UCLouvain
Faculté de philosophie, arts et lettres

Séminaire de linguistique computationnelle

Classification de données sociales

Bianca Ciobanica

LCLIG2140

Université catholique de Louvain

Juin 2024

Table des matières

1 Tâche

Ce travail porte sur la classification de données sociales, en l'occurence de posts issus de la plateforme en ligne Twitter [5]. La tâche est de prédire à quel parti politique politique français, EM ou FN, les tweets sont associés. Pour cela, nous utiliserons le texte publié par les utilisateurs que nous avons prétraité et qui servira pour le modèle de classification binaire à partir des annotations des catégories. Nous avons d'abord implémenté la régression logistique et ensuite entraîné un arbre de décision et un réseau de neurones afin de combiner les résultats à l'aide d'une approche par ensemble. Chaque modèle a été évalué à l'aide d'une validation croisée (*k-fold cross validation*) pour la sélection des hyperparamètres. Enfin, nous procédons à un vote en prenant la moyenne de la probabilité émise de la classe pour la prédiction.

1.1 Les partis politiques

L'EM (En Marche) est l'appelation pour la Renaissance, un parti politique français lancé en 2016 par le président de la république française Emmanuel Macron.

Le FN (Front National jusqu'en 2018) désigne le Rassemblement National, un parti politique français d'extrême droite présidé par Marine Le Pen au moment de la publication des tweets dans notre corpus.

1.2 Twitter

Racheté par la société X Corp. et actuellement nommé X, Twitter était un réseau social fondé en 2006. La plateforme comptait plus de 300 millions d'utilisateurs en 2017, dont environ plus d'1 million provenant de France [14]. Les publications des utilisateurs sont appelées des *tweets* et contiennent un langage spécifique comme des mentions d'autres utilisateurs à l'aide su symbole `@`, l'utilisation du mot *rt* (*retweet*) pour désigner indiquer le partage d'une publication d'un tiers, et enfin l'emploi de mots-dièse qui sont des mots (abbrégés, collés ou non) précédés de # afin de référencer à des sujets tendance qui synthétisent le contenu de la publication. Nous conserverons le nom Twitter pour ce travail.

2 Description du corpus

Le corpus est composé de 14 107 *tweets* chacun comprenant les informations suivantes :

- `tweet_id`, l'identifiant du *tweet*
- `party`, le label associé au *tweet* à savoir EM ou FN
- `user_location`, la localisation de l'utilisateur
- `created_at`, la date de publication du *tweet*
- `text`, le texte publié par l'utilisateur

À notre surprise, il n'y a pas de déséquilibre dans la distribution des classes des partis politiques au sein du corpus. Nous avons également vérifié dans le tableau ci-dessous 1 si l'équilibre était maintenu lors de la création du jeu d'entraînement et d'évaluation (également appelés *train* et *test set*) pour lesquels nous avons réservé 70 % des données pour l'entraînement et le reste pour l'évaluation. Puisque nous évaluons les performances des modèles au moyen d'une validation croisée lors de la sélection des hyperparamètres, nous n'avons pas créé un jeu de validation fixe.

Partition	**Proportions**	
	EM	**FN**
Corpus	50.5%	49.5%
Train	51%	49.1%
Test	50%	50%

Table 1: Proportions des classes EM et FN dans les partitions

2.1 Distribution et nature des textes

La nature des textes est très particulière en raison du contexte de notre étude (voir 1.2). Trois types de mots ont été identifiés (sans compter les émoticônes) : les mots-dièse, les mentions et les autres mots standards, dont les proportions sont regroupées dans le tableau 2.

Type de mot	**Corpus**	**Vocabulaire**
Autres	81%	63%
Mentions	10%	23%
Mot-dièse	9%	14%

Table 2: Proportions des types de mots dans le corpus et au sein du vocabulaire (mots uniques)

En examinant de plus près la composition des mentions et des hashtags à l'aide des graphiques 1 et 2, nous constatons que ces mots fournissent un contenu sémantique fortement lié aux partis politiques comme *#enmarche* ou *@jeunesmacron*. Il est donc important de les conserver lors de la prédiction malgré leur morphologie particulière. La mention la plus fréquente est *@emmanuelmacron* (6 %) et le mot-dièse le plus employé est *#macron* (14 %).

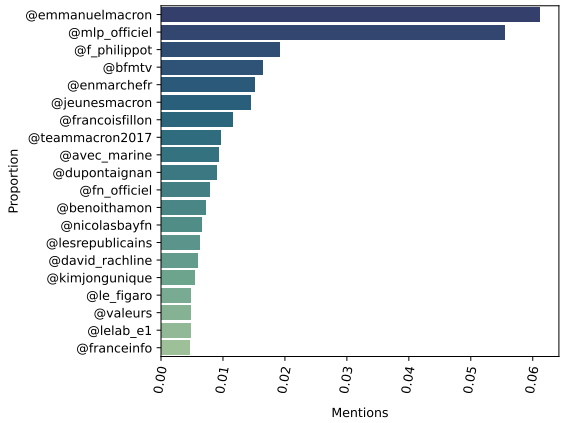

Figure 1: 20 mentions les plus présentes parmi toutes les mentions au sein du corpus

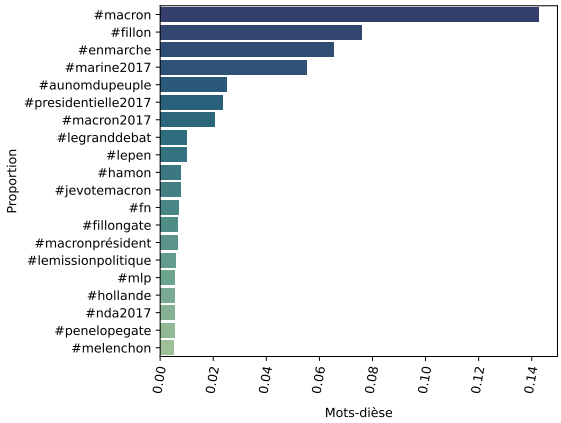

Figure 2: 20 mots-dièse les plus présents parmi tous les mots-dièse au sein du corpus

La distribution des mots au sein du corpus dans le graphique ci-dessous 3 semble suivre la loi de Zipf. Au total, nous avons 173 641 mots dont 20 485 mots uniques.

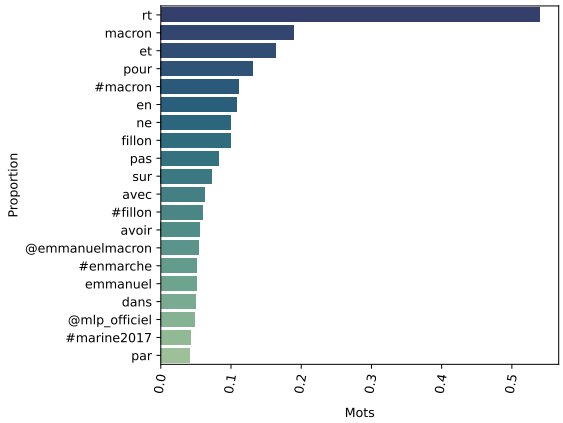

Figure 3: 20 mots les plus présents au sein du corpus

La loi de Zipf démontre que la Fréquence(mot) $\propto \frac{1}{\text{rang(mot)}}$. En d'autres termes, la fréquence d'une observation x est inversement proportionnelle à son rang [2]. Généralement, les mots les plus fréquents sont des mots grammaticaux ainsi que des mots appartenant au domaine du corpus. Dans notre corpus, le terme le plus employé est *rt*; il est présent dans plus de la moitié du corpus en raison du contexte de communication 1.2. La queue de la distribution présente donc beaucoup de mots rares; dans notre cas 55 % n'apparaissent qu'une seule fois (principalement des mentions d'utilisateurs). Nous avons décidé d'exclure ces mots rares lors de l'entraînement, car une occurence n'est pas suffisante pour parvenir à discriminer lors de la prédiction.

Si nous regardons la distribution des longueurs des *tweets* (en nombre de caractères) dans le graphique en 4, nous pouvons voir que les distributions des deux partitions (*split*), train ($\bar{x} = 131,\ sd = 23$) et test ($\bar{x} = 131,\ sd = 23$) présentent les mêmes caractéristiques. Le test de Kolmogorov-Smirnov ($p = 0.67$) nous confirme que ces distributions sont fortement similaires.

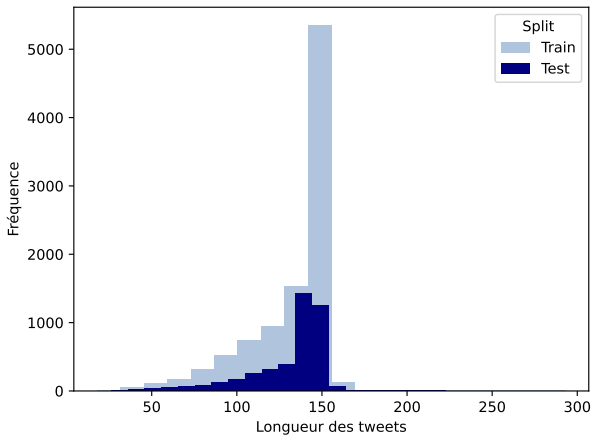

Figure 4: Longueur des *tweets* du jeu d'entraînement et du jeu de test

La vérification des distributions des jeux de données utilisés pour l'entraînement et l'évaluation des modèles est une étape non-néglideable afin d'éviter tout déséquilibre, car ceci peut engendrer des prédictions biaisées [8]. Il convient également de souligner que les données textuelles provenant des réseaux sociaux sont généralement très bruitées comparés aux textes conventionnels [4], [7], [9], [16]. Par exemple, le contenu textuel des *tweets* dont nous disposons comprend souvent des symboles spéciaux tels que des émoticônes, des abréviations, des fautes de langue ou frappe, du sarcasme, voire même du contenu multilingu en raison du contexte de communication. Ceci pose un défi majeur lors de la phase de prétraitement, car la structure de ces textes nécessite l'application de techniques spécifiques de nettoyage et de normalisation pour assurer la qualité et la cohérence des données utilisées par les modèles [7]. Toutefois, [3] ont démontré que le bruit peut être réduit notamment à l'aide de techniques automatiques lors du prétraitement, et c'est ce que nous avons tenté en partie. Un autre problème que nous avons identifié c'est que les messages sont tronqués (*#je...*), donc nous manquons de l'information pour certains. Cela peut être dû à l'extraction automatique des messages lors de la constitution du corpus et reste un problème auquel nous ne pouvons pas apporter de solution. Nous avons omis ces monts tronqués.

3 Méthodologie

Dans cette section, nous allons décrire les étapes réalisées pour prétraiter le texte, entaîner les modèles et les évaluer.

3.1 Prétraitement

Chaque *tweet* est d'abord tokenisé, converti en minuscule, et ensuite lemmatisé à l'aide de la librairie *spacy*. Nous conservons les caractères `@` et `#` des tokens, car comme nous l'avons expliqué dans la section 2.1, ces mots peuvent être porteurs de sens. Afin de ne pas encombrer le modèle de termes peu pertinents, certaines classes grammaticales ont été ignorées telles que les pronoms, auxiliaires, déterminants, ponctuation, espaces et autres symboles, toujours à l'aide de *spacy*. Nous n'avons pas voulu nous fier à liste prédéfinie de *stopwords*, par peur que la liste ne soit pas adaptée à notre et à nos données. Enfin, les messages sont extraits à partir d'un contenu `html` et contenaient dès lors des symboles encodés comme `>`, `&` ou `<` peu pertinents dans les textes que nous avons dû ignorer.

Bien qu'une amélioration possible consisterait à normaliser les mots, nous avons décidé de ne pas la mettre en œuvre pour ce travail, cette technique étant couteuse en termes de temps et nécessitant une analyse qualitative plus approfondie.

3.2 Encodage numérique des textes

La modélisation d'une tâche consiste à estimer une fonction f qui produit un résultat (l'*output*) $\hat{y}$ à partir d'éléments fournis en entrée x_n (l'*input*) [8]. Ceci peut être modélisé comme $\hat{c} = f(x)$

pour notre tâche où $\hat{c}$ désigne la classe prédite. Dans notre cas, nous voulons à partir d'un *tweet* donné en entrée obtenir la catégorie c de celui-ci ($c \in C$ où C désigne le nombre de catégories). Dans notre cas, nous avons affaire à une classification binaire avec $C \in ('EM', 'FN')$.

Pour ce faire, nous avons besoin de convertir les textes en une représentation numérique afin de les utiliser comme entrée pour notre modèle [10]. Ces représentations sont sous forme de vecteurs générés à l'aide de différentes techniques. Dans ce travail nous choisi l'approche simple par sac-de-mots qui consiste à encoder la fréquence des mots pour chaque *tweet*. Nous avons également tenté de créer des vecteurs à l'aide de tfidf, mais ceux-ci se sont avérés moins efficaces et une raison à cela est que le tfidf nécéssite de beaucoup de données afin de pouvoir offrir une généralisation fiable. L'output du modèle étant un nombre réel, il a également fallu encoder les catégories en attribuant 1 pour EM (la classe positive) et 0 pour FN (la classe négative).

4 Choix du modèle pour l'expérience 2

Pour cette expérience, nous avons choisi la régression logistique qui est un modèle linéaire simple et efficace utilisé pour la classification binaire. Son principe repose sur l'hypothèse qu'il existe une relation linéaire entre une variable nominale, généralement notée y et représentant la classe à prédire, et une variable indépendante, souvent un ensemble de caractéristiques extraites à partir des données (*features*) [10]. Dans notre cas, les *tweets* vont représenter les features pour lesquels nous possédons 11 227 mots uniques (les mots dont $Freq(x) = 1$ ont été exclus ici). Concrètement, la régression logistique estime cette relation en utilisant une fonction spécifique appelée la fonction sigmoïde, définie comme $y = \frac{1}{1+e^{-x}}$, où y est contraint à l'intervalle $[0, 1]$ [10]. Dans un contexte de classification binaire, ces probabilités sont interprétées comme la confiance du modèle dans la prédiction de la classe positive 1. Si $P(y = 1|x)$ est supérieure à un seuil déterminé (par exemple, 0.5), le modèle prédit la classe 1, sinon il prédit la classe 0.

Il va de soi de vérifier l'hypothèse de la linéarité pour déterminer si le modèle est adéquat à nos données. Une approche consiste à examiner visuellement les quantiles des résidus de notre modèle par rapport aux quantiles théoriques d'une distribution normale (représentés par la ligne rouge dans 5) qui permet d'évaluer l'écart des prédictions par rapport aux observations réelles [8]. Nous constatons en 5 que la plupart des résidus suivent cette distribution, ce qui indique que notre modèle semble capturer correctement la relation entre la classe à prédire et les mots associés. Le graphique révèle toutefois des hauts écarts par rapport à la ligne rouge dans la queue de la distribution des résidus ce qui signifie que le modèle surestime ou sous-estime les prédictions par rapport à ce qui est attendu. Une stratégie envisagée pour atténuer cet effet consiste à normaliser les vecteurs, une piste que nous envisagerons pour l'avenir [8].

4.1 Avantages et inconvéntients

Un avantage majeur de la régression logistique réside dans sa nature facilement interprétable contrairement à un modèle de type *black box* tels que les réseaux de neurones ou les méthodes

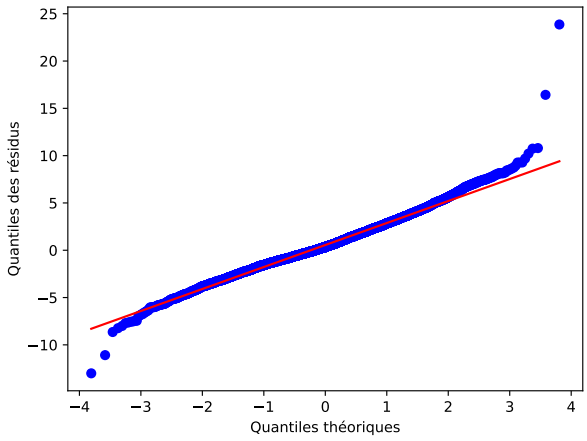

Figure 5: Graphique Q-Q des quantiles des résidus et des quantiles théoriques suivant une distribution normale

par ensemble [8]. Le modèle permet notamment de quantifier l'effet d'une variable indépendante (une feature) sur la prédiction de la classe, ce qui facilite la compréhension de l'impact de chaque feature sur le modèle. Le tableau 3 ci-dessous présente les 10 features les plus importantes pour chaque catégorie. Un coefficient positif signifie que la présence de la feature augmente la probabilité que l'observation soit classée dans la classe de référence 1 (la classe positive), tandis qu'un coefficient négatif signifie que la présence de la feature diminue cette probabilité. Il n'est pas surprenant de voir que les mentions et mots-dièse occupent le top 10, tous représentant des personnes bénéficiant d'une forte visibilité comme @mlp_officiel (la page Twitter officielle de Marine Le Pen) ou des mouvements politiques (comme @jeunesmacron, un mouvement de jeunesse du parti Renaissance). Ceci suggère donc que ces features jouent un rôle significatif dans la classification.

Classe 1 (EM)	**Coéfficient**	**Classe 0 (FN)**	**Coéfficient**
@jeunesmacron	2.829680	@avec_marine	-3.093617
@teammacron2017	2.594312	@f_philippot	-2.875320
@enmarcheparis20	2.235605	#marine2017	-2.813752
@bressonmike	1.936972	@kimjongunique	-2.738316
@alni92	1.825459	@nicolasbayfn	-2.696826
@emmanuelmacron	1.797280	@valeurs	-2.513365
@daarjeeling	1.730277	@mlp_officiel	-2.495346
@onzeleft	1.725699	@dupontaignan	-2.484577
@ccastaner	1.720781	@gilbertcollard	-2.425171
#enmarche	1.703647	@jeanmessiha	-2.366708

Table 3: Les features les plus importantes pour prédire chaque parti politique

Un désavantage de ce modèle est qu'il assume une relation linéaire entre x et y, ce qui peut ne pas toujours être le cas lorsque les classes ne sont pas linéairement séparables. Par conséquent, si la condition de co-linéarité n'est remplie pas, le modèle peut sous-estimer ou surestimer les effets des variables indépendantes sur la variable dépendante et produire des prédictions moins

précises et des interprétations erronées des résultats [8]. De plus, c'est un modèle également très sensible aux valeurs abbérantes, car en présence de celles-ci, le dernier va essayer va produire des estimations en incluant ces observations extrêmes et dégrader la performance prédictive du modèle [8].

4.2 Sélection d'hyperparamètres et évaluation

Nous avons initialisé le modèle avec diverses combinaisons de hyperparamètres (valeur de C, régularisation $L1$ ou $L2$, nombre d'itérations) et évalué à l'aide de la validation croisée sur 5 partitions aléatoires. L'avantage de cette méthode, malgré qu'elle soit couteuse en termes de temps et de ressources, est que nous renforcons la capacité du modèle à généraliser sur des données inconnues en diviseant le jeu d'entraînement en k échantillons aléatoires (au lieu d'un échantillon fixe) et donc d'éviter le surapprentissage des données d'entraînement [8]. Ces partitions serviront pour évaluer la performance du modèle avec les hyperparamètres sélectionnés. L'ajout de la régularisation ainsi que l'hyperparamètre C sont une manière de contrôler ce risque de surapprentissage. La régularisation consiste à pénaliser les poids de trop grande taille à l'aide de l'hyperparamètre λ. Il existe deux méthodes de régularisation : $L1$ (également appelée *Lasso*) pénalisant sur $\lambda \sum_{i=1}^{10}, |w_i|$, la somme de la valeur absolue des poids, et $L2$ (également appelée *Ridge*) pénalisant sur $\lambda \sum_{i=1}^{10} w_i^2$, la somme des carrés [10], [8]. L'hyperparamètre C quant à lui contrôle la force de la pénalité appliquée aux coefficients du modèle de régression logistique. Une valeur plus élevée de C correspond à une régularisation plus faible, permettant aux coefficients de s'adapter plus étroitement aux données d'entraînement, tandis qu'une valeur plus faible de C entraîne une régularisation plus forte, restreignant les coefficients à des valeurs plus petites [8]. Pour notre modèle, un $C = 0.6$ donnait les meilleures performances. Au delà de cette valeur, nous encourons un risque de surapprentissage qui peut être visualisé en 6. Nous pouvons voir que l'exactitude sur le train set augmente jusqu'à atteindre un score de 1 alors que celle sur le test set diminue drastiquement à force que C augmente.

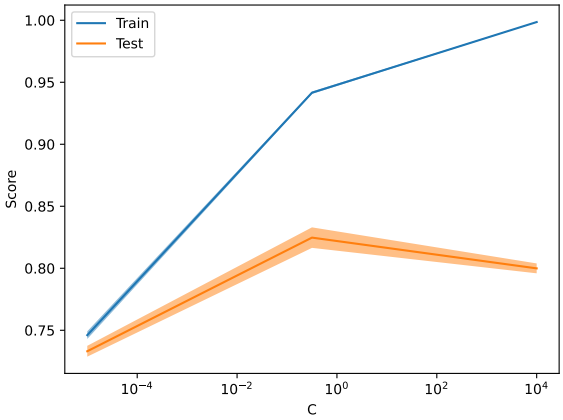

Figure 6: Évolution de la performance du modèle sur le train et test set en fonction du choix de l'hyperparamètre C

Pour notre modèle, la régularisation $L2$ a permis de générer un score de 83 % lors de la validation. Une fois les paramètres optimaux sélectionnés, nous avons testé ce modèle sur le jeu d'évaluation et avons obenu une précision légèrement améliorée de 85 %. La régularisation permet donc de mieux généraliser sur des nouvelles données en pénalisant la précision durant l'entraînement Le graphique ci-dessous en 7 montre en outre une capacité discriminatoire de 92 % de notre modèle.

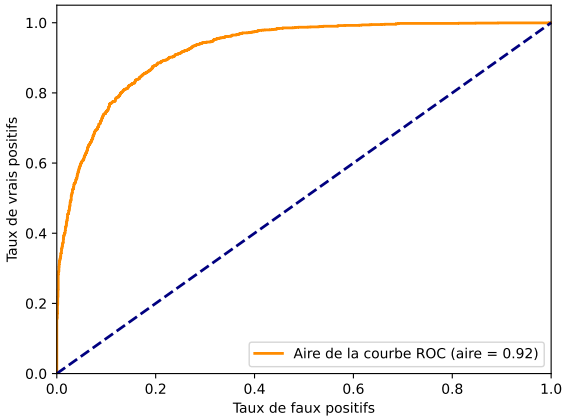

Figure 7: Courbe ROC (Receiver Operating Characteristic) montrant le taux de vrais positifs en fonction du taux de faux positifs

Toutefois, si nous regardons la tendance de la courbe d'apprentisage en 8, nous pouvons voir que le modèle a tout de même une légère tendance à surapprendre les données d'entraînement notamment dû au grand écart entre la courbe de la performance lors de l'entraînement et celle lors de l'évaluation. De plus, les deux courbes ne se stabilisent pas tout à fait à mesure que le nombre de données utlisées pour l'entraînement augmente et cela suggère que l'ajout de nouvelles données pourrait davantage améliorer la performance du modèle.

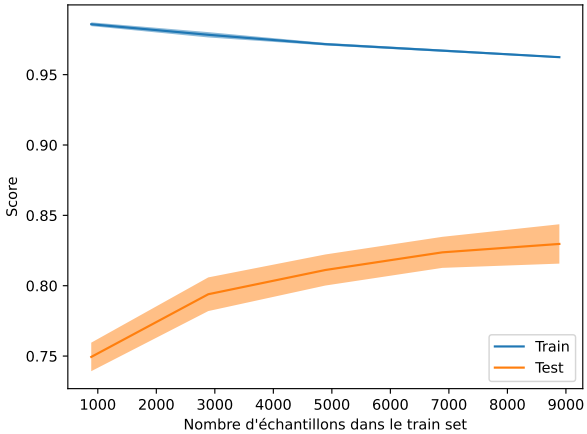

Figure 8: Courbe d'apprentissage du train et test set en fonction de la taille de l'échantillon

Nous avons tenté d'enlever les outliers identifiés en 5 en appliquant l'algorithme *Local Outlier*

Factor, mais nous n'avons pas observé une amélioration pour la performance. Une approche moins brute que la nôtre pour traiter les valeurs aberrantes pourrait accroître l'efficacité. Néanmoins, cela exigerait une analyse plus approfondie des variables contribuant aux écarts importants et cela reste également une piste pour l'avenir.

5 Méthode par ensemble

Une approche intéressante est de combiner les résultats de plusieurs modèles au lieu d'avoir un seul modèle comme référence; ce sont les méthodes par ensemble. L'idée est de combiner n classifieurs à partir desquels le résultat final sera choisi en fonction d'un critère de sélection : le vote par majorité ou en prenant la moyenne des probabilités émises par le modèle pour la classe prédite [8]. Nous avons pour cette expérience entraîné deux réseaux de neurones et un abre de décision.

5.1 Le réseau de neurones

Pour l'approche par réseau de neurones, nous avons entraîné un modèle avec des vecteurs sac-de-mots et une seconde version avec des embeddings d'un modèle CamemBERT [12], spécifiquement entraîné sur un corpus de tweets français [6]. Contrairement à l'approche sac-de-mots, l'intérêt d'utiliser un modèle BERT pour l'encodage des textes réside dans sa capacité à fournir une représentation contextualisée des mots avec 768 dimensions. [12]. Pour les deux, les hyperparmamètres optimaux sont la sigmoïde comme fonction d'activation, l'algorithme d'optimisation Adam [11], une seule couche cachée et trois neurones pour cette couche (une couche cachée serait capable d'estimer presque n'importe quelle fonction [10]) et un λ de 0.1 pour la régularisation $L2$.

5.2 L'arbre de décision

Nous avons également implémenté un arbre de décision pour lequel nous avons identifié les hyperparamètres optimaux suivants : un taux de réduction de complexité de 0.03 pour le post-pruning, une méthode de sélection de la caractéristique basée sur l'entropie, une limite de profondeur de l'arbre fixée à 3 pour éviter le surapprentissage, un nombre minimal d'échantillons par feuille de 2 pour garantir la robustesse des feuilles, et un nombre minimal d'échantillons pour une division de 2.

5.3 Performances de chaque modèle

Le tableau 4 regroupe les résultats pour chaque modèle. Les arbres de décision, avec une exactitude de 0.60 en moyenne, montrent une forte précision pour une classe (EM) mais un rappel très faible, et vice versa pour l'autre classe (FN). Cela démontre une faible capacité à généraliser en raison d'une suradaptation aux données d'entraînement; une faiblesse majeure des ces modèles [8]. Pour répondre à ce problème, nous pourrions essayer des méthodes non

Modèle	Précision	Rappel	F1-Score	Exactitude
Régression Logistique				0.84
EM	0.85	0.83	0.84	
FN	0.83	0.85	0.84	
Moyenne	0.84	0.84	0.84	
Arbres de Décision				0.60
EM	0.96	0.24	0.38	
FN	0.56	0.99	0.72	
Moyenne	0.76	0.61	0.55	
MLP				0.84
EM	0.85	0.82	0.83	
FN	0.82	0.85	0.84	
Moyenne	0.84	0.84	0.84	
MLP + BERT				0.79
EM	0.81	0.76	0.78	
FN	0.77	0.82	0.79	
Moyenne	0.79	0.79	0.79	

Table 4: Scores des modèles

supervisées comme le gradient boosting, qui combine plusieurs arbres de décision pour réduire les erreurs en vue d'améliorer la robustesse [8]. L'intégration des embeddings de BERT avec MLP n'a pas apporté de meilleurs résultats, probablement parce que le MLP n'est pas optimal pour traiter des vecteurs de représentation aussi denses que ceux produits par BERT. Enfin, la régression logistique et le MLP affichent des performances similaires avec une exactitude de 0.84 avec des scores équilibrés pour la précision et le rappel. Cependant, nous privilégions la régression logistique comme choix final en raison de ses résultats facilement interprétables et à son entraînement nécessitant moins de ressources.

5.4 Résultat final par ensemble

Voici les résultats finaux une fois le vote établi à partir des prédictions de l'ensemble des modèles (nous n'avons pas inclu le modèle réseau de neurones entraînés avec les embeddings de BERT étant donné que ses performances étaient faibles). Nous avons procédé à un *soft* vote qui est la probabilité moyenne pondéré. Le *soft* vote présentait un gain de 1 % en exactitude comparé au vote par majorité. Dans l'ensemble, un gain en précision par classe est légèrement observable.

Classe	Précision	Rappel	F1-score
EM	0.89	0.82	0.85
FN	0.83	0.89	0.86
Exactitude			0.86

Table 5: Performances à l'aide du *soft* vote pour la méthode par ensemble

6 Limites et pistes d'améliorations

Nous sommes conscients que la performance peut être améliorée notamment en réduistant le bruit présent dans les données. Cependant, la nature des données rend le prétraitement difficile et nécessiterait une analyse qualitative plus approfondie afin d'établir une stratégie efficace pour la normalisation. Nous étions également limités en termes de ressources de calcul pour l'entraînement, ce qui a restreint notre capacité à explorer des modèles plus complexes ou à effectuer des recherches approfondies dans l'optimisation des hyperparamètres.

Une autre manière d'améliorer les performances consisterait à créer de nouvelles variables qui pourraient contribuer à la prédiction. Par exemple, nous aurions pu prendre en compte la localisation de l'utilisateur à partir de la variable `user_location`. Cependant, ces données nécéssitent également une normalisation en raison de leur format hétérogène. Les données sur la localisation étaient toutefois manquantes la plupart du temps, ce qui aurait pu poser problème pour l'équilibre des catégories. Une solution à cela serait de s'appuyer sur le texte pour extraire la localisation automatiquement, comme cela a été illustré dans [15]. En revanche, nous pourrions procéder à une sélection de variables ce qui permetterait de réduire le nombre de dimensions en vue de réduire le bruit, la complexité du domaine et les ressources pour l'entraînement.

L'approche par clustering pourrait également enrichir les données et révéler les intentions et comportements des utilisateurs en identifiant leur réseau social à partir de leurs followers (si nous disposions de ces donées), mentions et hashtags.

Enfin, il n'y a pas de meilleur modèle étant donné que tous reponsent sur des estimations, mais plutôt un modèle avec des bonnes performances. Une alternative pour ce projet aurait été d'explorer des approches non supervisées, telles que les méthodes par ensemble. Ces dernières, à l'instar du gradient boosting, présentent l'avantage de combiner les atouts de chaque modèle pour compenser leurs lacunes, sans nécessiter des conditions préalables sur la distribution des données. De plus, ils sont particulièrement adaptées aux jeux de données présentant de nombreuses dimensions comme dans notre cas grâce à leur sélection efficace des variables. Ces modèles sont néanmoins peu interprétables en raison de leur algorithme complexe, mais offrent des résultats robustes et précis. Ainsi, lors de la modélisation, nous sommes souvent confrontés à un compromis entre la précision des prédictions et la facilité à comprendre et à expliquer le modèle, comme nous l'avons fait en optant pour la régression logistique. Le choix entre l'interprétabilité et la précision dépend surtout de la nature de la tâche en question et des besoins de l'utilisateur final. Par exemple dans le domaine médical, il est primordial de favoriser des prédictions à haute précision afin d'éviter des complications graves si l'on ne parvient pas à identifier la maladie [1]. En revanche, dans un cadre entrepreneurial, l'interprétabilité des modèles de classification pourrait être préférable au détriment d'une légère baisse de précision, étant donné son rôle crucial dans le processus de prise de décision [13].

7 Remerciements

Je souhaite vous remercier (et les autres membres du Cental) pour votre dévouement continu à nous fournir des cours de haute qualité. Votre disponibilté et gentillesse a toujours été appréciable.

References

[1] Ahsan, M., Luna, S., and Siddique, Z. (2022). Machine-learning-based disease diagnosis: A comprehensive review. *Healthcare (Basel)*, 10(3):541.

[2] Aitchison, L. et al. (2016). Zipf's law arises naturally when there are underlying, unobserved variables. *PLoS Computational Biology*, 12(12):e1005110.

[3] Baldwin, T., Cook, P., Lui, M., MacKinlay, A., and Wang, L. (2013). How noisy social media text, how diffrnt social media sources? In Mitkov, R. and Park, J. C., editors, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan. Asian Federation of Natural Language Processing.

[4] Eisenstein, J. (2013). What to do about bad language on the internet. In Vanderwende, L., Daumé III, H., and Kirchhoff, K., editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

[5] Fraisier, O., Cabanac, G., Pitarch, Y., Besançon, R., and Boughanem, M. (2018). #élysée2017fr: The 2017 french presidential campaign on twitter. In *International Conference on Web and Social Media*.

[6] Guo, Y. et al. (2021). BERTweetFR : Domain adaptation of pre-trained language models for French tweets. In Xu, W., Ritter, A., Baldwin, T., and Rahimi, A., editors, *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 445–450, Online. Association for Computational Linguistics.

[7] Han, B. and Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a #twitter. In Lin, D., Matsumoto, Y., and Mihalcea, R., editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378, Portland, Oregon, USA. Association for Computational Linguistics.

[8] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[9] Java, A. (2007). A Framework for Modeling Influence, Opinions and Structure in Social Media. In *22nd Conference on Artificial Intelligence (AAAI 2007)*.

[10] Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing (3rd ed. draft)*. Version préliminaire sur http://web.stanford.edu/~jurafsky/slp3/.

[11] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[12] Martin, L. et al. (2019). Camembert: a tasty french language model. *CoRR*, abs/1911.03894.

[13] Prabadevi, B., Shalini, R., and Kavitha, B. (2023). Customer churning analysis using machine learning algorithms. *International Journal of Intelligent Networks*, 4:145–154.

[14] Wikipedia contributors (2024). Twitter. https://fr.wikipedia.org/wiki/Twitter#. Consulté le 28 mai 2024.

[15] Wing, B. and Baldridge, J. (2011). Simple supervised document geolocation with geodesic grids. In *Annual Meeting of the Association for Computational Linguistics*.

[16] Yin, J. et al. (2012). Using social media to enhance emergency situation awareness. *Intelligent Systems, IEEE*, 27:52–59.