RAPPORT DE MODÉLISATION – Projet Trustpilot

Auteur : Pierre Poulouin, Alimira Rodrigues

Contexte : Analyse d'avis clients issus de Trustpilot (RingConn, OuraRing, Circular).

Objectif : prédire le ressenti global d'un client à partir du texte de son avis.

1. Étapes de réalisation du projet

Le projet a suivi une approche progressive et rigoureuse :

1. **Collecte et preprocessing**
   Fusion des colonnes Title et Content en une variable textuelle unique CleanText.
   Nettoyage des caractères spéciaux, mise en minuscules, suppression des stopwords, et lemmatisation.
   Résultat : un texte homogène prêt à être vectorisé via TF-IDF.
2. **Découpage des données**
   Split **stratifié 80/20** entre train.csv et test.csv à partir du fichier trustpilot_dataset_final_cleaned.csv.
   Cette stratification garantit la même proportion de notes (1–5) entre train et test.
3. **Modélisation multiclass (1 à 5)**
   Entraînement d'un modèle de **régression logistique (Logistic Regression)** sur 5 classes (notes 1–5).
   Première évaluation : bonnes performances sur les notes extrêmes (1 et 5), mais confusions fréquentes entre 3, 4 et 5.
4. **Transformation en classification binaire**
   Regroupement des notes : (1,2) → "mauvais", (3,4,5) → "bon".
   Justification : mieux aligné avec le besoin métier (*savoir si le client est satisfait ou non*).
   Analyse confirmée via les scripts de reporting (make_reports_from_predictions.py).

5. **Réentraînement du modèle binaire**
   Nouveau modèle TF-IDF + Logistic Regression, entraîné directement sur la cible binaire.
   Oversampling sur le jeu d'entraînement via RandomOverSampler pour équilibrer les classes.
   Évaluation sur le test set jamais vu.

---

2. Classification du problème

- **Type de problème :** Classification supervisée.
- **Nature de la tâche :** Analyse de sentiment (Sentiment Analysis).
- **Variable cible :** avis binaire (0 = mauvais, 1 = bon).
- **Variable d'entrée :** texte (CleanText).

---

3. Métriques de performance

Métrique principale :

F1-score

- Le **F1-score** a été choisi car il équilibre **précision** et **rappel**, ce qui est essentiel quand les classes sont légèrement déséquilibrées.
- Le projet vise à éviter les faux négatifs (ne pas rater un avis négatif), tout en limitant les faux positifs.
- En binaire, le F1 ≈ 0.96 → excellent équilibre.

Autres métriques :

- **F1-macro :** pour vérifier l'équilibre du modèle sur les deux classes (positif/négatif).
- **ROC-AUC (0.98)** : capacité du modèle à séparer correctement les deux classes.
- **PR-AUC (0.99)** : stabilité des prédictions positives même à fort rappel.
- **Accuracy (0.95)** : simple indicateur global, cohérent avec les autres mesures.

"Même en binaire, le F1-macro est conservé pour s'assurer que le modèle ne favorise pas uniquement la classe majoritaire (les avis positifs). C'est aussi cohérent avec la phase multiclass où cette métrique était déjà utilisée."

---

4. Choix du modèle et optimisation

Modèles testés :

- **Logistic Regression** (modèle principal)
- **SVM linéaire** (testé brièvement, plus lent sans gain significatif)
- **Random Forest** (testé, moins performant en texte brut)

Modèle retenu :

TF-IDF + Logistic Regression

Rapide, robuste et interprétable.

Parfaitement adapté au texte (fonctionne bien sur des vecteurs TF-IDF).

Très bon compromis entre performance et explicabilité pour un projet pédagogique.

Optimisation :

- Utilisation d'une **validation croisée (CV à 5 folds)** sur le jeu d'entraînement pour valider la robustesse.
- Aucun **GridSearch lourd** n'a été nécessaire : la performance était déjà optimale avec les hyperparamètres standards (C=1.0, solver='lbfgs').
- L'accent a été mis sur la **reproductibilité** (même seed, même split).

Modèles avancés :

- Pas de Deep Learning ni de Boosting testés ici : la baseline TF-IDF + LogReg offre déjà une excellente performance (F1 = 0.96).
- Le gain potentiel d'un modèle complexe ne justifie pas la perte d'interprétabilité.

---

5. Interprétation des résultats

Analyse d'erreurs :

- Les rares erreurs concernent surtout des **avis ambigus** (ton neutre, ironie, ou note 3 floue).
- Ces erreurs ne proviennent pas d'un manque d'apprentissage mais de la **subjectivité naturelle du langage**.
-

Interprétabilité :

Le choix de la **régression logistique** permet une interprétation directe :
on peut identifier les mots les plus discriminants pour chaque classe (coef_ du modèle).
exemple : mots “excellent”, “perfect”, “recommend” → poids positif fort.
mots “disappointed”, “worst”, “refund” → poids négatif fort.

Synthèse :

“Le modèle reconnaît très bien les avis positifs et négatifs, même sans ajout de features complexes.
La stabilité entre le seuil 0.5 et le seuil optimal (0.51) prouve une calibration correcte.
Les erreurs sont rares et cohérentes avec les zones de flou sémantique.”

---

6. Améliorations possibles

6.1. Axes d’amélioration envisagée

1. **Ajout de features interprétables**
   (longueur du texte, nombre d’exclamations, ratio de majuscules, subjectivité).
   → Ces ajouts ont été testés dans un script séparé afin de mesurer leur impact sur la performance, avec un gain d’environ **+0.5 à +1 pt de F1-score**.
2. **Essais de modèles plus avancés**
   (SVM ou XGBoost) afin de comparer la baseline à des approches non linéaires.
3. **Industrialisation**
   → Sauvegarde du modèle (.joblib), reproductibilité et intégration future dans un script d’analyse automatisé multi-marques.

6.2. Amélioration de la baseline : ajout de features interprétables

Après validation d'un premier modèle binaire performant (TF-IDF + Logistic Regression), j'ai souhaité tester l'ajout de **features simples et explicables** afin de renforcer la robustesse du modèle sans en complexifier la structure.
Les variables ajoutées ont été sélectionnées pour leur **intérêt linguistique direct** :

- **text_length** : longueur du texte (nombre de caractères)
- **word_count** : nombre de mots
- **exclamation_count** : nombre de "!" (émotion / emphase)
- **upper_ratio** : ratio de lettres majuscules (intensité émotionnelle)

Ces variables, calculées à partir du texte brut, ont été **normalisées et combinées au TF-IDF** via un FeatureUnion.
Aucune transformation lourde n'a été appliquée afin de conserver un pipeline simple, lisible et interprétable.
Résultats obtenus :

- F1-score (seuil 0.5) : **0.958**
- F1-score (seuil optimal = 0.415) : **0.961**
- ROC-AUC : **0.979**

L'ajout de ces indicateurs a **légèrement renforcé la stabilité** du modèle et permis une **amélioration marginale du F1-score**, sans perte de précision.
Le modèle reste donc **simple, robuste et interprétable**, constituant une baseline solide pour la suite du projet.

6.3. Modélisation avancée : XGBoost et interprétation avec SHAP

Pour explorer la piste des **modèles avancés**, j'ai entraîné un **XGBoost Classifier** sur les mêmes données vectorisées par TF-IDF.
Ce modèle de gradient boosting permet de **capturer les interactions non linéaires** entre les mots et expressions, contrairement à la régression logistique.
Principaux paramètres utilisés :

- n_estimators = 300
- max_depth = 6
- learning_rate = 0.1

- subsample = 0.8
- colsample_bytree = 0.8

Un **RandomOverSampler** a été appliqué sur le jeu d'entraînement afin d'équilibrer les classes.

Résultats obtenus :

- F1-score (seuil 0.5) : **0.951**
- F1-score (seuil optimal = 0.475) : **0.952**
- ROC-AUC : **0.974**

Ces résultats montrent que **XGBoost atteint un niveau de performance similaire** à la baseline, confirmant que la représentation TF-IDF capture déjà l'essentiel du signal discriminant.

Le modèle est également **bien calibré**, avec un seuil optimal très proche de 0.5.

L'intérêt principal de XGBoost réside dans sa **robustesse sur les cas ambigus** et sa compatibilité avec les outils d'explicabilité comme **SHAP**.

Grâce à SHAP, j'ai pu analyser l'influence des mots sur les prédictions :

- **Mots positifs influents :** *amazing*, *recommend*, *great*, *perfect*
- **Mots négatifs influents :** *refund*, *disappointed*, *worst*, *bad*

Ces résultats confirment la **cohérence linguistique** du modèle et la qualité de sa compréhension des textes.

SHAP permet aussi de **visualiser les contributions locales**, par exemple sur un avis individuel, pour expliquer pourquoi un texte est classé "positif" ou "négatif".

Conclusion :

Le test de XGBoost a permis de **valider la robustesse du pipeline** et d'ajouter une **dimension d'explicabilité avancée**.

Les performances restent stables, prouvant que le modèle initial était déjà optimal, mais cette approche apporte une **valeur analytique supplémentaire** pour la compréhension métier.

---

7. Conclusion générale

Le projet Trustpilot démontre qu'une approche classique bien structurée (TF-IDF + Logistic Regression) peut atteindre des performances très élevées pour une tâche de classification de texte.

Les étapes de nettoyage, de vectorisation et de validation ont été rigoureusement respectées.

Le passage du multiclass au binaire a été une décision data-driven, justifiée par les résultats et validée empiriquement.

Le modèle binaire atteint **F1 = 0.96** et **AUC = 0.98**, tout en restant entièrement interprétable et stable.
C’est une base solide pour un futur modèle amélioré, déjà pleinement exploitable dans un cadre métier.