École nationale de la statistique
et de l'analyse de l'information

Projet de fin d'études

Étudiants 3A

Classification des effets secondaires des médicaments

Réalisé par :

Léonie Courcoul

Audrey D'Abrigeon

Sarah Leroy

Valentin Redon

Sous la direction de :

Hanane Omichessan

Blaise Mbunga Mputu

2020-2021

Remerciements

En premier lieu, nous tenons à remercier nos tuteurs, Hanane Omichessan et Blaise Mbunga Mputu, pour leur encadrement mais aussi pour le partage de leurs connaissances. Ils nous ont permis de comprendre clairement les attentes du sujet et ont su répondre à nos questions. Ils ont également été très présents tout au long du projet, malgré la situation actuelle dans laquelle nous vivons, que ce soit lors des séances de suivi ou encore à distance.

Ensuite, nous profitons également de cette occasion pour remercier les professeurs de l'ENSAI pour nous avoir fourni les outils nécessaires au bon déroulement de notre projet.

Enfin, nous remercions toutes les personnes qui ont apporté leur soutien, de près ou de loin, à ce projet, y compris à la rédaction de ce rapport.

Abstract

After receiving market authorization, drugs are actively monitored by pharmacovigilance teams to track the various side effects experienced by patients. Today, with the development of social networks, more and more consumers are expressing their experiences directly on social networks, particularly on Twitter. It is therefore becoming essential for pharmacovigilance teams to be able to analyze and retrieve, using digital tools, the side effects expressed in tweets.

A first part of this study aims to determine the presence of a side effect in a tweet. First, we created a database of pre-labeled tweets. Then, we performed natural language processing on the database to pre-process, clean and extract features from the tweets. In order to predict whether one contains a side effect or not, we used machine learning algorithms after features extraction. The comparison of different algorithms showed that Random Forest and XGBoost give the best results for a Word2Vec vectorization on a balanced training database. Additionally, we performed a step of tuning and cross-validation to improve the performance of our models.

The second part of our study imply categorization of tweets presenting a side effect according to its class thanks to the use of a dictionary.

Overall, we implemented a global model that aims to obtain a categorization of a certain number of new tweets containing the name of a drug previously filled in.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AdaB	AdaBoost
ADAM	Adaptive Moment estimation
ADR	Adverse Drug Reaction
AMM	Autorisation de Mise sur le Marché
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CART	Classification And Regression Trees
CBOW	Continuous Bag of Words
GB	Gradient Boosting
IA	Intelligence Artificielle
KNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LR	Logistic Regression
NB	Naive Bayes
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
RF	Random Forest
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TPE	Tree Parzen Estimator
XGB (XGBoost)	Extreme Gradient Boosting

Introduction

Les trois premières phases d'un essai clinique réalisées sur un nouveau médicament permettent de mesurer sa toxicité et son efficacité. Si les résultats de ces phases sont concluants, le médicament obtient son autorisation de mise sur le marché (AMM). Le médicament est alors utilisé à grande échelle et des effets secondaires non identifiés lors des essais cliniques peuvent survenir en considérant des effectifs plus importants (ex. 1% de la population française). Dès lors, les équipes de pharmacovigilance (des laboratoires à l'origine du développement du médicament mais également celles des autorités sanitaires) analysent les retours d'expérience des patients qui sont, soit remontés par les professionnels (médecins ou pharmaciens), soit, de plus en plus souvent, directement décrits via les réseaux sociaux.

Dans un monde de plus en plus numérique, les patients ont tendance à exprimer leurs retours d'expérience sur les médicaments, et notamment l'apparition d'effets indésirables, directement sur les réseaux sociaux (Twitter, Facebook, etc.). Il devient donc primordial pour les équipes de pharmacovigilance d'être capable de répertorier et d'analyser les effets secondaires des médicaments exprimés sur Twitter, au risque sinon de passer à côté de certains effets indésirables et de voir leur AMM retirée s'ils passaient à côté d'enjeux sanitaires majeurs.

Notre projet s'inscrit dans cet objectif. Dans un premier temps, il s'agit de réaliser une classification binaire afin de détecter la présence ou non d'un effet secondaire dans un tweet contenant le nom d'un médicament à surveiller. Ensuite, une analyse des tweets pour lesquels la présence d'un effet secondaire est détectée sera réalisée en utilisant une base de données d'effets secondaires, avec pour objectif de récupérer l'effet secondaire mentionné et l'associer à une catégorie plus générale.

Ce rapport présentera d'abord les bases de données utilisées, puis les étapes de traitement du langage mises en place pour nettoyer et traiter ces données, avant de décrire les différentes méthodes de classification binaire mises en oeuvre pour identifier les tweets présentant un effet secondaire et les résultats obtenus. Enfin, une dernière partie sera consacrée à la catégorisation des effets indésirables présent dans ces derniers.

1 Description de la base de données

Afin d'être en mesure de construire un modèle de classification binaire permettant de déterminer si un tweet relate ou non un effet secondaire, il est nécessaire de disposer de données, ici des tweets labellisés contenant au moins un nom de médicament. Nous avons décidé de traiter exclusivement des tweets écrits en anglais car il s'agit de la langue la plus parlée dans le monde et également la plus utilisée sur Twitter.

1.1 Première approche : extraction de tweets à partir d'une liste de médicaments

La première approche envisagée pour collecter des tweets est de les récupérer à partir d'une liste de médicaments établie à partir de nos connaissances et de plusieurs articles de pharmacovigilance ([4], [7]).

Il est possible d'extraire des tweets à partir de l'API (Application Programming Interface) fournie par Twitter ([12]), à condition de posséder un compte développeur. Nous avons choisi d'utiliser Python et la librairie Tweepy ([9]), qui permet d'interroger de façon très simple l'API Twitter. Après s'être authentifié, la fonction Cursor permet de récupérer des tweeets à partir d'une requête pouvant contenir des mots clés et différents filtres. Il est possible de préciser en arguments de la fonction Cursor la langue, la date depuis laquelle les tweets sont récupérés (01/01/2018 ici) et le nombre de tweets récupérés.

Disposer de tweets contenant des noms de médicaments ne suffit pas pour construire un modèle de classification binaire puisqu'il faut que ces derniers soient annotés afin de savoir s'ils mentionnent explicitement un effet secondaire ou non. Une première possibilité est d'annoter les tweets manuellement mais cela demande beaucoup de temps et dans le cas présent, une certaine expertise concernant à la fois les médicaments (le cadre de la prescription par exemple) et les effets secondaires associés. Cette piste a donc été écartée car elle ne semblait ni pertinente vis-à-vis de la masse de données générées sur les réseaux, ni adaptée dans le cadre de ce projet. Une autre possibilité consiste à utiliser un dictionnaire regroupant tous les mots relatifs à un effet secondaire (par exemple "side effect") mais nous n'avons pas trouvé de telle base de donnée, et il aurait été compliqué de la construire nous-mêmes. De plus, cette méthode n'est probablement pas efficace pour des tweets car ce sont des écrits moins formels que les rapports médicaux ou les transcriptions associées. Une dernière possibilité consiste à utiliser un modèle de reconnaissance d'entités nommées en rapport avec la pharmacovigilance mais, là encore, la littérature est quasi inexistante sur ce sujet. Nous avons donc recherché s'il existait des données déjà labellisées. Cette première étape nous a néanmoins permis de nous familiariser avec le web scrapping en utilisant l'API Twitter et d'envisager différentes pistes pour annoter les tweets même si nous ne les avons pas mises en oeuvre.

1.2 Seconde approche : récupération de tweets pré-labellisés

Nous avons identifié deux corpus de tweets pour la pharmacovigilance disponibles en ligne.

1.2.1 Premier corpus : Twimed

Le premier, TwiMed ([1] dont les données sont disponibles ici [10]), est issu d'une étude dont le but était de fournir des corpus de textes comparables issus de Twitter et de PubMed (une base de données bibliographiques en ligne d'articles scientifiques, qui sert de référence dans le domaine de la recherche en médecine et en biologie). Ce corpus contient 1000 tweets et 1000 phrases issues de PubMed mais nous ne nous intéresserons ici uniquement aux tweets.

Ce corpus a été annoté par deux pharmaciens utilisant les mêmes directives. Deux types d'entités sont annotées : les médicaments et les maladies ou symptômes. Deux types de relations sont annotées entre ces entités : bénéfice ou effet indésirable. Des attributs supplémentaires concernant les entités peuvent être annotés comme la sévérité ou la polarité.

Chaque tweet du corpus est associé à un fichier texte contenant les annotations le concernant. Nous avons donc écrit un script Python qui récupère les identifiants des tweets ainsi que les annotations binaires, obtenues en regardant si les annotations de chaque tweet mentionnent ou non un effet secondaire.

1.2.2 Second corpus : Twitter Annotated Corpus

Le deuxième corpus, Twitter Annotated Corpus ([11]), développé dans le cadre d'une compétition, est divisé en deux parties :

1. Pour la première ([8]), les annotations sont binaires et indiquent si un tweet mentionne un effet secondaire ou non. Les données de cette partie ont été récupérées à partir d'un unique fichier texte contenant pour chaque tweet, son identifiant et l'annotation binaire associée (1 si le tweet mentionne un effet secondaire, 0 si ce n'est pas le cas). Cette partie du corpus contient 7574 tweets annotés par deux experts dans ce domaine ;
2. La seconde partie de ce corpus ([5]) contient des annotations plus détaillées se rapprochant plus de celles du premier corpus, disponibles dans des fichiers textes. Pour chaque identifiant de tweet, différents attributs sont disponibles (texte annoté, médicament cible, etc.) mais celui qui nous intéresse est le type d'annotation (effet secondaire ou indication). Nous avons donc écrit un nouveau script Python qui permet de lire ces fichiers et de récupérer les annotations binaires. Cette seconde partie du corpus contient 1754 tweets.

Après comparaison, aucun des tweets du premier corpus n'est présent dans le second. Cependant, il existe certains doublons dans les tweets du second corpus, qui contient après suppression 8119 tweets distincts.

1.3 Base de données finale

Après avoir récupéré les identifiants des tweets d'intérêt ainsi que les annotations binaires associées, nous avons utilisé la librairie Tweepy qui permet d'extraire les tweets à partir de leur identifiant grâce à la fonction get_status. L'API Twitter fixe une limite concernant le nombre de requêtes pouvant être réalisées par période de 15 minutes, qui dépend de la complexité des requêtes. Ces tweets sont stockés dans un DataFrame (librairie Pandas) contenant, pour chaque tweet, son identifiant, son texte, son annotation binaire et le corpus duquel il est issu. La *table 1* présente le nombre de tweets récupérés par corpus et par annotation binaire. Tous les tweets n'ont pas pu être extraits car certains ne sont plus disponibles (le tweet a été supprimé ou l'utilisateur a quitté Twitter).

TABLE 1 – Synthèse de la récupération des tweets.

	Corpus 1	Corpus 2	Total
Nombre d'identifiants	1000	8119	9119
Nombre de tweets récupérés	542	4251	4793
Effet secondaire (1)	208	621	829
Pas d'effet secondaire (0)	334	3630	3964

Puisque chaque tweet récupéré contient au moins un nom de médicament dans son texte, nous nous sommes intéressés aux différents médicaments présents dans notre base de données finale. Grâce aux annotations des deux corpus, nous avons récupéré une liste de 87 médicaments dont nous avons recherché la présence dans le texte des tweets. Pour la grande majorité des tweets de notre base (4068 sur 4793), un seul médicament a été détecté. Pour 312 tweets, aucun médicament n'a été détecté alors que pour les tweets restants plus d'un médicament a été détecté. Au total, 4984 occurrences de médicaments ont été détectées, la *figure 1* présente les 20 plus fréquents (la liste complète des médicaments est disponible en *annexe A*).

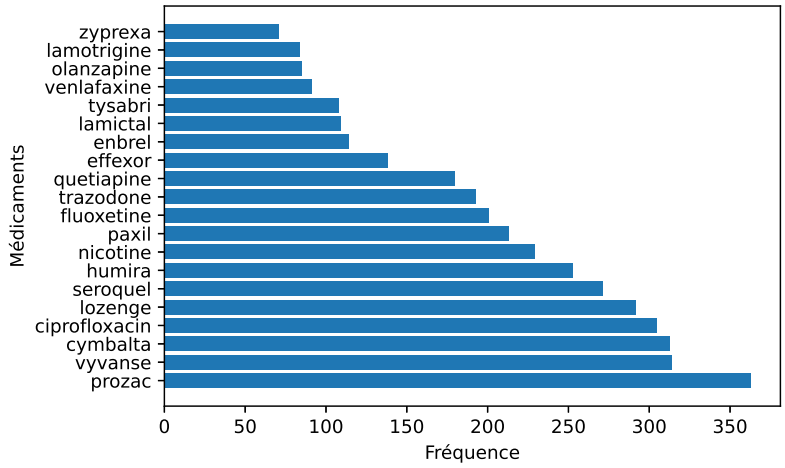

FIGURE 1 – Les 20 médicaments les plus fréquents dans la base de données finale.

2 Natural Language Processing (NLP)

Le traitement naturel du langage, que l'on nomme aussi Natural Language Processing (NLP) en anglais, est une technologie permettant aux machines de comprendre un texte non structuré et d'en extraire de l'information grâce à l'intelligence artificielle (IA).

2.1 Pré-processing et nettoyage des tweets

Le texte est une forme non structurée de données contenant beaucoup d'informations non pertinentes, ce qui nécessite une phase de pré-processing avant analyse. C'est en effet ce nettoyage et la standardisation du texte à travers le pré-processing qui permettent de supprimer les informations inutiles présentes dans le texte et de réaliser par la suite diverses analyses.

Le pré-proccesing du texte peut être résumé par les étapes suivantes :

- Normaliser les caractères (minuscule) ;
- Supprimer les caractères spéciaux (dièses, arobases, ponctuation, etc.) ;
- Supprimer les emoticônes ;
- Enlever les stopwords (mots n'étant pas porteurs de sens tels que "the", "which", "on" etc.) ;
- Enlever les mots de longueur inférieure à deux ;
- Enlever la répétition de caractère dans un mot (le mot "okkayyyyyy" deviendra "okay").

Après cette étape, nous devons normaliser le texte. Pour cela, il nous faut commencer par les tokeniser. Les tokens sont des mots ou termes individuels et la tokenisation est le processus qui consiste à séparer une chaîne de caractères en tokens. Nous appliquons par la suite la méthode de stemming sur les tokens afin de normaliser les mots par troncation. Ainsi, les mots sont réduits à leur forme "racine", par exemple le mot "connections" deviendra "connect" et le mot "consulting" devriendra "consult".

2.2 Vectorisation des tweets

Afin d'appliquer les algorithmes d'apprentissage supervisé au texte, il faut le convertir en vecteurs. Pour cela, plusieurs techniques de transformation existent. Les trois méthodes utilisées dans le cadre de ce projet sont détaillées ci-dessous : Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), et Word Embeddings (Word2Vec).

2.2.1 Bag of Words (BoW)

La méthode Bag of Words est la première méthode de vectorisation que nous avons mise en place. Considérons un corpus C composé de T tweets $\{t_1, \ldots, t_T\}$ et supposons que l'ensemble de ces tweets est représenté par N tokens uniques. Ces N tokens (qui représentent les mots des tweets) vont former un sac de mots et la matrice Bag of Words M sera alors de taille $T * N$. Ainsi, la ligne i de M contiendra la fréquence de chacun des N tokens dans le tweet t_i.

Considérons l'exemple suivant :

Le corpus C est composé de 2 tweets (T=2) :
t_1 : "I am under zoloft, I feel I have become so numb."
t_2 : "I am on a small dose of zoloft I feel good."

Le sac de mots (N=15) est la liste des tokens uniques présents dans le corpus, c'est à dire :
$[' I',' am',' under',' zoloft',' feel',' have',' become',' so',' numb',' on',' a',' small',' dose',' of',' good']$.

La matrice M de taille $2 * 15$ sera représentée sous la forme :

	I	am	under	zoloft	feel	have	become	so	numb	on	a	small	dose	of	good
t_1	3	1	1	1	1	1	1	1	1	0	0	0	0	0	0
t_2	2	1	0	1	1	0	0	0	0	1	1	1	1	1	1

À présent, les colonnes de la matrice M peuvent être utilisées comme un ensemble de vecteurs associés aux tokens du corpus afin de construire un modèle de classification.

Cependant, cette méthode présente deux limites majeures. Tout d'abord, elle ne prend pas en considération le sens sémantique de la phrase et ignore le contexte dans lequel le mot est utilisé. Ensuite, la représentation vectorielle est de très grande taille ce qui implique des temps de calcul très important dès que le nombre de tweets traités augmente. On parle alors de matrice sparse ou creuse, qui contient trop de zéros. On peut néanmoins limiter la taille de cette matrice en effectuant un très bon pré-processing puisqu'on ne garde que les mots importants, ce qui permet donc de réduire la taille du sac de mots.

2.2.2 Term Frequency-Inverse Document Frequency (TF-IDF)

La méthode TF-IDF permet d'obtenir la représentation vectorielle des mots dans un corpus. Elle est basée sur la fréquence d'apparition des mots dans une phrase et se différencie de la méthode Bag of Words par la prise en compte, non seulement de l'occurrence du mot dans un tweet, mais aussi de son occurrence dans l'ensemble du corpus. En effet, si un mot apparaît à plusieurs reprises dans un tweet, alors il a probablement plus d'importance que d'autres mots moins fréquents (TF) ; si un mot apparaît à plusieurs reprises dans un tweet mais est fréquent ou présent dans d'autres tweets, alors il est probable que ce mot soit fréquemment utilisé et qu'il ne faille pas lui accorder trop d'importance (IDF).

Ainsi, TF-IDF permet de pénaliser les mots très fréquents en leur attribuant des poids faibles alors que les mots qui sont rares du point de vue du corpus mais fréquents dans quelques tweets se verront attribuer un poids plus élevé.

Le calcul du score TF-IDF s'effectue suivant les formules suivantes :

$$TF = \frac{\text{Nombre d'occurrence du mot m dans le tweet t}}{\text{Nombre total de mots dans le tweet t}},$$

$$IDF = log(\frac{\text{Nombre de tweets dans le corpus}}{\text{Nombre de tweets dans lesquels le mot m apparait}}).$$

Finalement :

$$\text{TF-IDF} = TF \times IDF.$$

2.2.3 Word embedding (Word2Vec)

Le word embedding est, quant à lui, une méthode de représentation vectorielle permettant de réduire la dimension des vecteurs de mots tout en préservant les similarités de contexte présentes dans le corpus. De plus, cette méthode permet de prendre en compte le sens des mots, les relations sémantiques ainsi que les différents types de contexte dans lesquels les mots sont utilisés. Ainsi, dans cet espace, tous les mots partageant un sens commun auront tendance à être proches les uns des autres. Le word embedding peut être effectué par deux méthodes. La première, le Continuous Bag of Words (CBOW), tend à prédire la probabilité de présence d'un mot dans un contexte donné (un seul mot adjacent ou groupe de mots environnants). La seconde méthode, Skip-gram, à l'inverse tente de prédire le contexte associé à un mot donné. Dans la suite de notre travail, nous avons préféré la méthode Skip-gram car elle permettrait de mieux identifier les mots rares, ce qui est en adéquation avec la faible fréquence d'effets indésirables attendue.

3 Construction des modèles de classification binaire

Comme mentionné précédemment, nous avons dans un premier temps construit un modèle de classification binaire permettant de prédire si un tweet contient un effet secondaire (ADR pour Adverse Drug Reaction) ou non. Pour cela, nous avons testé plusieurs algorithmes d'apprentissage supervisé pour construire notre modèle prédictif pour la classification binaire des tweets.

3.1 Régression logistique

La première méthode algorithmique mise en place est la régression logistique. L'algorithme de régression logistique est utilisé en classification pour prédire une variable binaire (ici Oui/Non) en fonction d'un ensemble de variables indépendantes. La régression logistique permet de prédire la probabilité d'occurrence d'un évènement en ajustant les données sur une fonction logit.

3.2 Support Vector Machine

Le Support Vector Machine (SVM) est un algorithme de classification supervisée. Cet algorithme consiste en la représentation des vecteurs de mots (BoW, TF-IDF ou Word2Vec) dans un espace de dimension n (où n est la taille des vecteurs) puis en la classification des tweets, réalisée en cherchant l'hyperplan permettant de séparer au mieux les données en deux classes.

3.3 Random forest

Random Forest est un algorithme qui permet de réaliser des tâches de classification et de régression. Cette méthode combine plusieurs classifieurs faibles pour construire un modèle de classification plus puissant. Avec l'algorithme Random Forest, le but est de construire plusieurs arbres de décision indépendants au lieu d'un seul. Ainsi, pour classer un nouvel objet, chaque arbre "vote" pour la classe à laquelle l'objet devrait être assigné et la forêt choisit ensuite la classe qui a le plus de votes.

La construction de la forêt s'effectue selon la méthode suivante. On commence par découper notre jeu de données d'apprentissage (matrice de tweets vectorisés) en plusieurs sous-ensembles constitués d'échantillons tirés aléatoirement avec remise. Ensuite, on entraîne un modèle sur chaque sous-ensemble. On dispose alors d'autant de modèles que de sous-ensembles. Enfin, on combine tous les résultats des modèles (système de vote) pour obtenir le résultat final (décision de la forêt). Cette méthode permet ainsi de construire un classifieur robuste à partir de plusieurs modèles indépendants qui ne sont pas forcément aussi robustes.

3.4 XGBoost

Extreme Gradient Boosting (XGBoost) est une implémentation optimisée de l'algorithme d'arbres de boosting de gradient. Il mêle des algorithmes de résolution de modèles linéaires et d'arbres. L'idée de cet algorithme est de corriger, à chaque itération, les erreurs du passées en attribuant un poids plus élevé aux tweets ayant été mal classés. Ainsi, lors d'une nouvelle itération, la construction de l'arbre prendra en compte cette pondération. On pourra alors vérifier si, à cette nouvelle étape, les tweets ont été bien classés. Cet algorithme présente de nombreux avantages. Tout d'abord, il permet la parallélisation, ce qui le rend rapide en termes de temps de calcul. Ensuite, c'est un algorithme qui limite les risques de sur-apprentissage. De plus, il gère bien les valeurs manquantes. Enfin, un système de validation croisée est implémenté dans la méthode, ce qui permet de réaliser une validation croisée à chaque itération du processus de boosting.

3.5 Classifieur bayésien naïf

Le problème de classification consiste à trouver la classe "c" à laquelle il est le plus probable que le tweet "t" appartienne, c'est-à-dire trouver $\hat{c}$ tel que $\hat{c} = argmax_c(\mathbf{P}(c|t))$. Le classifieur bayésien naïf est un algorithme de classification qui permet de résoudre ce problème en utilisant le théorème de Bayes. En effet, le théorème de Bayes fournit une méthode pour estimer $\mathbf{P}(c|t)$:

$$\mathbf{P}(c|t) = \frac{\mathbf{P}(t|c) * \mathbf{P}(c)}{\mathbf{P}(t)}.$$

Le problème peut donc être simplifié et devenir :

$$\hat{c} = argmax_c(\mathbf{P}(t|c) * \mathbf{P}(c)).$$

Un tweet peut être considéré comme une liste de N mots w_i donc $\mathbf{P}(t|c)$ peut se réécrire de la façon suivante :

$$\mathbf{P}(t|c) = \prod_{i=1}^{N} \mathbf{P}(w_i|c).$$

En utilisant l'échelle logarithmique, il vient alors que :

$$\hat{c} = argmax_c(log(\mathbf{P}(t|c) * \mathbf{P}(c))) = argmax_c(log(\mathbf{P}(c)) + \sum_{i=1}^{N} log(\mathbf{P}(w_i|c).$$

Pour estimer les deux termes de cette fonction il suffit alors d'utiliser les fréquences. La probabilité d'une classe est alors la fréquence d'apparition de la classe dans notre jeu d'entraînement et la probabilité d'un mot sachant une classe est la fréquence d'apparition de ce mot dans la classe.

3.6 Réseau de neurones

Le réseau de neurones que nous avons implémenté prend en entrée la représentation vectorielle des mots (BoW, TF-IDF, ou Word2Vec) et retourne les probabilités d'appartenance à chacune des deux classes considérées. Il se compose d'une couche cachée à 10 neurones qui utilise la fonction d'activation "ReLU" (Rectified Linear Unit). La couche externe utilise quant à elle la fonction d'activation sigmoïde, qui permet de calculer les probabilités d'appartenance à chacune des deux classes.

Nous avons entraîné ce réseau de neurones sur 100 epochs en utilisant comme fonction de coût l'entropie croisée, qui est adaptée pour la classification binaire, et comme métrique l'accuracy. La descente de gradient a été effectuée avec des batchs de taille 10 et l'algorithme ADAM (Adaptive Moment estimation) car il est très polyvalent, efficace et stable.

Le réseau de neurones que nous avons implémenté est simple et les choix réalisés sont standards. D'autres architectures pourraient être implémentées pour essayer d'améliorer les performances de ce modèle.

4 Application du modèle à nos données et tuning

Comme nous l'avons présenté dans la première section, nous avons constitué une base de données contenant 4793 tweets et leurs annotations à partir de différents corpus déjà annotés. Nous avons alors utilisé cette base pour entraîner et tester nos différents modèles de classification binaire. Cette étape de construction et validation de modèles a été effectuée dans trois cas : sur une base déséquilibrée, sur une base équilibrée à la main et enfin sur une base équilibrée avec l'algorithme SMOTE (Synthetic Minority Oversampling Technique).

4.1 Mesure de précision

Nous avons décidé de comparer nos modèles en se basant sur le f1-score mais aussi sur l'accuracy. Le f1-score représente la moyenne pondérée de la précision et du recall et tient donc compte à la fois des faux positifs et des faux négatifs. De plus, il convient aux problèmes de distribution inégale des classes grâce au recall, détaillé plus bas. L'accuracy, quant à lui, représente la capacité du modèle à prédire correctement les valeurs positives et négatives. En résumé, on peut dire que le f1-score cherche le pourcentage de valeurs positives (présence d'ADR) tandis que l'accuracy cherche le pourcentage de bien classé (ADR ou non-ADR). Ces deux indices sont donc plutôt complémentaires.

Ces deux valeurs sont basées sur les définitions suivantes :

- **Vrais Positifs (VP) :** Tweets correctement prédits positifs, c'est-à-dire qu'ils sont classés ADR (effet secondaire) et leur véritable classe est aussi ADR.
- **Vrais Négatifs (VN) :** Tweets correctement prédits négatifs, c'est-à-dire qu'ils sont classés non-ADR (pas d'effet secondaire) et leur véritable classe est aussi non-ADR.
- **Faux Positifs (FP) :** Tweets faussement prédits positifs, c'est-à-dire qu'ils sont classés ADR alors que leur véritable classe est non-ADR.
- **Faux Négatifs (FN) :** Tweets faussement prédits négatifs, c'est-à-dire qu'ils sont classés non-ADR alors que leur véritable classe est ADR.

A partir de ces quatre définitions, il est possible calculer l'accuracy mais aussi la précision et le recall pour en déduire la formule du f1-score.

$$Precision = \frac{VP}{VP + FP}, \qquad Recall = \frac{VP}{VP + FN}$$

,

$$F1 - Score = 2\frac{Recall * Precision}{Recall + Precision}, \qquad Accuracy = \frac{VP + VN}{VP + VN + FP + FN}.$$

4.2 Données déséquilibrées

Comme nous l'avons présenté dans la première partie, notre base de données est issue de deux corpus pré-annotés relatant des effets secondaires (ADR=1) ou non (ADR=0). Notre base s'avère cependant être déséquilibrée puisque sur les 4793 tweets qui la composent, seulement 829 incluent au moins un effet secondaire. Ce déséquilibre peut représenter un problème lors de l'entraînement de notre modèle de classification binaire car ce dernier aura tendance à prédire tous les tweets comme non-ADR. Tout en ayant conscience de l'impact de ce déséquilibre, nous avons décidé, dans un premier temps, d'entraîner nos modèles sur cette base déséquilibrée.

Nous avons alors divisé notre base de données en un échantillon d'apprentissage composé de 4315 tweets (90%) et un échantillon de test 478 tweets (10%). Nous avons alors entraîné puis testé les modèles de la section précédente afin d'en comparer le f1-score (en bleu) et l'accuracy (en vert) pour chaque type de vectorisation.

TABLE 2 – F1-score global (bleu), et accuracy (vert) pour différentes méthodes de classification sur les données déséquilibrées.

	Bag of Words	TF-IDF	Word2Vec
Régression Logistique	0.55 0.86	0.30 0.83	0.41 0.84
SVM	0.55 0.83	0.55 0.87	0.15 0.85
Random Forest	0.49 0.86	0.42 0.85	0.30 0.83
XGBoost	0.60 0.86	0.51 0.83	0.48 0.84
Réseau de neurones (1 couche cachée)	0.49 0.80	0.52 0.82	0.56 0.84
Classifieur Bayésien Naïf	0.53 0.83		

Sur la base de données déséquilibrées les différentes méthodes de classification permettent d'obtenir des f1-scores globaux assez faibles (min-max : 0.15-0.60) mais un score d'accuracy intéressant (min-max : 0.80-0.87) (*table 2*). Il apparaît donc ici que le pourcentage de tweets bien classés est élevé pour tous nos modèles (acurracy élevé) mais que le pourcentage de valeurs positives est plutôt faible (f1-score $\leq$ de 0.5). Les matrices de confusions disponibles en *annexe B* illustrent cet effet. Cette différence non négligeable est en grande partie due au fait que le nombre de tweets ADR est beaucoup plus faible que le nombre de tweets non-ADR. Ainsi, les différents modèles prédisent très bien le fait de ne pas avoir d'effet secondaire mais ont tendance à mal classer les tweets contenant un effet secondaire.

4.3 Données équilibrées à la main

Afin de palier au problème de performance de nos algorithmes dû au déséquilibre de la base mis en avant précédemment, nous avons équilibré notre base de données à la main. Pour cela, nous avons conservé les 829 tweets dont l'annotation affirme qu'ils relatent au moins un effet secondaire puis, afin d'avoir autant de tweets ADR que de non-ADR, nous avons tiré aléatoirement 832 tweets dont l'annotation est non-ADR. La base de données que nous avons utilisé ici est donc constituée de 1661 tweets et est équilibrée. Cependant, cet équilibrage a conduit à considérablement réduire la taille de la base de données sur laquelle nos modèles sont entraînés puis testés. En effet, la base passe de 4793 tweets à 1661, soit une réduction de la taille de la base d'environ 65%, ce qui risque d'avoir un impact sur la qualité de nos modèles et notamment sur le f1-score global.

Comme précédement, notre base de 1661 tweets est divisée en un échantillon d'apprentissage composé de 1472 tweets (90%) et un échantillon de test 189 tweets (10%). Les nouvelles valeurs du f1-score (en bleu) et de l'accuracy (en vert) pour chaque modèle sont inscrites dans la *table 3* pour chaque type de vectorisation.

TABLE 3 – F1-score global (bleu), et accuracy (vert) pour différentes méthodes de classification sur les données équilibrées à la main.

	Bag of Words	TF-IDF	Word2Vec
Régression Logistique	0.75 0.75	0.74 0.73	0.72 0.71
SVM	0.76 0.76	0.76 0.75	0.74 0.72
Random Forest	0.78 0.78	0.73 0.74	0.70 0.68
XGBoost	0.75 0.75	0.74 0.74	0.68 0.68
Réseau de neurones (1 couche cachée)	0.78 0.77	0.72 0.71	0.74 0.69
Classifieur Bayésien Naïf	0.77 0.76		

Sur la base de données équilibrée à la main, on remarque que le f1-score est bien meilleur que sur la base déséquilibrée (min-max : 0.68-0.78) (*table 3*). Les scores d'accuracy, quant à eux, sont moins bons que ceux obtenus sur la base déséquilibrée et présentent souvent des valeurs plus faibles que le f1-score (min-max : 0.68-0.78). Ainsi, dans cette analyse, les différents modèles prédisent mieux le fait qu'un tweet inclut un effet secondaire mais le pourcentage de ceux qui sont bien classés est plus faible. La classification des effets secondaires est donc meilleure mais les performances des modèles peuvent être améliorées. Ce nouveau problème de performance peut être expliqué par la taille de notre base de donnée qui a fortement été diminuée par l'équilibrage.

4.4 Données équilibrées par SMOTE

Nous avons vu que travailler sur des données déséquilibrées posait un problème pour la performance de classification ADR mais qu'équilibrer la base à la main réduisait sa taille et donc la performance globale de nos algorithmes de classification. Pour tirer le meilleur de nos algorithmes, nous avons donc utilisé l'algorithme SMOTE (Synthetic Minority Oversampling Technique) qui permet de simuler des données afin d'équilibrer notre base en augmentant sa taille. Ainsi, les 3964 tweets non-ADR de notre base initiale sont conservés et notre base est augmentée par SMOTE jusqu'à contenir 3964 tweets ADR. La base de données que nous avons utilisée ici est donc constituée de 7928 tweets et est équilibrée, sans que sa taille ne soit réduite pour autant.

Nous avons divisé notre base de 7928 tweets en un échantillon d'apprentissage composé de 7144 tweets (90%) et un échantillon de test de 784 tweets (10%). Les nouvelles valeurs du f1-score (en bleu) et de l'accuracy (en vert) pour chaque modèle sont inscrites dans la *table 4* pour chaque type de vectorisation.

TABLE 4 – F1-score global (bleu), et accuracy (vert) pour différentes méthodes de classification sur les données équilibrées par SMOTE.

	Bag of Words	TF-IDF	Word2Vec
Régression Logistique	0.86 0.86	0.91 0.91	1 1
SVM	0.85 0.84	0.93 0.93	1 1
Random Forest	0.82 0.81	0.96 0.96	1 1
XGBoost	0.92 0.92	0.92 0.92	1 1
Réseau de neurones (1 couche cachée)	0.87 0.87	0.92 0.92	1 1
Classifieur Bayésien Naïf	0.75 0.75		

Sur la base de données équilibrée par SMOTE, on remarque que le f1-score ainsi que l'accuracy ont des valeurs très élevées pour l'ensemble de nos modèles allant jusqu'à un pour tous les modèles construits à partir de la vectorisation Word2Vec (*table 4*). Ainsi, l'ensemble de nos modèles ont de meilleures performances de classification.

Nous pouvons conclure qu'il est primordial de travailler sur une base équilibrée et de taille suffisante pour permettre de mieux entraîner des algorithmes de classification. Comme on l'a constaté, tous les modèles donnent d'excellents résultats pour la vectorisation Word2Vec. Le rapport entre résultats et complexité algorithmique étant faible pour le réseau de neurones, nous décidons de ne conserver que les quatre méthodes suivantes : Régression logistique, SVM, Random Forest et XGBoost pour la suite.

4.5 Tuning et cross-validation

Les modèles que nous avons retenus après entraînement et validation sur la base équilibrée SMOTE sont : la régression logistique, le SVM, le Random Forest et le XGBoost. Nous avons alors voulu estimer la fiabilité de nos modèles par validation croisée, c'est à dire en se basant sur une technique d'échantillonnage. En effet, même si nous avons montré dans la *table 4* que le modèle était très performant sur les données de test, en fonction de l'échantillon de validation, la performance de validation du modèle peut varier. La validation croisée permet alors de tirer plusieurs échantillons de validation d'une même base de données. L'estimation de la performance du modèle par validation croisée sera alors plus robuste.

La *figure 2* ci-dessous est un box-plot des estimations de la performance de chaque modèle par validation croisée 5-fold. On remarque sur ce graphique que la performance des modèles est moins bonne que celle que l'on avait obtenue dans la *table 4*. La validation croisée nous a donc permis d'obtenir des résultats plus robustes et nous pouvons en déduire que le modèle le plus performant est le XGBoost.

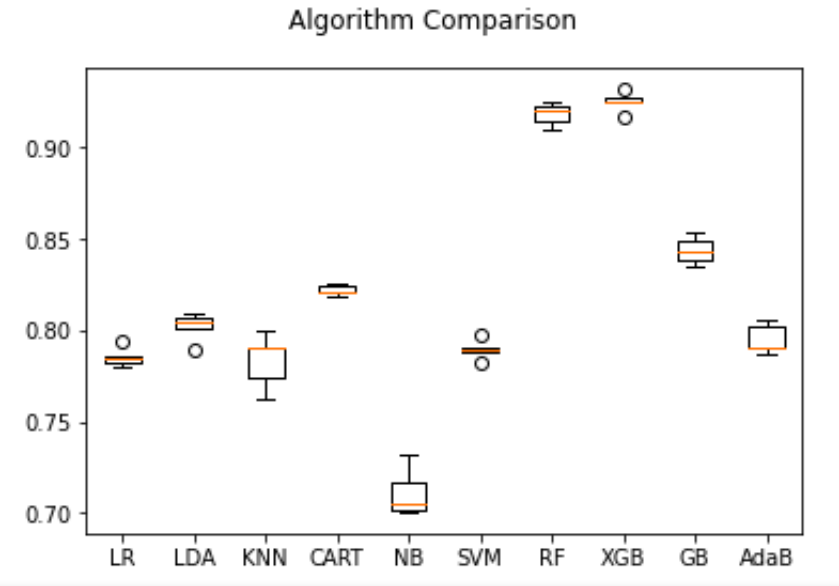

FIGURE 2 – Comparaison des performances des méthodes par validation croisée 5-fold.

Nous avons donc décidé de conserver le modèle XGBoost pour la classification binaire de nos tweets car c'est celui qui permet d'obtenir la meilleure performance par validation croisée. Nous avons alors voulu voir s'il était possible d'améliorer davantage ce modèle. Un moyen de le rendre plus performant est de choisir les hyperparamètres, c'est-à-dire les paramètres d'ajustement du modèle XGBoost.

Pour cela, nous avons effectué un tuning sur notre modèle à l'aide de la méthode d'optimisation bayésienne Hyperopt. Il s'agit d'une technique probabiliste basée sur un modèle permettant de trouver le minimum d'une fonction donnée en paramètre. Dans notre cas, il s'agit du f1-score. Ensuite, comme la plupart des algorithmes bayésiens, cette méthode va prendre en entrée une combinaison de paramètres en connaissance de cause. Il s'agira de loi à priori non informative. Enfin, Hyperopt utilise l'estimateur de Parzen des arbres (TPE : Tree Parzen Estimator).

Pour plus de détails sur cet estimateur mais aussi sur les paramètres utilisés, une explication est disponible en *annexe C*, écrite à l'aide de plusieurs articles ([3], [6]).

La *table 5* nous donne la valeur optimale des hyperparamètres pour le modèle XGBoost qui nous permettent d'obtenir la meilleure performance. La *table 6* quant à elle, nous donne l'accuracy et le f1-score, suite à l'utilisation des paramètres optimaux.

TABLE 5 – Résultats du tuning du modèle XGBoost.

colsample bytree	1
gamma	0.85
learning rate	0.12
max depth	11
min child weight	4
alpha	4
lambda	0.57
n estimators	200

TABLE 6 – F1-score global (bleu), et accuracy (vert) les résultats finaux pour le XGBoost.

	XGBoost
F1 Score	0.92
Accuracy	0.92

Dans la suite de notre travail, nous avons donc utilisé le modèle XGBoost entraîné sur la base de données équilibrée SMOTE et construit avec les valeurs d'hyperparamètres présentées dans la *table 5*.

5 Catégorisation de l'effet indésirable

Nous avons précédemment décrit comment classer les tweets selon s'ils contiennent un effet secondaire ou non. Nous allons maintenant présenter comment catégoriser l'effet secondaire par l'utilisation d'un dictionnaire issu du programme Canada Vigilance.

5.1 Dictionnaire des effets secondaires - Base Canada Vigilance

Canada Vigilance ([2]) est un programme de surveillance des effets indésirables des médicaments après leur mise sur le marché canadien. Elle regroupe les effets secondaires apparus après l'administration d'un médicament et remontés aux autorités canadiennes. Nous avons récupéré et nettoyé cette base afin de créer un dictionnaire d'effets indésirables classés dans différentes familles d'effets secondaires. Cette base contient 27 familles d'effets secondaires (description disponible en *annexe D*) regroupant chacune entre 64 et 1975 effets secondaires différents.

5.2 Catégorisation

Une fois qu'un tweet est prédit comme faisant mention d'un effet secondaire, nous cherchons à le catégoriser dans une des familles recensées dans la base Canada Vigilance. Nous commençons par regarder si un effet secondaire (pré-processé) de notre dictionnaire est présent dans le tweet. Si c'est le cas nous catégorisons l'effet secondaire dans la famille correspondante. S'il y a plusieurs effets secondaires, le tweet est catégorisé dans les différentes familles. En appliquant cette méthode aux 829 tweets ADR de notre base de données, nous observons que pour 388 d'entre eux aucun effet secondaire de la base Canada Vigilance n'est trouvé, ils ne sont donc pas catégorisés. Cette méthode n'est donc pas efficace pour tous les tweets, probablement car dans certains tweets l'effet secondaire n'est pas décrit de façon précise et formelle alors que c'est le cas dans la base Canada Vigilance.

Nous avons tenté de résoudre ce problème en développant une méthode complémentaire de catégorisation des effets secondaires. Ainsi, parallèlement à la recherche directe dans le dictionnaire, nous recherchons la similarité la plus probable entre le tweet et un effet secondaire de la base Canada Vigilance. Dans un premier temps, le tweet est vectorisé et les effets secondaires sont normalisés et vectorisés en utilisant la méthode Word2Vec. Dans un second temps, la similarité cosinus est calculée. Cette dernière est le rapport entre un produit vectoriel et le produit des normes 2 de chaque vecteur. Sa formule mathématique est définie comme suit :

$$S = \frac{u \wedge v}{\|u\|_2 \times \|v\|_2}$$

où u et v sont respectivement les vecteurs représentant un tweet et un effet secondaire, dans un espace de dimension 200. Cette valeur varie entre 0 (aucune similarité) et 1 (similarité parfaite).

Nous calculons donc la similarité entre chaque tweet et chaque effet secondaire. Ainsi, pour chaque tweet, nous avons n valeurs de similarité, où n est le nombre d'effets secondaires. Parmi ces n valeurs, nous gardons uniquement la valeur maximale comme étant l'effet secondaire le plus probable. Nous pouvons ensuite catégoriser notre tweet à partir de cet effet secondaire.

Enfin, nous combinons les résultats obtenus précédemment pour ne conserver que les familles d'effets secondaires. Si un effet secondaire est exprimé explicitement dans le tweet nous gardons la famille associée et si aucun effet secondaire de la base Canada Vigilance n'est présent dans le tweet, nous conservons la famille prédite par similarité.

Finalement, une fonction générale permet, à partir d'un nom de médicament, de récupérer un nombre (déterminé par l'utilisateur) de tweets, de réaliser la classification binaire et de catégoriser les tweets faisant mention d'un effet secondaire dans les différentes familles de la base Canada Vigilance. La *figure 3* ci-dessous illustre les résultats obtenus pour deux médicaments : statins et antipsychotic.

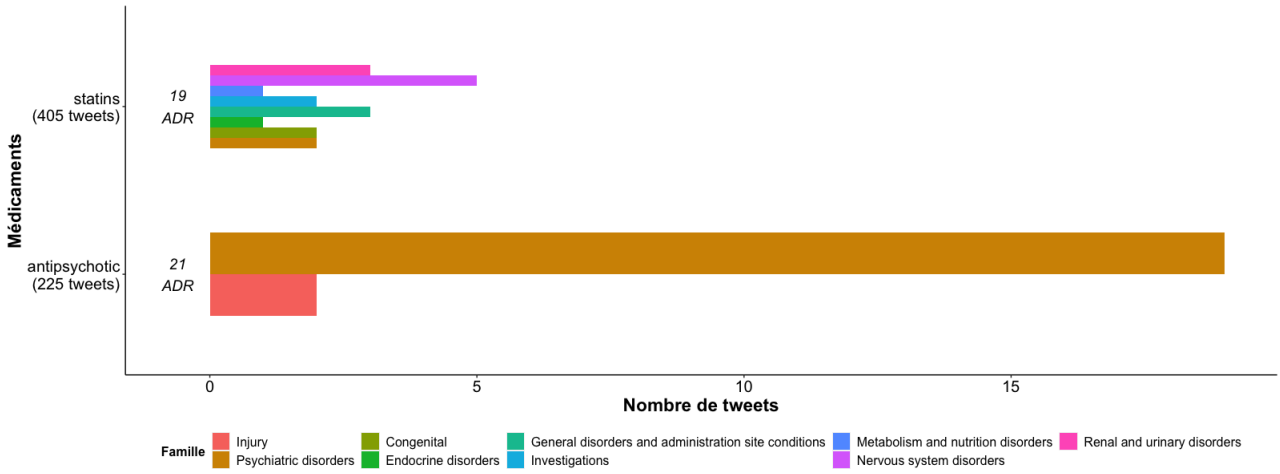

FIGURE 3 – Histogramme des familles d'effets secondaires pour plusieurs médicaments.

Nous avons récupérés 405 et 225 tweets contenant respectivement les médicaments statins et antipsychotic. Parmi ces tweets, respectivement 19 et 21 sont classés comme contenant un effet indésirable. Nous remarquons que la famille "Psychiatric disorders" (agoraphobie, la fatigue mentale, etc.) est particulèrement présente pour les tweets ADR de l'antipsychotic. A ce stade, il faudrait se demander s'il s'agit réellement d'un effet secondaire ou de la raison de la prise du médicament. Cet exemple illustre ainsi la difficulté de détecter et classer les effets secondaires dans des tweets. De plus, il est possible que certains tweets contiennent à la fois la raison de la prise du médicament et un ou plusieurs effets secondaires. Or, la méthode de catégorisation que nous avons développée ne permet pas de faire cette distinction.

6 Conclusion et discussion

Le but de ce travail était de développer un modèle de classification binaire permettant de détecter les tweets contenant un nom de médicament qui mettent en évidence un effet secondaire. Dans un second temps, il s'agissait de catégoriser les effets secondaires relatés dans les tweets pour associer chaque médicament à une ou plusieurs familles d'effets secondaires.

Après avoir construit notre propre base de données à l'aide de plusieurs bases déjà annotées issues de la littérature scientifique, nous avons implémenté plusieurs algorithmes de classification binaire. L'équilibrage SMOTE ainsi que la validation croisée et le tuning nous ont finalement permis de construire un modèle XGBoost avec un f1-score et une accuracy de 0.92. Enfin, l'algorithme final permet de collecter des tweets en lien avec une molécule donnée en paramètre, de ne conserver que ceux classés ADR par notre modèle puis d'associer à chaque tweet une ou plusieurs familles d'effets secondaires par recherche de mots-clés ou similarité en se basant sur la base Canada Vigilance.

Toutefois, cet algorithme présente encore quelques défauts. Tout d'abord, certains tweets sont classés comme ADR et se voient associés une famille d'effets secondaires qui se trouve être en réalité la raison et non la conséquence de la prise du traitement. Par exemple, pour l'Ibuprofène, les maux de tête risquent d'être considérés comme mot-clé d'un effet indésirable alors qu'il s'agit de la raison pour laquelle l'individu a pris le médicament.

De plus, le tweet est un langage particulier qui s'éloigne du langage courant et encore plus du langage scientifique. Ainsi, le calcul de la similarité entre les tweets avec ADR et les transcriptions d'ADR de la base Canada Vigilance est soumise aux différences de langage (familier vs. médical), ce qui peut réduire la qualité de nos résultats.

Enfin, lorsque l'effet secondaire est présent directement dans le tweet, on devrait pouvoir retrouver cet effet secondaire lorsque nous comparons les similarités. Malheureusement, ce n'est pas toujours le cas. Une possible raison à cela peut être la méthode de vectorisation choisie. En effet, nous avons travaillé avec une vectorisation Word2Vec en utilisant le Continous Bag of Words mais une autre technique de vectorisation telle que le Doc2Vec, donnerait peut-être de meilleurs résultats. Cette dernière fonctionne sur le même principe que le Word2Vec mais prend en compte le label du document. Dans notre cas, cela permettrait de prendre en compte la famille d'effets secondaires.

Il est difficile de trouver des données annotées en grande quantité, et le faire à la main est long et demande des connaissances médicales. Dans ce travail, nous n'avons utilisé que les annotations binaires des corpus car les annotations plus détaillées étaient peu nombreuses et hétérogènes entre les deux corpus. Ainsi, avec plus de temps, il aurait été intéressant de construire une base de données plus dense, avec des annotations plus complètes pour pouvoir construire des modèles plus complexes et effectuer des analyses plus poussées.

Enfin, des ordinateurs plus performants nous auraient permis d'aller plus loin dans notre travail en implémentant, par exemple, des méthodes telles que le Bidirectional Encoder Representations from Transformers (BERT), à condition aussi de disposer de beaucoup de données. Cette méthode permet de mieux comprendre les requêtes longues comprenant des phrases complètes avec des prépositions. Ainsi, les mots comme "for" ou "which" que nous avons supprimé jusqu'à présent deviennent important et permettent de contextualiser la requête.

Bibliographie

[1] Nestor Alvaro, Yusuke Miyao et Nigel Collier. « TwiMed : Twitter and PubMed Comparable Corpus of Drugs, Diseases, Symptoms, and Their Relations ». Dans : *JMIR Public Health and Surveillance* 3 (mai 2017), e24. doi : 10.2196/publichealth.6396. url : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5438461/.

[2] *Base Canada Vigilance*. url : https://www.canada.ca/en/health-canada/services/drugs-health-products/medeffect-canada/adverse-reaction-database/canada-vigilance-online-database-data-extract.html.

[3] James Bergstra et al. « Algorithms for Hyper-Parameter Optimization ». Dans : *Advances in Neural Information Processing Systems*. T. 24. Curran Associates, Inc., 2011. url : https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[4] Thomas Ly et al. « Evaluation of Natural Language Processing (NLP) systems to annotate drug product labeling with MedDRA terminology ». Dans : *Journal of Biomedical Informatics* 83 (2018), p. 73-86. issn : 1532-0464. url : https://www.sciencedirect.com/science/article/pii/S1532046418301060.

[5] Azadeh Nikfarjam et al. « Pharmacovigilance from social media : Mining adverse drug reaction mentions using sequence labeling with word embedding cluster features ». Dans : *Journal of the American Medical Informatics Association : JAMIA* 22 (mar. 2015). doi : 10.1093/jamia/ocu041. url : https://pubmed.ncbi.nlm.nih.gov/25755127/.

[6] Yoshihiko Ozaki et al. « Multiobjective tree-structured parzen estimator for computationally expensive optimization problems ». en. Dans : *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. Cancún Mexico : ACM, juin 2020, p. 533-541. isbn : 978-1-4503-7128-5. doi : 10.1145/3377930.3389817. url : https://dl.acm.org/doi/10.1145/3377930.3389817 (visité le 27/02/2021).

[7] Ahmad P. Tafti et al. « Adverse Drug Event Discovery Using Biomedical Literature : A Big Data Neural Network Adventure ». Dans : *Journal of Medical Internet Research* 5 (déc. 2017). doi : 10.2196/medinform.9170.

[8] Abeed Sarker et Graciela Gonzalez. « Portable automatic text classification for adverse drug reaction detection via multi-corpus training ». Dans : *Journal of Biomedical Informatics* 53 (2015), p. 196-207. issn : 1532-0464. doi : https://doi.org/10.1016/j.jbi.2014.11.002. url : https://www.sciencedirect.com/science/article/pii/S1532046414002317.

[9] *Tweepy*. url : https://docs.tweepy.org/en/latest/.

[10] *TwiMed*. url : https://github.com/nestoralvaro/TwiMed.

[11] *Twitter Annotated Corpus.* URL : http://diego.asu.edu/downloads/twitter_annotated_corpus/.

[12] *Twitter API.* URL : https://developer.twitter.com/en/docs/twitter-api.

Annexes

A Liste des médicaments présents dans les tweets de notre base de données

Médicament	Nombre d'occurrences	Médicament	Nombre d'occurrences
prozac	363	sertraline	15
vyvanse	314	lisinopril	15
cymbalta	313	mtx	15
ciprofloxacin	305	avelox	14
lozenge	292	metho	14
seroquel	271	carbamazepine	13
humira	253	singulair	13
nicotine	229	avastin	13
paxil	213	boniva	13
fluoxetine	201	docetaxel	12
trazodone	193	bevacizumab	11
quetiapine	180	topiramate	11
effexor	138	depakote	9
enbrel	114	montelukast	8
lamictal	109	melatonin	8
tysabri	108	mirtazapine	8
venlafaxine	91	provigil	7
olanzapine	85	pregabalin	7
lamotrigine	84	buprenorphine	6
zyprexa	71	tegretol	6
rivaroxaban	58	valproate	6
baclofen	55	vimpat	6
paroxetine	52	kenalo	5
victoza	49	suboxone	5
citalopram	46	viibryd	5
zoloft	45	ziprasidone	4
tamoxifen	45	effient	3
xarelto	45	concerta	2
prednisone	40	melphalan	2
levaquin	40	saphris	2
topamax	28	nasal spray	2
geodon	28	lovetas	1
adderall	27	avalox	1
duloxetine	25	zeldox	1
cortisone	24	nolvadex	1
pristiq	24	subutex	1
celexa	23	focalin	1
metoprolol	22	zestril	1
abilify	21	leflunomide	1
remicade	21	tamazapam	1
ambien	19	infliximab	1
ritalin	17	fluvoxamine	1
lexapro	16	luvox	1
modafinil	15		

B Matrices de confusion

B.1 Données non équilibrées

		Logistique					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	366	17	381	1	374	9
	ADR	52	43	78	17	68	27

		Support Vector Machines					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	347	36	377	6	377	6
	ADR	45	50	57	38	57	38

		Random Forest					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	377	6	379	4	377	6
	ADR	62	33	69	26	77	18

		XGBoost					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	362	21	354	29	368	15
	ADR	45	50	53	42	60	35

		Réseau de neurones					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	335	48	342	41	355	28
	ADR	48	47	47	48	47	48

B.2 Données équilibrées à la main

		Logistique					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	69	22	67	24	67	24
	ADR	26	72	27	71	27	71

		Support Vector Machines					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	71	20	68	23	62	29
	ADR	26	72	24	74	24	72

		Random Forest					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	74	17	71	20	60	31
	ADR	25	73	30	68	29	69

		XGBoost					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	70	21	69	22	64	27
	ADR	27	71	27	71	33	65

		Réseau de neurones					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	67	24	65	26	47	44
	ADR	20	78	29	69	14	84

B.3 Données Equilibrées avec SMOTE

		Logistique					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	332	70	360	42	402	0
	ADR	38	344	28	354	0	382

		Support Vector Machines					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	321	81	354	48	402	0
	ADR	42	340	9	373	0	382

		Random Forest					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	299	103	385	17	402	0
	ADR	44	338	14	368	0	382

		XGBoost					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	370	32	370	32	402	0
	ADR	32	350	32	350	0	382

		Réseau de neurones					
		Prédiction BoW		Prédiction TF-IDF		Prédiction W2Vec	
		non-ADR	ADR	non-ADR	ADR	non-ADR	ADR
Vraie valeur	non-ADR	339	63	341	61	402	0
	ADR	39	343	4	378	0	382

C Fonction XGBoost sous Python

C.1 Explication des paramètres

— La valeur **gamma** nous informe de la perte minimale requise pour pouvoir diviser notre noeud en sous-noeud. Cette variable prend ses valeurs dans $\mathbf{R}^+$. Ainsi, plus γ est grand, plus on s'autorise à avoir de la perte d'information et donc plus l'algorithme sera conservateur.

— **Max depth** correspond à la profondeur maximale d'un arbre. Ce paramètre permet de contrôler le sur-ajustement. En effet, une profondeur élevée indique un modèle complexe. Les chances d'un sur-apprentissage sont ainsi plus élevées.

— **Colsample bytree** correspond au ratio de variable utilisé lors de la phase de sous échantillonnage.

— **N estimators** correspond au nombre d'itérations, c'est-à-dire au nombre de sous-échantillons que l'on utilise pour le bootstrap.

— **Min child weight** correspond, comme son nom l'indique, à un terme minimal pour pouvoir diviser un nœud. Ce terme est dans notre cas la somme minimale des poids d'instances. Ainsi, si la valeur trouvée est inférieure à la valeur fixée, alors la division de l'arbre s'arrête.

— **Reg lambda** et **reg alpha** correspondent respectivement au terme de régularisation dans les normes L2 (équivalente à la régression Ridge) et L1 (équivalente à la régression Lasso). Dans les deux cas, plus ce terme augmente, plus le modèle est conservateur. Ils sont principalement utilisés lorsqu'il y a une très grande dimension.

C.2 Explication de Hyperopt

Pour pouvoir utiliser le package hyperopt dans Python, il nous faut implémenter trois parties.

— **Une fonction à optimiser**
Tout d'abord il faut implémenter une fonction objective à minimiser. Dans notre cas, nous voulons maximiser le f1-score. Nous prenons donc son opposé.

— **L'espace des hyperparamètres**
Ensuite, nous devons implémenter l'espace des paramètres. Plutôt que de prendre des valeurs discrètes, nous leurs assignons une loi de probabilité. Nous la prenons la plus large possible.

- **max depth** : Loi uniforme discrète entre 0 et 14 par pas de 1 ;
- **learning rate** : Loi uniforme logarithmique entre 0.01 et 1 ;
- **gamma** : Loi uniforme discrète entre 0.5 et 1 par pas de 0.05 ;
- **reg alpha** : Loi uniforme discrète entre 0 et 100 par pas de 1 ;
- **reg lambda** : Loi uniforme entre 0 et 1 ;

— **colsample bytree** : Loi uniforme discrète entre 0.5 et 1 par pas de 0.05 ;

— **min child weight** : Loi uniforme discrète entre 1 et 6 par pas de 1.

Par la suite, nous noterons X comme étant le vecteur des hyperparamètres où chacune des valeurs du vecteur est un paramètre à optimiser. De plus, comme les paramètres sont indépendants les uns des autres, la loi du vecteur est simplement le produit des densités de chacun des paramètres.

— **L'algorithme de Tree Parzen Estimator (TPE)**

Entrée

— $D = \{(X_1, y_1), \ldots, (X_k, y_k)\}$: avec X_i le i-ème vecteur d'hyperparamètres où les valeurs qui le composent sont tirées aléatoirement dans leur loi a priori. $y_i = f(X_i)$ est la fonction objective définie auparavant. Par défaut, k = 20 dans Python.

— n_{i} : Nombre d'itérations.

— n_{c} : Nombre de candidats par itération. Par défaut, $n_{\text{c}} = 24$ dans Python.

— γ : Quantile d'observations pour les y_i, donc $\gamma = p\,(y < y^*)$. Par défaut $\gamma = 0,25$ dans Python.

Pour i allant de 1 à n_{i} :

— $D_g \leftarrow \{(X_i, y_i)$ dans D tel que $y_i < y^*.\}$

— $D_b \leftarrow \{(X_i, y_i)$ dans D tel que $y_i >= y^*.\}$

— Construire $g(X)$ et $b(X)$ les densités de probabilité respectivement issues des observations de D_g et D_b. **(1)**

— $C \leftarrow \{Z_j \sim g(X) \mid j = 1, \ldots, n_{\text{c}}\}$

— $Z^* \leftarrow \text{argmax}_{Z \in C}\, \text{EI}_{y^*}(Z)$ avec EI appelé Expected Improvement.

— $D \leftarrow D \cup \{(Z^*, f\,(Z^*))\}$ avec f la fonction objective associée.

Sortie

Retourner le vecteur X de D qui minimise y.

(1) En faisant cela, on suppose que $p(X \mid y) = \begin{cases} g(X) & \text{si } y^* > y \\ b(X) & \text{si } y \geq y^* \end{cases}$.

L'Expected Improvement est construit comme suit :

$$\text{EI}_{y^*}(X) = \int_{-\infty}^{y^*} (y^* - y)\, p(y \mid X) dy. \tag{1}$$

En utilisant la formule de Bayes, on a que

$$\mathrm{EI}_{y^*}(X) = \int_{-\infty}^{y^*} (y^* - y)\, \frac{p(X \mid y)p(y)}{p(X)} dy.$$

Dénominateur :
De plus, on a défini γ comme étant le quantile de loi des observations de y. Donc $\gamma = p\,(y < y^*)$ et par définition on a que :
$p(X) = \int_{\mathbb{R}} p(X \mid y)p(y)dy = \int_{-\infty}^{y^*} p(X \mid y)p(y)dy + \int_{y^*}^{+\infty} p(X \mid y)p(y)dy = \gamma g(X) + (1-\gamma)b(X)$.

Numérateur :

$$\int_{-\infty}^{y^*} (y^* - y)\, p(X \mid y)p(y)dy = g(X) \int_{-\infty}^{y^*} (y^* - y)\, p(y)dy = \gamma y^* g(X) - g(X) \int_{-\infty}^{y^*} y \times p(y)dy$$

Par conséquent, on a que

$$EI_{y^*}(X) = \frac{\gamma y^* g(X) - g(X) \int_{-\infty}^{y^*} y \times p(y)dy}{\gamma g(X) + (1-\gamma)b(X)} \propto \left(\gamma + \frac{b(X)}{g(X)}(1-\gamma)\right)^{-1}. \tag{2}$$

Comme on cherche à maximiser l'Expected Improvement, il nous faut minimiser le rapport $\frac{b(X)}{g(X)}$. Cela revient à minimiser $b(X)$ ou maximiser $g(X)$. Dans les deux cas, il faut mettre en avant les vecteurs X_i dont la valeur y_i associée est inférieure à un y^* fixé.

Au final, cet algorithme permet petit à petit d'affiner la loi de $g(x)$ et ainsi obtenir une bonne estimation des hyperparamètres.

D Description du dictionnaire Canada Vigilance

Le dictionnaire d'effets secondaires obtenu à partir du programme Canada Vigilance permet de regrouper les réactions à un médicament en 27 familles. Ci-dessous nous présentons les différentes familles.

— **Blood and lymphatic system disorders :** les troubles du sang et du système lymphatique, 203 effets secondaires dans cette catégorie (anémie, troubles plaquettaires etc)

— **Cardiac disorders :** les troubles cardiaques, 252 effets secondaires (perforation cardiaque, hypertrophie ventriculaire etc)

— **Congenital, familial and genetic disorders :** les troubles congénitaux, familiaux et génétiques, 479 effets secondaires (otocéphalie, malformation du cerveau etc)

— **Ear and labyrinth disorders :** les troubles de l'oreille et du conduit auditif, 64 effets secondaires (gonflement auriculaire, surdité permanente etc)

— **Endocrine disorders :** les troubles endocriniens ou hormonaux, 98 effets secondaires (atrophie de la thyroïde, hypoaldostéronisme etc)

— **Eye disorders :** les troubles oculaires, 407 effets secondaires (irritation de la cornée, rupture maculaire etc)

— **Gastrointestinal disorders :** les troubles gastro-intestinaux, 657 effets secondaires (hernie abdominale, pancréatite hémorragique etc)

— **General disorders and administration site conditions :** les troubles généraux et au niveau du site d'administration, 621 effets secondaires (hémorragie au niveau du site du cathéter, perte de contrôle des jambes etc)

— **Hepatobiliary disorders :** les troubles hépatobiliaire (ie au niveau du foie), 141 effets secondaires (cirrhose cardiaque, hépatite etc)

— **Immune system disorders :** les troubles du système immunitaire, 116 effets secondaires (allergie permanente, rejet de greffe d'intestin etc)

— **Infections and infestations :** les infections et les infestations, 1139 effets secondaires (abcès au cerveau, infection par le virus de l'herpès etc)

— **Injury, poisoning and procedural complications :** les blessures, les empoisonnements et les complications suite à la procédure d'administration du médicament, 822 effets secondaires (corps étranger dans l'oeil, fracture des cervicales supérieures etc)

— **Investigations :** les troubles trouvés par prise de sang ou divers examens médicaux poussés montrant une évolution par rapport à l'état avant la prise du médicament, 1975 effets secondaires (diminution du nombre d'immunoglobulines, prolongement du temps de coagulation etc)

— **Metabolism and nutrition disorders :** les troubles nutritifs et du métabolisme, 192 effets secondaires (déficience en vitamine A, intolérance au lait etc)

— **Musculoskeletal and connective tissue disorders :** les troubles musculo-squelettiques et du tissu conjonctif, 335 effets secondaires (kyste de la paroi thoracique, atrophie du cartilage etc)

— **Neoplasms benign, malignant and unspecified (incl cysts and polyps) :** les tumeurs bénignes, malignes et non spécifiées (y compris kystes et polypes), 937 effets secondaires (fibrome, carcinome des cellules rénales etc)

— **Nervous system disorders :** les troubles du système nerveux, 685 effets secondaires (amnésie, somnolence etc)

— **Pregnancy, puerperium and perinatal conditions :** les conséquences sur la grossesse, la puerpéralité (période suivant l'accouchement) et les conditions périnatales, 148 effets secondaires (infarctus du placenta, tachysystole utérin etc)

— **Product issues :** les problèmes liés aux produits, 118 effets secondaires (problème de seringue, le produit a pris feu etc)

— **Psychiatric disorders :** les troubles psychiatriques, 406 effets secondaires (agoraphobie, fatigue mentale etc)

— **Renal and urinary disorders :** les troubles rénaux et urinaires, 249 effets secondaires (occlusion de l'artère rénale, rein oedémateux etc)

— **Reproductive system and breast disorders :** les troubles du système reproductif et mammaires, 329 effets secondaires (fibrose mammaire, ménopause prématurée etc)

— **Respiratory, thoracic and mediastinal disorders :** les troubles respiratoires, thoraciques et médiastinaux, 425 effets secondaires (kyste nasal, obstruction nasale etc)

— **Skin and subcutaneous tissue disorders :** les troubles de la peau et des tissus sous-cutanés, 388 effets secondaires (nécrose épidermique, fibrose de la peau etc)

— **Social circumstances :** les impacts sur la qualité de vie (changement de travail, utilisateur d'un dispositif d'assistance cardiaque etc)

— **Surgical and medical procedures :** les conséquences sont des procédures chirurgicales et médicales, 854 effets secondaires (opération de l'oreille, chirurgie plastique etc)

— **Vascular disorders :** les troubles vasculaires, 232 effets secondaires (hémorragie interne, rupture de veine etc)