université virtuelle Burkina Faso

Master Fouilles de Données et Intelligence Artificielle

Module: **Construction de modèles et leur Déploiement**

Sujet : Modèle de Détection de Tweet Suspect

ETUDIANTS: Ali BARRO, Boubacar KOANDA

ENSEIGNANT: Dr Abdoul Kader KABORE

Table des matières

I. Introduction

Dans le cadre du Master en Fouilles de Données et Intelligence Artificielle à l'Université Virtuelle du BURKINA FASO, un cours sur la construction des modèles de Machine Learning et leur déploiement. Ce cours a été dispense par Dr Abdoul Kader KABORE. A l'issu du cours théorique et quelques travaux dirigés, un exercice pratique a été proposé par l'enseignant afin de mettre en pratiques les différentes techniques vues dans le cours. En effet, dans le cas de ce travail, il est question de proposer un modèle de Machine Learning permettant de classifier si un discours est suspect (menaces, terrorismes, intimidation...) ou non. Pour ce faire, nous allons utiliser un modèle linéaire basé sur la régression logique, ce qui simplifie l'entraînement et permet de bénéficier de sa grande performance pour les tâches de classification de texte.

II. Les étapes de développement du modèle de Machine Learning

Le développement d'un modèle d'IA passe par plusieurs étapes qui partent de la collecte de données, leur nettoyage en passant par la conception du modèle proprement dit jusqu'à son déploiement pour utilisation. Parmi ces étapes clés nous avons :

- **Le chargement et l'exploration des données** : qui a consisté dans notre cas à charger le dataset (tweets_suspects.csv) à l'aide des librairies pythons appropriées comme ***pandas***. Ensuite s'en est suivi une exploration des données afin de comprendre leur structure et leur contenu, en utilisant la même librairie. Enfin, nous avons vérifié les valeurs manquantes.
- **Prétraitement des données** : a ce niveau, nous avons nettoyer ces données qui nous a permis de s'assurer leur qualité. C'est à cette étape que nous avons supprimer les urls, les mentions et les hashtags. D'autres traitements ont été effectués comme le retrait des caractères spéciaux, la ponctuation qui n'apportent pas d'information utile pour notre tâche de classification. A ceux-là nous ajoutons la division du texte des tweets en mots ce qui permet de traiter chaque mot individuellement lors de l'analyse. Les mots ont été mis en minuscule pour uniformiser le texte et éviter les doublons dus à la case. Nous retirons éventuellement des stopwords (mot vides) qui n'ont pas de valeur significative pour notre analyse. Nous réduisons les mots à leur forme de base grâce à la lemmatisation, ce qui permet d'harmoniser les différentes variations d'un même mot. Nous utilisons nltk pour la tokenisation et la lemmatisation, tout en retirant les stopwords. (**Tokenisation et normalisation**).
- **Embedding des données :** Nous avons utilisé TfidfVectorizer pour transformer le texte en vecteurs numériques. Nous choisissons une méthode d'embedding, comme TF-IDF pour transformer le texte des tweets en vecteurs numériques pour qu'ils soient utilisables par notre modèle de classification.

TF-IDF qui pondère les mots en fonction de leur fréquence dans le corpus et dans chaque tweet. Nous appliquons la méthode d'embedding choisie sur l'ensemble des tweets pour obtenir une représentation vectorielle de chaque tweet.

- **Construction du modèle de classification :** Nous divisons les données en ensembles d'entraînement et de test, puis entraînons un modèle de régression logistique. Nous sélectionnons un algorithme de classification, tel que la **régression logistique.** Nous divisons les données en ensembles d'entraînement et de test pour évaluer la performance de notre modèle. Nous entraînons le modèle sur l'ensemble d'entraînement, en ajustant les paramètres pour optimiser la classification. Nous utilisons l'ensemble de test pour évaluer les performances du modèle, en calculant des métriques telles que la précision, le rappel et le F1-score. Nous calculons des métriques telles que le rapport de classification et affichons la matrice de confusion et la courbe ROC.
- **Rapport et visualisation :** Les résultats sont présentés sous forme de graphiques. Nous créons des visualisations pour illustrer les résultats, comme des courbes ROC, des matrices de confusion et des histogrammes des classes. Une matrice de confusion est également tracée pour visualiser les erreurs de classification.
- **Création d'une interface interactive pour tester le modèle avec de nouveaux tweets**

  L'utilisateur pourra saisir de nouveaux tweets via une page web streamlit, et le modèle pré-entraîné les classera comme "Suspect" ou "Non suspect":

  - **Chargement du modèle et du TF-IDF** : Le modèle préalablement entraîné (avec Logistic Regression dans l'exemple précédent) et le

vectoriseur TF-IDF sont chargés depuis les fichiers ***logistic_regression_model.pkl*** et ***tfidf_vectorizer.pkl***.

- **Prétraitement du tweet saisi**: Chaque tweet saisi par l'utilisateur est nettoyé et lemmatisé (comme cela a été fait dans la phase d'entraînement).
- **Transformation en vecteur**: Le texte du tweet est ensuite converti en vecteur numérique via le vectoriseur TF-IDF.
- **Prédiction**: Le modèle Logistic Regression fait une prédiction et affiche si le tweet est **"Suspect"** ou **"Non suspect"**.
- **Interactivité**: L'utilisateur peut saisir plusieurs tweets jusqu'à ce qu'il tape "quit" pour quitter le programme.

III. Les résultats obtenus

A l'issu des différentes étapes énumérées, nous avons un certain nombre de résultats. L'exploration des données nous permis d'identifier deux classes dans notre jeu de données à savoir la **classe1** qui caractérise les tweets suspects (menaces, terrorisme, intimidation, …) et la **classe 0**, qui caractérise les tweets non suspects

Parmi ces résultats, nous notons un déséquilibre des classes. Nous avons environ 90% des tweets suspect et 10% pour les tweets non suspects (figure 1 ci-dessous).

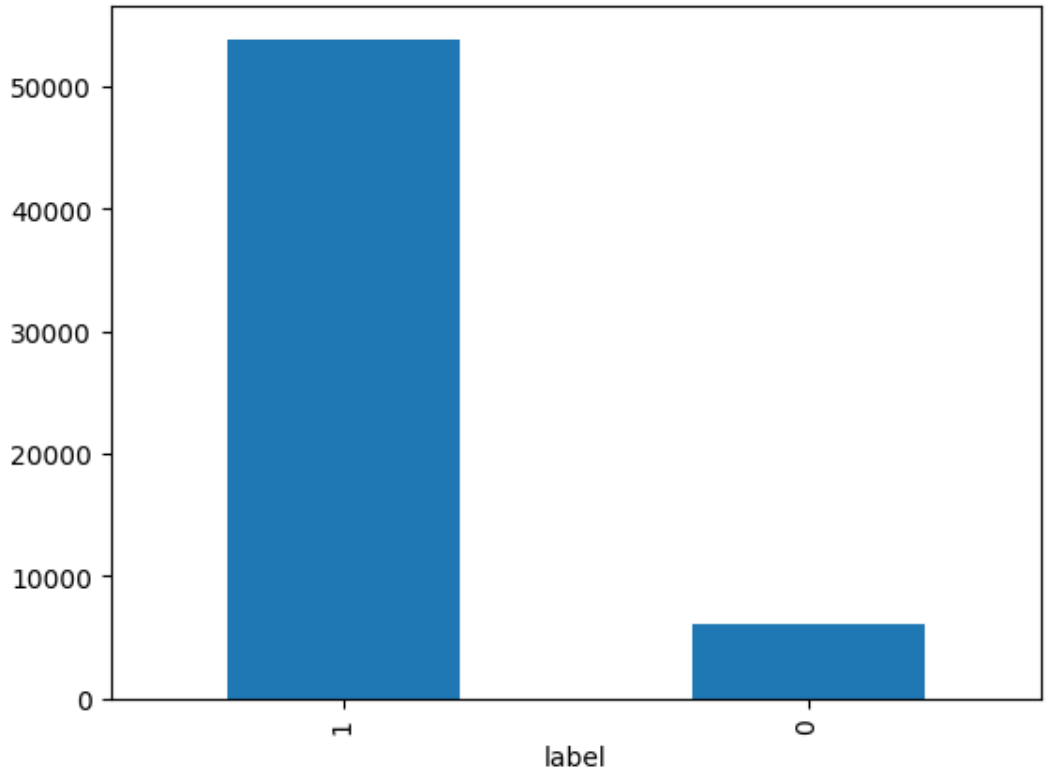


Figure 1: La distribution des différentes classes de notre dataset.

Nous avons utilisé un modèle de Deep Learning dont le résumé est donné à l'image suivante (figure 2). En sus, nous avons utilisé le modèle de régression logistique de part sa popularité dans les classes de classification.

```
model.summary()
```

[11] ✓ 0.0s

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	640,128
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 1)	65

Total params: 1,945,349 (7.42 MB)

Trainable params: 648,449 (2.47 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 1,296,900 (4.95 MB)

```
Epoch 1/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9092 - loss: 0.3079 - val_accuracy: 0.9653 - val_loss: 0.1315
Epoch 2/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9740 - loss: 0.0986 - val_accuracy: 0.9668 - val_loss: 0.1362
Epoch 3/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9810 - loss: 0.0733 - val_accuracy: 0.9684 - val_loss: 0.1366
Epoch 4/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9837 - loss: 0.0577 - val_accuracy: 0.9685 - val_loss: 0.1531
Epoch 5/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9877 - loss: 0.0396 - val_accuracy: 0.9653 - val_loss: 0.1970
Epoch 6/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9940 - loss: 0.0199 - val_accuracy: 0.9653 - val_loss: 0.2406
Epoch 7/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9973 - loss: 0.0093 - val_accuracy: 0.9621 - val_loss: 0.2734
Epoch 8/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9984 - loss: 0.0062 - val_accuracy: 0.9642 - val_loss: 0.3155
Epoch 9/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9987 - loss: 0.0046 - val_accuracy: 0.9642 - val_loss: 0.3224
Epoch 10/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9989 - loss: 0.0037 - val_accuracy: 0.9624 - val_loss: 0.3112
Epoch 11/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9987 - loss: 0.0040 - val_accuracy: 0.9647 - val_loss: 0.3392
Epoch 12/100
750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.9986 - loss: 0.0045 - val_accuracy: 0.9648 - val_loss: 0.3560
Epoch 13/100
...
    accuracy                           0.96     12000
   macro avg       0.92      0.86      0.89     12000
weighted avg       0.96      0.96      0.96     12000
```

Figure 2: le résumé et accuracy du model de deep learning utilisé

Quant au modèle de régression logistique, l'accuracy se présente dans la figure ci-dessous.

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.55	0.71	1222
1	0.95	1.00	0.97	10778
accuracy			0.95	12000
macro avg	0.97	0.77	0.84	12000
weighted avg	0.95	0.95	0.95	12000

Figure 3: accuracy du modèle regression logistique

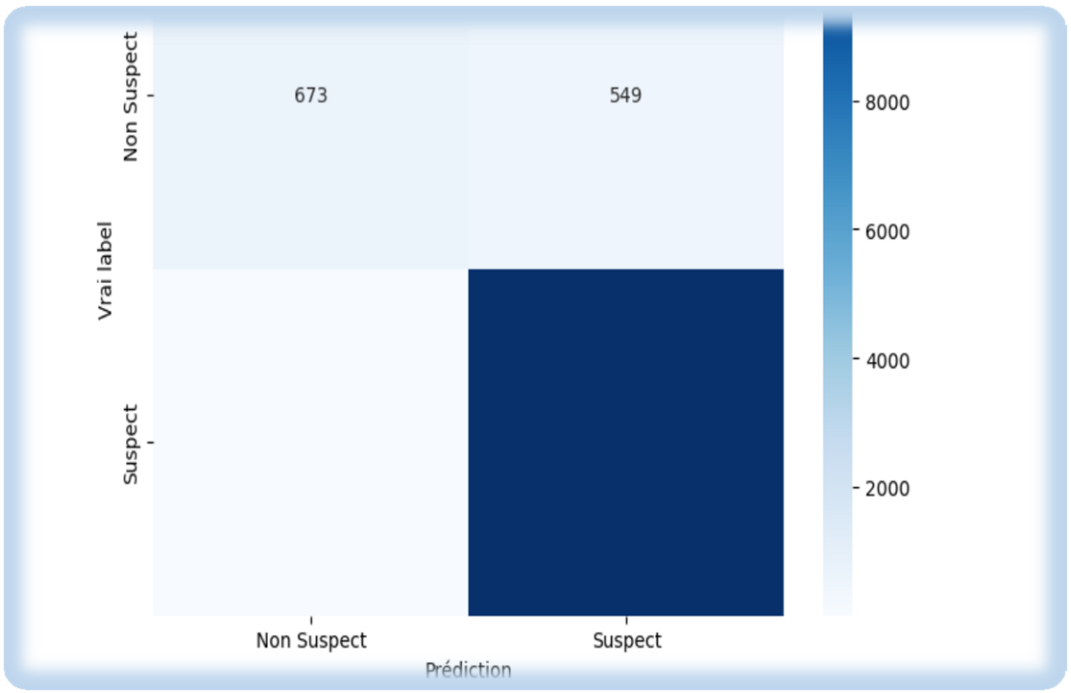
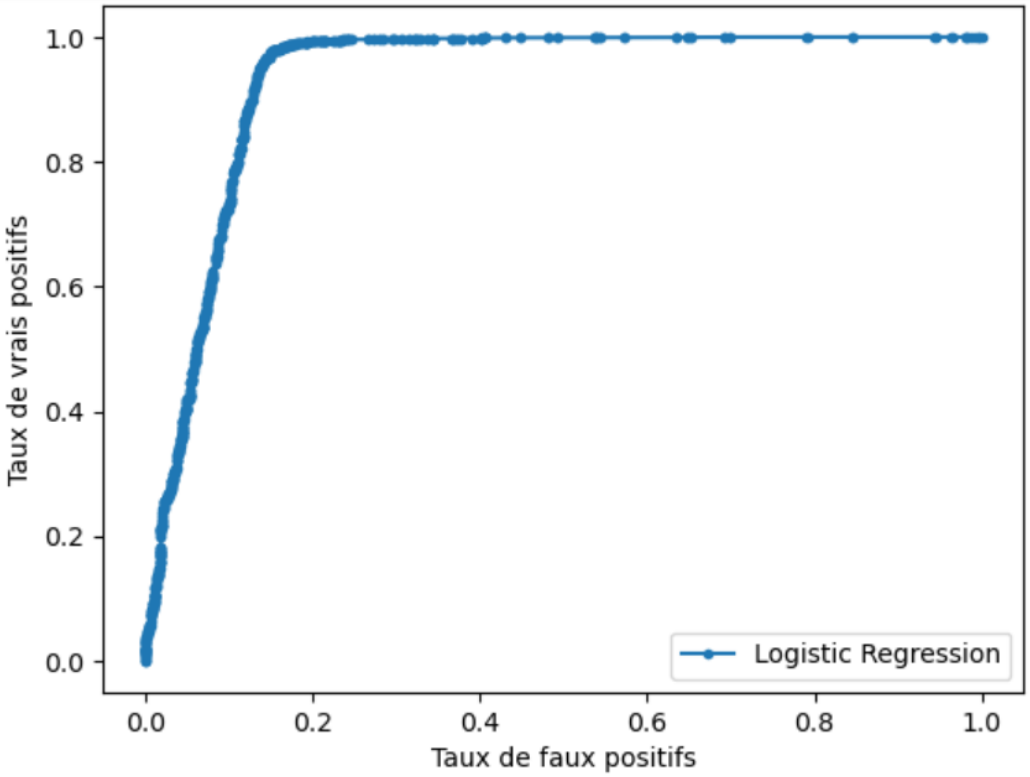


Figure 4: La matrice de confusion issue de l'évaluation de la régression logistique




```
AUC: 0.9329688682073168
```

Figure 5: La courbe ROC de la regression logistique

L'analyse de ses différentes métriques montre que le modèle présente une forte capacité de détection des cas positifs avec un rappel de 1,00 pour la classe "Suspect", ce qui est excellent si l'objectif principal est d'identifier les cas "Suspect". La précision globale élevée et l'AUC de 0,93 confirment une bonne performance générale du modèle.

Création d’une interface interactive pour tester le modèle avec de nouveaux tweets

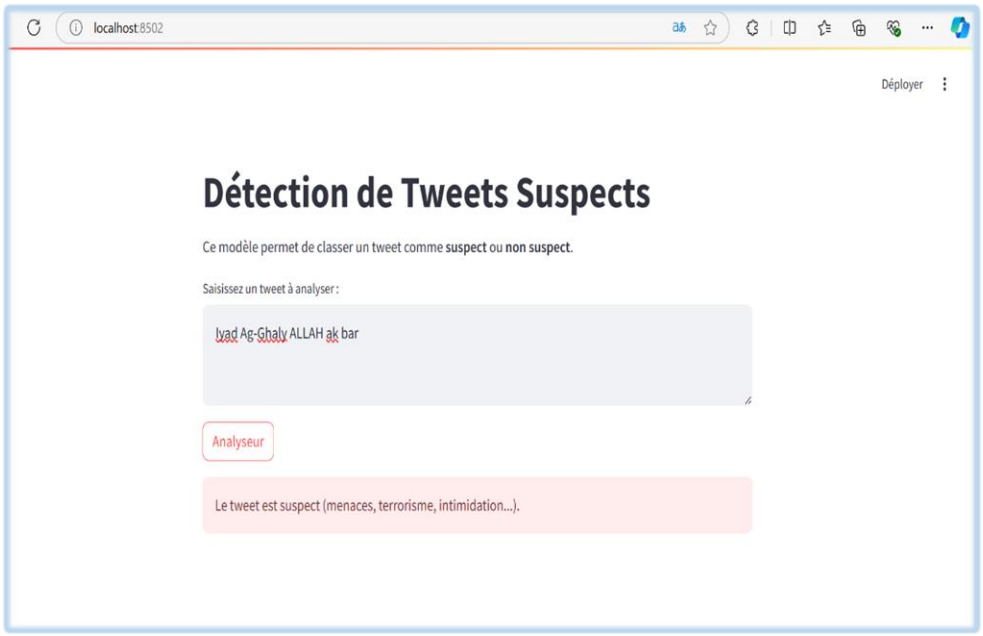


Figure 6 : Une capture d'écran streamlit du modèle de ML

IV. Les réflexions sur le projet

Dans ce projet de détection de discours suspect, nous avons affronté le défi majeur du déséquilibre des classes, avec 90 % des tweets classés comme suspects. Pour y répondre, nous avons utilisé une régression logistique et des modèles de deep learning, en intégrant des techniques de pondération des classes pour améliorer la robustesse de la classification. La régression logistique a offert une base solide et interprétable, permettant une compréhension claire des facteurs influençant les prédictions. Parallèlement, les modèles de deep learning ont capturé des relations complexes et contextuelles dans les données textuelles, augmentant ainsi la performance globale du système. Cependant, nous avons rencontré des défis techniques liés à l'intégration des embeddings et aux exigences computationnelles des modèles de deep learning, notamment des problèmes de dépendances et de gestion des ressources. Ce projet a également mis en lumière l'importance de traiter les biais et les faux positifs, soulignant les implications éthiques de l'utilisation de tels outils dans des contextes sensibles. Les résultats obtenus démontrent le potentiel des approches combinées, tout en indiquant des pistes d'amélioration future, telles que l'optimisation des modèles et l'exploration de techniques avancées de traitement du langage naturel pour renforcer encore la précision et la fiabilité des détections.

V. Conclusion

ce projet de détection de discours suspect a permis de mettre en pratique diverses techniques d'apprentissage supervisé et d'ingénierie des données, montrant l'importance des choix de modèle et des stratégies de traitement des déséquilibres de classes dans le domaine de la détection de contenu sensible. En combinant des méthodes comme la régression logistique, les modèles de deep learning, et les techniques de pondération des classes, nous avons atteint des performances significatives malgré les défis posés par les données textuelles et l'architecture des modèles. Ce projet met en lumière les potentialités mais aussi les limites des outils de NLP pour des applications réelles et critiques. Pour aller plus loin, des améliorations pourraient inclure l'optimisation de modèles de deep learning plus avancés ou l'intégration de données supplémentaires pour accroître la précision, la robustesse et l'équité des prédictions. Cette expérience nous a également sensibilisés aux enjeux éthiques et à la nécessité d'une évaluation rigoureuse pour garantir un usage responsable de ces modèles dans des contextes applicatifs.