Universités de Montpellier

Faculté des Sciences de Montpellier

Département d'informatique

Projet

Machine Learning

HAI817

2022/2023

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

Réalisé Par :

MEFLAH WIDED

RAMDANE SOUHAIB

Encadré par :

Mr. Pascal Poncelet

M1 GL

Table des matières

1. Introduction

La propagation de fausses informations est devenue une préoccupation majeure dans le monde de l'information en ligne. La détection de ces informations trompeuses est donc devenue une tâche importante pour garantir la fiabilité des données. Dans ce rapport, nous présentons notre projet de détection de fausses informations, structuré en sept notebooks distincts. Nous commençons par une analyse approfondie de notre jeu de données, suivi des étapes d'ingénierie des données et de classification. Nous analysons également les erreurs de classification, validons et comparons nos modèles pour sélectionner le meilleur classifieur, les meilleurs paramètres de nettoyage et la meilleure entrée de classification. Ensuite, nous explorons les techniques de modélisation de sujet, de reconnaissance d'entités et de sélection de variables pour aller plus loin dans la détection de fausses informations. Enfin, nous présentons notre pipeline de classification de texte et son application sur un nouveau jeu de données.

2. Structure de notre projet

Notre projet est structuré en utilisant sept notebooks distincts.

- **Vrai_Faux.ipynb** : Ce notebook a été utilisé pour la première tâche de classification : {VRAI} vs. {FAUX} avec comme entrée “texte”, c’est notre notebook principal qui contient toutes les étapes du projet.
- **VraiFaux_Autre.ipynb**: Ce notebook a été utilisé pour la deuxième tâche de classification : {VRAI ou FAUX} vs. {AUTRE} avec comme entrée “texte”.
- **Vrai_Faux_Other_Mixture.ipynb** :il a été utilisé pour la troisième tâche de classification : {VRAI}vs {FAUX}vs{MIXTE}vs{AUTRE} avec comme entrée texte.

Ces trois notebooks servent à comparer les résultats de classification des différents échantillonnages.

- **Titre.ipynb** : Ce notebook a été utilisé pour la classification des titres avec la 1ere tâche de classification afin de comparer ses résultats avec la classification du texte.
- **Topic modelling.ipynb**: Ce notebook a été utilisé pour l'évaluation de l'utilité des sujets modélisés en tant que features en entrée des modèles de classification.
- **Entity recognition.ipynb** :Ce notebook a été utilisé pour l'évaluation de l'utilité des entités extraites en tant que features en entrée des modèles de classification.
- **Feature selection.ipynb** :Ce notebook a été utilisé pour la sélection des variables les plus importantes pour la classification des textes pour les différentes tâches de classification.

3. Analyse de notre jeu de données

La phase d'analyse d'un jeu de données est une étape cruciale pour pouvoir exploiter pleinement les informations contenues dans les données. Dans notre cas, nous disposons de deux dataframes : df_train et df_test. Avant d'entamer la phase d'ingénierie des données, les étapes suivantes ont été effectuées :

- **La concaténation des dataframes df_train et df_test :** Pour une analyse complète du jeu de données, nous avons d'abord concaténé les dataframes df_train et df_test en un seul dataframe df_all. Cette étape est essentielle pour disposer de toutes les données disponibles.
- **Description du jeu de données résultant** : Le jeu de données résultant contient 1876 articles représentés par les colonnes suivantes : public_id, text, title, our rating et ID. La colonne our rating contient l'évaluation de la fiabilité de l'article par les évaluateurs du site de fact-checking. Cette colonne contient quatre valeurs possibles : true pour vrai, false pour faux, mixture pour un mélange de vérité et de fausseté, et other pour une évaluation qui ne correspond pas à ces catégories.
- **Le déséquilibre du jeu de données** : Il est important de noter que ce jeu de données est déséquilibré, ce qui peut affecter la performance du modèle de classification.
- **Vérification des valeurs manquantes** : Nous avons vérifié la présence de valeurs manquantes dans le jeu de données. Nous avons constaté que les colonnes our rating et text ne contenaient aucune valeur manquante, ce qui nous dispense de tout traitement des valeurs manquantes dans ce jeu de données.

4. Ingénierie des données

Dans cette partie, nous abordons la première étape de ce processus, qui prépare les deux étapes ultérieures : le nettoyage de données et la représentation de texte, sans les mettre en œuvre immédiatement.

4.1.Nettoyage des données

Le nettoyage de données permet de réduire le bruit et les erreurs dans les données, ce qui peut améliorer la qualité des résultats de l'analyse et de l'apprentissage automatique. Pour ce faire, on a défini les fonctions suivantes :

MyCleanText : Cette fonction effectue le nettoyage du texte en supprimant les caractères spéciaux, la ponctuation, les mots vides, les nombres, et en appliquant la lemmatisation des mots et la racinisation des verbes.

TextNormalizer : Elle est utilisée pour transformer les données en utilisant la fonction MyCleanText. Les paramètres de nettoyage mentionnés dans la fonction MyCleanText sont passés en tant que booléens qui peuvent être activés ou désactivés afin de personnaliser le nettoyage des données.

Par la suite, on choisira la combinaison de paramètres de nettoyage qui améliore le mieux notre modèle de classification afin de l'utiliser dans le pipeline qui sera défini ultérieurement.

4.2.Représentation du texte

La représentation de texte TF-IDF est utilisée dans notre projet, car elle permet de tenir compte à la fois de la fréquence d'apparition d'un mot dans un document et de sa fréquence dans l'ensemble du corpus. Cette approche met en évidence les mots qui sont spécifiques à un document et qui ont donc une forte valeur informative pour la classification, ce qui en fait un choix pertinent pour notre pipeline de traitement de texte.

5. Les tâches de classification

Dans cette partie, nous abordons les tâches de classification où les assertions ont été classées et encodées en groupes, les classes ont été équilibrées, et la classification a été effectuée pour chaque échantillonnage pour les deux entrées de texte brut et de titre de texte.

5.1.Classement et encodage des assertions

Dans les trois tâches de classification, il est nécessaire de classifier les assertions en groupes en fonction de leurs labels, puis de convertir ces groupes en valeurs numériques compréhensibles pour un modèle de machine learning. Pour la première tâche, les assertions avec les labels "mixture" et "other" sont éliminées, et les labels "vrai" et "faux" sont convertis en 1 et 2 respectivement. Pour la deuxième tâche, les labels "mixture" et "other" sont regroupés sous la catégorie "AUTRE", tandis que les labels "vrai" et "faux" sont regroupés sous la catégorie "TrueFalse". Ces catégories sont ensuite converties en 1 et 2 respectivement. Pour la troisième tâche, chaque label est converti en une valeur numérique : "vrai" correspondant à 1, "faux" correspondant à 2, "mixture" correspondant à 3 et "other" correspondant à 4.

5.2.Équilibrage des classes

D'après l'analyse de notre jeu de données, nous avons constaté qu'il est déséquilibré. Par conséquent, l'équilibrage des classes a été effectué en ajustant la taille de chaque classe pour obtenir un nombre égal d'exemples de chaque classe de vérité.

Pour les trois tâches de classification, nous avons effectué un équilibrage des classes en **sous-échantillonnan**t la classe majoritaire et en **suréchantillonnant** les classes minoritaires pour

atteindre une taille égale entre les classes. La taille finale de chaque classe pour chaque tâche de classification est de 421 pour la première tâche, 561 pour la deuxième tâche, et 250 pour la troisième tâche.

5.3. Classification

Notre tâche consistait à choisir le meilleur classifieur avec les meilleurs paramètres de nettoyage. Pour cela, nous avons créé une fonction que nous avons nommée "ma_fonction". Cette fonction effectue une validation croisée K-fold afin de tester plusieurs classifieurs, notamment MultinomialNB, LogisticRegression, KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier et SVC, en variant les combinaisons de paramètres de nettoyage que nous souhaitons appliquer sur nos données à chaque appel de cette fonction. Cette variation est effectuée à l'aide de la fonction de normalisation de texte, déjà expliquée dans la section "nettoyage des données". Nous avons évalué les performances de chaque classifieur à chaque appel en calculant la précision, le rappel, la mesure F1 et l'exactitude. De plus, nous avons affiché le rapport de classification et la matrice de confusion pour chaque classifieur. Il convient de noter que cette classification est effectuée sur deux types d'entrées différents : le texte brut et le titre du texte, afin de choisir la meilleure entrée pour le modèle. Nous avons également effectué cette classification pour les trois tâches de classification, afin de sélectionner ultérieurement le meilleur échantillonnage.

6. Analyse des erreurs, validation et comparaison des modèles

Dans cette partie, nous allons comparer nos différents résultats de classification. Nous allons sélectionner le meilleur modèle avec la meilleure configuration de nettoyage, ainsi que le meilleur échantillonnage et l'entrée de classification la plus efficace.

6.1. Choix de la meilleure entrée de classifieur (texte/titre)

Le choix de l'entrée du classifieur est une décision importante qui peut avoir un impact significatif sur les résultats. Ainsi, nous avons également effectué la classification sur la colonne des titres. Toutefois, les résultats présentés dans le tableau ci-dessous montrent que les résultats de la classification des textes s'avèrent meilleurs que ceux obtenus à partir de la classification des titres. En effet, la classification du texte est préférable pour détecter les fake news car les titres peuvent être trompeurs, les fausses informations étant souvent dissimulées dans le contenu de l'article. De plus, le contexte est important pour comprendre le sujet, et le contenu de l'article contient plus de détails et de nuances qui permettent une détection plus précise des fake news. Il est donc recommandé de se concentrer sur la classification du contenu pour obtenir les meilleurs résultats dans la détection de fake news.

Entrée	*Meilleur classifieur*	*Exactitude*	*Précision*	*Rappel*	*F1 mesure*
Texte	SVM	0.776	0.776	0.777	0.775
Titre	SVM	0.653	0.624	0.735	0.694

6.2. Choix du meilleur échantillonnage

Les résultats présentés dans le tableau ci-dessous montrent que La première tâche de classification, qui consiste à distinguer les articles de presse réels des faux, a donné de meilleurs résultats que les autres tâches. Cela s'explique par la clarté des classes et la relative simplicité de la tâche, car il y a moins de caractéristiques ou de facteurs à considérer pour distinguer les articles de presse vrais des faux. En revanche, les autres tâches de classification ont donné des résultats moins satisfaisants pour plusieurs raisons. Tout d'abord, l'ambiguïté des classes a compliqué la deuxième tâche de classification, où la catégorie "autre" peut être difficile à définir précisément. Cette ambiguïté peut rendre la distinction entre les articles de presse vrais/faux et les autres plus ardue. De même, la présence d'une catégorie "mixte" dans la troisième tâche de classification peut également ajouter de la complexité et de l'ambiguïté à la classification. En outre, les autres tâches de classification peuvent être plus complexes que la première, ce qui peut rendre la classification plus difficile pour le modèle. Par exemple, dans la deuxième tâche, les articles de presse qui ne sont ni vrais ni faux peuvent être plus difficiles à catégoriser, car ils peuvent être basés sur des opinions sans être nécessairement faux ou vrais.

Echantillonnage	*Meilleur classifieur*	*Exactitude*	*Précision*	*Rappel*	*F1 mesure*
{VRAI} vs. {FAUX}	SVM	0.776	0.776	0.777	0.775
{VRAI ou FAUX} vs. {AUTRE}	SVM	0.706	0.707	0.712	0.707
{VRAI} vs. {FAUX} vs. {MIXTE} vs. {AUTRE}	RF	0.628	0.646	0.623	0.616

6.3. Choix des meilleurs paramètres de nettoyage

En se concentrant sur un des modèles de classification à savoir SVM, Les résultats présentés dans le tableau ci-dessous montrent que la combinaison de paramètres de nettoyage qui a donné les meilleurs résultats pour la détection de fake news est removestopwords avec lowercase et removedigit, et sans racinisation ou lemmatisation.

En supprimant les stop words, on peut réduire le nombre de variables dans les données et se concentrer sur les mots qui ont une signification plus importante pour la détection de fake news.

En transformant les données en minuscules, on peut normaliser le texte et réduire le bruit.

En supprimant les chiffres, on peut réduire le bruit dans les données et se concentrer sur les mots qui ont une signification plus importante pour la détection de fake news.

La racinisation et la lemmatisation peuvent introduire des erreurs ou des distorsions dans les données.

Paramètres de nettoyage activés	Paramètres de nettoyage désactivés	Exactitude	Précision	Rappel	F1 mesure
removestopwords, lowercase, removedigit, getstemmer, getlemmatisation	Aucun	0.774	0.78	0.764	0.771
removestopwords, lowercase, removedigit	getstemmer, getlemmatisation	0.777	0.776	0.777	0.775
removestopwords, lowercase, removedigit, getstemmer	getlemmatisation	0.771	0.773	0.766	0.769
removestopwords, lowercase, getstemmer	removedigit, getlemmatisation	0.772	0.777	0.762	0.768
removestopwords, lowercase, getstemmer, getlemmatisation	removedigit	0.777	0.78	0.767	0.773
removestopwords, lowercase, getlemmatisation	removedigit, getstemmer	0.77	0.774	0.761	0.766
removestopwords, lowercase, removedigit, getlemmatisation	getstemmer	0.773	0.774	0.77	0.771

6.4. Choix du meilleur classifieur

- **1ère tâche de classification {VRAI} vs. {FAUX}**

Les résultats présentés dans le tableau ci-dessous montrent que le modèle SVM semble être le meilleur choix. En effet, ce modèle a la plus haute mesure F1 parmi tous les modèles évalués, ainsi qu'une précision et un rappel élevés. De plus, sa matrice de confusion montre que le modèle a une bonne capacité à classer les deux classes de manière équilibrée. Bien que ce modèle ait le temps d'exécution le plus long, cela peut être justifié par sa performance élevée.

Le SVM a probablement donné le meilleur résultat car il est efficace pour les tâches de classification binaire, il est capable de séparer les 2 classes en maximisant la marge entre elles ce qui peut conduire à de meilleurs résultats de classification .de plus il est capable de trouver une frontière de décision complexe et non linéaire entre les deux classes, ce qui peut être important pour la détection de fake news. En comparaison, les autres modèles peuvent avoir des limitations en termes de complexité de la frontière de décision.

Classifieur	Matrice de confusion	Exactitude	Précision	Rappel	F1 mesure
MultinomialNB	[[367 54] [166 255]]	0.739	0.688	0.875	0.768
LR	[[322 99] [98 323]]	0.766	0.766	0.767	0.765
KNN	[[324 97] [153 268]]	0.703	0.679	0.77	0.719
CART	[[268 153] [137 284]]	0.656	0.661	0.633	0.643
RF	[[324 97] [117 304]]	0.736	0.721	0.805	0.757
SVM	**[[326 95] [94 327]]**	**0.776**	**0.776**	**0.777**	**0.775**

- **3ème tâche de classification {VRAI} vs. {FAUX} vs. {MIXTE} vs. {AUTRE}**

Le modèle RF semble être le meilleur choix. En effet, ce modèle a la plus haute mesure F1 parmi tous les modèles évalués, ainsi qu'une précision et un rappel élevés.

Le RF a probablement donné le meilleur résultat car il peut identifier les caractéristiques spécifiques de chaque classe, ce qui lui permet de mieux distinguer les différentes classes et d'obtenir de meilleurs résultats.

Remarque : La deuxième tâche de classification {VRAI ou FAUX} vs. {AUTRE} a donné SVM comme meilleur résultat, ce qui peut être justifié de la même manière que la première tâche de classification, car il s'agit également d'une classification binaire.

7. Pour aller plus loin

Dans cette partie, nous allons explorer les techniques de la modélisation par sujet, la reconnaissance d'entités, et la sélection des variables pour aller plus loin dans la détection de fake news. Nous terminerons cette partie par la sélection du meilleur choix des hyperparamètres de notre meilleur classifieur.

7.1. Modélisation par sujet

En utilisant la modélisation par sujets, il est possible d'extraire des informations à partir de grands ensembles de données de textes. Dans ce cas, chaque texte a été étiqueté avec des sujets spécifiques qui peuvent être utilisés comme features en entrée des modèles de classification afin de savoir si cela peut aider à améliorer les performances des modèles de classification en identifiant les sujets qui sont fortement corrélés avec les fake news.

L'analyse des résultats indique que les sujets les plus associés à la propagation de fausses informations sont liés à des thèmes d'actualité tels que la pandémie de COVID-19, les vaccins, la politique gouvernementale et les conflits militaires. Ces sujets sont considérés comme sensibles car ils ont un impact important sur l'opinion publique et sont donc vulnérables aux campagnes de désinformation. Les résultats montrent également que certains mots-clés tels que "vaccin", "COVID", "virus" et "pays" sont fortement associés aux fausses informations. Les sujets sont donc susceptibles d'être ciblés par des campagnes de désinformation visant à manipuler l'opinion publique.

Les résultats présentés dans le tableau ci-dessous montrent que la classification directe du texte a donné des résultats significativement meilleurs que la modélisation par sujet dans ce contexte. On peut conclure que dans le cadre de la détection de fake news, les sujets ne sont peut-être pas les caractéristiques les plus pertinentes à considérer pour identifier une information erronée ou trompeuse. En effet, La détection de fake news nécessite une analyse approfondie du contenu sémantique des textes plutôt que de simplement se fier aux sujets, qui peuvent être similaires dans les deux types de nouvelles et ne pas fournir suffisamment d'informations pour permettre une distinction précise.

Classification	*Exactitude*	*Précision*	*Rappel*	*F1 mesure*
Texte brut	0.776	0.776	0.777	0.775
Modélisation par sujet	0.54	0.53	0.62	0.57

7.2. Reconnaissance d’entité

L’extraction d'entités peut aider à identifier les noms d'organisations, de personnes et de lieux qui sont fortement corrélés avec la production de fausses nouvelles. Ces informations peuvent ensuite être utilisées comme features en entrée pour les modèles de classification, afin d'améliorer leur précision et leur capacité à détecter les fausses nouvelles.

Les résultats de l'analyse suggèrent que la mention d'entités dans les articles est fortement liée à la véracité de ces articles. Les entités mentionnées le plus souvent sont des noms familiers tels que "Trump", "China", "U.S." et "American". Cette forte corrélation pourrait indiquer que les fausses nouvelles ont tendance à utiliser des noms populaires pour attirer l'attention des lecteurs.

Les résultats présentés dans le tableau ci-dessous montrent que l'extraction d'entités en tant que fonctionnalités en entrée des modèles de classification donne des performances similaires à celles obtenues sans cette extraction.

Ces résultats suggèrent que l'extraction d'entités ne semble pas améliorer significativement les performances de classification dans le cadre de la détection de fausses informations.

Classification	*Exactitude*	*Précision*	*Rappel*	*F1 mesure*
texte brut	0.776	0.776	0.777	0.775
Extraction d’entité	0.74	0.73	0.78	0.75

7.3. Sélection des variables

Nous avons utilisé des méthodes de sélection de variables pour identifier les features les plus discriminantes pour les trois tâches de classification. Nous avons comparé les listes obtenues pour chaque tâche.

Pour la tâche 1 de classification {VRAI} vs. {FAUX}, les features discriminantes de la liste sont assez difficiles à interpréter, car elles sont des mots qui peuvent apparaître dans des contextes très différents. Il est donc difficile de tirer des conclusions significatives à partir de cette liste.

Pour la tâche 2 de classification {VRAI ou FAUX} vs. {AUTRE}, la liste des features discriminantes suggère que les sujets de la santé publique et de la sécurité sont particulièrement importants pour distinguer les deux classes. Cela peut s'expliquer par le fait que de nombreux articles de vérification des faits portent sur des allégations de santé publique ou de sécurité, telles que les allégations sur les vaccins ou les théories du complot sur les attentats terroristes.

Pour la tâche 3 de classification {VRAI} vs. {FAUX} vs. {MIXTE} vs. {AUTRE}, la liste des features discriminantes suggère que la classification des articles dans les catégories {VRAI} ou {FAUX} est plus difficile que pour la tâche 2, car la plupart des features discriminantes sont liées à des ambiguïtés ou des nuances dans les faits présentés. La présence de certains mots, tels que "negotiating", "intended" ou "misleadingly", peut indiquer que les articles sont difficiles à classer car ils présentent des faits qui peuvent être interprétés de différentes manières. Les autres features discriminantes sont assez spécifiques et sont souvent liés à des noms propres, tels que les noms de personnes, de lieux ou d'organisations.

En comparant les listes obtenues pour les 3 tâches, on peut voir que la tâche 2 est celle qui semble la plus facile à résoudre en utilisant des features discriminantes, car il existe des sujets particuliers qui sont particulièrement liés à une classe ou à l'autre. Les tâches 1 et 3 sont plus difficiles à résoudre car les features discriminantes sont plus ambigües et sont souvent liées à des contextes plus larges.

7.4. Choix des hyperparamètres pour le meilleur classifieur

L'étape consiste à déterminer les meilleurs paramètres pour le modèle de classification SVM.

Les hyperparamètres obtenus pour SVM (C=10, gamma=0.001 et kernel=linear) sont les meilleurs car ils ont permis d'obtenir le score d'accuracy le plus élevé lors de l'application de la technique de GridSearchCV.

Le paramètre C=10 indique une forte pénalité pour les erreurs de classification, ce qui conduit à un modèle plus rigoureux.

Le paramètre gamma=0.001 indique une faible influence pour les points plus éloignés, ce qui peut aider à obtenir une meilleure généralisation.

Le paramètre kernel=linear indique que les données sont linéairement séparables. Ces valeurs d'hyperparamètres ont été choisies en effectuant une recherche exhaustive de toutes les combinaisons possibles dans l'espace des hyperparamètres, et ont donné les meilleurs résultats pour la classification de textes dans ce cas spécifique.

Il est donc possible que ces paramètres aient permis de mieux séparer les classes et de modéliser des frontières de décision plus complexes, ce qui a amélioré les performances de notre modèle SVM.

Remarque : Il convient de souligner que cette étape ne concerne que l'entrée texte et la première tâche de classification, car l'analyse effectuée a montré qu'elle fournissait les meilleurs résultats. Toutefois, pour la troisième tâche de classification qui a désigné le RF comme meilleur classifieur malgré des résultats moins performants, nous avons choisi de sélectionner les meilleurs paramètres pour le classifieur RF afin de mieux maîtriser les différents hyperparamètres des différents classifieurs.

8. Pipeline de classification de texte

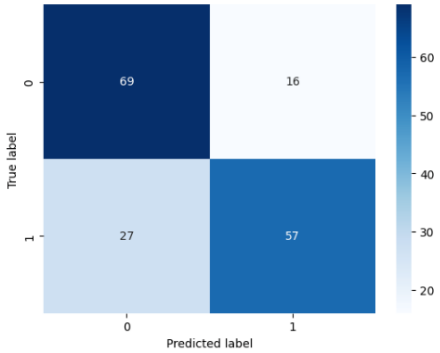
Dans cette partie, nous présenterons la création de notre pipeline de classification, sa sauvegarde et son application à un nouveau jeu de données pour prédire les étiquettes de classe.

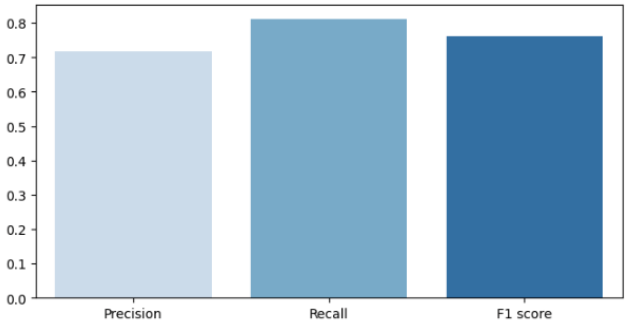
8.1. Création du pipeline

Afin de créer une chaîne de traitement complète pour fournir un modèle performant de classification de texte, capable de prédire la classe de nouveaux textes, nous avons défini une série d'étapes. Tout d'abord, nous avons effectué le nettoyage et la normalisation du texte en utilisant la classe TextNormalizer, en combinant les parametres de nettoyage qui ont donné les meilleurs résultats, tels que la suppression des stopwords, la mise en minuscule et la suppression des chiffres. Ensuite, nous avons procédé à l'extraction de caractéristiques en utilisant TfidfVectorizer, qui a converti les textes en vecteurs numériques de caractéristiques. Pour classifier les textes en fonction de leurs caractéristiques extraites, nous avons utilisé l'apprentissage supervisé avec le modèle SVM, qui a été optimisé en fonction des paramètres qui ont donné les meilleurs résultats. Enfin, pour évaluer les performances du modèle, nous avons utilisé des mesures telles que la précision, le rappel, le score F1 et la matrice de confusion sur un ensemble de test.

Interprétation des résultats

Le modèle de classification de texte a une précision globale de 74,56%, ce qui est raisonnable. La précision est plus élevée pour la classe False (classe 2) que pour la classe "True" (classe 1), avec un score de 78% contre 72%. Le rappel est plus élevé pour la classe "True" que pour la





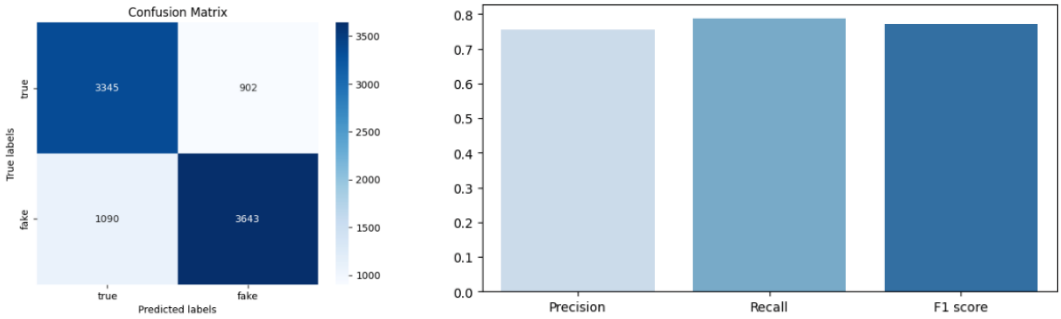
classe False", avec un score de 81% contre 68%. Le score F1 est de 76%. La matrice de confusion montre que le modèle a tendance à classer plus de textes dans la classe "True" que dans la classe "False".

8.2. Sauvegarde du pipeline

Il s'agit de sauvegarder notre modèle et le pipeline pour une utilisation ultérieure sans avoir à réentraîner le modèle ou à reconstruire le pipeline à chaque fois.

8.3. Utilisation de pipeline

Nous avons cherché un nouveau jeu de données pour tester notre modèle, et nous avons trouvé un ensemble de données contenant uniquement des vraies informations et un autre ensemble de données contenant des fausses informations. Nous avons étiqueté les deux ensembles de données et les avons concaténés pour créer un nouveau jeu de données de taille 17 960 articles. Ensuite, nous avons appliqué notre pipeline au nouveau jeu de données pour prédire les étiquettes de classe.



Le modèle a obtenu une précision de 0.75 pour la classe 1 (true news) et 0.80 pour la classe 2 (fake news), avec un rappel de 0.79 pour la classe 1 et 0.77 pour la classe 2. Le score F1 est de 0.77 pour l'ensemble des données de test, ce qui indique que le modèle est assez performant. La matrice de confusion montre que le modèle a prédit correctement 3345 true news et 3643 fake news, mais a également classé à tort 1090 true news comme étant des fake news et 902 fake news comme étant des true news.

9. Conclusion

En somme, ce projet nous a permis d'explorer différentes techniques de classification de texte et de les appliquer à la détection de fake news. Bien que le jeu de données ait présenté des défis, nous avons pu surmonter ces obstacles pour atteindre notre objectif. Ce travail peut être poursuivi en utilisant des techniques plus avancées de traitement de langage naturel, telles que l'utilisation de réseaux de neurones profonds, pour améliorer la précision de la détection de fake news.