FACULTÉ DES SCIENCES
DE MONTPELLIER

UNIVERSITÉ DE MONTPELLIER
M1 INFORMATIQUE - IMAGINE

Rapport de projet :

Détection automatique de Fake News à partir de données textuelles

HAI817I - MACHINE LEARNING 1

Étudiants :
IMPARATO Adèle – 22200570
LUCIANI Nicolas – 22200159
HUTTE Norman – 21908587
MAURIN Christina – 21600781

Professeurs :
M. PONCELET Pascal
M. TODOROV Konstantin

Avril 2023

Introduction

Ce projet s'inscrit dans le cadre de l'unité d'enseignement "HAI817I : Machine Learning 1" du second semestre du Master 1 IMAGINE et vise à développer un détecteur de Fake News (information fallacieuse).

Le jeu de données provient de *CLEF2022* et contient des articles de presse collectés à partir de sites de fact-checking. Ces articles sont labelisés comme 'true', 'false', 'other' ou 'mixture'. On compte au total 1264 articles dont 211 'true', 578 'false', 117 'other' et 358 'mixture'.

	public_id	text	title	our rating
0	5a228e0e	Distracted driving causes more deaths in Canad...	You Can Be Fined $1,500 If Your Passenger Is U...	false
1	30c605a1	Missouri politicians have made statements afte...	Missouri lawmakers condemn Las Vegas shooting	mixture
2	c3dea290	Home Alone 2: Lost in New York is full of viol...	CBC Cuts Donald Trump's 'Home Alone 2' Cameo O...	mixture
3	f14e8eb6	But things took a turn for the worse when riot...	Obama's Daughters Caught on Camera Burning US ...	false
4	faf024d6	It's no secret that Epstein and Schiff share a...	Leaked Visitor Logs Reveal Schiff's 78 Visits ...	false

FIGURE 1 – Extrait du jeu de données

Ce sont sur ces différentes assertions que différents modèles vont être testés afin de déterminer lesquels sont les plus performants pour prédire si un article de presse correspond à une classe ou une autre. Voici les différents output possibles :

— La première classera les articles selon deux classes : Vrai / Faux.

— La seconde classera les articles selon deux classes : Vrai, Faux / Autre.

— La dernière classera les articles selon quatre classes : Vrai / Faux / Autre / Mixte.

Avant cette étape de classification, plusieurs travaux d'ingénierie de données vont être effectués. Ces pré-traitements permettront d'améliorer les performances des modèles employés.

1 Ingénierie des données

Afin de préparer les données, on leur applique divers pré-traitements permettant aux modèles de classification de comprendre et de traiter celles-ci de manière optimale. Globalement, ces techniques réduisent la complexité des données. Voici les techniques de pré-traitement choisies :

- **Elimination des contractions**
  Dans un premier temps, on élimine les différentes contractions du texte de chaque article. En effet, comme ce sont des données tirées de l'anglais, on retrouve par exemple des contractions du type "*don't*" que l'on souhaite ainsi transformer en "*do*" et "*not*".
- **Regex Tokenisation**
  La tokenisation en expression régulière permet de diviser le texte des articles en "tokens" (i.e. mots ou caractères) en considérant une expression régulière, ici $r"(\backslash w+|\#\backslash d|\backslash ?|!)|.|:|;$. Cela permet d'obtenir une liste de tokens qui vont pouvoir subir d'autres techniques de pré-traitement par la suite.
- **Minusculation**
  Afin de traiter plus efficacement les données, une minusculation des tokens est appliquée. Cela permet d'éviter de considérer deux mots similaires, comme "*Media*" et "*media*" par exemple, comme étant deux mots différents.
- **Remplacement des nombres**
  Cette technique de pré-traitement permet de remplacer l'ensemble des différents nombres tokenisés par le token "number". Indépendamment, les nombres ne représentent rien, il est donc difficile de les traiter autrement. Il est à noter que cette technique permet uniquement de remplacer des strings contenant exclusivement des chiffres. Des nombres comme "1.5", "-2" ou encore "$500" ne seront donc pas remplacés.
- **Elimination des stop words**
  L'élimination de divers stop words fournis par le package nltk.corpus.stopwords de Natural Language Toolkit permet aussi de pré-traiter les données. Ces mots font partie d'un vocabulaire qui n'apporte pas/peu de sens au texte. Ils peuvent donc être supprimés des textes.
- **Elimination des mots indésirables**
  De plus, il a été observé que certains mots tels que des lettres individuelles ou encore des ponctuations peuvent être éliminés puisqu'il n'apportent pas ou très peu de sens aux textes.
- **Lemmatisation**
  Par la suite, on procède à la lemmatisation des différents tokens, ce qui va permettre de transformer chaque token en sa forme neutre canonique. Cette technique est efficace pour comparer plus facilement les tokens des textes et générer une taille du vocabulaire plus courte.
- **Upsampling / Downsampling**
  L'upsampling permet de rajouter des données tandis que le downsampling permet de supprimer des données. Ces deux méthodes de ré-échantillonnage ont toutes les deux pour but d'équilibrer le nombre de données dans les différentes classes afin de supprimer un potentiel biais. Il est à noter que ce ré-échantillonnage n'a été utilisé que dans le cadre de l'exécution à 2 sorties (i.e. Vrai / Faux) puisque les données contiennent beaucoup plus d'éléments notés 'true' que 'false'.

En somme, ces pré-traitements permettent une meilleure compréhension du sens des données textuelles par les modèles et une amélioration de leur performance dans leur tâche de classification.

2 Tâches de classification

Après le pré-traitement des données, il convient ensuite d'entraîner différents modèles de classification sur celles-ci afin de pouvoir prédire la classe d'un nouveau document. Pour ce faire, on transforme les données textuelles en données numériques à l'aide de deux 'vectoriseurs'.

Premièrement, le TfidfVectorizer va être utilisé, suivi du CountVectorizer. Ces deux méthodes de vectorisation sont toutes deux basées sur l'approche du modèle "sac de mots". Le but d'un vecteur de sac de mots est de représenter chaque article comme un vecteur contenant la fréquence des mots de ce document. Il y a donc un dictionnaire commun à l'ensemble des articles. La simple différence entre ces deux vecteurs est que CountVectorizer compte simplement le nombre d'occurrences de chaque mot dans un document alors que le TfidfVectorizer utilise un score qui prend en compte à la fois la fréquence du mot dans le document et la fréquence inverse du mot dans tous les documents de la collection. De plus, il est possible de spécifier la taille des n-grammes utilisés dans le vecteur à l'aide du paramètre $ngram\_range$, ce qui va permettre d'influencer les résultats.

Une fois les données converties en vecteurs numériques, plusieurs modèles de classification ont été testés afin de trouver celui qui mène aux meilleurs résultats :

- **k-Nearest Neighbors (kNN)** : capable de trouver des similarités entre des articles et de prédire la classe d'un article inconnu en fonction de données similaires (ses k voisins les plus proches). Après plusieurs tests, il a été décidé de considérer les 3 voisins les plus proches et d'utiliser la distance de Manhattan ($p = 1$).
- **Decision Tree (DT)** : crée un modèle en mesure de prédire la classe d'un élément en apprenant des règles simples de décision découlant de son apprentissage. Après plusieurs tests, il a été décidé de limiter la recherche à 5 échantillons minimaux par feuille ainsi que la profondeur maximale à 50. Ces paramètres se sont montrés les plus intéressants en termes de résultats et de temps de calcul.
- **Multinomial Naive Bayes (NB)** : capable de traiter des données textuelles de grande dimension et de trouver des associations entre les mots. Il est à noter que cet algorithme a été utilisé sans smoothing ($alpha = 0, force\_alpha = True$).
- **Multinomial Logistic Regression (LR)** : modèle de classification linéaire capable de prédire la probabilité qu'un article soit vrai ou faux (ou autre, ou mixte) en cherchant le seuil optimal séparant ces classes.
- **Random Forest (RF)** : ensemble d'arbres de décision indépendants.

3 Analyse des erreurs, validation et comparaisons des modèles

3.1 Word clouds

Les word cloud, ou en français "nuage de mots" permettent de visualiser les mots les plus fréquents d'une classe à l'autre. Ci-dessous, les nuages de mots pour les classes 'True', 'False', 'Mixture' et 'Other' après pré-traitement des données.

FIGURE 2 – False cloud

FIGURE 3 – True cloud

FIGURE 4 – Other cloud

FIGURE 5 – Mixture cloud

Il est à noter que malgré la suppression des stop words, il reste encore certains mots tels que "people" ou "one" qui reviennent dans tous les nuages, ces mots-là ne sont généralement pas représentatifs de la classe en elle-même puisqu'ils sont repris dans de nombreux documents.

3.2 Résultats obtenus

De nombreux tests ont été effectués pour chaque classificateur. D'une part, pour trouver les paramètres optimaux, d'autre part, pour les tester sur les différents output possibles. Mais aussi, différentes combinaisons de pré-traitements ont été testées afin d'obtenir les meilleurs résultats possibles. Ci-dessous, voici plusieurs tableaux récapitulant les meilleurs résultats obtenus :

Classifieur	Vectoriseur	Pré-traitements	Classes de prédiction	Training accuracy	Testing accuracy
kNN	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables,	True / False	0.78	0.60
LR	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling	True / False	0.94	0.66
NB	CountVectorizer	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling	True / False	0.98	0.68
RF	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling	True / False	0.94	0.65
RF	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling, upsampling	True / False	0.96	0.67
DT (meilleurs paramètres)	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling, upsampling	True / False	0.98	0.59

FIGURE 6 – Sélection de résultats obtenus pour la prédiction en deux classes : Vrai / Faux

Classifieur	Vectoriseur	Pré-traitements	Classes de prédiction	Training accuracy	Testing accuracy
kNN	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.93	0.86
NB	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.98	0.80
LR	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.83	0.84
DT	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.89	0.62
NB	CountVectorizer	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.96	0.79
LR	CountVectorizer	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.99	0.74
DT	CountVectorizer	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.86	0.64

FIGURE 7 – Sélection de résultats obtenus pour la prédiction en deux classes : Vrai, Faux / Autre, Mixte

Classifieur	Vectoriseur	Pré-traitements	Classes de prédiction	Training accuracy	Testing accuracy
NB	TF-IDF	regex, minusculation, lemmatization	True / False / Other / Mixture	0.97	0.53
NB	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True / False / Other / Mixture	0.97	0.53

FIGURE 8 – Sélection de résultats obtenus pour la prédiction en quatre classes : Vrai / Faux / Autre / Mixte

3.3 Analyse des résultats obtenus

Globalement, on observe que les résultats sont peu satisfaisants dans le sens où les testing accuracies restent encore trop éloignées des training accuracies malgré les pré-traitements et paramètres utilisés. En effet, on souhaiterait dans l'idéal obtenir un classificateur qui a une bonne training accuracy (autour de 0.90), et qui donne une testing accuracy au maximum inférieure de 0.10 par rapport à la training accuracy. Au delà de ça, on parle de surentraînement. Le surentraînement a lieu lorsqu'un algorithme de classification connaît tellement bien les données d'entraînement qu'il n'est pas capable de prédire correctement la classe d'un nouveau document.

En dehors de cela, on peut constater que les algorithmes performent mieux sur deux classes que sur quatre classes. En effet, lorsqu'ils doivent prédire un output parmi plus de deux classes, cela laisse place à plus de sources de biais et d'erreurs. De plus, il semblerait que les classificateurs sélectionnés aient plus de facilité à prédire entre Vrai,Faux / Autre,Mixte plutôt qu'entre Vrai / Faux.

En outre, on observe que le TfidfVectorizer performe légèrement mieux que le CountVectorizer et que les méthodes de ré-échantillonage (i.e. downsampling, upsampling) améliorent les résultats pour les prédictions Vrai / Faux. Par contre, on constate que malgré avoir évalué un nombre conséquent de combinaisons de paramètres pour l'algorithme de Decision Tree (pour la prédiction Vrai / Faux), celui-ci conserve des résultats médiocres.

3.4 Meilleurs résultats

Classifieur	Vectoriseur	Pré-traitements	Classes de prédiction	Training accuracy	Testing accuracy
LR	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True,False / Other, Mixture	0.83	0.84
NB	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization	True / False / Other / Mixture	0.97	0.53
RF	TF-IDF	contractions, regex, minusculation, nombres, stop words, mots indésirables, lemmatization, downsampling, upsampling	True / False	0.96	0.67

FIGURE 9 – Meilleurs résultats obtenus pour les trois types de classification possibles.

	precision	recall	f1-score	support
false	0.66	0.92	0.77	315
true	0.71	0.29	0.41	210
accuracy			0.67	525
macro avg	0.69	0.60	0.59	525
weighted avg	0.68	0.67	0.63	525

FIGURE 10 – Mesures supplémentaires pour le troisième rang du tableau ci-dessus.

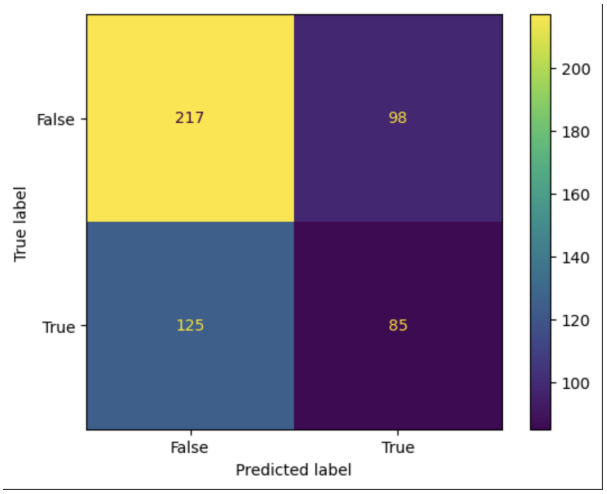

FIGURE 11 – Matrice de confusion pour le troisième rang du tableau ci-dessus.

Sur les deux figures ci-dessus, on observe que l'algorithme Random Forest performe mieux à détecter si un article est faux plutôt que vrai. En effet, on observe sur la matrice de confusion un grand nombre de prédictions réussies lorsqu'il s'agit d'une Fake News. Par contre, quand il s'agit de prédire un article vrai comme 'vrai', l'algorithme performe mal. Cette même observation est confirmée par le rappel (i.e. pourcentage de positifs bien prédits par notre modèle) qui est proche de 1 pour 'false' et relativement encore très bas pour 'true'.

4 Améliorations possibles

Afin de perfectionner nos résultats, plusieurs améliorations auraient été possibles, en voici la liste :

- **Stemming**
  Dans le même thème que la lemmatisation, le stemming nous aurait permis de potentiellement améliorer le pré-traitement des données.
- **Mutual information**
  Nous avons par exemple tenté d'exploiter la Mutual Information des différents tokens afin d'obtenir une valeur déterminant leur maintien ou non dans notre étude. Nous avons alors dans un premier temps implémenté celle-ci à la main, et ce, pour chacune des trois classifications étudiées. Malheureusement, les résultats obtenus pour les différents termes étaient toujours d'ordre 10^{-3} et n'étaient ainsi pas exploitables. Nous avons alors tentés d'utiliser celle proposée par Scikit-Learn mais notre tentative s'est résolue par un échec.
- **Considérer les titres des articles**
  Une éventuelle amélioration aurait été de traiter les titres de la même manière que les articles. Cela aurait permis d'obtenir une information supplémentaire sur les documents traités.
- **Paramétrisation des classificateurs**
  Nous aurions également pu tester un plus grand nombre de combinaisons de paramètres pour nos classificateurs afin de trouver un algorithme donnant des résultats plus concluants.
- **Utilisation d'un neural network**
  Bien sur, l'implémentation d'un neural network aurait pu donner des résultats surprenants à condition d'être paramétré correctement.

5 Conclusion

En conclusion, ce rapport a exploré les méthodes et techniques de machine learning appliquées à la détection de Fake News, et cela en examinant les différentes étapes du processus de modélisation. Ces étapes comprennent le pré-traitement des données, la sélection des caractéristiques, l'entraînement du modèle et l'évaluation des performances.

Grâce à une analyse approfondie, les avantages et les inconvénients des différentes approches de modélisation choisies ont été identifiés et finalement, c'est le classificateur Random Forest qui s'est avéré être le plus convaincant, une fois combiné à toutes les techniques de pré-traitement étudiées dans ce rapport.