5_TP

TP Complet – Classification de Tweets : Pipeline Data Science & Tests Unitaires

Objectif général

Vous êtes en charge de construire une solution de classification automatique de tweets (catastrophes vs tweets normaux). Votre pipeline devra respecter **les bonnes pratiques de Data Science industrielle**, incluant **le nettoyage, la modélisation, les tests unitaires**, et **la dockerisation complète du projet**.

Jeu de données

Le fichier à utiliser est un dataset de tweets

Partie 1 – Exploratory Data Analysis (EDA)

Objectifs

- Comprendre la forme et le contenu du dataset
- Identifier les anomalies, les valeurs aberrantes ou manquantes
- Analyser la distribution des textes et des classes

Questions à traiter

- Quelle est la forme du DataFrame ?

- Y a-t-il des valeurs manquantes ? Des doublons ?
- Quelles colonnes vont nous être utiles ?
- Existe-t-il des incohérences ou des outliers ?
- Y a-t-il des tweets anormalement courts ou longs ?
- Quelle est la répartition de la variable cible ?
- En regardant des tweets aléatoires, la cible semble-t-elle prévisible ?

Tests unitaires à inclure

- Vérification de la présence et des types des colonnes (`text` , `target`)
- Détection automatique de valeurs manquantes ou doublons
- Vérification que tous les textes sont non-vides
- Validation du nombre de classes possibles
- Test sur les longueurs de texte (moyenne, min, max)

Partie 2 – Traitement de texte (Text Preprocessing)

Objectifs

- Nettoyer et transformer les tweets pour la modélisation
- Réduire la dimensionnalité du corpus
- Extraire une version épurée et utile du texte

Étapes à suivre

- Écrire une fonction pour :
  - Tokeniser un tweet
  - Supprimer la ponctuation, les chiffres, les mots courts (< 3 lettres)

    - Supprimer les stopwords
    - Appliquer stemming ou lemmatisation
- Reconstituer un corpus nettoyé
- Analyser :
    - Le nombre total de tokens
    - Le nombre de tokens uniques
    - Le nombre de tokens apparaissant une seule fois
- Visualiser les tokens les plus fréquents (WordCloud)

Tests unitaires à inclure

- Fonction `clean_text` :
    - Nettoie correctement texte vide, ponctuation, chiffres, mots courts
- Vérification que tous les tokens ont plus de 2 lettres
- Vérification que les stopwords sont supprimés
- Test que le vocabulaire diminue bien après nettoyage
- Vérification de l'impact du stemming/lemmatisation

Partie 3 – Modélisation et Évaluation

Objectifs

- Construire un pipeline complet : vectorisation + modèle
- Évaluer les performances du modèle

Étapes à suivre

- Utiliser un `TfidfVectorizer` ou `CountVectorizer` et Word2vec
- Choisir un modèle de classification (Logistic Regression, SVM, etc.)
- Mettre en place un pipeline avec `scikit-learn`

- Utiliser `train_test_split` ou `cross_val_score`
- Évaluer avec : accuracy, precision, recall, f1-score

Tests unitaires à inclure

- Vérifier que le pipeline s'entraîne sans erreur sur un jeu d'exemple
- Vérifier la forme des prédictions
- Vérifier que les métriques sont retournées correctement
- Comportement attendu sur texte vide, ou très court

Partie 4 – Organisation du code et des tests

Objectifs

- Avoir une structure modulaire et testable
- Séparer code métier, scripts, notebooks et tests

Structure de tp attendue

```
tweet_project/
├── data/
│   └── tweets.csv
├── notebooks/
│   ├── 01_EDA.ipynb
│   └── 02_Preprocessing_Modeling.ipynb
├── src/
│   ├── preprocessing.py     # fonctions de nettoyage et NLP
│   └── modeling.py          # pipeline sklearn
├── tests/
│   ├── test_preprocessing.py
```

```
│   └── test_modeling.py
├── main.py                    # script d'exécution
├── requirements.txt
├── Dockerfile
└── README.md
```

Exigences

- Utilisation de `pytest`
- Chaque fonction de `src/` doit avoir ses tests associés
- Les tests doivent couvrir :
  - Nettoyage
  - Tokenisation
  - Pipeline de modélisation
- Utilisez des jeux de données minimaux pour vos tests

Partie 5 – Dockerisation du projet

Objectifs

- Fournir un environnement exécutable identique pour tous
- Exécuter les tests et le pipeline via des conteneurs

À faire

- Écrire un `Dockerfile`
- Copier le projet dans l'image
- Installer les dépendances via `requirements.txt`
- Autoriser deux types de commandes :
  - Exécution des tests (`pytest`)

- Exécution du script principal (`main.py`)

README

Le fichier `README.md` doit expliquer :

- Comment installer et lancer le tp
- Les commandes Docker principales