

# Classifiez automatiquement des biens de consommation



Projet n°6

Laureenda  
DEMEULE



OPENCCLASSROOMS



- ❏ Contexte et objectifs
- ❏ Exploration des données
- ❏ Étude de faisabilité
- ❏ Classification supervisée
- ❏ Résultats et performances
- ❏ API et déploiement
- ❏ Conclusion et perspectives

# Sommaire

## Contexte

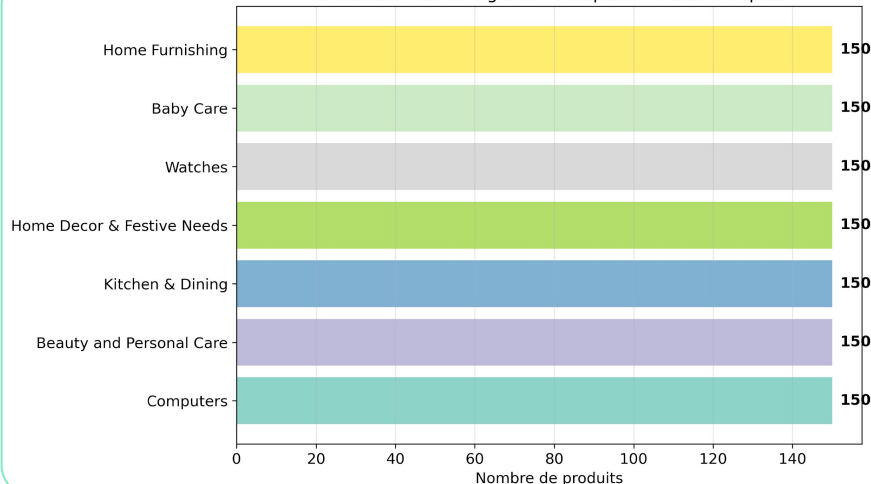
- Place de Marché
  - marketplace anglophone
- Classification manuelle peu fiable et chronophage
- Automatiser la classification des produits

## Objectifs

- Etudier la faisabilité d'un moteur de classification automatique des articles en catégories
  -  À partir des descriptions textuelles
  -  À partir des images des produits
- Enjeux → réduction des coûts, amélioration UX, scalabilité

-  1 050 produits
-  7 catégories équilibrées
-  100% images disponibles
-  Textes courts (~38 mots)
-  Hiérarchie 4+ niveaux

Distribution des Catégories Principales - Dataset Flipkart



## Exploration des données

# Étude de faisabilité

Nettoyage des données

Resources NLTK : stopwords, tokenizer, lemmatiseur

- ❑ texte en minuscules
- ❑ tokenisation : découpage de mots et suppression ponctuation
- ❑ lemmatisation : réduction des mots à leur forme de base et suppression des stopwords (mot sans valeurs informatives)

```
STOPWORDS = set(stopwords.words('english'))
LEMMATIZER = WordNetLemmatizer()
TOKENIZER_RE = re.compile(r"[A-Za-z]+")

def clean_text(text: str) -> str:
    """Nettoie le texte : minuscules, suppression ponctuation, stopwords, lemmatisation"""
    if not isinstance(text, str):
        return ''
    text = text.lower()
    tokens = TOKENIZER_RE.findall(text)
    tokens = [LEMMATIZER.lemmatize(t) for t in tokens if t not in STOPWORDS and len(t) > 2]
    return ' '.join(tokens)
```

# Étude de faisabilité

Extraction de caractéristiques :

- Texte : TF-IDF, Word2Vec
- Image : ORB-BoVW, MobileNet

## Texte

TF/IDF :

- Pondération intelligente
- Élimination du bruit

Word2Vec (Word Embeddings) :

- Représentation dense
- Robustesse

## Image

ORB + BoVW

- Regroupement grossier
- Sensible au bruit de fond

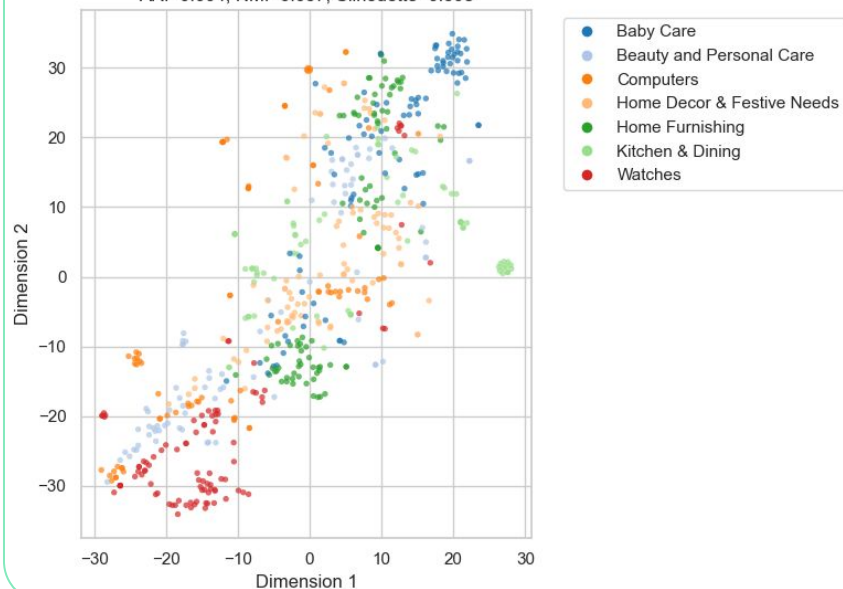
MobileNetV2 (Deep Learning)

- Représentations sémantiques de haut niveau
- Meilleure séparabilité visuelle

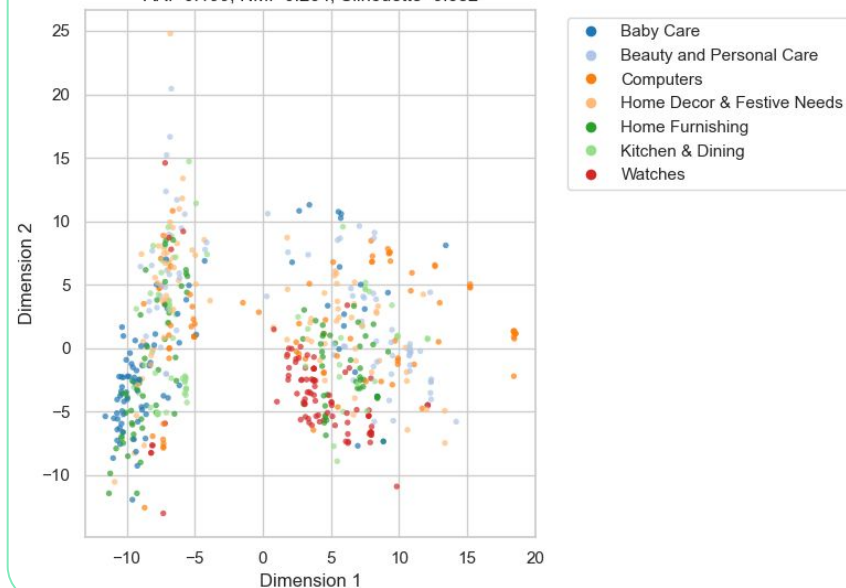
# Étude de faisabilité

## Réduction dimensionnelle et clustering

Texte TF-IDF + t-SNE + KMeans  
ARI=0.004, NMI=0.057, Silhouette=0.398



Texte Word2Vec + PCA + KMeans  
ARI=0.199, NMI=0.264, Silhouette=0.382



# Étude de faisabilité

## Réduction dimensionnelle et clustering

Image ORB-BoVW + PCA + KMeans  
ARI=0.030, NMI=0.065, Silhouette=0.068

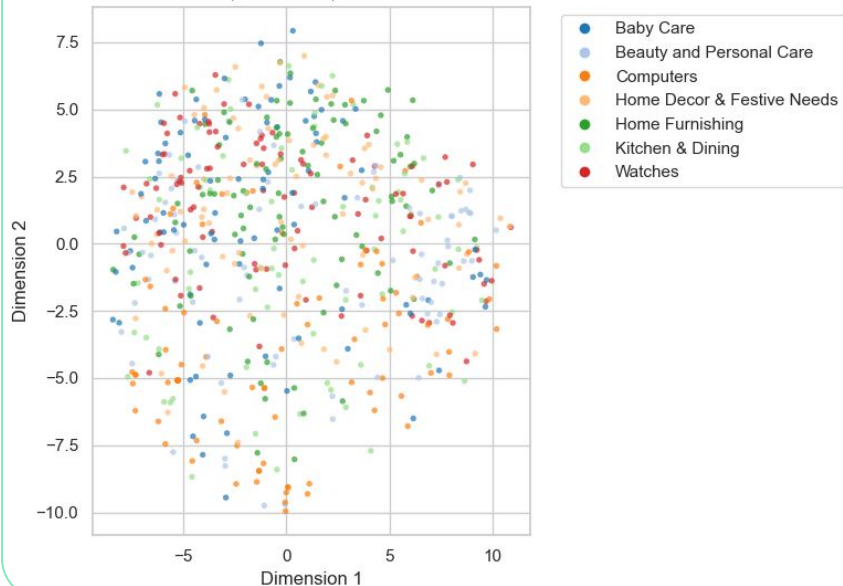
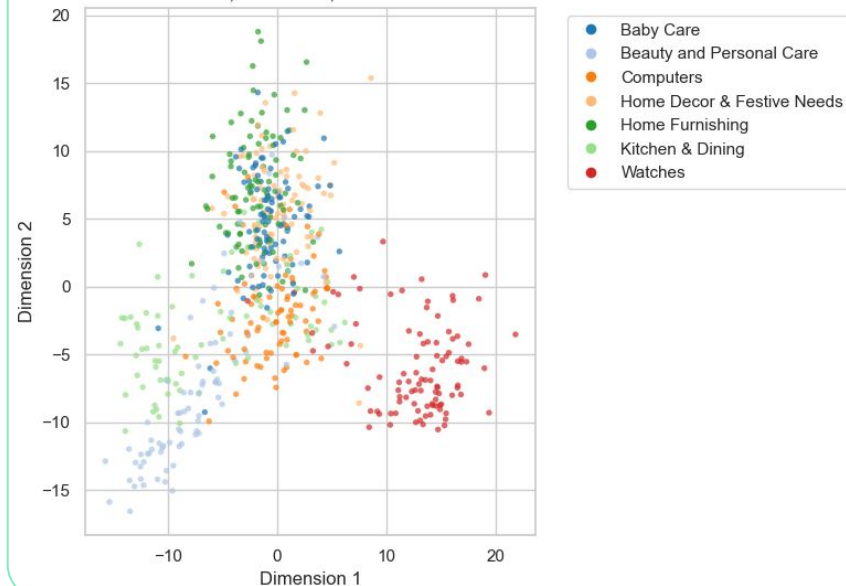


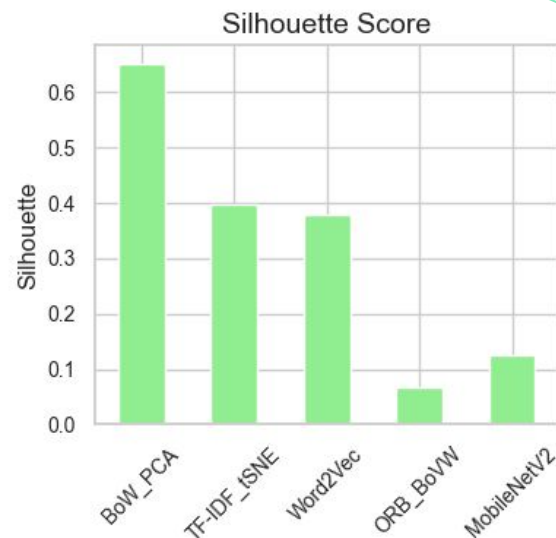
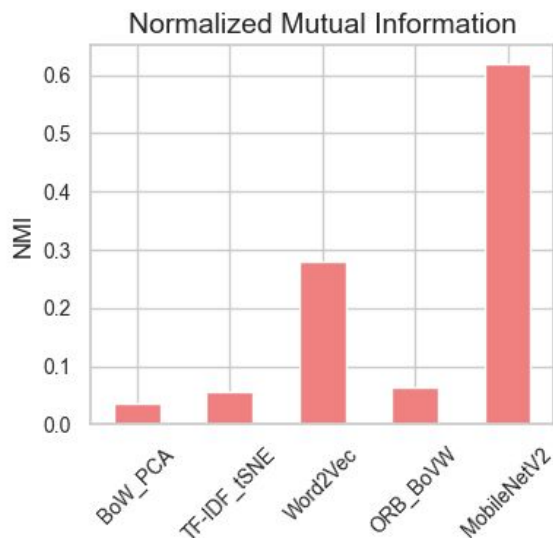
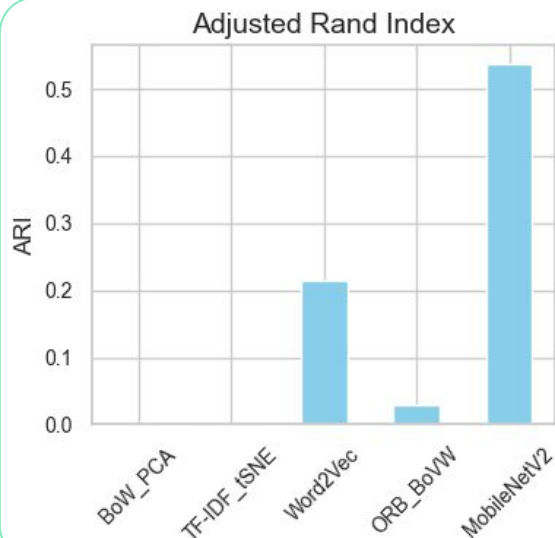
Image MobileNetV2 + PCA + KMeans  
ARI=0.538, NMI=0.620, Silhouette=0.126





# Étude de faisabilité

Évaluation : ARI, NMI, Silhouette Score



## Approches multiples testées



Classification textuelle :

- TF-IDF + ML (LogReg, SVM, RF, NB)



Classification visuelle :

- MobileNetV2 features + ML classique
- Fine-tuning MobileNetV2

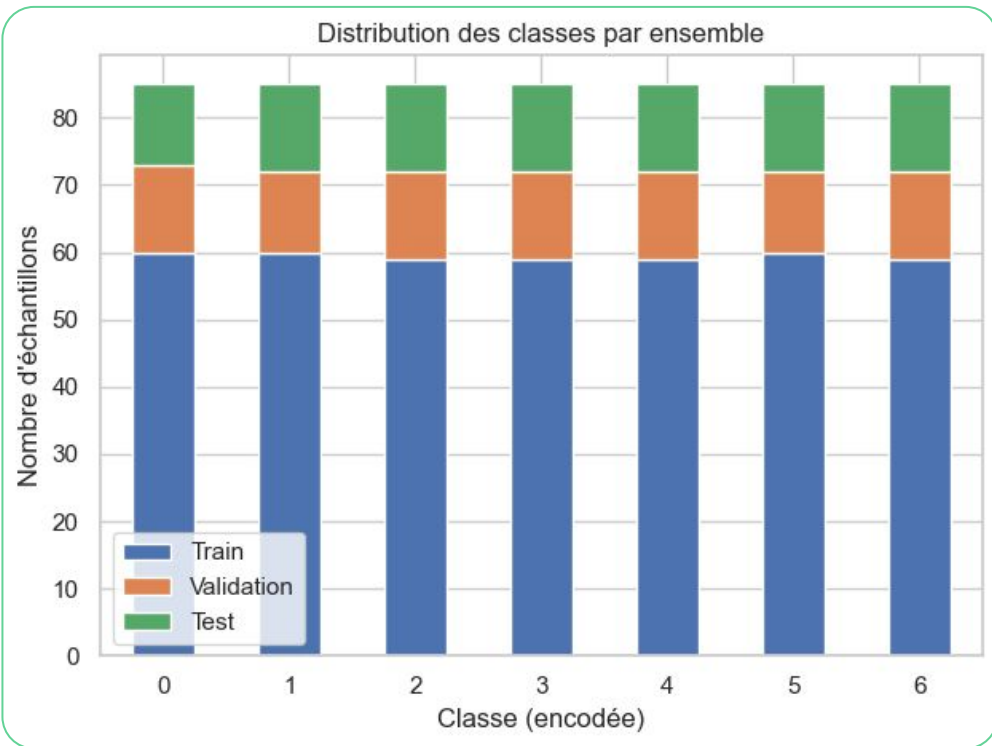


Deep Learning :

- Fusion multimodale texte + image classique et MobileNetV2

Classification  
supervisée

# Classification supervisée



Division stratifiée des données (train/val/test)

- 70% Entraînement
- 15% Validation
- 15% Test
- ✓ Équilibrage préservé
- 🎯 600 échantillons total

# Classification supervisée

## Classification sur caractéristiques textuelles et Modèles de référence (baseline)

```
=== BASELINES ===  
Baseline - Classe majoritaire: 0.144  
Baseline - Aléatoire stratifié: 0.133
```

```
=== CLASSIFICATION TEXTUELLE ===  
Forme TF-IDF train: (416, 2680)
```

```
Entraînement des modèles textuels...
```

```
--- Logistic Regression ---  
Validation: 0.955, Test: 0.933  
--- Naive Bayes ---  
Validation: 0.955, Test: 0.922  
--- Random Forest ---  
Validation: 0.899, Test: 0.856  
--- SVM ---  
Validation: 0.955, Test: 0.922
```

```
🏆 Meilleur modèle textuel: Logistic Regression  
Test accuracy: 0.933  
TF-IDF vectorizer sauvegardé.
```

TF-IDF + ML  
classiques (LR,  
NB, SVM, RF)

➤ Meilleur :  
Logistic  
Regression  
(93.3%)

## Classification sur caractéristiques visuelles

```
=== CLASSIFICATION VISUELLE ===  
Extraction des caractéristiques...  
Traité 416/416 images  
Traité 89/89 images  
Traité 90/90 images  
Formes: Train (416, 1280), Val (89, 1280), Test (90, 1280)
```

```
Entraînement des modèles visuels...
```

```
--- Logistic Regression ---  
Validation: 0.831, Test: 0.844  
--- Random Forest ---  
Validation: 0.831, Test: 0.767  
--- SVM ---  
Validation: 0.809, Test: 0.833
```

```
🏆 Meilleur modèle visuel: Logistic Regression  
Test accuracy: 0.844  
Image scaler sauvegardé.
```

MobileNetV2  
features + ML  
classiques

➤ Meilleur :  
Logistic  
Regression  
(84.4%)

# Classification supervisée

Classification sur  
caractéristiques visuelles

🏆 ÉVALUATION DU MODÈLE FINE-TUNÉ  
Évaluation du modèle fine-tuné sur le test set...

🔧 MobileNetV2 Fine-Tuné - Test Accuracy: 0.389 (38.9%)  
Predictions shape: (90, 7)  
Predicted classes range: [0, 6]  
True classes range: [0, 6]  
Unique predicted classes: [0 1 2 3 4 5 6]

📊 COMPARAISON AVEC FEATURE EXTRACTION:  
Feature Extraction: 0.844 (84.4%)  
Fine-Tuning: 0.389 (38.9%)  
Amélioration: -0.456 (-53.9%)

📌 Le fine-tuning nécessite peut-être plus d'ajustements

Fusion Multimodale

=== FUSION MULTIMODALE ===  
Variance expliquée par SVD: 1.000  
Dimensions fusionnées: (416, 1696)

Entraînement des modèles fusion...

--- Logistic Regression ---  
Validation: 0.831, Test: 0.844

--- Random Forest ---  
Validation: 0.876, Test: 0.856

--- Gradient Boosting ---  
Validation: 0.888, Test: 0.856

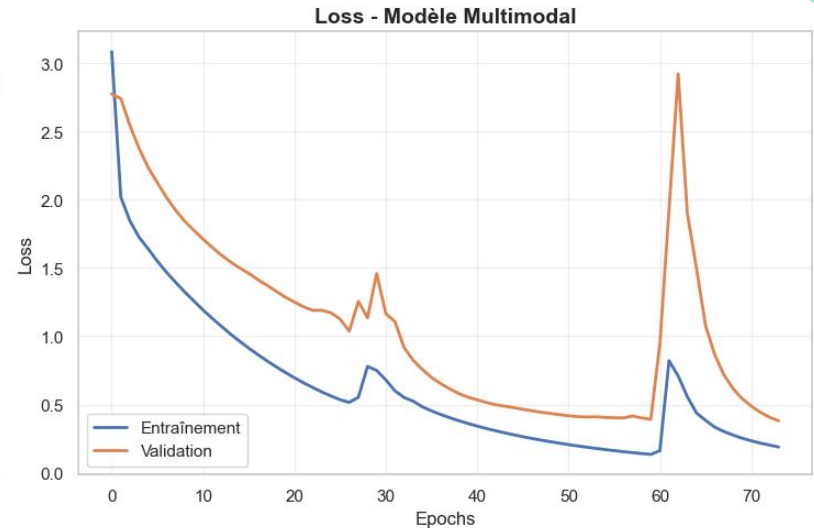
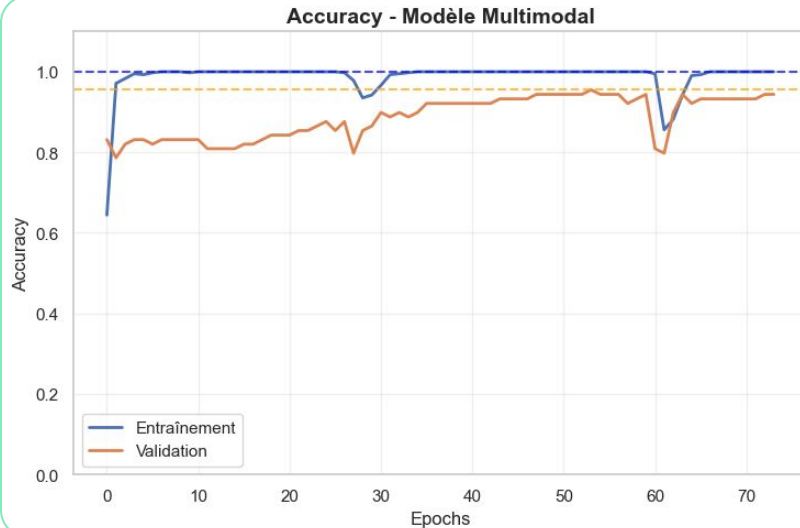
🏆 Meilleur modèle fusion: Gradient Boosting  
Test accuracy: 0.856

Concaténation  
features texte +  
image

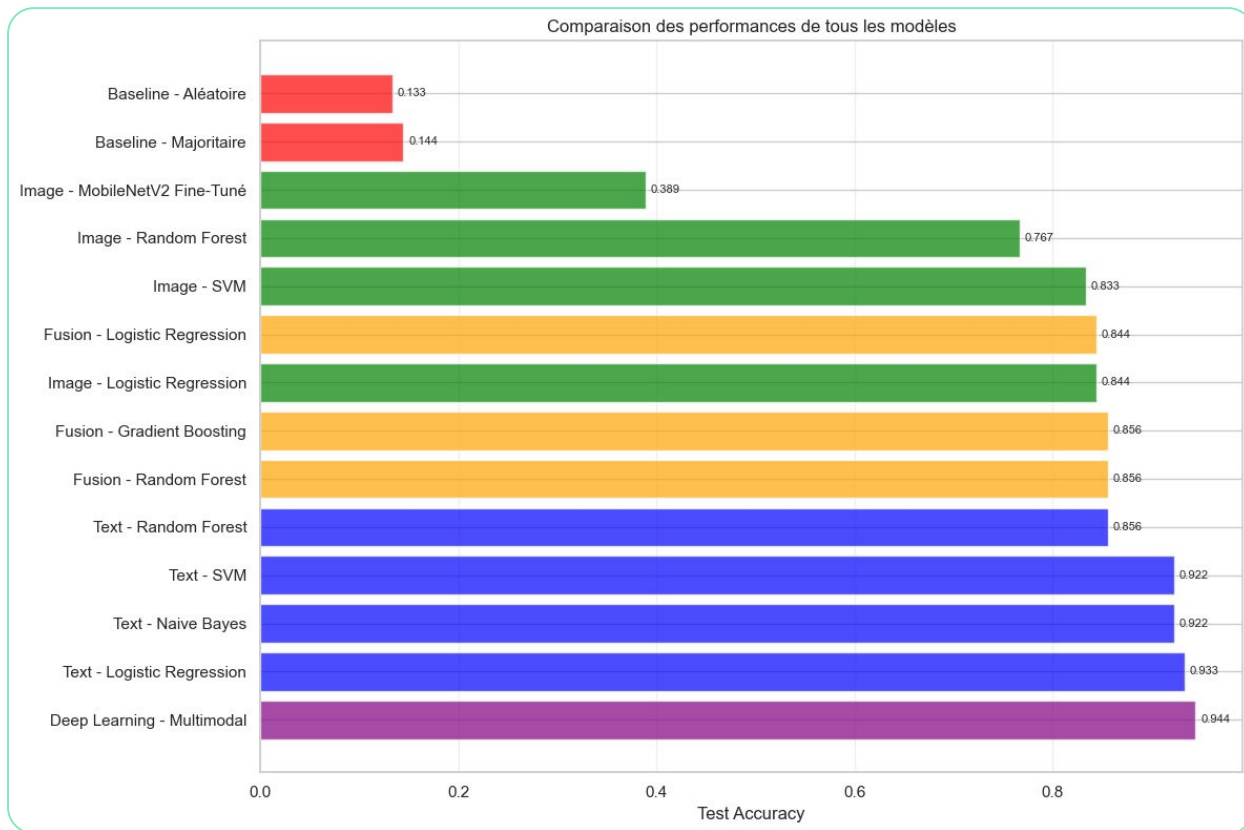
➤ Meilleur :  
Random  
Forest  
(85.6%)

# Classification supervisée

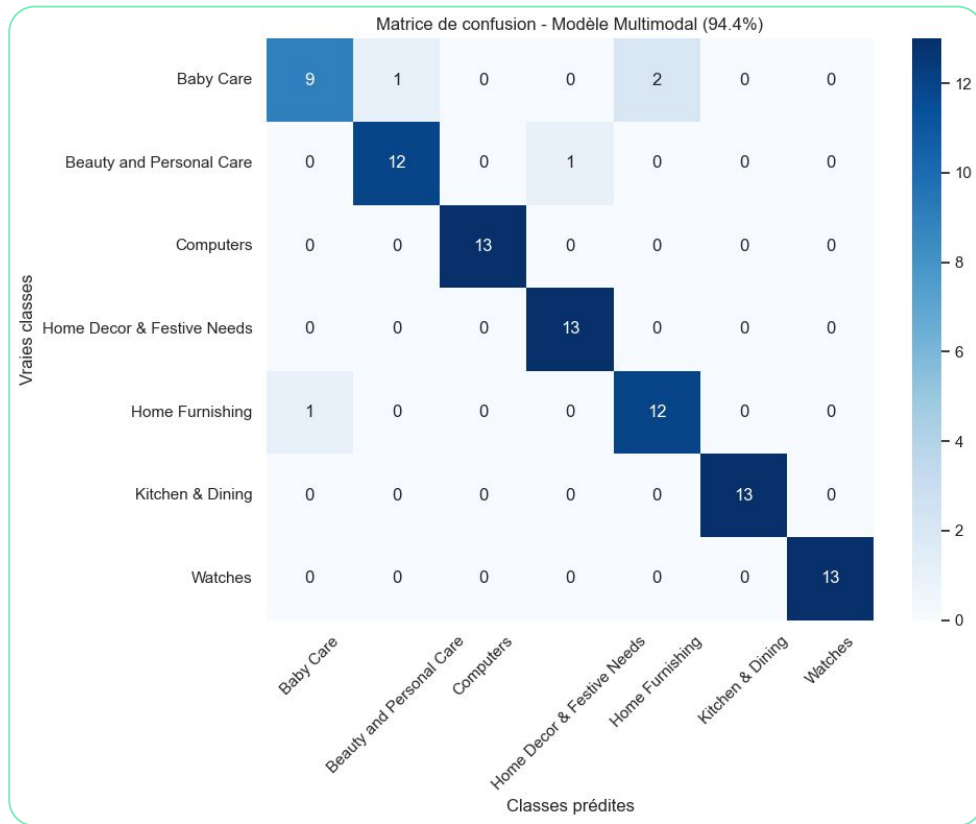
## Deep Learning Multimodal



# Résultats et performances



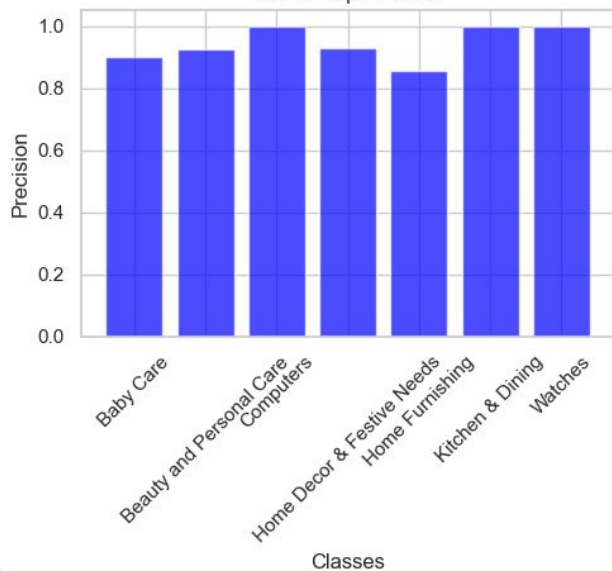
# Résultats et performances



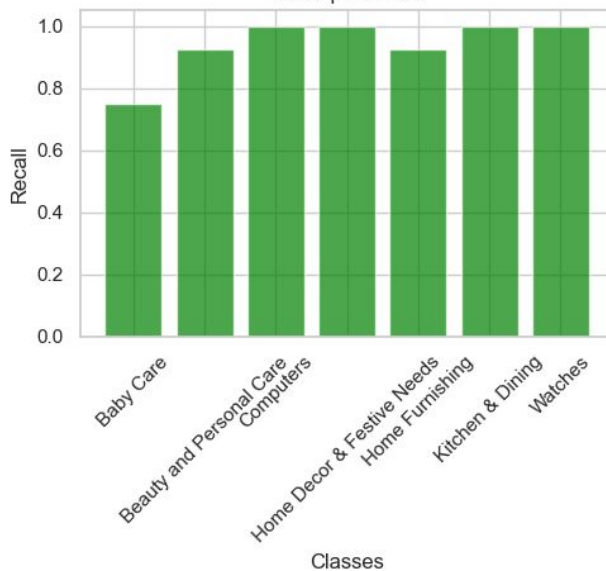


# Résultats et performances

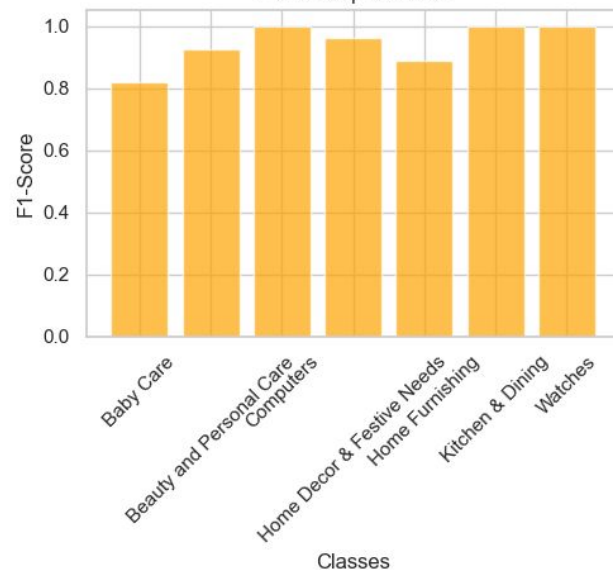
Precision par classe



Recall par classe



F1-Score par classe



# API et déploiement

## Curl

```
curl -X 'POST' \
  'http://0.0.0.0:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@0d3e1dc0d38be59e9927e47f152e48aa.jpg;type=image/jpeg'
```

## Request URL

`http://0.0.0.0:8000/predict`

## Server response

Code	Details
------	---------

200

## Response body

```
{
  "predicted_label": "Kitchen & Dining",
  "score": 0.99420565366745
}
```



FastAPI (Python) avec Swagger docs



Endpoints :



POST /predict : prédiction unitaire



POST /predict-batch : prédictions multiples



GET /health : santé de l'API



Pipeline optimisé :



Upload image → Preprocessing → Inférence → JSON



Gestion d'erreurs robuste



Validation des formats

## Conclusion

### 🎯 Objectifs atteints ✅

- Accuracy de 94.4% (objectif: 90%)
- Automatisation complète de la classification
- Pipeline de production prêt

## Perspectives

- Déploiement du modèle Deep Learning Multimodal
- Monitoring continu des performances
- Data augmentation pour améliorer la robustesse
- Collecte de feedback utilisateur



Merci de votre  
attention !