Projet de Classification sur Données Textuelles (Logs)

1. Présentation du projet

Ce projet consiste à **entraîner un modèle de classification supervisée** sur des **données de type logs textuels**, afin d'identifier le **type d'erreur d'authentification** (ex. : *Password Mismatch*, *Password breakdown*, *Password difference*, etc.).

Le pipeline inclut :

- Le **traitement de texte** (NLP),
- La **vectorisation** des données textuelles,
- L'**entraînement d'un modèle de classification** autre qu'un arbre de décision,
- La **simulation du sur-apprentissage**,
- La **correction par deux méthodes**,
- Et enfin, le **déploiement web** via un outil comme Streamlit Ou Django

2. Choix de l'algorithme de classification

Nous avons choisi **la Régression Logistique (Logistic Regression)** comme modèle de classification. C'est un algorithme linéaire simple mais puissant, adapté aux tâches de classification binaire et multiclasse.

🔹 Principe de fonctionnement

La régression logistique est un modèle de classification qui permet de prédire la probabilité qu'un événement appartienne à une certaine catégorie (par exemple oui/non, malade/pas malade) en se basant sur des variables explicatives. Elle fonctionne en combinant ces variables pour produire un score, puis transforme ce score en une probabilité comprise entre 0 et 1. Si cette probabilité dépasse un certain seuil (souvent 0,5), le modèle prédit que l'événement va se produire ; sinon, il prédit qu'il ne se produira pas. C'est donc un outil qui permet de prendre des décisions binaires à partir de données numériques.

🔹 Avantages

- Facile à interpréter
- Rapide à entraîner
- Efficace sur les données textuelles après vectorisation

🔹 Inconvénients

- Moins performant sur des données très non linéaires
- Peut sur-apprendre si mal régularisé

3. Sur-apprentissage (Overfitting)

🔸 Définition

Le sur-apprentissage correspond à un modèle qui **apprend trop bien les détails** et le bruit des données d'entraînement, ce qui **nuit à sa capacité de généralisation** sur de nouvelles données.

🔸 Symptômes

- Très bonne performance sur les données d'entraînement
- Mauvaise performance sur les données de test

🔸 Simulation du sur-apprentissage

Dans ce projet, on peut **provoquer le sur-apprentissage** en :

- Utilisant une régularisation **trop faible** (valeur de C élevée dans Logistic Regression)
- **Trop entraîner** le modèle sans early stopping
- Vectoriser le texte avec **des n-grams ou une très haute dimension**

4. Méthodes pour corriger le sur-apprentissage

✅ Méthode 1 : Régularisation

La régression logistique inclut un paramètre **C** (inverse de la force de régularisation). Pour éviter le sur-apprentissage :

- Réduire **C** (ex. : `C = 0.01`)
- Cela ajoute une pénalisation sur les poids trop grands
- Encourage le modèle à ne pas s'adapter au bruit

✅ Méthode 2 : Validation croisée (Cross-validation)

La **validation croisée k-fold** consiste à :

- Diviser les données en **k sous-groupes**
- Entraîner le modèle **k fois**, en changeant le groupe utilisé pour l'évaluation à chaque fois
- Cela permet de :
    - Détecter les cas de sur-apprentissage
    - Mieux estimer la performance réelle
    - Choisir les bons hyperparamètres (grid search)

5. Traitement des données textuelles

Les fichiers logs sont du **texte non structuré**, donc il faut les transformer en vecteurs pour les utiliser dans un modèle ML. Voici le pipeline :

📌 Étapes de traitement NLP :

1. **Extraction des logs utiles** : utilisateur, message, type d'erreur, etc.
2. **Nettoyage des chaînes de texte** : suppression des caractères spéciaux, normalisation
3. **Tokenisation** : découpage des textes en mots
4. **Suppression des stopwords**

5. **Lemmatisation** : réduction des mots à leur forme de base
6. **Vectorisation** :
    - **Classique** :
        - Bag of Words (BoW)
        - TF-IDF
        - One-hot encoding
    - **Probabilistique** :
        - N-grams
    - **Moderne** :
        - Word2Vec
        - FastText
        - GloVe

6. Objectifs du projet

Étape	Description
1.	Entraîner un modèle performant mais sur-apprenant
2.	Corriger ce sur-apprentissage par régularisation
3.	Appliquer une validation croisée pour fiabiliser le modèle
4.	Comparer les performances obtenues
5.	Déployer le modèle avec une interface web (Streamlight)

7. Métriques d’évaluation

Pour évaluer les performances du modèle, on utilisera :

- **Accuracy**
- **Precision**

- **Recall**
- **F1-score**
- **Courbes de validation / learning curves (facultatif)**