**Rapport de Projet : Tweet Suspect Detection**

**Nom du Projet :** Tweet Suspect Detection  
**Auteur :** Benewende Pierre Bonkoungou

Répertoire Git: https://github.com/VirtuelsDev/Tweet\_Suspect\_Detection.git

**Date :** 14 novembre 2024

# 1. Contexte et Objectifs

Les réseaux sociaux sont devenus des plateformes où les utilisateurs partagent divers types de contenu, mais cela inclut parfois des messages suspects ou nuisibles. **Tweet Suspect Detection** a pour objectif d’identifier de tels messages, afin de contribuer à la sécurité en ligne. En combinant des techniques de **Natural Language Processing (NLP)** et de **machine learning**, ce projet vise à développer une application capable de :

* **Nettoyer et préparer** des données textuelles (tweets),
* **Extraire des caractéristiques** pertinentes pour détecter les messages suspects,
* **Entraîner un modèle** de classification,
* **Déployer une interface** pour permettre aux utilisateurs de tester le modèle.

# 2. Structure du Projet

Le projet est organisé en plusieurs dossiers et fichiers pour faciliter l'exécution, la maintenance et le développement. La structure est décrite ci-dessous :

## Structure du projet

1. Tweet\_Suspect\_Detection/

2. │

3. ├── data/

4. │ ├── tweets\_suspect.csv # Données brutes collectées (tweets non traités)

6. │

7. ├── notebooks/

8. │ ├── 1\_data\_exploration.ipynb # Analyse exploratoire des données

9. │ ├── 2\_data\_preprocessing.ipynb # Nettoyage et prétraitement des données

10. │ ├── 3\_model\_training.ipynb # Entraînement des modèles

11. │ └── 4\_model\_evaluation.ipynb # Évaluation et validation des modèles

12. │

13. ├── src/

14. │ ├── \_\_init\_\_.py # Initialisation du package

15. │ ├── data\_preprocessing.py # Script de prétraitement des données

16. │ ├── feature\_engineering.py # Script pour la vectorisation et extraction

17. │ ├── model\_training.py # Script pour l'entraînement des modèles

18. │ ├── model\_evaluation.py # Script pour l'évaluation des modèles

19. │ └── utils.py # Fonctions utilitaires

20. │

21. ├── requirements.txt # Liste des dépendances Python du projet

22. ├── app.py # Application Streamlit pour l'interface utilisateur

23. ├── deploy\_model.py # Script de déploiement du modèle

24. ├── README.md # Documentation générale du projet

25. └── config.yaml # Fichier de configuration (hyperparamètres, etc.)

26.

## Détails des fichiers principaux

1. **data/** : Contient les données brutes (raw\_data.csv)
2. **notebooks/** : Inclut les notebooks pour chaque étape du projet, de l’exploration des données à l’évaluation des modèles.
3. **src/** :
   * **data\_preprocessing.py** : Gère le nettoyage des tweets (suppression des URL, ponctuations, etc.).
   * **feature\_engineering.py** : Effectue la vectorisation des textes (TF-IDF, Word Embeddings).
   * **model\_training.py** : Contient le code pour entraîner les modèles (Logistic Regression, Random Forest, etc.).
   * **model\_evaluation.py** : Évalue les modèles à l'aide de métriques de performance.
   * **utils.py** : Fonctions auxiliaires pour le chargement de données et le nettoyage.
4. **app.py** : Interface utilisateur développée avec Streamlit.
5. **deploy\_model.py** : Gère le déploiement du modèle en production.
6. **config.yaml** : Stocke les hyperparamètres et autres configurations du projet.

# 3. Description des Étapes de Développement

## 3.1 Analyse Exploratoire des Données (1\_data\_exploration.ipynb)

* **Objectif** : Comprendre la structure et le contenu des données brutes.
* **Actions réalisées** :
  + Chargement des données depuis data/raw\_data.csv.
  + Analyse de la distribution des mots, fréquences des termes, et visualisation des mots les plus fréquents pour comprendre les schémas sous-jacents.
* **Résultats** : Identification des mots communs et des structures de langage, qui ont orienté le choix des techniques de prétraitement.

## 3.2 Prétraitement des Données (2\_data\_preprocessing.ipynb et data\_preprocessing.py)

* **Objectif** : Nettoyer les données pour les rendre exploitables par les modèles.
* **Étapes** :
  + **Suppression des URLs, mentions, ponctuation** : Nettoyage du texte pour enlever les éléments non pertinents.
  + **Tokenisation et suppression des stop words** : Découpage des phrases en mots pour analyse et suppression des mots les moins informatifs.
* **Sortie** : Enregistrement des tweets nettoyés dans processed\_data.csv.

## 3.3 Extraction des Caractéristiques (feature\_engineering.py)

* **Objectif** : Convertir le texte en vecteurs numériques pour l'analyse.
* **Techniques** :
  + **TF-IDF (Term Frequency-Inverse Document Frequency)** : Utilisé pour capturer l'importance de mots en fonction de leur fréquence.
  + **Word Embeddings** : Représentations vectorielles des mots pour capturer les relations sémantiques entre termes.
* **Impact** : Améliore la capacité des modèles à distinguer les tweets suspects des autres.

## 3.4 Entraînement des Modèles (3\_model\_training.ipynb et model\_training.py)

* **Objectif** : Construire un modèle de classification efficace pour détecter les tweets suspects.
* **Modèles testés** :
  + **Logistic Regression** : Modèle de référence pour la classification binaire.
  + **Random Forest** : Apprentissage basé sur les arbres, utilisé pour capturer les non-linéarités.
  + **SVM (Support Vector Machine)** : Pour maximiser la marge de séparation entre classes.
* **Entraînement** : Chaque modèle est entraîné sur les données transformées.
* **Optimisation** : Recherche des hyperparamètres optimaux pour chaque modèle en utilisant config.yaml.

## 3.5 Évaluation et Validation des Modèles (4\_model\_evaluation.ipynb et model\_evaluation.py)

* **Objectif** : Mesurer la performance des modèles sur un ensemble de test.
* **Métriques** :
  + **Accuracy** : Taux de précision global.
  + **Recall et F1-Score** : Mesure de la capacité à identifier les tweets suspects.
  + **Courbe ROC et AUC** : Indicateurs pour évaluer la performance du modèle dans la classification binaire.
* **Résultat** : Le modèle **[Nom du modèle sélectionné]** a montré la meilleure performance et a été choisi pour la phase de production.

## 3.6 Déploiement et Interface Utilisateur (app.py et deploy\_model.py)

* **Objectif** : Permettre aux utilisateurs d'accéder au modèle via une interface interactive.
* **Technologie utilisée** : Streamlit, un framework Python pour les applications web interactives.
* **Fonctionnalité** :
  + L’utilisateur peut entrer un tweet dans l’interface Streamlit pour voir s'il est détecté comme suspect.
* **Déploiement** : deploy\_model.py contient le script de déploiement pour rendre le modèle accessible en ligne.

## 4. Exécution du Projet

### Prérequis

Installez les dépendances listées dans requirements.txt en exécutant la commande :

1. pip install -r requirements.txt

2.

### Instructions d'Exécution

1. **Analyse et Prétraitement des Données** :
   * Lancez les notebooks dans notebooks/ pour explorer et préparer les données, dans cet ordre :
     + 1\_data\_exploration.ipynb
     + 2\_data\_preprocessing.ipynb
2. **Entraînement et Évaluation des Modèles** :
   * Utilisez 3\_model\_training.ipynb pour entraîner les modèles et 4\_model\_evaluation.ipynb pour évaluer leurs performances.
3. **Interface Utilisateur** :
   * Lancer l'application Streamlit avec la commande suivante :

1. streamlit run app.py

2.

* + Accédez à l'interface pour tester le modèle avec de nouveaux tweets.

## 5. Conclusion et Perspectives

Le projet **Tweet Suspect Detection** constitue une solution efficace pour l'identification de tweets suspects, exploitant des techniques avancées de traitement du langage naturel. Le modèle final peut être déployé pour une utilisation en production, et l'interface utilisateur permet un usage intuitif et rapide.

## Améliorations Futures

* **Support Multilingue** : Ajout de données dans d'autres langues pour augmenter la robustesse.
* **Utilisation de Deep Learning** : Essai de modèles avancés comme LSTM ou BERT pour améliorer les performances.
* **Analyse en Temps Réel** : Mise en place d'un système pour analyser les tweets en temps réel.

**Signature**  
Benewende Pierre Bonkoungou