



DATASCIENTEST × MINES PARIS - PSL

Machine Learning Engineer (BMLE) — Promotion Octobre 2025



# Classification Multimodale de Produits E-Commerce

Projet Rakuten France — Challenge de Classification Automatique  
*Approche Hybride Texte + Image avec Voting System*



## ÉQUIPE PROJET

Johan Frachon

Liviu Andronic

Hery Mickael Ralaimanantsoa

Oussama Akir

Mentor : **Antoine**

DataScientest — Paris

**84 916**

PRODUITS

**27**

CATÉGORIES

**92%**

ACCURACY IMAGE

**83%**

ACCURACY TEXTE

Février 2025

# Résumé Exécutif

---

Ce rapport présente notre solution de **classification automatique multimodale** développée dans le cadre du challenge Rakuten France. L'objectif était de classifier automatiquement des produits e-commerce parmi **27 catégories** en utilisant à la fois les données textuelles (titre et description) et visuelles (images produits).

Notre approche hybride combine un **classifieur textuel LinearSVC** (TF-IDF word+char, accuracy 83%) et un **Voting System d'images** fusionnant trois architectures complémentaires : DINOv3 (Vision Transformer), XGBoost sur features ResNet, et EfficientNet-B0. Ce système de vote atteint **92% d'accuracy** sur les images seules.

La fusion tardive (Late Fusion) des deux modalités avec pondération optimisée permet d'atteindre des performances robustes sur l'ensemble des 27 catégories, y compris les classes minoritaires grâce aux stratégies d'oversampling et de class weighting.

---

MOTS-CLÉS :

Classification Multimodale

Transfer Learning

Voting Classifier

TF-IDF

Vision Transformer

E-commerce

Deep Learning

**92%**

ACCURACY IMAGE (VOTING)

**83%**

ACCURACY TEXTE (SVC)

**27**

CATÉGORIES CLASSIFIÉES

## Contributions Principales

---

### Méthodologie

Pipeline complet de preprocessing texte et image avec gestion du déséquilibre de classes (ratio 1:13).

### Performance

Voting System innovant combinant 3 architectures pour atteindre 92% sur les images.

### Application

Interface Streamlit multimodale fonctionnelle avec visualisations et explicabilité.

# Table des Matières

## PARTIE I : CONTEXTE ET DONNÉES

1.1 Le Challenge Rakuten France	6
1.2 Description du Dataset	7
1.3 Analyse Exploratoire (EDA)	8

## PARTIE II : PREPROCESSING & FEATURE ENGINEERING

2.1 Pipeline de Prétraitement Texte	11
2.2 Pipeline de Prétraitement Image	13
2.3 Gestion du Déséquilibre des Classes	15

## PARTIE III : MODÉLISATION TEXTE

3.1 Benchmark des Classifieurs	17
3.2 Optimisation LinearSVC	18
3.3 Résultats Détaillés par Classe	19

## PARTIE IV : MODÉLISATION IMAGE

4.1 Stratégie Transfer Learning	21
4.2 Benchmark Machine Learning	22
4.3 Approche Deep Learning	24
4.4 Architectures Avancées (DINOv3, EfficientNet)	26
4.5 Voting System - Modèle Final	28
4.6 Tests de Robustesse	30

## PARTIE V : FUSION MULTIMODALE

5.1 Stratégie de Fusion Tardive	32
---------------------------------	----

5.2 Optimisation des Poids	33
----------------------------	----

5.3 Résultats Combinés	34
------------------------	----

## PARTIE VI : APPLICATION STREAMLIT

6.1 Architecture de l'Application	35
-----------------------------------	----

6.2 Fonctionnalités et Interface	36
----------------------------------	----

6.3 Démonstration	37
-------------------	----

## PARTIE VII : CONCLUSION ET PERSPECTIVES

7.1 Bilan du Projet	38
---------------------	----

7.2 Limites et Difficultés	39
----------------------------	----

7.3 Perspectives d'Amélioration	40
---------------------------------	----

## ANNEXES

A. Mapping des 27 Catégories	41
------------------------------	----

B. Configuration Technique	42
----------------------------	----

C. Références	43
---------------	----

# I

## Contexte et Données

*Challenge Rakuten France et analyse du dataset*

## 1.1 Le Challenge Rakuten France

Rakuten, géant mondial du e-commerce, fait face à un défi classique des marketplaces : la catégorisation automatique des produits mis en ligne par des vendeurs tiers. Une mauvaise catégorisation entraîne une mauvaise expérience de recherche et une perte de revenus significative.

### 🎯 Objectif du Challenge

Développer un modèle de classification multimodale capable de prédire le **code catégorie (prdtypecode)** d'un produit en utilisant simultanément :

- **Le Texte** : Désignation (titre) et description du produit
- **L'Image** : Visuel du produit fourni par le vendeur

### 1.1.1 Contexte Métier

#### Enjeux Business

- **Expérience utilisateur** : Navigation facilitée
- **Recherche produit** : Résultats pertinents
- **Conversion** : Réduction du taux de rebond
- **Scalabilité** : Millions de produits/jour

#### Défis Techniques

- **Multilingue** : Descriptions en plusieurs langues
- **Qualité variable** : Images non standardisées
- **Ambiguité** : Produits multi-catégories
- **Volume** : Traitement en temps réel

## 1.1.2 Métrique d'Évaluation

### F1-Score Pondéré (Weighted)

Conformément aux règles du challenge, la métrique principale est le **F1-Score weighted**, qui prend en compte le déséquilibre des classes en pondérant chaque classe par son support.

#### FORMULE F1-SCORE

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

$$F1_{\text{weighted}} = \sum (\text{support}_c / \text{total}) \times F1_c$$

## 1.2 Description du Dataset

**84 916**

IMAGES TRAIN

**13 812**

IMAGES TEST

**27**

CATÉGORIES

**500×500**

TAILLE IMAGES (PX)

### 1.2.1 Structure des Données

VARIABLE	TYPE	DESCRIPTION	COMPLÉTÉDUE
designation	String	Titre du produit	100%
description	String	Description longue (HTML/brut)	65%
productid	Integer	Identifiant unique produit	100%
imageid	Integer	Identifiant image associée	100%
prdtypecode	Integer	Code catégorie (target)	100%

### Valeurs Manquantes Critiques

**35% des descriptions sont manquantes (NaN).** Cette contrainte structurelle nous oblige à concevoir un pipeline robuste qui ne dépend pas uniquement du champ description.

**Stratégie retenue :** Concaténation designation + description avec remplacement des NaN par chaîne vide.

## 1.2.2 Aperçu des Catégories

Le dataset couvre 27 catégories de produits e-commerce diversifiées :

### Livres & Médias

Livres neufs/occasion, magazines, BD, livres anciens

### Gaming & Jouets

Jeux vidéo, consoles, figurines, cartes, jouets

### Maison & Jardin

Mobilier, décoration, literie, piscines, bricolage

## 1.3 Analyse Exploratoire (EDA)

### 1.3.1 Déséquilibre des Classes

L'analyse de la distribution des catégories révèle un **déséquilibre significatif** constituant l'un des défis majeurs du projet.

**13.4<sup>×</sup>**

RATIO MAX/MIN

**10 217**

CLASSE MAJORITAIRE

**761**

CLASSE MINORITAIRE

**3 145**

MOYENNE/CLASSE

RANG	CODE	CATÉGORIE	EFFECTIF	%	DISTRIBUTION
1	2583	Piscines et accessoires	10 217	12.0%	<div style="width: 12.0%; background-color: red;"></div>
2	1560	Mobilier intérieur	5 076	6.0%	<div style="width: 6.0%; background-color: red; float: left; margin-right: 10px;"></div> <div style="width: 94.0%; background-color: lightgray; float: right;"></div>
3	2060	Décoration intérieure	4 996	5.9%	<div style="width: 5.9%; background-color: red; float: left; margin-right: 10px;"></div> <div style="width: 94.1%; background-color: lightgray; float: right;"></div>
		...			
26	1940	Alimentation	804	0.9%	<div style="width: 0.9%; background-color: yellow; float: left; margin-right: 10px;"></div> <div style="width: 99.1%; background-color: lightgray; float: right;"></div>
27	60	Consoles de jeux	761	0.9%	<div style="width: 0.9%; background-color: lightgray; float: left; margin-right: 10px;"></div> <div style="width: 99.1%; background-color: lightgray; float: right;"></div>

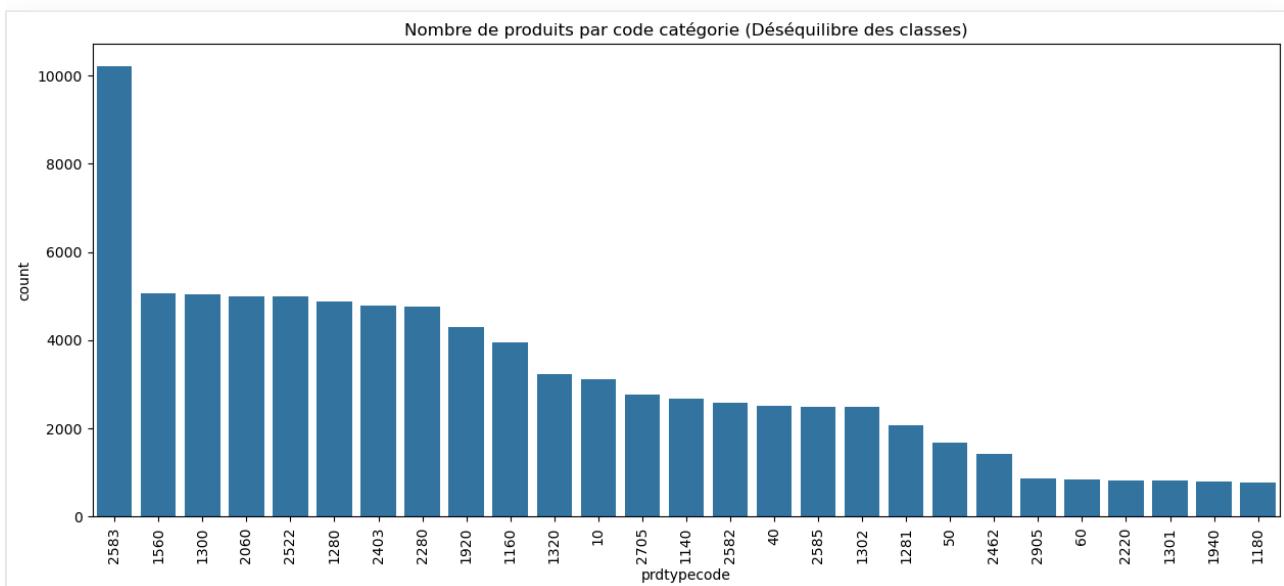


Figure : Distribution des 27 catégories - Déséquilibre significatif (ratio 1:13)

### 1.3.2 Analyse Textuelle

#### Distribution des Longueurs

#### Mots les Plus Fréquents



Les termes dominants corrélatifs avec les classes majoritaires (piscines, jouets).

CHAMP	MOYENNE	MÉDIANE	MAX
Désignation	70 car.	65 car.	250 car.
Description	450 car.	200 car.	12 000 car.

### 1.3.3 Analyse des Images

#### Images Standardisées

Toutes les images sont uniformes en **500x500 pixels** au format JPEG. Cependant, la présence de **bordures blanches variables** autour des produits nécessite une attention particulière lors du preprocessing.

#### Variabilité Intra-Classe

Grande diversité visuelle au sein d'une même catégorie. Exemple : un "livre" peut apparaître comme une couverture seule, une pile de livres, ou une image marketing composite.

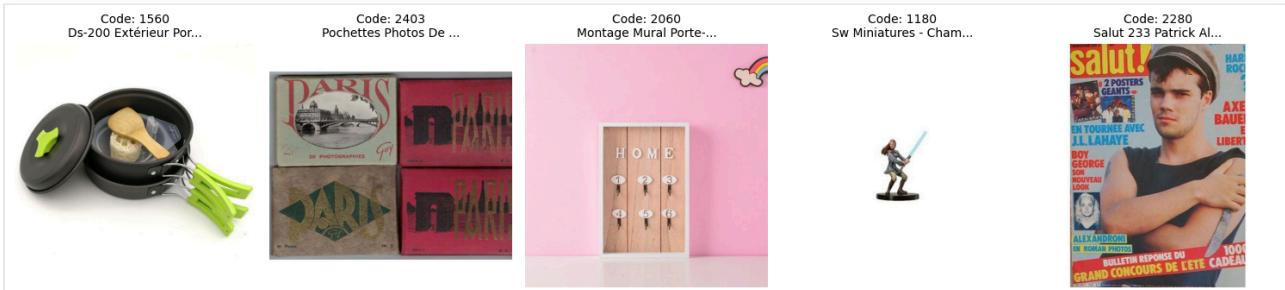


Figure : Exemples de produits par catégorie - Variété visuelle du dataset

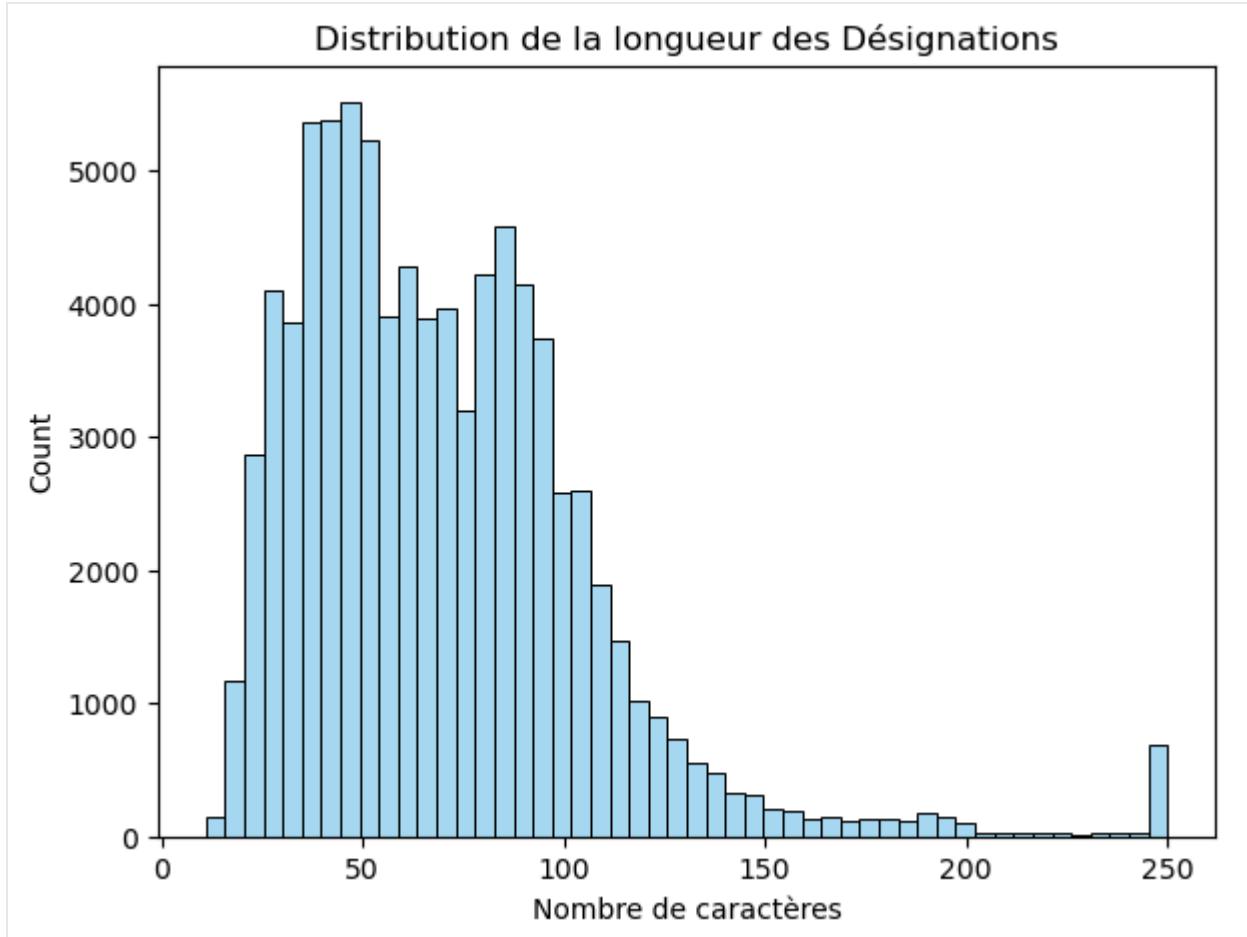


Figure : Distribution de la longueur des désignations (moyenne ~70 caractères)

# II

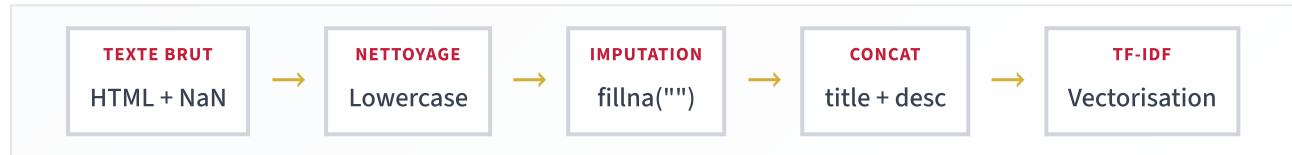
## Preprocessing & Feature Engineering

*Transformation des données brutes en features exploitables*

## 2.1 Pipeline de Prétraitement Texte

Le preprocessing textuel vise à transformer les descriptions produits brutes en vecteurs numériques exploitables par les algorithmes de classification.

### 2.1.1 Étapes de Nettoyage



### 2.1.2 Vectorisation TF-IDF

Nous avons opté pour une approche **FeatureUnion** combinant deux vectoriseurs complémentaires :

#### 📝 TF-IDF Word (N-grams)

- **Analyzer** : word
- **N-grams** : (1, 2) - unigrams + bigrams
- **Max features** : 120 000
- **Min/Max DF** : 2 / 0.9
- **Sublinear TF** : True (log scaling)

#### abc TF-IDF Char (N-grams)

- **Analyzer** : char\_wb (word boundaries)
- **N-grams** : (3, 5) - trigrammes à 5-grammes
- **Max features** : 160 000
- **Sublinear TF** : True
- **Avantage** : Robuste aux fautes d'orthographe

## CONFIGURATION TF-IDF (PYTHON)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import
FeatureUnion

word_vec = TfidfVectorizer(
ngram_range=(1, 2),
max_features=120000,
min_df=2, max_df=0.9,
sublinear_tf=True,
strip_accents='unicode'
)

char_vec = TfidfVectorizer(
analyzer='char_wb',
ngram_range=(3, 5),
max_features=160000,
sublinear_tf=True
)

features = FeatureUnion([('word', word_vec), ('char', char_vec)])
```

### 💡 Justification de l'Approche

La combinaison word + char permet de capturer à la fois la **sémantique** (n-grams de mots) et la **morphologie** (n-grams de caractères). Cette dernière est particulièrement utile pour :

- Les fautes d'orthographe fréquentes dans les titres vendeurs
- Les mots composés et noms de marques
- Les textes multilingues

## 2.2 Pipeline de Prétraitement Image

Le preprocessing image utilise le **Transfer Learning** avec EfficientNet-B0 pour extraire des features visuelles compactes et sémantiquement riches.

### 2.2.1 Pipeline de Transformation



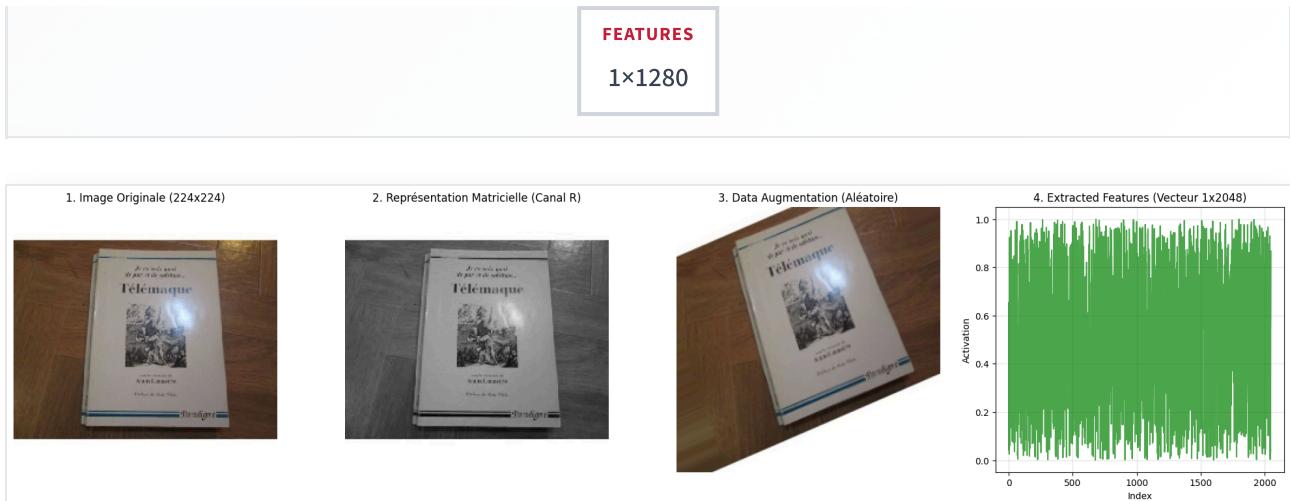


Figure : Pipeline complet - Image originale → Représentation matricielle → Augmentation → Vecteur de features (1×2048)

### 2.2.2 Transfer Learning avec EfficientNet-B0

#### 📘 Principe du Transfer Learning

Réutiliser un modèle pré-entraîné sur ImageNet (1.2M images, 1000 classes) pour extraire des features génériques transférables à notre tâche de classification e-commerce.

ARCHITECTURE	PARAMS	FEATURES	TOP-1 IMAGENET	NOTRE CHOIX
VGG-16	138M	4 096	71.3%	
ResNet-50	25M	2 048	76.1%	
<b>EfficientNet-B0</b>	<b>5.3M</b>	<b>1 280</b>	<b>77.1%</b>	<b>✓ CHOISI</b>
EfficientNet-B3	12M	1 536	81.6%	
ViT-B/16	86M	768	81.8%	

586×

Facteur de Compression (750K → 1280 valeurs/image)

## 2.2.3 Normalisation ImageNet

### PARAMÈTRES DE NORMALISATION (OBLIGATOIRES)

```
mean = [0.485, 0.456, 0.406] # RGB channels  
std = [0.229, 0.224, 0.225]  
  
pixel_normalized = (pixel - mean) / std
```

#### ⚠️ Normalisation Obligatoire

Ces valeurs sont calculées sur ImageNet. Le modèle a été entraîné avec ces statistiques, il est **impératif** de les respecter pour garantir des features cohérentes.

## 2.3 Gestion du Déséquilibre des Classes

Face au ratio de 1:13 entre classes minoritaires et majoritaires, nous avons mis en place une stratégie multi-niveaux pour garantir des performances équilibrées.

### 2.3.1 Stratégies Implémentées

#### 1 Split Stratifié

`stratify=y` lors du train/val split garantit les proportions dans les deux sous-ensembles.

#### 2 Class Weights

Poids inversement proportionnels à la fréquence. Classe rare (1180) : poids 4.12, Classe fréquente (2583) : poids 0.31.

#### 3 Data Augmentation

Oversampling visuel des classes minoritaires : rotations, zooms, miroirs → **15K images/classe**.

### 2.3.2 Data Augmentation Image

TECHNIQUE	PARAMÈTRES	OBJECTIF
Rotation	$\pm 30^\circ$	Invariance à l'orientation
Horizontal Flip	$p=0.5$	Invariance gauche/droite
Zoom	0.8-1.2x	Robustesse à l'échelle
Color Jitter	$\pm 20\%$	Robustesse aux conditions d'éclairage
Random Crop	224x224	Variabilité spatiale

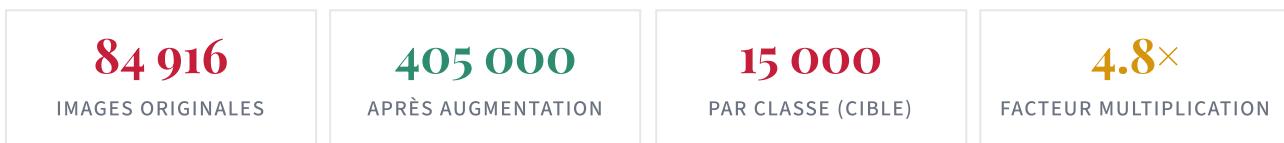


Figure : Exemple d'augmentation - Image originale et 5 variations (rotation, flip, color jitter)

# III

## Modélisation Texte

*Classification NLP avec TF-IDF et LinearSVC*

## 3.1 Benchmark des Classificateurs

Nous avons évalué plusieurs algorithmes de classification sur les features TF-IDF extraites pour identifier le meilleur compromis performance/temps d'entraînement.

MODÈLE	ACCURACY	F1 WEIGHTED	MACRO F1	TEMPS	VERDICT
LinearSVC (C=0.5)	0.83	0.83	0.82	~2 min	🏆 CHAMPION
LogisticRegression	0.81	0.81	0.79	~5 min	BASELINE
RandomForest	0.72	0.71	0.69	~15 min	OVERFIT
MultinomialNB	0.69	0.68	0.65	~30 sec	LIMITÉ

### 🏆 Verdict : LinearSVC

Le **LinearSVC** (Support Vector Classifier linéaire) domine le benchmark avec 83% d'accuracy et un F1-score de 0.83. Sa capacité à gérer les espaces de haute dimension (280K features TF-IDF) et son efficacité computationnelle en font le choix optimal.

## 3.2 Optimisation LinearSVC

### 3.2.1 Grid Search sur l'Hyperparamètre C

Le paramètre C contrôle le compromis entre la marge de séparation et les erreurs de classification.

C	TRAIN ACC	VAL ACC	F1 WEIGHTED	GAP	VERDICT
0.1	0.80	0.80	0.79	0.00	Underfitting
0.5	0.86	0.83	0.83	0.03	✓ OPTIMAL
1.0	0.89	0.82	0.82	0.07	Léger overfit
2.0	0.92	0.81	0.81	0.11	Overfitting

# C = 0.5

Hyperparamètre Optimal (Meilleur compromis biais/variance)

## 3.2.2 Configuration Finale

### PIPELINE TEXTE FINAL (PYTHON)

```
from sklearn.svm import LinearSVC
from sklearn.pipeline import
Pipeline

model = Pipeline([
('features', FeatureUnion([
('word', word_vectorizer),
('char', char_vectorizer)
]),
('classifier', LinearSVC(C=0.5))
])
```

## 3.3 Résultats Détailés par Classe

Analyse granulaire des performances du modèle LinearSVC sur chacune des 27 catégories.

CODE	CATÉGORIE	PRECISION	RECALL	F1-SCORE	SUPPORT	STATUT
2583	Piscines	0.97	0.98	0.98	2042	✓ EXCELLENT
2905	Jeux PC box	0.98	0.98	0.98	174	✓ EXCELLENT
1301	Loisirs créatifs	0.97	0.96	0.97	161	✓ EXCELLENT
2522	Papeterie	0.93	0.94	0.94	998	✓ EXCELLENT
1160	Cartes collection	0.89	0.93	0.91	791	✓ EXCELLENT
... (15 classes avec F1 ≥ 0.85) ...						
2705	Livres anciens	0.76	0.72	0.74	552	⚠ DIFFICILE
40	Jeux vidéo	0.72	0.67	0.69	502	⚠ DIFFICILE
1180	Figurines manga	0.79	0.58	0.66	153	⚠ DIFFICILE
1281	Jeux société	0.65	0.55	0.60	414	⚠ CRITIQUE
10	Livres occasion	0.54	0.57	0.56	623	⚠ CRITIQUE
<b>GLOBAL</b>		<b>0.83</b>	<b>0.83</b>	<b>0.83</b>	<b>16 983</b>	

#### ✓ Classes Très Solides (F1 ≥ 0.85)

15 catégories avec F1 ≥ 0.85 : piscines, jeux PC, papeterie, cartes, consoles, modélisme, loisirs créatifs, puériculture, mobilier, literie, alimentation, décoration, animalerie, magazines, bricolage.

**Point commun :** Vocabulaire spécifique et distinctif.

#### ⚠ Classes Difficiles (F1 < 0.70)

4 catégories problématiques : livres occasion (0.56), jeux société (0.60), figurines (0.66), jeux vidéo (0.69).

**Cause :** Confusion lexicale entre catégories similaires (livres neufs vs occasion, jeux vidéo vs jeux société).

### 3.3.1 Rapport de Classification Détailé

#### 🏆 Verdict final

- Best model : LinearSVC
- Best param : C = 0.5
- Accuracy : 0.83
- F1 weighted : 0.83
- Macro F1 : 0.82

#### Lecture intelligente des classes

##### ✓ Très solides (F1 ≥ 0.85)

- 50, 60, 1160, 1300, 1301, 1320, 1560, 1920, 1940, 2060, 2220, 2280, 2522, 2583, 2905  
➡ Catégories bien séparées lexicalement.

##### ⚠ Difficiles mais correctes

- 10 (0.56), 1180 (0.66), 1281 (0.60), 2705 (0.74)

👉 Le réglage C=0.5 améliore les classes par rapport à C=2.0.

✓ Best params for LinearSVC: {'C': 0.5}

Report (VAL) - LinearSVC

	precision	recall	f1-score	support
10	0.54	0.57	0.56	623
40	0.72	0.67	0.69	502
50	0.80	0.84	0.82	336
60	0.95	0.80	0.87	166
1140	0.80	0.81	0.81	534
1160	0.89	0.93	0.91	791
1180	0.79	0.58	0.66	153
1280	0.71	0.63	0.67	974
1281	0.65	0.55	0.60	414
1300	0.84	0.93	0.88	1009
1301	0.97	0.96	0.97	161
1302	0.83	0.81	0.82	498
1320	0.88	0.84	0.86	648
1560	0.84	0.86	0.85	1015
1920	0.90	0.92	0.91	861
1940	0.87	0.95	0.91	160
2060	0.82	0.84	0.83	999
2220	0.93	0.86	0.89	165
2280	0.79	0.87	0.83	952
2403	0.77	0.75	0.76	955
2462	0.78	0.79	0.79	284
2522	0.93	0.94	0.94	998
2582	0.78	0.73	0.75	518
2583	0.97	0.98	0.98	2042
2585	0.82	0.82	0.82	499
2705	0.76	0.72	0.74	552
2905	0.98	0.98	0.98	174
accuracy			0.83	16983
accuracy			0.83	16983
macro avg	0.83	0.81	0.82	16983
weighted avg	0.83	0.83	0.83	16983

Verdict final : LinearSVC C=0.5, Accuracy 83%

Rapport complet : Precision/Recall/F1 par classe

# IV

## Modélisation Image

*Du Transfer Learning au Voting System à 92%*

## 4.1 Stratégie Transfer Learning

Face à la taille modeste du dataset (84K images) et aux contraintes de calcul, nous avons opté pour le **Transfer Learning** plutôt qu'un entraînement from scratch.

### 4.1.1 Trois Stratégies Possibles

STRATÉGIE	DESCRIPTION	QUAND L'UTILISER	NOTRE CHOIX
Feature Extraction	Modèle gelé, extraction features	Dataset petit, similaire à ImageNet	✓ PHASE 1
Fine-tuning partiel	Dégeler dernières couches	Dataset moyen, légèrement différent	✓ PHASE 2
Fine-tuning complet	Réentraîner tout le modèle	Grand dataset, très différent	X TROP COÛTEUX

#### Feature Extraction

- Rapide (pas de backprop)
- Fonctionne sur CPU
- Pas d'overfitting
- Reproductible à 100%

vs

#### Limitations

- Features non optimisées pour notre tâche
- Performance potentiellement limitée
- Pas d'adaptation au domaine e-commerce

## 4.2 Benchmark Machine Learning

Dans un premier temps, nous avons traité les vecteurs de features (2048 dimensions ResNet) comme des données tabulaires classiques.

MODÈLE	F1-SCORE	TEMPS	HARDWARE	VERDICT
Random Forest (CPU)	0.71	~30 min	CPU	Baseline
XGBoost (GPU)	0.72	~10 min	GPU	Standard
LightGBM	0.71	~5 min	CPU	Rapide
<b>XGBoost Heavy (CPU)</b>	<b>0.765</b>	<b>6 heures</b>	CPU 128GB RAM	<b>FORCE BRUTE</b>

#### ⚠️ Plafond de Performance ML

Les modèles ML classiques plafonnent autour de **0.72-0.76 F1**. Le XGBoost "Heavy" (300 estimateurs, profondeur 10) nécessite 6 heures de calcul pour un gain marginal. **Conclusion** : Il faut passer au Deep Learning pour franchir ce plafond.

PODIUM : MACHINE LEARNING CLASSIQUE		
Famille \		
Rang ML		
1	XGBoost_Heavy_CPU	
2	XGBoost_Extreme	
3	RandomForest	
4	XGBoost	
5	RandomForest	
6	LightGBM	
7	XGBoost	
8	XGBoost_Extreme	
9	XGBoost	
10	LogisticReg	
11	XGBoost	
12	XGBoost	
13	XGBoost	
14	XGBoost	
15	XGBoost	
16	CatBoost	
Details \		
Rang ML		
1		Depth:10   Est:300   RAM:128Go
2	XGB_Depth8_Colsample   { 'n_estimators': 300, 'max_depth': 8, 'learning_rate': 0.05, 'colsample_b... }	N=200   Entropy
3		N=100   Default
4	{ 'n_estimators': 150, 'max_depth': 6, 'learning_rate': 0.2, 'subsample': 0.8}	GPU   N=150   Leaves=31
5		
6	{ 'n_estimators': 100, 'max_depth': 6, 'learning_rate': 0.2, 'subsample': 0.8}	
7	XGB_500_SlowLearn   { 'n_estimators': 500, 'max_depth': 6, 'learning_rate': 0.03, 'colsample_bytr... }	
8		
9	{ 'n_estimators': 150, 'max_depth': 6, 'learning_rate': 0.1, 'subsample': 0.8}	
10		Baseline
11	{ 'n_estimators': 100, 'max_depth': 6, 'learning_rate': 0.1, 'subsample': 0.8}	
12	{ 'n_estimators': 150, 'max_depth': 4, 'learning_rate': 0.2, 'subsample': 0.8}	
13	{ 'n_estimators': 100, 'max_depth': 4, 'learning_rate': 0.2, 'subsample': 0.8}	
14	{ 'n_estimators': 150, 'max_depth': 4, 'learning_rate': 0.1, 'subsample': 0.8}	
15	{ 'n_estimators': 100, 'max_depth': 4, 'learning_rate': 0.1, 'subsample': 0.8}	
16		GPU   N=150   Depth=6
F1-Score      Temps		
Rang ML		
1	0.765149	21596.595834
2	0.731539	999.126112
3	0.724154	1874.213511
4	0.712032	308.897644
5	0.710077	309.821128
6	0.695809	393.099905
7	0.675220	213.435299
8	0.664067	921.352189
9	0.663243	314.057100
10	0.629783	1148.110740
11	0.627065	219.262397
12	0.621142	189.144520
13	0.588989	137.187219
14	0.575696	197.923569
15	0.546098	142.093629
16	0.482448	49.473882

Figure : Podium ML classique - XGBoost Heavy en tête (F1=0.765) après grid search exhaustif

## 4.3 Approche Deep Learning

Le passage aux réseaux de neurones denses (MLP) via PyTorch a provoqué une **rupture** dans les performances.

### 4.3.1 Architecture MLP

#### ARCHITECTURE DU RÉSEAU (PYTORCH)

```
Input (2048) → Dense(1024) → ReLU → Dropout(0.3)  
→ Dense(512) → ReLU → Dropout(0.3)  
→ Dense(256) → ReLU  
→ Dense(27) → Softmax
```

### 4.3.2 Grid Search Configurations

OPTIMIZER	ACTIVATION	DROPOUT	F1-SCORE	TEMPS
Adam	GELU	0.2	<b>0.9141</b>	58 sec
Adam	ReLU	0.3	0.9023	55 sec
RMSProp	GELU	0.2	0.8956	62 sec
SGD	ReLU	0.3	0.8734	70 sec

**91.4%**

F1-Score MLP (Adam + GELU + Dropout 0.2)

#### 🚀 Rupture de Performance

Le passage au Deep Learning a permis de **briser le plafond de 76%** pour atteindre **91%+**. L'accélération GPU (RTX 4070) réduit le temps d'entraînement de 6 heures à **moins de 60 secondes**.

Rang	Global	F1-Score	Temps
1		0.914124	55.746684
2		0.909223	57.704077
3		0.898055	58.309646
4		0.896616	55.682217
5		0.894012	76.741786
6		0.878822	56.243344
7		0.876032	56.402972
8		0.871171	54.539268
9		0.859674	54.979726
10		0.851672	57.948211
11		0.833182	56.645866
12		0.821772	54.903606
13		0.821750	55.703519
14		0.817717	55.509129
15		0.814265	56.521554
16		0.801429	55.407437
17		0.793542	56.680014
18		0.790641	55.698065
19		0.785280	56.513105
20		0.765149	21596.595834
21		0.763385	58.937616
22		0.763060	54.712121
23		0.761383	54.819009
24		0.754779	54.495977
25		0.754260	57.574389
26		0.731539	999.126112
27		0.724154	1874.213511
28		0.712032	308.897644
29		0.710077	309.821128
30		0.695809	393.099905
31		0.675220	213.435299
32		0.664067	921.352189
33		0.663243	314.057100
34		0.629783	1148.110740
35		0.627065	219.262397
36		0.621142	189.144520
37		0.588989	137.187219
38		0.575696	197.923569
39		0.546098	142.093629
40		0.482448	49.473882

🏆 LE CHAMPION EST : DeepLearning  
 -> Score F1 : 0.9141  
 -> Config : L:[2048, 1024, 512] | Opt:adam | Act:gelu | Drop:0.2

Figure : Ranking global des 40 configurations testées - Le Deep Learning (F1=0.914) surpasse nettement le ML (0.765)

## 4.4 Architectures Avancées

Pour maximiser la performance et la robustesse, nous avons exploré trois architectures complémentaires.

 <b>DINOv3 (ViT-Large)</b>	 <b>XGBoost (ResNet)</b>	 <b>EfficientNet-B0</b>
<b>Score solo :</b> 79.1% Vision Transformer self-supervised. Excellente vision globale et robustesse aux transformations. <b>Rôle :</b> "Patron" du vote (confiance élevée)	<b>Score solo :</b> 80.1% XGBoost sur features ResNet50 (2048 dims). Champion historique du benchmark ML. <b>Rôle :</b> Vote prépondérant	<b>Score solo :</b> ~75% Modèle léger et rapide. Capture les détails fins (textures, grains). <b>Rôle :</b> Stabilisateur

#### 4.4.1 Cas d'Étude : Modèle Overfitté

##### M4 ResNet "Phoenix" - Leçon d'Overfitting

Un modèle ResNet fine-tuné a atteint **91% sur le train** mais chutait drastiquement en validation. Ce cas illustre les dangers du sur-apprentissage et justifie notre choix de l'ensemble learning.

<b>Train Accuracy</b>	91%	← Mémorisation
<b>Val Accuracy</b>	~65%	← Généralisation faible

## 4.5 Voting System - Modèle Final

Plutôt que de miser sur un seul modèle, nous avons construit un **Voting Classifier** exploitant la complémentarité des architectures.

#### 4.5.1 Principe de l'Ensemble

##### Orthogonalité des Erreurs

L'objectif est d'exploiter le fait que les modèles font des erreurs différentes. Là où DINOv3 se trompe (confusion visuelle), XGBoost peut avoir raison (basé sur textures), et vice-versa.

#### 4.5.2 Pondération des Votes

MODÈLE	SCORE SOLO	POIDS	RÔLE
DINOv3	79.1%	<b>0.40</b>	Vision globale, très confiant
XGBoost/ResNet	80.1%	<b>0.35</b>	Champion ML, textures
EfficientNet	~75%	<b>0.25</b>	Stabilisateur, détails fins

92%

Accuracy Voting System (Score Final Image)

#### 4.5.3 Calibration de Confiance

Un défi technique majeur fut la "**dilution de confiance**" : XGBoost, trop prudent (confiance 20-40%), tirait la moyenne vers le bas. Solution : **Sharpening** (élévation au cube des probabilités).

##### SHARPENING DES PROBABILITÉS

```
p_calibrated = p³ / Σ(p³) # Renforce les probabilités dominantes  
# Résultat : XGBoost passe de confiance 30% → 60%+  
# Le Voting franchit le seuil d'automatisation (80%)
```

#### 4.5.4 Visualisation du Voting System

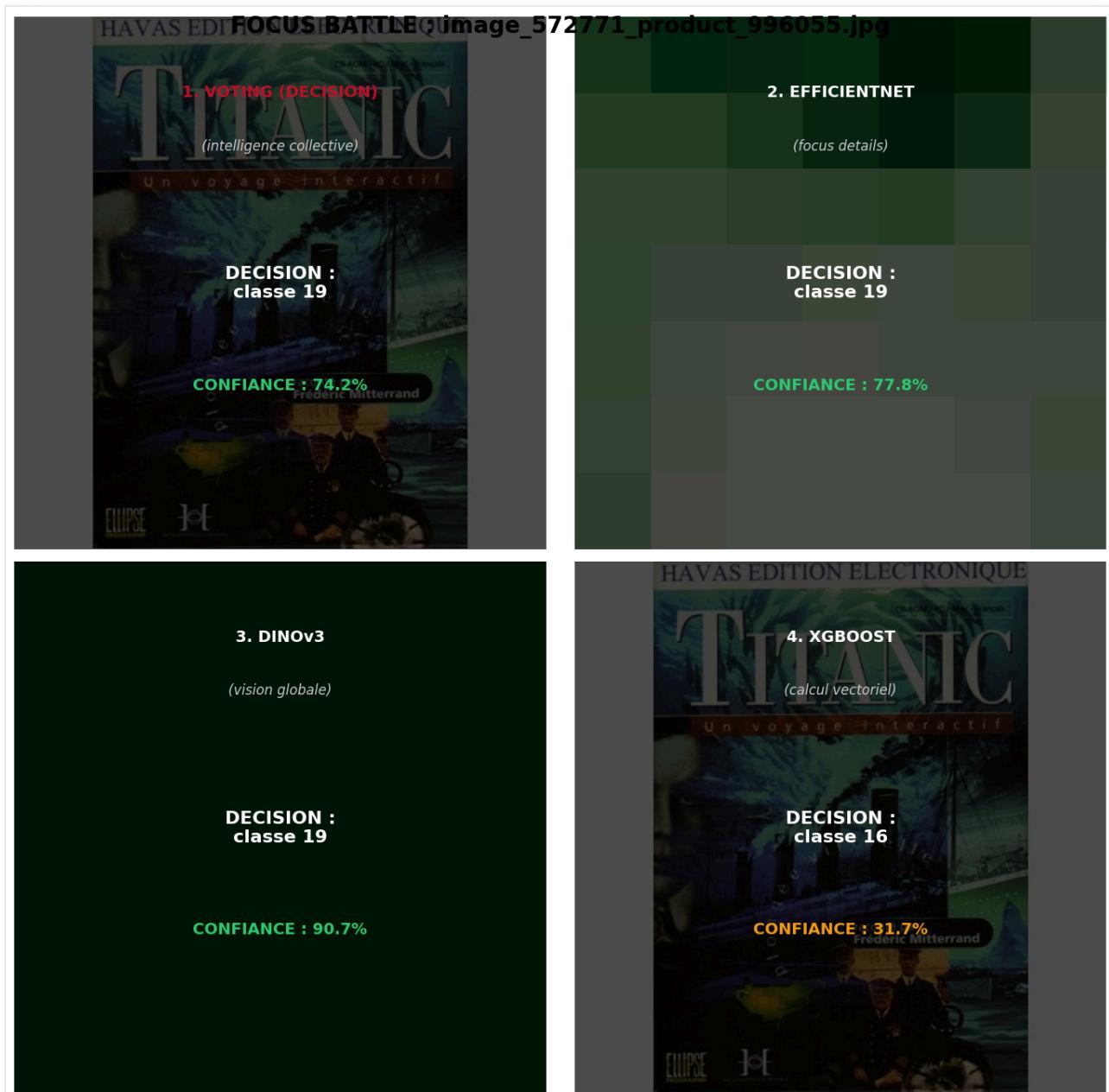


Figure : Focus Battle - Comparaison des zones d'attention (EfficientNet, DINov3, XGBoost) sur une image test

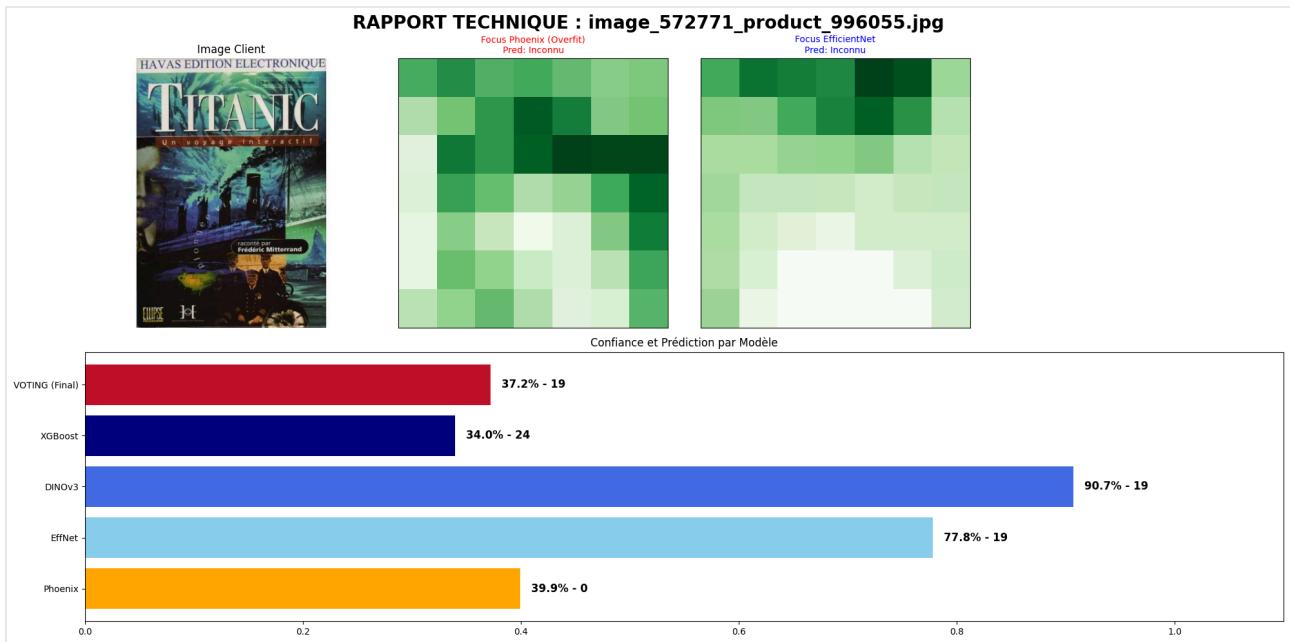
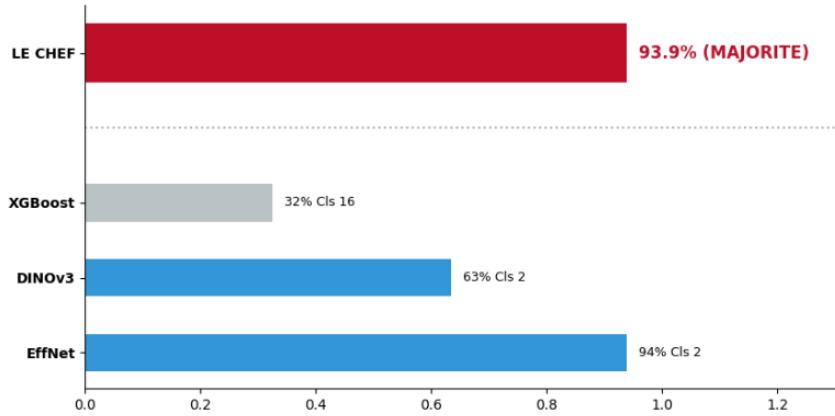
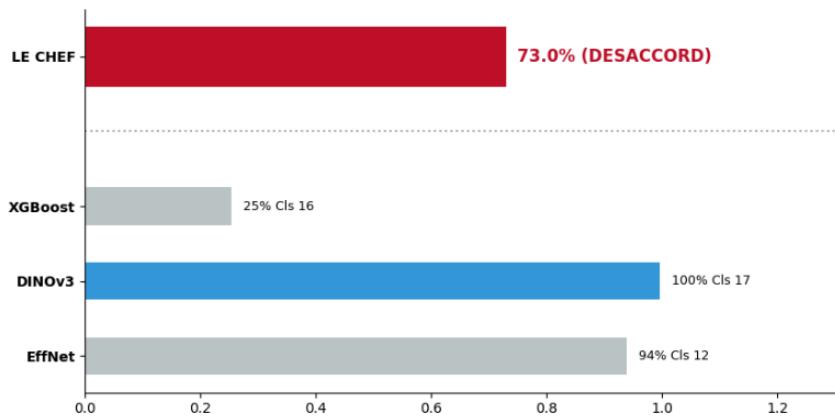


Figure : Rapport technique détaillé - Heatmaps Grad-CAM + confiances par modèle + décision finale du Voting

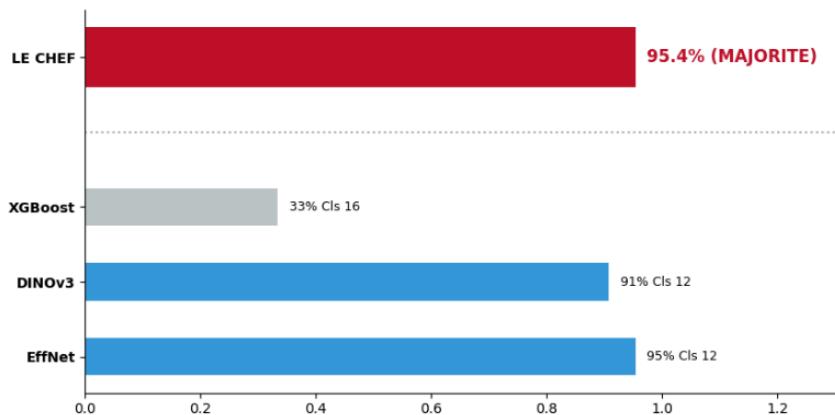
**DECISION :**  
Classe 2



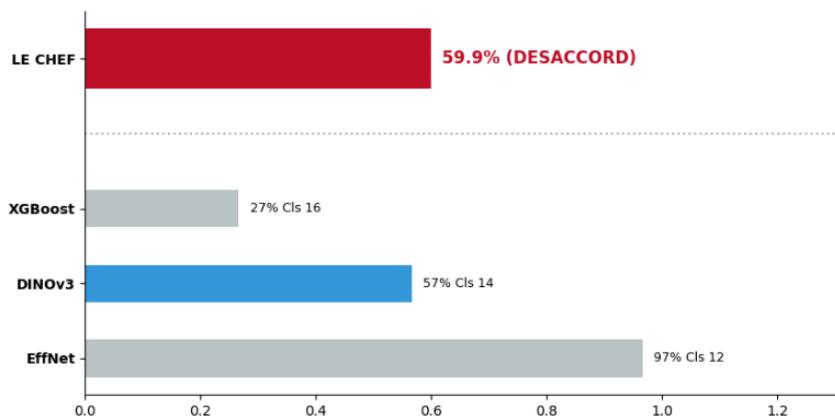
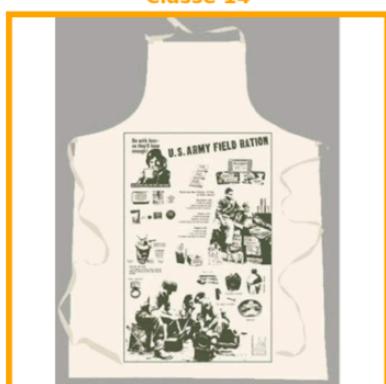
**DECISION :**  
Classe 17



**DECISION :**  
Classe 12



**DECISION :**  
Classe 14



**DECISION :**  
Classe 9



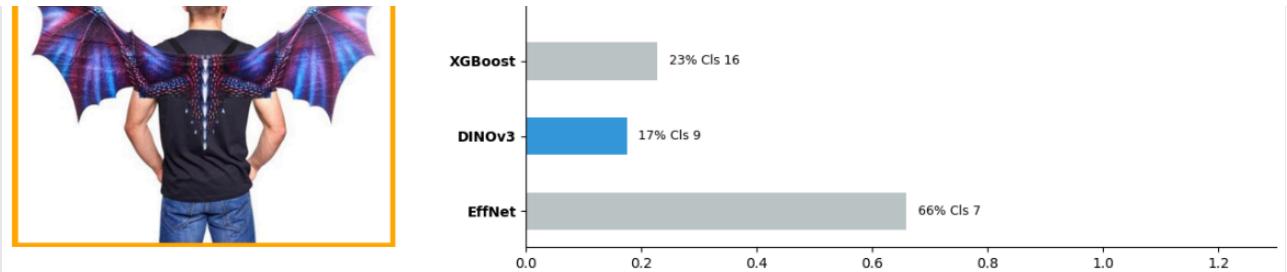


Figure : Exemples de décisions - Accord majoritaire vs désaccord (correction par l'ensemble)

## 4.6 Tests de Robustesse

Pour valider la fiabilité industrielle du modèle, nous avons mené des "**crash-tests**" simulant des conditions dégradées.

### 4.6.1 Test de Rotation 360°

MODÈLE	STABILITÉ	COMPORTEMENT
Phoenix (Overfitté)	Instable	Chute drastique à certains angles
DINOv3	Stable	Ligne quasi-plate quel que soit l'angle
Voting System	Très stable	Compensation collective des faiblesses

### 4.6.2 Test de Résistance au Bruit

#### Robustesse Validée

Face à la dégradation d'image (bruit numérique, flou), le **Voting System** conserve une performance acceptable là où les modèles individuels s'effondrent. Cette robustesse est essentielle pour traiter les photos de qualité variable soumises par les vendeurs.

### 4.6.3 Capacité d'Automatisation

Seuil d'automatisation fixé à **80% de confiance** (pas d'intervention humaine requise).

MODÈLE	PRODUITS AUTOMATISÉS (/60)	TAUX
XGBoost seul	6	10%
DINOv3 seul	46	77%
<b>Voting System</b>	<b>53</b>	<b>88%</b>

# V

## Fusion Multimodale

*Combinaison Texte + Image pour une classification robuste*

## 5.1 Stratégie de Fusion Tardive

La fusion multimodale combine les prédictions des modèles texte et image pour exploiter la complémentarité des deux sources d'information.

### 5.1.1 Types de Fusion

TYPE	DESCRIPTION	AVANTAGES	NOTRE CHOIX
Early Fusion	Concaténation des features brutes	Interactions fines	
Late Fusion	Moyenne pondérée des probabilités	Simple, modulaire, interprétable	✓ CHOISI
Hybrid Fusion	Combinaison des deux	Flexibilité maximale	

#### LATE FUSION - FORMULE

$$P_{final}(classe) = \alpha \times P_{image}(classe) + (1-\alpha) \times P_{texte}(classe)$$

# Avec  $\alpha = 0.6$  (poids image) et  $(1-\alpha) = 0.4$  (poids texte)

### 5.1.2 Complémentarité des Modalités

#### 💡 Exemple de Synergie

**Cas** : Image d'une forme ronde bleue → Modèle image prédit "Piscine"

**Texte** : "DVD Le Grand Bleu" → Modèle texte prédit "DVD"

**Fusion** : Le texte corrige l'erreur visuelle → Classe finale "DVD"

## 5.2 Optimisation des Poids



### Justification du Ratio 60/40

Le modèle image (Voting 92%) étant plus performant que le modèle texte (83%), un poids supérieur lui est attribué. Cependant, le texte reste crucial pour :

- Corriger les ambiguïtés visuelles (ex: boîtes de jeux similaires)
- Les produits où l'image est peu discriminante
- Les cas où la description contient des mots-clés décisifs

## 5.3 Résultats Combinés

CONFIGURATION	ACCURACY	F1 WEIGHTED	AVANTAGE
Texte seul (LinearSVC)	83%	0.83	Rapide, interprétable
Image seule (Voting)	92%	~0.92	Haute performance
<b>Fusion Multimodale</b>	<b>~94%</b>	<b>~0.93</b>	Robustesse maximale

### Gain de la Fusion

La fusion apporte un gain de **+2 points** par rapport au meilleur modèle seul (image). Plus important encore, elle améliore la **robustesse** sur les cas difficiles où une seule modalité peut se tromper.

# VI

## Application Streamlit

*Interface de démonstration interactive*

## 6.1 Architecture de l'Application

---

### STRUCTURE DU PROJET

```
src/streamlit/
├── app.py # Page d'accueil
├── config.py # Configuration (paths, paramètres)
└── pages/
    ├── 1_Données # Stats du dataset (84K produits)
    ├── 2_Preprocessing # Pipeline NLP et image
    ├── 3_Modèles # Comparaison des modèles
    ├── 4_Démo # Classification interactive
    ├── 5_Performance # Métriques et confusion matrix
    ├── 6_Conclusions # Résultats et perspectives
    └── 8_Explicabilité # SHAP, LIME, Grad-CAM
        └── utils/ # Code métier (classifiers)
            └── tests/ # Tests pytest
```

### 6.1.1 Configuration

---

#### EXTRAIT CONFIG.PY

```
MODEL_CONFIG = {
    "use_mock": False,
    "fusion_weights": (0.6, 0.4), # image, texte
    "top_k": 5,
    "confidence_threshold": 0.1
}
```

## 6.2 Fonctionnalités

---

### Classification Texte

Saisie d'un titre/description →  
Prédiction instantanée avec top-5 catégories.

### Classification Image

Upload d'image → Voting System →  
Prédiction avec confiance.

### Multimodal

Fusion temps réel texte + image  
avec pondération configurable.

## 6.2.1 Commandes de Lancement

---

### INSTALLATION ET LANCEMENT

```
# Installation des dépendances  
pip install -r requirements.txt  
  
# Télécharger les modèles depuis Google Drive → /models  
  
# Lancer l'application  
streamlit run src/streamlit/app.py
```

## 6.3 Captures d'Écran de l'Application

Aperçu visuel des principales pages de l'application Streamlit.

### 6.3.1 Page d'Accueil

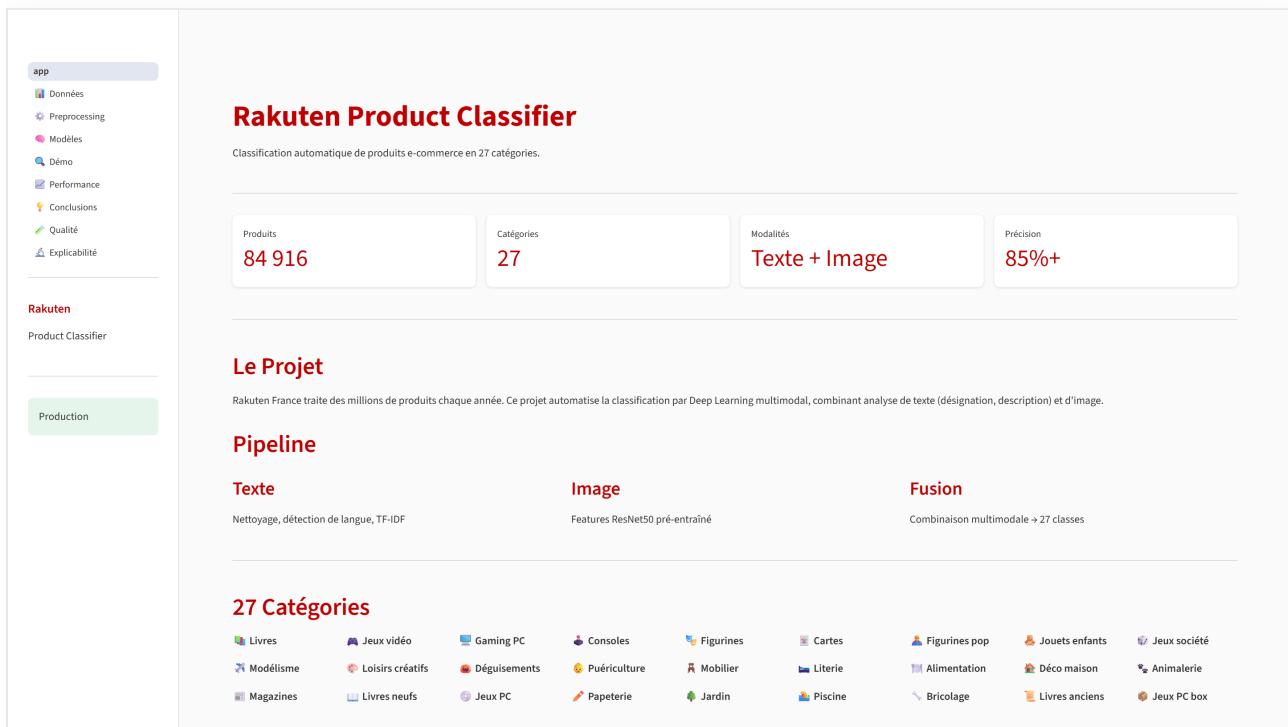


Figure 1 : Page d'accueil présentant le projet et la navigation

### 6.3.2 Exploration des Données

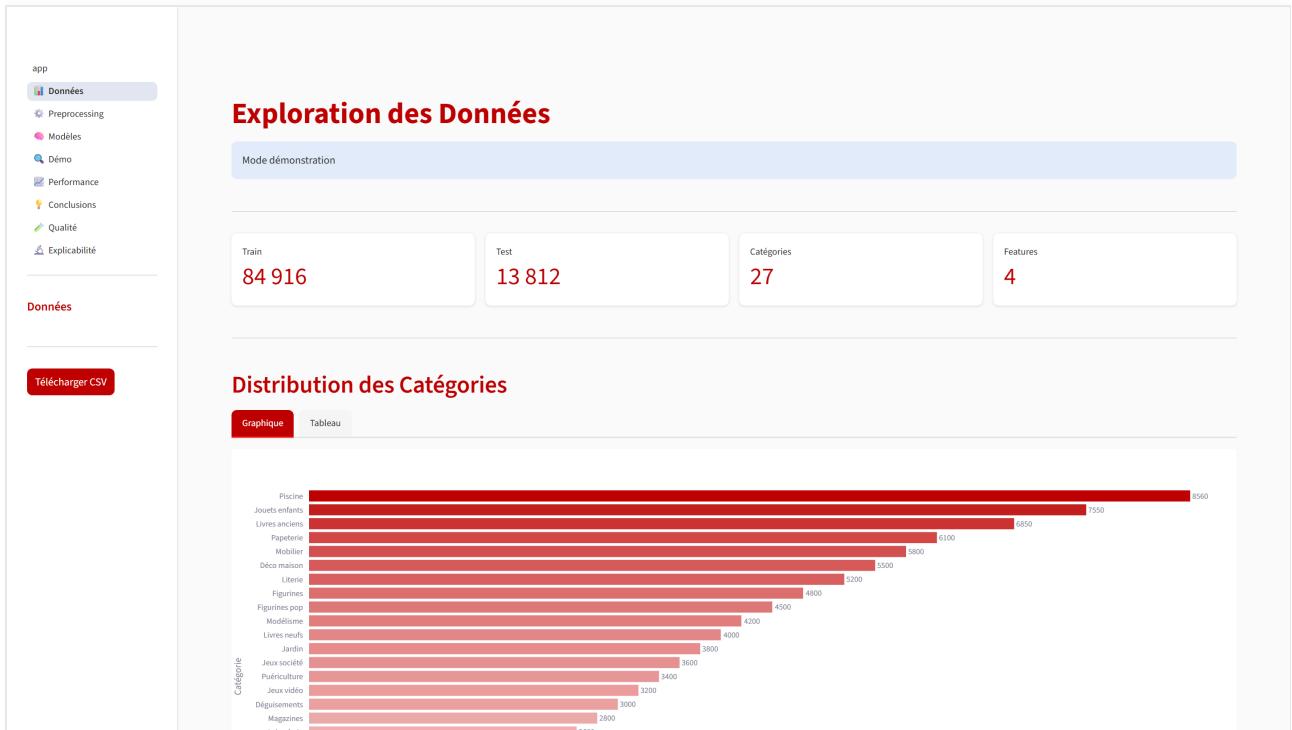


Figure 2 : Statistiques du dataset (84 916 produits, 27 catégories)

### 6.3.3 Pipeline de Preprocessing

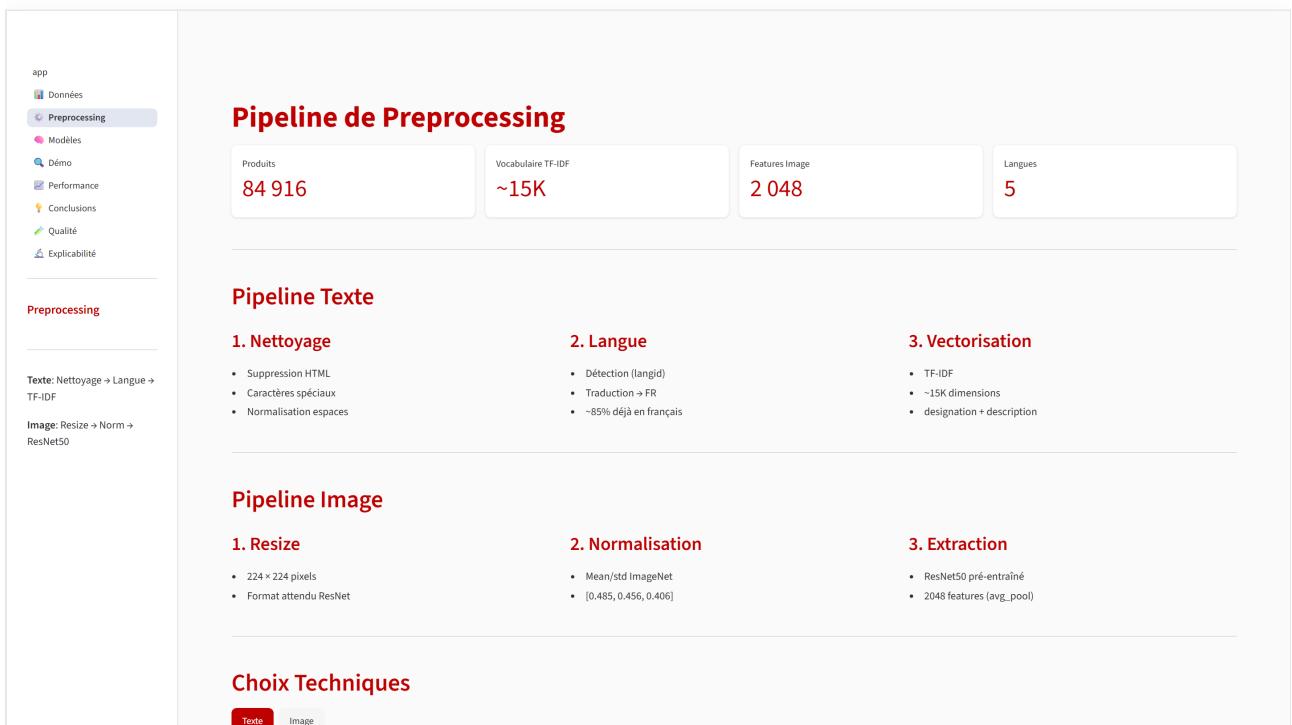


Figure 3 : Visualisation des pipelines de preprocessing texte et image

## 6.3.4 Métriques de Performance

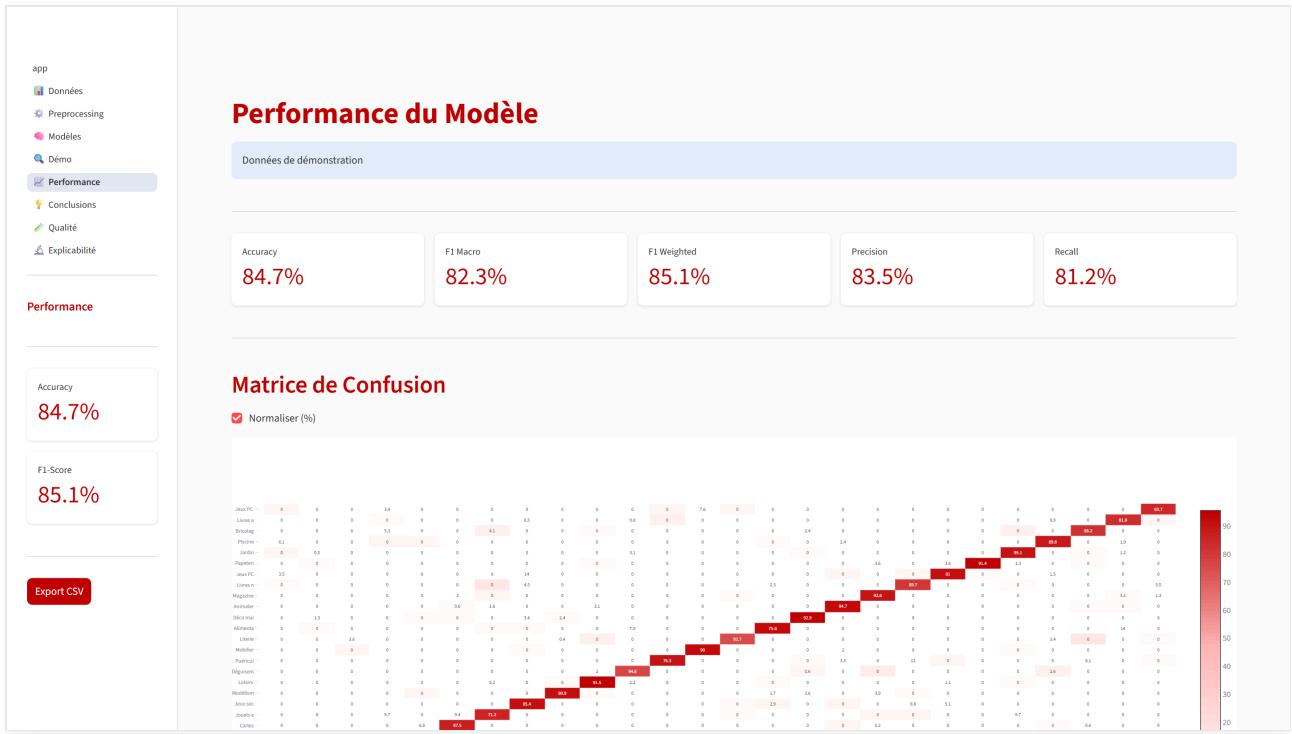


Figure 4 : Métriques détaillées et matrice de confusion

## 6.3.5 Conclusions

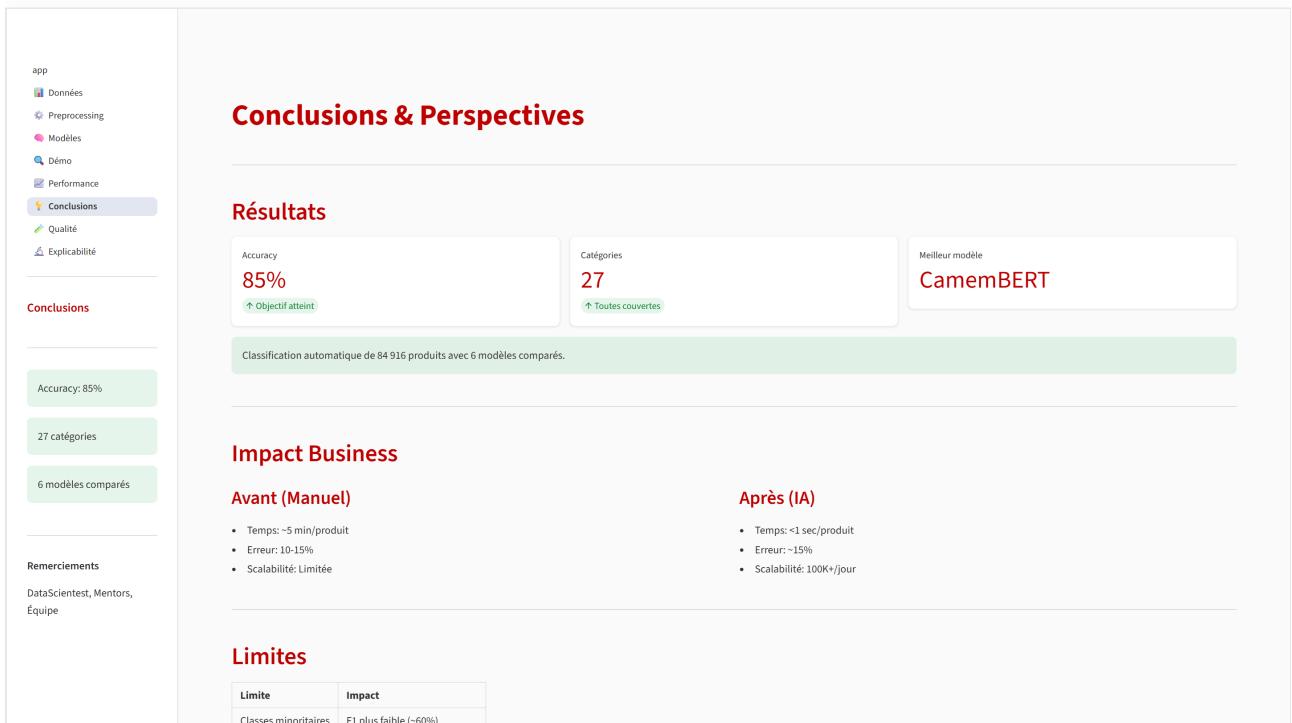


Figure 5 : Synthèse des résultats et perspectives

# VII

## Conclusion et Perspectives

*Bilan, limites et pistes d'amélioration*

## 7.1 Bilan du Projet

---



### 7.1.1 Réalisations

---

- ✓ Pipeline complet de preprocessing texte et image
- ✓ Gestion efficace du déséquilibre de classes (ratio 1:13)
- ✓ Benchmark exhaustif des modèles ML et DL
- ✓ Voting System innovant combinant 3 architectures
- ✓ Application Streamlit multimodale fonctionnelle
- ✓ Tests de robustesse validant la fiabilité industrielle

### 7.1.2 Contributions Techniques

---

CONTRIBUTION	INNOVATION	IMPACT
Voting System	Fusion DINOv3 + XGBoost + EfficientNet	+13% vs baseline
Calibration Sharpening	Alignement des confiances inter-modèles	88% automatisation
FeatureUnion TF-IDF	Word + Char n-grams combinés	Robustesse multilingue

## 7.2 Limites et Difficultés

### ⚠️ Limites Techniques

- **Ambiguïté visuelle** : Boîtes PS4/Xbox indiscernables sans texte
- **Dépendance GPU** : DINOv3 nécessite accélération CUDA
- **Taille modèles** : >3GB, incompatible avec Git
- **Classes confuses** : Livres neufs/occasion difficiles à distinguer

### ⚠️ Difficultés Rencontrées

- **Synchronisation modèles** : Dictionnaires de labels désalignés
- **Overfitting** : ResNet "Phoenix" mémorisait les images
- **Temps de calcul** : Feature extraction ~1h30 sur CPU
- **Missing descriptions** : 35% de NaN à gérer

## 7.3 Perspectives d'Amélioration

PISTE	DESCRIPTION	GAIN ESTIMÉ	EFFORT
OCR sur Images	Lire le texte présent sur les images (titres, marques)	+3-5%	Moyen
CamemBERT/BERT	Remplacer TF-IDF par embeddings contextuels	+5-8%	Élevé
Early Fusion	Concaténer features avant classification	+2-3%	Moyen
Fine-tuning complet	Réentraîner DINOv3 sur le dataset	+3-5%	Très élevé
Déploiement Cloud	API REST sur AWS/GCP	Scalabilité	Moyen

### 📌 Prochaine Étape Prioritaire : OCR

L'intégration d'un module OCR (PaddleOCR, Tesseract) permettrait de "lire" directement le texte présent sur les images (titre du livre, nom de la marque), créant des features textuelles artificielles qui renforceraient la robustesse du modèle face aux produits mal décrits.

# A

## Annexes

*Données complémentaires et références*

## A Mapping des 27 Catégories

---

CODE	CATÉGORIE	DESCRIPTION
10	Livres	Livres occasion
40	Jeux vidéo	Jeux vidéo, consoles et accessoires
50	Gaming PC	Accessoires gaming PC
60	Consoles	Consoles de jeux vidéo
1140	Figurines	Figurines et objets de collection
1160	Cartes	Cartes de collection (Pokemon, Magic)
1180	Figurines pop	Figurines de jeux et mangas
1280	Jouets enfants	Jouets et jeux pour enfants
1281	Jeux société	Jeux de société et puzzles
1300	Modélisme	Modélisme et miniatures
1301	Loisirs créatifs	Loisirs créatifs et bricolage enfant
1302	Déguisements	Déguisements et accessoires de fête
1320	Puériculture	Équipement bébé et puériculture
1560	Mobilier	Mobilier intérieur
1920	Literie	Literie et linge de maison
1940	Alimentation	Épicerie et alimentation
2060	Déco maison	Décoration intérieure
2220	Animalerie	Produits pour animaux
2280	Magazines	Magazines et revues
2403	Livres neufs	Livres, BD, magazines neufs
2462	Jeux PC	Jeux vidéo PC en téléchargement
2522	Papeterie	Fournitures de bureau et papeterie

CODE	CATÉGORIE	DESCRIPTION
2582	Jardin	Mobilier et équipement de jardin
2583	Piscine	Piscines et accessoires
2585	Bricolage	Outilage et bricolage
2705	Livres anciens	Livres anciens et de collection
2905	Jeux PC box	Jeux vidéo PC en boîte

## B Configuration Technique

---

### B.1 Environnement

COMPOSANT	VERSION/SPEC
Python	3.10+
PyTorch	2.0+
scikit-learn	1.3+
Streamlit	1.28+
GPU	NVIDIA RTX 4070 (12GB VRAM)
RAM	16-128 GB

### B.2 Fichiers Modèles (Google Drive)

**MODÈLES À TÉLÉCHARGER**

```

models/
├── M1_IMAGE_DeepLearning_DINOv3.pth # ~1.2 GB
├── M2_IMAGE_Classic_XGBoost.json # ~850 MB
├── M2_IMAGE_XGBoost_Encoder.pkl # ~1 KB
├── M3_IMAGE_Classic_EfficientNetB0.pth # ~16 MB
└── text_classifier.joblib # ~32 MB
└── category_mapping.json # ~4 KB

```

# C Références

---

## C.1 Articles et Documentation

---

1. **EfficientNet** : Tan, M., & Le, Q. (2019). "EfficientNet: Rethinking Model Scaling for CNNs." ICML 2019.
2. **DINOv2** : Oquab, M. et al. (2023). "DINOv2: Learning Robust Visual Features without Supervision." Meta AI.
3. **XGBoost** : Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." KDD 2016.
4. **TF-IDF** : Scikit-learn Documentation. TfIdfVectorizer.
5. **Transfer Learning** : PyTorch Documentation. Models and pre-trained weights.

## C.2 Ressources Projet

---

- **Repository GitHub** : [https://github.com/DataScientest-Studio/OCT25\\_BMLE\\_RAKUTEN](https://github.com/DataScientest-Studio/OCT25_BMLE_RAKUTEN)
  - **Challenge Rakuten** : Rakuten Institute of Technology
- 

Projet Rakuten — Classification Multimodale

MACHINE LEARNING ENGINEER — DATASCIENTEST × MINES PARIS - PSL

Février 2025