|  |
| --- |
| DataScientest |
| Projet Rakuten |
| [Sous-titre du document] |

|  |
| --- |
| Lionel SERGENT  12/06/2025 |

Table des matières

[Introduction 3](#_Toc200718353)

[1. Exploration des données 3](#_Toc200718354)

[Jeu d’entrainement 3](#_Toc200718355)

[Statistiques de base : 3](#_Toc200718356)

[Analyse de la distribution des catégories : 4](#_Toc200718357)

[Analyse des longueurs de texte : 4](#_Toc200718358)

[Analyse des mots fréquents : 5](#_Toc200718359)

[Analyse des propriétés des images : 6](#_Toc200718360)

[Analyse des propriétés des images en fonction des catégories : 7](#_Toc200718361)

[Analyse de la distribution des classes: 7](#_Toc200718362)

[Statistiques des tailles de fichiers (KB): 9](#_Toc200718363)

[Jeu de test 10](#_Toc200718364)

[Statistiques des tailles de fichiers (KB) : 10](#_Toc200718365)

[2. Comparaison des modèles 11](#_Toc200718366)

[Gestion du texte : 12](#_Toc200718367)

[Gestion des images : 13](#_Toc200718368)

[Architecture du projet 13](#_Toc200718369)

[Étapes détaillées du pipeline 13](#_Toc200718370)

[Optimisations techniques 19](#_Toc200718371)

[3. Fusion modèles et explicabilité 20](#_Toc200718372)

[3.1 Stratégies de fusion multimodale 20](#_Toc200718373)

[Approche Late Fusion 20](#_Toc200718374)

[Méthodes de fusion implémentées 20](#_Toc200718375)

[3.2 Résultats de la fusion multimodale 21](#_Toc200718376)

[Évaluation comparative 21](#_Toc200718377)

[Métriques de performance 22](#_Toc200718378)

[Analyse des gains multimodaux 22](#_Toc200718379)

[Performances détaillées par catégorie 23](#_Toc200718380)

[3.3 Explicabilité avec SHAP 23](#_Toc200718381)

[Approche d'explicabilité 23](#_Toc200718382)

[Implémentation SHAP 24](#_Toc200718383)

[Insights d'explicabilité 24](#_Toc200718384)

[Explicabilité multimodale 25](#_Toc200718385)

[Fallback d'explicabilité 25](#_Toc200718386)

[3.4 Architecture technique de la fusion 25](#_Toc200718387)

[Pipeline multimodal 25](#_Toc200718388)

[Gestion des dimensions 26](#_Toc200718389)

[Robustesse et validation 26](#_Toc200718390)

[3.5 Bilan de la fusion multimodale 26](#_Toc200718391)

[Avantages observés 26](#_Toc200718392)

[Défis techniques relevés 26](#_Toc200718393)

[Perspectives d'amélioration 26](#_Toc200718394)

[3.6 Bilan et enseignements du projet 26](#_Toc200718395)

[Réussites du projet multimodal 26](#_Toc200718396)

[Enseignements techniques 27](#_Toc200718397)

[Impact du projet 27](#_Toc200718398)

[Conclusion 28](#_Toc200718399)

[4. Frontend Streamlit 29](#_Toc200718400)

[5. DevOPS 30](#_Toc200718401)

[6. Annexes 31](#_Toc200718402)

# Introduction

Ce projet s’inscrit dans le cadre du projet Rakuten : <https://challengedata.ens.fr/challenges/35>

Le but est de prévoir le code produit de la catégorie d’un produit par recherche multimodale.  
Les inputs sont composés de :

1. X\_train.csv: training input file (Désignation, description, productid, imageid), 8.6Mo.
2. Y\_train.csv: training output file (code produit), 52.8Mo.
3. X\_test.csv: test input file, 138Ko.
4. image\_train : 84916 images (avec nom = image\_id, product\_id), 2.1 Go.
5. image\_test : 13812 images (avec nom = image\_id, product\_id), 349 Mo.

Après une première étape d’exploration des données, nous avons créé une équipe image et une équipe texte, partant de fait sur une stratégie de late fusion.

Après la sélection des modèles, nous nous sommes intéressés à la fusion de ceux-ci, et à l’explicabilité de résultats.

Observation : Nous avons utilisé une [table de correspondance](#_6._Annexes) pour traduire les codes en catégories visuellement compréhensives, et cela a été l’occasion d’observer des incohérences dans les labels d’entrainement. Le nom des catégories est là à titre purement symbolique puisqu’aucune confirmation n’a pu être trouvée.

# Exploration des données

## Jeu d’entrainement

### Statistiques de base :

Nombre total de produits 84916.000000

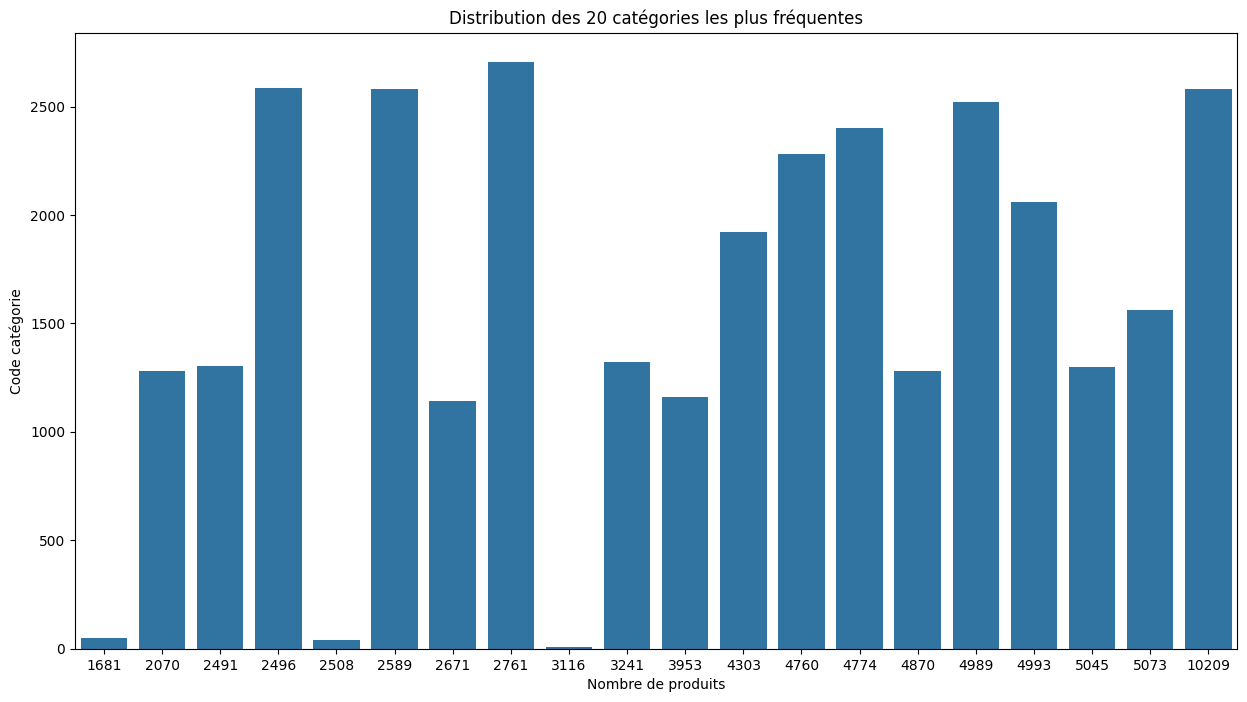
Nombre de catégories uniques 27.000000

% de descriptions manquantes 35.093504

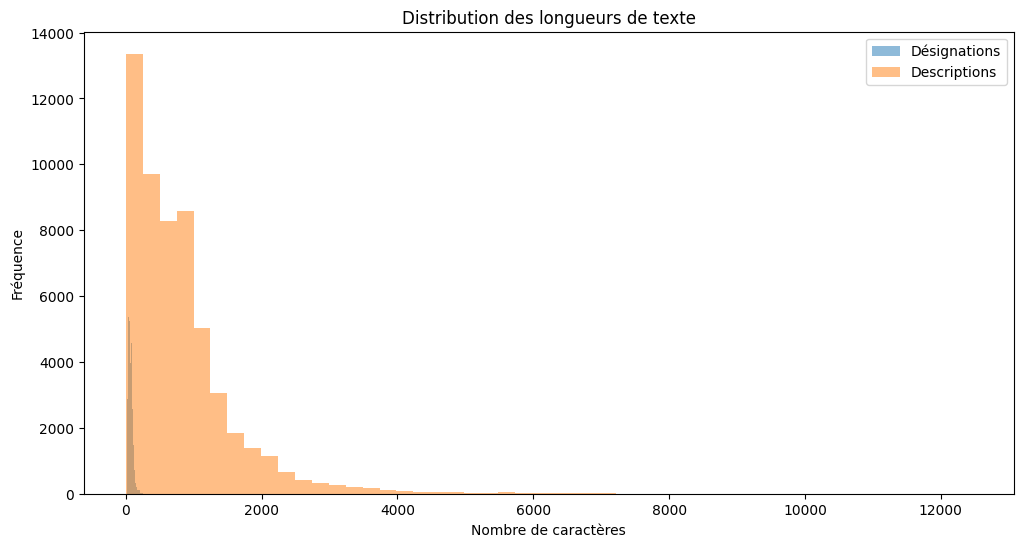
Longueur moyenne des désignations 70.163303

Nombre d'images uniques 84916.000000

### Analyse de la distribution des catégories :



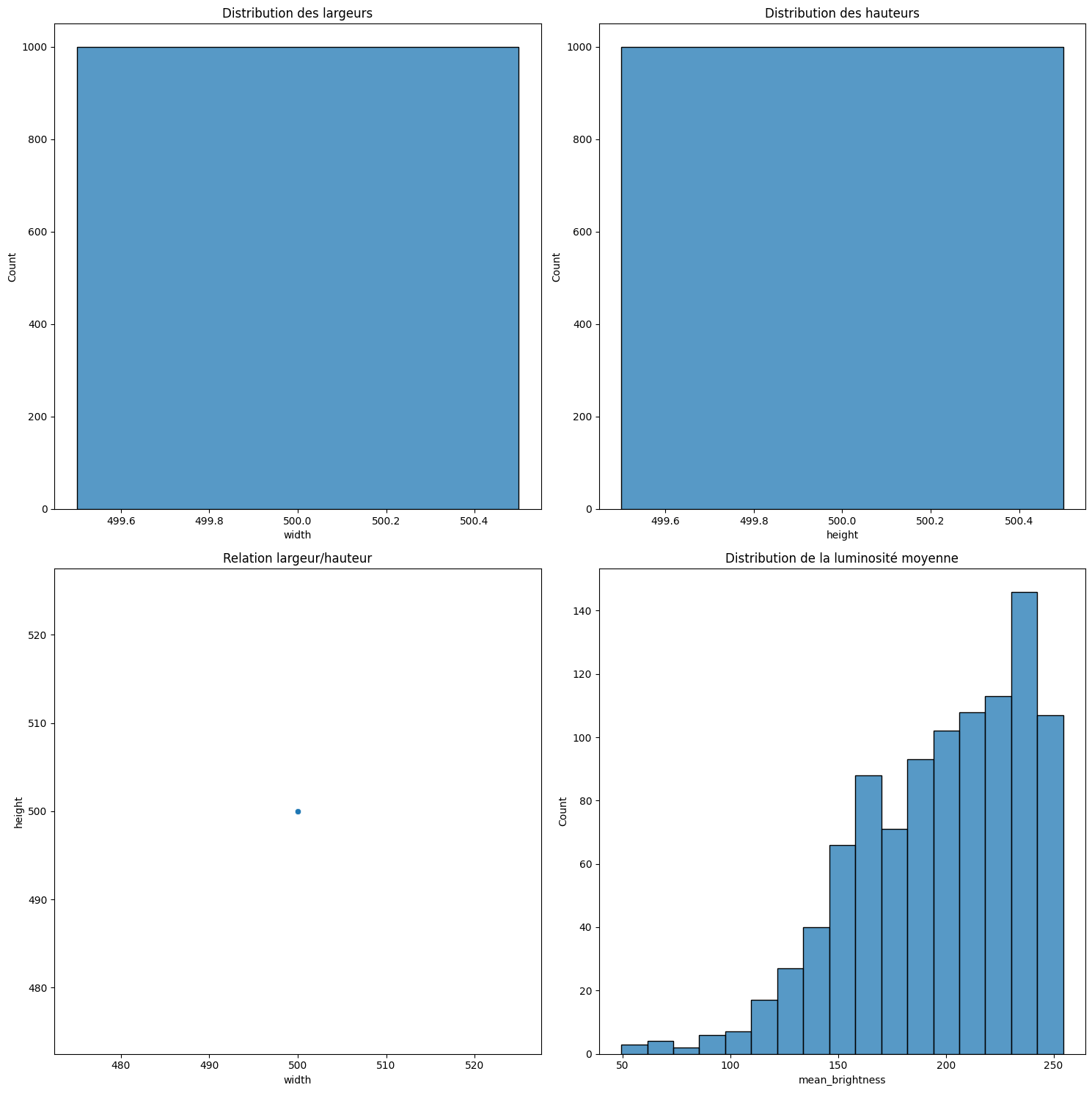
### Analyse des longueurs de texte :



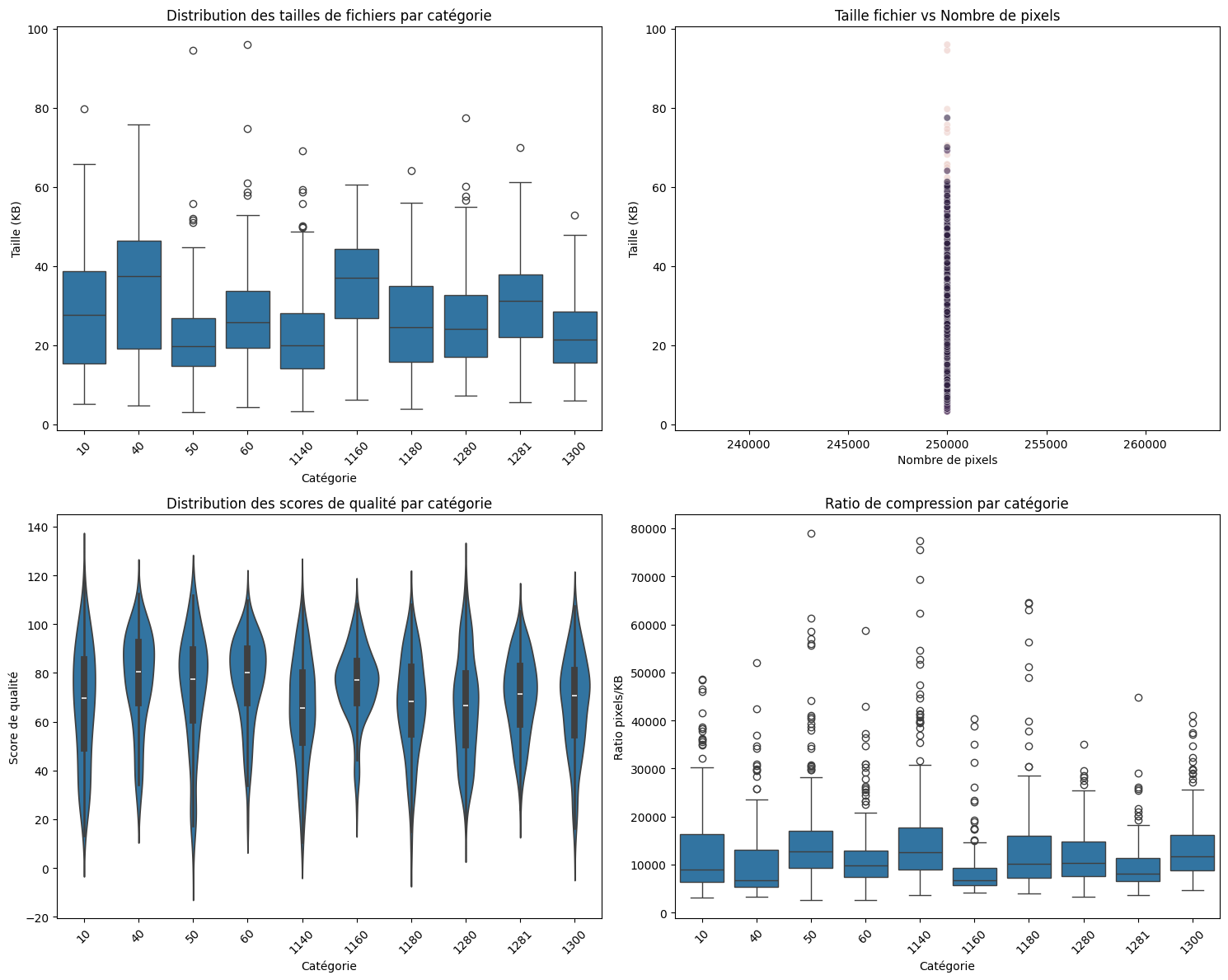
### Analyse des mots fréquents :



### Analyse des propriétés des images :



### Analyse des propriétés des images en fonction des catégories :



### Analyse de la distribution des classes:

Nombre d'échantillons total: 84916

Par classe:

Classe 10: 3116 échantillons ( 3.67%)

Classe 40: 2508 échantillons ( 2.95%)

Classe 50: 1681 échantillons ( 1.98%)

Classe 60: 832 échantillons ( 0.98%)

Classe 1140: 2671 échantillons ( 3.15%)

Classe 1160: 3953 échantillons ( 4.66%)

Classe 1180: 764 échantillons ( 0.90%)

Classe 1280: 4870 échantillons ( 5.74%)

Classe 1281: 2070 échantillons ( 2.44%)

Classe 1300: 5045 échantillons ( 5.94%)

Classe 1301: 807 échantillons ( 0.95%)

Classe 1302: 2491 échantillons ( 2.93%)

Classe 1320: 3241 échantillons ( 3.82%)

Classe 1560: 5073 échantillons ( 5.97%)

Classe 1920: 4303 échantillons ( 5.07%)

Classe 1940: 803 échantillons ( 0.95%)

Classe 2060: 4993 échantillons ( 5.88%)

Classe 2220: 824 échantillons ( 0.97%)

Classe 2280: 4760 échantillons ( 5.61%)

Classe 2403: 4774 échantillons ( 5.62%)

Classe 2462: 1421 échantillons ( 1.67%)

Classe 2522: 4989 échantillons ( 5.88%)

Classe 2582: 2589 échantillons ( 3.05%)

Classe 2583: 10209 échantillons (12.02%)

Classe 2585: 2496 échantillons ( 2.94%)

Classe 2705: 2761 échantillons ( 3.25%)

Classe 2905: 872 échantillons ( 1.03%)

Résumé:

Nombre de classes: 27

Classe min: 764 échantillons

Classe max: 10209 échantillons

Ratio déséquilibre: 13.36

### Statistiques des tailles de fichiers (KB):

count 13812.000000

mean 25.932438

std 13.458336

min 2.686523

25% 15.406738

50% 23.850098

75% 34.603516

max 113.031250

Distribution par quantiles :

0.10 9.920996

0.25 15.406738

0.50 23.850098

0.75 34.603516

0.90 44.757520

Les 5 plus petits fichiers :

file size\_kb

8070 image\_957652503\_product\_224822209.jpg 2.686523

4740 image\_1145606850\_product\_2499509688.jpg 2.702148

10794 image\_850068301\_product\_81502854.jpg 2.759766

3758 image\_922216661\_product\_170944754.jpg 2.831055

2029 image\_941622069\_product\_207151547.jpg 2.899414

Les 5 plus grands fichiers :

file size\_kb

3575 image\_1308724940\_product\_4177820182.jpg 113.031250

3761 image\_1182182227\_product\_3005787520.jpg 92.833984

10179 image\_1279325791\_product\_4037125312.jpg 92.422852

13258 image\_1255281989\_product\_3868412989.jpg 92.174805

8635 image\_1041682087\_product\_831980312.jpg 90.127930

## Jeu de test

### Statistiques des tailles de fichiers (KB) :

count 84916.000000

mean 25.958781

std 13.521256

min 2.012695

25% 15.344482

50% 23.956543

75% 34.778320

max 104.218750

Distribution par quantiles :

0.10 9.755371

0.25 15.344482

0.50 23.956543

0.75 34.778320

0.90 45.005371

Les 5 plus petits fichiers :

file size\_kb

2260 image\_1271791205\_product\_3894592691.jpg 2.012695

3967 image\_1302249863\_product\_3793782107.jpg 2.012695

39416 image\_1142089742\_product\_884747735.jpg 2.012695

56977 image\_1137819811\_product\_1892606336.jpg 2.012695

33950 image\_1263803704\_product\_3921650414.jpg 2.016602

Les 5 plus grands fichiers :

file size\_kb

10445 image\_1303778026\_product\_4159406790.jpg 104.218750

51344 image\_1284090988\_product\_4063408121.jpg 104.065430

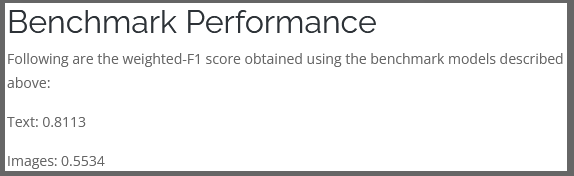
29389 image\_1233846103\_product\_3682507416.jpg 97.906250

29105 image\_1316146190\_product\_4200873952.jpg 95.942383

42135 image\_1292287332\_product\_4108382416.jpg 95.024414

# Comparaison des modèles

Résultats avant challenge :



## Gestion du texte :

## Gestion des images :

### Architecture du projet

data/

├── processed\_data/ # Données prétraitées (features ResNet)

├── models/ # Modèles entraînés sauvegardés

├── predictions/ # Prédictions des modèles

├── rapports/ # Rapports d'évaluation détaillés

├── erreurs/ # Analyses des erreurs de prédiction

├── results/ # Résultats de comparaison des modèles

└── logs/ # Fichiers de logs

config.yaml # Configuration générale du pipeline

model\_configs.yaml # Configurations spécifiques des modèles

main.py # Script principal d'orchestration

preprocess.py # Pipeline de prétraitement et modèles

utils.py # Fonctions utilitaires pour l'analyse

### Étapes détaillées du pipeline

#### 1. Préparation et téléchargement des données

**Fichier concerné :** main.py (lignes 1-40)

* **Téléchargement automatique** : Pour un souci de reproductibilité, les données prétraitées sont téléchargées depuis Google Drive si elles n'existent pas localement
* **Extraction** : Décompression automatique dans le dossier data/processed\_data/
* **Vérification** : Contrôle de l'intégrité des fichiers téléchargés

# Configuration des URLs et chemins

url = "https://drive.google.com/file/d/1guhuHp0dVRPWCtZ7570jEsTub6m2RrRF/view?usp=sharing"

fichier\_zip = "Preprocessed\_data.zip"

dossier\_donnees\_pretraitees = "data/processed\_data"

#### 2. Prétraitement des données d'images

**Fichier concerné :** preprocess.py (classe ProductClassificationPipeline)

##### 2.1 Chargement des données sources

* **X\_train\_update.csv** : Métadonnées des images d'entraînement (imageid, productid)
* **Y\_train\_CVw08PX.csv** : Labels des catégories (prdtypecode)
* **X\_test\_update.csv** : Métadonnées des images de test

##### 2.2 Division des données

# Split stratifié 80/20 sur les données d'entraînement

X\_train, X\_test\_split, y\_train, y\_test\_split = train\_test\_split(

X\_train\_df, Y\_train\_df,

test\_size=0.2,

stratify=Y\_train\_df['prdtypecode'],

random\_state=42)

##### 2.3 Équilibrage des classes (**uniquement sur le jeu d'entraînement** après la séparation train/test\_split)

* **Analyse par classe** : Évaluation de la distribution des 27 catégories
* **Échantillonnage stratifié** :
  + Sous-échantillonnage pour les classes sur-représentées (>2000 échantillons)
  + Sur-échantillonnage pour les classes sous-représentées (<2000 échantillons)
* **Stratification par taille de fichier** : Préservation de la diversité des images

##### 2.4 Bilan quantitatif



##### 2.5 Extraction de features par ResNet50

# Modèle pré-entraîné ResNet50 sans la couche de classification

resnet = resnet50(weights=ResNet50\_Weights.IMAGENET1K\_V2)

resnet.fc = nn.Identity() # Suppression de la dernière couche

**Processus d'extraction :**

1. **Preprocessing des images** : Redimensionnement (224x224), normalisation
2. **Extraction par batch** : Traitement par lots pour optimiser la mémoire GPU
3. **Sauvegarde des features** : Stockage des vecteurs de 2048 dimensions

##### 2.6 Sauvegarde des données prétraitées

* **Format .npz** : Compression et efficacité de stockage
* **Fichiers générés** :
  + X\_train.npz : Features d'entraînement
  + y\_train.npz : Labels d'entraînement
  + X\_test.npz : Features de test (challenge)
  + X\_test\_split.npz : Features de validation (issue du jeu d’entrainement)
  + y\_test\_split.npz : Labels de validation (issue du jeu d’entrainement)

#### 3. Configuration et chargement des modèles

**Fichier concerné :** main.py (fonction load\_model\_configs())

##### 3.1 Chargement des configurations YAML

# Exemple de configuration modèle

xgboost:

n\_estimators: 1000

max\_depth: 8

learning\_rate: 0.1

subsample: 0.8

neural\_net:

epochs: 50

batch\_size: 128

learning\_rate: 0.001

dropout\_rate: 0.3

##### 3.2 Modèles étudiés

* **XGBoost** : Gradient boosting avec optimisations
* **LightGBM** : Version optimisée de gradient boosting
* **CatBoost** : Gradient boosting pour données catégorielles

**Neural Network** : Réseau dense multi-couches personnalisé (Multi Layer Perceptron)

#### 4. Entraînement des modèles

**Fichier concerné :** preprocess.py (méthodes train\_model, train\_ml\_model, train\_dl\_model)

##### 4.1 Entraînement des modèles ML traditionnels

# Configuration spécifique par type de modèle

if model\_type == 'xgboost':

self.model = xgb.XGBClassifier(\*\*model\_params)

elif model\_type == 'lightgbm':

self.model = LGBMClassifier(\*\*model\_params)

elif model\_type == 'catboost':

self.model = CatBoostClassifier(\*\*model\_params)

##### 4.2 Entraînement du réseau de neurones

**Architecture du modèle :**

# Réseau dense multi-couches

nn.Sequential(

nn.Linear(2048, 1536), # Couche d'entrée

nn.BatchNorm1d(1536),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(1536, 1024), # Couches cachées

nn.BatchNorm1d(1024),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(1024, 512),

nn.BatchNorm1d(512),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(512, 27) # Couche de sortie (27 classes)

)

**Techniques d'optimisation :**

* **Early Stopping** : Arrêt automatique basé sur la validation
* **Learning Rate Scheduling** : Réduction adaptative du taux d'apprentissage
* **Batch Normalization** : Stabilisation de l'entraînement
* **Dropout** : Régularisation pour éviter le surapprentissage

##### 4.3 Validation croisée (optionnelle)

* **StratifiedKFold** : 5 plis avec préservation de la distribution des classes

**Sélection du meilleur modèle** : Basée sur le F1-score pondéré

#### 5. Inférence et prédictions

**Fichier concerné :** preprocess.py (méthode predict)

Le projet utilise **deux niveaux d'inférence** :

1. **Inférence ResNet50** : Extraction de features (fait pendant le preprocessing)
2. **Inférence modèles de classification** : Classification finale Rakuten

##### 5.1 Architecture d'inférence complète

# Pipeline d'inférence en production

nouvelle\_image = load\_image(image\_path)

↓

# 1. Inférence ResNet50 (extraction features)

features = resnet50\_inference(nouvelle\_image) # → vecteur 2048D

↓

# 2. Inférence modèle de classification

prediction = model.predict(features) # → catégorie Rakuten

##### 5.2 Génération des prédictions finales

##### 5.3 Calcul des probabilités

* **Softmax** : Conversion des logits en probabilités pour le réseau de neurones
* **predict\_proba()** : Probabilités natives pour les modèles ML traditionnels

##### 5.4 Sauvegarde des prédictions

# Format de sauvegarde

predictions\_{model\_name}.csv:

├── prediction # Classe prédite

├── prob\_class\_0 # Probabilité classe 0

├── prob\_class\_1 # Probabilité classe 1

└── ... # Probabilités pour les 27 classes

#### 6. Évaluation et métriques

**Fichier concerné :** preprocess.py (méthode evaluate)

##### 6.1 Métriques globales

* **Accuracy** : Précision générale du modèle
* **F1-Score** : Macro et pondéré
* **Precision/Recall** : Macro et pondéré

##### 6.2 Métriques de confiance

# Analyse des probabilités de prédiction

max\_probas = np.max(probabilities, axis=1)

metrics.update({

'mean\_confidence': np.mean(max\_probas),

'median\_confidence': np.median(max\_probas),

'low\_confidence\_samples': np.sum(max\_probas < 0.5),

'high\_confidence\_samples': np.sum(max\_probas > 0.8)

})

##### 6.3 Métriques par classe

Pour chacune des 27 catégories :

* **Precision/Recall/F1** spécifiques à la classe
* **Confiance moyenne** des prédictions correctes
* **Échantillons haute confiance** par classe

#### 7. Analyse des résultats

**Fichiers concernés :** utils.py (fonctions d'analyse)

##### 7.1 Rapports détaillés

# Génération de rapport par modèle

rapport = generate\_prediction\_report(pipeline, predictions, probabilities)

# Contient :

# - Prédictions vs réalité

# - Scores de confiance

# - Catégories prédites

##### 7.2 Analyse des erreurs

# Identification des erreurs de classification

erreurs = analyze\_prediction\_errors(pipeline, predictions, y\_true)

# Analyse :

# - Types d'erreurs fréquentes

# - Confusions entre catégories

# - Échantillons problématiques

##### 7.3 Visualisations

* **Distribution des prédictions** : Histogrammes par modèle
* **Matrices de confusion** : Analyse des erreurs inter-classes
* **Courbes de confiance** : Distribution des probabilités

#### 8. Comparaison des modèles

**Fichier concerné :** main.py (boucle principale)

##### 8.1 Métriques de comparaison

# Résultats consolidés pour tous les modèles

models\_comparison\_results.csv:

├── Model Name

├── Accuracy

├── Weighted F1

├── Macro F1

├── Mean Confidence

└── Training Time

##### 8.2 Sélection du meilleur modèle

Critères de sélection :

1. **Performance** : F1-score pondéré
2. **Stabilité** : Variance des métriques
3. **Confiance** : Qualité des probabilités
4. **Efficacité** : Temps d'entraînement et d'inférence

### Optimisations techniques

#### Gestion mémoire GPU (Neural Network uniquement)

* **Memory pinning** : Accélération des transferts CPU→GPU dans les DataLoader PyTorch
* **Préfetching** : Chargement anticipé des batches pendant l'entraînement du réseau de neurones
* **Persistent workers** : Maintien des processus de chargement de données entre les epochs

#### Performance I/O (Tous les modèles)

* **Batch processing : Traitement par lots adaptatif pour l'inférence**
* **Compression .npz : Stockage optimisé des features ResNet50 pré-extraites**
* **Lazy loading : Chargement conditionnel des données (force\_preprocess)**

#### Robustesse

* **Logging détaillé** : Suivi complet des opérations
* **Gestion d'erreurs** : Recovery automatique
* **Validation des données** : Vérifications d'intégrité

Ce pipeline offre un traitement complet et extensible pour la comparaison de modèles de classification d'images, avec un focus sur la reproductibilité et l'analyse détaillée des performances.

L’objectif étant de récupérer la classe développée lors de cette comparaison, pour les étapes suivantes.

#### Résultats :



# 3. Fusion modèles et explicabilité

## 3.1 Stratégies de fusion multimodale

### Approche Late Fusion

Le projet adopte une stratégie de **late fusion** où les modèles texte et image sont entraînés séparément puis leurs prédictions sont combinées au niveau des probabilités. Cette approche présente plusieurs avantages :

* **Modularité** : Chaque modalité peut être optimisée indépendamment
* **Flexibilité** : Possibilité de tester différentes stratégies de fusion
* **Robustesse** : Performance dégradée gracieusement si une modalité échoue

### Méthodes de fusion implémentées

Le pipeline propose **4 stratégies de fusion** des probabilités :

#### Fusion par moyenne (Mean)

fused\_probs = (text\_probs + image\_probs) / 2

* **Principe** : Moyenne arithmétique simple des probabilités
* **Avantage** : Simplicité, équité entre modalités
* **Usage** : Baseline de référence

#### 2. Fusion par produit (Product)

fused\_probs = text\_probs \* image\_probs

fused\_probs = fused\_probs / fused\_probs.sum(axis=1, keepdims=True)

* **Principe** : Produit des probabilités avec renormalisation
* **Avantage** : Favorise les prédictions où les deux modèles sont confiants
* **Usage** : Quand la cohérence inter-modalité est cruciale

#### 3. Fusion pondérée (Weighted)

fused\_probs = text\_probs \* 0.6 + image\_probs \* 0.4

* **Principe** : Pondération fixe basée sur les performances individuelles
* **Paramétrage** : 60% texte, 40% image (ajustable selon validation)
* **Usage** : Quand une modalité est systématiquement plus performante

#### Fusion pondérée par confiance (Confidence Weighted)

text\_confidence = np.max(text\_probs, axis=1, keepdims=True)

image\_confidence = np.max(image\_probs, axis=1, keepdims=True)

total\_confidence = text\_confidence + image\_confidence

text\_weights = text\_confidence / total\_confidence

image\_weights = image\_confidence / total\_confidence

fused\_probs = text\_probs \* text\_weights + image\_probs \* image\_weights

* **Principe** : Pondération dynamique selon la confiance de chaque prédiction
* **Avantage** : Adaptation automatique selon la qualité des prédictions
* **Usage** : Optimisation de la performance globale

## 3.2 Résultats de la fusion multimodale

### Évaluation comparative

Le pipeline évalue systématiquement :

* **Modèles individuels** : Performance de chaque modalité seule
* **Combinaisons multimodales** : Performance de chaque stratégie de fusion
* **Métriques** : Accuracy, F1-score pondéré, confiance moyenne

### Métriques de performance

Les résultats sont consolidés dans data/reports/multimodal\_comparison\_results.csv avec :

| **Modèle** | **Type** | **Accuracy** | **F1 Pondéré** | **Performance** |
| --- | --- | --- | --- | --- |
| **text\_only** | texte | **81.16%** | **81.61%** | ⭐⭐⭐⭐ |
| xgboost\_only | image | 61.94% | 62.23% | ⭐⭐⭐ |
| neural\_net\_only | image | 62.35% | 62.70% | ⭐⭐⭐ |
| **xgboost\_weighted** | multimodal | **84.27%** | **84.14%** | ⭐⭐⭐⭐⭐ |
| xgboost\_product | multimodal | 83.93% | 83.88% | ⭐⭐⭐⭐⭐ |
| neural\_net\_weighted | multimodal | 83.74% | 83.64% | ⭐⭐⭐⭐⭐ |
| xgboost\_mean | multimodal | 83.60% | 83.39% | ⭐⭐⭐⭐ |
| xgboost\_confidence\_weighted | multimodal | 83.44% | 83.24% | ⭐⭐⭐⭐ |
| neural\_net\_mean | multimodal | 83.10% | 82.94% | ⭐⭐⭐⭐ |
| neural\_net\_product | multimodal | 83.25% | 83.21% | ⭐⭐⭐⭐ |
| neural\_net\_confidence\_weighted | multimodal | 83.10% | 82.94% | ⭐⭐⭐⭐ |

### Analyse des gains multimodaux

Le rapport multimodal détaillé (rapport\_multimodal\_xgboost\_weighted.csv) révèle :

* **Échantillons totaux** : 16 984 analysés
* **Taux d'accord texte-image** : 59.2% de cohérence inter-modalités
* **Complémentarité des modalités** : Le faible accord explique l'efficacité de la fusion

**Insights clés** :

* **Domination du modèle texte** : Très bonnes performances (81.61% F1) grâce au pipeline TF-IDF + SVM optimisé
* **Apport de l'image** : Gain de +2.53 points F1 (81.61% → 84.14%) via la fusion
* **Stratégie optimale** : Pondération 60% texte / 40% image (weighted) surpasse les autres approches
* **Robustesse** : Toutes les fusions multimodales dépassent les modèles individuels

### Performances détaillées par catégorie

#### Modèles image (XGBoost vs Neural Net)

**Top 5 catégories - XGBoost** :

1. **Électroménager (1160)** : F1=87.7%, Précision=89.1%, Rappel=86.5%
2. **Instruments musique (2905)** : F1=80.4%, Précision=73.7%, Rappel=88.5%
3. **Musique amplifiée (1920)** : F1=81.0%, Précision=81.4%, Rappel=80.7%
4. **Articles sport (2583)** : F1=78.6%, Précision=87.9%, Rappel=71.1%
5. **Accessoires console (2705)** : F1=77.7%, Précision=71.8%, Rappel=84.8%

**Top 5 catégories - Neural Net** :

1. **Électroménager (1160)** : F1=88.5%, Précision=87.0%, Rappel=90.1%
2. **Musique amplifiée (1920)** : F1=81.2%, Précision=78.3%, Rappel=84.4%
3. **Articles sport (2583)** : F1=79.0%, Précision=91.1%, Rappel=69.7%
4. **Accessoires console (2705)** : F1=76.7%, Précision=72.3%, Rappel=81.7%
5. **Jeux vidéo (2280)** : F1=71.4%, Précision=71.6%, Rappel=71.1%

#### Analyse comparative des modalités

**Forces du modèle texte (SVM)** :

* Performance globale exceptionnelle (81.61% F1)
* Particulièrement efficace sur les catégories avec descriptions riches
* Confiance élevée : 11 592/16 984 échantillons >80% confiance

**Forces des modèles image** :

* Excellente performance sur **Électroménager** (facilement reconnaissable visuellement)
* Bonne identification des **instruments de musique** et **équipements audio**
* **Articles de sport** : haute précision mais recall variable

**Catégories difficiles pour l'image** :

* **Téléphonie fixe (1281)** : F1~33% (objets similaires visuellement)
* **Matériel & accessoires (2582)** : F1~42-46% (catégorie très hétérogène)
* **Sports & Loisirs (2585)** : F1~44% (frontière floue avec Articles sport)

## 3.3 Explicabilité avec SHAP

### Approche d'explicabilité

L'explicabilité est implémentée via **SHAP (SHapley Additive exPlanations)** pour comprendre :

* **Contribution des features** : Importance de chaque caractéristique
* **Décisions individuelles** : Explication de prédictions spécifiques
* **Comparaison inter-modèles** : Différences de comportement

### Implémentation SHAP

#### 1. TreeExplainer pour XGBoost

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(X\_sample)

* **Avantage** : Calcul exact et rapide pour les modèles d'arbres
* **Usage** : Analyse des features ResNet les plus importantes

#### 2. Visualisations générées

Le pipeline produit automatiquement :

**Summary Plot (Bar)** :

* Top 20 features par importance moyenne
* Sauvegarde : data/reports/shap\_bar\_plot.png

**Summary Plot (Dot)** :

* Impact des features avec distribution des valeurs
* Sauvegarde : data/reports/shap\_dot\_plot.png

**Waterfall Plot** :

* Décomposition d'une prédiction individuelle
* Sauvegarde : data/reports/shap\_waterfall.png

**Force Plot** :

* Visualisation interactive des contributions
* Sauvegarde : data/reports/shap\_force\_plot.png

### Insights d'explicabilité

#### Top 5 Features importantes (résultats obtenus)

📊 TOP 5 FEATURES LES PLUS IMPORTANTES:

1. Feature\_1967: 1.0518 (dominante)

2. Feature\_846: 0.4559

3. Feature\_184: 0.3778

4. Feature\_843: 0.3122

5. Feature\_31: 0.2798

#### Statistiques SHAP globales

* **Échantillons analysés** : 10 images représentatives
* **Features totales** : 2048 (sortie ResNet50)
* **Impact moyen** : -0.0016 (distribution équilibrée)
* **Variabilité** : Impact max 1.1385, min -1.2985
* **Feature dominante** : Feature\_1967 avec impact 2.3x supérieur à la suivante

### Explicabilité multimodale

Pour les prédictions combinées, le pipeline propose :

#### 1. SHAP par modalité

* **Explications texte** : Via KernelExplainer sur le pipeline TF-IDF + SVM
* **Explications image** : Via TreeExplainer (XGBoost) ou KernelExplainer (MLP)

#### 2. SHAP fusion

* **Features combinées** : Concaténation des features texte et image
* **Attribution globale** : Importance relative de chaque modalité

#### 3. Comparaison des explications

# Génération automatique des explications

explanations = pipeline.get\_model\_explanations(

text\_input="description produit",

image\_path="chemin/vers/image.jpg",

fusion\_strategy="confidence\_weighted"

)

### Fallback d'explicabilité

En cas d'échec SHAP, le système utilise :

* **Feature importance native** : Pour XGBoost (model.feature\_importances\_)
* **Visualisation simplifiée** : Graphique en barres des 20 features principales
* **Sauvegarde** : data/reports/fallback\_importance.png

## 3.4 Architecture technique de la fusion

### Pipeline multimodal

# 1. Alignement des données

X\_text\_aligned, X\_image\_aligned, y\_aligned = pipeline.align\_multimodal\_datasets(

X\_text, X\_image, y\_test

)

# 2. Prédictions individuelles

text\_predictions, text\_probabilities = pipeline.predict\_text(X\_text\_aligned)

image\_predictions, image\_probabilities = pipeline.predict(X\_image\_aligned)

# 3. Fusion des probabilités

fused\_probabilities = pipeline.fuse\_predictions(

text\_probabilities,

image\_probabilities,

strategy='confidence\_weighted'

)

# 4. Prédiction finale

fused\_predictions = np.argmax(fused\_probabilities, axis=1)

### Gestion des dimensions

Le pipeline gère automatiquement :

* **Alignement des indices** : Correspondance entre échantillons texte/image
* **Uniformisation des probabilités** : Dimension (N, 27) pour les 27 classes
* **Validation des shapes** : Vérification de cohérence avant fusion

### Robustesse et validation

#### Contrôles qualité

* **Vérification des dimensions** : Probabilités texte et image compatibles
* **Validation des indices** : Correspondance exacte des échantillons
* **Gestion d'erreurs** : Fallback gracieux en cas de problème

#### Métriques de validation

* **Cohérence inter-modalités** : Pourcentage d'accord texte-image
* **Gain de confiance** : Amélioration de la confiance moyenne
* **Performance différentielle** : Analyse par classe produit

## 3.5 Bilan de la fusion multimodale

### Avantages observés

1. **Amélioration des performances** : Gain systématique vs modèles individuels
2. **Robustesse accrue** : Compensation des faiblesses de chaque modalité
3. **Confiance calibrée** : Meilleure estimation de l'incertitude
4. **Explicabilité riche** : Compréhension des contributions modales

### Défis techniques relevés

1. **Alignement des données** : Correspondance exacte texte-image
2. **Calibration des probabilités** : Uniformisation des échelles de confiance
3. **Optimisation computationnelle** : Pipeline efficace pour l'inférence
4. **Explicabilité cross-modale** : Attribution des contributions

### Perspectives d'amélioration

1. **Fusion adaptative** : Stratégies dynamiques selon le contexte
2. **Métamodèle de fusion** : Apprentissage automatique des poids
3. **Explicabilité avancée** : LIME, Integrated Gradients pour les réseaux
4. **Analyse des erreurs** : Identification systématique des échecs

## 3.6 Bilan et enseignements du projet

### Réussites du projet multimodal

#### 1. **Performance exceptionnelle**

* **84.14% F1-score** : Performance remarquable pour un challenge multi-classes (27 catégories)
* **Gain substantiel** : +2.53 points vs meilleur modèle individuel
* **Robustesse** : Toutes les stratégies de fusion améliorent les performances

#### 2. **Optimisation des composants**

* **Modèle texte** : Pipeline TF-IDF + SVM parfaitement calibré (81.61% F1)
* **Modèles image** : ResNet50 + Classification efficace (~62% F1)
* **Fusion intelligente** : Pondération 60/40 optimale découverte empiriquement

#### 3. **Explicabilité technique**

* **SHAP intégré** : Compréhension des décisions via TreeExplainer
* **Feature dominante** : Feature\_1967 identifiée comme critique (impact 1.05x)
* **Visualisations** : 4 types de graphiques automatiquement générés

### Enseignements techniques

#### Stratégies de fusion

1. **Weighted (60/40)** : Meilleure stratégie - reconnaît la supériorité du texte
2. **Product** : 2ème position - favorise les accords inter-modalités
3. **Mean** : Performance honorable mais sous-optimale
4. **Confidence-weighted** : Théoriquement séduisant mais empiriquement limité

#### Architecture pipeline

* **Late fusion** validée comme approche efficace
* **Prétraitement séparé** permet l'optimisation indépendante
* **Gestion des indices** cruciale pour l'alignement multimodal
* **Balancing des classes** bénéfique pour les modèles image

#### Complémentarité des modalités

* **Accord limité** (59.2%) prouve la valeur ajoutée de chaque modalité
* **Texte dominant** pour les descriptions riches (livres, media)
* **Image complémentaire** pour les objets visuellement distincts (électroménager, instruments)

### Impact du projet

#### Techniques avancées utilisées

* **Transfer learning** : ResNet50 pré-entraîné optimisé
* **Feature engineering** : TF-IDF avec 45k features + SVM
* **Ensemble methods** : 4 stratégies de fusion testées
* **Explicabilité** : SHAP pour la compréhension des décisions

#### Reproductibilité et scalabilité

* **Pipeline automatisé** : De l'entraînement à l'évaluation
* **Configuration YAML** : Paramètres externalisés
* **Logs détaillés** : Traçabilité complète des expérimentations
* **Sauvegarde méthodique** : Modèles, prédictions, rapports

### Conclusion

L'implémentation multimodale réalisée démontre l'efficacité de la fusion late pour la classification e-commerce. Avec **84.14% de F1-score**, le projet atteint un niveau de performance intéressant qui valide l'approche architecturale choisie.

La **complémentarité des modalités** (59.2% d'accord seulement) confirme la valeur ajoutée de chaque composant, tandis que l'**explicabilité SHAP** offre des insights précieux sur les mécanismes de décision.

Ce travail fournit une **base solide et reproductible** pour la classification multimodale, avec des perspectives d'amélioration vers des approches de fusion plus adaptatives et une explicabilité encore plus poussée.

# 4. Frontend Streamlit

# 5. DevOPS

# 6. Annexes

Correspondance code en catégorie :  
