# Raffinement du Modèle et Soumission de Test

Titre : Détection automatique du cyberharcèlement sur les réseaux sociaux à l’aide de l’intelligence artificielle

Membres de l’équipe :

1. Liban Souleiman Abdillahi

2. Nasteho Mahamoud Ibrahim

## 1. Aperçu

La phase de raffinement du modèle est cruciale pour améliorer les performances d’un système d’apprentissage automatique. Elle permet d’optimiser les paramètres, de tester différentes architectures et d’assurer une meilleure généralisation du modèle sur des données inconnues.

## 2. Évaluation du Modèle

Lors de l’évaluation initiale, le modèle BERT a obtenu une précision de 94 %, mais présentait des déséquilibres dans la classification des messages ambigus. La matrice de confusion a révélé une sensibilité plus faible pour les cas de harcèlement implicite. Ces résultats ont motivé des ajustements pour améliorer la détection fine.

## 3. Techniques de Raffinement

Nous avons utilisé plusieurs techniques de raffinement, notamment l’ajustement des hyperparamètres, l’essai de modèles alternatifs comme RoBERTa et l’intégration de méthodes d’ensemble. Ces approches ont permis d’augmenter la robustesse du modèle face à des données variées.

## 4. Réglage des Hyperparamètres

Le taux d’apprentissage, la taille du batch et le nombre d’époques ont été ajustés. Par exemple, un taux d’apprentissage de 2e-5 et un batch size de 32 ont donné de meilleurs résultats. Ces réglages ont permis de réduire le surapprentissage et d’améliorer la précision sur les données de validation.

## 5. Validation Croisée

Nous avons utilisé une validation croisée en k-fold (k=5) pour évaluer la stabilité du modèle. Cette méthode a permis de détecter les variations de performance selon les sous-ensembles de données, et d’assurer une meilleure généralisation.

## 6. Sélection des Caractéristiques

Bien que BERT utilise des embeddings contextuels, nous avons testé l’ajout de caractéristiques linguistiques comme la longueur du message ou la présence de mots offensants. Ces ajouts ont légèrement amélioré la précision du modèle dans les cas ambigus.

# Soumission de Test

## 7. Aperçu

La phase de soumission de test consiste à appliquer le modèle final sur un jeu de données inconnu pour évaluer ses performances réelles. Cette étape est essentielle avant tout déploiement.

## 8. Préparation des Données pour les Tests

Le jeu de données de test a été nettoyé et prétraité de manière identique aux données d’entraînement : suppression des caractères spéciaux, tokenisation, et encodage via le tokenizer BERT.

## 9. Application du Modèle

Extrait de code :

# Application du modèle BERT sur les données de test  
from transformers import BertTokenizer, BertForSequenceClassification  
import torch  
  
tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')  
model = BertForSequenceClassification.from\_pretrained('bert-base-uncased')  
inputs = tokenizer(test\_texts, return\_tensors='pt', padding=True, truncation=True)  
outputs = model(\*\*inputs)  
predictions = torch.argmax(outputs.logits, dim=1)

## 10. Métriques de Test

Les métriques utilisées incluent la précision, le rappel, le score F1 et la matrice de confusion. Le modèle a obtenu une précision de 95 %, un rappel de 92 % et un score F1 de 93 %, ce qui montre une amélioration par rapport à la phase initiale.

## 11. Déploiement du Modèle

Le modèle est prêt à être déployé via une API REST utilisant Flask. Il peut être intégré dans une plateforme de modération de contenu pour détecter automatiquement les messages harcelants.

## 12. Implémentation du Code

Extrait de code pour le déploiement :

# Déploiement du modèle avec Flask  
from flask import Flask, request, jsonify  
app = Flask(\_\_name\_\_)  
  
@app.route('/predict', methods=['POST'])  
def predict():  
 data = request.json['text']  
 inputs = tokenizer(data, return\_tensors='pt', padding=True, truncation=True)  
 outputs = model(\*\*inputs)  
 prediction = torch.argmax(outputs.logits, dim=1).item()  
 return jsonify({'prediction': prediction})  
  
app.run()

# Conclusion

La phase de raffinement du modèle a permis d’améliorer significativement les performances du système de détection du cyberharcèlement. Grâce à l’ajustement des hyperparamètres, à l’intégration de nouvelles caractéristiques et à une validation rigoureuse, le modèle est désormais plus robuste et prêt pour une utilisation réelle. La phase de test a confirmé sa capacité à généraliser sur des données inconnues, avec des métriques élevées et une stabilité appréciable.

# Références

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Khafajeh, H. (2024). Cyberbullying Detection in Social Networks Using Deep Learning. IAJIT.

Aggarwal, A., & Mahajan, R. (2024). Shielding Social Media: BERT and SVM Unite. Journal ISI.

Hasan, M. et al. (2023). A Review on Deep-Learning-Based Cyberbullying Detection. MDPI.

Kumar, R. et al. (2024). Bias and Cyberbullying Detection Using Transformers. MDPI.

Rahman-Laskar, A. et al. (2024). Cyberbullying Detection in a Multi-classification Codemixed Dataset. Computación y Sistemas.

Gomez, R. et al. (2022). Curating Cyberbullying Datasets: a Human-AI Collaborative Approach. Springer.

Altayeva, A. et al. (2024). Hybrid Deep Learning Architecture for Cyberbullying Detection. IJACSA.

Philipo, E. et al. (2024). Cyberbullying Detection: Exploring Datasets, Technologies, and Approaches. arXiv.

Aravindhan, R. (2025). Deep Learning Algorithms for Robust Cyberbullying Detection. IJRPR.

Saini, A. et al. (2023). Cyberbullying Detection on Social Networking Sites Using BERT. AIP.

Balakrishnan, V., & Kaity, M. (2023). Cyberbullying detection and machine learning: A systematic literature review. Springer.