'piscine', 'installee', 'maniere', 'rapide', 'suffire', 'poser', 'pataugeoire', 'surface', 'plan', 'gonfler', 'remplir', 'piscine', 'eau', 'piscine', 'raccordement', 'pompe', 'filtr', 'inclus', 'outre', 'replier', 'stockage', 'transport', 'lorsque', 'lui', 'utilisee', 'convier', 'enfant', 'partir', 'materiau', 'pvc', 'total', 'diam', 'diametre', 'raccordement', 'capacite', 'paroi', 'lamine', 'couche', 'partition', 'epaiss', 'fonction', 'pulverisation', 'connectee', 'tuyau', 'de', 'arrosage', 'installation', 'rapide', 'convier', 'enfant', 'partir']

# **Détection et suppression des listes vides de tokens en fin du préprocessing texte :**

Tout prétraitement agit et génère des cas particuliers qu’il faut gérer.

Donc, une vérification a été mise en place en fin de phase pour détecter les textes trés trés courts qui se retrouvaient associés avec un vecteur nul , articles que l’on va considérer comme des exceptions (outlier).

Note : Tout mot rare c’est à dire avec une seule occurrence au sein d’une classe donnée a été supprimé (selon la culture de chaqu’un, à penser un peu comme un filtre de nettoyage de basse fréquence textuelle, ou filtre probabiliste de bruit gaussien )

nbr of rows to be suppressed due to designation after preprocessing is empty : 303

**on a donc supprimé ces 303 exceptions.**

**Note importante:**

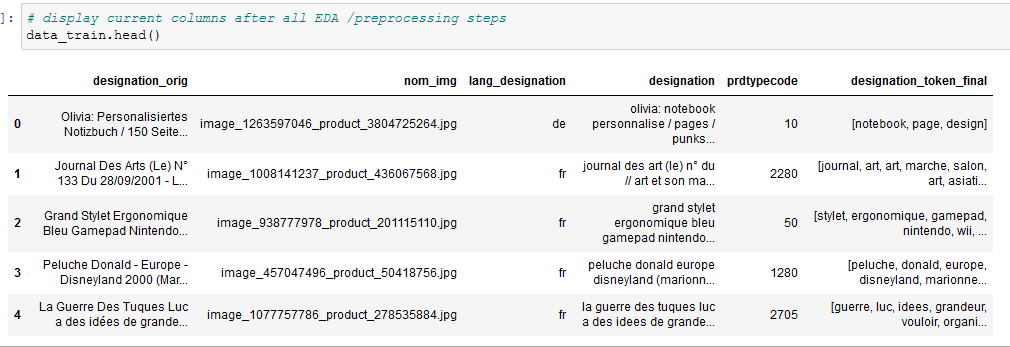
Comme des échantillons ont été supprimés **(**texte de ces produits vide suite au prétraitement appliqué),

en fin de notebook EDA-dataViz, le dataframe data\_train est sauvegardé dans un CSV post EDA.

Il contiendra la colonne prdtypecode qui sera à considerer comme le nouveau vecteur target.

**Le nom du fichier de sortie de la phase d’exploration et de préprocessing est X\_train\_rakuten\_afterEDA\_preprocessing.csv**

Celui-ci servira à l’entrée dans la phase de classification.



Principales colonnes :

designation : tous les textes en francais désormais

designation\_token\_final : liste des tokens déduits de designation

prdtypecode : classe d’appartenance du produit (cible à prédire)

nom\_img : image du produit

Colonnes historiques :

designation\_orig : textes multilangue

lang\_designation : langue détectée qui a servi pour sa traduction en francais

# **Préprocessing image**

Contexte et objectif :

En datascience, travailler sur des problématiques utilisant des images implique de savoir comment faire des transformations sur celles-ci en amont du traitement d'un modèle. Ces techniques ont pour but d'essentialiser l'information, de retravailler l'image pour obtenir une meilleure compréhension de la part de votre algorithme.

Dans notre projet, toutes les images comportent des bords blancs que nous allons supprimer.

**Partie uniquement significative sur le modéle Deep Learning multimodale qui exploite le texte ET l’image des produits Rakuten.**

Raison de la suppression de 237 articles :

Des fichiers image avec une taille inférieure à 2 kb sont considérés comme inexploitables, assimilés à du bruit et non de l’information qui pourrait etre utile au calcul des poids du réseau de neurones.

The file C:\Users\admin\Documents\projet\_Rakuten\cropped\_image\_train\\image\_953088054\_product\_216795580.jpg has a too small size of 942 bytes

The file C:\Users\admin\Documents\projet\_Rakuten\cropped\_image\_train\\image\_1142089742\_product\_884747735.jpg has a too small size of 0 bytes **(\*)**

number of deleted rows due to image size smaller than 2K : 237

new shape of dataset after preprocessing : (84376, 6)

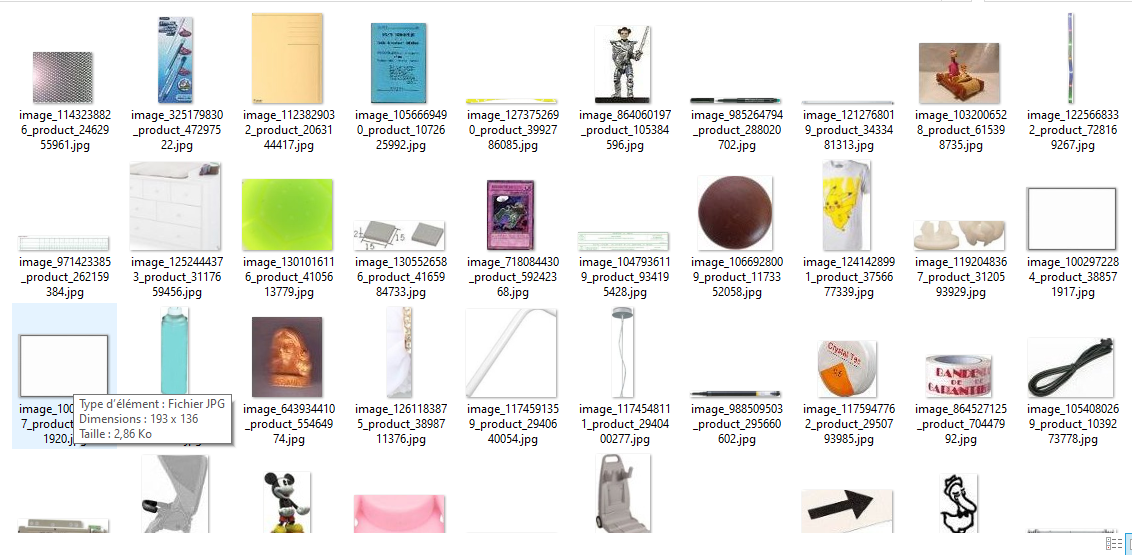
(\*) A l’origine, il n’y avait pas de fichiers image de taille nulle sur le site du challenge Rakuten MAIS l’opération de cadrage (cropping) qui fonctionne avec un seuil codé à 192 en l’occurence (valeur sur l’échelle des gris de valeur max 255) a eu comme léger effet de bord de produire 113 fichiers de taille nulle.

Voici deux exemples d’images pour illustrer la raison de la non détection des objets présents dans l’image : (oui, il y a bien quelque chose d’affiché 😊 )

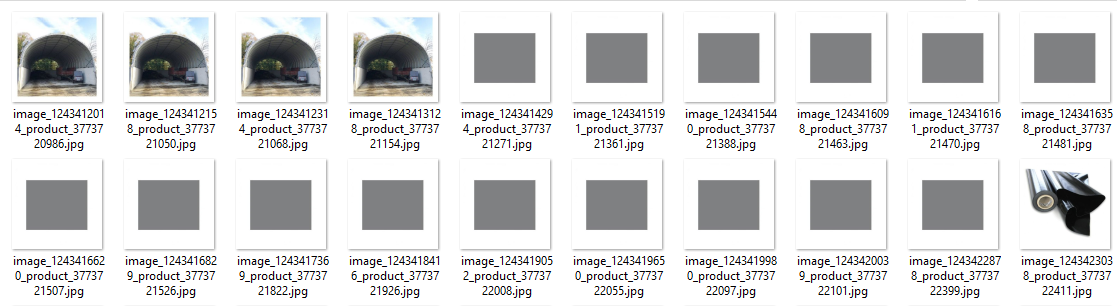


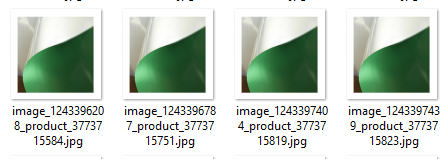


# **Quelques cas explicites pour illustrer la difficulté de tirer de l’information différenciatrice de certaines images**



Focus sur les trés petites images (4 dans la première ligne) et des motifs uniformes (deux dans la deuxième ligne)



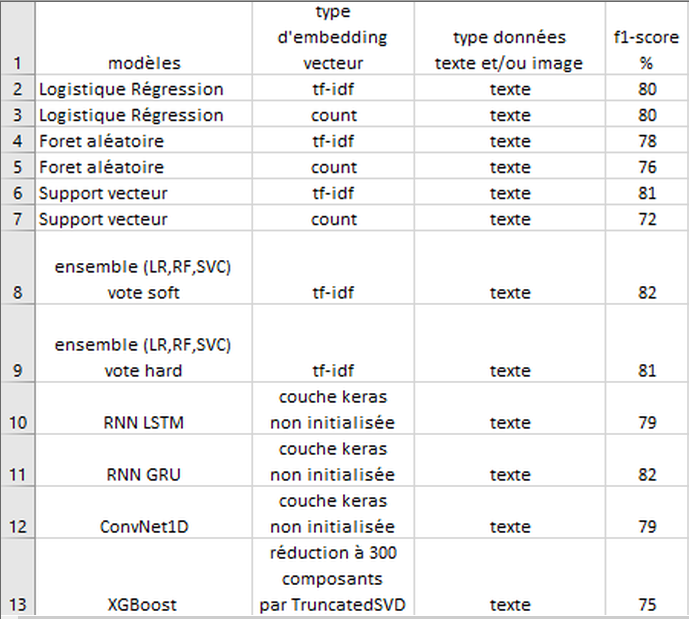


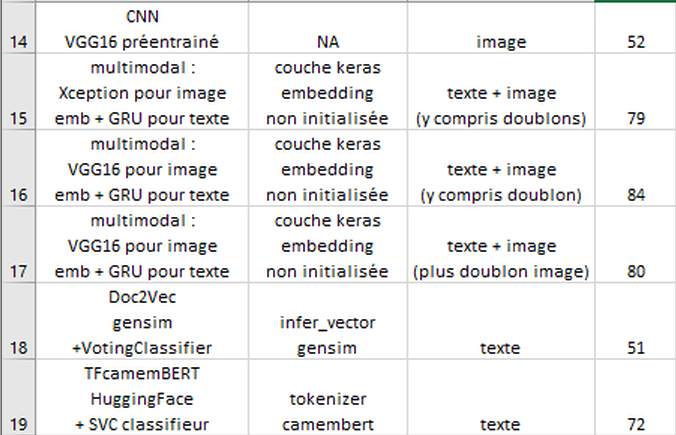
Quelques séries d’images pour illustrer que, sans contexte de description textuelle, pour nous aider, nous serions incapables de déduire ce qui est représenté sur ces images et à fortiori donc de les classer

**Sans approfondir, on comprend que la performance de la vision par ordinateur que l’on va implémenter avec différentes architectures de réseaux de neurones sera limitée par cette proportion d’images qui portent peu de features sur lesquelles s’appuyer pour modéliser une classification.**

# **Tableau récapitulatif des performances f1-score pondéré (weighted) de chaque modèle évalué**

Avant de focaliser sur les deux modèles que nous souhaitons proposer, le tableau ci-joint va parfaitement illustrer notre travail conséquent de recherche de modèles pouvant apporter une réponse performante de prédiction mais aussi notre montée en compétence et culture de ce domaine NLP dynamique.





# **Présentation du modèle Machine Learning unimodal texte proposé**

# Pour la partie texte, nous avons testé trois approches :

* Machine Learning Classique
* Deep Learning
* Combinaison de Machine Learning & Deep Learning

**1. Machine Learning Classique**

# **1.1. Vectorisation : TfidfVectorizer vs CountVectorizer**

Avant d’appliquer les algorithmes de machine learning, nous devons vectoriser le texte afin de le transformer en données numériques interprétable par l’algorithmes.

Parmi les différents méthodes de vectorisation, nous avons choisi :

* CountVectorizer : représentation de fréquence des mots dans le texte
* TFIDF Vectorizer : représentation de fréquence et de poids (importance) dans le texte

Pour cela, nous avons comparé CountVectorizer et TFIDF Vectorizer sur le texte sans stemmatisation.

Les Trois algorithmes classiques utilisés sont Régression Logistique, SVM, Random Forest, et puis Voting Classifier qui regroupe les trois algorithmes.

Nous constatons qu’en général, TFIDF Vectorizer est plus performant de 2% en terme de précision. Sa performance est plus marqué avec SVM avec une différence de 10% de précision en contrepartie au temps d’exécution plus long.

Par la suite, nous allons garder TFIDF Vectorizer

|  |  |  |
| --- | --- | --- |
|  | **TfidfVectorizer** | **CountVectorizer** |
| **Régression Logistique** |  |  |
| **SVM** |  |  |
| **Random Forest** |  |  |
| **Voting**  **Classifier** |  |  |

**1.2. Stemmatisation**

Maintenant, nous allons appliquer les algorithmes de machine learning classique sur le texte stemmatisé.

Avec la stemmatisation, la performance reste même avec au plus 1% de précision améliorée. Néanmoins le temps d’exécution est visiblement de réduite de 38%-53%.

Nous allons donc utiliser le texte stemmatisé avec TFIDF vectorizer.

Parmi les algorithmes, SVM et VotingClassifier restent les plus performanants avec une précision de 0,81. Logistique Régression a une performance similaire avec une précision de 0,81. Par contre nous remarquons une temps d’exécution énorme avec SVM et VotingClassifier.

Le bon compromis serait de garder Logistique Régression qui donne une performance similaire avec un temps d’exécution plus rapide.

|  |  |  |
| --- | --- | --- |
|  | **STEMMATISATION** | **SANS STEMMATISATION** |
| Régression Logistique |  |  |
| SVM |  |  |
| Random Forest |  |  |
| **VotingClassifier** |  |  |

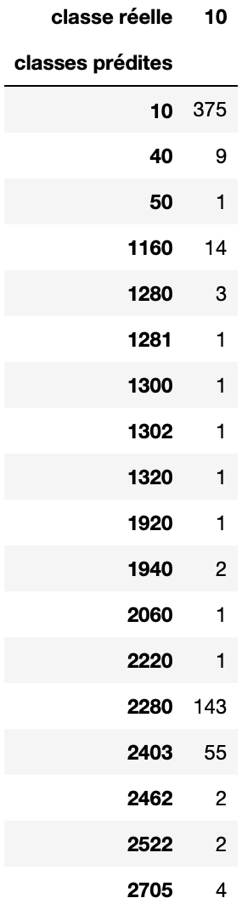
# **1.3. LES CLASSES LES MIEUX/ MOINS BIEN PREDITES**

Les classes les mieux prédites et les moins bien prédites restent les même dans les différents algorithmes essayés :

* Les mieux prédites sont les classes 2583 et 2905
* Les moins bien prédites sont les classes 10 et 1281

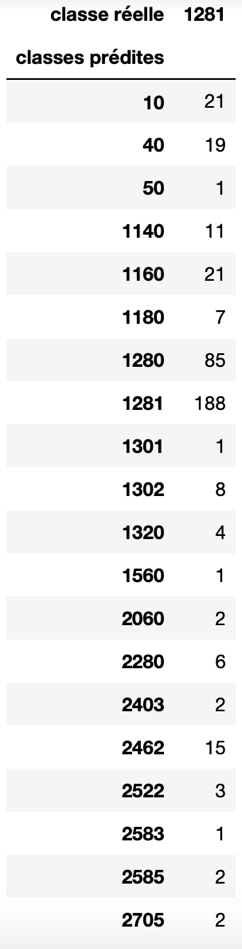
Les prédictions de la classe 10 :

Les fausses prédictions fréquentes sont les classes 2280, 2403, 2705. On remarque la similitude des classes, toutes liées à la lecture (10 éditions , 2280 magazines, 2403 livres\_par\_lot, 2705 livres)



Les prédictions de la classe 1281 :

Les fausses prédictions fréquentes sont la classe 1280. On remarque la similitude des deux classes, tous les deux liées au jeu d’enfants (1281 jeux\_enfant , 1280 jouets\_enfant)

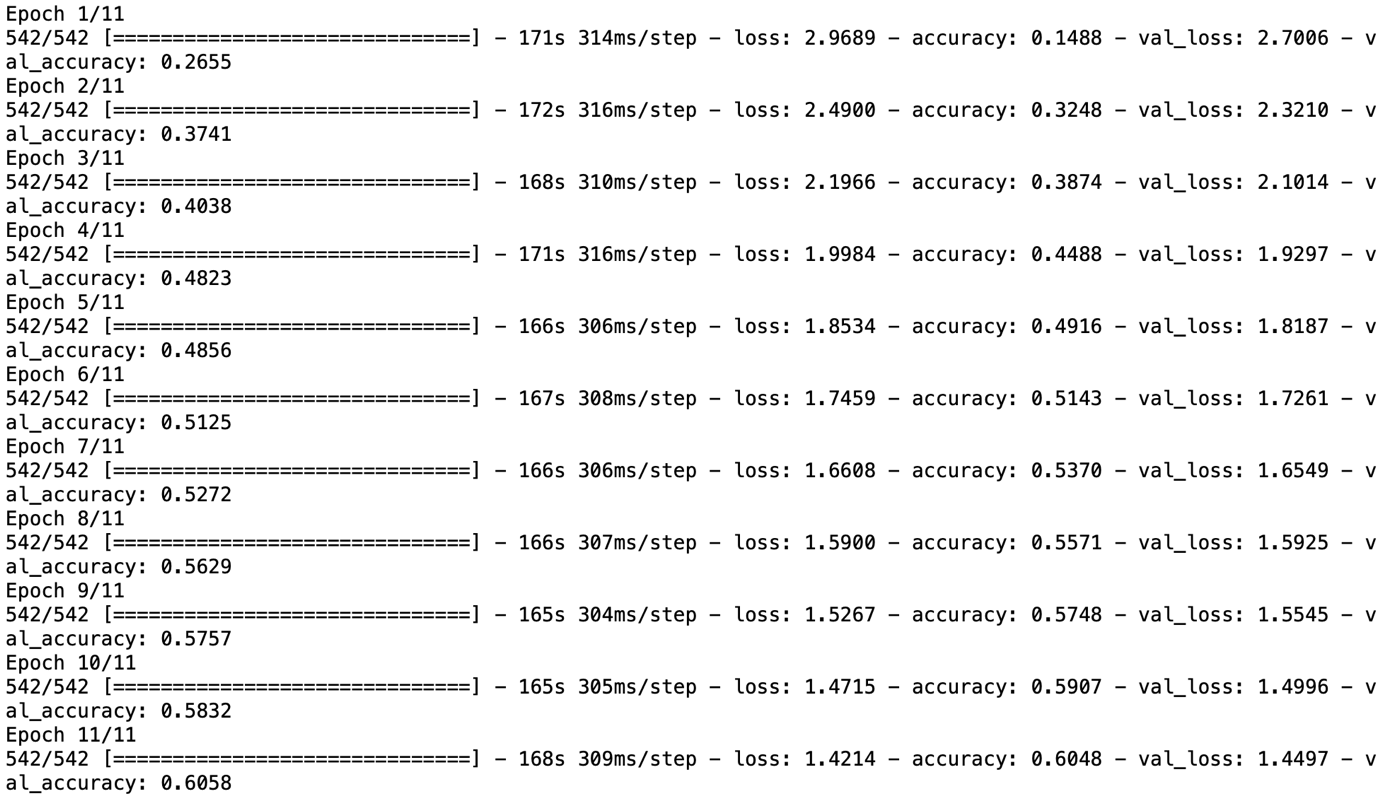


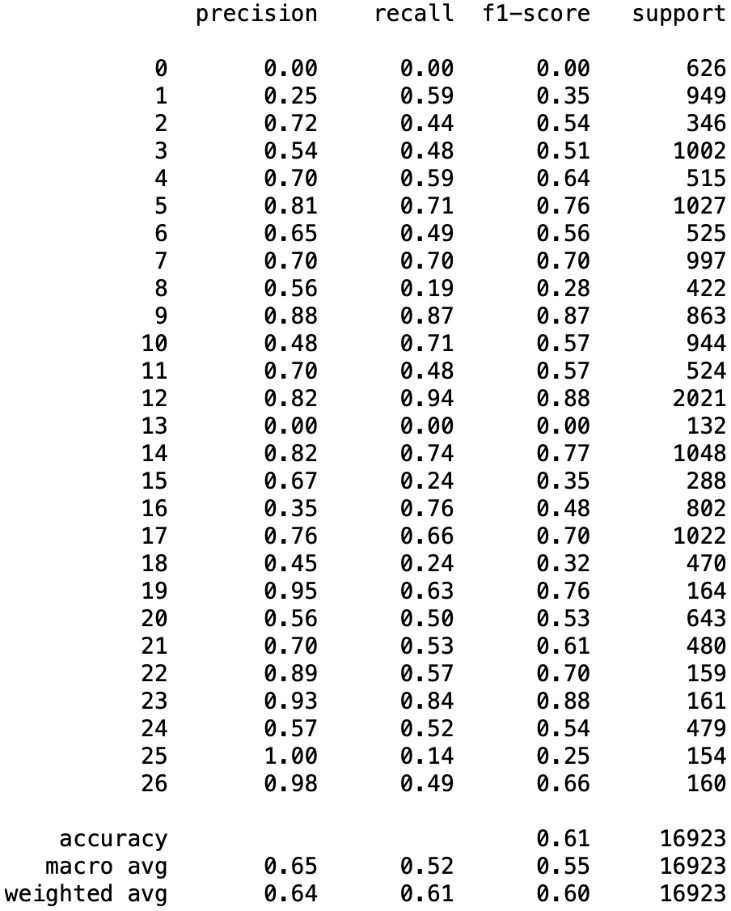
**2. Deep Learning – Word Embedding**

Pour la première tentation avec Word Embedding, nous avons choisi la taille de vocabulaire à 1000, 11 epochs

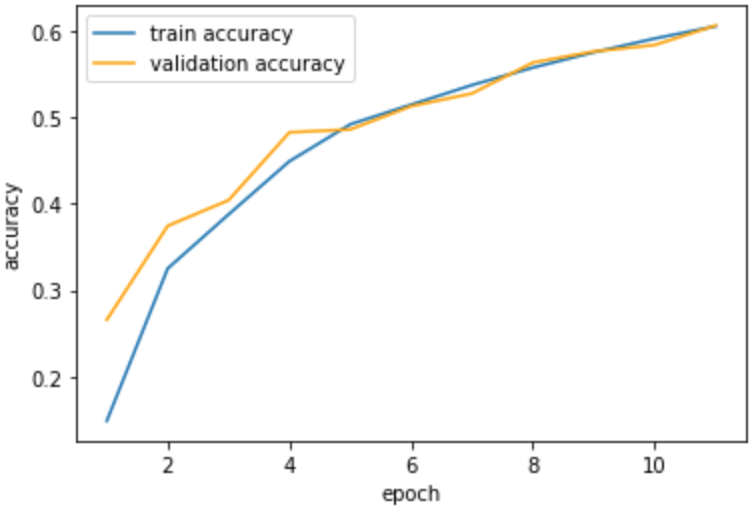
Vu le grand nombre de texte de notre data qui est à l’ordre de 50.000, le modèle ne prédit pas correctement et donne une précision de 0.6 et il y a une grande disparité de précision entre les classes.

Il faut donc augmenter la taille de vocabulaire pour les prochains entraînements.

****

****

La précision de l’échantillon de validation reste très proche à l’échantillon d’entraînement, donc il n’y a pas de problème de surapprentissage.

****

**3. Combinaison Machine Learning – Deep Learning**

Pour la combinaison de Machine Learning Classique et Deep Learning,

* Nous avons utilisé Doc2Vec pour générer le vecteur sur notre texte stemmatisé
* Nous avons appliqué nos algorithmes de machine learning classique

La combinaison de machine learning + deep learning est beaucoup moins performant que le machine learning classique malgré son temps d’exécution plus court.

|  |  |  |
| --- | --- | --- |
|  | **MIXTE** | **CLASSIQUE** |
| Régression Logistique |  |  |
| SVM |  |  |
| Random Forest |  |  |
| VotingClassifier |  |  |

# **Présentation du modèle Deep Learning multimodale (texte + image) proposé**

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

text\_input (InputLayer) [(None, 200)] 0 []

image\_input (InputLayer) [(None, 150, 150, 3 0 []

)]

embedding\_2 (Embedding) (None, 200, 640) 30656640 ['text\_input[0][0]']

vgg16 (Functional) (None, 4, 4, 512) 14714688 ['image\_input[0][0]']

gru\_2 (GRU) (None, 640) 2461440 ['embedding\_2[0][0]']

flatten\_4 (Flatten) (None, 8192) 0 ['vgg16[0][0]']

flatten\_5 (Flatten) (None, 640) 0 ['gru\_2[0][0]']

concatenate\_2 (Concatenate) (None, 8832) 0 ['flatten\_4[0][0]',

'flatten\_5[0][0]']

dense\_4 (Dense) (None, 64) 565312 ['concatenate\_2[0][0]']

batch\_normalization\_2 (BatchNo (None, 64) 256 ['dense\_4[0][0]']

rmalization)

activation\_2 (Activation) (None, 64) 0 ['batch\_normalization\_2[0][0]']

dropout\_2 (Dropout) (None, 64) 0 ['activation\_2[0][0]']

dense\_5 (Dense) (None, 27) 1755 ['dropout\_2[0][0]']

==================================================================================================

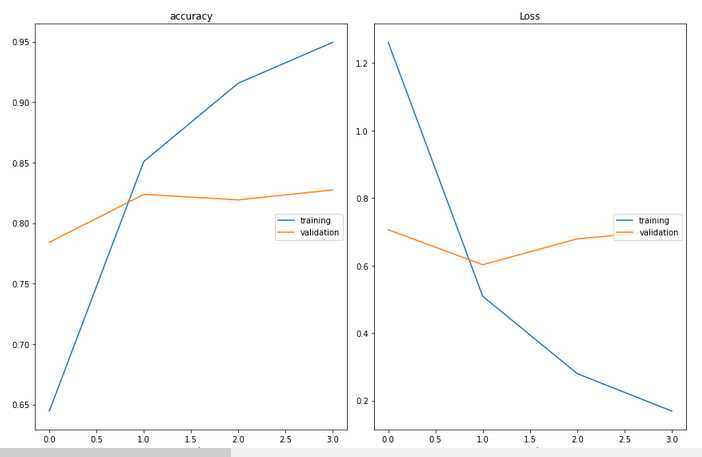
Total params: 48,400,091

Trainable params: 33,685,275

Non-trainable params: 14,714,816



Learning curve affiché en live :



accuracy

training (min: 0.645, max: 0.949, cur: 0.949)

validation (min: 0.784, max: 0.827, cur: 0.827)

Loss

training (min: 0.169, max: 1.262, cur: 0.169)

validation (min: 0.602, max: 0.706, cur: 0.705)

Note : l’outil d’affichage à la volée après chaque epoch a un décalage de -1 sur l’axe des abscisses qui représente les epochs donc le min de val\_loss est atteint en fin de l’epoch et correspond à une accuracy de 82,7%

Note : dans la version finale du script python Rakuten\_text\_image\_classification\_App.py qui est intégré au Web UI streamlite, j’ai préféré remettre les courbes offline basées sur l’historique.

Résultats du modèle :

Rapport de classification de scikit learn :

precision recall f1-score support

editions 0.48 0.72 0.58 285

magazines 0.71 0.88 0.79 472

equipement\_playstation 0.80 0.81 0.81 172

jouets\_enfant 0.72 0.70 0.71 471

livres 0.95 0.64 0.76 247

fournitures\_scolaires 0.95 0.90 0.93 538

materiel\_jardin 0.68 0.69 0.68 286

mobilier\_interieur 0.79 0.84 0.81 464

jeux\_enfants 0.70 0.44 0.54 210

literie 0.87 0.91 0.89 422

livres\_par\_lot 0.86 0.73 0.79 436

figurine 0.75 0.80 0.78 213

materiel\_piscine 0.98 0.97 0.97 1081

univers\_de\_fiction 0.67 0.39 0.50 74

equipements\_telecommandes 0.90 0.98 0.94 484

materiel\_jeux\_video 0.82 0.70 0.75 141

jeux\_de\_cartes\_collection 0.96 0.92 0.94 397

decoration 0.74 0.83 0.78 495

jeux\_video 0.76 0.77 0.77 244

console\_de\_jeu 0.98 0.79 0.88 76

materiel\_bebe 0.82 0.74 0.78 325

peche\_lampe\_plein\_air 0.85 0.81 0.83 244

materiel\_animaux\_compagnie 0.86 0.81 0.83 98

jeux\_sur\_PC 1.00 1.00 1.00 89

accessoires\_outil 0.79 0.78 0.78 240

articles\_nourriture 0.98 0.77 0.87 75

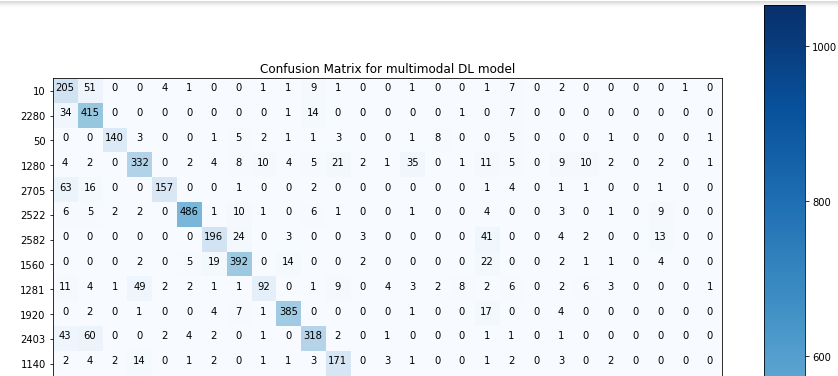
vetements\_nouveau-né\_et\_billard\_flechettes 0.96 0.95 0.95 75

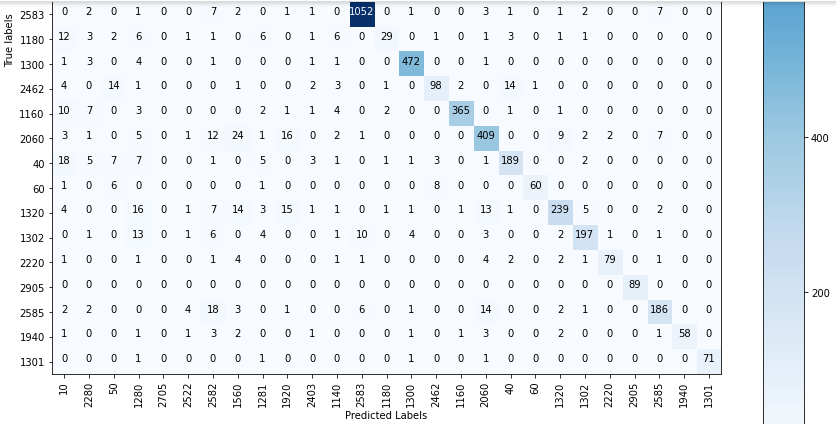
accuracy 0.82 8354

macro avg 0.83 0.79 0.80 8354

weighted avg 0.83 0.82 0.82 8354

Matrice de confusion :





**Les erreurs commises par le modèle sont somme toute logiques pour ceux qui ont eu à consulter la logique de classification fournie par Rakuten qui comporte vraiment** **une part de recouvrement entre certaines classes.**

mismatched classification listed when representing at least 10% of the right predictions amount of the class

class 10 (editions) wrongly predicted to class 2280 (magazines) , 51 for 205 correct predictions

class 1280 (jouets\_enfant) wrongly predicted to class 1300 (equipements\_telecommandes) , 35 for 332 correct predictions

class 2705 (livres) wrongly predicted to class 10 (editions) , 63 for 157 correct predictions

class 2582 (materiel\_jardin) wrongly predicted to class 1560 (mobilier\_interieur) , 24 for 196 correct predictions

class 2582 (materiel\_jardin) wrongly predicted to class 2060 (decoration) , 41 for 196 correct predictions

class 1281 (jeux\_enfants) wrongly predicted to class 10 (editions) , 11 for 92 correct predictions

class 1281 (jeux\_enfants) wrongly predicted to class 1280 (jouets\_enfant) , 49 for 92 correct predictions

class 2403 (livres\_par\_lot) wrongly predicted to class 10 (editions) , 43 for 318 correct predictions

class 2403 (livres\_par\_lot) wrongly predicted to class 2280 (magazines) , 60 for 318 correct predictions

class 1180 (univers\_de\_fiction) wrongly predicted to class 10 (editions) , 12 for 29 correct predictions

class 1180 (univers\_de\_fiction) wrongly predicted to class 1280 (jouets\_enfant) , 6 for 29 correct predictions

class 1180 (univers\_de\_fiction) wrongly predicted to class 1281 (jeux\_enfants) , 6 for 29 correct predictions

class 1180 (univers\_de\_fiction) wrongly predicted to class 1140 (figurine) , 6 for 29 correct predictions

class 2462 (materiel\_jeux\_video) wrongly predicted to class 50 (equipement\_playstation) , 14 for 98 correct predictions

class 2462 (materiel\_jeux\_video) wrongly predicted to class 40 (jeux\_video) , 14 for 98 correct predictions

class 60 (console\_de\_jeu) wrongly predicted to class 2462 (materiel\_jeux\_video) , 8 for 60 correct predictions

Mettre le focus sur les quelques classes particulièrement confondues ( ) en effectuant par exemple des classifications binaires entre celles-ci et itérer sur des variantes de préprocessing.

Pour le modèle doc2Vec dont la plus-value est basée sur l’exploitation du contexte des mots dans la phrase, travailler sur le texte naturel plutôt que celui reconstitué à partir des tokens

Pour les images, essayer d’utiliser durant la phase de préprocessing, des modéles expérimentaux tels que SwinIR (Image Restoration Using Swin Transformer) ou image-super-resolution (SRCNN-Tensorflow)

# **Que le début du chemin :**

Ce projet très intéressant qui, en parallèle des cours / exercices de la plateforme, a structuré notre formation de data scientist nous a donné une première expérience.

Désormais, pour devenir un professionnel en Data Science, nous devons rejoindre la communauté DatAlumni DataScientest car le partage de compétences, de news, de complicités avec certains membres favorise la synergie métier et un esprit de camaraderie 😊

Sous 3 à 6 mois, se forger un catalogue de 5 projets dans les domaines ou on souhaite se spécialiser, investissement qui va venir en appui pour décrocher un job DS junior (avoir son github public est un must)

**Il faut aussi définir une méthodologie d’amélioration continue (auto-formation) et l’exécuter (volonté et rigueur) carrément vitale dans ce métier qui s’appuie sur des technologiques dynamiques !!**

Axe pythonique :

Améliorer son niveau de programmation en python est la base de tout car c’est nécessaire pour produire un code robuste mais aussi pour tirer le meilleur parti des codes de la communauté open source.

Factoriser donc bien découper en fonctions mono tache avec une réflexion si un héritage de classes est possible

Etre rigoureux sur la qualité des commentaires (les revues croisées entre collégues développeur utiles)

(role de la fonction, ses paramètres d’entrée , ses méthodes, le format des données retournées)

Utiliser des éditeurs comme VSCode avec les extensions Linter pylint et mypy qui effectuent de nombreux controles qui, corrigées, permettent d’assurer un bon niveau de qualité du code.

Axe automatisation Data Scientist :

Travailler en mode pipeline sur les phases de préparation des données,

Sur la phase de modélisation avec l’adoption de dictionnaires de modèles prédéfinis, génériques par domaine.

Axe maitrise de la configuration optimale des modèles :

Rechercher une méthode de tuning des hyperparamètres non manuelle en Deep Learning (framework optuna ou autre ?)

En Machine Learning, ne pas se suffire de savoir configurer génériquement les modèles liés aux techniques d’ensembles mais investir spécifiquement car c’est souvent un passage obligé si des contraintes de performance de prédiction sont nécessaires pour atteindre les KPIs.

Axe MLOps :

S’apprendre à penser pour nos modéles développés R&D à leur finalité en intégrant une stratégie vis-à-vis de leur exploitation en production

(github code ET version des data utilisées pour entrainement ), KPIs de suivis de perf en live, analyse des nouvelles données dans le temps pour détecter l’existence d’une dérive de distribution, vérifier qu’aucune donnée (feature) est manquante ou erronée dans le temps

Axe veille technologique :

State Of the Art par domaine, par infrastructures cloud

Axe communication DS :

Travailler les méthodes d’interprétabilité des modèles

Créer un cycle d’amélioration de sa capacité de démonstration de ses modèles.

Prendre du plaisir !!