UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Master 2 MoSEF

Classification des produits Rakuten France : Étude comparative des méthodes textuelles et visuelles

Auteurs :
Tonin RIVORY
Aya MOKHTAR

Professeurs :
M. Amed COULIBALY
M. Roman YURSHAK

25 Avril 2025

1 Contexte et objectifs du projet

1.1 Contexte

Rakuten France est une place de marché d'e-commerce de premier plan en Europe, qui met en relation des millions d'utilisateurs avec une vaste gamme de produits couvrant l'électronique grand public, l'ameublement, la mode, la beauté et la papeterie. Pour chaque produit, la plateforme collecte et affiche plusieurs types d'informations : un titre ("designation"), une description détaillée lorsque disponible, et une image illustrative. De plus, la plateforme attribue à chaque article un code métier, le prdtypecode, correspondant à l'une des 27 catégories fonctionnelles définies par Rakuten.

1.2 Objectifs

L'objectif principal de ce projet est de construire deux modèles, l'un textuel et l'autre visuel, capables de prédire pour chaque produit de Rakuten France son code métier (`prdtypecode`). Plus précisément, nos modèles doivent :

- Analyser le titre et la description textuelle (lorsqu'elle est disponible) pour en extraire des caractéristiques discriminantes ;
- Exploiter l'image associée au produit pour capturer les informations visuelles pertinentes à la catégorisation ;
- Fournir en sortie, pour chaque modalité, une prédiction du `prdtypecode` assortie des probabilités sur l'ensemble des 27 classes.

2 Exploration des données

Avant de construire nos modèles, nous avons réalisé une analyse exploratoire des données afin de mieux comprendre leur structure, leurs biais et les prétraitements nécessaires.

2.1 Répartition de la cible : `prdtypecode`

La variable cible (`prdtypecode`) présente une distribution fortement déséquilibrée (Figure 1). Quelques catégories dominent largement l'échantillon (par exemple le code « 2583 » avec plus de 10 000 occurrences), alors que certaines n'apparaissent que dans quelques dizaines d'exemples (codes 50 ou 60). Ce déséquilibre nous a conduit à prévoir, dès l'entraînement, des stratégies de rééquilibrage (pondération de la loss, sur-échantillonnage ou under-sampling).

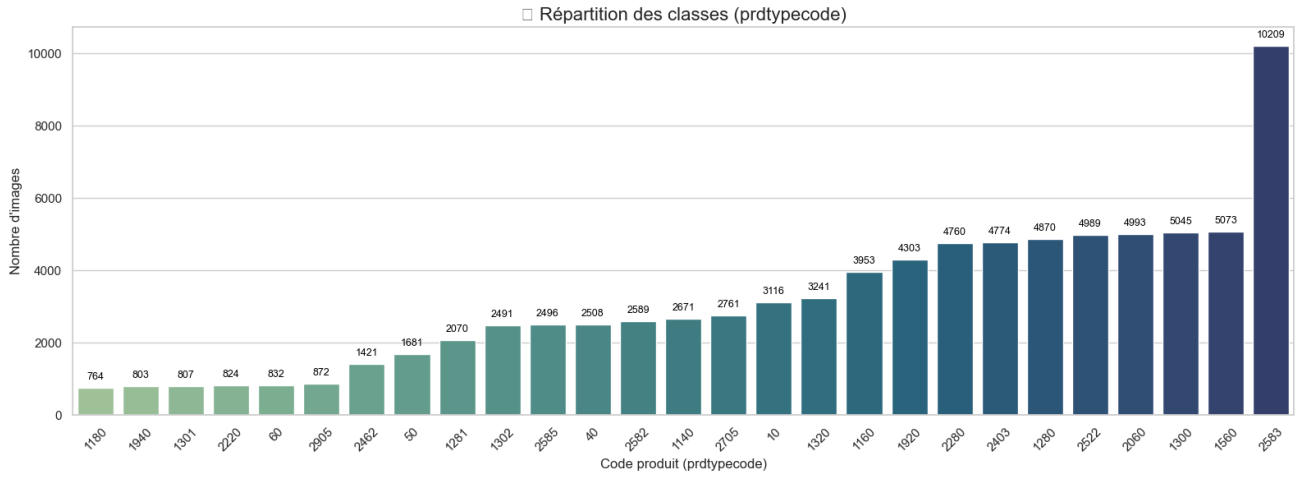

FIGURE 1 – Distribution des effectifs par `prdtypecode`.

2.2 Analyse du contenu textuel

Chaque produit dispose d'un titre (désignation) et, dans 65 % des cas, d'une description détaillée ; afin de tirer pleinement parti de ces informations, nous avons fusionné ces deux champs en un seul « texte complet », séparés par un espace. Cette unification enrichit le contexte sémantique en fournissant plus de mots-clés et d'exemples concrets, renforce l'apprentissage en évitant les séquences trop courtes et leurs limites informatives, et simplifie sensiblement le pipeline en ne requérant qu'une seule étape de tokenisation sur un champ unique.

2.3 Analyse du contenu des images

Pour évaluer la qualité visuelle des images, nous avons calculé un score de netteté basé sur la variance du filtre de Laplace sur un échantillon d'environ 10 000 images. Ce score permet de détecter les images potentiellement floues : plus la variance est faible, plus l'image manque de détails. L'analyse par classe a révélé que certaines catégories présentent systématiquement des images peu détaillées. Toutefois, nous avons choisi de conserver les images jugées floues par Laplace, car certaines représentaient simplement des produits uniformes ou peu texturés, sans être véritablement flous, comme sur la figure ci-dessous :

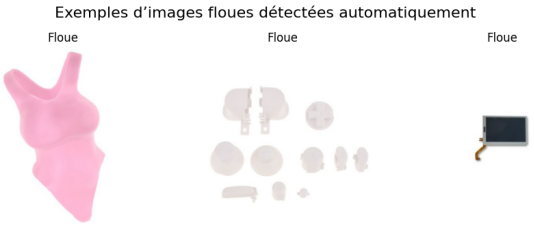

FIGURE 2 – Exemple d'images détectées comme flou

3 Preprocessing des données

3.1 Texte

Le corpus textuel, constitué de la fusion préalable des champs « designation » et « description », a été entièrement nettoyé afin de retirer tout élément non informatif. Nous avons d'abord converti l'ensemble du texte en minuscules et supprimé les accents. Puis, nous avons éliminé les balises HTML, les URL, la ponctuation et les chiffres isolés qui n'apportaient pas de valeur sémantique. Enfin, nous avons réduit les espaces multiples à un seul espace. À l'issue de ce prétraitement, chaque fiche produit est représentée par un texte prêt à être exploité par nos modèles par la suite.

3.2 Images

Compte tenu de nos ressources limitées, nous avons choisi de travailler sur un échantillon des données images afin de pouvoir entraîner nos modèles dans des conditions réalistes. Nous avons ainsi gardé 300 observations pour les 27 classes, pour un total de 8100 lignes de données avec des classes équilibrées. Pour garantir la qualité des images utilisées, nous avons mis en place un filtrage simple : les images trop petites, quasiment vides ont été automatiquement écartées.
Nous avons effectué notre modélisation selon deux méthodes d'entraînement distinctes :

- **Images en mémoire** : les images nettoyées sont chargées directement sous forme de tableaux NumPy. Testée sur un petit échantillon.
- **ImageDataGenerator** : les images sont lues dynamiquement depuis le disque au moment de l'entraînement. Cette méthode permet de traiter un peu plus de données malgré les limites mémoire. Elles sont testées sur un échantillon moyen.

Finalement, les modèles entraînés sur les images chargées en mémoire ont obtenu les meilleurs résultats. Nous avons donc retenu cette méthode pour nos expérimentations.

4 Modélisation

4.1 Modélisation textuelle

Modèles testés :

Pour la modélisation textuelle, nous avons exploré deux approches complémentaires : une méthode classique combinant **TF-IDF** et **régression logistique**, puis un réseau de neurones convolutionnel multicanal (**TextCNN**) finement ajusté.
Dans la première démarche, le texte nettoyé a été vectorisé à l'aide d'un TF–IDF configuré pour extraire jusqu'à 10 000 n-grammes (unigrammes et bigrammes), en filtrant les termes trop rares ou trop fréquents. Ces vecteurs pondérés, reflétant l'importance relative de chaque mot ou expression, ont servi d'entrée à une régression logistique multinomiale. En validation, ce pipeline a obtenu un F_1-score pondéré de **0,7829**. Il excelle sur les catégories majoritaires ($F_1 > 0{,}90$), mais reste limité pour les classes les moins représentées ($F_1 \approx 0{,}55$), soulignant la difficulté de capturer finement les subtilités contextuelles avec une approche purement *bag-of-words*.

Modèle retenu pour le texte : CNN multicanal (TextCNN)

Pour capturer au mieux les motifs contextuels du texte, nous avons mis au point un **CNN multicanal**. Après une couche d'Embedding de dimension 300 (vocabulaire limité à 50 000 mots) et une normalisation par BatchNormalization, quatre filtres convolutifs parallèles de tailles 2, 3, 4 et 5 extraient respectivement des bigrammes à pentagrammes. Chaque filtre génère 128 cartes de caractéristiques, régularisées par un terme L2, puis réduit via GlobalMaxPooling2D. Les quatre vecteurs résultants sont concaténés, passent par une couche dense de 256 unités (L2 + BatchNormalization), alimentent deux phases de dropout à 50 % et se terminent par une softmax sur 27 classes

Sur l'ensemble de validation, notre *TextCNN* atteint une accuracy de 0,8254 après 8 époques d'entraînement et un F_1-score pondéré de 0,8252, soit un gain de plus de quatre points par rapport à la combinaison TF–IDF + régression logistique ($F_1 \approx 0{,}783$). Les courbes d'apprentissage révèlent une montée rapide de l'accuracy d'entraînement (de 0,46 à 0,95) tandis que l'accuracy de validation plafonne autour de 0,82 dès la troisième époque. Un écart croissant apparaît entre les performances d'entraînement et de validation, signalant un phénomène de surapprentissage malgré les mécanismes de régularisation. La perte de validation se stabilise autour de 1,00 alors que celle d'entraînement continue de diminuer jusqu'à 0,44 : les callbacks `ReduceLROnPlateau` et `EarlyStopping` ont permis de stopper l'entraînement au bon moment et de restaurer les meilleurs poids (époque 8).
Enfin, l'analyse par classe montre des F_1-scores supérieurs à 0,90 pour les catégories majoritaires et autour de 0,70 pour les classes rares (contre $\approx 0{,}55$ avec la régression logistique).

Modèle	F_1-score pondéré
TF–IDF + Reg. Log.	0,7829
TextCNN	0,8252

TABLE 1 – Comparaison des performances

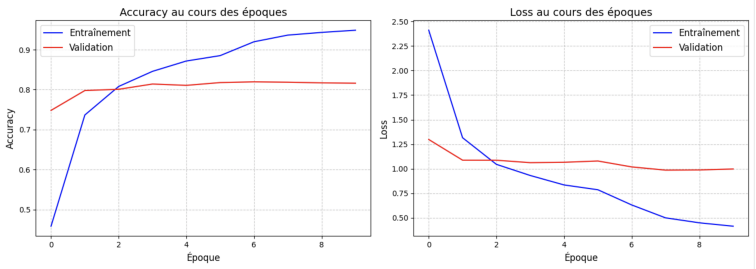

FIGURE 3 – Courbes de performances

4.2 Modélisation visuelle

Modèles testés :

Afin de prédire la catégorie des produits à partir d'images, nous avons exploré plusieurs architectures de réseaux de neurones pré-entraînés. L'objectif était de trouver un bon compromis entre performance de classification, temps d'entraînement et ressources matérielles disponibles.
Nous avons d'abord testé **EfficientNetB0**, un modèle réputé pour son efficacité et son faible coût computationnel. En parallèle, nous avons expérimenté **ResNet50**, un modèle très robuste, souvent utilisée comme référence dans la classification d'images.
Dans la pratique, ResNet50 a montré une capacité légèrement supérieure à extraire des caractéristiques discriminantes sur notre jeu de données.
Nous avons mis en place un entraînement en deux phases : une première phase avec toutes les couches gelées (*feature extraction pure*), suivie d'une seconde phase où seules les 27 dernières couches du modèle sont rendues entraînables pour un *fine-tuning* progressif.
Dans tous les cas, les callbacks `EarlyStopping` et `ReduceLROnPlateau` ont été utilisés afin d'éviter le sur-apprentissage et d'optimiser le processus d'entraînement.
Enfin, nous avons également testé un modèle **EfficientNetB2**, qui accepte des images en 260×260 pixels. Ce modèle, plus lourd, a été appliqué uniquement sur un petit sous-échantillon. Les résultats obtenu n'étaient pas à la hauteur des autres modèles testés.

Modèle retenu : ResNet50

Suite à nos expérimentations, nous avons retenu **ResNet50** comme modèle principal, car il a obtenu les meilleurs résultats, avec un *F1-score* pondéré de **0,5192**, contre **0,4805** pour l'architecture *EfficientNetB0*.
Le modèle a été entraîné directement en une seule phase, sans découpage intermédiaire. La structure finale du modèle comprend une couche de *GlobalAveragePooling*, suivie d'une couche dense avec une activation `softmax` pour produire les probabilités de classification des 27 catégories. L'entraînement a été réalisé à l'aide de l'optimiseur `Adam` et d'une fonction de perte `categorical_crossentropy`, bien adaptée à notre tâche multi-classe.
Cette approche simple, mais efficace, a permis d'exploiter la puissance de ResNet50 tout en l'adaptant à notre jeu de données spécifique avec de bonnes performances.
Concernant l'interprétabilité du modéle, les courbes d'*accuracy* montrent une progression régulière sans sur-apprentissage, avec une précision de validation atteignant environ 40 % et une perte (*loss*) qui diminue de façon stable. De plus, le rapport de classification confirme des performances solides sur plusieurs classes, avec une bonne répartition des scores entre classes fréquentes et classes rares.
En résumé, le modèle ResNet50 constitue une base efficace et stable, avec un fort potentiel d'amélioration via de l'augmentation de données ciblée ou l'utilisation d'une fonction de coût pondérée.

Modèle	F_1-score pondéré
ResNet50	0,5192
EfficientNetB0	0,4805

TABLE 2 – Comparaison des performances

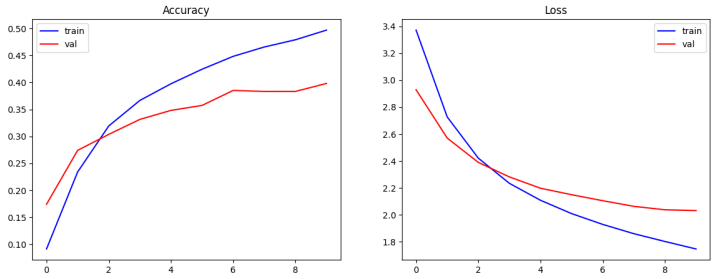

FIGURE 4 – Courbes de performances

5 Conclusion et axes d'amélioration

Ce projet nous a permis d'explorer des approches textuelles et visuelles pour prédire le code produit à partir des données Rakuten. Malgré des contraintes matérielles (pas assez de GPU), nous avons obtenu des résultats satisfaisants avec un TextCNN pour le texte et un ResNet50 pour les images. Pour aller plus loin, Plusieurs pistes peuvent être envisagées pour renforcer la performance globale du système. Tout d'abord, entraîner les modèles visuels sur l'ensemble des données disponibles, plutôt que sur un sous-échantillon, permettrait d'exploiter pleinement la richesse des images produits, en particulier dans les catégories fréquentes. Ensuite, l'adoption de techniques d'agrégation comme le voting ou le stacking entre plusieurs modèles complémentaires (ex. : TextCNN, TF-IDF + régression, ResNet50, etc.) pourrait améliorer la stabilité des prédictions et mieux capter la diversité des cas. Une étape clé consisterait également à concevoir un modèle multimodal fusionnant à la fois les informations textuelles et visuelles, afin de bénéficier de la complémentarité des deux modalités. Par ailleurs, reprendre l'entraînement dans un environnement disposant de ressources GPU permettrait de tester des architectures plus profondes et d'augmenter la taille des batchs, améliorant ainsi l'efficacité et la qualité de l'apprentissage.