TP Final : Développer une API d'Analyse de Sentiments

Contexte

Vous travaillez pour une entreprise fictive, **SocialMetrics AI**, spécialisée dans l'analyse de données pour les réseaux sociaux. Le client, Daunale Treupe, souhaite surveiller les opinions exprimées sur **X (anciennement Twitter)**. Votre mission est de concevoir un service permettant d'évaluer le sentiment des tweets en fonction de leur contenu.

Objectifs

- Développer une API avec Flask.
- Implémenter un modèle de machine learning basé sur la régression logistique.
- Réentraîner régulièrement le modèle avec des données annotées.
- Générer et analyser des matrices de confusion pour évaluer les performances.

Besoins Fonctionnels

1. **Endpoint d'Analyse des Sentiments** :
    - Créer une API avec **Flask** comprenant un endpoint `POST`.
    - L'endpoint doit accepter une liste de tweets sous forme de tableau de chaînes (`string[]`).
    - Pour chaque tweet, calculer un score de sentiment entre **-1** (très négatif) et **1** (très positif).
    - Retourner la réponse au format JSON :

      ```
      {
        "tweet1": score1,
        "tweet2": score2,
        ...
      }
      ```

2. **Base de Données MySQL** :
    - Créer une table nommée `tweets` pour stocker les tweets annotés. Cette table servira de dataset pour entraîner le modèle.
    - Structure de la table `tweets` :
        - `id` : Identifiant unique.
        - `text` : Contenu du tweet.
        - `positive` : 1 si le tweet est jugé positif, 0 sinon.
        - `negative` : 1 si le tweet est jugé négatif, 0 sinon.

3. **Modèle de Machine Learning** :
    - Utiliser une **régression logistique** de `scikit-learn` pour analyser les sentiments.
    - Entraîner le modèle sur les données de la table `tweets` :
        - `positive` comme label pour les prédictions positives.
        - `negative` comme label pour les prédictions négatives.
    - Tester les performances du modèle sur un jeu de données de validation.

4. **Réentraînement du Modèle** :

- Mettre en place un mécanisme pour réentraîner le modèle chaque semaine avec les données les plus récentes de la table `tweets`.
- Automatiser ce réentraînement à l'aide d'un cronjob ou d'une tâche planifiée.

5. **Rapport d'Évaluation** :
   - Générer une **matrice de confusion** pour les prédictions positives (basées sur `positive`) et une autre pour les prédictions négatives (basées sur `negative`).
   - Rédiger une analyse des performances du modèle en se basant sur les observations suivantes :
     - Précision, rappel et F1-score pour les classes positives et négatives.
     - Identifier les biais éventuels dans les prédictions.
     - Proposer des pistes pour améliorer les performances.

Contraintes Techniques

- Utiliser **Python** et le framework **Flask** pour l'API.
- Utiliser une base de données **MySQL** pour stocker les données annotées.
- Implémenter le modèle de machine learning avec **LogisticRegression** de `scikit-learn`.
- Fournir un script pour automatiser le réentraînement du modèle.
- Documenter l'installation, l'utilisation de l'API, et les étapes de réentraînement.

Livrables Attendus

1. **Code Source** sur un repo **public** sur Github :
   - Fichiers Python pour l'API, le modèle et la gestion de la base de données.
   - Script pour configurer la base de données MySQL et créer la table `tweets`.
   - Script pour automatiser le réentraînement.
2. **Rapport d'Évaluation** en **pdf** sur Github aussi (dans le même repo) :
   - Les deux matrices de confusion (positive et négative).
   - Analyse détaillée des performances du modèle :
     - Précision, rappel et F1-score pour chaque classe.
     - Observations sur les erreurs fréquentes ou éventuels biais.
     - Recommandations pour améliorer le modèle.

Barème d'Évaluation

Catégorie	Critères	Points
1. Fonctionnalités de l'API		**50**
	Endpoint POST fonctionnel : accepte une liste de tweets et retourne les scores attendus	10
	Structure JSON correcte dans la réponse	10
	Connexion à la base de données MySQL fonctionnelle	10
	Gestion des erreurs : cas des listes vides, formats incorrects, etc.	10

	Documentation de l'API : guide clair avec exemples d'utilisation	10
2. Modèle de Machine Learning		**30**
	Implémentation correcte du modèle LogisticRegression	10
	Réentraînement automatisé via cronjob ou tâche planifiée	10
	Performance de base : prédictions cohérentes et fonctionnelles	10
3. Base de Données MySQL		**10**
	Structure correcte de la table `tweets` : colonnes `id`, `text`, `positive`, `negative`	5
	Stockage correct des données annotées	5
4. Rapport d'Évaluation		**30**
	Matrice de confusion pour les prédictions positives, avec interprétation	5
	Matrice de confusion pour les prédictions négatives, avec interprétation	5
	Calcul et interprétation des mesures : précision, rappel, F1-score	10
	Analyse des performances : forces, faiblesses, biais	5
	Recommandations claires pour améliorer le modèle	5
Total		**120**
