SIMPLON.CO

Fake news detection

Alexis, Alicia et Morgan

1. Plan du rapport

Introduction

Choix du dataset

Analyse des datasets

Préparation des données pour le modèle

Modèle utilisé et résultat

Explicabilité du modèle

Base de données

Application

API

Conclusion et perspectives

I. Introduction

Les fake news sont des contenus délibérément fabriqués, trompeurs ou inexacts, conçus pour influencer les opinions publiques, semer la confusion ou servir des intérêts particuliers. Selon une étude, les fake news ont tendance à se propager plus rapidement et à toucher davantage de personnes que les informations vérifiées (1). Ces informations erronées peuvent avoir des conséquences graves sur la démocratie, la santé publique, les relations internationales et bien d'autres domaines.

Avec l'avènement des médias sociaux et la facilité de diffusion de l'information en ligne, il est devenu de plus en plus difficile de distinguer les faits de la fiction. Selon des études (2,3), environ 64% des adultes en ligne ont été exposés à des fake news, et près de 45% d'entre eux ont partagé ces informations sans les vérifier . Ces chiffres alarmants soulignent l'ampleur du problème et la nécessité de prendre des mesures pour lutter contre les fausses informations.

La lutte contre les fake news est devenue une priorité, tant pour les gouvernements que pour les individus. Il est crucial de pouvoir détecter et vérifier l'authenticité des informations avant de les partager. c'est dans le but d'aider à identifier de potentielles fake news que nous proposons cette application. Basé sur un modèle de machine learning, il permettra à tout public de tester différents types de textes ou articles.

Choix du dataset

Afin de pouvoir cerner au mieux ce qui fait la différence entre fausse ou vraie information, nous avons choisi d'utiliser deux dataset. Le premier contient 20800 lignes et 4 colonnes (titre, auteur, texte et label). Il provient de kaggle (Fake News | Kaggle). Nous avons par la suite augmenté le nombre de donnée en utilisant un deuxième dataset provenant lui aussi de kaggle (Fake News Classification | Kaggle). Il contient 72134 lignes et 3 colonnes (titre, texte et label). Les deux datasets ne comportaient pas de valeurs dupliquées. Quelques valeurs manquantes se retrouvent sur les dataset. Nous n'avons pas conservé les lignes où était présente un NaN.

Analyse des datasets

Nous avons dans un premier observé la répartition des données labélisé réel (0) ou fausse (1). Pour les deux df, on peut observer une taille équivalente entre les deux classes (Figure 1).

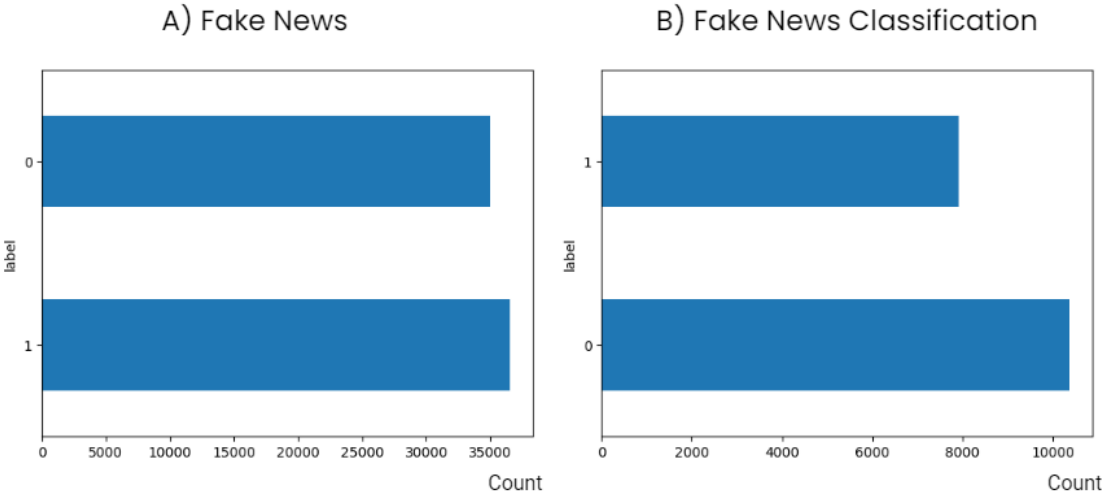


Figure 1: répartition des fake news(1) et des vrai news (0) sur les deux datasets

Pour le dataset 1, nous avons voulu savoir s'il y avait une différence entre les auteurs des différents textes. Nous avons pu remarquer que les textes labélisé réel possédait des noms/ prénoms alors que la plupart des textes fake news sont en majorité des pseudo ou non nommées (Figure 2).

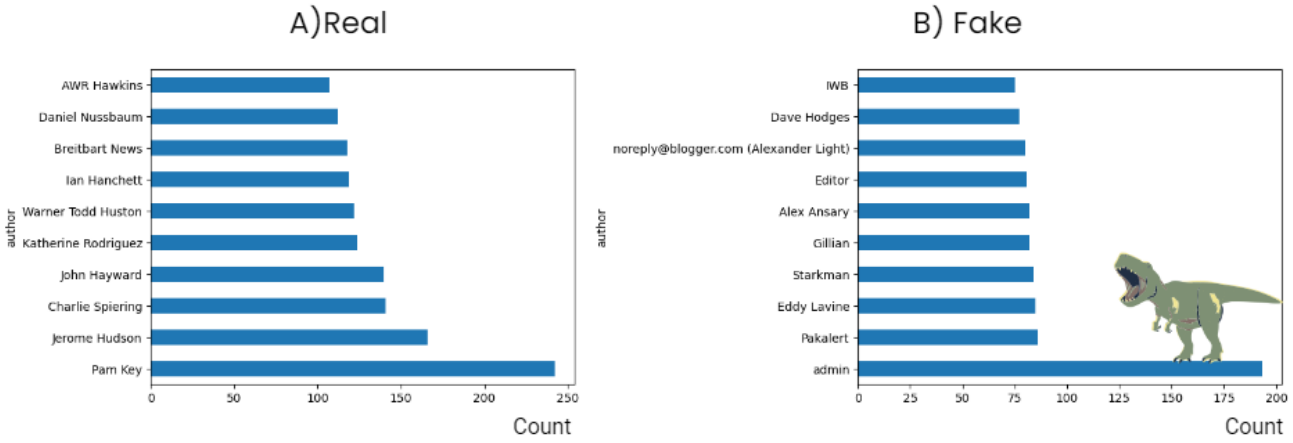


Figure 2: les auteurs les plus représentés dans le dataset pour les textes réels ou fake news.

Nous nous sommes ensuite concentré sur les mots les plus représentés dans les dataset par wordclouds notamment (Figure 3). Comme on peut le voir, il ne semble

pas y avoir une grosse différence entre les fake news et les vrais.

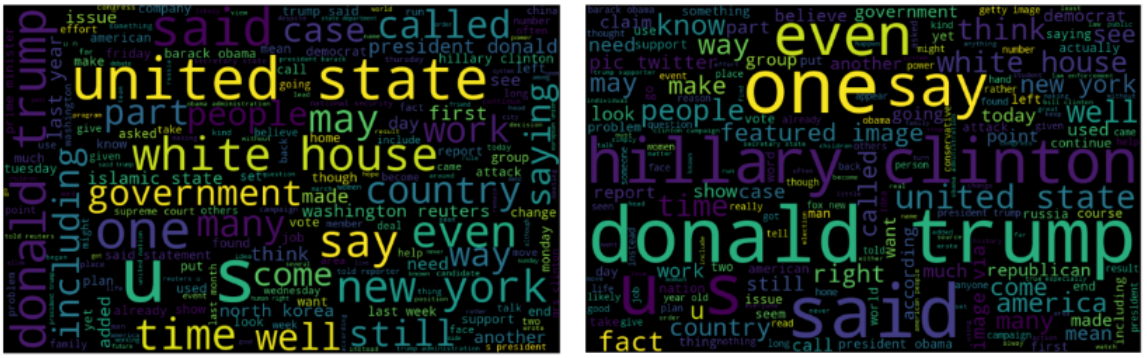


Figure 3: Représentation en wordcloud des mots les plus présent dans les fake ou real news.

Nous avons donc observé les associations de mots (2 mots pour bigrams et 3 mots pour trigrams). Un exemple des trigrams obtenu est représenté Figure 4. On observe des différences d'association entre les deux classes.

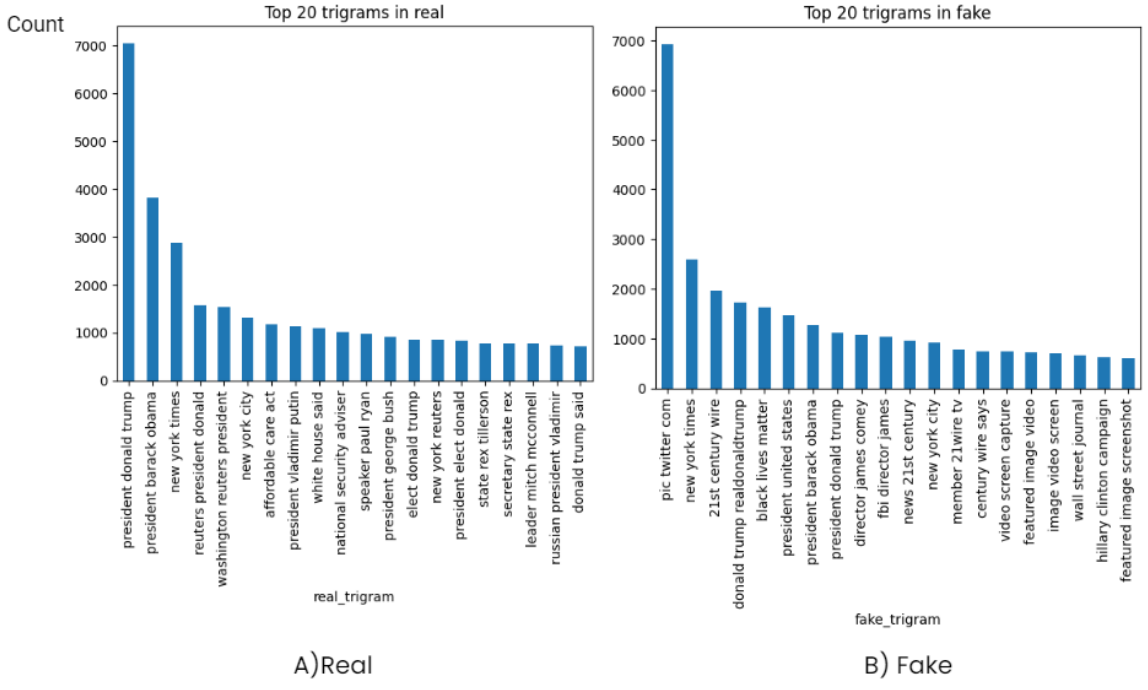


Figure 4: Différents trigrammes observer pour les deux classes.

Préparation des données pour le model

Après l'élimination des stops words, les mots sont vectorisés pour passer à des features en numérique.Une stemmatisation a été utilisé afin de réduire les mots à leur racine et donc d'enlever toute les variation d'un même mot afin que le modèle puisse les traiter plus facilement

Pour cela nous avons utilisé tfidf vectorizer de scikit learn avec en paramètres un stop word (stop_words ='english'). Une pipeline à été réalisée afin de pouvoir réaliser le même pré-processing et d'obtenir une prédiction sur un texte fourni par l'utilisateur.

La préparation a été utilisée sur le dataset mais uniquement la feature " text " a été retenue, en vue de pouvoir utiliser le modèle manuellement par l' utilisateur et pas seulement sur le datasets.

Modèle utilisé et résultats

Dans un premier temps une logistique régression à été réalisée.Afin d'entraîner le modèle, un Split du dataset à été réalisé avec les paramètres suivants: train_test_split(x,y,test_size = 0.2, stratify=y, random_state=2).

Une évaluation du modèle à été réalisée en utilisant un rapport de classification et une courbe d'apprentissage (Figure 6).

Accuracy score: 0.9401442307692308

	precision	recall	f1-score	support
Real	0.94	0.94	0.94	2077
Fake	0.94	0.94	0.94	2083
accuracy			0.94	4160
macro avg	0.94	0.94	0.94	4160
weighted avg	0.94	0.94	0.94	4160

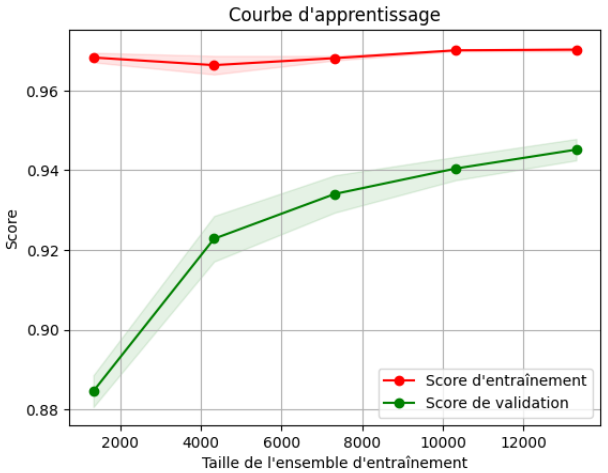


Figure 6: Rapport de classification et courbe d'apprentissage du modèle de régression logistique.

Comme on peut l'observer sur la Figure, notre modèle possède un bon score (une accuracy a plus de 90%. Une cross validation à été réalisée afin de voir si le score était fiable. Les résultats ont confirmé les très bons scores. On observe un très

bon apprentissage du modèle (courbe d'apprentissage, figure), cependant on ne peut pas savoir si le modèle va tomber en overfitting par manque de données. De plus les datasets ne comportant que des données sur le sujet de la politique Américaine, il y a fort à parier que notre modèle généralise très mal sur d'autres données.

Un XGBClassifier(n_estimators=5, max_depth=5, learning_rate=1, objective='binary:logistic') a ensuite été testé. Le rapport de classification peut être observé Figure 7.

```
Accuracy score: 0.9024038461538462
              precision    recall  f1-score   support

        Real       0.90      0.90      0.90      2077
        Fake       0.90      0.90      0.90      2083

    accuracy                           0.90      4160
   macro avg       0.90      0.90      0.90      4160
weighted avg       0.90      0.90      0.90      4160
```

Figure7: Rapport de classification du modèle XGBClassifier .

On peut observer que les scores sont presque équivalents à la régression logistique. Cependant le temps de calcul nécessaire à faire tourner le modèle ainsi que l'applicabilité du XGB par rapport au résultats obtenus, nous a poussé à utiliser pour la suite du projet la régression logistique.

Création d'une pipeline

Une pipeline a été créée afin de faciliter la prédiction par l'API. Il contient la vectorisation ainsi que le modèle (Figure 8).

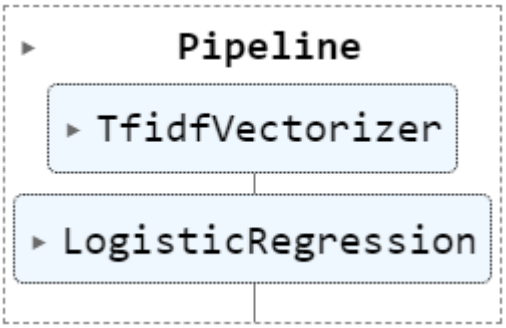


Figure 8: Schématisation de la pipeline utilisé

Le model a dans un premier temps entraîné avec le dataset Fake News. Il obtient un score d'accuracy de 94,3 % en cross validation. Afin d'observer sa véracité, des prédictions ont été réalisées avec le second dataset Fake News Classification. Un

rapport de classification et une matrice de confusion ont été obtenus (Figure 9). On observe que le modèle est assez bon, pour 24044 prédictions, il ne s'est trompé que 4456 fois.

	precision	recall	f1-score	support
0	0.89	0.77	0.83	13734
1	0.74	0.87	0.80	10310
accuracy			0.81	24044
macro avg	0.82	0.82	0.81	24044
weighted avg	0.83	0.81	0.82	24044

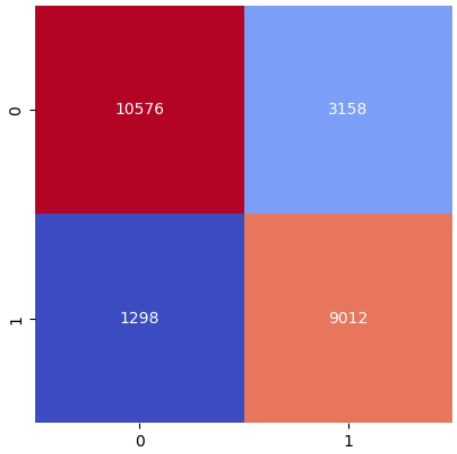


Figure 9: Rapport de classification ainsi que la matrice de confusion du model present dans la pipeline .

Nous avons bien entendu par la suite entraîner à nouveau le model avec le second dataset.

Explicabilité du modèle

Un lime à été réalisé sur le modèle afin de comprendre quel mots était utilisé par le modèle pour différencier les deux classes (Figure 10 et Figure 11).
Afin de faire le l'applicabilité, il faut paramétrer un explainer (explainer.explain_instance(text, pipeline.predict_proba, num_features=20)) puis lui demandé de l'afficher (exp.show_in_notebook(text=True)).

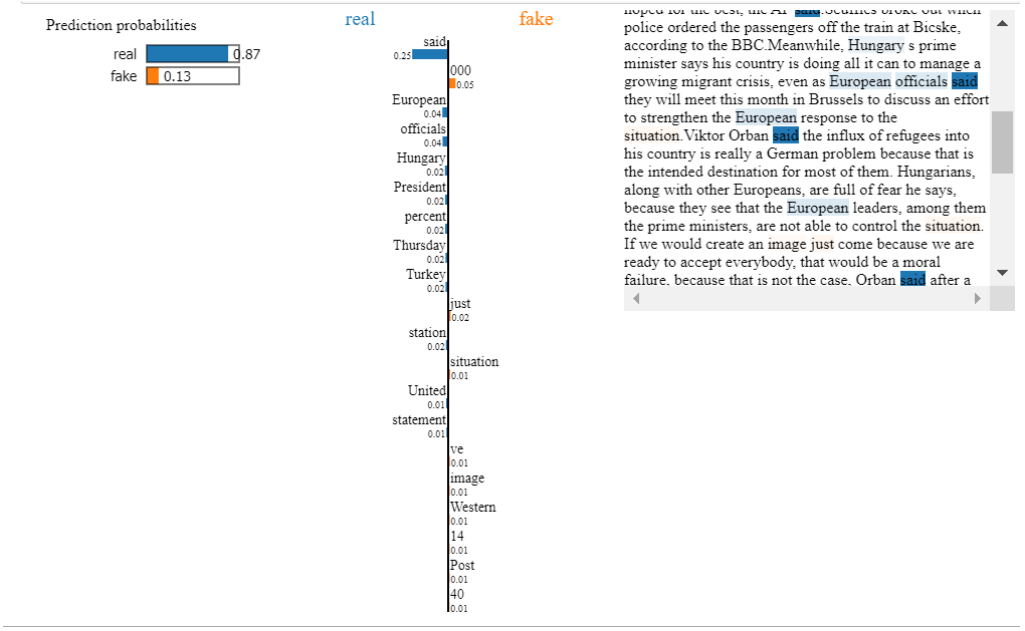


Figure 10: Résultat obtenue sur une vrai news .

Les mots que le modèle identifié comme déterminant des vrais news seront coloré en bleu et ceux qui détermine les fake news en orange

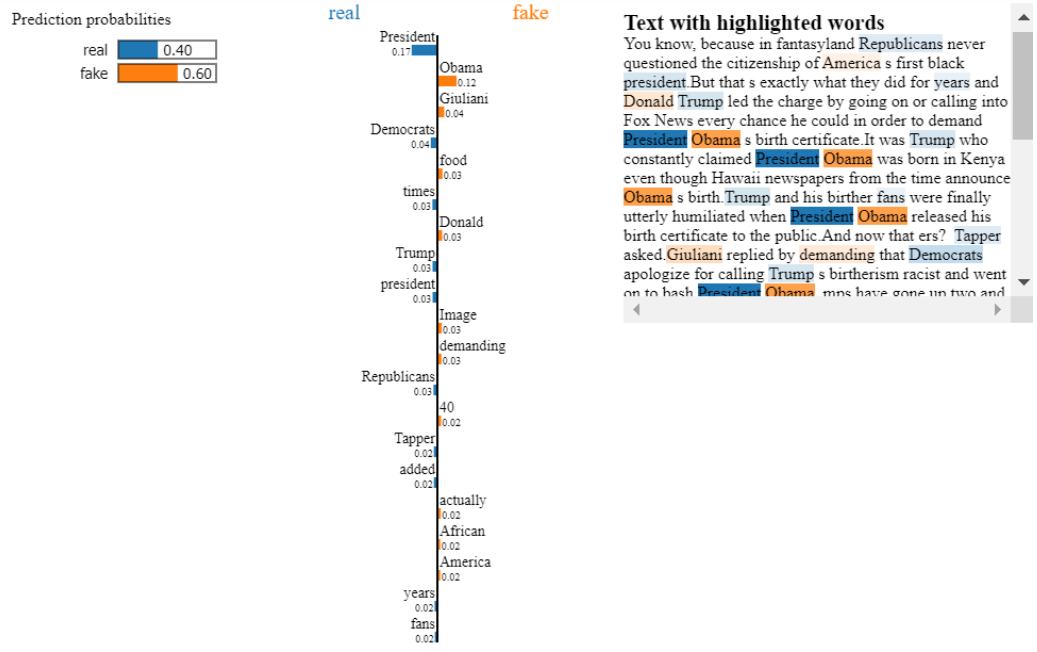


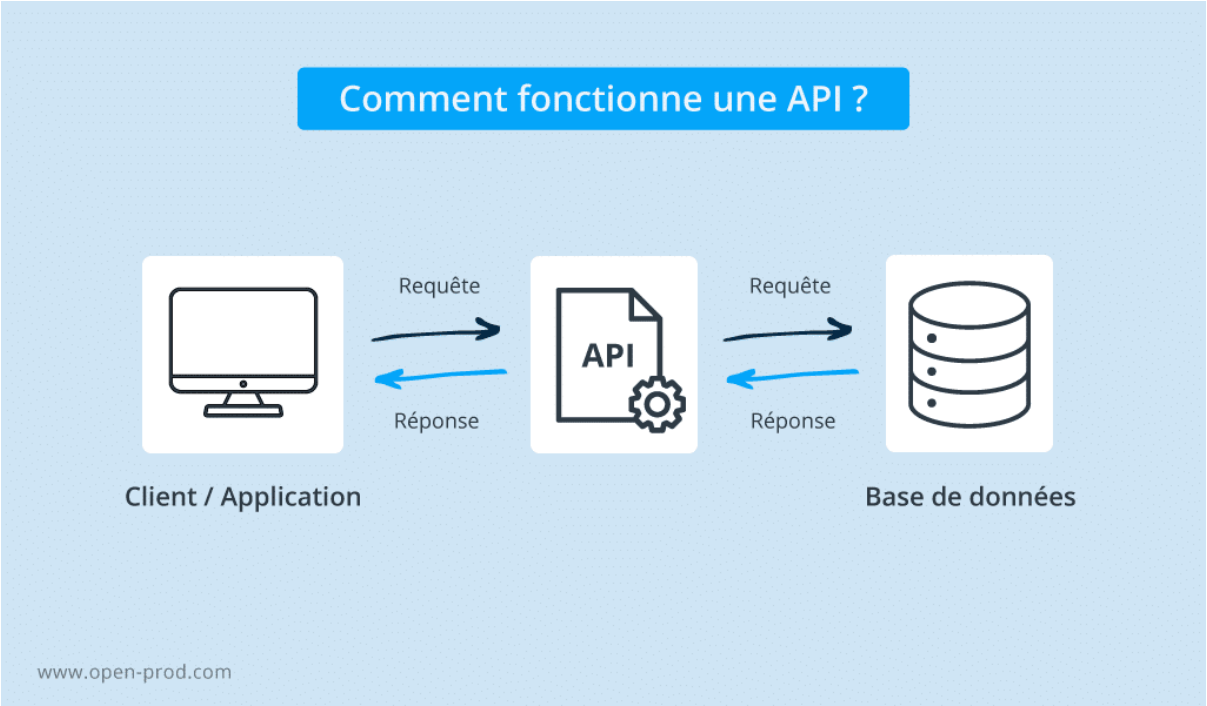
Figure 11: Résultat obtenue sur une fake news .

On peut observer que dans le texte de la première figure ci-dessus le mot “said” influence le modèle à prédire que celui-ci est vrai, alors que sur la deuxième figure le terme “Obama” influence celui-ci a prédire que le texte est faux.

Base de donnée

Nous avons dans un premier temps stocker les données brut , notre idée était de faire un système ressemblant à une solution ELT c'est à dire extraire les données des csv et de kaggle puis les stocker en base de données pour enfin les transformer et les utiliser pour nos modèle et notre exploration des données .

Pour cette base de données nous avons choisi une BD azure flexible mysql , ce qui nous permet de profiter de l'environnement azure pour déployer une web app qui sera notre api et fera le lien entre notre solution d'application et la base de donnée azure



Application

Nous avons choisi comme support d'application Streamlit , ce qui nous permet avant tout de facilement déployer une visuel pour donner un support utilisable pour notre solution machine learning

Api

Nous avons déployé une api sur azure , cette api permet de faire grâce à des requêtes get de récupérer des donnée de la base de donnée et de faire une prédiction et grâce à des requêtes post de mettre à jour notre base de donnée avec nos prédiction

Conclusion

Nous possédons une bonne base pour la production d'une application permettant la détection de fake news. Cependant quelques points importants restent à améliorer afin d'obtenir une prédiction plus fiable.

Un bot discord a été créé afin que les utilisateurs puissent vérifier des news sans passer par un site.

Perspective

Il aurait été intéressant d'utiliser les bigram ou trigrams comme features pour entraîner le model. Cela aurait pu avoir comme effet d'améliorer la différenciation entre fake news de réal news.

La pipeline pourra être améliorée avec des fonctions personnalisées afin d'obtenir des prédictions plus fidèles.

De plus, entraîner de nouveau notre modèle sur des données autres que la politique américaine serait nécessaire dans un futur proche afin d'améliorer notre généralisation.

Nous envisageons aussi de réaliser des entraînements à partir des résultats ou données proposées par les utilisateurs.

Biblio

(1) Newman, N., Fletcher, R., Kalogeropoulos, A., Levy, D. A., & Nielsen, R. K. (2020). Reuters Institute Digital News Report 2020. Reuters Institute for the Study of Journalism.
(2) Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151.
(3) Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. Journal of Economic Perspectives, 31(2), 211-236.