Présentation de la Solution

CHOUKROUN Simon

12 août 2022

Table des matières

1 Introduction

Le cyberespace représente un espace sans frontière, anonyme ; un espace de liberté et de partage, ce qui le rend difficilement contrôlable et suscite de nombreux enjeux de pouvoir. Il est ainsi devenu un véritable terrain de jeux de certains acteurs étatiques, financiers, médiatiques et politiques. Depuis maintenant plusieurs années, un phénomène explose sur le web, celui des fake news. Ce terme que l'on peut traduire, en français, par informations fausses ou fallacieuses s'est imposé pour désigner toutes les informations truquées qui circulent sur les réseaux sociaux et dans certains médias. Le 22 juillet dernier, YouTube a annoncé une série de mesures pour bannir les fake news liées aux avortements, un mois après la révocation de ce droit par la Cour suprême des Etats-Unis. Le pays est, en effet, partisan d'un internet libre où acteurs privés (GAFAM) et société civile ont un rôle à jouer aux côtés des Etats.

Au contraire, d'autres pays comme la Chine et la Russie défendent la souveraineté des Etats sur leurs réseaux. Depuis 2018, certains Etats ont tenté de mettre en place des normes internationales nécessaires à la protection des citoyens dans le cyberespace en particulier face à la multiplication des fake news dont les conséquences peuvent être extrêmement graves. Très récemment, en France, l'affirmation selon laquelle il y aurait eu deux millions d'immigrés supplémentaires sous le quinquennat d'Emmanuel Macron, la citation fausse de Bruno Le Maire demandant aux Français de conduire « un jour sur deux » pour limiter leur facture d'essence ou encore le chiffre de 50 milliards d'euros de fraude sociale chaque année illustrent les risques sociaux et politiques de ces fausses informations.

Malgré ces tentatives de coopération, le phénomène s'amplifie et les informations erronées voire fausses se multiplient. Nous le voyons aujourd'hui dans le contexte de la guerre entre la Russie et l'Ukraine.

Ainsi, il est donc devenu impératif de mettre en place des technologies de fact checking afin de vérifier la fiabilité des contenus d'informations, C'est dans cet objectif que nous vous proposons une solution permettant d'évaluer la véracité d'un contenu sur le Web.

2 Exploration de données

Dans un premier temps, nous nous occupons de visualiser les données de notre Dataset.

On constate que la colonne media ne contient aucune information.

On analyse la taille des textes (en mots).

On décide de restreindre notre domaine d'étude aux textes qui contiennent 1000 mots ou moins.
Puis, on vérifie la proportion des Real News et Fakes news de notre nouveau Dataset (avec restriction effectuée).
On crée ensuite une liste qui se nomme non_fr qui est composée des textes qui ne sont pas en français ainsi que de leur indice (c'est-à-dire la position dans le Dataset).

L'idée est de supprimer tout ce qui n'est pas utile pour l'algorithme (un texte en anglais ne ferait que perturber l'algorithme pendant l'entrainement).

On s'occupe ensuite d'explorer les WordClouds de notre Dataset (Plus un mot est présent dans le texte,plus il est pris en considération dans le WordCloud.)

3 Pré-Traitement

Dans cette partie, nous nous occupons d'améliorer l'analyse de notre Dataset. On effectue un nettoyage des données, puis on vectorise les mots. On utilise deux types de vectorisation, la Simple Vectorisation (Count-

Vectorizer) qui calcule l'occurence d'un mot dans une phrase. Puis, la TF-IDF Vectorisation qui consistent à calculer la fréquence des mots dans les phrases.

3.1 Nettoyage du Dataset

On supprime les textes qui ne sont pas en français et on crée une fonction clean/text qui supprime les majuscules en les rendant minuscules, supprime la ponctuation, les chiffres et les alphabets étrangers pour ne pas perturber notre algorithme.

3.2 Méthode n°1 : Simple vectorisation

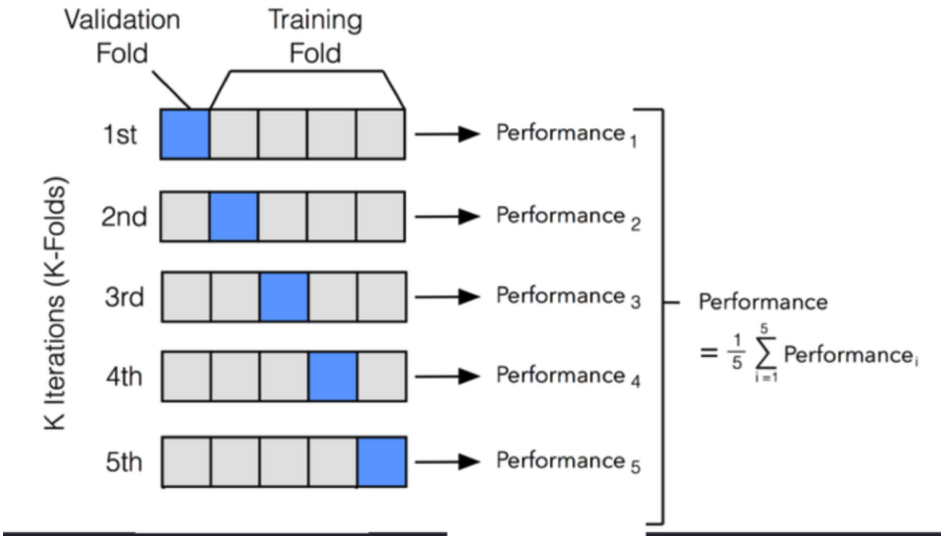
CountVectorizer prend une phrase et la coupe en mot. Ensuite, CountVectorizer marque le nombre d'occurences par mot dans une colonne.

3.3 Méthode 2 : tf-idf vectorisation

TfidfVectorizer et CountVectorizer sont des méthodes pour convertir des données textuelles en vecteurs, car le modèle ne peut traiter que des données numériques. Dans TfidfVectorizer, nous considérons le poids global d'un mot dans le document. Cela nous aide à traiter les mots les plus fréquents. En l'utilisant, nous pouvons les pénaliser. TfidfVectorizer pondère le nombre de mots par une mesure de la fréquence de leur apparition dans les documents.

4 Entrainement

Il faut savoir que tous les scores sont obtenus par cross-validation.



La Cross-validation consiste à entraîner puis validé notre modèle sur plusieurs découpes possible du trainset.

On teste plusieurs modèles de classification pour ne garder que les plus prometteurs.

```
name lgbm - Score : 0.947 +/- 0.004 - Time : 11.0 sec
name mlp - Score : 0.938 +/- 0.006 - Time : 30.0 sec
name log - Score : 0.932 +/- 0.004 - Time : 13.0 sec
name rfc - Score : 0.932 +/- 0.006 - Time : 13.0 sec
```

4.1 Premiers tests

On choisit donc de garder 4 modèles de Machine Learning :

Logistic Regression
Random Forest Classifier
Light Gradient Boosting Machine Classifier
Multi-layer Perceptron Classifier

4.2 Hyperparamétrisation

Définition : Les hyperparamètres correspondent aux paramètres d'ajustement des algorithmes d'apprentissage automatisé ou de machine learning. C'est au concepteur de l'algorithme de les configurer (souvent le data scientist).
Un algorithme d'apprentissage automatique ne peut, en effet, fonctionner sans ces hyperparamètres. Leur réglage (tuning) nécessite un travail statistique afin de déterminer ceux qui donneront le meilleur résultat.

On se lance donc à la recherche des meilleurs paramètres pour chacun des modèles.

4.2.1 Logistic Regression

La régression logistique est une méthode d'analyse statistique permettant de prédire un résultat binaire, tel que oui ou non, sur la base d'observations antérieures d'un ensemble de données. Un modèle de régression logistique prédit une variable de données dépendante en analysant la relation entre une ou plusieurs variables indépendantes existantes.

On utilise une recherche par grille GridSearchCV nous permet de trouver le modèle avec les meilleurs hyper paramètres en comparant les différentes performances de chaque combinaison grâce à la technique de cross-validation (encore une fois)
Ensuite on enregistre le modèle, avec les meilleurs paramètres, dans un dossier pickle.

Conclusion : Meilleurs paramètres :

- ☐ Solveur : liblinear ;
- ☐ Pénalité : '12' ;
- ☐ C : 1.06.

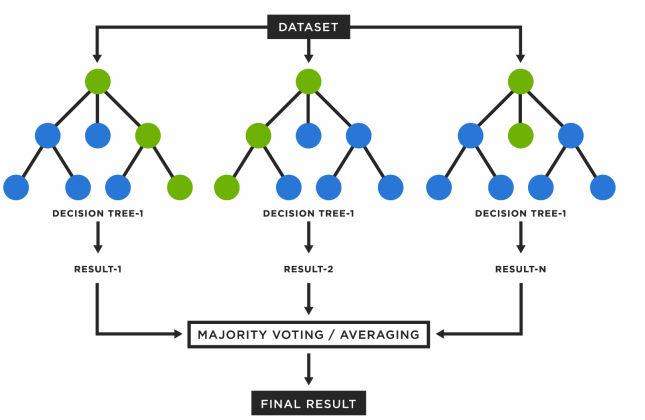
liblinear est plus rapide que newton & lbfgs pour le même résultat.

Accuracy :

0.932468

4.2.2 Random Forest Classifier

Random forest est un algorithme d'apprentissage automatique supervisé. Il s'agit de l'un des algorithmes les plus utilisés en raison de sa précision, de sa simplicité et de sa flexibilité.



On utilise une recherche par grille RandomizedSearchCv qui teste des combinaisons aléatoires d'une gamme de valeurs (nous devons définir le nombre d'itérations) en appliquant une validation croisée.
Cependant, il ne garantit pas que nous ayons les meilleurs paramètres.
Mais il est plus rapide que GridSearchCV car toutes les valeurs de paramètres ne sont pas testées.
Ensuite, on enregistre le meilleur modèle dans un dossier pickle.

Conclusion : Meilleurs paramètres :

- ☐ n _estimators : 800 ;
- ☐ max _features : log 2 ;
- ☐ min _samples _split : 10 ;
- ☐ min _samples _leaf : 2 ;
- ☐ max _depth : None.

Meilleure Accuracy :

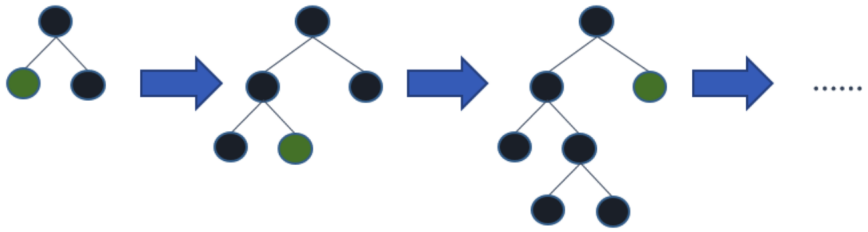
$$\boxed{0.943317}$$

4.2.3 Light Gradient Boosting Machine Classifier

Tout d'abord, on utilise une hypermaramétrisation bayésienne. (i.e : L'optimisation bayésienne est une méthode pour éviter d'avoir à chercher dans toutes les valeurs possibles des paramètres. Cela alterne entre de l'exploration (recherche de valeurs de paramètres inconnues) et l'utilisation de valeurs de paramètres dont l'impact sur le modèle est certain. Cet optimisation garde en mémoire ce qui a marché et ce qui n'a pas marché.

LightGBM est un algorithme qui reprend le principe du Gradient Boosting. En fait, il s'agit d'arbres de décisions (comme Random Forest Classifier) sauf qu'ici les arbres sont améliorés les uns après les autres.
Le premier arbre donne un résultat et on regarde l'erreur produite (la différence entre la valeur prédite et la valeur à prédire).
Ensuite, on entraine un second arbre avec les erreurs du premier.
Cet arbre essaie d'améliorer le résultat du premier.
On continue avec un 3eme arbre qui corrige les erreurs du 2nd etc...
Quand on n'arrive plus à améliorer les erreurs, l'algorithme s'arrête.

Light GBM (ou Light Gradient Boosting Machine) est un peu différent d'un algo de Boosting normal car il créé les arbres de décision feuille après feuille (leaf-wise) au lieu de le faire par niveau (level-wise).



Leaf-wise tree growth

Également, Light dans Light GBM signifie qu'il s'agit d'un algo optimisé qui consomme moins de RAM que la plupart des algos.

Conclusion : Meilleurs paramètres :

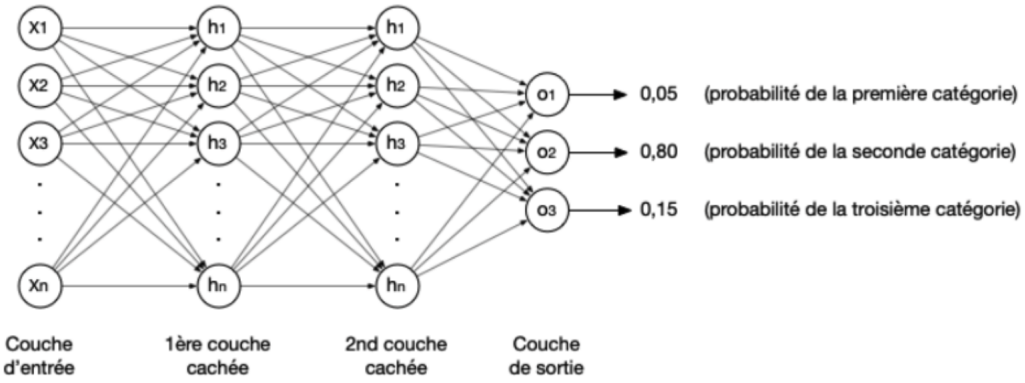
- ☐ bagging_fraction : 0.8563788970071229 ;
- ☐ boosting_type : gbdt ;
- ☐ feature_fraction : 0.2718860837301973, ;
- ☐ learning_rate : 0.8005608178965701 ;
- ☐ max_bin : 288 ;
- ☐ max_depth' : 30 ;
- ☐ min_data_in_leaf : 20 ;
- ☐ min_gain_to_split : 0.0 ;
- ☐ n_estimators : 517 ;
- ☐ num_leaves : 3000 ;
- ☐ reg_alpha : 0.0 ;
- ☐ reg_lambda : 77.56973209509714.

Accuracy :

0.9496183206106871

4.2.4 Multi-layer Perceptron Classifier

Le perceptron à multi-couche c'est un réseau de neurone.
Dans l'idée il s'agit d'une couche de réseaux de neurones d'entrée (les features) de plusieurs couches de neurones intermédiaires (les hidden layers) et d'une couche de sortie (ici, une classification binaire = un seul neurone).
Chaque neurone est lié à TOUS les neurones.
L'information circule du début vers la fin en passant par tous les neurones.
Les couches de neurones du début ont des informations simples tandis que les couches finales ont des informations plus complexes. C'est l'ensemble des chemins parcourus par les features jusqu'à la couche finale qui donne un résultat.



Conclusion : Meilleurs paramètres :

- ☐ bagging_fraction : 0.8563788970071229 ;
- ☐ boosting_type : gbdt ;
- ☐ feature_fraction : 0.2718860837301973 ;
- ☐ learning_rate : 0.8005608178965701 ;
- ☐ max_bin : 288 ;
- ☐ max_depth : 30 ;
- ☐ min_data_in_leaf : 20 ;
- ☐ min_gain_to_split : 0.0 ;
- ☐ n_estimators : 517 ;
- ☐ num_leaves : 3000 ;
- ☐ reg_alpha : 0.0 ;
- ☐ reg_lambda : 77.56973209509714.

Accuracy :

$$\boxed{0.9496183206106871}$$

5 Evaluation

Dans cette partie, nous allons tester nos modèles sur notre Dataset de test. On s'interessera aux scores F1, à la précision et la matrice de confusion.

5.1 Pré-traitement du jeu de test

Voir la partie Pré-traitement ci-dessus.

5.2 Modèles seuls

Définitions

Précision : Il s'agit de la mesure des cas positifs correctement identifiés parmi tous les cas positifs prédits. Ainsi, il est utile lorsque les coûts des faux positifs sont élevés.

$$\text{Precision} = \frac{\text{True Positive}}{(\text{ True Positive} + \text{False Positive})}$$

Accuracy : Mesure de tous les cas correctement identifiés.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

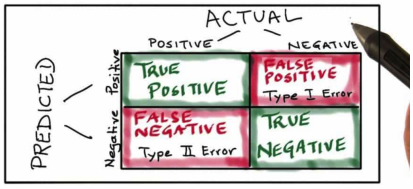
Rappel : Mesure des cas positifs correctement identifiés parmi tous les cas positifs réels. C'est important lorsque le coût des faux négatifs est élevé.

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

Score F1 : Il s'agit de la moyenne harmonique de la Précision et du Rappel et donne une meilleure mesure des cas mal classés que la Métrique de Précision.

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Matrice Confusion :



▷ Les vrais positifs – ici, les fake news détectées comme des fake news par l'algo.
▷Les faux positifs – ici, les real news détectées comme des fake news par l'algo.
▷Les vrais négatifs – ici, les real news détectées comme des real news par l'algo.
▷Les faux négatifs – ici, les fake news détectées comme des real news par l'algo.

5.3 Logic Regression

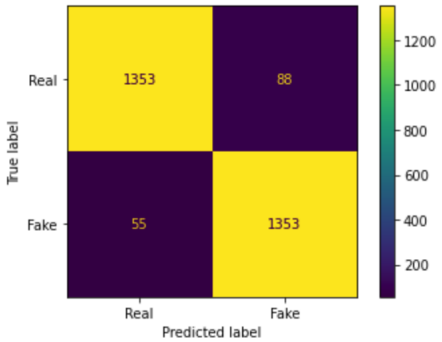
Accuracy :

0.9498069498069498

F1 score :

0.9498069498069499

Matrice de confusion :



5.4 Random Forest

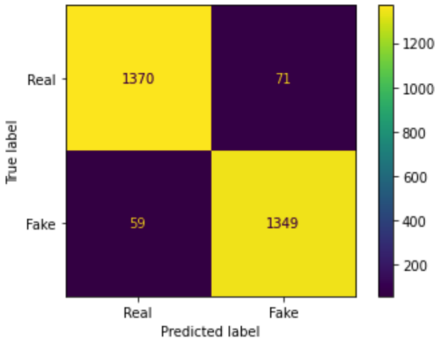
Accuracy :

0.9543699543699544

F1 score :

0.9540311173974539

Matrice de confusion :



5.5 LGBM

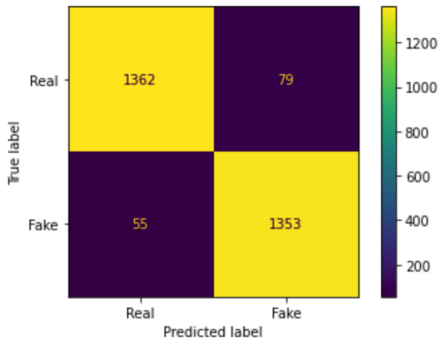
Accuracy :

0.952965952965953

F1 score :

0.9528169014084508

Matrice de confusion :



5.6 MLP

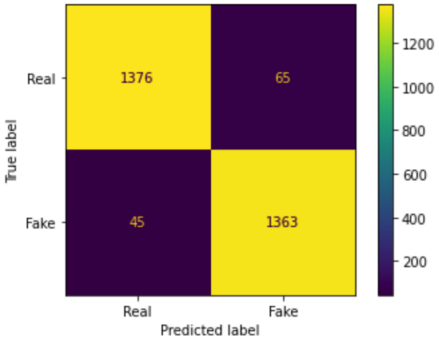
Accuracy :

0.9613899613899614

F1 score :

0.9612129760225671

Matrice de confusion :



5.7 Ensemble (voting)

Pour encore améliorer les performances, il est possible d'utiliser une méthode d'ensemble i.e. combiner les prédictions des quatre modèles.

C'est ici que nous obtenons les meilleurs scores.

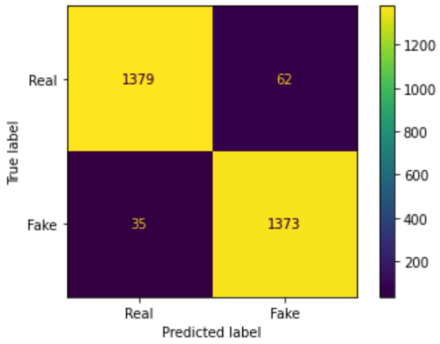
Accuracy :

0.965952965952966

F1 score :

0.9658811115019346

Matrice de confusion :



Notre matrice de confusion montre que notre algorithme est efficace.
En effet,notre algorithme identifie bien les faux positifs (les fake news détectées comme des fake news par l'algo (1379)) et les vrais positifs (les real news détectées comme des real news (1373)) qui est nettement plus importants que les faux positifs (real news détectées comme des fake news) et les faux négatifs (les fake news détectées comme des real news.)

6 Prédiction en direct

Il s'agira de prédire si la news qu'on écrit est Real ou Fake selon les 4 modèles évoquées précédemment.

7 Les améliorations possibles

L'analyse de texte est bien meilleure avec du Deep Learning car il est possible de traiter le texte comme une séquence. C'est à dire que les mots sont dans le bon ordre. Certains modèles, comme les transformers, permettent même de comprendre le contexte de la phrase pour mieux analyser l'importance de chaque mot dans celle ci.