

Sommaire du Rapport Fil Rouge Électif-IAS

Sécurisation des doses médicamenteuses prescrites par intelligence artificielle - détection de l'iatrogénie médicamenteuse

Projet supervisé par le CHU de Lille



Projet réalisé en équipe :

ACEVEDO Nicolas

ALDANA Pablo

LENGLET Marie

PURIM Andreis

SKEWES Pablo

Introduction

I. Les problèmes de prescription des médicaments

- I.1. État de l'art
- I.2. Définir les erreurs médicamenteuses
- I.3. Analyse et impacts des erreurs de prescription

II. Analyse de la BDD réelles des ordonnances (phase de pré-processing)

- II.1. Les données du CHU
- II.2. Gestion des erreurs humaines dans certaines lignes du CSV
- II.3. Gestion d'un dataframe aussi grand
- II.4. Le regroupement par code ATC, voie d'administration et unité de prescription
- II.5. Changement des unités
- II.6. La gestion de la fréquence
- II.7. Les colonnes supprimées

III. Clustering des ordonnances (machine learning non supervisé)

- III.1. Descriptions des différentes stratégies
- III.2. Les algorithmes de regroupement
- III.3. Word2Vec
- III.4. Simulation et résultats des modèles

IV. Application dans un système d'alerte

V. Apprentissage supervisée

VI. Vos ressentis et analyse des conférences des intervenants extérieurs

Conclusion et perspectives

Bibliographie

Introduction

Actuellement, au CHU de Lille, certaines des ordonnances prescrites par les médecins sont contrôlées par les pharmaciens de l'hôpital qui peuvent demander la révision d'une ordonnance jugée non cohérente. Cette analyse pharmaceutique en amont de la délivrance des médicaments permet de détecter des erreurs d'écriture et d'optimiser la prise en charge médicamenteuse, pour assurer la sécurité des patients. Le contrôle en pharmacie des traitements n'est cependant pas possible pour chaque prescription et concerne principalement les patients entrants et les ordonnances réévaluées, car ce travail représente une charge de travail très importante. Automatiser l'analyse pharmaceutique des ordonnances par une intelligence artificielle permettrait de réviser une plus grande proportion d'ordonnances et de décharger le service de pharmacie. Les pharmaciens pourraient étudier seulement les ordonnances jugées problématiques par l'IA, avant la préparation du traitement par l'automate.

Par ailleurs, la préparation des prescriptions se fait manuellement aujourd'hui à Lille, mais une nouvelle pharmacie robotisée ouvrira en 2025. Quand la pharmacie sera robotisée, il y aura une vérification de la bonne délivrance des médicaments par la machine, mais pas de vérifications des traitements prescrits. Notre travail du fil rouge est de mettre au point un **module d'apprentissage par l'IA de reconnaissance d'une ligne de prescription pharmaceutiquement conforme**, pour pré-analyser les ordonnances avant la production de doses par le robot de la pharmacie.

I. Les problèmes de prescription des médicaments

I.1. État de l'art

Actuellement au CHU de Lille, les prescriptions sont révisées par les pharmaciens, aucun système intelligent automatisé n'est utilisé à la pharmacie pour analyser la conformité des ordonnances. Cependant, des systèmes d'analyse d'IA existent pour l'analyse des prescriptions. Principalement avec deux approches différentes: Une création de relation entre médicament-médicament et médicament patient, ou réaliser une prédiction en base a des données médicale du patient.

Le premier travail a été développé par Yen Po Harvey Chin et al. Ils ont trouvé des associations grâce à du machine learning non supervisé pour pouvoir associer un patient et un médicament, mais aussi un médicament avec un autre. De cette façon, ces règles d'association ont été déployées dans cinq hôpitaux à Taïwan, avec une performance de certitude de 80% et une sensibilité de 80%-96%.

Le deuxième travail a été réalisé par Jennifer Corny et al., ils ont analysé des données de patients pour prédire un score de probabilité pour qu'une ordonnance soit révisée par un pharmacien. Ce modèle a été construit grâce au Light GBM et decision tree pour finalement avoir une classification binaire. En même temps un pharmacien a revisé lui même des ordonnances pour avoir un set d'entraînement, qui a été validé par le modèle par la suite. Finalement, ils comparent leurs solutions avec d'autres solutions classiques déjà utilisées pour retrouver une précision plus élevée.

Nous pouvons voir qu'il y a différentes méthodes et visions d'appréhender la problématique, en fonction des données auxquelles on a accès, et le contexte de l'étude. Notamment on voit que le machine learning est un outil qui aide à ces solutions, et qui permet de trouver des solutions plus efficaces.

I.2. Définir les erreurs médicamenteuses

Les ordonnances regroupent l'information sur un patient dans un moment précis, et le traitement prescrit, qui peut contenir plusieurs médicaments. On retrouve alors trois types principaux de problèmes : est-ce que la quantité prescrite est cohérente avec les doses habituellement prescrites pour cette molécule ? Est-ce que le médicament est compatible avec l'état du patient ? Y a-t-il des problèmes d'interaction avec les autres médicaments de la même ordonnance ?

Le premier est un problème de décision de la dose administrée, de la voie d'administration et de la fréquence. Ce sont des erreurs qui peuvent provoquer une intoxication pour le patient.

Le deuxième correspond à une mauvaise association d'un médicament pour l'état d'un patient. Par exemple, certains médicaments ne doivent pas être prescrits pour des enfants, ou pour des personnes sous un poids seuil. D'autres médicaments ne doivent pas être prescrits si les analyses biologiques indiquent un problème au foie par exemple.

Le troisième est un problème d'interactions entre des médicaments. Par exemple, deux médicaments peuvent contenir la même molécule. Ainsi, un patient prenant ces deux médicaments en même temps pourrait risquer un surdosage et une intoxication médicamenteuse. D'autres médicaments peuvent avoir des actions physiques similaires, par exemple abîmer le foie, et les prendre en même temps serait dangereux.

Dans les erreurs médicamenteuses les plus courants selon la Haute Autorité de Santé (2015) sont :

- non-respect des bonnes pratiques de prescription : non-respect des règles de prescription, absence de support unique de prescription / administration ou retranscription des prescriptions ;
- non prise en compte du traitement personnel dans l'analyse pharmaceutique ;
- non-respect des conditions de préparation des traitements anticancéreux et des solutions pour nutrition parentérale (dont non-conformité des locaux) ;
- absence d'identification et gestion sécurisée des médicaments à risque (présence d'électrolytes concentrés (potassium, magnésium, calcium) dans les services) ;
- non-sécurisation des conditions de stockage dans les services (armoires, frigos, etc.);
- non-respect des bonnes pratiques d'administration, notamment administration par un professionnel non habilité, administration sur la base d'une retranscription ou non-respect des règles de traçabilité : traçabilité globale, traçabilité a priori, traçabilité tardive, traçabilité de la distribution et non de la prise, etc.

I.3. Analyse et impacts des erreurs de prescription

Les erreurs de prescription peuvent avoir une conséquence directe sur le patient mais cela a aussi un coût en personnel et monétaire pour les établissements de santé. En 2009, l'enquête nationale ENEIS a mis en évidence que 32,9% d'événements indésirables graves (EIG) sont liés aux soins. Parmi ceux liés aux médicaments, 51,2% sont des problèmes évitables. Après résultat de l'enquête, ils ont pu estimer que les problèmes de EIG sont la cause de 315 000 - 400 000 hospitalisations en France, dont 46% évitables. Ce qui signifie plus de personnel, et de matériel de santé à payer.

Avec ce contexte, nous comprenons l'urgence d'améliorer les systèmes dans un centre hospitalier pour éviter et diminuer ces problèmes. Cela va permettre d'optimiser les ressources, et améliorer la qualité de vie générale.

II. Analyse de la BDD réelles des ordonnances (phase de pré-processing)

II.1. Les données du CHU

Le CHU nous a donné accès, par une bulle sécurisée, à un fichier CSV contenant les ordonnances validées pharmaceutiquement par le CHU. Les identités des patients ont été pseudonymisées, donnant à chacun un numéro d'identification.

Ce fichier contient initialement près de 3 millions de lignes, où chaque ligne correspond à un médicament dans une ordonnance, ou bien à plusieurs pris ensemble dans le cas d'une perfusion.

Le fichier donné par le CHU contient par ailleurs 101 colonnes, décrivant des données sur chaque médicament prescrit, sur son dosage précis, sur l'ordonnance associée, et sur le patient en question ainsi que sur ses analyses biologiques.

Le fichier ne contenant que des ordonnances validées pharmaceutiquement, nous ne disposons que d'éléments classés "positifs" (c'est à dire validés par la pharmacie). En absence de prescriptions classées "négatives" nous sommes donc obligés de travailler sur des méthodes d'apprentissage non supervisé. Cependant, nous détaillerons tout de même en partie IV comment nous aurions fait en apprentissage supervisé.

II.2. Gestion des erreurs humaines dans certaines lignes du CSV

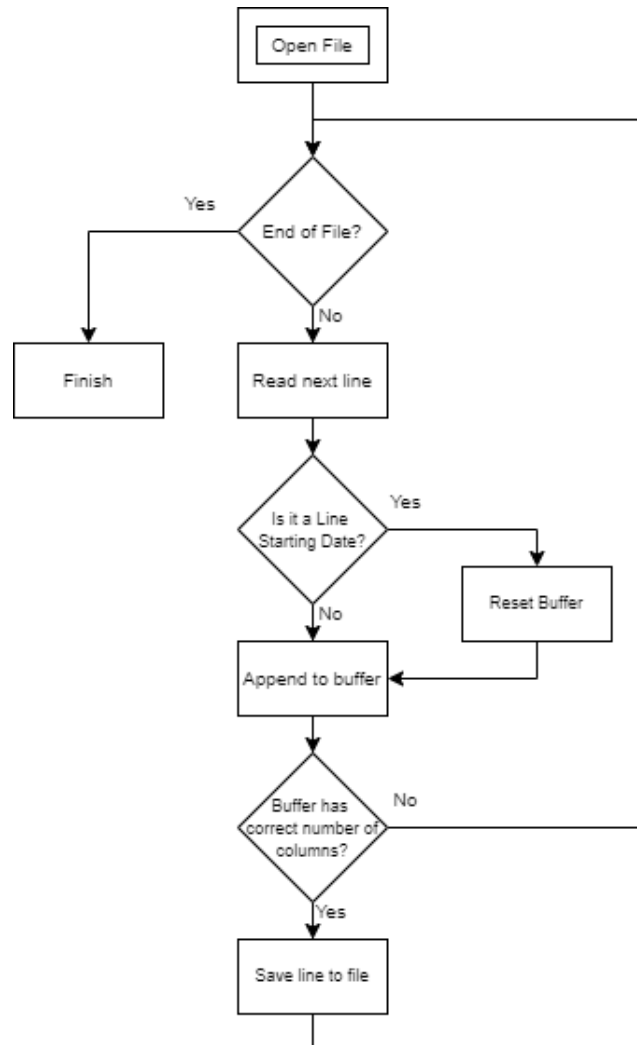
Lorsque nous avons essayé d'importer le fichier de données "securimed.csv" pour la première fois, nous nous sommes rendu compte qu'il était impossible de l'importer dans un DataFrame, car il y avait de nombreuses lignes avec des erreurs. Les données csv sont séparées par des ";" et une erreur qui s'est produite (et qui a rendu l'importation impossible) était que certaines lignes n'avaient pas exactement 100 caractères ";". Certaines des raisons que nous avons identifiées qui pourraient causer ce problème sont : les sauts de ligne dans les commentaires du médecin et/ou utilisation de ";" dans les commentaires du médecin.

Même si ces lignes défectueuses ne représentaient pas une part considérable de l'ensemble des données (jamais plus de 8% du total des lignes), nous les avons tout de même considérées d'une importance unique car ces lignes avaient des instructions détaillées de la part des médecins - et pouvaient donc être des cas "aberrants" importants pour la validation du modèle.

Pour faire face à ces problèmes, nous avons créé le module "*pre_import*" où nous créons des fonctions pour ouvrir les fichiers en tant que texte, les traiter et les nettoyer.

Une fonction permettant de corriger les lignes contenant des erreurs est *write_filtered_csv(filename, savename, treat_errors = False)*. Cette fonction est simple : elle appelle une autre fonction pour lire complètement le fichier et écrire au format CSV dans le fichier savename les lignes qui contiennent le nombre correct de colonnes (dans ce cas, 101 colonnes).

Si la ligne n'a pas le nombre correct de colonnes, elle est écrite dans le fichier *temporary_import_error.csv*. Si le booléen passé est vrai, le programme va essayer de réparer les lignes en suivant cette logique :



Logique de base du code de correction.

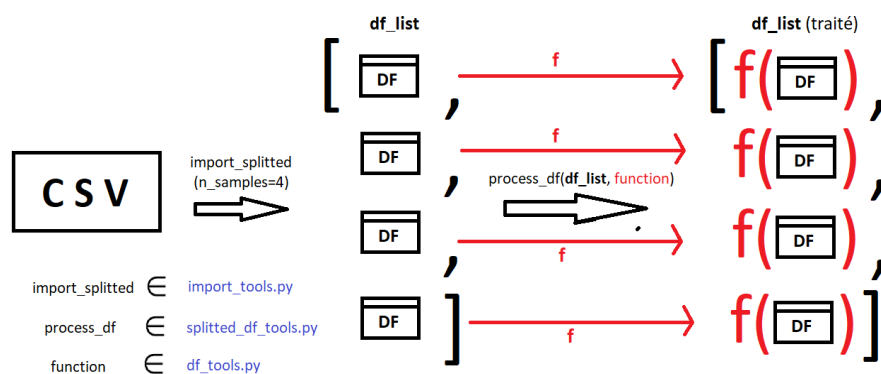
Il y a quelques détails supplémentaires, mais cette méthode a été choisie parce que les lignes qui sont correctes sont assurées d'être écrites correctement dans le fichier, tandis que les lignes défectueuses peuvent être traitées correctement sans risquer d'endommager les lignes correctes, et de plus, comme elles sont enregistrées dans un csv, elles peuvent être observées par un opérateur.

II.3. Gestion d'un dataframe aussi grand

Une fois le fichier csv filtré créé, il était prêt à être traité, mais les serveurs dont nous disposions avaient d'importantes restrictions de mémoire. En effet, il était impossible d'importer la totalité du fichier csv dans un DataFrame, ce qui représentait un problème important puisque l'idée était de traiter un maximum de données afin d'entraîner les modèles avec ces données. C'est pourquoi nous avons été obligés d'ouvrir le csv en plusieurs parties, pour cela nous avons créé un module d'importation "import_tools.py" où nous avons enregistré les différentes fonctions pour importer les données en fonction de ce dont nous avons besoin. L'une des plus importantes est "*import_splitted*", qui importe toutes les données mais divisées en une quantité choisie et qui sauvegarde chaque division dans un

DataFrame et met les DataFrames divisés dans une liste python (ce qui fonctionne sans problème). Cela permet d'importer toutes les données mais génère aussi un problème : puisque nous avons dû créer des fonctions pour traiter les données, il serait compliqué de décider à chaque moment s'il faut créer une fonction qui s'applique à une portion du DataFrame ou à toutes les parties du DataFrame séparément (à toute la liste de DataFrames).

Pour résoudre ce problème, nous avons décidé de faire en sorte que toutes les fonctions de traitement des données supposent qu'elles reçoivent un seul DataFrame et le traitent. Nous avons stocké toutes ces fonctions dans un module "*df_tools*", par exemple les fonctions permettant de supprimer des colonnes, de transformer les données en données numériques, de remplacer les données NaN, de normaliser les données, etc. Ensuite, lorsque vous avez besoin de traiter l'ensemble des données, vous utilisez les outils créés dans un module "*splitted_df_tools*", dont la fonction la plus importante est "*process_df*", qui reçoit une liste python avec des DataFrames et une fonction, qui reçoit elle même un dataframe. La fonction "*process_df*" renvoie finalement une liste de DataFrames tous traité par la fonction prise en argument. Une fois que vous avez effectué suffisamment de filtres et réduit considérablement la quantité de données, vous pouvez essayer d'utiliser la fonction "*mergeDF*" du même module qui tente de les concaténer en un seul DataFrame (à utiliser dans les modèles).



Il est important de mentionner que nous avons donné la priorité aux implémentations "inplace" pour les fonctions `df_tools` afin de ne pas surcharger la mémoire. Dans le cas où une fonction de `df_tools` reçoit plus d'un paramètre, il n'y a aucun problème à utiliser la fonction `process_df` car elle peut être appelée comme suit :

```
df_list = process_df(df_list, lambda df: function(df, param1, param2, ...))
```


II.4. Le regroupement par code ATC, voie d'administration et unité de prescription

Nous avons divisé le DataFrame initial en plusieurs DataFrames, en les séparant par principe actif ainsi que par voie d'administration et unité de prescription. Ainsi les prescriptions sont regroupées par **CODE_ATC**, **VOIE_ADMIN** et **UNITE_PRESC**.

Nous avons préféré travailler sur des clusters fondés sur les principes actifs plutôt que sur les noms commerciaux des médicaments, ces derniers étant peu pertinents pour notre analyse. En effet, deux médicaments de noms différents peuvent contenir la même molécule active. Si nous nous basions sur les noms commerciaux des médicaments, une ordonnance comportant deux médicaments différents contenant la même molécule pourrait être classifiée conforme, mais en réalité causer des problèmes de surdosage ou d'intoxication.

Par ailleurs, nous avons décidé de regrouper les prescriptions autour du CODE_ATC ainsi que des VOIE_ADMIN et UNITE_PRESC, au lieu de le faire simplement autour du CODE_ATC, car pour un même code ATC, on ne peut par exemple pas comparer les quantités de médicament pris en voie orale ou en voie cutanée. En effet, les doses pour une même molécule peuvent beaucoup varier en fonction de la voie d'administration. Par ailleurs, les unités de prescriptions ne seront pas toujours comparables, en effet, comment comparer par exemple l'effet thérapeutique d'une dose en grammes et d'une dose en bouffées. Ne pas distinguer les différentes voies d'administration ou les unités de prescription perturberait les algorithmes de clustering car les comparaisons entre voies d'administration pour un même principe actif ne sont pas toujours possibles.



II.5. Changement des unités

Concernant le poids et la taille des patients, nous nous sommes rendus compte que toutes les valeurs n'avaient pas les mêmes unités. En effet, les poids des patients sont inscrits en grammes ou en kilogrammes, et les tailles sont renseignées en mètres ou en centimètres. Nous avons donc décidé d'uniformiser ces données en mettant tous les poids des patients en kilogrammes et toutes les tailles en mètres.

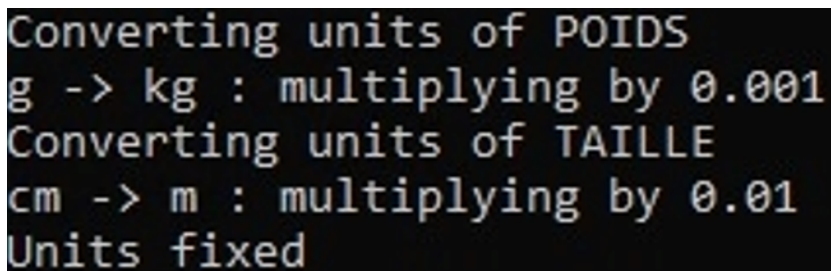
Par ailleurs, les données absentes, dites "NaN", dans les colonnes de taille et poids sont remplacées par la moyenne des tailles et la moyenne des poids. En effet, supprimer les lignes où un "NaN" était présent dans la colonne taille ou bien dans la colonne poids supprimait trop de lignes qui seraient pertinentes pour entraîner le modèle. De plus, supprimer les colonnes taille et poids n'était pas une bonne solution non plus car ce sont

des données précieuses. Ajouter le poids et la taille moyens permet de remplir ces colonnes avec des valeurs neutres pour le clustering.

Il existe également de nombreuses unités différentes dans UNITE_PRESC pour la colonne QTE_PRESC, qui est une colonne très importante car elle indique la dose prescrite d'un médicament. Étant donné la grande variété d'unités dans les données, nous avons été contraints de séparer les DataFrames par UNITE_PRESC également, de sorte que nous traitons chaque médicament comme un tuple de (CODE_ATC, VOIE_ADMIN, UNITE_PRESC), sinon les doses prescrites n'auraient aucun sens entre elles. Idéalement, nous aurions pu convertir toutes les unités en une seule unité (au moins par médicament) afin d'entraîner les modèles uniquement sur CODE_ATC. Toutefois, cela était impossible dans la pratique car, par exemple, certains médicaments étaient exprimés en mg et d'autres en ml, de sorte que pour convertir les unités, il fallait savoir exactement comment convertir une unité en une autre, connaissance qui pouvait être complétée par les professionnels de la santé.

C'est pourquoi lorsque nous avons créé la fonction "*fix_units*" qui standardise les unités, nous avons implémenté qu'elle puisse recevoir comme paramètre un dictionnaire de dictionnaires, où le premier niveau de clés est constitué des colonnes où l'on veut éditer les unités. Le second niveau de clés est constitué des unités à convertir et leurs valeurs sont des tuples du nom de l'unité à convertir et du nombre par lequel la valeur originale doit être multipliée pour être convertie dans la nouvelle unité. Grâce à cela, notre programme pourrait éventuellement s'enrichir d'informations sur la conversion des unités de chaque médicament. Un exemple est présenté ci-dessous :

```
convert_dict = {'POIDS': {'g': ('kg', 1/1000)}, 'TAILLE': {'cm': ('m', 1/100)}}  
fix_units(df, convert_dict=convert_dict)
```



```
Converting units of POIDS  
g -> kg : multiplying by 0.001  
Converting units of TAILLE  
cm -> m : multiplying by 0.01  
Units fixed
```

II.6. La gestion de la fréquence

A l'origine, les données comprenaient différents types de fréquences. Il s'agissait de *en moment*, *en heures*, *par jour*, *toutes les X heures*, *toutes les X minutes*, *toutes les X heures-minutes*, et *autres*. Cependant, le seul type de fréquence qui était réellement présent dans la base de données était *en moment*.









Chaque médicament de chaque ordonnance a une fréquence de prise, par exemple 3 fois par jour. Les données affichent cette information en assignant à chaque ligne le moment de la journée : *matin*, *matinée*, *midi*, *goûter*, *soir*, *coucher* et *nuît*. Pour inclure la fréquence dans notre analyse et apprentissage des modèles, nous avons décidé de standardiser les fréquences.

Tout d'abord, nous avons laissé deux attributs pour représenter la fréquence, le moment de la journée et l'espace temporel entre chaque dose. Cette décision a pris en compte l'importance du moment de la journée pour les médicaments lorsque c'est nécessaire, disons par exemple, ceux avec des composants somnifères, qui doivent être consommés au moment de dormir. Pour les médicaments qui n'ont pas besoin d'être pris dans un moment spécifique, nous avons ajouté un nouveau moment, *aucun*, pour remarquer sa non-importance.






Pour les intervalles temporels entre chaque dose, nous avons compté le nombre de lignes pour chaque médicament dans une ordonnance, et l'avons divisé par le nombre de minutes dans une journée, 1440, retournant ainsi combien de minutes il y a entre chaque prise. Disons qu'un médicament doit être consommé 4 fois par jour à la même dose. Cette méthode nous dit que le patient prend la dose chaque 360 minutes, c'est-à-dire, chaque 6 heures.

Finalement, nous avons défini une ordonnance comme étant les lignes qui partagent les attributs **DATE_INTERVENTION**, **CODE_ATC**, **P_ORD_C_NUM**, **QTE_PRESC**, **UNITE_PRESC** et **VOIE_ADMIN**. Ce choix, selon nous, garantit la bonne séparation des ordonnances avec le minimum de caractéristiques à considérer, ce qui permet une meilleure utilisation de la mémoire au moment de traiter les données.

Pour la mise en œuvre, les données ont été divisées en 20 groupes, en raison de la taille de l'ensemble de données. Ensuite, dans chaque groupe, les médicaments ont été regroupés en fonction des attributs susmentionnés et enregistrés dans un fichier .csv avant de passer au groupe suivant. Avec cette opération, nous n'avons eu aucun problème d'utilisation de la mémoire, car seulement environ 150 000 lignes ont été traitées en même temps. Enfin, tous les fichiers .csv ont été fusionnés en un seul grand fichier, créant ainsi la nouvelle base de données que nous avons utilisée.

Med	Moment	Fréquence (différents formats)	Etc
	Soir	XXX	...
	Nuit	XXX	...
	Matin	XXX	...
	Matin	XXX	...
	Nuit	XXX	...
	Soir	XXX	...
	Soir	XXX	...
	Nuit	XXX	...

XXX -> f (en min)

Med	Moment	Fréquence	Etc
	Aucun	(24*60)/3	...
	Matin	24*60	...
	Soir	24*60	...
	Aucun	(24*60)/2	...
	Nuit	24*60	...

~3M lignes réduites en ~1.8M lignes

II.7. Les colonnes supprimées

Après étude des données, nous avons trouvé que certaines colonnes n'étaient pas pertinentes pour notre étude. C'est le cas de toutes les colonnes qui enregistrent les dates de prescriptions. Les colonnes d'unités ne sont d'autre part plus utiles grâce à l'unification des unités effectuée dans le II.4.

À l'aide de plusieurs fonctions que nous avons créées, nous comptons les valeurs distinctes de chaque colonne et les enregistrons dans un fichier json pour les consulter à tout moment. Grâce à cela, nous avons pu remarquer que toutes les colonnes d'unités (à l'exception de TAILLE, POIDS et QTE_PRESC) n'avaient qu'une seule unité, donc l'existence de ces colonnes n'avait pas de sens. Ce fichier que nous avons créé nous a également permis, de manière générale, de discerner l'importance de chaque colonne pour voir ce que nous devons garder.

Nous avons également supprimé la colonne de **CODE_UCD** et **NOM_COMMERCIAL**, car pour l'analyse nous utiliserons le **CODE_ATC** qui définit le principe actif, comme expliqué en II.6.

Par ailleurs, les identifiants redondants concernant les patients et les ordonnances, comme les identifiants-patients de la sécurité sociale redondants avec les identifiants-patients du CHU, ont été supprimés car inutiles ici.

Finalement l'idée est de garder seulement les données qui concernent la posologie et l'état du patient respectif.

III. Clustering des ordonnances (machine learning non supervisé)

III.1. Descriptions des différentes stratégies

Pour réaliser le regroupement des données il existe principalement trois méthodes : regroupements experts, informatiques et statistiques.

Le regroupement expert se base sur la connaissance du contexte des données. Ici cela voudrait dire que nous aurions besoin de l'expertise d'un pharmacien pour établir un regroupement et une implémentation qui suit cette connaissance scientifique. Par exemple, faire une classification qui va suivre la hiérarchie des codes ATC. De cette façon on peut avoir des niveaux de hiérarchie qui peuvent améliorer les modèles de classification.

Le regroupement informatiques c'est d'explicitier des règles d'associations logiques pour la classification. De cette façon on encadre les classifications, ce qui permet après pour le modèle de suivre ces règles logiques. Les règles étant normalement définies avec un sens d'interprétation des données.

Dans le regroupement statistique, c'est de trouver des corrélation entre les données pour faire des clusters. Dans ce regroupement on ne fait pas d'interprétation des données, on prend juste des valeurs numériques. Dans notre cas, ça serait utiliser la corrélation entre l'état du patient et le dosage administré. C'est ce regroupement qu'on utilisera pour nos modèles, à cause de notre ignorance dans le domaine pharmaceutique. Mais on proposera une flexibilité dans les modèles pour pouvoir insérer du regroupement expert et améliorer les résultats finals.

III.2. Les algorithmes de machine learning pour étudier chaque prescription individuellement

Nous avons mis au point plusieurs méthodes de machine learning destinées à étudier chaque prescription individuellement. Ces méthodes se trouvent dans notre module "*machine_learning.py*", notamment un modèle de k-means clustering, un modèle de hierarchical clustering, une régression linéaire et un ridge avec Crossed Validation.

L'algorithme du k-means clustering prend en entrée les données à clusteriser et un nombre de clusters à former.

Le modèle de hierarchical clustering a une différence majeure avec le modèle k-means : il n'a pas besoin de prendre en argument le nombre de clusters que l'on attend. En effet, il calcule lui-même le nombre de clusters. Il prend en entrée une distance à partir de laquelle deux clusters ne doivent pas être fusionnés. Avec un set de données contenant des prescriptions rejetées pharmaceutiquement, ou avec le feed-back d'un pharmacien, nous pourrions tester notre modèle hierarchial et affiner la distance à donner en entrée.

Concernant la régression linéaire, on prédit la dose prescrite à partir des valeurs des autres colonnes de la ligne à analyser. En comparant la dose réelle prescrite et la dose prédite, on pourra déterminer si une ligne est valide ou si on doit envoyer une alerte. Par ailleurs, si une alerte est lancée, nous pourrions envoyer la dose que l'algorithme a prédite, et qui pourrait être la correction de la dose initialement prescrite.

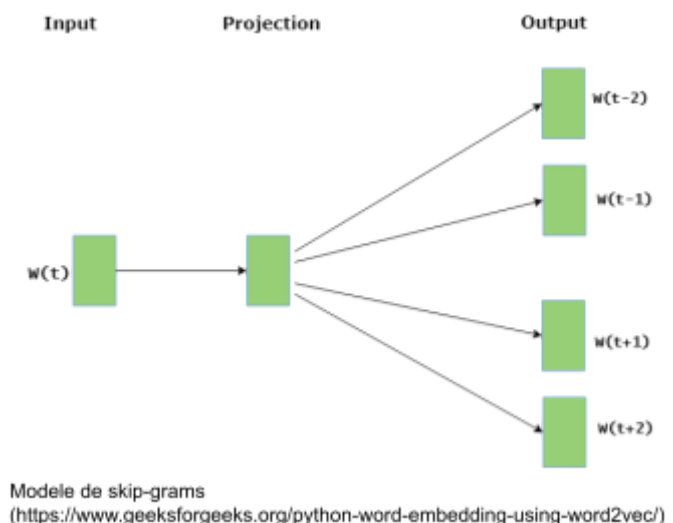
En conclusion, tous les modèles ont des hyper-paramètres réglables. Cela signifie que nous pourrions les ajuster plus finement pour chaque médicament. Cependant, avec nos ressources actuelles nous ne pouvons pas tester tous les paramètres pour tous les médicaments, cela serait bien trop long. Si nous disposions de données d'ordonnances rejetées par la pharmacie, nous pourrions faire de l'apprentissage supervisé sur nos modèles pour régler les paramètres et mieux ajuster les modèles aux données.

III.3. Word2Vec

Pour analyser si l'insertion d'un médicament dans une ordonnance déjà établie va provoquer un conflit avec les autres médicaments présents, nous avons utilisé l'algorithme de Word2Vec Skip Gram.

Word2Vec est un algorithme de traitement de texte pour l'apprentissage d'association des différents mots dans un texte établi. Il consiste principalement à prendre tous les mots présents dans le texte pour créer un vocabulaire, et ensuite voir pour chaque mot, les mots voisins qu'il a. Ce voisinage définit la relation entre le mot que nous analysons et ses voisins. Ainsi, on construit une matrice de relations qui va définir toutes les relations avec des paramètres qui seront entraînés grâce à un réseau de neurones. Ainsi on pourra finalement avoir les similitudes entre chaque binôme grâce à un score.

Dans Word2Vec il y a deux approches, l'une est le CBOW, qui grâce à un contexte de mots peut prédire les mots le plus appropriés aux contextes. L'autre est le **skip grams**, qui marche dans l'autre sens : on donne un mot et le modèle nous donne comme sortie le contexte.



Pour l'application avec les ordonnances, nous avons construit un vocabulaire avec tous les codes ATC qui étaient présents dans le fichier. On les a regroupés dans des phrases, où chaque phrase correspond à une ordonnance selon son ID. De cette façon, chaque code ATC va être associé avec d'autres médicaments seulement s'il est dans la même ordonnance que lui. Pour cela nous avons construit un fichier avec tous les codes ATC dans des phrases pour après pouvoir tokeniser toutes les phrases et mots dans un vecteur. On a utilisé la librairie Gensim qui comporte déjà un modèle Word2Vec skip gram pour analyser un texte.

Grâce au modèle, après lui avoir donné le fichier texte, on peut lui demander le contexte de chaque code ATC et il donne un contexte avec les dix code ATC plus proche de l'input. Ceci nous permettra de construire une application pour le CHU : lorsqu'un médecin voudra insérer un nouveau médicament dans une ordonnance existante, nous pourrions analyser le contexte de son principe actif et voir si les autres médicaments qui sont dans

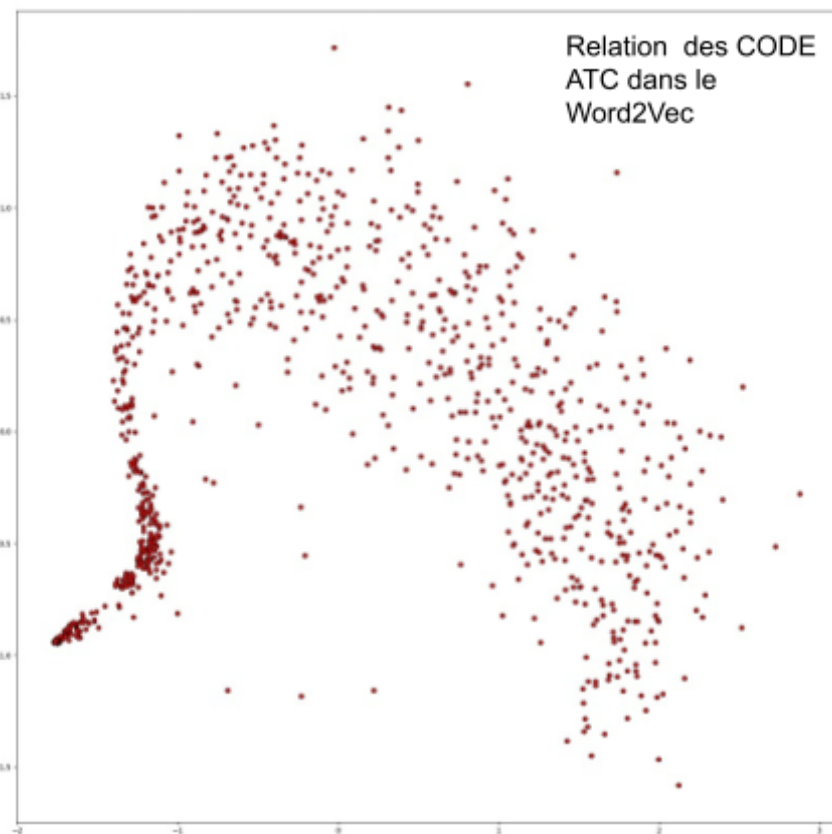
l'ordonnance se trouvent dans le contexte du word2vec. Si ce n'est pas le cas alors l'application donne une alerte, pour indiquer qu'il y a un binôme de médicaments qui ne sont pas proches entre eux.

```
Most similar with: 'N05BA06' Skip Gram : [('c01bd01', 0.18890608847141266),
('r03al04', 0.18855656683444977), ('b01ac06', 0.16072481870651245), ('a06ad
65', 0.15923376381397247), ('s01ee01', 0.13725270330905914), ('c03da01', 0.1
27894327044487), ('a11cc05', 0.1230086162686348), ('c09ca04', 0.085460588335
9909), ('a10ab04', 0.06798446178436279), ('b01ab01', 0.03364056721329689)]
```

Exemple de contexte pour le code: "N05BA06"

III.4. Simulation et résultats

Pour le Word2Vec nous avons construit le modèle, et l'avons entraîné grâce au fichier texte avec les codes ATC correspondant. Grâce à une réduction de dimension (PCA) de la matrice de relations du vocabulaire, on peut observer la relation entre chaque code ATC. Dans la figure chaque point est un code ATC, et nous observons qu'il y a un cluster ou plusieurs codes sont regroupés. Ce qui signifie que ce sont des médicaments souvent associés ensemble. Mais il y a



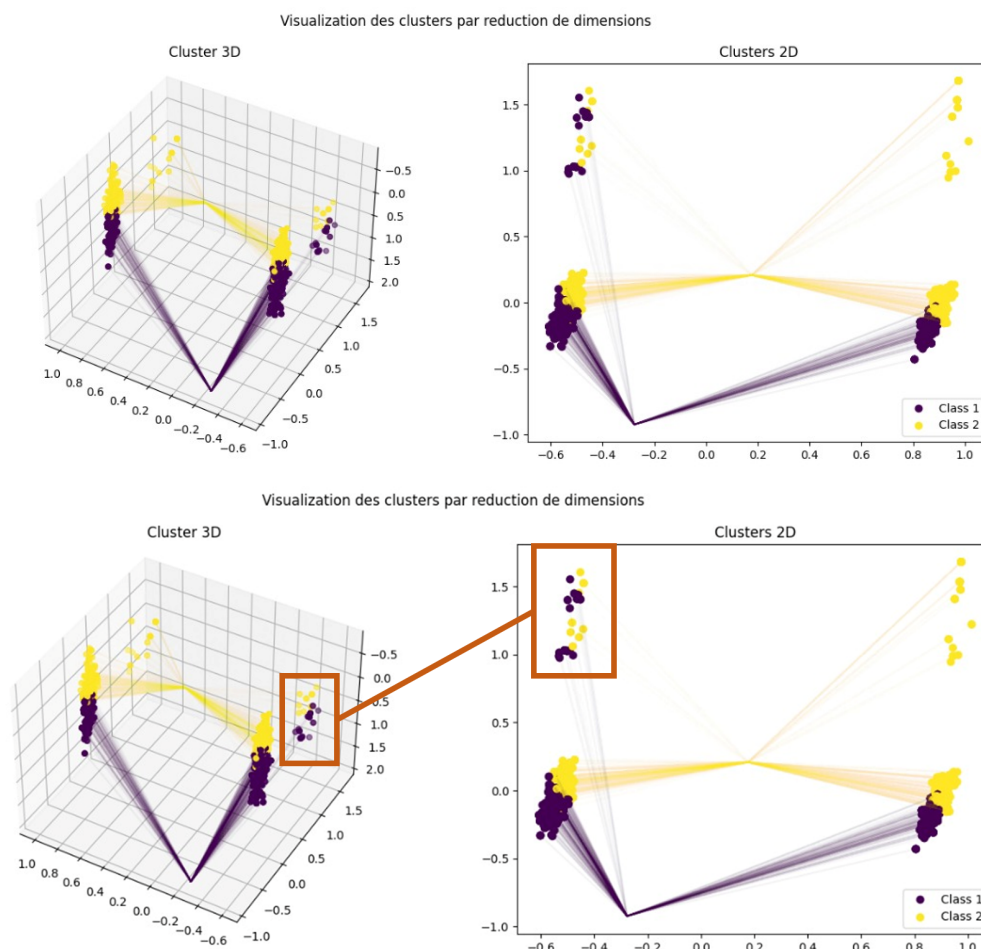
des outliers qui se retrouvent plus seul et dispersés. Ces résultats nous permettent de comprendre que pour les médicaments plus proches entre eux, il sera plus facile de trouver une similitude, et donc pouvoir décrire une bonne ordonnance. Mais pour les ATC plus éloignées la similitude est plus floue.

Comme expliqué précédemment, nous avons également créé des algorithmes d'apprentissage automatique (basés sur sklearn) pour regrouper les ordonnances. Ces

algorithmes sont complétés par des fonctions de visualisation utilisant matplotlib et seaborn, et donc plus facilement lisibles par l'opérateur.

Comme il existe un nombre conséquent de combinaisons entre les modèles et les médicaments à tester pour la visualisation, nous avons décidé de faire la visualisation pour le modèle k-means avec le doliprane.

Comme chaque point de données a une dimension de 37 (sur les 37 colonnes utilisées), nous avons utilisé l'ACP pour les représenter en 3D et en 2D. Cela a donné des résultats très intéressants, car nous avons pu constater que parfois, des clusters qui ne sont pas "évidents" en 2D peuvent être plus facilement expliqués en 3D, comme l'exemple ci-dessous :



Création des clusters par le modèle k-means pour l'utilisation de Doliprane.

Les lignes sont connectées au "centre" du cluster. Ici on peut voir le comportement de cet ensemble de points dans une représentation 3D et 2D.

Comme nous pouvons le constater, lorsque nous voyons les points en 2D, il y a quelques points violets au-dessus de la ligne $y=0$, plus dominés par les points jaunes. Cependant, lorsque nous voyons la dimension 3D, nous réalisons que ces points violets sont plus proches dans la dimension (x,y,z) .

On ne peut pas visualiser les clusters par le modèle k-means pour tous les autres médicaments car cela prend du temps, mais on voit bien que sur l'exemple du doliprane, il y a une formation de clusters assez cohérents.

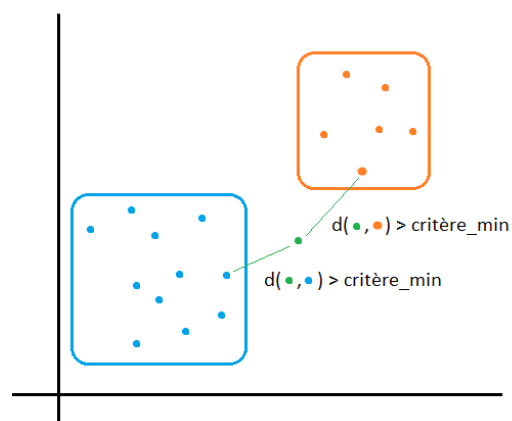
IV. Application dans un système d'alerte

Nous avons créé un module *"warning_system.py"* contenant toutes les fonctions relatives au système d'alertes. La première fonction du module *"prepare_line"* permet de préparer une ligne de prescription, en lui appliquant les mêmes traitements appliqués aux données initiales utilisées pour entraîner les modèles.

Pour chaque modèle de machine learning, nous avons écrit une fonction associée, qui prend en argument la ligne de prescription à analyser et un dictionnaire contenant des informations du modèle entraîné pour les données ayant les mêmes **CODE_ATC**, **VOIE_ADMIN** et **UNITE_PRESC** que la ligne que l'on souhaite analyser. Cette fonction renvoie un booléen : True signifie qu'on envoie une alerte et False signifie que selon ce modèle, la ligne de prescription est bonne.

Les fonctions *"warning_lr"* et *"warning_clusters"* permettent de déterminer si la prescription à étudier est validée individuellement. La fonction *"warning_word2vec"* permet de déterminer si la prescription étudiée est compatible avec les autres médicaments que le patient prend déjà à ce moment-là.

Pour les modèles de clustering, on détermine quel centre, parmi les centres des clusters du modèle concerné, est le plus proche du point représentant la ligne de prescription analysée. Puis on calcule le rayon du cluster concerné, c'est-à-dire la distance entre le centre et le point le plus éloigné appartenant au même cluster. On calcule aussi la distance entre le point à analyser et le centre du cluster. Si cette dernière distance est inférieure au rayon du cluster, c'est-à-dire si le point est à l'intérieur de la sphère délimitant le cluster, alors la prescription est considérée comme bonne. A contrario, si le point est en dehors de la sphère, on renvoie False, équivalent à une alerte.



Pour le modèle de Régression Linéaire, à partir de la ligne à étudiée auquel on a retiré la dose prescrite, on détermine avec le modèle la prédiction la dose prescrite. Puis, on observe si la vraie dose prescrite appartient à un intervalle de tolérance autour de la dose prédite. Les bornes de cet intervalle sont définies par la dose prédite +/- l'écart-type de la colonne de quantité prescrite. Si la vraie dose prescrite appartient à l'intervalle, la prescription est considérée comme bonne, sinon on renvoie False, ce qui est équivalent à une alerte.

Pour le modèle de Word2vec, comme nous n'avons pas de données d'ordonnance incorrectes, nous n'avons pas réussi une validation du modèle. Mais le principe est le suivant : on regarde quels sont les médicaments que le patient prend déjà (enregistré dans un dossier "Patients") et on regarde s'ils sont dans le contexte du médicament à analyser, pour savoir si le nouveau médicament serait compatible avec le traitement que le patient prend déjà. Si le traitement actuel du patient n'est pas dans le contexte du nouveau médicament à étudier; alors on renvoie False, c'est-à-dire une alerte.

De plus, nous avons codé une fonction plus générale, appelée *"warning_system"*, qui prend en argument la ligne prescrite à analyser ainsi qu'un mode, puis qui renvoie un booléen. Cette fonction appelle les fonctions présentées précédemment pour les modèles k-means, hierarchical clustering et régression linéaire, et garde les booléens qu'elle renvoie. Ensuite, en fonction du mode choisi, la fonction renvoie un booléen final pour indiquer si la prescription est bonne ou si il y a une alerte. Les modes sont les suivants : le mode 'any' renvoie une alerte si un seul des modèles renvoie une alerte. Le mode 'two True' renvoie une alerte si parmi les trois modèles testés, deux renvoient une alerte. Le mode 'all' retourne une alerte si tous les modèles renvoient une alerte.

Comme nous n'avons pas de données sur les ordonnances erronées, il n'y avait pas de véritable moyen de tester le programme pour évaluer ses résultats. Même si le programme fonctionne bien et ne comporte aucune erreur, il est impossible de déterminer l'exactitude des modèles pour un cas plus général. Cela aurait pu être mieux si nous avions eu des données erronées pour confirmer qu'une alerte est bien levée pour l'entrée de ces données.

Pour "essayer" de tester le programme, nous avons sauvegardé plusieurs modèles au format json dans un dossier Models (pour les modèles tels que K-means, Agglomerative Clustering, linear regression, et ridgeCV), où le nom de chaque fichier correspond à la "key" qui représente un médicament. Pour le modèle Word2Vec, celui-ci était sauvegardé et prêt à être consulté ainsi que l'historique des médicaments que chaque patient a pris (ceci était sauvegardé dans le dossier Patients). En général, l'idée est que les fonctions du module *"warning_system.py"* n'utilisent pas les données pour créer les alertes, mais seulement les informations des modèles entraînés déjà enregistrés. En ce sens, le programme a bien marché.

Malgré ce qui précède, pour essayer de tester le programme, nous avons importé 100 lignes de doliprane, et essayé de délivrer le programme d'alerte qui utilise le modèle K-means (du module *machine_learning.py*), et enregistré les résultats délivrés par le programme d'alerte. Ensuite, nous reproduisons la même chose mais en multipliant QTE_PRESC par 3 pour simuler une erreur dans l'entrée de ces données et ainsi simuler une ligne de données erronée, et nous sauvegardons également les données. Enfin, en supposant que le premier processus correspondait à des lignes correctes et que le second correspondait à des lignes erronées, nous pouvons créer la matrice de confusion correspondante, où nous pouvons voir que des résultats satisfaisants sont obtenus.

```
>>> good_false = good.count(False)
>>> good_true = good.count(True)
>>> bad_false = bad.count(False)
>>> bad_true = bad.count(True)
>>> cm = np.array([[good_false,good_true],[bad_false,bad_true]])
>>> cm
array([[96,  4],
       [ 9, 91]])
```

V. Apprentissage supervisé

Même si nous n'avons pas obtenu les données des ordonnances avec réserve, nous vous proposons dans ce paragraphe les étapes et algorithmes d'apprentissage supervisé en utilisant les variables/clusters/résumés construits en III) permettant de construire un modèle discriminant des ordonnances avec ou sans réserve.

Nos solutions proposées utilisent des méthodes d'apprentissage non supervisé. Cela est dû au fait que nous n'avons pas accès à des données d'ordonnances non valides. Si bien que nous avons développé des outils qui apprennent seulement avec des 'positifs', et ceux-ci peuvent être améliorés considérablement une fois que nous disposerons des autres données. Dans cette section, nous allons présenter les possibles démarches que nous aurions faites, si les données avaient été disponibles.

Pour l'apprentissage utilisant des clusters, disposer des données classées comme 'non conformes' nous permettrait de diviser les clusters en deux. Dans l'un on trouvera un cluster des faux, et l'autre des données vraies. Ce qui permet finalement de faire une classification des données. Ceci a pour objectif de ne pas seulement utiliser les distances aux plus proches voisins, tout en pouvant réaliser le modèle des plus proches voisins et voir combien de voisins sont vrais et combien sont faux autour d'un point. Ainsi, nous pourrions classer toute ordonnance de façon binaire. De plus, une fois l'application implémentée dans un environnement réel, le feedback du docteur nous permettrait d'insérer de nouvelles données, également classifiées, pour faire les clusters les plus précis possible.

Concernant les interactions inter-médicaments, si on dispose des ordonnances avec des médicaments qui ne doivent pas être ensemble, on pourrait réaliser également avec la méthode de word2vec, un vocabulaire des codes ATC qui ne doivent pas être ensemble. De cette façon, une fois qu'un médecin veut insérer un nouveau médicament dans une ordonnance, on va confronter les deux contextes donnés par notre modèle: le premier qui est le contexte des code ATC plus proches et valides, et le deuxième qui est le contexte des code ATC plus proches et non valides. Après cette confrontation, il sera possible de prendre une décision pour valider ou non le médicament dans cette ordonnance.

Nous pensons que ces deux approches vont permettre de rendre nos modèles plus précis, et d'avoir de meilleurs résultats. Également disposer des données qui ne sont pas validées par un médecin permet de tester si l'application peut catégoriser de bonne façon les ordonnances.

VI. Vos ressentis et analyse des conférences des intervenants extérieurs

Nous tenions tout d'abord à remercier tous les intervenants extérieurs, Chloé DELANNOY-ROUSSELIÈRE, Michèle VASSEUR, Pascal ODOU, Régis BEUSCART, Aurelien VANNIEUWENHUYZE, Jean-Marie RENARD, Grégoire FICHEUR, Marc BROUCQSAULT, ainsi que les professeurs de l'électif IAS, Slim HAMMADI, Yannick DUSCH, Hayfa ZGAYA, Pascal YIM, Sarah BEN OTHMAN et Claire BELART pour leurs conférences et leurs enseignements précieux.

Dès la première semaine, grâce aux interventions de Chloé DELANNOY-ROUSSELIÈRE, Michèle VASSEUR et Pascal ODOU, nous avons découvert et compris le contexte et l'enjeu de notre Fil Rouge, notamment l'enjeu de diminuer la iatrogénie médicamenteuse. Nous avons appris le fonctionnement de la pharmacie du CHU de Lille et son projet de pharmacie automatisée à l'horizon 2025. Ces conférences nous ont aussi permis de mieux appréhender le parcours d'un médicament et les risques d'erreurs lors de leur prescription, préparation ou distribution, ainsi que les systèmes de sécurisation pour diminuer ces erreurs, comme les armoires à pharmacie automatisées.

Par ailleurs, la visite de la pharmacie du CHU de Lille a été une belle opportunité pour nous de voir et comprendre le fonctionnement de la pharmacie, et de poser nos questions à Chloé DELANNOY-ROUSSELIÈRE qui nous a guidés tout au long du Fil Rouge.

Les interventions des professionnels de la santé, dont ceux cités précédemment ainsi que Régis BEUSCART, Jean-Marie RENARD et Grégoire FICHEUR nous ont permis de cerner différents problèmes actuels dans le monde de la santé et le potentiel de l'IA dans ce domaine. Ils nous ont également bien fait comprendre que les données médicales sont des données sensibles, raison pour laquelle nous avons par exemple signé une charte dans le cadre du Fil Rouge.

Avoir des conférences de la part de professionnels de la Data dans le monde de l'entreprise, notamment Aurélien VANNIEUWENHUYZE, directeur de l'entreprise Qstom-IT et Marc BROUCQSAULT, président d'ALTAO, était très intéressant. En effet, cela nous offrait une vision différente de la vision académique de l'IA, de ses enjeux et de son futurs pour les entreprises. Finir l'électif en essayant de trouver une idée de système d'IA à vendre à M. BROUCQSAULT était très stimulant et amusant pour nous, d'autant plus qu'il a dit qu'il nous achèterait notre idée. Cela nous a permis de mettre en pratique notre créativité et pour certains notre goût de l'entrepreneuriat.

Conclusion et perspectives

La santé étant un domaine complexe, qui a plusieurs acteurs qui interagissent entre eux, et c'est aussi un domaine avec des problèmes profonds. Un des problèmes est avec la pharmacie, c'est la prescription de médicaments aux patients. Réaliser des ordonnances, c'est mettre en jeu une logistique matérielle et personnelle coûteuse pour la santé. Les principaux problèmes sont dans la validation des doses, la mauvaise administration, et la non prise en compte du traitement suivi par le patient. Ces problèmes vont avoir une conséquence directe sur le patient, en allongeant son séjour, ou en lui imposant une nouvelle hospitalisation. Ces problèmes, dans un pourcentage élevé, sont évitables.

Une solution proposée est d'utiliser les données d'un centre hospitalier, pour en ressortir un modèle d'intelligence artificielle, grâce à du machine learning pour prédire les ordonnances qui doivent être révisées. Dans notre étude nous proposons deux méthodes, une par la clusterisation des ordonnances, et l'autre par du traitement de texte pour étudier les interactions médicamenteuses au sein d'une même ordonnance. La première va séparer les données. Ensuite, pour chaque code ATC on réalise des clusters, ce qui permettra ensuite lorsqu'on voudra prédire si un point de prescription est bien valide pour un patient si il est proche d'un cluster. Dans le cas où il se trouve éloigné, alors on émet une alerte. Le deuxième prendra tous les codes ATC des médicaments présents dans une ordonnance pour faire une analyse du texte, et voir la similitude et les relations des codes. L'objectif est de retrouver le contexte de chaque code ATC, et pouvoir définir quels codes ATC sont plus proches entre eux.

Après un traitement des données que nous avons reçues de la part du CHU de Lille, nous avons réalisé les modèles présentés pour avoir des résultats des différents clusters suivant différents algorithmes et pouvoir définir des relations entre les codes ATC. Ceci permettra de le mettre en place dans un contexte réel pour avoir une validation de la part d'un médecin.

Ainsi à cause des données d'ordonnances validées avec lesquelles on a travaillé, nous avons seulement pu faire des algorithmes d'apprentissage non supervisés. Mais si on disposait de données avec des labels qui indiquent si l'ordonnance est correcte ou pas, nous pourrions implémenter aussi des algorithmes supervisés qui vont aider et améliorer la performance du travail. Également, si les modèles sont implémentés dans un contexte réel, après chaque prédiction il aurait un feedback d'un pharmacien pour faire un apprentissage continu. Finalement ce modèle va permettre, grâce à l'implantation d'une pharmacie robotisée de demain, d'avoir une meilleure optimisation des ressources ainsi que d'augmenter la sécurité des patients.

Bibliographie

Chin, Y. P. H., Song, W., Lien, C. E., Yoon, C. H., Wang, W.-C., Liu, J., Nguyen, P. A., Feng, Y. T., Zhou, L., Li, Y. C. J., & Bates, D. W. (2021). Assessing the International Transferability of a Machine Learning Model for Detecting Medication Error in the General Internal Medicine Clinic: Multicenter Preliminary Validation Study. In *JMIR Medical Informatics* (Vol. 9, Issue 1, p. e23454). JMIR Publications Inc. <https://doi.org/10.2196/23454>

Chris Manning (2021) CS224n: Natural Language Processing with Deep Learning, Stanford.

url:<https://web.stanford.edu/class/cs224n/materials/Gensim%20word%20vector%20visualization.html>

Corny, J., Rajkumar, A., Martin, O., Dode, X., Lajonchère, J.-P., Billuart, O., Bézie, Y., & Buronfosse, A. (2020). A machine learning–based clinical decision support system to identify prescriptions with a high risk of medication error. In *Journal of the American Medical Informatics Association* (Vol. 27, Issue 11, pp. 1688–1694). Oxford University Press (OUP). <https://doi.org/10.1093/jamia/ocaa154>

Direction de la recherche, des études, de l'évaluation et des statistiques (2009). Enquête nationale sur les événements indésirables liés aux soins (ENEIS)

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Muyile, K. M., Gentens, K., Dupont, A. G., & Cornu, P. (2021). Evaluation of an optimized context-aware clinical decision support system for drug-drug interaction screening. In *International Journal of Medical Informatics* (Vol. 148, p. 104393). Elsevier BV. <https://doi.org/10.1016/j.ijmedinf.2021.104393>

Sumedh Kadam (2018) Python Word Embedding using Word2Vec, Geeksforgeeks. url: <https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/>

Haute Autorité de Santé (2015) Sécuriser la prise en charge médicamenteuse en établissement de santé. url: https://www.has-sante.fr/jcms/c_2574453/fr/securiser-la-prise-en-charge-medicamenteuse-en-etablissement-de-sante