Rapport de stage

Intelligence artificielle et dépistage de maladies

Julien Levarlet

Année 2020–2021

Stage de deuxième année réalisé dans l'entreprise CHRU Nancy-Brabois

Maître de stage : Dr Marc Merten

Encadrant universitaire : Isabelle Heudiart

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Julien Levarlet

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31916825

Année universitaire : 2020–2021

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Intelligence artificielle et dépistage de maladies rares du métabolisme détectées par chromatographie gazeuse couplée à la spectrométrie de masse

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Nancy, le 15 août 2021

Signature :

Rapport de stage

Intelligence artificielle et dépistage de maladies

Julien Levarlet

Année 2020–2021

Stage de deuxième année réalisé dans l'entreprise CHRU Nancy-Brabois

Julien Levarlet
20, rue Adolphe Le Hir
29 000, Quimper
06 37 39 45 77
julien.levarlet@telecomnancy.eu

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

CHRU Nancy-Brabois
Rue du Morvan
54 500, Vandoeuvre-lès-Nancy
03 83 15 30 30

Maître de stage : Dr Marc Merten

Encadrant universitaire : Isabelle Heudiart

Remerciements

Je tiens à remercier mon maître de stage Dr Marc Merten pour ce sujet de stage très intéressant. Son aide m'a été précieuse par le temps passé à m'expliquer les concepts de biologie et de métabolisme qui étaient nouveaux pour moi, ainsi que la confiance qu'il m'a accordée sur la réalisation du projet.

Je tiens également à remercier Nicholas Jallan, expert externe, pour ses conseils très utiles à la fois sur le traitement des données et l'apprentissage automatique. Cela m'a fait gagner beaucoup de temps tout au long de ce projet.

Je remercie Dr Jean Marc Vuillaume pour son aide et sa présence attentive tout au long du stage.

Je remercie Dre Elise Jeannesson d'avoir pris le temps de diagnostiquer d'anciens chromatogrammes utiles au projet et pour toutes les ressources qu'elle a portées à ma connaissance à propos de la chromatographie gazeuse couplée à la spectrométrie de masse.

Enfin je souhaite remercier toutes les personnes du laboratoire qui ont su me réserver un bon accueil et m'aider au moindre problème rencontré.

Table des matières

1 Introduction

Ce rapport présente mon stage de deuxième année à Télécom Nancy. Ce stage s'est déroulé au CHRU de Nancy-Brabois dans le bâtiment de biologie médicale et biopathologie, pour une durée de 8 semaines, du 31 mai au 23 juillet 2021.

Le projet qui m'a été confié, est de mettre en place un programme d'aide à la décision lors de la lecture d'analyses d'urines pour la détection de maladies rares du métabolisme. En effet, la lecture d'analyse d'urine demande beaucoup d'expérience et de temps pour détecter les éventuels symptômes d'une maladie. Or une erreur ou un retard de diagnostic d'une maladie peut être très problématique, car le patient ne pourra pas recevoir le traitement adéquat.

L'objectif est de faire gagner du temps au biologiste en faisant un pré-diagnostic indiquant si le chromatogramme est normal ou non et dans un second temps à quelle catégorie de problème correspond ce chromatogramme. Cela permettra de passer plus rapidement sur des cas simples et d'affecter plus de temps sur les cas complexes.

2 Problématique

2.1 Présentation du laboratoire de biologie médicale et biopathologie

Le laboratoire de biologie médicale et biopathologie, se situe au sein de l'hôpital pour adulte de Nancy-Brabois. Ce laboratoire, actif depuis le 1^{er} juillet 2019, regroupe un grand nombre de spécialités, dont celle de biochimie-biologie moléculaire et nutrition où s'effectue mon stage. Il comporte d'autres spécialités comme la génétique, la pharmacologie, l'immunologie ou la microbiologie.
De nombreuses machines, comme celle de chromatographie gazeuse couplée à la spectrophotométrie de masse (GCMS) sur laquelle j'effectue mon stage, permettent d'y effectuer des analyses fiables. Le laboratoire est d'ailleurs un centre de référence, où des hôpitaux extérieurs envoient des prélèvements.

Photo du laboratoire issue de recrutement.chru-nancy.fr

2.2 Contexte du stage

L'aide au diagnostic de maladies métaboliques via un algorithme d'Intelligence Artificielle (IA) est un domaine récent. Si l'on s'intéresse aux dates de publication des articles scientifiques sur le thème, on se rend compte que le sujet a été peu traité, mais qu'il commence depuis trois ans à prendre de l'ampleur. Mais ce sujet reste beaucoup moins traité que la reconnaissance faciale par exemple (voir figure 2.1).
Les résultats de recherche proviennent du site PubMed.

Pourtant, ce domaine est demandeur d'outils aidant au diagnostic, car ce type d'analyse demande des années d'expérience pour être efficace dans leur lecture. Une personne ayant peu l'habitude de ce genre de diagnostic pourra passer parfois passer une heure sur une analyse pour finalement conclure que rien d'anormal n'est visible. Le problème étant qu'il y a une grande diversité de profils *normaux* comme *anormaux*.

Les maladies rares étant difficiles à détecter, il est important de mettre en place un outil qui aide le biologiste à trouver le plus précisément et le plus efficacement la présence de maladies.

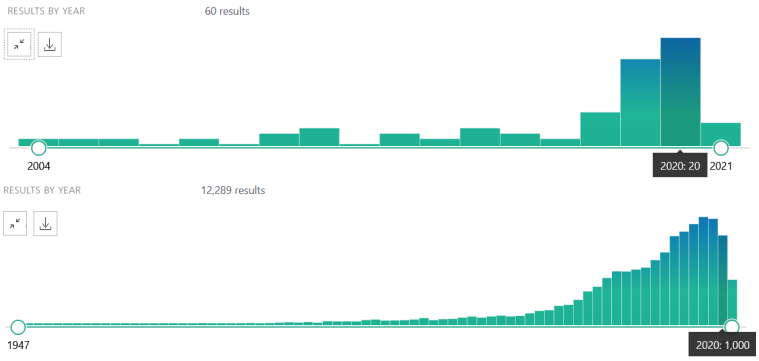

FIGURE 2.1 – Comparaison du nombre d'articles par an en cherchant par mot clé, "inborn error of metabolism machine leaning" pour la figure du dessus et "facial recognition" pour la figure du dessous

2.3 Description du projet

Le but du projet est de déterminer quels sont les informations utiles à extraire du logiciel, et de mettre en place un algorithme IA permettant de faire un diagnostic, le plus précis possible à partir des données recueillies par la machine de GCMS et la base de données du CHRU.

Pour réaliser le programme d'IA, il sera possible de récupérer les données de GCMS ainsi que l'interprétation qui a été faite entre 2019 et 2021, durée d'utilisation de la machine actuelle. Cela pourra être utile dans le but d'avoir une base de données d'entrainement pour cet algorithme.

Le résultat de ce stage se présentera sous la forme d'une application dans laquelle il sera possible d'indiquer un chromatogramme. Cette application, à partir de ce fichier, devra donner une classification de celui-ci dans une des catégories qui auront été définies au cours du stage (comme le type de maladie).

Pour la réalisation de ce projet, le langage python sera utilisé, car il permet d'utiliser un grand nombre de modules qui pourront être utiles au cours de ce projet à la fois sur la lecture des données ainsi que sur la partie IA.

Pour mener à bien le projet, le Dr Marc Merten apportera son aide pour la compréhension de la partie biologie du stage (interprétation des données notamment).
Pour la partie bio-informatique, Nicholas Jallan, consultant externe avec une spécialisation en machine learning, s'est rendu disponible pour donner des pistes et valider au fur et à mesure l'avancée du projet.

Le stage a pu être effectué partiellement en télétravail. Le temps passé au laboratoire a été surtout dédié aux tâches nécessitant un accès aux données.

2.4 Environnement de travail

Les maladies métaboliques sont détectées par lecture de données de GCMS, obtenues par analyse de l'urine des patients. Je vais présenter la machine de GCMS réalisant ces analyses et les deux ordinateurs associés.

La machine effectuant les analyses d'urine par GCMS (voir figure 2.2) permet d'obtenir un chro-

matogramme ainsi que les spectres de masse à chaque instant d'échantillonnage. Ces résultats d'analyse sont des éléments qui permettent le diagnostic de maladies par un biologiste.

FIGURE 2.2 – Photo de la machine de Bruker [1] effectuant les mesures de GCMS

La machine, fabriquée par l'entreprise Bruker fait passer l'échantillon par un chromatographe gazeux (figure 2.3), puis par un spectromètre de masse (figure 2.4). Le chromatographe consiste en une colonne de chromatographie. À l'entrée est placé l'échantillon sur un élément chauffant. Au fur et à mesure du temps, la température va augmenter et les molécules vont se détacher pour entrer dans la colonne dans laquelle se déplace un gaz. Toutes les molécules identiques se retrouveront en même temps au bout de la colonne du chromatogramme. Le temps d'arrivée (appelé temps de rétention), permet de quantifier chaque molécule de l'échantillon. Ensuite les molécules entrent dans le spectre de masse pour être identifiées. Cela consiste à casser la molécule en différents éléments. Ces éléments sont ensuite passés dans un analyseur, qui les trie par ratio de masse sur charge (m/z). Les différents m/z des fragments mesurés sont uniques pour chaque molécule. La machine permet donc de quantifier et identifier chaque molécule organique présente dans l'urine.

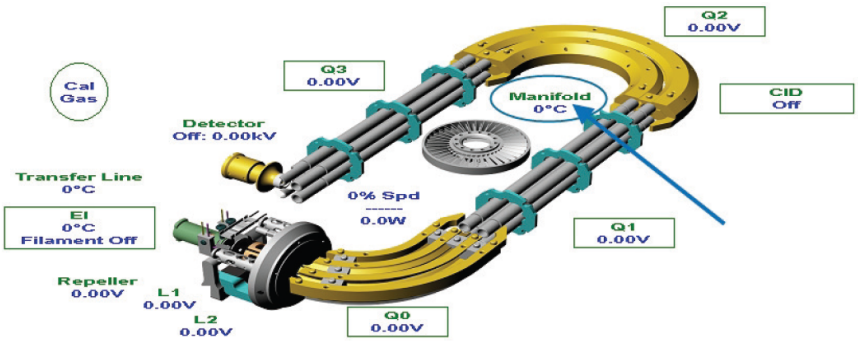

FIGURE 2.3 – Schéma de fonctionnement du chromatogramme de la machine de GCMS de Bruker [1]

Un premier ordinateur, connecté à la machine, permet aux techniciens de lancer les analyses, de récupérer les courbes ainsi que de faire la détection des molécules sur le chromatogramme. Le fichier contenant les données de l'analyse sera ensuite stocké dans un répertoire spécifique.
Un second ordinateur, appelé ordinateur de retraitement, a accès à ce répertoire. Il sert à l'analyse du chromatogramme par un expert. Une fois le diagnostic posé, celui-ci sera ajouté à la base de données du CHRU.
La visualisation de ces données se fait depuis le logiciel *MD Data Review* fourni également par Bruker.

Le stage se déroulera en ayant accès aux données via l'ordinateur de retraitement sur lequel est installé le logiciel *MS Data Review*, voir figure 2.5.

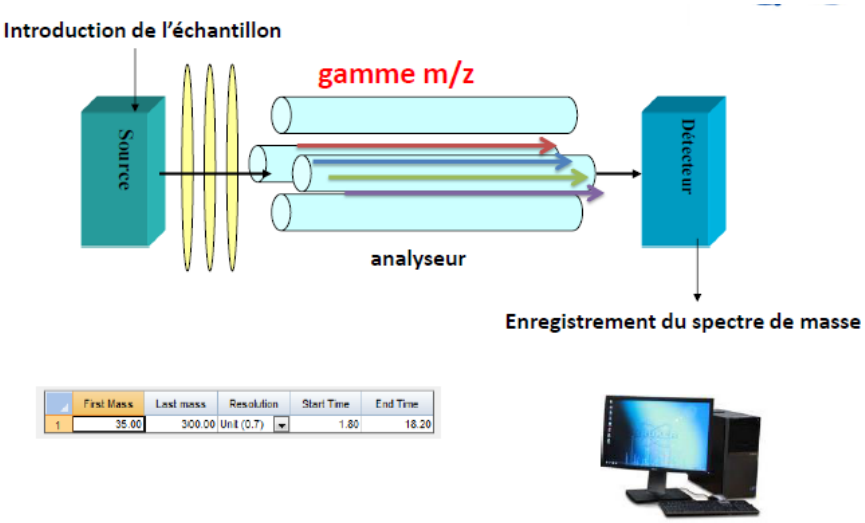

FIGURE 2.4 – Schéma de fonctionnement du spectre de masse de la machine de GCMS de Bruker [1]

En cas de problème vis-à-vis de l'accès aux données, il est possible d'utiliser l'ordinateur branché à la machine de GCMS, mais il est dédié aux techniciens du laboratoire.

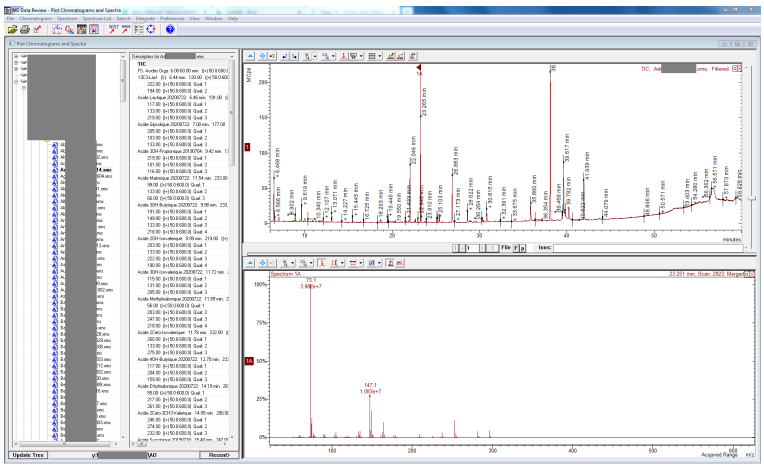

FIGURE 2.5 – Capture d'écran du logiciel MS Data Review de Bruker

L'accès aux données ne peut se faire que par le logiciel *MS Data Review*, car elles sont stockées dans un fichier de type *XMS* qui est un type propriétaire de l'entreprise Bruker. Il n'est donc pas possible d'utiliser de manière directe ce type de fichier.

La machine est sous Windows 7, et le réseau du CHRU est configuré de manière à empêcher l'accès à internet aux applications non internes au CHRU. La mise en place d'une application fonctionnant sur cet ordinateur pourra donc être compliquée.

3 Extraction des données

Dans le but de réaliser un programme d'IA, le point le plus important à considérer sont les données de départ.
Dans le cas présent, il est possible de constituer une base de données d'environ 1000 interprétations, ce qui en fait une base de données limitée, sachant que les maladies rares par définition ne sont pas des phénomènes courants et certaines maladies ne seront pas forcément présentes dans celle-ci.

Il faudra donc être particulièrement attentif dans ce projet à mettre au maximum à profit les données. Il sera notamment nécessaire de limiter au maximum le nombre de valeurs en entrée du programme d'IA, car généralement plus la quantité d'informations en entrée est grande, plus il faudra d'exemples pour que le programme soit efficace.

3.1 Stratégie

Dans un premier temps, il faudra savoir quelles sont les données qu'il est possible d'obtenir via les analyses de GCMS, et définir parmi celles-ci lesquelles seront utiles pour obtenir un programme fonctionnel. Cela pourra être fait en se renseignant auprès des techniciens et du Dr Marc Merten.

Dans un second temps, les données ne seront pas directement accessibles dans un format lisible avec python. L'objectif est de pouvoir les obtenir dans un format de type *CSV* qui est facilement lisible en utilisant Python. Afin de faciliter l'extraction de toutes les données il serait utile de trouver un moyen de les extraire automatiquement. Pour cela, il sera possible de contacter l'entreprise Bruker pour avoir plus d'information sur les possibilités d'extraction de données depuis le logiciel.

3.2 Réalisations

Lorsqu'un diagnostic est fait sur le résultat d'une analyse de GCMS, celui-ci est basé sur plusieurs éléments visibles directement sur le logiciel.

Premièrement le chromatogramme est l'élément le plus important, c'est une fonction qui donne la présence de molécules au cours du temps (voir figure 3.1).
Chaque molécule aura toujours le même temps d'apparition lors de la chromatographie gazeuse. Le chromatogramme permet donc de connaître la quantité de chaque molécule présente dans l'urine.

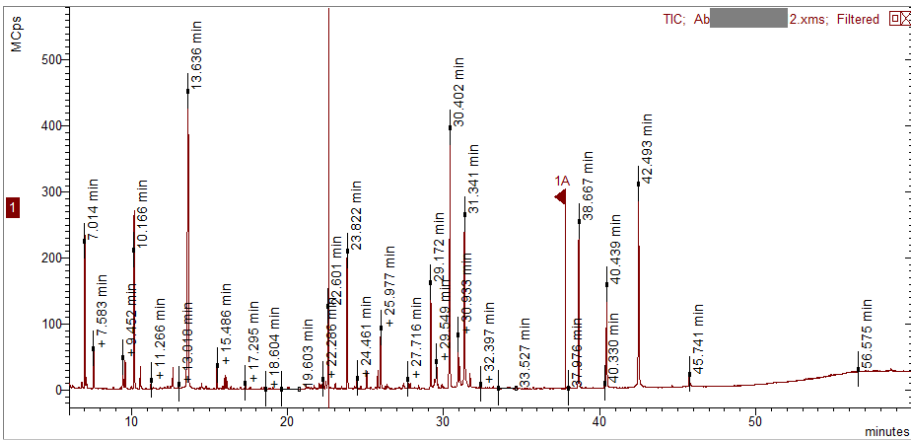

FIGURE 3.1 – Capture d'écran d'un chromatogramme

Deuxièmement, certaines molécules peuvent avoir un temps d'apparition commun. C'est là que les spectres de masse sont utiles. Ils donnent sous forme d'une courbe la composition de la molécule. Le spectre de masse de chaque molécule est unique, il permet donc d'identifier la molécule. De plus le logiciel de Bruker possède un outil d'identification automatique des molécules et affiche donc pour chaque molécule son nom et son temps d'apparition (voir figure 3.2).

Si l'on veut conserver le strict minimum en terme d'information, il semble donc logique de conserver uniquement le chromatogramme, quitte à avoir parfois une petite incertitude sur le nom d'une molécule.
Mais il y a un problème avec cette courbe, car les temps de rétention et l'intensité des pics sont parfois légèrement différents, en fonction des maintenances ou encore de l'encrassement de la machine. Or certaines molécules peuvent être considérées comme des points de référence, car elles apparaissent toujours sur les analyses.
C'est le cas de l'acide lactique (marqué au carbone 13), l'acide 4-phénylbutyrique, l'acide phenylacetique et l'acide heptadecanoique qui sont exogènes à l'urine, ajoutées par l'équipe et qui servent de référence temporelle. L'acide 4-phenylbutyrique est en plus une référence en intensité. Il est donc possible de corriger ces problèmes en connaissant les temps de rétention des quatre molécules de référence.

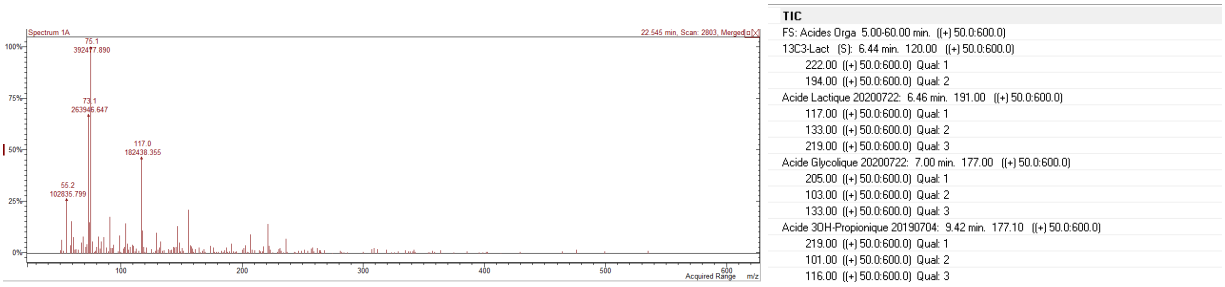

FIGURE 3.2 – À gauche un exemple de Spectre de masse, à droite les molécules détectées

Les données à extraire du logiciel sont donc le chromatogramme ainsi que le nom des molécules détectées.

Afin de savoir ce qu'il est possible d'extraire du logiciel, j'ai contacté différentes personnes de l'entreprise Bruker. Ce qui est à retenir est qu'il est possible d'extraire un chromatogramme et un fichier contenant les molécules détectées correspondantes. Mais il n'est pas possible d'automatiser l'extraction sur tous les chromatogrammes (fonctionnalité non prévue dans l'application), et les fichiers contenant les molécules peuvent être créées pour un dossier contenant plusieurs

analyses, ayant été faites avec la même méthode. Une méthode est un fichier qui est mis a jour à chaque calibration de l'appareil, et qui permet de détecter avec plus de précision les molécules, si une maintenance a introduit un décalage temporel par exemple. Or cette calibration est faite plusieurs fois par an. Il faudra donc extraire pour chaque analyse les deux fichiers manuellement depuis le logiciel.

3.3 Résultats

Il est donc possible d'obtenir pour chaque analyse, à partir du fichier *XMS* deux fichiers *CSV* permettant de connaître l'allure du chromatogramme au cours du temps, en ayant la valeur d'intensité sur environ 9000 points dans le premier fichier. Le temps de rétention et le nom des molécules apparaissent dans le second fichier.

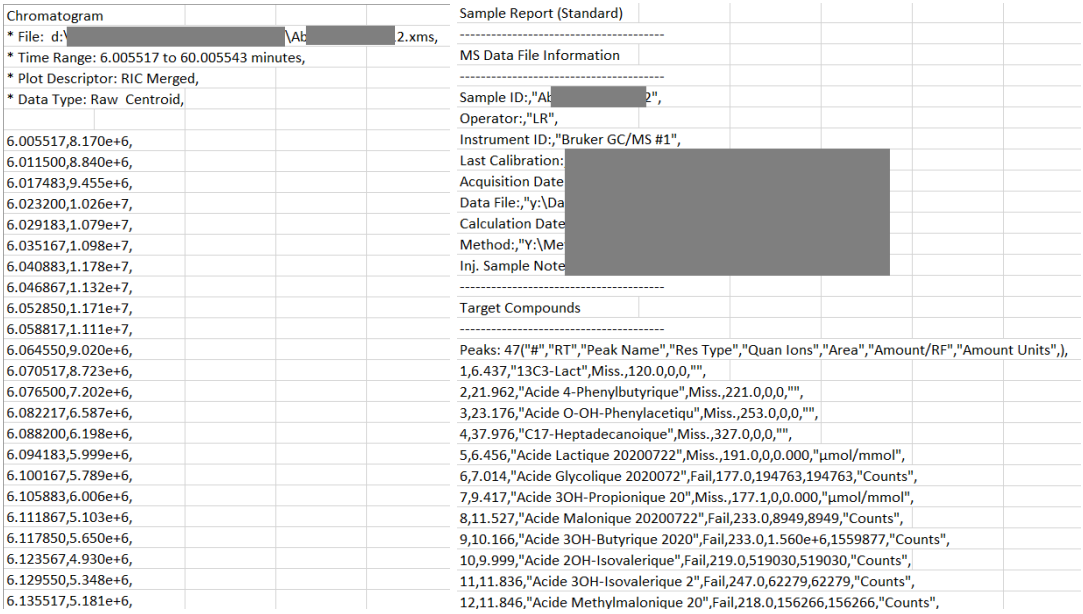

FIGURE 3.3 – Contenu du fichier *CSV* contenant le chromatogramme (nom de fichier finissant par *-chromatogram.csv)* sur la gauche et sur la droite les molécules détectées (nom de fichier finissant par *-ms.csv)*

4 Pré-traitements

Maintenant que les données intéressantes sont extraites, il faut trouver une méthode permettant de réduire au maximum le vecteur d'entrée du programme d'IA, ainsi que rendre le plus fiable possible les données conservées.

4.1 Stratégie

Dans un premier temps, il faut réussir à compenser les écarts en temps et en intensité qui peuvent être observés dans le chromatogramme en prenant en compte les temps de rétention lus dans le fichier contenant les molécules détectées.

Dans un second temps, la fréquence à laquelle la machine enregistre la valeur de l'intensité n'est pas constante, elle varie pour chaque point de la courbe. Il faut donc procéder à un ré-échantillonnage de la courbe. Plus la période utilisée pour ce ré-échantillonnage sera faible, moins il y aura de points sur la courbe, mais plus la délimitation des pics sera imprécise. Or il semble que certaines zones nécessitent plus de précision que d'autres, c'est-à-dire que deux pics proches peuvent être la conséquence de maladies très différentes et qu'il est donc nécessaire de pouvoir les différencier. Le ré-échantillonnage pourra donc être fait par zones où la fréquence d'échantillonnage dépendra de la précision nécessaire.

Ensuite, les pics les plus petits n'ayant pas forcément moins d'importance dans le diagnostic, l'intensité pourra être passée au logarithme afin de mettre en évidence tous les pics.
On constate également qu'un biais sur l'intensité varie au cours du temps. L'idée est de le quantifier avec un filtrage basse fréquence afin de le soustraire à la courbe.

4.2 Réalisations

Le programme s'exécutera dans l'ordre suivant :

- Ajustement des temps de rétention pour avoir une bonne correspondance des références, puis sélection de l'intervalle $[6, 45]$ minutes
- Passage en logarithme et ajustement de l'intensité
- Ré-échantillonnage par zones
- Calcul et soustraction du biais
- Élimination du bruit

Le programme reçoit en entrée le nom de fichier correspondant à l'analyse à traiter. Cela lui permet d'avoir accès à la fois au fichier *chromatogram* et au fichier *ms*. À partir de cela on peut

obtenir une première courbe par lecture du fichier *chromatogram*, voir figure 4.1

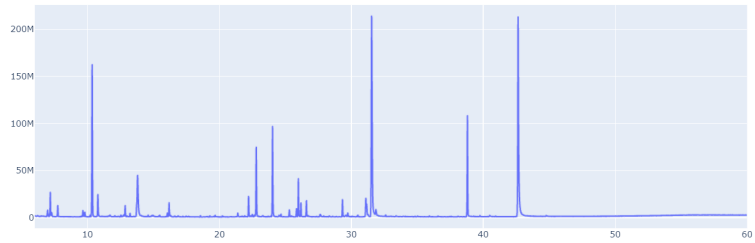

FIGURE 4.1 – Courbe brute obtenue par lecture d'un fichier *chromatogram.csv*

Ensuite par lecture du fichier *ms*, on a connaissance des temps de rétention des 4 molécules de référence qui sont : l'acide lactique, l'acide 4-phénylbutyrique, l'acide phenylacetique et l'acide heptadecanoique, ayant chacune un temps de référence, respectivement 7, 21.5, 23 et 38 minutes. Le but est de décaler toutes les valeurs de manière à ce que les temps de rétention des pics correspondent aux références.
Pour faire cela tous les points seront décalés linéairement entre deux pics de référence. Pour le calcul voir listing 4.1, il présente le décalage des points, contenus dans la liste t, compris entre deux pics de référence de temps de rétention *oldStart* pour le premier et *oldEnd* pour le second et dont les temps de référence sont *start* et *end*.
Les points n'étant pas compris dans l'intervalle $[6, 45]$ minutes ne sont eux pas conservés, car ces données ne sont pas intéressantes pour le diagnostic.

```
t -= (t[0] - start)
for i in range(len(t)):
    t[i] = start + (end-start) * (t[i] - start)/(oldEnd-oldStart)
```

Listing 4.1 – Calcul de l'abscisse de chaque point

Ensuite l'intensité de chacun des points de la courbe est passée en logarithme puis normalisée par rapport au second pic de référence (supposé constant pour toutes les analyses). Ceci, afin de faire apparaitre tous les pics importants, et de garantir que l'intensité lue restera comparable entre les différentes analyses.
Remarque : ce calcul (équation 4.1) est bijectif, ce qui garantit que l'on n'a pas de perte d'information au cours de cette étape.
La valeur de pic de référence sera choisi comme étant 10, ce qui est une valeur proche des résultats obtenus en faisant une normalisation simple.

$$intensite_p = (ln(intensite_p) - mean(chromatogram_{ln})/std(chromatogram_{ln}) * 10/intensite_{reference} \tag{4.1}$$

Dans cette équation, "p" est le point courant, "reference" est le second pic de référence, l'indice "ln" correspond au passage en logarithme du chromatogramme, "mean" est la moyenne et "std" l'écart type.

Maintenant dans le but de ne plus avoir un échantillonnage dépendant des chromatogrammes et de réduire le nombre de points, il y a une étape de ré-échantillonnage.
Après observation des chromatogrammes, et échange avec Dr Marc Merten, nous en sommes arrivés à la conclusion qu'il n'y a pas la même densité de pics dans tout le chromatogramme, et donc que le besoin en précision n'est pas le même partout.
La solution adoptée est donc de faire ce ré-échantillonnage par zones. Il y a quatre types de

zones retenues. Leurs périodes d'échantillonnage sont de 1, 2, 6 ou 20 secondes. Ces temps ont été déterminés en faisant de nombreux essais et en vérifiant si les pics importants sont bien toujours visibles après traitement. La figure 4.2 présente un exemple de chromatogramme après ré-échantillonnage. On voit bien sur cet exemple que la zone proche de 25 minutes est très dense contrairement à celle entre 32 et 40 minutes.

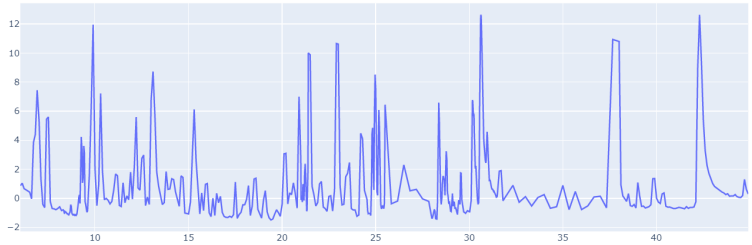

FIGURE 4.2 – Exemple de chromatogramme après ré-échantillonnage

Ensuite, les chromatogrammes ont tendance à présenter un biais au cours du temps, dû à la machine de GCMS. Il faut donc corriger ce biais afin d'éviter les variations d'intensité dû à celui-ci. Pour extraire ce biais de la courbe, l'idée est d'utiliser un filtrage passe bas, permettant d'obtenir l'allure du chromatogramme en basse fréquence. Ce biais est ensuite soustrait au chromatogramme afin de l'éliminer. La fréquence de coupure utilisée est de 0.002 Hz, déterminée en faisant des essais.
Ce biais est visible figure 4.3.

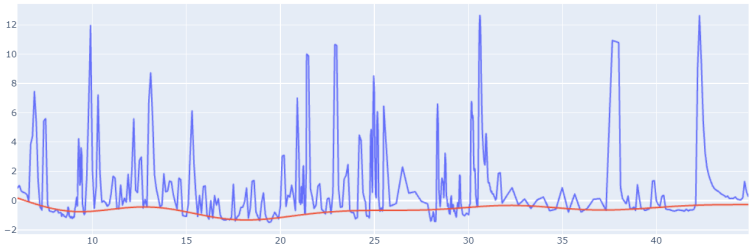

FIGURE 4.3 – Exemple de détection du biais sur un chromatogramme

Enfin, dans le but de retirer ce qui peut s'apparenter à du bruit sur le chromatogramme (amplitude faible et très haute fréquence), deux méthodes ont été mises en place. La première est d'utiliser un filtrage passe bas et la seconde est de faire un seuillage de nos valeurs d'intensité.
La première méthode a un inconvénient très problématique pour la réalisation de notre objectif, c'est que les pics étant des éléments haute fréquence, leur taille peut-être affectée par ce traitement, ce qui n'est pas souhaitable. C'est pourquoi la seconde méthode a été retenue.
Les valeurs étant en dessous du seuil de 1.6, seront mise à 0. Seuls les pics dépassant cette valeur seront conservés.

4.3 Résultats

Avec ces traitements, il est possible, à partir d'un chromatogramme d'environ 9000 points, de condenser l'information qu'il contient en 505 points, et de corriger de manière efficace les différents biais observés. Cela fait une réduction de l'ordre de 1700 % du nombre de points à gérer dans la partie intelligence artificielle, ce qui est une très bonne chose compte tenu du peu de données à disposition.

Le résultat final est composé d'un nombre limité de pics, permettant d'identifier plus facilement les molécules importantes pour l'établissement du diagnostic, voir figure 4.4.

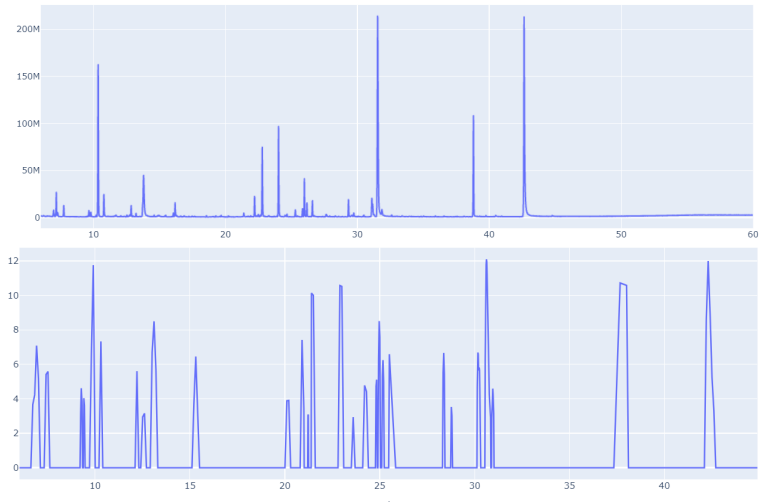

FIGURE 4.4 – Comparaison du chromatogramme avant le traitement (au-dessus) et après le traitement (en dessous). Les abscisses ne sont pas identiques sur les deux graphiques

5 Algorithme d'apprentissage automatique

Dans cette partie, l'implémentation de tous les algorithmes utilisés provient de l'API Scikit-learn [2] et Xgboost [6]. Utiliser des classes et fonctions existantes permet de gagner du temps sur la mise en place du code, et de consacrer plus de temps au choix du meilleur algorithme.

5.1 Mise en place de la base de données

Le nombre de données qu'il est possible d'extraire est de 1367. C'est sur ces données que l'apprentissage de notre programme d'IA va se baser. Il faut donc relier ces chromatogrammes aux diagnostics posés par le Dr Marc Merten. La base de données fait correspondre le nom d'un chromatogramme à un label (catégorie de diagnostic dans laquelle se place l'analyse). Les catégories définies sont les suivantes :

- Normal
- Cétose
- Maladie métabolique connue
- Bactéries
- Médicaments
- Autre

Dans le but d'être sûr que la base de données soit fiable, j'ai vérifié avec Dr Marc Merten que pour chaque chromatogramme, celui-ci soit correctement traité, et que la catégorie affectée soit bien visible sur celui-ci.

5.2 Stratégie

5.2.1 Présentation de l'apprentissage automatique

Dans le fonctionnement d'un programme d'IA, il y a deux phases : une première phase dite d'entrainement et une seconde dite de prédiction.
Dans la phase d'entrainement, on met en entrée du programme des données connues, ainsi que leur label associé. Dans notre cas, ce sera le chromatogramme après traitement avec sa catégorie. Le programme va donc adapter de manière automatique ses paramètres internes pour donner au maximum le bon label à chaque chromatogramme de la base de données.
Dans la phase de prédiction, les paramètres ne seront plus modifiés et le programme pourra être utilisé pour faire des prédictions sur de nouvelles données. Ici on aura un nouveau chromatogramme en entrée, la sortie étant le label prédit. Cette phase correspondra à l'aide au diagnostic pour le biologiste.

À l'issue de la phase d'entrainement, pour savoir si un programme d'IA a correctement appris à reconnaitre les chromatogrammes, il n'est pas forcément bon de vérifier si les données d'entrainement sont correctement prédites. En effet, notre de but est de savoir si le programme est efficace sur des données avec lesquelles il n'a jamais été en contact. La solution est alors de mettre de côté une partie de la base de données pour mesurer l'efficacité du programme après entrainement. La base de données est donc découpée en deux parties non égales. Une partie pour la phase d'entrainement et une seconde pour les tests. En général une grande majorité est attribuée à l'entrainement. De cette manière on peut détecter les cas de *surapprentissage*, c'est-à-dire que le programme apprenne "trop" la base de données et soit donc incapable de généraliser.

5.2.2 Choix de l'algorithme utilisé

Dans cette section, je serai amené à tester plusieurs méthodes, afin de pouvoir en comparer les résultats. Il est important de mettre en place tous les éléments permettant de rendre les résultats reproductibles.
La base de données d'entrainement et de test sont donc deux bases de données fixes (et non construites aléatoirement comme c'est souvent le cas).

Puisque que la base de données n'est pas très grande (1367 données), je commence par un programme séparant les chromatogrammes normaux des non-normaux. En effet, cela est moins complexe que de séparer selon les 6 catégories définies dans la partie 5.1. Si cela fonctionne bien, alors j'envisagerai de passer à la prédiction de la catégorie, qui a la particularité de pouvoir posséder plusieurs labels (par exemple bactéries et médicaments vont souvent de pair) ce qui complexifie le problème.

Pour le choix des algorithmes à tester, je me suis référé à la page dédiée sur le site web de Scikit-learn : scikit-learn. En plus d'un choix d'algorithme d'IA, on pourra si besoin utiliser un algorithme de réduction de dimension, dans le cas où les résultats sur les 505 points après traitement ne sont pas suffisants. Ces algorithmes permettront de réduire ce nombre de points et donc peut-être de faciliter l'apprentissage.

5.2.3 Mesure de la performance

Pour savoir si l'algorithme d'IA est performant, il faut définir les métriques qui nous intéressent et pouvoir les comparer aux résultats d'un autre biologiste sur ces analyses.

Pour savoir sur quelles métriques se baser, il faut réfléchir aux contraintes que l'on veut imposer au résultat de notre programme. Il existe une multitude de métriques, la plus courante étant la mesure de la justesse, $justesse = \frac{nombre\ correctement\ predis}{nombre\ total}$. Or ici en plus de chercher à obtenir une bonne justesse, il est très important de ne pas passer à côté d'échantillons non-normaux. Il faut donc réduire le plus possible le nombre de non-normaux prédits normaux.

5.3 Réalisations

Pour avoir une idée du résultat qu'il est possible d'obtenir par une personne, j'ai demandé à une collègue de mon maître de stage Dre Elise Jeannesson de me préparer un diagnostic de quelques

chromatogrammes en les triant dans les différentes catégories. Cet exercice étant chronophage, il a été possible de n'avoir un diagnostic que pour 16 analyses (13 normaux et 3 non-normaux). Cela ne sera donc pas forcément statistiquement significatif, mais donne déjà un premier ordre d'idée.
Les catégories données ne sont pas toujours bonnes, mais la détection des chromatogrammes normaux se fait presque sans faute avec seulement une erreur sur 16. Les résultats pour la classification normal/non-normal sont présentés dans la figure 5.1

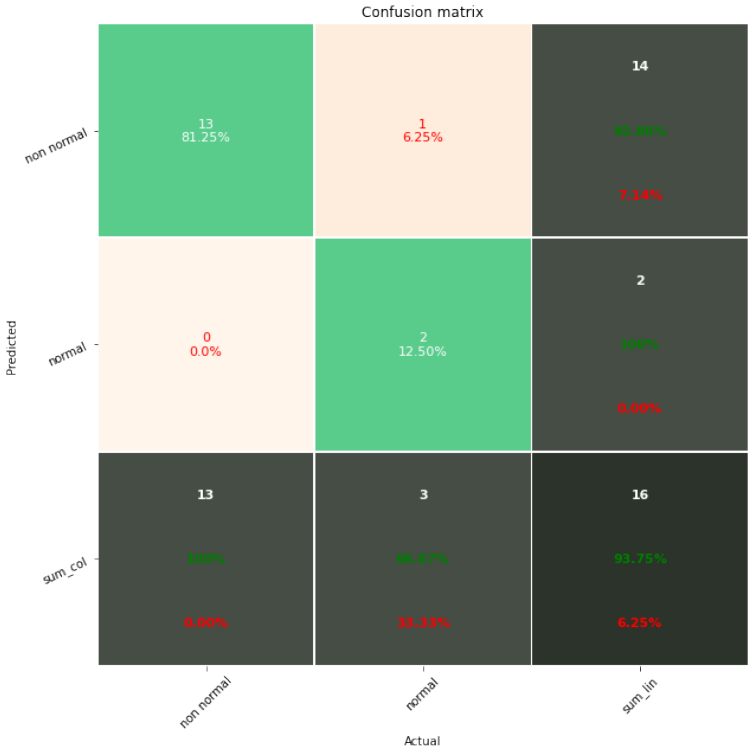

FIGURE 5.1 – Matrice de confusion obtenue par Dre Elise Jeannesson

Pour les métriques utilisées, j'ai utilisé principalement le rappel (équation : 5.2) et la précision (équation : 5.1).

$$precision = \frac{nb\ echantillon\ correctement\ attribues\ a\ la\ classe}{nb\ echantillon\ attribues\ a\ la\ classe} \tag{5.1}$$

$$rappel = \frac{nb\ echantillon\ attribus\ a\ la\ classe}{nb\ echantillon\ de\ la\ classe} \tag{5.2}$$

La figure 5.1 présente la matrice de confusion obtenue. C'est un outil permettant de voir la répartition des classes prédites par rapport à la classe réelle.
La précision se lit sur les lignes et le rappel sur les colonnes. Par exemple ici le rappel sur les normaux est de $2/2 \longrightarrow 100\%$. Le rappel sur les non-normaux est de $13/13 \longrightarrow 100\%$. Ces mesures de rappel et de précision sont les mesures idéales que l'on recherche avec notre programme d'IA, car aucun non-normal n'est prédit normal.

Pour la mise en place du programme d'IA avec Scikit-learn, la structure du code est toujours la même peu importe le type d'algorithme utilisé et les paramètres choisis. La structure est présentée ci-dessous.

```
model = Algorithm(parameters)
model.fit(X_train, y_train)
```

```
model.score(X_test, y_test)
```

Listing 5.1 – Entrainement type d'un algorithme d'IA avec Scikit-learn [2]

Afin de choisir les meilleurs paramètres pour un algorithme donné, il est possible d'utiliser l'algorithme de grid-search, qui permet de donner une liste de paramètres à parcourir et qui indiquera la meilleure combinaison de ces paramètres pour avoir une justesse maximale.

Pour avoir un ordre d'idée des résultats qu'il est possible d'obtenir, j'ai essayé l'algorithme de grid-search sur un KNeighborsClassifier de Scikit-learn [2] avec les paramètres suivants :

```
grid_params = {
    'n_neighbors': np.arange(1,20),
    'metric': ['euclidean', 'manhattan', 'chebyshev', 'minkowski'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}
```

Listing 5.2 – Paramètres parcourus avec grid-search

La meilleure combinaison trouvée est 'algorithm' : 'ball_tree', 'metric' : 'chebyshev' et 'n_neighbors' : 6.
La matrice de confusion résultante est figure 5.2. On observe tout de suite qu'il y a 16% des échantillons de tests qui ont été prédits normaux au lieu de non-normaux, ce qui est problématique, car on cherche justement à réduire le plus possible cette valeur. Pour la justesse, elle est ici d'environ 80% ce qui est peu pour le moment, mais est encourageant pour la suite, car il sera possible de faire mieux en essayant d'autres types d'algorithmes.

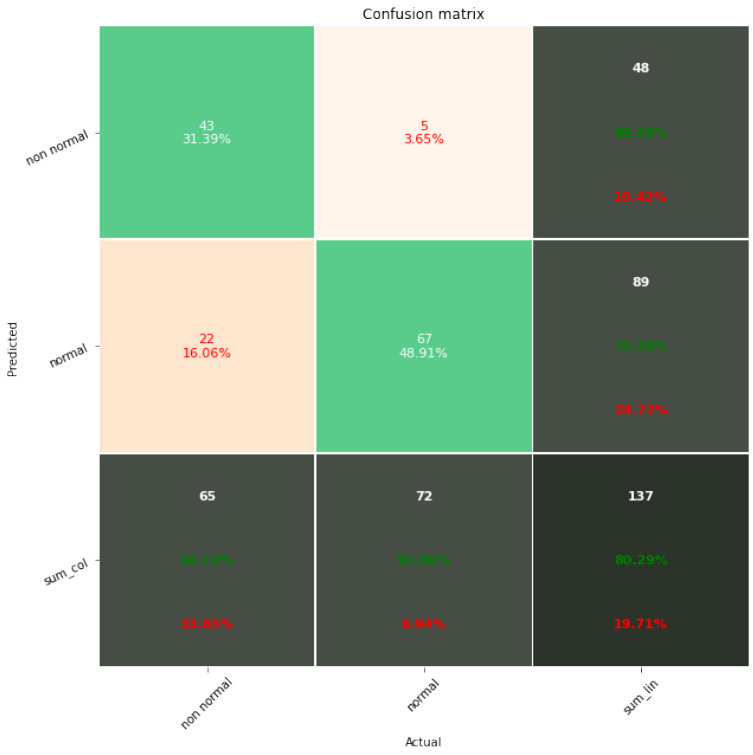

FIGURE 5.2 – Matrice de confusion avec les paramètres aléatoires

L'idée pour obtenir de meilleurs résultats, est d'utiliser des algorithmes de réduction de dimension. Les difficultés d'apprentissage venant sans doute du fait que le vecteur d'entrée de 505 points est trop grand par rapport au nombre d'exemples dans la base de données d'entrainement. Di-

verses méthodes existent et sont utilisables depuis l'api de Scikit-learn. Leur utilisation se fait comme ci-dessous :

```
reduction = Algorithm(n_component=size)
X_train = reduction.fit_transform(X_train, y_train) # entrainement
X = reduction.transform(X) # reduction d'un vecteur
```

Listing 5.3 – Utilisation type de la réduction de dimension

Enfin comme le temps restant pour mon stage commençait à manquer, plutôt que de s'intéresser individuellement à chaque méthode de réduction de dimension et chaque algorithme d'IA, j'ai écrit un programme parcourant toutes les possibilités.

```
dimention_reduction = [None, PCA, TruncatedSVD, LocallyLinearEmbedding,
    SparseRandomProjection]
IA_techniques = [RandomForestClassifier(), SVC(), AdaBoostClassifier(),
    XGBClassifier(use_label_encoder=False, eval_metric='logloss')]
grid_params =[
    {
        'n_estimators': np.arange(10, 100, 10),
        'criterion': ['gini', 'entropy']
    },
    {
        'kernel' : ['linear', 'poly', 'rbf', 'sigmoid'],
        'degree' : np.arange(2,5)
    },
    {
        'n_estimators' : np.arange(10, 100, 10)
    },
    {
        'booster' : ['gbtree', 'gblinear', 'dart'],
        'eta' : np.arange(0.1,0.9,0.2)
    }
]
data_size = [50, 250, 450]
```

Listing 5.4 – Ensemble des algorithmes et paramètres parcourus

dimention_reduction donne tous les algorithmes de réduction de dimension parcourus (*None* correspondant à pas de réduction), les tailles de vecteurs après réduction étant dans *data_size*. Enfin les algorithmes d'IA sont choisi parmi *IA_techniques* avec un grid-search pour déterminer le choix des paramètres présent des *grid_params*.

Le parcours de toutes ces méthodes peut prendre du temps. Dans mon cas le temps de calcul a été d'environ 1 h 30.

5.4 Résultats

Après parcours de l'ensemble des méthodes décrites ci-dessus, il apparait que la meilleure combinaison en privilégiant la justesse est un algorithme de forêt aléatoire, dont le paramètre *n_estimators* est de 70 (70 arbres de décision composant la forêt aléatoire). Ce cas-ci n'a pas utilisé de réduction de dimension. La figure 5.3 présente la matrice de confusion résultante.

En observant la matrice de confusion, il apparait que cette méthode est bien meilleure en termes de justesse, de précision sur les normaux et de rappel sur les non-normaux que le premier test

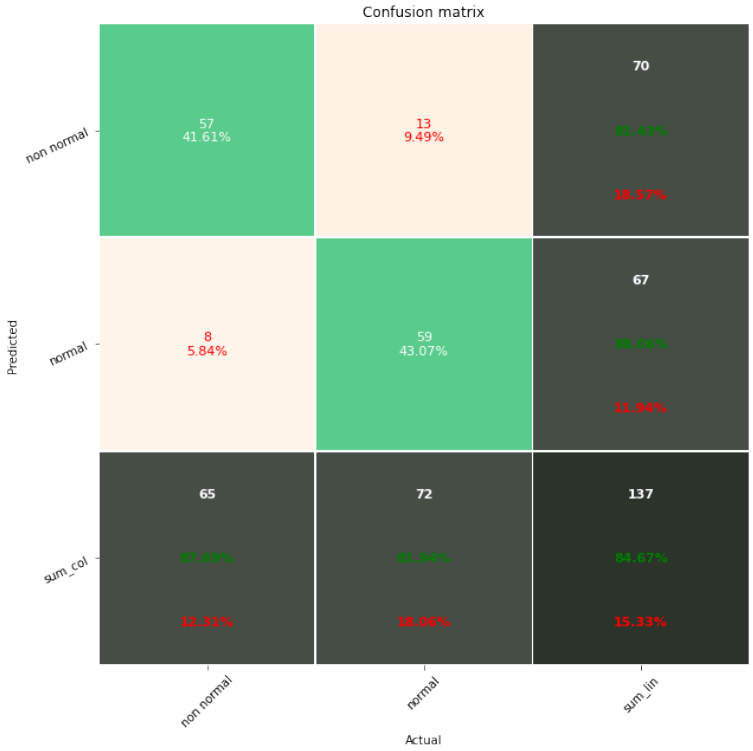

FIGURE 5.3 – Matrice de confusion de la meilleure combinaison trouvée

sur les KNeighborsClassifier. Cependant, cela reste moins bon que l'objectif qui est d'avoir un résultat équivalent au diagnostic réalisé par un humain.

Cela n'est donc pas pleinement satisfaisant. Il faudra continuer à chercher de nouvelles méthodes pour améliorer les résultats.

6 Création d'un logiciel

Pour permettre au personnel du laboratoire d'utiliser le programme d'intelligence artificielle sans connaissance particulière en informatique, il est important de mettre en place un logiciel s'intégrant correctement à l'environnement déjà présent.

6.1 Stratégie

Les besoins relatifs à ce logiciel sont les suivants :

- Obtenir le résultat du programme pour n'importe quel chromatogramme présent sur le réseau
- Permettre l'impression de ce résultat afin de conserver celui-ci dans les archives
- Continuer d'enrichir la base de données avec les analyses futures

L'environnement de travail au laboratoire ne permettant pas de faire une installation de python sur l'ordinateur, il faudra donc créer un fichier exécutable à partir du logiciel créé.

6.2 Réalisations

Dans un premier temps, pour créer un exécutable à partir du projet en python, j'ai utilisé le module *auto-py-to-exe* [5]. Cela permet de créer facilement un exécutable en changeant les paramètres de compilation depuis une interface type web. Le module se charge lui-même d'ajouter à l'exécutable toutes les dépendances utiles pour le fonctionnement du programme.

Dans un second temps, j'ai utilisé le module *PySimpleGUI* [3] pour la création d'une fenêtre simple. Ce module permet notamment d'intégrer un explorateur de fichier, afin d'indiquer le fichier du chromatogramme, des champs pour la saisie de valeurs, ainsi qu'une zone d'affichage des courbes.

Enfin pour la création du PDF, j'ai utilisé le module *FPDF* [4]. Cela me permet d'intégrer au PDF des images et du texte.

6.3 Résultats

La présentation de fenêtre est la suivante :

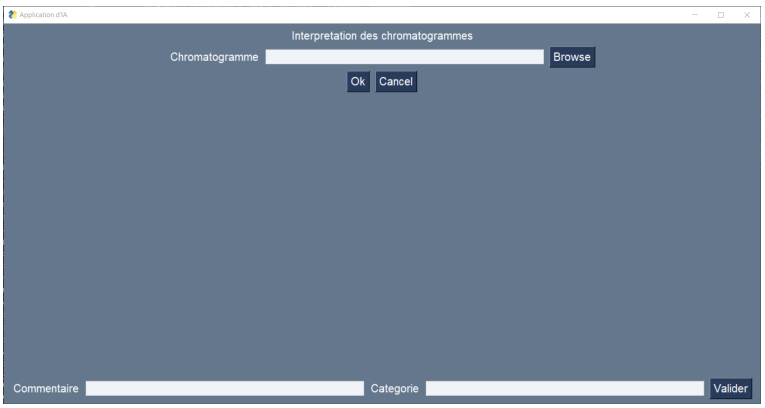

FIGURE 6.1 – Fenêtre du logiciel créé

Pour savoir comment utiliser le logiciel, la marche à suivre est en annexe A.

L'exécutable créé ne fonctionnant pas sur windows 7, il a fallu s'adapter. Le logiciel sera donc sur un autre ordinateur du laboratoire. Le fichier PDF sera ensuite enregistré dans un répertoire accessible depuis l'ordinateur de retraitement.

7 Bilan du stage

7.1 Résultats

L'objectif de ce stage était de mettre en place un programme d'aide à la décision pour la détection de maladies rares du métabolisme. Cet objectif initial était ambitieux compte tenu de la quantité de travail à réaliser, et du fait que rien n'était en place au niveau du laboratoire pour accélérer le processus de développement de ce programme.

Pour ce qui est de la création de la base de données, l'extraction et le traitement des données existantes sont entièrement réalisés. Le logiciel créé permet de continuer d'augmenter le nombre d'échantillons dans cette base de données, pour éviter de devoir recommencer un travail d'extraction des données dans le futur.

Pour ce qui est de la prise de décision en elle-même, nous avons vu que les résultats sont encourageants, mais ne permettent pas au biologiste de se reposer sur le diagnostic de l'IA, car la justesse pour le moment est trop faible. Mais la mise en place de détection de surplus d'une molécule en particulier peut s'avérer utile pour rapidement repérer certaines maladies métaboliques.

La structure pour permettre dans le futur une détection des maladies par catégories est déjà en place, mais n'a pas pu être finalisée par manque de temps. Ce type de prédiction sera plus utile que la classification normal/non-normal, car chaque catégorie est due à des marqueurs métaboliques différents.

Ce stage a fait office de preuve de concept, pour mettre en place une démarche d'aide au diagnostic des maladies métabolique. Le projet pourra être approfondi dans le futur pour obtenir des résultats plus fiables.

7.2 Difficultés rencontrées

Au cours de ce stage, j'ai été confronté à des difficultés. La majorité venait du fait que le CHRU met en place beaucoup de sécurité informatique. Il a donc fallu un peu de temps en début de stage afin d'obtenir un poste de travail pleinement fonctionnel, adapté à ma mission de stage.

Ensuite les ordinateurs étant configurés de manière à empêcher le fonctionnement de tous logiciels extérieurs au CHRU, il n'a pas été possible d'y installer python ni tout autre logiciel qui aurait pu me servir au cours du stage. Les ordinateurs étant sous Windows 7, il a été compliqué de trouver une solution pour créer un exécutable fonctionnant sur les machines du laboratoire.

La direction des systèmes informatiques (DSI) n'étant pas directement sur place, il était difficile

d'être en lien avec les interlocuteurs compétents pour la résolution de mes soucis informatiques.

Heureusement, j'avais à ma disposition quelques exemples de chromatogrammes sur lesquels travailler, en attendant d'avoir accès directement aux données depuis l'ordinateur de retraitement.

Enfin le nom des chromatogrammes est noté par un technicien lors de l'analyse, mais la convention de nommage des fichiers peut varier d'un technicien à l'autre, ce qui a rendu plus difficile la liaison entre base de données de diagnostics et les chromatogrammes. La solution a été d'ajouter manuellement tous les chromatogrammes non liés automatiquement à la base de données.

7.3 Ouverture

Ce projet pourra être approfondi en ajoutant dans un premier temps la détection des catégories. Ensuite, il sera possible de travailler sur les éléments classifiés dans la catégorie *autre*. Cette catégorie est très difficile à interpréter, car certaines maladies appartenant à celle-ci ne sont représentées que par quelques cas en France. Il est donc compliqué de trouver des informations sur leur caractérisation dans les chromatogrammes. Il est possible d'aider le biologiste à comprendre le type de maladie en lui présentant des chromatogrammes similaires, avec le diagnostic donné dans ce type de cas. Cependant, cela suppose une base de données suffisamment grande pour contenir tous les cas très rares.

Au cours de mon stage, j'ai pu me rendre compte que pour réaliser un programme d'aide au diagnostic, il est crucial d'avoir beaucoup de données de qualité. Je pense donc qu'il serait très utile d'avoir des bases de données partagées entre les hôpitaux afin de pouvoir plus facilement mettre en place des projets comme celui dont mon stage a fait l'objet.

Enfin la question de la distribution du logiciel créé a été soulevée, mais est pour l'instant restée en suspens. Faut-il le commercialiser en déposant un brevet, rendre le code open source, ou y a-t-il d'autres solutions ? Ces questions devront trouver une réponse quand le projet sera pleinement fonctionnel. L'objectif étant de le rendre accessible aux autres hôpitaux demandeurs.

8 Conclusion

Mon stage au CHRU de Nancy-Brabois m'a mis dans une situation où je devais agir en autonomie et faire preuve d'initiative, car personne au laboratoire n'était en mesure de m'aider sur la partie informatique de mon stage. Je communiquais uniquement à distance avec Nicholas Jallan pour de lui faire part de l'avancée de mon travail et afin qu'il me conseille pour continuer. Cet échange a été très enrichissant pour moi. Cela m'a permis d'avoir un retour de quelqu'un d'expérimenté et donc d'apprendre et de m'améliorer grandement dans le domaine de l'apprentissage automatique.

J'ai dû également m'intégrer au sein de l'équipe du laboratoire, dans un domaine très différent de celui de l'informatique qui est le domaine médical. Au début, il était un peu compliqué de faire comprendre ma mission auprès de l'équipe du laboratoire, mais au fur et à mesure des interactions, j'ai pu susciter sa curiosité en donnant une envie de collaborer. J'ai pu me rendre compte de l'importance que revêt la partie managériale dans un travail d'équipe.

J'ai pu voir que mon sujet de stage intriguait les personnes travaillant au laboratoire, y compris en dehors du secteur où j'ai effectué mon stage. Cela a permis d'aborder des sujets très intéressants notamment sur l'arrivée de l'IA dans le domaine médical. J'ai pris conscience du potentiel d'utilisation de l'IA dans un secteur comme celui où j'ai effectué mon stage. Ce potentiel n'est pas pleinement utilisé pour le moment, mais il est possible que dans un futur proche des biologistes, voire des chefs de service et professeurs soient demandeurs de la mise en place de ce type d'algorithmes.

À la fin de mon stage, j'ai présenté l'avancée du projet sous forme d'un diaporama à une quinzaine de personnes (biologistes, techniciens, internes, ...). J'ai été sollicité pour une seconde présentation en septembre 2021 pour les personnes qui n'ont pas pu être présentes.

J'ai apprécié réaliser ce stage dans un domaine inconnu pour moi. Le sujet était très ambitieux surtout sur un temps aussi court. C'était un challenge étant donné mon peu de connaissances initiales en bio-informatique. Ce challenge me semble réussi en partie, car j'ai pu mettre en place les bases du projet, même si j'aurais aimé avoir le temps et les moyens de le finaliser.
Travailler sur un sujet comme celui-ci donne un sens à mon travail, car l'objectif est de contribuer à une meilleure prise en charge d'enfants à l'hôpital.

Bibliographie / Webographie

[1] Bruker. *Manuel d'utilisation.* 4, 5, 29

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software : experiences from the scikit-learn project. In *ECML PKDD Workshop : Languages for Data Mining and Machine Learning*, pages 108–122, 2013. 13, 16, 31

[3] PySimpleGUI Inc. Pysimplegui documentation. pysimplegui.readthedocs.io. 19

[4] Mariano Reingart. Fpdf documentation. pyfpdf.readthedocs.io. 19

[5] Brent Vollebregt. auto-py-to-exe documentation. pypi.org/project/auto-py-to-exe/. 19

[6] xgboost developers. Xgboost documentation. xgboost.readthedocs.io. 13

Glossaire

Bruker Entreprise fabriquant du matériel de recherche et de laboratoire en biologie www.bruker.com 4–7, 29

GCMS Acronyme anglais pour chromatographie gazeuse couplée à la spectrophotométrie de masse, permet la détection et l'identification des composés organiques dans un échantillon (ici l'urine) 2–6, 11, 27, 29

IA Ensemble de techniques visant à apprendre un modèle de décision. Ici on se focalisera sur l'apprentissage automatique, qui consiste à résoudre un problème à partir de données, par une approche mathématique 2, 3, 6, 9, 13–17, 21, 23, 31, 34

PubMed Moteur de recherche qui référence des articles scientifiques dans le domaine de la biologie ainsi que biomédical, développé par le U.S. National Institutes of Health's. https://pubmed.ncbi.nlm.nih.gov/ 2

temps de rétention Temps mesuré d'arrivée pour une molécule au bout de la colonne de chromatographie de la machine de GCMS 4, 7–10

Liste des illustrations

Listings

Annexes

A Utilisation du logiciel

Au démarrage du fichier exécutable noté *IA-GCMS*, la fenêtre suivante apparait :

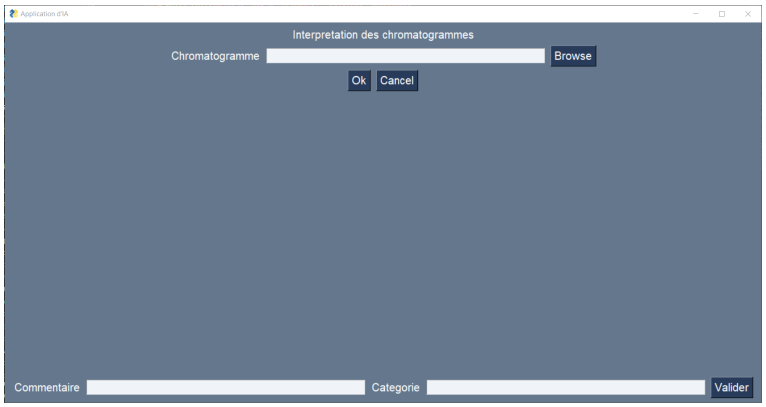

FIGURE A.1 – Fenêtre de démarrage

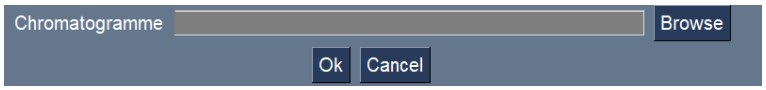

FIGURE A.2 – Zoom sur le champ recherche de chromatogramme

La première chose à faire est d'indiquer un chromatogramme via le bouton **Browse**. Celui-ci ouvre un explorateur de fichier. Il sera uniquement possible de sélectionner des fichiers *-chromatogram.csv*. Il faudra faire attention à ce que le fichier *-ms.csv* se trouve bien dans le même dossier et avec le même nom que le chromatogramme. Une fois le fichier ajouté, le chemin apparait dans le champ **Chromatogramme**.
Le bouton **Ok** sert ensuite à faire le traitement du chromatogramme, détecter les éventuels pics problématiques et créer le PDF résumant les résultats obtenus.
Le bouton **Cancel** sert lui à quitter l'application.

Une fois le traitement fini, le chromatogramme va s'afficher avant et après traitement (voir figure A.3), dans le but de vérifier que cela correspond bien au chromatogramme observé depuis l'ordinateur de retraitement.
Dans le même temps, le PDF créé pendant la phase de traitement va s'ouvrir automatiquement (voir figure A.4). Ce PDF peut être directement retrouvé depuis le dossier où est enregistré le chromatogramme à diagnostiquer. Il contient le chromatogramme avant et après traitement, ainsi que les résultats de l'analyse par le programme d'IA.

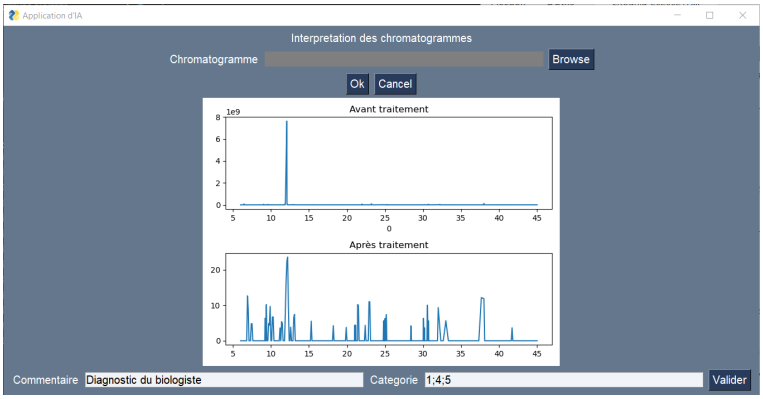

FIGURE A.3 – Fenêtre avec affichage du chromatogramme

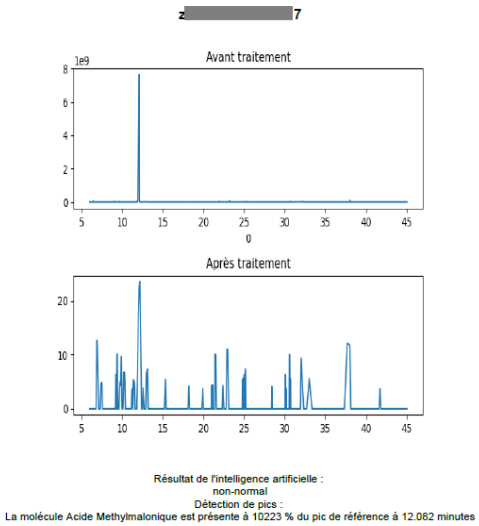

FIGURE A.4 – PDF créé et enregistré automatiquement

Enfin après le diagnostic par un biologiste du chromatogramme, il est possible d'ajouter directement le diagnostic à la base de données. Si la fenêtre a été fermée entre le moment du traitement et celui de la mise dans la base de données, il suffit de refaire les étapes précédentes.
L'ajout à la base de données se fait en bas de la fenêtre, juste en dessous du chromatogramme (voir figure A.5).
Il est ensuite possible de noter le diagnostic dans le champ **Commentaire**, ainsi que les catégories (nombres correspondant aux catégories, séparés par des virgules s'il y en a plusieurs), puis de cliquer sur le bouton **Valider**.

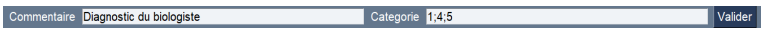

FIGURE A.5 – Zone d'ajout à la base de données

Résumé

Ce document présente mon stage de deuxième année d'école d'ingénieur, Télécom Nancy, au laboratoire de biologie médicale et biopathologie du CHRU de Nancy-Brabois. Il s'inscrit dans la détection de maladies rares du métabolisme à partir des analyses d'urine des patients par chromatographie gazeuse couplée à la spectrométrie de masse (GCMS).

L'objectif de ce stage est de mettre en place une démarche d'aide au diagnostic des analyses d'urine, afin d'aider le biologiste à gagner du temps sur leur interprétation qui est complexe.

Pendant ce stage, j'ai développé une application python permettant de donner un premier diagnostic obtenu avec de l'intelligence artificielle, ainsi que de faciliter la mise à jour de la base de données.

Mots-clés : GCMS, maladies métaboliques, apprentissage automatique

Abstract

This document presents my second year of engineering school's "Télécom Nancy" internship at the "biologie médicale et biopathologie" laboratory at the CHRU Nancy-Brabois. It is about the detection of inborn errors of metabolism by the analysis of gas chromatography - mass spectrum from patient's urine.

The purpose of this internship is to build a method to help the diagnosis of urine results. This is done to help the biologist save time on these complex analyzes's interpretations.

During the internship, I developed a software that can give a first diagnosis with machine learning techniques and help to maintain the database up to date.

Keywords : GCMS, inborn error of metabolism, machine learning