Projet machine learning
Détection automatique des fake news à partir de données textuelles (Fake News Detection)

BELOT Mathieu : 21904568
CANHOTO Mickaël : 22304810
DEURVEILHER Jean Louis : 22102662
FONTAINE Emmanuel : 22312091
NGUYEN Thi-Christine : 21704571

26 mai 2024

Table des matières

1 Introduction

Dans le cadre de notre projet de Machine Learning, nous avons étudié la détection de fausses informations à partir d'articles de presse. Pour ce faire, nous avons utilisé un jeu de données issu de CLEF2022 (Conference and Labs of the Evaluation Forum). Ce jeu de données contient des articles de presse collectés à partir de sites de fact-checking.

Notre projet s'articule autour de plusieurs étapes. Tout d'abord, nous avons effectué un pré-traitement afin de transformer les textes en données exploitables par des modèles de classification. Ensuite, l'ingénierie des données nous permet de représenter les textes de manière structurée, rendant les données compréhensibles et utilisables pour les algorithmes de classification.

Après cela, la phase de classification nous a conduits à tester divers algorithmes tels que les arbres de décision, les SVC, le Naïve Bayes, et les K-NN. Les tâches de classification que nous avons abordées incluent la distinction entre articles vrais et faux, la différenciation entre articles vrais/faux et autres, ainsi que vrai, faux, mixte, et autre.

Enfin, notre projet se conclut par une analyse comparative des différents choix effectués en termes de pré-traitements, d'ingénierie des données, de sélection des features, et de modèles de classification, afin de comprendre l'impact de ces choix sur la qualité de la classification.

2 Ingénierie des données

La première partie consiste à traiter les données que nous avons afin de pouvoir les manipuler efficacement.

2.1 Pré-traitement

La première manipulation à réaliser est le pré-traitement des donneés. Ceci va nous permettre d'avoir des données uniformes. Nous allons faire des opérations telles que des contractions, suppression des espaces, suppression de la ponctuation...
Suite à cela, nous allons récupérer les "stop word". Les "stop word" sont des mots très utilisés dans une langue, mais qui ne portent pas beaucoup de sens lors d'une analyse textuelle. Pour récupérer les "stop word" et les exclure de notre classification, nous avons choisi d'utiliser la bibliothèque **nktl** qui permet de récupérer les "stop word" de la langue choisie.

Pour réaliser ce premier traitement, nous avons fait une fonction "clean_text" qui va prendre en entrée un texte et appliquer des opérations de nettoyage.

Contractions	Remplacement des contractions par leurs formes complètes : **can't → cannot**
Remove punctuation	Enlève les ponctuations
Lowercase	Enlève les majuscules
Alphabétique et numérique	Enlève les caractères spéciaux
Remove digits	Enlève les chiffres écrits numériquement
Digits to string	Ecrit les chiffres numériques en lettre
Remove stop world	Enlève les "stop words" tels que **the, and, a, ...**
Lemmatisation	Réduire les mots à leur formes de base ou formes canoniques : **eating, eaten → eat**
Racinisation	Remplace les mots par un autre mot relativement proche : **running, runner → run**
Tagging	Attribut un tag à chaque mot (verb, adjectives...).

TABLE 1 – Liste des pré-traitements effectués

Voici les différences que nous obtenons : **Texte original - Texte après traitement**

Distracted driving causes more deaths in Canada..	distracted driving cause death canada impaired...
Missouri politicians have made statements after...	missouri politician made statement mass shooting...
Home Alone 2 : Lost in New York is full of violence...	home alone lost new york full violence opinion...
But things took a turn for the worse when riot...	thing took turn worse riot police fired tear...

2.2 Vectorisation : TF-IDF

Après l'application du pré-traitement sur nos données, nous allons utiliser la méthode TF-IDF qui va permettre de prendre en compte l'importance d'un terme dans un document. En effet, dans la détection de fake news certains termes sont plus significatifs que d'autres.

Nous avons choisi d'utiliser des n-grammes (séquence de n mots consécutifs) dans le TD-IDF. Cela va nous permettre de récupérer des relations entre les mots. Nous allons pouvoir récupérer ainsi des expressions qui pourraient indiquer des fake news et ainsi avoir une meilleure représentation.
Par exemple, les mots youth et youtube possèdent un poids d'environ -0.108 ce qui signifie qu'ils n'ont peu d'importance dans l'article.

Pour chaque document, nous savons quels sont les mots les plus utilisés. Cependant, au vu du nombre de documents et de l'utilisation des n-grammes qui n'est pas forcément précise, nous nous sommes rendu compte que cette méthode n'était pas forcément la plus pertinente. Nous avons donc choisi d'utiliser une autre méthode (topic modelling).

2.3 Topic modelling : LDA

Le topic modelling est une technique d'apprentissage automatique non supervisée qui permet d'identifier les différents sujets principaux d'un ensemble de documents. Nous avons choisi d'intégrer le modèle de topic modelling LDA (Latent Dirichlet Allocation).

Il est crucial de bien choisir le nombre de sujets : un nombre trop faible peut entraîner une sous-représentation des sujets, tandis qu'un nombre trop élevé risque de mener à un sur-apprentissage (apprentissage par cœur du jeu de données).

Nous avons décidé d'extraire une dizaine de sujets, ce qui a permis d'obtenir des données homogènes avec peu de sujets sous-représentés. Le LDA a été retenu pour représenter nos données, car il est plus interprétable et précis que d'autres méthodes vectorielles pour notre objectif.

Voici quelques exemples de topics :

1. said eu cent people year uk government britain 000 court
2. new state year students education school york need schools children
3. food 2020 20 services tax party works pizza care chick
4. said people government just like country think ve state right

Et nous avons une moyenne de 117 docs par topic.

2.4 Le upsampling

Le upsampling va permettre de rééquilibrer les classes. On va donc récupérer les classes minoritaires et appliquer un suréchantillonage. Pour générer les nouveaux échantillons (égalisation des classes), nous utilisons la méthode SMOTE.

En effet, nous avons remarqué que notre jeu de données contient beaucoup plus de Faux que de Vrai, nous allons donc faire un upsampling pour éviter d'être biaisé.
Avant l'upsampling nous avions :

- Classe false : 578 exemples
- Classe mixture : 358 exemples
- Classe other : 117 exemples
- Classe true : 211 exemples

Maintenant nous avons 578 exemples pour toutes les classes.

3 Classification

Maintenant que nous avons traité nos données afin de pouvoir les étudier, nous allons réaliser leurs classifications. Cela nous permettra de répondre à la problématique en réalisant un apprentissage automatique.

Nous allons nous intéresser à trois des tâches de classification :
— VRAI vs. FAUX vs. MIXTE vs. AUTRE (quatre classes)
— VRAI vs. FAUX (deux classes)
— VRAI ou FAUX vs. AUTRE (trois classes)

3.1 Méthodes de classification

Pour ce faire nous allons utiliser 4 classifieurs :
— Le **Naive Bayes** basé sur le théorème de Bayes avec une hypothèse d'indépendance entre le caractéristiques.
— Le **SVC** (Support Vector Clustering) qui utilise un algorithme de partition avec le SVM pour séparer les donneés dans un espace.
— Le **Decision Tree** qui est un modèle prédictif dans lequel il partitionne les donneés en sous-groupes.
— Et finalement le **KNN** (k-Nearest Neighbors) qui classe les données en fonction de leurs voisins plus proches.

Nous allons en plus utiliser un K-fold pour séparer nos données d'entrainement avec nos données de test. Pour faire cela, nous allons diviser les données k fois en utilisant chaque sous-ensemble comme donnée de test. Cela va nous permettre d'avoir une estimation plus fiable. Nous avons décidé de partir sur **10-fold** ce qui donne une représentation tout en évitant une trop grande variance et un biais accru.

Nous allons pouvoir pour chaque classification tracer la courbe de précision et de entropy loss pour chaque fold.

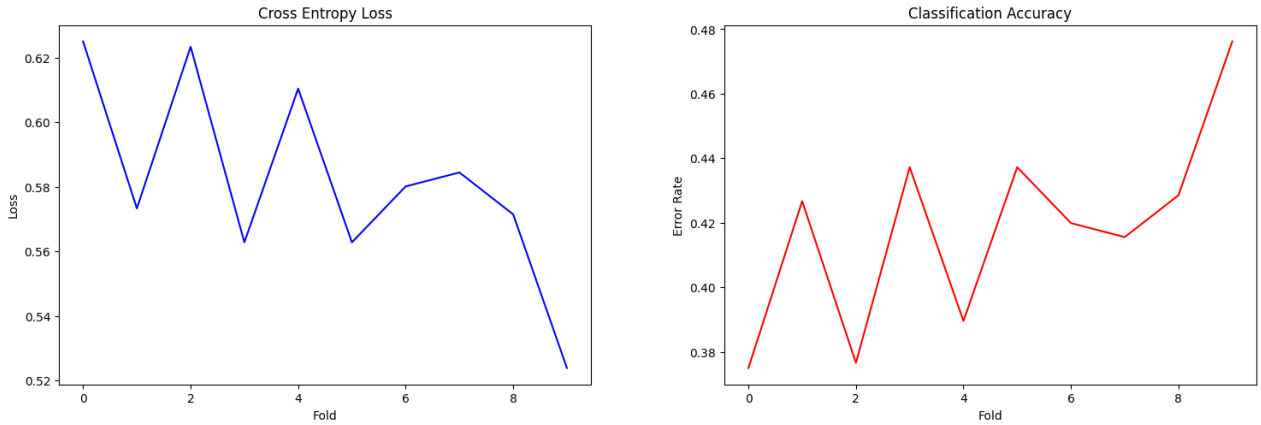


FIGURE 1 – Traçage des courbes d'EntropyLoss et d'Accuracy pour la méthode Naive Bayes

Nous pouvons suivre le même fonctionnement mais pour le nombre de voisin du K-NN.

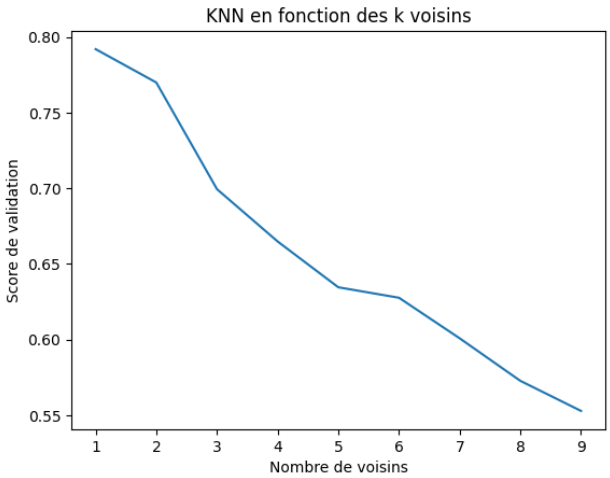


FIGURE 2 – KNN en fonction du nombre de voisins k

Nous pouvons constater que les courbes de Cross Entropy Loss et de Classifaication Accuracy pour le Naive Bayes sont inversement proportionnelles. Lorsque Cross Entropy Loss diminue, cela indique que le modèle devient plus certain/précis dans ses prédictions, ce qui conduit à une augmentation de l'Accuracy.

De plus, pour le KNN, on voit que le choix du nombre de voisins **k** est primordial. Si on a un nombre de voisins trop grand alors il se pourrait que les voisins contiennent des points d'autres classes et si trop petit, alors il se sera trop sensible au bruit. Nous pouvons le voir sur notre courbe, où plus on a de voisins, plus la précision diminue.

3.2 Les tâches de classification

3.2.1 VRAI vs. FAUX vs. MIXTE vs. AUTRE

Dans un premier temps, nous allons travailler sur tous les articles, qu'ils soient vrai, faux, un mixte des deux ou autres. On effectue la classification sur nos 4 classifieurs. On peut ensuite, pour chaque classifieur, calculer son score pour chaque k-fold puis le score moyen. Et par la suite, établir la matrice de confusion nous donnant les prédictions du classifieur. Finalement, nous allons en plus comparer chaque classifieur dans une courbe générale.

Voici les scores moyens 10-folds de chaque classifieur :
— Naïve Bayes : 0.596
— SVC : 0.866
— Decision Tree : 0.693
— KNN : 0.810

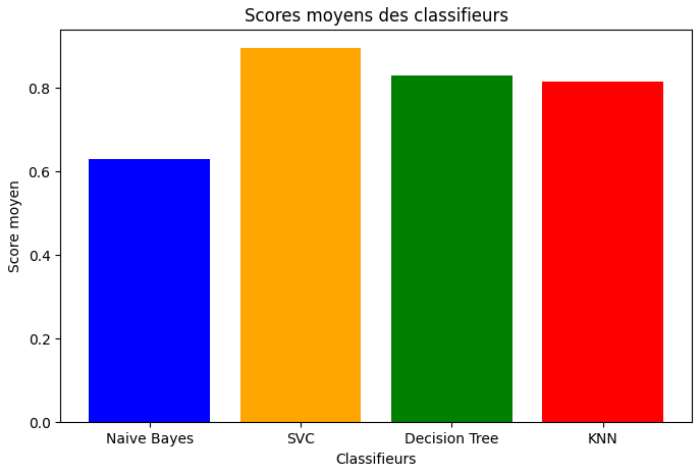


FIGURE 3 – Moyenne des classifieurs (VRAI vs FAUX vs MIXTE vs AUTRE)

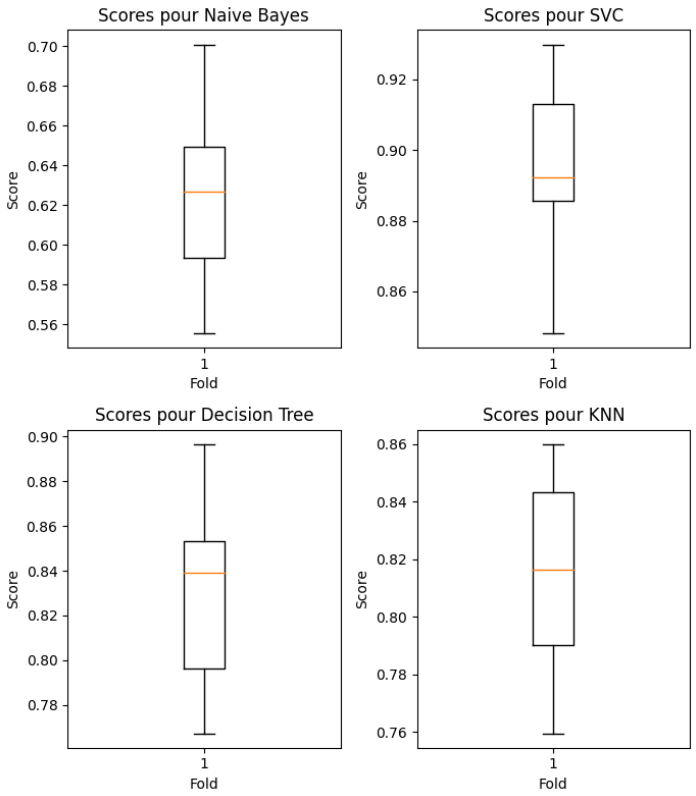
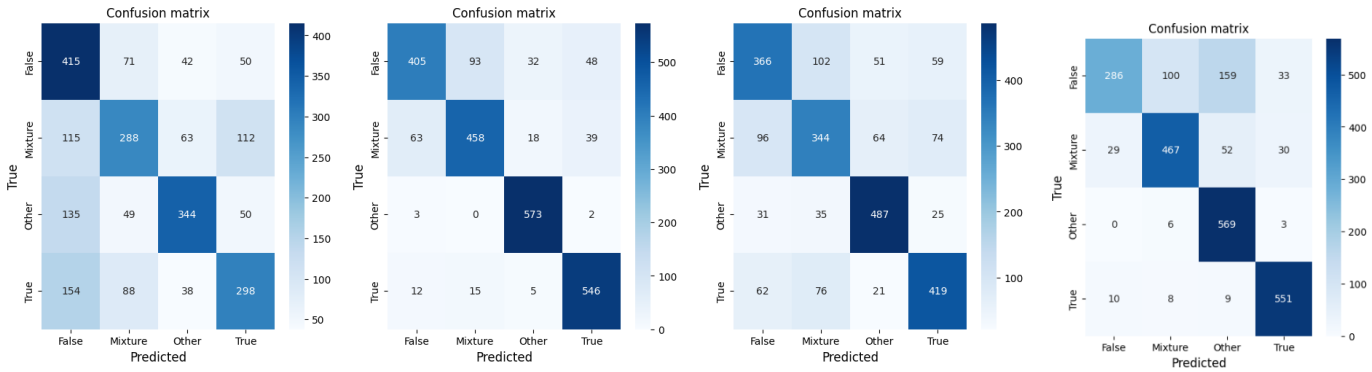


FIGURE 4 – Distribution des données selon les classifieurs (VRAI vs FAUX vs MIXTE vs AUTRE)

Nous pouvons remarquer que le Naïve Bayes et le Decision Tree ont un score moyen. Le score moyen du Naïve Bayes peut être dû au type de donnée qui n'est pas adapté. Pour le Decision Tree, il a dû surajuster les données. Le SVC et le KNN ont un scrore très élevé ce qui montre une grande précision. Grâce aux boîtes à moustaches, nous pouvons voir la distribution des informations.

Et leur matrice de confusion :



(a) Naive Bayes Confusion Matrix (b) SVC Confusion Matrix (c) Decision Tree Confusion Matrix (d) KNN Confusion Matrix

FIGURE 5 – Matrices de confusion des différents types de classification

Les matrices de confusions vont nous permettre d'analyser les précision et le rappel et ainsi connaître les tendances, plus de précision à 2. Nous pouvons remarquer que le Naives Bayes reconnait bien les Faux, le SVC, Decision Tree et KNN reconnaissent bien les Other. De manière générale c'est le SVC qui est le plus homogène et donc plus précis.

3.2.2 VRAI vs. FAUX

Nous allons maintenant travailler uniquement sur les articles vrai et faux. Nous devons donc créer un tableau X et Y contenant uniquement les valeurs de nos articles. Une fois cela fait, nous pouvons commencer à classifier nos articles. Toutes matrices de confusion ainsi que les résultats sont disponibles dans le notebook associé.

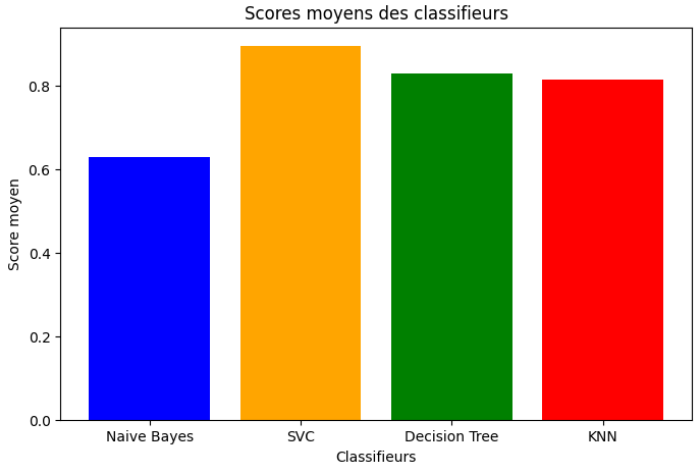
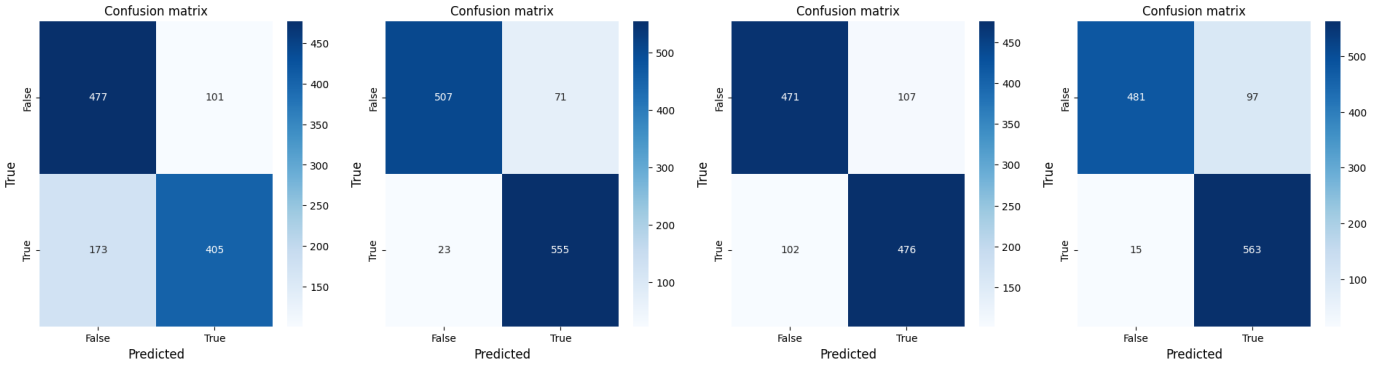


FIGURE 6 – Moyenne des classifieurs (VRAI vs FAUX)



(a) Naive Bayes Confusion Matrix (b) SVC Confusion Matrix (c) Decision Tree Confusion Matrix (d) KNN Confusion Matrix

FIGURE 7 – Matrices de confusion des différents types de classification

On remarque que le SVC est encore le plus efficace ici, avec les meilleurs scores de manière générale.

3.2.3 VRAI ou FAUX vs. AUTRE

Tout comme la partie précédente, nous allons créer un tableau X et Y spécifiquement pour cette tâche de classification. Cependant, il faut aussi redimensionner la quantité d'articles afin de ne pas avoir une sous-représentation. Nous allons donc réaliser un upsampling des articles autres.

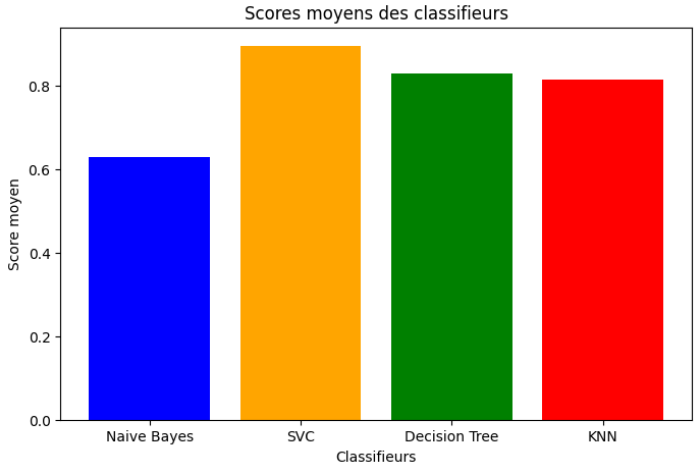
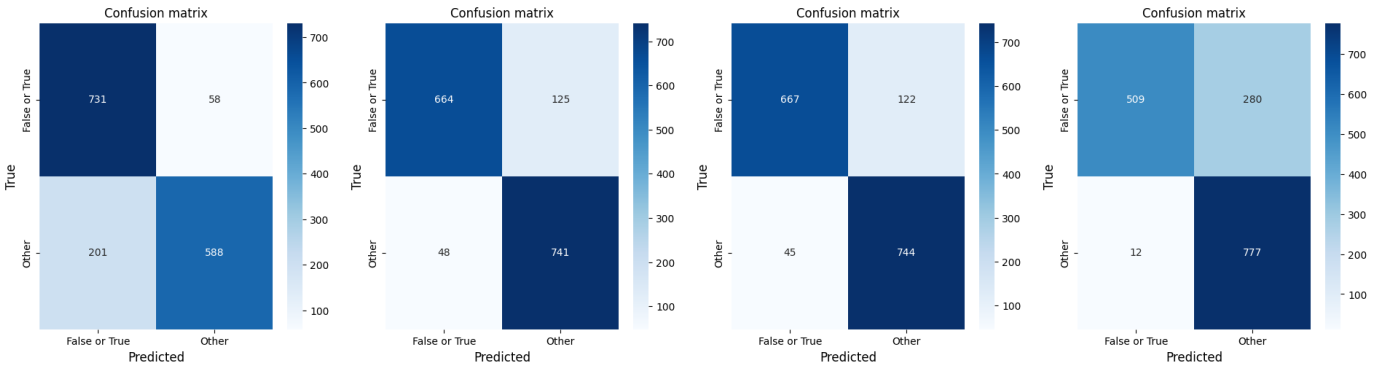


FIGURE 8 – Moyenne des classifieurs (VRAI ou FAUX vs AUTRE)



(a) Naive Bayes Confusion Matrix (b) SVC Confusion Matrix (c) Decision Tree Confusion Matrix (d) KNN Confusion Matrix

FIGURE 9 – Matrices de confusion des différents types de classification

4 Analyse des erreurs, validation et comparaisons des modèles

4.1 Analyse et validation

L'analyse des erreurs et la validation de nos modèles sont des étapes cruciales pour comprendre l'impact de nos choix de prétraitement, de sélection des features et de classification. Nous avons systématiquement évalué la performance de chaque modèle en utilisant des métriques telles que la précision, le rappel et la F-mesure. Ces évaluations nous ont permis de déterminer non seulement l'efficacité des modèles, mais aussi de comprendre les raisons sous-jacentes de leurs performances.

4.1.1 Prétraitement :

— **Suppression des stop words :** Nous avons constaté que la suppression des stop words améliore légèrement la précision des modèles en éliminant les mots non significatifs, permettant ainsi aux modèles de se concentrer sur les mots porteurs de sens.
— **Lemmatisation vs. racinisation :** La lemmatisation a montré de meilleurs résultats que la racinisation, car elle conserve le sens des mots mieux que la racinisation, qui peut parfois déformer le contexte en réduisant trop les mots.

4.1.2 Vectorisation :

— **TF-IDF avec n-grammes :** L'utilisation de n-grammes a permis de capturer les expressions courantes, améliorant ainsi la représentation textuelle. Cependant, l'ajout de n-grammes de grande taille a parfois conduit à un sur-ajustement.
— **Topic Modelling (LDA) :** Le LDA a fourni une représentation plus interprétable et a aidé à capturer les sujets principaux des documents, améliorant ainsi la précision des modèles de classification.

4.1.3 Upsampling :

SMOTE : L'application de SMOTE pour équilibrer les classes a significativement amélioré les performances des modèles en empêchant les biais vers les classes majoritaires.

4.1.4 Résultats des différentes méthodes de classification avec nos paramètres et pré-traitement

Nous avons comparé plusieurs algorithmes de classification, chacun ayant ses propres avantages et inconvénients. Voici un résumé des résultats obtenus :

Classe	Précision (NB)	Rappel (NB)	F-mesure (NB)	Précision (SVC)	Rappel (SVC)	F-mesure (SVC)
False	0.718	0.507	0.595	0.839	0.701	0.764
Mixture	0.498	0.581	0.537	0.810	0.793	0.801
Other	0.595	0.707	0.646	0.912	0.991	0.950
True	0.515	0.584	0.547	0.860	0.944	0.900

Table 2 – Comparaison des performances par classe pour les modèles Naïve Bayes et SVC

Classe	Précision (KNN)	Rappel (KNN)	F-mesure (KNN)	Précision (DT)	Rappel (DT)	F-mesure (DT)
False	0.880	0.495	0.632	0.659	0.633	0.646
Mixture	0.804	0.808	0.806	0.618	0.595	0.606
Other	0.721	0.984	0.834	0.782	0.843	0.811
True	0.893	0.953	0.922	0.726	0.725	0.726

Table 3 – Comparaison des performances par classe pour les modèles KNN et Decision Tree

— **Naïve Bayes** : Simple, rapide, mais suppose l'indépendance des features.
— **SVC** : Efficace pour les données non linéaires, bonne généralisation.
— **Decision Tree** : Interprétable, mais prompt à l'overfitting.
— **K-NN** : Simple, non paramétrique, mais sensible au bruit.

4.2 Comparaison de différents modèles

Dans cette dernière partie, nous allons tester empiriquement d'autres choix que nous aurions pu faire et l'impact sur la classification.

4.2.1 Différents paramètres des classifieurs

La première chose que nous pouvons faire est de modifier les paramètres des classifieurs. En effet, certains classifieur tels que le Naïves Bayes ou le SVC ont différents moyens de classifier.
Voici quelques résultats moyens obtenus avec des paramètres différents pour le Naïve Bayes :

Score pour le Naive Bayes ComplementNB : 0.55
Score pour le Naive Bayes BernouilliNB : 0.52

Le ComplementNB est généralement utilisé pour les classes déséquilibrées. Nous aurions pu alors utiliser ce classifieur sans upsampler au préalable. Le BernouilliNB est, quant à lui, adapté pour les données binaires, donc pour détecter uniquement les articles vrais ou faux. Ces paramètres sont donc plus efficaces dans des conditions spécifiques. C'est pourquoi le résultat obtenu est plus faible.

Le choix du noyau pour le SVC peut être un point à étudier, ici nous avons regardé pour les noyaux Polynomial et RBF (Radial Basis Function).
Voici les résultats obtenus avec un noyau différent :

Score pour le SVC poly : 0.40
Score pour le SVC rbf : 0.68

Le noyau Polynôme peut capturer des relations plus complexes, mais risque de surajuster les données si le degré est trop élevé. Alors que le noyau RBF est plus flexible et est capable de capturer des relations non linéaires, mais il nécessite un réglage plus minutieux des hyperparamètres.

4.2.2 Utilisation des variables vectorisées

Une autre modification que nous pouvons faire est de modifier les variables à traiter. Nous avons dans cette partie utilisé le classifieur Naïve Bayes avec les articles vectorisés.
Nous obtenons un résultat moyen de 0.63 ce qui est un bon résultat même si les variables n'ont pas été upsamplé.

5 Conclusion

Notre projet de machine Learning nous a permis, en suivant une méthodologie structurée, d'aborder différentes étapes allant de l'analyse du jeu de données, en passant par le pré-traitement des données textuelles, jusqu'à l'application et l'évaluation de divers modèles de classification.

Nous avons transformé les articles de presse en représentations exploitables pour les modèles de classification. Plusieurs algorithmes de classification ont alors été entraînés. Ensuite, les performances des modèles ont été évaluées à l'aide de métriques variées telles que la matrice de confusion, le rappel, la précision et la F-mesure. Cette évaluation a permis de comparer l'efficacité relative des différents modèles et des approches de pré-traitement.

En conclusion, les techniques de pré-traitement et d'ingénierie des données textuelles sont cruciales pour améliorer la performance des modèles de classification. Aucun modèle unique ne s'est révélé supérieur dans toutes les configurations. Toutefois... Cela nous a permis d'effectuer un travail de traitement de données à grande échelle et d'effectuer un apprentissage automatique à partir de données.