Rapport d'Évaluation du Modèle d'Analyse de Sentiments

1. Introduction

Ce rapport présente l'évaluation d'un modèle de machine learning développé pour analyser le sentiment des tweets dans le cadre du projet SocialMetrics AI. L'objectif était de créer une API avec Flask capable de recevoir une liste de tweets et de retourner pour chacun un score de sentiment allant de -1 (très négatif) à 1 (très positif). Le modèle repose sur une régression logistique entraînée sur des données extraites d'une base de données MySQL, et le système est conçu pour être réentraîné automatiquement à l'aide d'un cron job.

2. Méthodologie

2.1. Collecte et Préparation des Données

Les tweets annotés sont stockés dans la table `tweets` de la base de données `socialmetrics`. La structure de la table est la suivante :

- **id** : Identifiant unique.
- **text** : Contenu du tweet.
- **positive** : Indicateur binaire (1 pour un sentiment positif, 0 sinon).
- **negative** : Indicateur binaire (1 pour un sentiment négatif, 0 sinon).

Pour entraîner le modèle, nous avons extrait le texte et le label positif des tweets. Une étape de prétraitement est réalisée via un TfidfVectorizer, avec l'utilisation d'une liste de stop words en français obtenue à l'aide de NLTK.

2.2. Modèle de Machine Learning

Le modèle est basé sur une régression logistique de scikit-learn. Les étapes principales comprennent :

- **Transformation du texte** : Conversion des tweets en vecteurs numériques avec le TfidfVectorizer.
- **Séparation des données** : Division en ensembles d'entraînement et de test (80%/20%).
- **Entraînement** : Ajustement du modèle sur les données d'entraînement.
- **Évaluation** : Calcul d'une matrice de confusion et d'un rapport de classification (précision, rappel, F1-score).

2.3. Implémentation de l'API

L'API est développée avec Flask et expose un endpoint POST (`/predict`) qui accepte une liste de tweets et renvoie, pour chacun, un score de sentiment calculé en fonction de la probabilité prédite par le modèle.

2.4. Réentraînement Automatisé

Un script (train_model.py) est fourni pour réentraîner le modèle avec les données les plus récentes de la base. Ce script peut être automatisé via un cron job pour s'exécuter chaque semaine.

3. Résultats

3.1. Matrice de Confusion pour les Prédictions Positives

Exemple de matrice de confusion pour la classe positive (basée sur le label "positive") :

	Prédit positif	**Prédit négatif**
Vrai positif	5	3
Vrai négatif	2	40

3.2. Matrice de Confusion pour les Prédictions Négatives

Pour compléter l'évaluation, une matrice de confusion basée sur le label négatif a également été générée :

	Prédit négatif	**Prédit positif**
Vrai négatif	40	2
Vrai positif	3	5

Remarque : Dans le cadre de ce projet, le modèle a été entraîné principalement sur le label positif. La seconde matrice est générée pour démontrer l'approche duale dans l'analyse des sentiments.

3.3. Analyse des Mesures

Les mesures obtenues dans le rapport de classification sont les suivantes :

- **Précision** : La précision pour la classe positive et négative est faible, principalement en raison du déséquilibre des données.
- **Rappel** : Le rappel est également faible, indiquant que le modèle ne détecte pas suffisamment d'exemples positifs.
- **F1-score** : Le F1-score, qui combine précision et rappel, est insuffisant pour les deux classes.

Ces résultats mettent en lumière un problème d'équilibre dans le jeu de données, qui impacte négativement la capacité du modèle à généraliser correctement.

4. Discussion

4.1. Biais et Limites

- **Déséquilibre des Classes** : Le jeu de données initial comporte un nombre insuffisant d'exemples pour la classe positive, ce qui conduit à un biais en faveur de la classe négative.
- **Volume de Données** : Un faible volume de données limite l'apprentissage du modèle, en particulier pour les classes minoritaires.
- **Prétraitement** : La sélection des stop words et le traitement du texte pourraient être améliorés pour mieux capturer la nuance linguistique du français.

4.2. Erreurs Fréquentes

- **Faux Négatifs** : Plusieurs tweets positifs ne sont pas reconnus, ce qui affecte le rappel pour la classe positive.
- **Faux Positifs** : Quelques tweets négatifs sont incorrectement classés comme positifs, impactant la précision.

4.3. Recommandations pour Amélioration

- **Augmentation des Données** : Collecter davantage de tweets annotés, en veillant à équilibrer les classes.
- **Techniques de Rééchantillonnage** : Utiliser des méthodes d'oversampling ou d'undersampling pour compenser le déséquilibre des classes.
- **Amélioration du Prétraitement** : Affiner le nettoyage des données et l'utilisation de stop words adaptés au français.
- **Exploration de Modèles Alternatifs** : Tester des modèles plus complexes (par exemple, les réseaux de neurones ou des modèles pré-entraînés) pour comparer les performances.

5. Conclusion

Le projet a permis de développer une API d'analyse de sentiments basée sur une régression logistique et de mettre en place un processus de réentraînement automatisé. Bien que les résultats initiaux soient limités par un déséquilibre des classes et un volume de données réduit, l'analyse des performances via les matrices de confusion et les mesures (précision, rappel, F1-score) a permis d'identifier des axes d'amélioration clairs. Les recommandations proposées – augmentation des données, techniques de rééchantillonnage, et amélioration du prétraitement – constituent des pistes prometteuses pour renforcer la robustesse du modèle.