Université de Lille Faculté de Pharmacie de Lille

Année Universitaire 2022/2023

THESE
POUR LE DIPLOME D'ETAT
DE DOCTEUR EN PHARMACIE

Soutenue publiquement le 8 décembre 2022
Par M Pawlak Geoffrey

Intelligence artificielle et machine learning dans les stratégies de drug discovery

---

Membres du jury :

Président et Directeur de Thèse:
Monsieur Philippe Chavatte, Professeur des
Universités et Assesseur en charge des Relations Internationales à la Faculté de Pharmacie de Lille

Assesseurs :
Madame Delphine Allorge, Professeur des Universités, Praticien Hospitalier et Doyen de la faculté de pharmacie de Lille.
Monsieur Thomas Morgenroth, Maître de Conférences, Faculté de pharmacie de Lille.

Membre extérieur :
Guilbert Grégory, Président directeur général de Pharmacodietetics

Faculté de Pharmacie de Lille
3 Rue du Professeur Laguesse – 59000 Lille
03 20 96 40 40
https://pharmacie.univ-lille.fr

Université de Lille

Président	Régis BORDET
Premier Vice-président	Etienne PEYRAT
Vice-présidente Formation	Christel BEAUCOURT
Vice-président Recherche	Olivier COLOT
Vice-présidente Réseaux internationaux et européens	Kathleen O'CONNOR
Vice-président Ressources humaines	Jérôme FONCEL
Directrice Générale des Services	Marie-Dominique SAVINA

UFR3S

Doyen	Dominique LACROIX
Premier Vice-Doyen	Guillaume PENEL
Vice-Doyen Recherche	Éric BOULANGER
Vice-Doyen Finances et Patrimoine	Damien CUNY
Vice-Doyen Coordination pluriprofessionnelle et Formations sanitaires	Sébastien D'HARANCY
Vice-Doyen RH, SI et Qualité	Hervé HUBERT
Vice-Doyenne Formation tout au long de la vie	Caroline LANIER
Vice-Doyen Territoires-Partenariats	Thomas MORGENROTH
Vice-Doyenne Vie de Campus	Claire PINÇON
Vice-Doyen International et Communication	Vincent SOBANSKI
Vice-Doyen étudiant	Dorian QUINZAIN

Faculté de Pharmacie

Doyen	Delphine ALLORGE
Premier Assesseur et Assesseur en charge des études	Benjamin BERTIN
Assesseur aux Ressources et Personnels	Stéphanie DELBAERE
Assesseur à la Santé et à l'Accompagnement	Anne GARAT
Assesseur à la Vie de la Faculté	Emmanuelle LIPKA
Responsable des Services	Cyrille PORTA
Représentant étudiant	Honoré GUISE

Professeurs des Universités - Praticiens Hospitaliers (PU-PH)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	ALLORGE	Delphine	Toxicologie et Santé publique	81
M.	BROUSSEAU	Thierry	Biochimie	82
M.	DÉCAUDIN	Bertrand	Biopharmacie, Pharmacie galénique et hospitalière	81
M.	DINE	Thierry	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
Mme	DUPONT-PRADO	Annabelle	Hématologie	82
Mme	GOFFARD	Anne	Bactériologie - Virologie	82
M.	GRESSIER	Bernard	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
M.	ODOU	Pascal	Biopharmacie, Pharmacie galénique et hospitalière	80
Mme	POULAIN	Stéphanie	Hématologie	82
M.	SIMON	Nicolas	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
M.	STAELS	Bart	Biologie cellulaire	82

Professeurs des Universités (PU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	ALIOUAT	El Moukhtar	Parasitologie - Biologie animale	87
Mme	AZAROUAL	Nathalie	Biophysique - RMN	85
M.	BLANCHEMAIN	Nicolas	Pharmacotechnie industrielle	85
M.	CARNOY	Christophe	Immunologie	87
M.	CAZIN	Jean-Louis	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86

M.	CHAVATTE	Philippe	Institut de Chimie Pharmaceutique Albert Lespagnol	86
M.	COURTECUISSE	Régis	Sciences végétales et fongiques	87
M.	CUNY	Damien	Sciences végétales et fongiques	87
Mme	DELBAERE	Stéphanie	Biophysique - RMN	85
Mme	DEPREZ	Rebecca	Chimie thérapeutique	86
M.	DEPREZ	Benoît	Chimie bioinorganique	85
M.	DUPONT	Frédéric	Sciences végétales et fongiques	87
M.	DURIEZ	Patrick	Physiologie	86
M.	ELATI	Mohamed	Biomathématiques	27
M.	FOLIGNÉ	Benoît	Bactériologie - Virologie	87
Mme	FOULON	Catherine	Chimie analytique	85
M.	GARÇON	Guillaume	Toxicologie et Santé publique	86
M.	GOOSSENS	Jean-François	Chimie analytique	85
M.	HENNEBELLE	Thierry	Pharmacognosie	86
M.	LEBEGUE	Nicolas	Chimie thérapeutique	86
M.	LEMDANI	Mohamed	Biomathématiques	26
Mme	LESTAVEL	Sophie	Biologie cellulaire	87
Mme	LESTRELIN	Réjane	Biologie cellulaire	87
Mme	MELNYK	Patricia	Chimie physique	85
M.	MILLET	Régis	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	MUHR-TAILLEUX	Anne	Biochimie	87
Mme	PERROY	Anne-Catherine	Droit et Economie pharmaceutique	86
Mme	ROMOND	Marie-Bénédicte	Bactériologie - Virologie	87
Mme	SAHPAZ	Sevser	Pharmacognosie	86
M.	SERGHERAERT	Éric	Droit et Economie pharmaceutique	86
M.	SIEPMANN	Juergen	Pharmacotechnie industrielle	85
Mme	SIEPMANN	Florence	Pharmacotechnie industrielle	85

M.	WILLAND	Nicolas	Chimie organique	86

Maîtres de Conférences - Praticiens Hospitaliers (MCU-PH)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	BLONDIAUX	Nicolas	Bactériologie - Virologie	82
Mme	DEMARET	Julie	Immunologie	82
Mme	GARAT	Anne	Toxicologie et Santé publique	81
Mme	GENAY	Stéphanie	Biopharmacie, Pharmacie galénique et hospitalière	81
M.	LANNOY	Damien	Biopharmacie, Pharmacie galénique et hospitalière	80
Mme	ODOU	Marie-Françoise	Bactériologie - Virologie	82

Maîtres de Conférences des Universités (MCU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	AGOURIDAS	Laurence	Chimie thérapeutique	85
Mme	ALIOUAT	Cécile-Marie	Parasitologie - Biologie animale	87
M.	ANTHÉRIEU	Sébastien	Toxicologie et Santé publique	86
Mme	AUMERCIER	Pierrette	Biochimie	87
M.	BANTUBUNGI-BLUM	Kadiombo	Biologie cellulaire	87
Mme	BARTHELEMY	Christine	Biopharmacie, Pharmacie galénique et hospitalière	85
Mme	BEHRA	Josette	Bactériologie - Virologie	87
M.	BELARBI	Karim-Ali	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	BERTHET	Jérôme	Biophysique - RMN	85
M.	BERTIN	Benjamin	Immunologie	87
M.	BOCHU	Christophe	Biophysique - RMN	85
M.	BORDAGE	Simon	Pharmacognosie	86
M.	BOSC	Damien	Chimie thérapeutique	86
M.	BRIAND	Olivier	Biochimie	87

Mme	CARON-HOUDE	Sandrine	Biologie cellulaire	87
Mme	CARRIÉ	Hélène	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
Mme	CHABÉ	Magali	Parasitologie - Biologie animale	87
Mme	CHARTON	Julie	Chimie organique	86
M.	CHEVALIER	Dany	Toxicologie et Santé publique	86
Mme	DANEL	Cécile	Chimie analytique	85
Mme	DEMANCHE	Christine	Parasitologie - Biologie animale	87
Mme	DEMARQUILLY	Catherine	Biomathématiques	85
M.	DHIFLI	Wajdi	Biomathématiques	27
Mme	DUMONT	Julie	Biologie cellulaire	87
M.	EL BAKALI	Jamal	Chimie thérapeutique	86
M.	FARCE	Amaury	Institut de Chimie Pharmaceutique Albert Lespagnol	86
M.	FLIPO	Marion	Chimie organique	86
M.	FURMAN	Christophe	Institut de Chimie Pharmaceutique Albert Lespagnol	86
M.	GERVOIS	Philippe	Biochimie	87
Mme	GOOSSENS	Laurence	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	GRAVE	Béatrice	Toxicologie et Santé publique	86
Mme	GROSS	Barbara	Biochimie	87
M.	HAMONIER	Julien	Biomathématiques	26
Mme	HAMOUDI-BEN YELLES	Chérifa-Mounira	Pharmacotechnie industrielle	85
Mme	HANNOTHIAUX	Marie-Hélène	Toxicologie et Santé publique	86
Mme	HELLEBOID	Audrey	Physiologie	86
M.	HERMANN	Emmanuel	Immunologie	87
M.	KAMBIA KPAKPAGA	Nicolas	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	KARROUT	Younes	Pharmacotechnie industrielle	85
Mme	LALLOYER	Fanny	Biochimie	87

Mme	LECOEUR	Marie	Chimie analytique	85
Mme	LEHMANN	Hélène	Droit et Economie pharmaceutique	86
Mme	LELEU	Natascha	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	LIPKA	Emmanuelle	Chimie analytique	85
Mme	LOINGEVILLE	Florence	Biomathématiques	26
Mme	MARTIN	Françoise	Physiologie	86
M.	MOREAU	Pierre-Arthur	Sciences végétales et fongiques	87
M.	MORGENROTH	Thomas	Droit et Economie pharmaceutique	86
Mme	MUSCHERT	Susanne	Pharmacotechnie industrielle	85
Mme	NIKASINOVIC	Lydia	Toxicologie et Santé publique	86
Mme	PINÇON	Claire	Biomathématiques	85
M.	PIVA	Frank	Biochimie	85
Mme	PLATEL	Anne	Toxicologie et Santé publique	86
M.	POURCET	Benoît	Biochimie	87
M.	RAVAUX	Pierre	Biomathématiques / Innovations pédagogiques	85
Mme	RAVEZ	Séverine	Chimie thérapeutique	86
Mme	RIVIÈRE	Céline	Pharmacognosie	86
M.	ROUMY	Vincent	Pharmacognosie	86
Mme	SEBTI	Yasmine	Biochimie	87
Mme	SINGER	Elisabeth	Bactériologie - Virologie	87
Mme	STANDAERT	Annie	Parasitologie - Biologie animale	87
M.	TAGZIRT	Madjid	Hématologie	87
M.	VILLEMAGNE	Baptiste	Chimie organique	86
M.	WELTI	Stéphane	Sciences végétales et fongiques	87
M.	YOUS	Saïd	Chimie thérapeutique	86
M.	ZITOUNI	Djamel	Biomathématiques	85

Professeurs certifiés

Civ.	Nom	Prénom	Service d'enseignement
Mme	FAUQUANT	Soline	Anglais
M.	HUGES	Dominique	Anglais
M.	OSTYN	Gaël	Anglais

Professeurs Associés

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	DAO PHAN	Haï Pascal	Chimie thérapeutique	86
M.	DHANANI	Alban	Droit et Economie pharmaceutique	86

Maîtres de Conférences Associés

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	CUCCHI	Malgorzata	Biomathématiques	85
M.	DUFOSSEZ	François	Biomathématiques	85
M.	FRIMAT	Bruno	Pharmacologie, Pharmacocinétique et Pharmacie clinique	85
M.	GILLOT	François	Droit et Economie pharmaceutique	86
M.	MASCAUT	Daniel	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	MITOUMBA	Fabrice	Biopharmacie, Pharmacie galénique et hospitalière	86
M.	PELLETIER	Franck	Droit et Economie pharmaceutique	86
M.	ZANETTI	Sébastien	Biomathématiques	85

Assistants Hospitalo-Universitaire (AHU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	CUVELIER	Élodie	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81

M.	GRZYCH	Guillaume	Biochimie	82
Mme	LENSKI	Marie	Toxicologie et Santé publique	81
Mme	HENRY	Héloïse	Biopharmacie, Pharmacie galénique et hospitalière	80
Mme	MASSE	Morgane	Biopharmacie, Pharmacie galénique et hospitalière	81

Attachés Temporaires d'Enseignement et de Recherche (ATER)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	GEORGE	Fanny	Bactériologie - Virologie / Immunologie	87
Mme	N'GUESSAN	Cécilia	Parasitologie - Biologie animale	87
M.	RUEZ	Richard	Hématologie	87
M.	SAIED	Tarak	Biophysique - RMN	85
M.	SIEROCKI	Pierre	Chimie bioinorganique	85

Enseignant contractuel

Civ.	Nom	Prénom	Service d'enseignement
M.	MARTIN MENA	Anthony	Biopharmacie, Pharmacie galénique et hospitalière

Faculté de Pharmacie de Lille

3 Rue du Professeur Laguesse – 59000 Lille
03 20 96 40 40
https://pharmacie.univ-lille.fr

L'Université n'entend donner aucune approbation aux opinions émises dans les thèses ; celles-ci sont propres à leurs auteurs.

Remerciements

À M Philippe Chavatte, je vous remercie d'avoir accepté d'encadrer ce travail et d'avoir présidé le jury. Je vous remercie de votre aide, de votre disponibilité, de votre compréhension et pour votre temps consacré à ce travail.

À Mme Delphine Allorge et M Thomas Morgenroth, je vous remercie d'avoir accepté de faire partie du jury de thèse.

À mes parents, pour m'avoir permis de réaliser ces études, d'avoir été des modèles de persévérance et de ténacité. Cette thèse clôture ces années d'études et est l'aboutissement de vos sacrifices et de votre soutien indéfectible.

À l'ensemble de ma famille, je vous remercie de m'avoir soutenu durant ces années, dans les moments de bonheur comme dans les moments difficiles.

À mes amis Benjamin, Alexis et Alexander, j'ai passé des années merveilleuses avec vous, merci pour votre soutien, votre bonne humeur et toutes ces soirées passées ensemble. Cette partie s'arrête mais un nouveau chapitre s'ouvre.

À Blandine, pour son soutien sans faille et ses encouragements tout au long de l'écriture de ce travail. Merci pour ton amour et pour tout ce que tu m'apportes au quotidien.

Sommaire

Table des abréviations

- **WGS** : Whole genome sequencing
- **VP** : Vrai positif
- **VN** : Vrai négatif

- **FP** : Faux positif
- **FN** : Faux négatif
- **VPP** : Valeur prédictive positive
- **VPN** : Valeur prédictive négative
- **ROC** : Receiving Operating Characteristics
- **SVM** : Support Vector Machine
- **KNN** : K-Nearest Neighbors
- **PCA** : Principal Component Analysis
- **ICA** : Independent Component Analysis
- **t-SNE** : t-distributed Stochastic Neighbor Embedding
- **DBSCAN** : Density-based spatial clustering of applications with noise
- **LSTM** : Long Short Term Memory
- **PLS** : Partial Least Squares regression
- **COX-2** : Cyclooxygénase-2
- **DBN** : Deep Belief Network
- **PIP** : Points d'Interaction Pharmacophorique
- **EC_{50}** : Concentration efficace médiane
- **HIF** : Hypoxia-inducible factor
- **BRD4** : Bromodomain-containing protein 4
- **PRMT5** : protein arginine methyltransferase 5
- **VEGFR2** : Vascular Endothelial Growth Factor Receptor 2
- **MDM2** : Murine Double minute 2
- **PDB** : Protein Data Bank
- **PAINS** : Pan Assay Interference Compounds
- **PPAR** : Peroxisome proliferator-activated receptor
- **RXR** : Retinoid X receptor
- **DDR1** : Discoidin domain receptor

Partie 1 : Revue bibliographique

I Introduction

Les premières traces d'usage de plantes renfermant des substances médicinales remontent au néolithique où l'on sait que le pavot était cultivé en quantité mais aucune information n'est disponible quant à la finalité de ces cultures. Les 660 tablettes d'argiles mésopotamiennes dites tablettes de Nineveh datent d'environ 1700 av. J.C. et décrivent l'utilisation de nombreuses plantes, avec une prise en compte des différentes parties. Dans la période pré-hellénistique les Papyrus d'Ebers, datant d'environ 1500 av. J.C, regroupent plus de 800 prescriptions, incluant la plupart des indications des tablettes de Nineveh. Les bains y étaient indiqués pour les problèmes cutanés et les purgatifs pour les désordres intestinaux. La magie, la superstition et la religion sont très présentes et prépondérantes.

Avec la théorie des humeurs et le corpus hippocratique, l'idée était de traiter un patient et non pas une maladie. Ce fût une avancée sur les méthodes superstitieuses et magiques car ce système adopte une vision holistique du patient, néanmoins, de par sa vision centrée sur le patient, elle empêche une étude approfondie des maladies et par conséquent, des études expérimentales. Une vision plus pragmatique se retrouve dans l'herboristerie grecque avec Dioscoride (40 - 90 ap. J.C.) et son *De Materia Medica* écrit entre 60 et 78 ap. J.C. Ce corpus est une source d'information majeure sur l'usage des plantes dans le monde hellénistique. S'inscrivant dans une démarche empirique plutôt que théorique, c'est un ouvrage qui regroupe des informations concernant plus de 600 plantes sur leurs propriétés humorales, leurs caractéristiques et leurs effets secondaires. Cette œuvre a fortement influencé le monde arabe après la chute de Rome. Pline l'ancien (23 - 79 ap. J.C.) a effectué un travail sur environ 900 plantes, moins rigoureux et moins détaillé que *De Materia medica*, il incorpore volontiers de la pensée magique et superstition. Puis Galien (129 - 199 ap. J.C.) fournit un énorme travail compilé en 20 volumes. *Opera Omnia* est un recueil de la connaissance médicale de son époque. Il y ajoute la notion suivante : "Les contraires se guérissent par les contraires". Un déséquilibre provoqué par un excédent d'humeur peut ainsi être traité en administrant une substance ayant les propriétés opposées aux humeurs excédentaires. Cette notion influencera la médecine pendant près de deux

millénaires avec l'utilisation de la polypharmacie ainsi que la notion considérant que le corps est capable de sélectionner les ingrédients pouvant corriger un défaut de balance humorale. Le renouveau de la médecine des humeurs dans le monde arabe a provoqué une demande inédite de formulations complexes requérant des compétences spécialisées pour leur fabrication. Ce qui a donné lieu au développement d'une nouvelle profession : apothicaire. Abu Ali Al-Hussain Ibn Abdallah Ibn Sina (Avicenne 980 - 1037 ap. J.C.), héritier de Galien, est connu pour son *Canon de la médecine* où il fusionne les traductions grecques et arabe. Cet ouvrage présente la description de 760 drogues.

La popularité des herbiers fut qu'ils devinrent une source principale d'informations thérapeutiques allant jusqu'à éclipser les écrits de Galien et Avicenne, et ils encouragèrent des médecins et apothicaires à conduire leurs propres essais. Ce fut le cas de James Lind qui, en 1747, fut le premier homme dans l'histoire médicale à conduire un essai clinique contrôlé dont le but fut de prouver que l'administration de jus de citron pouvait soigner le scorbut. Seuls les marins qui avaient reçu deux oranges et un citron et ceux ayant reçu du cidre furent guéris, les autres ayant reçu de l'huile de vitriol, du vinaigre, un électuaire d'ail, de la myrrhe ou de la moutarde de ne présentèrent aucunes améliorations cliniques. Un autre médecin précurseur des essais précliniques et cliniques fut Anton von Stöck (1731 - 1803), qui conduisit des expérimentations sur l'animal, mais aussi sur lui-même afin de déterminer les doses thérapeutiques de composés considérés comme trop dangereux pour avoir une application en thérapeutique. Il administra finalement ces composés à des doses croissantes aux patients jusqu'à observation d'effets bénéfiques. Il investigua les effets de la grande ciguë (*Conium maculatum L.*) dont il détermina des effets anti-cancéreux, effets réfutés au XIXème siècle, puis étudia les effets de la Datura (*Datura stramonium L.*), de la jusquiame noire (*Hyoscyamus niger*) ou encore de l'aconite (*Aconitum napellus*). Il en résulta une incorporation de nombreuses plantes au répertoire des thérapeutiques possibles.

Le passage de la pensée magique à une recherche pharmacologique rationnelle a fait découvrir de nombreuses substances thérapeutiques. Avec l'augmentation exponentielle de la quantité de connaissance en biologie et en pharmacologie, les méthodes statistiques et informatiques capables de prendre en charge cet important volume de données très hétérogènes représentent une opportunité dans la recherche pharmacologique. Dans ce travail nous verrons les techniques propres à l'exploitation des données massives, les techniques principales d'intelligence artificielle et de machine learning, de leurs utilisations dans une

stratégie de drug discovery, et enfin un exemple d'utilisation de ces techniques dans un flux de sélection de molécules candidates actives sur un récepteur.

II Big data et intelligence artificielle

II.a Principes de base

L'informatisation de la médecine et des moyens de la recherche a provoqué une augmentation massive des données générées et ce dans tous les domaines médicaux. On assiste ainsi à une augmentation exponentielle des possibilités d'intégration des données, que ce soit par exemple dans la pratique courante médicale par l'étude du parcours patient au cours de sa prise en charge hospitalière afin d'étudier en vraie vie les parcours de soin et d'optimiser sa prise en charge médicale [1] ou par une orientation vers une médecine personnalisée en intégrant des données issues d'analyses hétérogènes telles que les données "omiques" comme le séquençage sur génome entier (WGS), le transcriptome, le protéome et l'interactome [2,3] afin d'appréhender le patient dans sa plus grande complexité, sans compter la construction d'une médecine prédictive en intégrant les données de la clinique et de la recherche par le biais de module de prédiction de diagnostic et de progression de la maladie [2].

II.a.1 Données massives

Cette évolution dans la façon dont les données sont récoltées, traitées et utilisées a vu naître le domaine des données massives (Big data) pouvant être caractérisé par six adjectifs, les 6V :

- Volume : C'est la composante principale des données massives, le volume a un impact fort sur le traitement et le stockage des données.
- Variété : Elle reflète les nombreuses sources de données et l'hétérogénéité des données générées. On peut les séparer en données structurées (données facilement intégrables par une machine tel un tableau de données) et en données non structurées (données sans format prédéfini, source d'ambiguïtés comme les vidéos)
- Vitesse : Elle note la valeur éphémère de l'information, elle est fonction de la fréquence de renouveau de la donnée. Le flux de données étant massif et

continu, il est nécessaire d'effectuer un traitement rapide de l'information pour avoir une prise de décision pertinente et au plus proche de la réalité.

- Valeur : Cette caractéristique est intimement liée à l'objectif de l'étape analytique, c'est le potentiel de valeur qu'il est possible d'obtenir avec les résultats de l'étape analytique.
- Véracité : Toute donnée peut comporter des biais, des erreurs ou du bruit. La connaissance de ces limitations dans la collecte et l'analyse des données est critique et constitue la principale source d'erreur.
- Variabilité : Il est nécessaire de déterminer à quelle fréquence la structure des données change, cette notion est essentielle dans le traitement du langage naturel où un mot peut avoir plusieurs sens différents selon le contexte. Il est donc nécessaire de se donner la possibilité d'avoir une donnée ayant du sens en prenant en compte toutes les circonstances possibles.

L'ensemble des caractéristiques vues précédemment rendent impossible le stockage, la préparation et l'analyse des données sur un seul ordinateur, il est nécessaire de passer par des technologies spécifiques qui permettent une distribution du stockage et des calculs sur plusieurs unités de calcul, ces solutions étant disponibles pour un partage local ou décentralisé c'est à dire en "cloud computing" [4,5].

IV.a.2 Intelligence artificielle et machine learning

L'intelligence artificielle peut être définie comme étant l'imitation par une machine des capacités cognitives humaines mises en œuvre dans l'apprentissage et la résolution de problèmes [6]. L'intelligence artificielle inclut deux domaines majeurs, le machine learning et le deep learning (Figure 1).

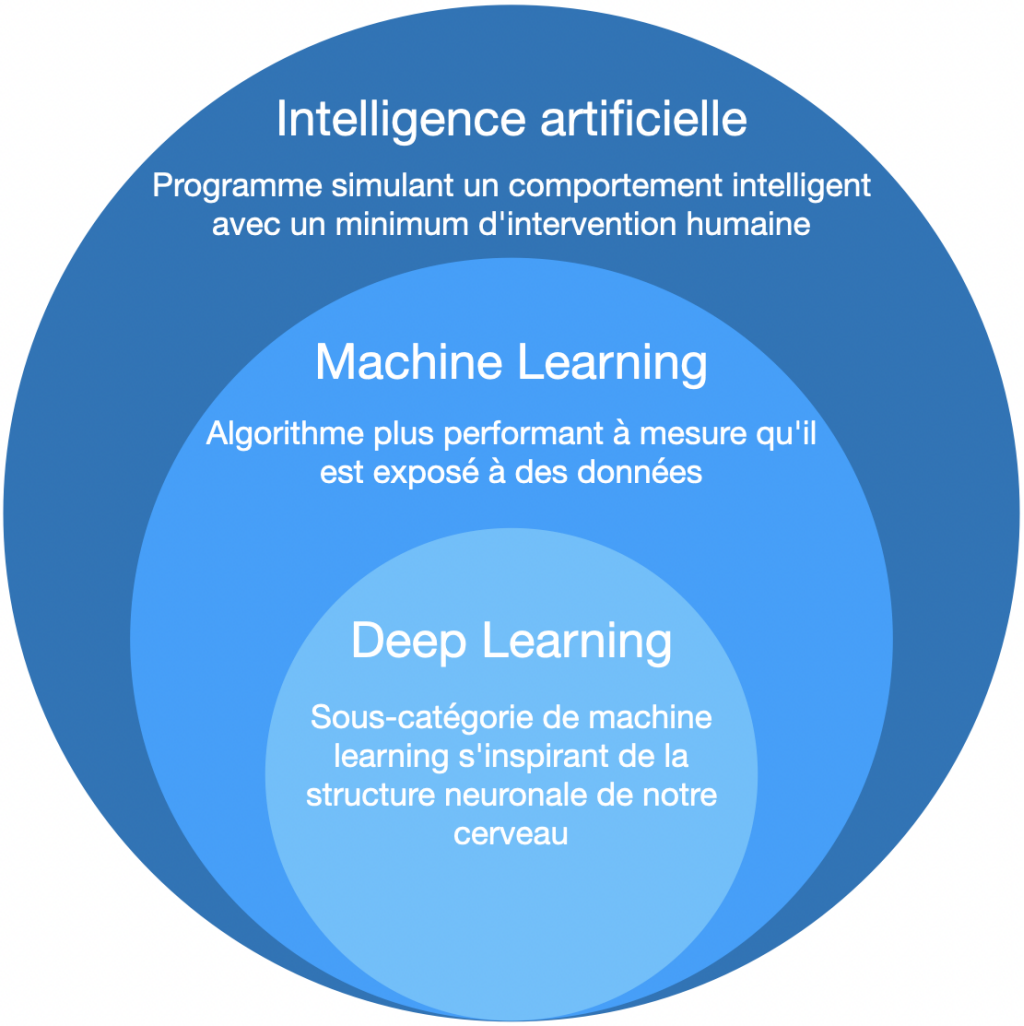


Figure 1 : Structuration des domaines de l'intelligence artificielle

Le machine learning est basé sur la capacité d'un système à "apprendre" à résoudre une problématique posée en fonction des données auxquelles il est exposé. Le machine learning est essentiellement une forme de statistiques appliquées avec une grande utilisation de la puissance de calcul des ordinateurs pour estimer des fonctions complexes. La plupart des algorithmes de machine learning se basent sur l'algorithme d'optimisation de descente de gradient. Ils peuvent être distingués en deux catégories, ceux destinés à un apprentissage supervisé et ceux destinés à un apprentissage non supervisé [7]. Un modèle de machine learning comporte des paramètres qui concernent les variables du modèle (le poids accordé à chaque variable du jeu de données par exemple) et des hyperparamètres c'est à dire les paramètres de l'algorithme d'apprentissage lui-même comme la profondeur de l'arbre de décision pour un algorithme d'arbre de décision ou le nombre de voisins à prendre en considération pour un algorithme des k plus proches voisins.

Le deep learning a été conçu pour résoudre des problèmes que les algorithmes de machine learning étaient incapables de résoudre. L'un d'eux est le fléau de la dimension qui apparaît quand on a un nombre réduit d'observations face à un nombre beaucoup plus important de variables, ou paramètres. Les algorithmes classiques de machine learning n'ont pas assez d'observations pour trouver une solution avec autant de paramètres et même s'ils en avaient assez, la quantité de données serait telle que c'est, cette fois-ci, le problème du temps de calcul qui empêcherait la résolution du problème. Le deep learning utilise des réseaux d'unités fonctionnelles, s'inspirant de l'organisation des neurones dans notre cerveau, offrant la possibilité d'intégrer des données hétérogènes, d'extraire des caractéristiques propres à chaque opérations faites dans les couches du réseaux, ceci afin de servir, par exemple, à la reconnaissance d'image, le traitement du langage naturel ou l'analyse de diagnostics médicaux [8].

II.b Machine Learning

Le machine learning se base sur la capacité surhumaine des ordinateurs pour résoudre des problèmes mathématiques basés sur les statistiques. Un algorithme de machine learning va apprendre et améliorer ses performances par cycle d'apprentissage en optimisant une valeur, qui peut être une erreur par rapport à une référence, appelée fonction de coût. A chaque cycle d'apprentissage, l'entraînement d'un algorithme de machine learning suit la démarche générale suivante :

1. Le jeu de données initial est séparé en jeu d'entraînement et en jeu de test. Différentes méthodes sont possibles, soit c'est un ratio fixe de telle façon que 70% des données sont incluses dans le jeu d'entraînement et 30% dans le jeu de test, soit le jeu de donnée initial est séparé en n parties et chaque partie est considérée tour à tour comme le jeu de test et le reste comme jeu d'entraînement, cette méthode est appelée la validation croisée.
2. Le modèle est entraîné sur le jeu d'entraînement.
3. Les performances du modèle sont évaluées sur le jeu de test

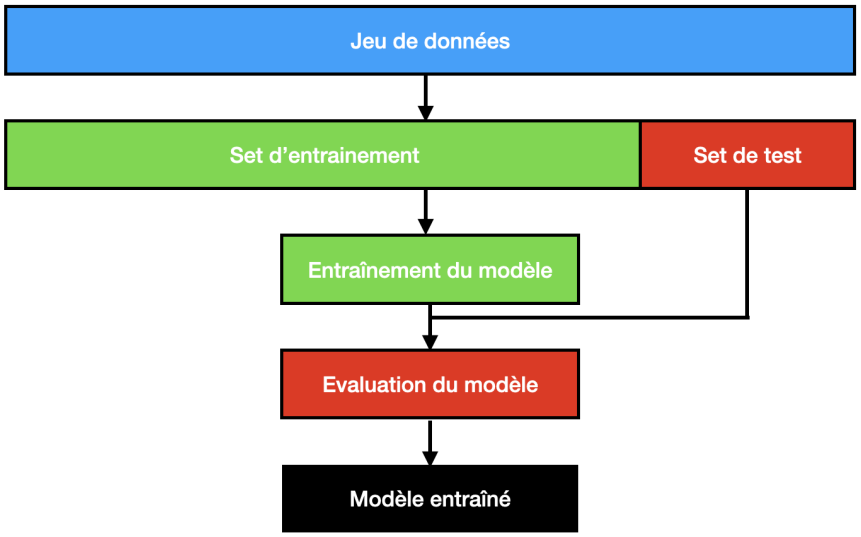


Figure 2 : Démarche classique d'entraînement d'un modèle

Une autre méthode est d'utiliser en plus des jeux d'entraînement et de test, un jeu de validation du modèle qui permet une optimisation des hyperparamètres du modèle, le modèle ne change plus qu'indirectement.

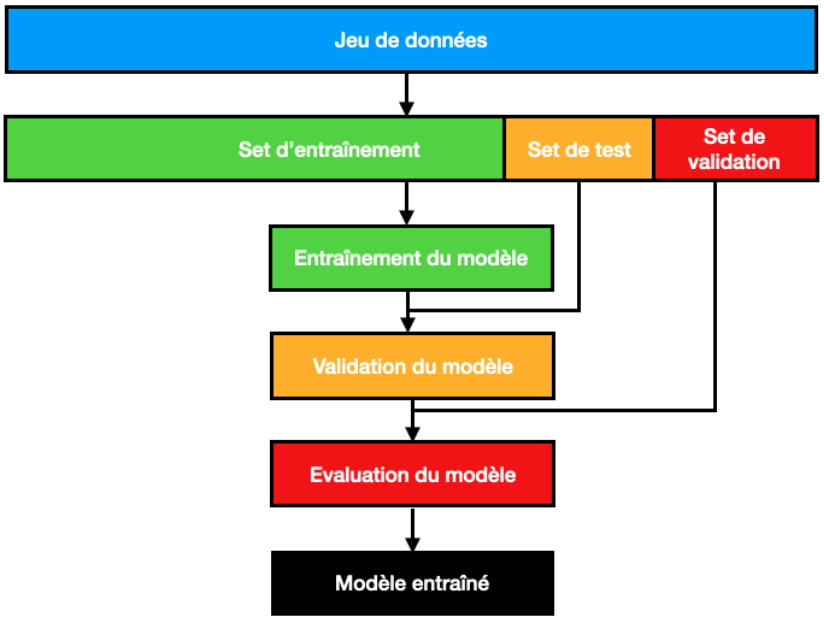


Figure 3 : Démarche d'entraînement d'un modèle avec étape de validation

II.b.1 Descente de gradient

Un algorithme de machine learning est capable de résoudre un problème statistique posé par des cycles d'apprentissage. Pour affiner la résolution de ce problème, il est nécessaire d'optimiser l'algorithme en minimisant ou en maximisant

une fonction, appelée critère, fonction de coût, fonction de perte ou fonction d'erreur quand il s'agit de minimiser cette fonction.

La méthode de descente de gradient est un algorithme d'optimisation utilisé pour trouver un maximum ou un minimum local. Pour cela la fonction que l'on veut optimiser doit remplir deux conditions :

- être dérivable, sinon il est impossible de calculer le gradient.
- être convexe, sinon aucun optimum n'est possible. Il est toutefois possible de d'optimiser une fonction quasi-convexe mais l'algorithme peut se retrouver bloqué sur un point particulier.

Le gradient peut être vu comme la pente de la courbe que représente la fonction de coût à un point donné et dans une direction donnée. Il est obtenu en calculant la dérivé première de la fonction de coût, noté $\nabla f(x)$.

Afin de trouver le minimum, l'algorithme de descente de gradient va choisir un point de départ, calculer le gradient à ce point, va faire un "pas", calculé de la façon suivante : $\eta \, . \, \nabla f(x)$, dans la direction opposée au gradient et répéter ces étapes jusqu'à ce que le nombre maximal d'itérations soit atteint ou que le "pas" soit devenu trop faible pour que l'optimisation soit suffisante (Figure 4).

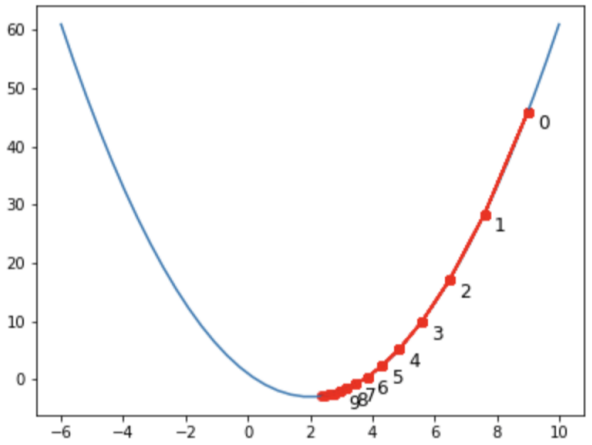


Figure 4 : Algorithme de descente de gradient

Le paramètre η est appelé le taux d'apprentissage, il a une forte influence dans la recherche de l'optimum car :

- S'il est trop faible l'algorithme nécessitera beaucoup d'itérations pour converger.
- S'il est trop élevé, l'algorithme aura beaucoup de mal à trouver l'optimum, voire diverger.

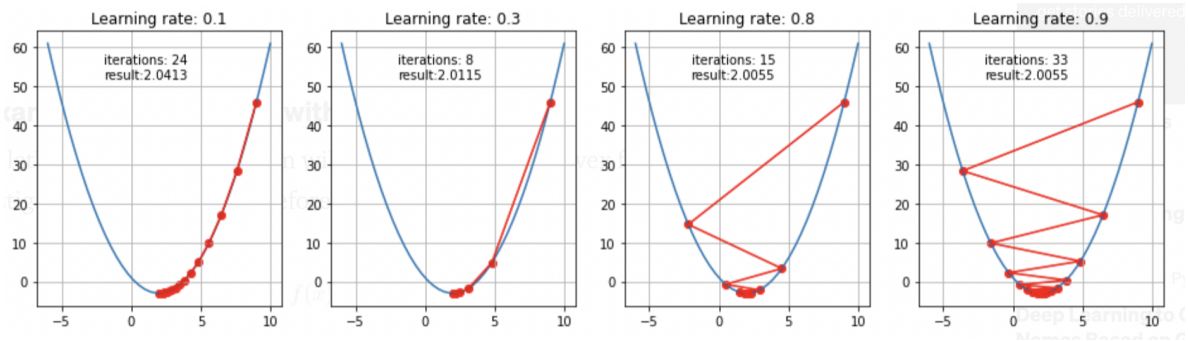


Figure 5 : Influence du taux d'apprentissage sur l'algorithme de descente de gradient

II.b.2 Evaluation d'un modèle

L'évaluation d'un modèle se base sur des critères différents selon le type de problème posé et selon le type de modèle développé.

Parmi les critères les plus souvent utilisés dans les problèmes de régression, on trouve :

- Le coefficient de détermination ajusté ou non. Il permet de mesurer combien la variabilité de la variable dépendante est expliquée par le modèle. Il est nécessaire d'ajuster cette valeur lorsque le modèle a beaucoup de variables explicables. En effet le modèle devient alors compliqué et il va très bien répondre aux données d'entraînement mais va avoir une performance médiocre sur les données de test, c'est ce qu'on appelle le sur-apprentissage. Il est nécessaire alors d'ajuster la valeur du coefficient de détermination en pénalisant l'ajout d'une nouvelle variable.
- L'erreur quadratique à la moyenne et sa racine carrée. L'erreur quadratique à la moyenne s'obtient en faisant le carré de l'écart entre les valeurs prédites par le modèle et les vraies valeurs expérimentales : $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$

  Ce critère n'a pas de valeur en soi, il doit être mis en perspective avec d'autres modèles afin de permettre une prise de décision. Le plus souvent, c'est la racine carrée de l'erreur quadratique à la moyenne qui est utilisée pour ramener les valeurs qui sont souvent élevées au même niveau que les valeurs prédites et ainsi faciliter l'interprétation.
- L'erreur moyenne absolue. Ce critère est similaire à l'erreur quadratique à la moyenne mais au lieu de calculer le carré de l'erreur, on calcule la valeur

absolue de cette erreur : $MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}\right|$. Néanmoins l'erreur quadratique à la moyenne pénalise beaucoup plus les grands écarts de prédictions que l'erreur absolue à la moyenne.

Enfin, lors des problèmes de classification, l'idée est d'évaluer la capacité d'un modèle à classer correctement les cas qu'on lui apporte. Les critères utilisés pour les problèmes de classification sont :

- La matrice de confusion. C'est le moyen le plus simple pour visualiser la capacité du modèle à classer les cas soumis. C'est une matrice carrée (même nombre de lignes que de colonnes) avec d'un côté les valeurs prédites et de l'autre, les valeurs vraies. La figure 6 montre une matrice de confusion pour un modèle de classification binaire. Les valeurs positives et négatives correctement prédites sont appelées respectivement vrais positifs (VP) et vrais négatifs (VN) et les valeurs positives et négatives incorrectement prédites sont appelées respectivement faux positifs (FP) et faux négatifs (FN).

		REEL	
		P	N
PREDICTION	N	Vrai positif	Faux positif
	P	Faux négatif	Vrai négatif

Figure 6 : Exemple de matrice de confusion pour une classification binaire

- Les critères dérivés de la matrice de confusion. Nombre de critères sont dérivés de la matrice de confusion avec pour chacun, un objectif particulier. Quatre critères sont fréquemment utilisés.
    - La précision. Elle mesure le pourcentage de cas correctement classés sur l'ensemble. Elle se calcule de la façon suivante : $Précision = \frac{VP + VN}{P_{réel} + N_{réel}}$. La précision rend compte des performances globales du modèle sur toutes les classes. Elle ne rend pas compte des performances du modèle dans chaque classe, c'est pour cela qu'il ne faut pas l'utiliser seule.

- Les valeurs prédictives positives et négatives (VPP et VPN respectivement). Ces critères mesurent la capacité d'un modèle à correctement prédire l'appartenance à une classe en particulier. Elles sont calculées comme suit : $VPP = \frac{VP}{VP + FP}$; $VPN = \frac{VN}{VN + FN}$. Ce critère est utile quand on se concentre sur la capacité du modèle à être précis dans une classe en particulier.
- La sensibilité. La sensibilité montre la capacité du modèle à classer un cas dans la classe A sachant que ce cas appartient à cette classe. Elle est calculée de la sorte : $Sensibilité = \frac{VP}{FN + VP}$. Ce critère est important pour les modèles de détection de pathologies par exemple.
- La spécificité. La spécificité rend compte de la capacité d'un modèle à ne pas classer un cas dans la classe A sachant que le cas n'appartient pas à cette classe. Elle est calculée de la façon suivante : $Spécificité = \frac{VN}{VN + FP}$. La spécificité est importante dans les tests diagnostiques pour certifier que le test ne détecte que les cas que l'on souhaite détecter (lorsque, par exemple, on ne veut détecter que les malades).
- La courbe ROC (Receiving Operating Characteristics). C'est la courbe qui a pour abscisse la spécificité et pour ordonnée la sensibilité. Comme la sensibilité et la spécificité sont deux valeurs en conflit, car plus on veut avoir de cas réellement positifs, plus on risque d'avoir des cas négatifs, la courbe ROC rend compte de l'équilibre entre ces deux critères et permet de choisir un optimum pour ces deux valeurs (Figure 7).

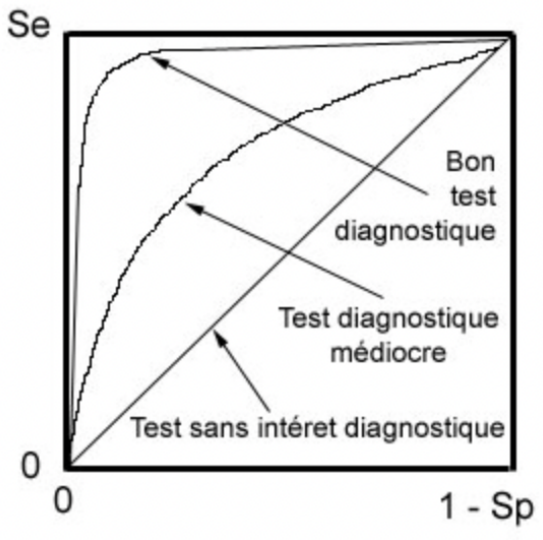


Figure 7 : Exemple de courbe ROC

Cette liste de critères est une liste non exhaustive et d'autres critères peuvent être choisis en fonction du problème posé et de l'objectif à atteindre.

II.b.3 Sous et sur-apprentissage

Durant l'apprentissage du modèle sur les données d'entraînement, il peut se présenter deux problèmes : le sur-apprentissage et le sous-apprentissage (Figure 8).

Le sur-apprentissage est le problème rencontré quand le modèle a déduit ses paramètres du bruit et des fluctuations aléatoires des données d'entraînement. Il est donc incapable de faire des prédictions correctes sur les données de validation ou de test.

Le sous-apprentissage apparaît quand le modèle n'est pas capable de modéliser les données d'entraînement et de généraliser sur les données de validation ou de test. D'une façon générale, le sous-apprentissage est facile à détecter avec les bons critères d'évaluation et les façons d'y répondre sont simples. Il suffit soit de complexifier le modèle en incluant de nouvelles variables à un modèle de régression par exemple, soit d'abandonner le modèle et d'essayer un autre modèle plus approprié.

De son côté, le sur-apprentissage est plus complexe à résoudre et il existe plusieurs moyens pour pallier ce problème.

Lors de l'étape d'entraînement on peut appliquer une validation croisée qui divise les données d'entraînements en plusieurs groupes, le modèle est entraîné sur

tous les groupes sauf un. Cette validation croisée sert à mieux évaluer les performances du modèle et de mieux détecter un éventuel sur-apprentissage.

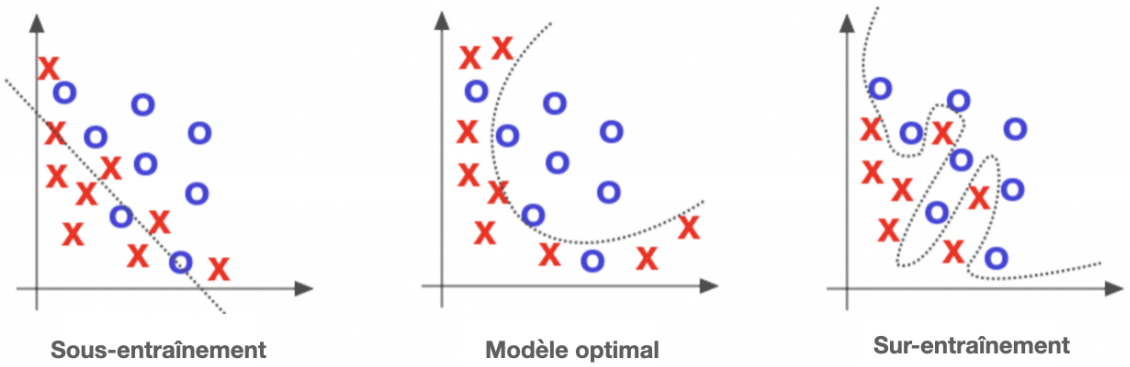


Figure 8 : Sur-apprentissage, apprentissage optimal et sous-apprentissage

Il est possible de mettre en place une sélection de variables avant entraînement et des méthodes de régularisation pendant l'entraînement du modèle pour réduire les risques. La sélection de variables permet de supprimer des variables corrélées entre elles et la régularisation diminue la variance au sein du jeu de données.

Enfin, il est possible de forcer l'arrêt de l'entraînement du modèle de façon précoce. Pour cela, il est nécessaire de trouver la durée d'entraînement optimale pour avoir le meilleur compromis sur-entraînement / sous-entraînement.

II.b.4 Apprentissage supervisé

La notion d'apprentissage supervisé signifie que les données d'entrée du modèle comportent à la fois les variables prédictives et la variable prédite. Le modèle va modifier les relations entre les variables prédictives afin de prédire au mieux la variable prédite, [9] (par le biais d'une optimisation par l'algorithme de descente de gradient par exemple). L'algorithme "apprend" donc la relation $f: x \rightarrow y$ lors de l'entraînement (x étant la variable prédictive et y la variable prédite) et se sert de cette relation pour prédire les données qu'on lui apporte par la suite.

Beaucoup d'algorithmes d'apprentissage supervisé existent, nous allons en voir quelques-uns.

II.b.4.a Modèles linéaires

Les modèles linéaires modélisent la relation entre les variables explicatives et les variables prédites de façon linéaire : $\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$, où le vecteur $w = (w_1, \ldots, w_p)$ représente les coefficients des p variables prédictives et w_0 l'ordonnée à l'origine.

II.b.4.a.1 Régression linéaire simple

Le modèle de régression linéaire simple va modifier la valeur des coefficients du vecteur w de façon à minimiser la somme des carrés des résidus : $min \left|\left|X_w - y\right|\right|^2$ (Figure 9)

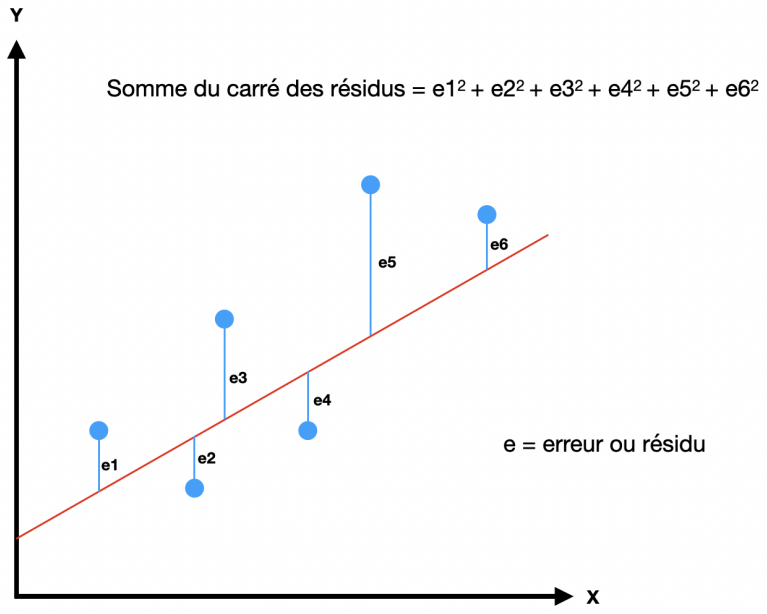


Figure 9 : Illustration de la fonction de coût (somme du carré des résidus) d'un modèle de régression linéaire

L'estimation des coefficients dépend de l'indépendance des variables prédictives et au fur et à mesure que des variables prédictives sont ajoutées au modèle, l'indépendance des variables devient compromise et on voit apparaître une multicolinéarité entre les variables. Ceci a pour conséquence d'exagérer la valeur des coefficients et donc de fausser les paramètres du modèle.

II.b.4.a.2 Régression Ridge

La régression Ridge vise à pallier les valeurs aberrantes de coefficients en imposant une pénalité sur la taille des coefficients : $min \left|\left|X_w - y\right|\right|^2 + \alpha||w||^2$.

La paramètre de complexité $\alpha \geq 0$ contrôle la force de la pénalité, plus sa valeur est haute plus les variables sont résistantes à la multicolinéarité.

II.b.4.a.3 Régression Lasso

La régression Lasso vise aussi à pallier le problème de multicolinéarité en imposant une pénalité égale à la valeur absolue du coefficient. La pénalité est donc plus grande qu'avec la régression Ridge : $min \frac{1}{2n_{echantillons}} \left|\left|X_w - y\right|\right|^2 + \alpha||w||$.

Ceci provoque une sélection d'un sous-ensemble de variables. On peut donc avoir un modèle final avec moins de paramètres (de variables explicatives) que le modèle initial. Cette technique peut être utilisée quand on veut sélectionner des variables pertinentes dans un ensemble de données de grande dimension.

II.b.4.a.4 Elastic-Net

Le modèle Elastic-Net [10] a été introduit pour améliorer la gestion de la multicolinéarité. En effet la régression Lasso ne va choisir qu'une seule variable parmi un ensemble de variables colinéaires, au détriment des autres qui pouvaient aussi être importantes dans l'interprétation du modèle. Pour cela on ajoute à la régression Lasso une pénalité Ridge : $min \frac{1}{2n_{echantillons}} \left|\left|X_w - y\right|\right|^2 + \alpha\rho||w|| + \frac{\alpha(1-\rho)}{2}||w||^2$.

II.b.4.a.5 Régression logistique

La régression logistique permet d'étudier les relations entre une variable expliquée qualitative et des variables explicatives quantitatives ou qualitatives. En dépit du terme régression, ce modèle est plus souvent utilisé pour résoudre des problèmes de classification ou de prédiction de survenue d'un événement.

Dans le cas d'un problème de classification binaire la probabilité qu'un événement soit classé dans la classe 1 est noté : $p(X_i) = \frac{1}{1+exp(-X_i w - w_0)}$.

II.b.4.a.6 Perceptron

Le perceptron est un modèle de classification binaire très simple [11]. C'est la base des réseaux de neurones contemporains calquée sur le fonctionnement des neurones biologiques. Le perceptron intègre en entrée des informations, les intègre en leur apportant des poids spécifiques et somme leur valeur. Il envoie un signal de sortie si la valeur obtenue active une fonction d'activation, tout comme le neurone qui agrège les informations issues de ses dendrites et envoie un influx nerveux par son axone (Figure 10).

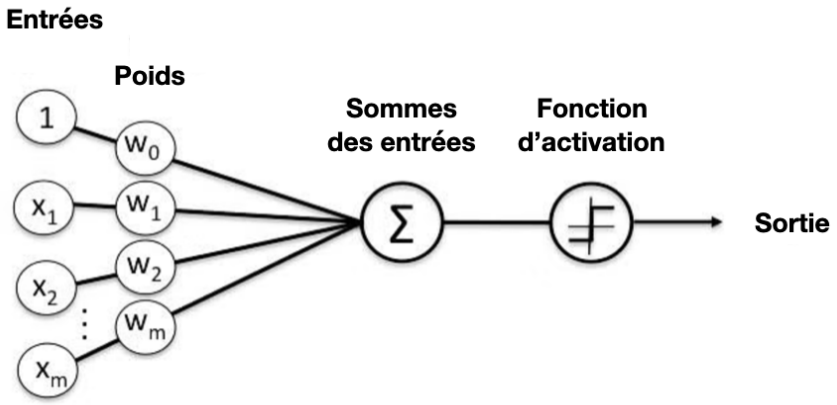


Figure 10 : Illustration d'un perceptron

La fonction d'activation la plus simple est la fonction de Heaviside (Figure 11) qui est la fonction échelon. Le perceptron s'active si la somme de ses entrées pondérées de leur poids respectif est supérieure à zéro.

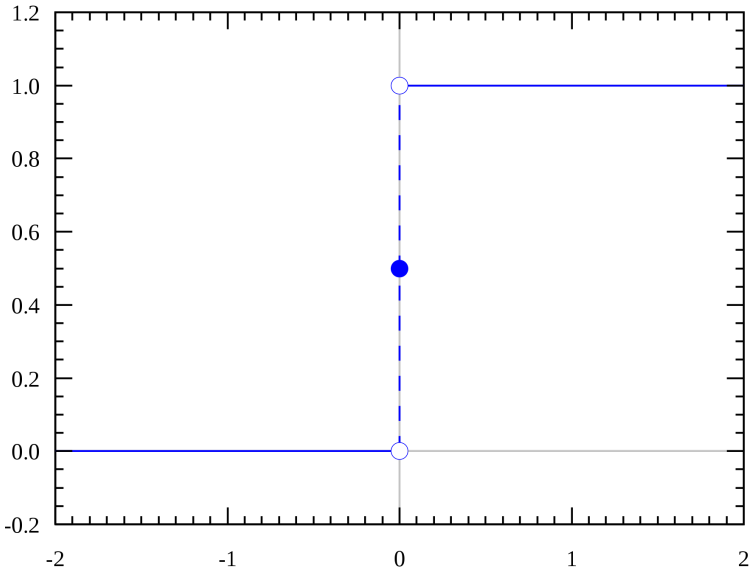


Figure 11 : Fonction de Heaviside

Mais d'autres fonctions d'activations sont possibles comme le montre la figure 12.

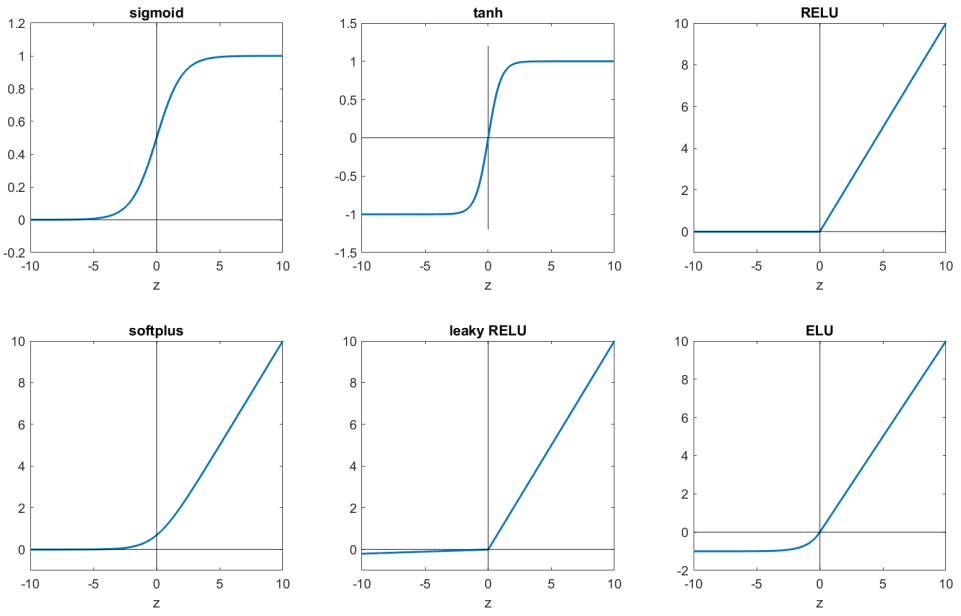


Figure 12 : Fonctions d'activation possibles

II.b.4.b Support Vector Machine (SVM)

Le modèle SVM est utilisé pour répondre à des problèmes de classification. Son objectif est de trouver un hyperplan (ligne pour un espace de dimension 2, surface pour un espace de dimension 3, volume pour un espace de dimension 4, et ainsi de suite) capable de séparer du mieux possible les données comme le montre la figure 13.

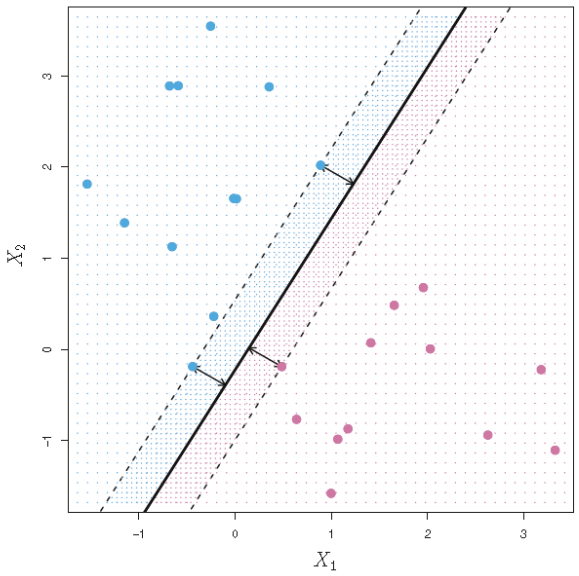


Figure 13 : Support Vector Machine

Le modèle va chercher à maximiser la marge entre les points des différentes classes et la droite centrale, plus la marge est grande plus le modèle sera capable de généraliser. Ce type de modèle est rapide à entraîner sur des jeux de données raisonnables et a une bonne précision de prédiction à condition que les données ne contiennent ni trop de bruit ni trop de données aberrantes.

II.b.4.c Algorithme des plus proches voisins

L'algorithme des plus proches voisins ou KNN (K-Nearest Neighbors) peut être utilisé pour la classification ou la régression. Cet algorithme est particulier car il ne construit pas de modèle à proprement parler, il se base sur le jeu de données entier pour effectuer ses prédictions. Pour une nouvelle donnée dont on veut prédire la valeur (régression) ou la classe (classification), l'algorithme va chercher les K échantillons du jeu de données les plus proches de notre observation. Ensuite l'algorithme va se baser sur la valeur de ces voisins pour prédire celle de la nouvelle observation.

Dans le cas d'une régression, ce peut être la moyenne ou la médiane des valeurs retenues tandis que dans le cas d'une classification, ce peut être le mode des valeurs retenues.

Il existe plusieurs méthodes de calcul de distance, notamment la distance euclidienne, la distance de Hamming, de Minkowski ou la distance de Manhattan. L'algorithme des plus proches voisins est un algorithme très simple qui ne nécessite

pas d'entraînement. Par contre, il est nécessaire de conserver en mémoire l'ensemble des données qui servent à établir les prédictions à la volée.

II.b.4.d Arbre de décision

L'algorithme d'arbre de décision est un algorithme qui permet de résoudre des problèmes de classification ou de régression en apprenant des règles de décisions simples du jeu de données d'entraînement.

L'arbre de décision se construit en séparant le jeu de données selon les valeurs d'une variable du jeu de données. A chaque séparation la prédiction se fait de plus en plus précise, puis on sépare le jeu de données selon une autre variable et ainsi de suite jusqu'à ce que toutes les variables soient utilisées ou jusqu'à ce que le nombre maximal de séparation autorisé par l'utilisateur soit atteint.

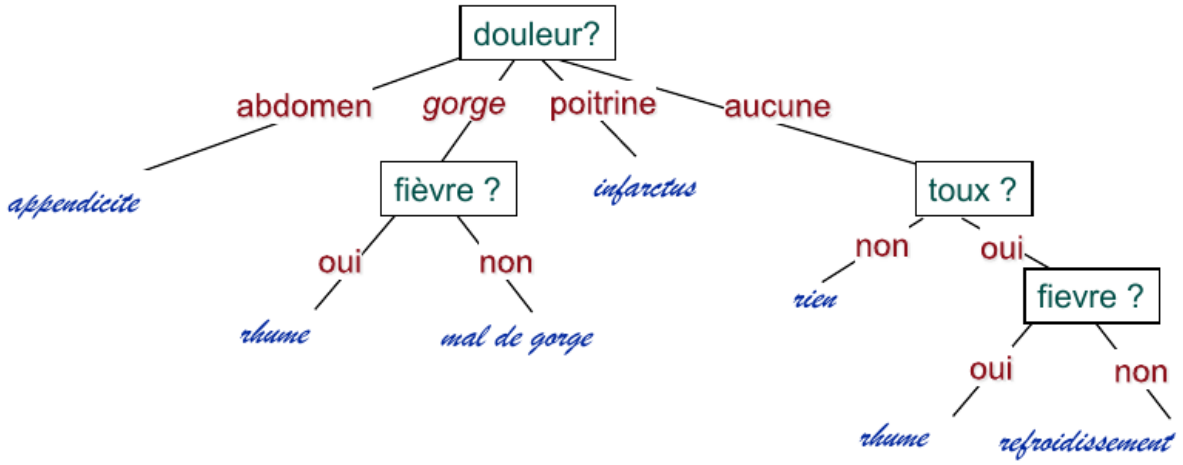


Figure 14 : Exemple d'un arbre de décision

Les avantages de ce modèle résident dans le fait qu'il est facilement visualisable et compréhensible, qu'il effectue implicitement une sélection de variables, qu'il est capable de traiter des données quantitatives, continues ou discrètes, et qualitatives. Néanmoins ces avantages sont à mettre en perspective avec ses inconvénients, l'algorithme par arbre de décision est enclin au sur-apprentissage. Il est sensible au changement de variance dans le jeu de donnée et il crée des arbres de décision biaisés si les classes à déterminer sont déséquilibrées dans le jeu d'entraînement.

II.b.4.e Algorithmes de sélection de variables

Ces algorithmes servent soit à sélectionner les variables pertinentes dans le but de les utiliser plus tard dans un modèle, soit à réduire les dimensions d'un jeu de données qui en comporte déjà beaucoup.

Différentes méthodes sont possibles pour retirer des variables. On peut citer, par exemple, celle qui consiste à éliminer les variables qui ont une variance en dessous d'un certain seuil. La sélection de variables peut se faire d'après des tests univariés tels que le test du chi-deux ou le test F dans le but de ne garder que les variables les plus différentes.

Enfin, une autre méthode consiste à entraîner le modèle sur l'ensemble des données, de déterminer l'importance des variables dans ce premier modèle puis de supprimer les variables avec la plus faible importance, de réitérer cette démarche jusqu'à obtenir le nombre désiré de variables.

D'autres méthodes sont possibles, comme la pénalisation vue dans la régression Lasso ou encore le recours à des arbres de décision.

II.b.5 Apprentissage non supervisé

L'idée de l'apprentissage non supervisé réside dans le fait que l'utilisateur ne guide pas le modèle dans son entraînement. Il laisse l'algorithme découvrir des motifs et des informations qui étaient jusqu'alors voilés. Deux composantes de l'apprentissage non supervisé : la réduction de dimension (Décomposition en composantes et Manifold learning) et la classification (clustering), sont présentées.

II.b.5.c Décomposition en composantes

L'objectif de ces techniques est de décomposer un jeu de données selon des vecteurs, axes ou encore composantes expliquant au mieux la variance du jeu de données. On a au final un jeu de données avec moins de variables mais avec une variance qui reste conservée.

II.b.5.c.1 Analyse en composantes principales

L'analyse en composantes principales (PCA) vise à décomposer un jeu de données en composantes orthogonales expliquant au maximum sa variance. La première étape est une étape de standardisation des données pour contenir l'ensemble des variables du jeu de données dans les mêmes bornes. Ensuite l'algorithme calcule la matrice de covariance des variables du jeu de données. Puis il calcule les vecteurs propres et les valeurs propres. Les vecteurs propres de la

matrice de covariance sont les “directions” des axes qui projettent le plus de variance, ce sont les composantes principales. Les valeurs propres sont les coefficients attachés aux vecteurs propres, elles représentent la part de variance propre à chaque composante principale. Pour finir, seul les composantes représentant la majeure partie de la variance du jeu de données, d’après leur valeur propre, sont conservées, on obtient donc une matrice avec les vecteurs propres sélectionnés appelés vecteur des variables.

II.b.5.c.1 Analyse en composante indépendante

L’analyse en composante indépendante (ICA) a pour objectif de séparer un signal comportant plusieurs variables en des sous-composantes additives qui sont au maximum indépendantes. Contrairement à l’ACP, l’ACI ne fait que séparer un signal en plusieurs sous-signaux indépendants.

II.b.5.a Manifold learning

Le manifold learning est une approche de réduction de dimension non linéaire. Il se base sur le postulat que la forte dimensionnalité des données n’est que fictive. Un jeu de données avec deux ou trois variables (dimensions) est assez facile à représenter mais quand le nombre de dimensions est supérieur à trois la représentation est difficile. Il est alors nécessaire de passer par des algorithmes de réduction de dimension qui vont apprendre des données à fortes dimensions, sans classification prédéterminée pour projeter les données vers un espace de dimensions réduit.

II.b.5.a.1 ISOMAP

L’algorithme ISOMAP (Isometric Feature Mapping) est un algorithme de réduction de dimension non-linéaire qui cherche à préserver la distance géodésique entre les points d’un jeu de données. La distance géodésique est la distance qui sépare deux points en respectant la “forme” du jeu de donné. Elle est à mettre en comparaison avec la distance euclidienne qui, elle, ne calcule que la distance séparant deux points indépendamment du jeu de données (Figure 15).

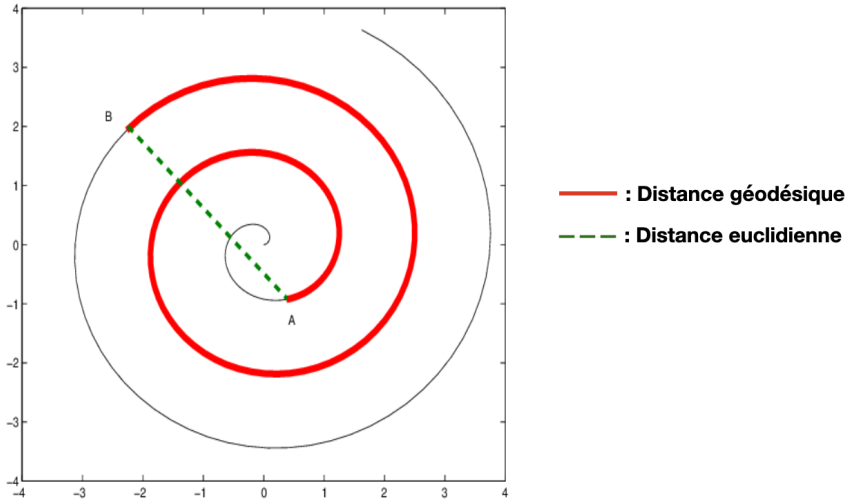


Figure 15 : Distance géodésique vs distance euclidienne

Comme l'objectif est de réduire le nombre de dimensions avec le moins de perte possible, il est naturel de chercher à respecter la distance géodésique tout au long de l'algorithme. Pour cela l'algorithme utilise les relations locales entre un nombre déterminé de points afin de réduire localement le nombre de dimensions et ainsi chercher à respecter la "forme" des données.

II.b.5.a.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

L'algorithme t-SNE prend en compte non pas la distance séparant les points deux à deux mais convertit cette distance en probabilité.

La première étape consiste à calculer les similarités entre les points du jeu de données par le calcul de probabilités conditionnelles pour obtenir une liste de probabilités conditionnelles de référence.

Puis les données sont représentées dans un espace de dimension moins grand que l'espace de référence en les représentant de façon aléatoire car la représentation idéale est inconnue et l'algorithme calcule la liste des similarités des données.

Enfin, l'algorithme converge si les deux listes de similarités sont identiques, on a donc une représentation des données dans un espace de dimension plus faible que les données originelles mais avec la même distribution des données. Pour cela l'algorithme utilise la divergence de Kullback-Leibler et la descente de gradient pour minimiser cette divergence.

La figure 16 montre la comparaison entre deux techniques de réduction de dimension, la PCA et t-SNE, sur le jeu de données MNIST qui est une base de

données de chiffres écrits à la main. Elle montre la capacité de l'algorithme à respecter la structure locale.

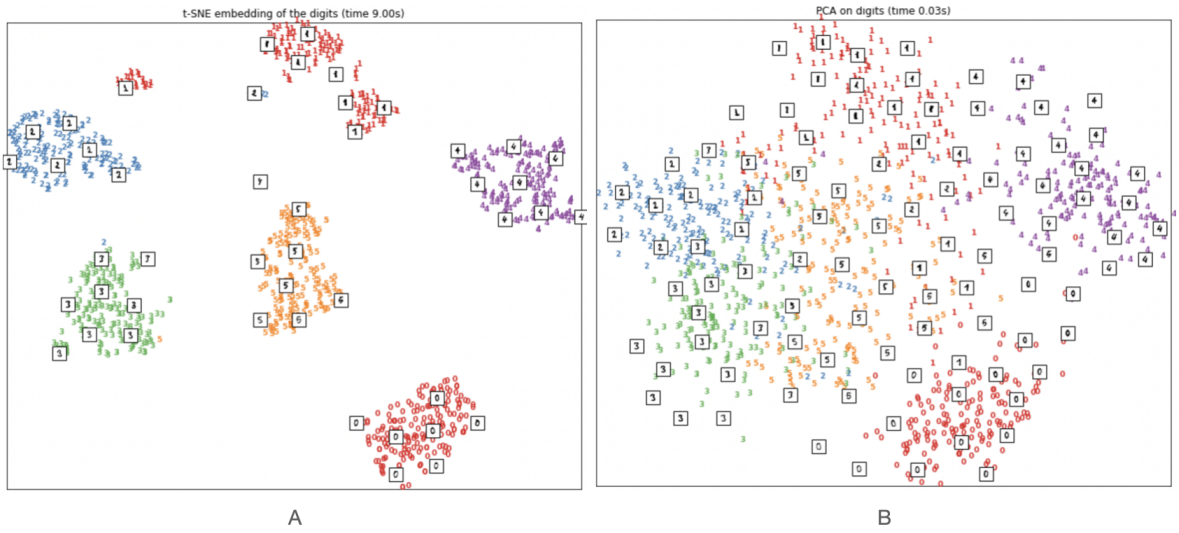


Figure 16 : Comparaison des algorithmes de PCA (A) et t-SNE (B) sur les données MNIST

II.b.5.b Clustering

Les algorithmes de clustering visent à regrouper les données similaires, à apprendre la structure présente au sein du jeu de données.

II.b.5.b.1 Algorithme des k-moyennes

L'algorithme des k-moyennes vise à regrouper les données par groupes de variance égale en cherchant les centroïdes qui minimisent la somme des carrés à la moyenne intra-groupe. Cet algorithme nécessite que l'utilisateur spécifie le nombre de groupes dans lesquels classer les données.

Tout d'abord l'algorithme définit aléatoirement le nombre de centroïdes voulus par l'utilisateur. Il associe ensuite aux centroïdes les points du jeu de données les plus proches, par défaut via la distance euclidienne. Ensuite l'algorithme calcule les centroïdes qui seront le centre de gravité des points associés. Il répète cette opération jusqu'à ce que les centroïdes ne bougent plus.

Pour déterminer le nombre idéal de clusters, une des méthodes les plus utilisées est la méthode du coude. C'est une méthode graphique qui consiste à choisir le nombre de clusters à partir duquel l'inertie, qui est la somme des distances entre chaque point et son centroïde associé, se stabilise (Figure 17).

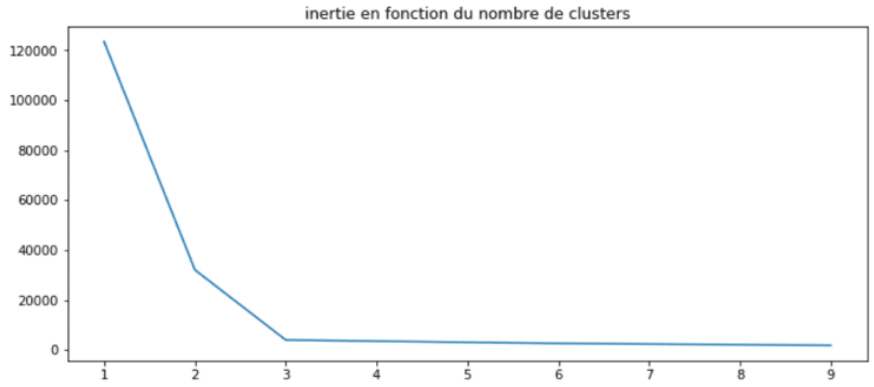


Figure 17 : Méthode du coude pour déterminer le nombre idéal de clusters, ici trois.

Une autre méthode plus précise consiste à calculer le coefficient de silhouette qui est le rapport entre la moyenne des distances intra-clusters et le maximum entre la moyenne des distances intra-cluster et la distance moyenne au cluster le plus proche : $s = \frac{b-a}{mac(a,b)}$ avec a la moyenne des distances intra-cluster et b la distance moyenne au cluster le plus proche.

La valeur du coefficient est comprise entre [-1:1]. Une valeur de -1 signifie que l'observation est associée au mauvais centroïde et donc au mauvais cluster, tandis qu'une valeur de +1 signifie que l'observation est à l'intérieur de son cluster, pour finir une valeur proche de zéro signifie qu'elle est proche de la frontière du cluster. On choisit donc le nombre de clusters qui a la valeur du coefficient de silhouette la plus élevée.

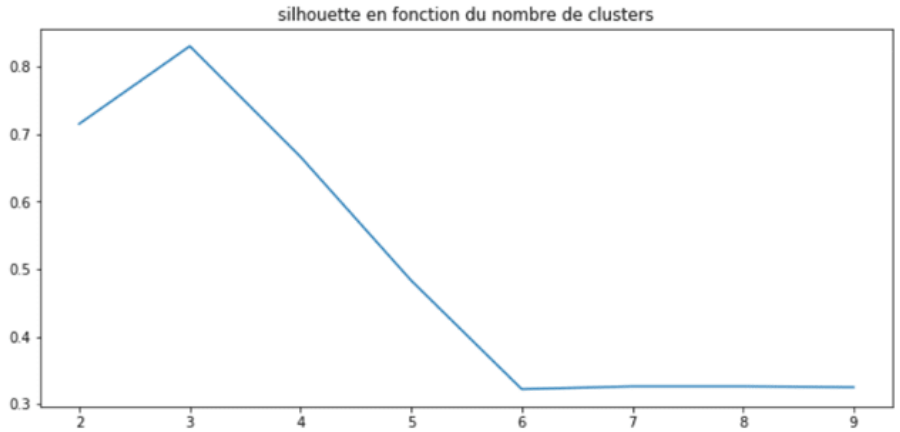


Figure 18 : Graphique du coefficient de silhouette en fonction du nombre de clusters.

Les performances de cet algorithme sont étroitement liées à la structure des données comme le montre la figure 19.

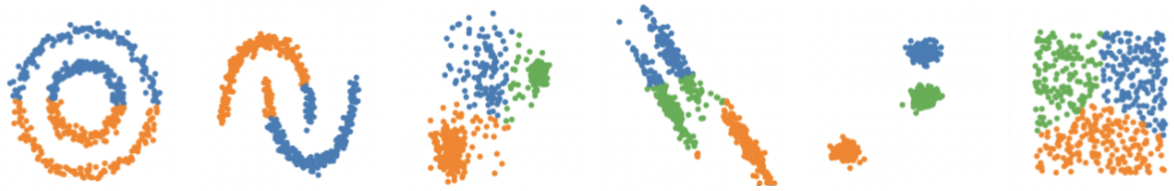


Figure 19 : Influence de la structure des données sur les performances de l'algorithme des k-moyennes

II.b.5.b.2 Clustering hiérarchique

Les algorithmes de clustering hiérarchique constituent une famille d'algorithmes qui trouvent des clusters imbriqués en séparant ou en fusionnant les données successivement selon que le clustering hiérarchique soit ascendant ou descendant. La visualisation des résultats de ces algorithmes se fait sous forme de dendrogramme (Figure 20).

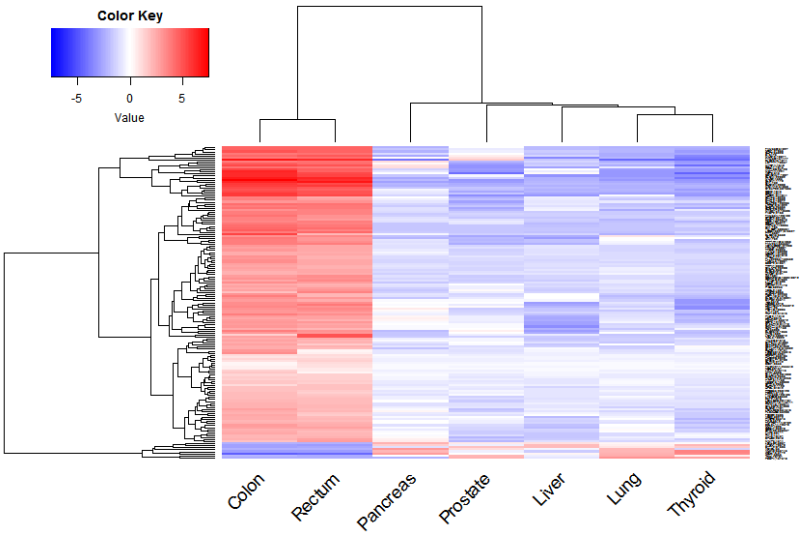


Figure 20 : Représentation en dendrogrammes de données de RNAseq

Premièrement le clustering hiérarchique ascendant procède de la façon suivante :

- Il considère chaque point comme étant un cluster individuel.

- Il calcule la matrice des distances
- Les points qui sont proches sont regroupés dans un seul et même cluster.
- L'algorithme réitère l'étape de calcul de la matrice des distances et le regroupement.

Deuxièmement le clustering hiérarchique descendant considère l'ensemble du jeu de données comme un seul cluster puis divise le jeu de données selon un critère de similarité. Les données différentes des autres données vont donc former un autre cluster. Ce type d'algorithme de clustering hiérarchique est peu utilisé en pratique.

II.b.5.b.3 DBSCAN

L'algorithme DBSCAN (density-based spatial clustering of applications with noise) vise à diviser le jeu de données en n clusters homogènes et compact, il a recours à la notion de densité compacte. Il fonctionne comme suit :

- Pour chaque observation, l'algorithme regarde le nombre de points situés dans son voisinage à une distance ε.
- Si l'algorithme compte un certain nombre d'observations dans ce ε-voisinage alors l'observation est considérée comme une observation cœur, il a alors décelé une observation dense.
- Toutes les observations proches d'une observation coeur appartiennent au même cluster, on a donc une propagation de proche en proche des observations coeur qui constituent le cluster.
- L'algorithme considère que toute observation qui n'est pas une observation cœur ou qui n'a pas dans son voisinage d'observation cœur est une anomalie.

Pour cet algorithme il est donc nécessaire de spécifier la distance ε et le nombre d'observations nécessaires pour considérer un point comme une observation cœur.
Pour une distance ε trop faible toutes les observations du jeu de données seront considérées comme des anomalies. *A contrario* si ε est trop grand, l'ensemble du jeu de données sera considéré comme un seul et même cluster.

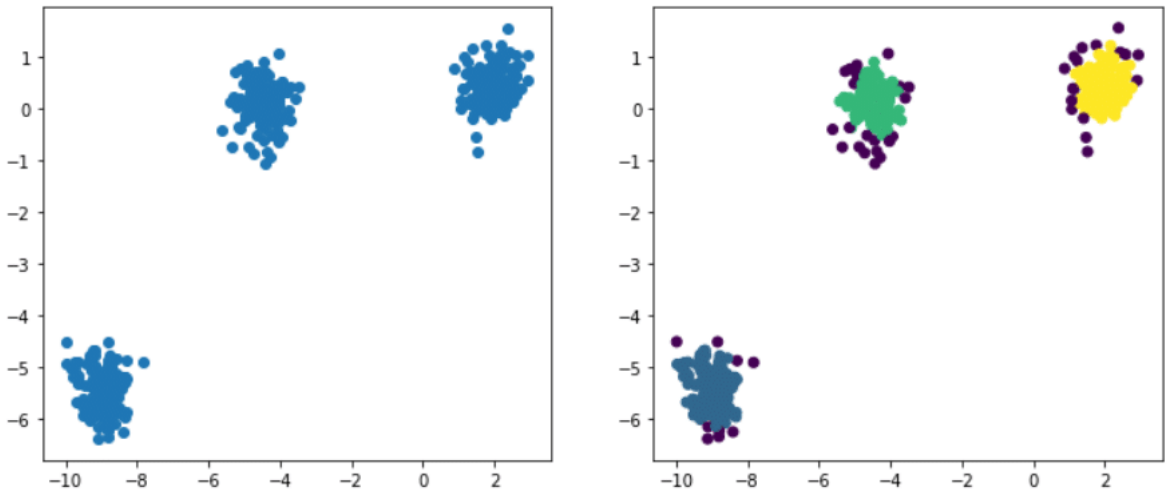


Figure 21 : Représentation d'un clustering par DBSCAN

II.c Deep Learning

Les algorithmes de machine learning traditionnels sont limités dans leur capacité à traiter les données naturelles dans leur forme brute. Pour mettre en place un algorithme de machine learning capable de reconnaître des motifs à l'intérieur d'une image, il est nécessaire de construire précautionneusement un modèle d'extraction de variables faisant appel à des compétences d'expert du domaine, afin d'obtenir à partir des données brutes une représentation que le système de détection de motif est capable d'intégrer et de classifier. Cette capacité des algorithmes de découvrir automatiquement des représentations nécessaire à une classification ou une détection s'appelle l'apprentissage des représentations [7].

Les méthodes de Deep Learning sont des techniques d'apprentissages des représentations avec plusieurs niveaux de représentation, obtenus en combinant de simples modèles non linéaires qui transforment la représentation vers une représentation d'un niveau plus abstrait. L'association de ces modèles simples permettent l'apprentissage de tâches très complexes, on peut noter en médecine l'analyse d'image [12] (scanner, radio, IRM), analyse de séquence génomique [13] ou encore la classification et prédiction de structure de protéines [14–16].

Parmi les algorithmes de Deep Learning les plus connus on trouve l'unité principale des réseaux de neurones, que nous avons déjà vu dans la section consacrée au machine learning, qui est le perceptron, les réseaux neuronaux récurrents ou encore les réseaux de neurones convolutifs.

II.c.1 Réseau de neurones artificiel

Les réseaux de neurones artificiel calquent l'organisation neuronale de notre cerveau. Ils sont constitués d'une couche de neurones (perceptron) d'entrée, une ou plusieurs couches cachées et d'une couche de sortie (Figure 22). Ce type de réseau est aussi appelé perceptron multicouches.

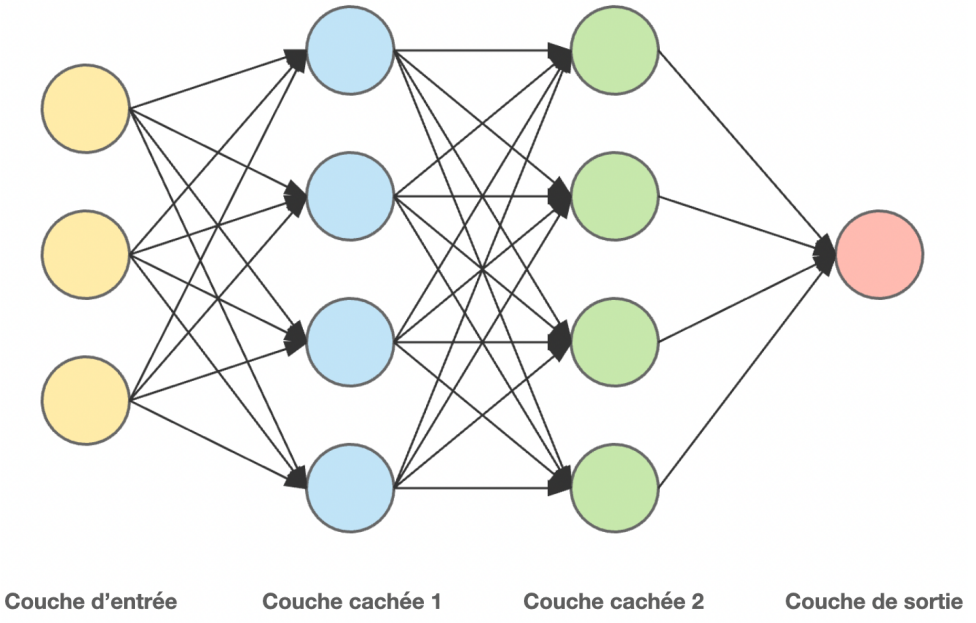


Figure 22 : Réseau de neurones artificiels

L'unité constitutive d'un réseau de neurones artificiels est le perceptron (Figure 23) qui attribue un poids à chacune de ses entrées. Dans le réseau on a donc un poids w_i pour chaque entrée d'un neurone. L'entraînement d'un réseau de neurones artificiels se déroule comme suit :

- Tout d'abord l'ensemble des poids des entrées est fixé aléatoirement.
- Pour chaque cas soumis à l'algorithme une prédiction est faite suivant le sens entrée vers sortie.
- L'algorithme compare la sortie de l'algorithme avec la valeur cible selon une fonction d'optimisation et calcule une erreur de prédiction.
- L'erreur de prédiction est répercutée sur les neurones du réseau dans le sens sortie vers entrée. L'ajustement des poids des entrées d'un nœud va être fait selon la contribution de l'entrée à l'erreur, l'ajustement est fait selon l'algorithme de descente de gradient. Cette partie de répartition de l'erreur de prédiction s'appelle la *backpropagation*.

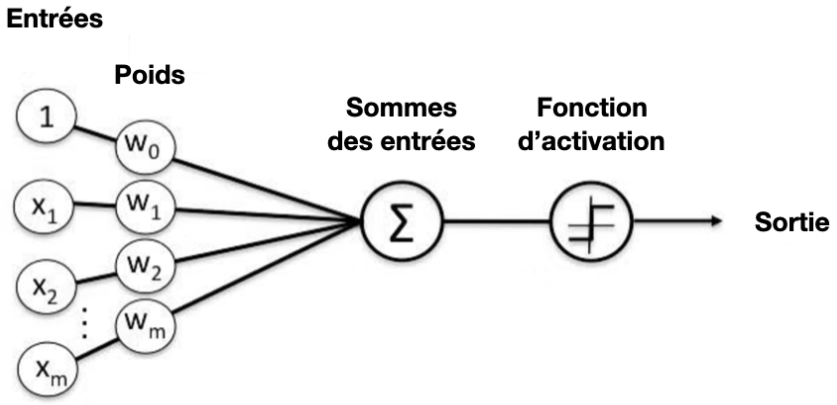


Figure 23 : Illustration d'un perceptron

II.c.2 Réseau de neurones récurrents

Jusqu'à présent nous avons surtout vu l'application d'algorithmes qui s'intéressent à établir soit un modèle de régression ou de classification. Bien qu'ils puissent être très performants dans leur domaine, ils peuvent rester impuissants quand il s'agit de prédire des données séquentielles ou des séries temporelles. Il faut pour cela ajouter à une nouvelle entrée la valeur de l'entrée précédente, il y a donc récurrence de l'information (Figure 24).

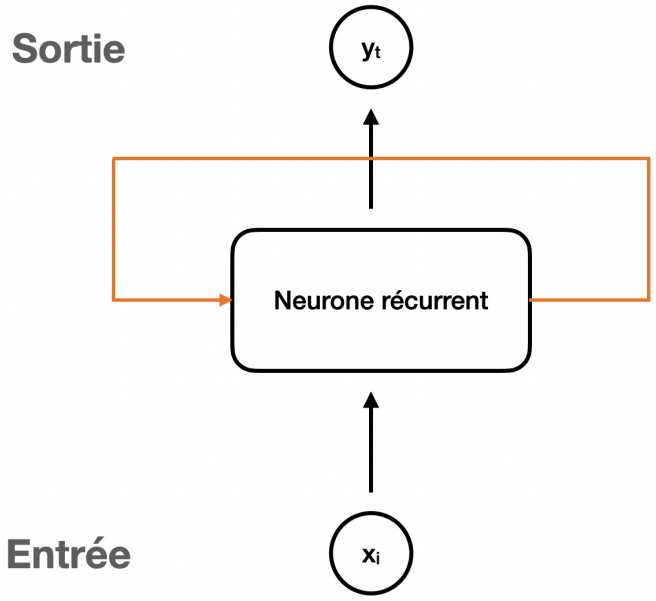


Figure 24 : Représentation d'un neurone récurrent

Dans le réseau, une entrée cachée h_i est donnée en argument à la prochaine prédiction. Pour une prédiction y_i l'algorithme utilise les entrées x_t et x_{t-1} (Figure 25).

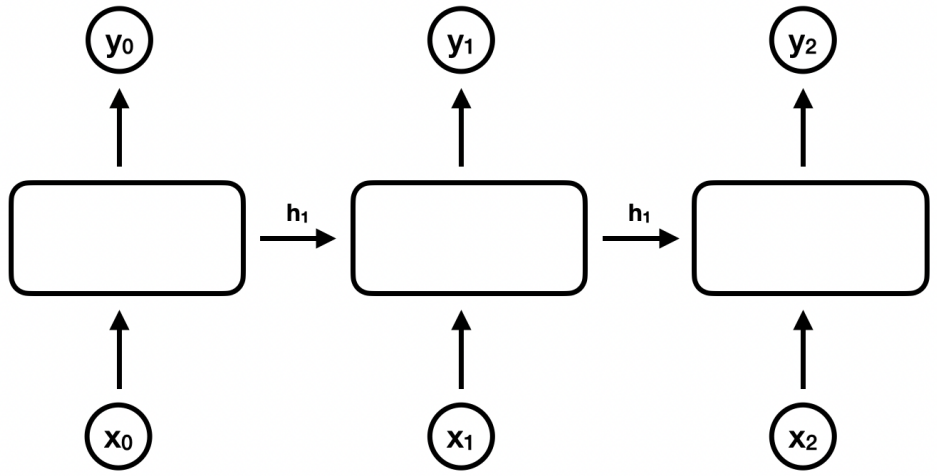


Figure 25 : Illustration d'un réseau de neurones récurrent

Ce type de réseau de neurone présente un inconvénient majeur, il a une mémoire à court terme, c'est-à-dire qu'il ne retient que l'état caché de la précédente entrée. Ainsi pour faire du traitement du langage naturel, il faut que l'algorithme puisse apprendre du contexte d'une phrase et donc de l'ensemble des mots. Il a besoin d'une mémoire plus longue. Cet inconvénient majeur a été résolu via le développement d'un modèle de réseaux de neurone à mémoire plus longue, le LSTM (Long Short Term Memory).

II.c.3 Réseau de neurones convolutif

En termes de classification d'image, le réseau de neurones convolutif s'est imposé comme un des meilleurs modèles de reconnaissance de représentations et de classification. Contrairement au réseau de neurones artificiels, le réseau de neurone convolutif est constitué de deux parties, une partie convolutive qui vise à extraire les caractéristiques d'une image en les compressant via le recours à des filtres appelés carte de convolution, et d'une partie de classification qui est faite d'une couche de neurones totalement connectée qui collecte les informations extraites par la partie convolutive et effectue un classement de l'image.

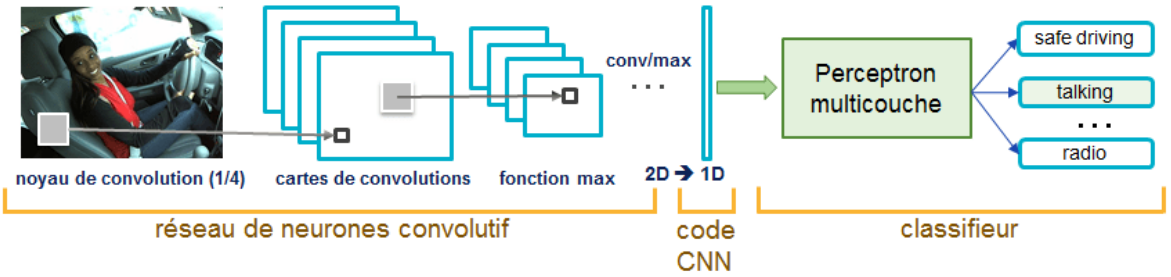


Figure 26 : Illustration d'un réseau de neurones convolutif.

De nombreux modèles de réseaux de neurones sont possibles et il est impossible de tous les présenter. On peut néanmoins citer les réseaux de neurones sur graphes qui voient leur popularité fortement augmenter notamment dans le domaine de prédiction d'effets indésirables [17], de la détection de fausses informations sur les réseaux sociaux [18] ou dans la conception *in silico* de molécules [19], les auto-encodeurs [20–22] ou encore les réseaux antagonistes génératifs utilisé pour renforcer la résistance des modèles de reconnaissance d'image [23] ou tout simplement en art [24].

III Intelligence artificielle dans une stratégie de drug discovery

Le processus de drug discovery est un processus long et coûteux, qui l'est de plus en plus avec le temps, Rennane *et al.* [25] ont estimé que le coût de développement d'un médicament oscillait entre 113 millions et 6 milliard de dollars [26] et le coût de développement de nouvelles entités moléculaires se situerait entre 318 millions et 2.8 milliard de dollars [27] alors qu'un repositionnement thérapeutique coûterait environ 41.3 millions de dollars [28]. De plus, c'est un processus risqué avec de fort risque qu'un candidat médicaments ne passe pas la phase clinique de son développement [29] Il est estimé qu'un seul candidat médicament sur dix passe les essais cliniques de phase II [30,31].

III.a Criblage virtuel

L'étape de criblage, ou screening, consiste en l'identification de composés biologiquement actifs sur des cibles déterminées. Quand cette technique allie robotique, bioinformatique et informatique pour étudier de grandes quantités de molécules sur des chimiothèques et ciblothèques, on parle alors de criblage haut débit ou screening haut débit. La possibilité de réaliser cette étape virtuellement est une occasion supplémentaire de réduire les coûts de développement en présélectionnant des composés qui ont plus de chances de passer ensuite l'étape de criblage.

Il existe deux types de criblage virtuel : le criblage virtuel basé sur la structure du ligand et le criblage virtuel basé sur la structure de la cible à atteindre.

III.a.1 D'après la structure du ligand

Le criblage virtuel basé sur la structure du ligand apprend de l'activité biologique et des propriétés physico-chimiques de ligands actifs et inactifs pour prédire l'activité inconnue de composés issus d'un ensemble plus vaste. Puisque cette technique n'utilise pas les données de structure de la cible thérapeutique, elle se révèle utile quand les informations sur la cible sont indisponibles ou quand elles sont de faible qualité [32]. Les méthodes développées pour effectuer du criblage virtuel basé sur la structure du ligand consistent le plus souvent en de multiples étapes et chaque étape utilise un algorithme particulier pour filtrer les composés retenus à l'étape précédente. Par exemple, un ensemble de composés peut être sélectionné pour son activité envers une cible, puis cet ensemble est filtré selon les propriétés pharmacocinétiques afin de sélectionner ceux qui ont le plus de chance de donner un candidat médicament viable.

Les modèles et logiciels de screening virtuel sont nombreux, le tableau 1 regroupe les algorithmes de criblage virtuel basé sur la structure du ligand.

Auteurs	Nom	Objectif	URL
Zoete et al. [33]	SwissSimilarity	Identifier des composés similaires à une molécule de référence selon la conformation 2D ou 3D.	http://www.swisssimilarity.ch
Imbernón et al. [34]	METADOCK	Optimisation des calculs d'interaction entre cible et ligand.	
Riniker et Landrum [35]	Open-Source	Platforme qui permet d'évaluer des empreintes digitales 2D pour le criblage.	
Li et al [36]	USR-VS	Comparer en deux secondes 93,9 millions de conformères 3D, et de présenter les 100 molécules les plus similaires selon leur forme 3D.	http://usr.marseille.inserm.fr
Suzuki et al. [37]	PKRank	Ordonner les molécules sélectionnées suite à un screening selon des données hétérogènes.	
Patel et al. [38]	PyGOLD	Implémentation en Python de l'outil de docking GOLD	
Banegas-Luna et al. [39]	BRUSELAS	Outil de recherche de similarité selon la conformation 3D et la présence de pharmacophores.	http://bio-hpc.eu/software/Bruselas/
Wang et al. [40]	RADER	Outil pour faciliter l'évaluation de la sélection virtuelle basée sur les leurres.	
Mochizuki et al. [41]	QEX	Une amélioration de la méthode d'estimation quantitative du caractère médicamenteux (QED).	
Zhang et al. [42]	IVS2vec	Un modèle qui allie une technologie de traitement du langage naturel (Word2vec) et un réseau de neurones profond pour classer un ensemble de cible à partir d'une molécule selon la forte ou la faible probabilité de fixation de cette molécule sur la cible.	https://github.com/haiping1010/IVS2Vec

Arcon et al. [43]	AutoDockBias	Classer par ordre de priorité les ligands probables à une cible.	https://autodockbias.wordpress.com
Ebejet et al. [44]	Ligity	Une méthode hybride basée sur la structure des ligands pour le criblage virtuel de grandes bases de données de petites molécules.	
Zhu et al. [45]	D3Similarity	Une méthode de screening de composés contre le SARS-Cov-2GCAC.	http://ccbb.jnu.ac.in/gcac
Liu et al. [46]	DeepScreening	Serveur web qui utilise des données publiques ou fournies par l'utilisateur pour effectuer un criblage virtuel des sondes chimiques ou des médicaments pour une cible d'intérêt spécifique.	http://deepscreening.xielab.net
Soufan et al. [47]	DpubChem	Outil en ligne qui met en œuvre les techniques d'apprentissage automatique afin d'améliorer la précision des modèles et de permettre des analyses efficaces des expériences de la base de données PubChem BioAssay	https://www.cbrc.kaust.edu.sa/dpubchem/

Tableau 1 : Algorithmes de criblage virtuel basé sur la structure du ligand.

Ces méthodes de screening virtuel ont été employées pour identifier des molécules ayant, potentiellement, une activité thérapeutique, on peut noter l'étude de *Zhang et al.* dans laquelle un classifieur binaire, utilisant un réseau de neurone convolutif et l'algorithme de machines à vecteur de support, a permis l'identification de 174 composés aux potentielles propriétés antimalariques. Les expériences de validation ont confirmé l'activité antimalarique de 25 de ces composés dont le meilleur avait une concentration efficace médiane (EC_{50}) de 95.6 nM [48]. L'implémentation de ces algorithmes a une réelle place dans la recherche de nouveaux traitements comme dans l'identification de composés efficaces contre l'infection au SARS-Cov-2 [49–51], la recherche d'inhibiteurs d'Aurora kinase A, d'inhibiteurs de PI3Kα, d'inhibiteurs sélectifs d'histone déacétylase 8 ou d'inhibiteurs de p-hydroxyphenylpuruvate dioxygenase [6].

III.a.2 D'après la structure de la cible

Le criblage virtuel basé sur la structure de la cible utilise les informations de structure 3D de protéines cibles afin de prédire les interactions entre le ligand et sa cible et d'identifier les résidus engagés dans ces interactions protéine-ligand. Le docking ou amarrage moléculaire est la principale technique employée pour étudier ces interactions. De nombreuses méthodes d'intelligence artificielle et de simulation ont été développées pour prédire et étudier l'affinité ligand-cible dans cette méthode de criblage virtuel [6,52,53] (Tableau 2).

Auteurs	Nom	Objectif	URL
Labbé et al. [54]	MTiOpenScreen	Un outil en ligne dédié au docking et au criblage virtuel de petites molécules.	https://bioserv.rpbs.univ-paris-diderot.fr/services/MTiOpenScreen/
Schellhammer et Rarey [55]	FlexX-Scan	Un outil pour le criblage virtuel à haut débit basé sur la structure prenant en compte les points d'interaction protéique favorables dans le site de liaison de la protéine.	
Perez-Castillo et al. [56]	CompScore	Une méthode qui utilise la notation par consensus pour améliorer les performances de notation.	https://bioquimio.udla.edu.ec/compscore/
Skalic et al. [57]	PlayMolecule BindScopre	Un ensemble d'outils servant le processus de drug discovery dont un outil de docking automatisé et de classification des ligand selon leur affinité avec une cible.	https://www.playmolecule.com
Fang et al. [58]	GeauxDock	Un programme de docking moléculaire basé sur l'algorithme de Monte Carlo et présente une nouvelle fonction de notation combinant des termes d'énergie basés sur la physique avec des potentiels statistiques et basés sur la connaissance.	http://www.brylinski.org/geauxdock
Pires et al. [59]	Easy VS	Un outil qui permet, à partir du code PDB d'une protéine, de cribler un ensemble de base de données publique et de filtrer les molécules retenues selon leur propriétés physico-chimiques, de les classer selon leur similarité avec une personnalisation des paramètres de docking	http://biosig.unimelb.edu.au/easyvs/
Ibrahim et al. [60]	DEKOIS 2.0	Une base de données visant à étendre et à compléter la collection d'ensembles de leurres disponibles publiquement.	http://www.pharmchem.uni-tuebingen.de/dekois/
Shin et al. [61]	PL-PatchSurfer 2	Une méthode de criblage virtuel utilisant les surfaces des protéines et des ligands en les représentant par un ensemble de patchs locaux qui se chevauchent.	https://kiharalab.org/plps2/

Litfin et al. [62]	SPOT-ligand 2	Une méthode de criblage qui prend en compte l'homologie de liaison, qui tire parti des paires de liaisons protéine-ligand connues.	http://sparks-lab.org
Ropp et al. [63]	Gypsum-DL	Une méthode qui se concentre sur la préparation des molécules préalablement au docking.	https://durrantlab.pitt.edu/gypsum-dl/
Akbar et al. [64]	ENRI	Une méthode qui utilise un classifieur binaire pour sélectionner les conformations de récepteurs pouvant faire l'objet de médicaments.	
Wallach et al. [65]	AtomNet	Un outil qui incorpore un réseau neuronal convolutif profond basé sur la structure, conçu pour prédire la bioactivité de petites molécules pour des applications de découverte de médicaments.	
Bao et al. [66]	DeepBSP	Une méthode qui peut prédire directement l'écart quadratique moyen (rmsd) de la position d'arrimage d'un ligand par rapport à sa position de liaison native.	
Adeshina et al. [67]	vScreenML	Un outil de criblage qui utilise la méthode XGBoost.	
Stepniewska-Dziubinska et al. [68]	Pafnucy	Un outil qui comprend un réseau neuronal profond permettant d'estimer l'affinité de liaison des complexes ligand-récepteur.	https://gitlab.com/cheminfIBB/pafnucy
Zheng et al. [69]	Onion-Net	Un outil qui utilise un réseau 3D de neurone convolutif pour la intégrer les informations de la relation protéine-ligand.	
Ballester et Mitchell [70]	RFScore-v3	Un algorithme de notation de score d'affinité qui utilise une forêt d'arbres de décision.	
Ashtawy et Mahapatra [71]	BgN(BsN)-Score	Un algorithme de notation qui utilise la technique de Bagging et d'Ensemble Learning.	
Durrant et McCammon [72]	NNscore2.0	Un algorithme de notation qui utilise un réseau neuronal.	
Wang et Zhang [73]	Δ_{vina}RF	Un algorithme de notation d'affinité qui utilisé un algorithme de forêt d'arbre de décision pour ajuster les paramètres du modèle et la sélection de variables.	

Tableau 2 : Algorithmes de criblage virtuel basé sur la structure du ligand.

Ces algorithmes jouent un rôle dans le screening comme la prédiction d'activité d'inhibiteurs d'HIF-1, d'inhibiteur de BRD4 ou d'agonistes des récepteurs β2 adrénergiques [6]. En cancérologie ces algorithmes se révèlent efficaces par exemple par la découverte d'un nouveau traitement pour la leucémie aiguë myéloïde dirigée contre la kinase GSK-3β [74], l'identification d'un inhibiteur de PRMT5 dans le cancer du poumon non à petites cellules, d'inhibiteurs de VEGFR2 dans le carcinome rénal et d'inhibiteurs de MDM2-p53 [6].

III.b Prédiction des propriétés physicochimiques

Parmi les paramètres qui affectent fortement les propriétés pharmacocinétiques d'un candidat médicament se trouvent les propriétés physicochimiques telles que la solubilité, le coefficient de partage, le degré d'ionisation ou le coefficient de perméabilité. Il est donc normal de prendre en compte ces paramètres dans le processus d'optimisation thérapeutique et de les prédire. Pour cela des techniques d'intelligence artificielle ont été développées pour estimer les paramètres physicochimiques à partir de la structure moléculaire (description symbolique de la structure chimique sous forme de chaîne de caractères), des mesures d'énergie potentielle ou des estimations de densité électronique autour des atomes par exemple [75–77]. De nombreuses techniques de machine learning sont utilisées telles que l'algorithme des k-plus proches voisins, l'analyse discriminante linéaire, la régression des moindres carrés partiels, la forêt d'arbres décisionnels ou les machines à vecteurs de support [78]. Une étude comparative de six algorithmes développés pour prédire l'absorption intestinale de composés a montré que le modèle reposant sur des machines à supports de vecteurs obtient la plus haute précision avec 91.54% [79]. Différentes applications en ligne sont également disponibles comme ALGOPS (http://www.vcclab.org/lab/alogps/) qui permet une prédiction en ligne du logP, de la solubilité en milieu aqueux et du pKa de composés [80], ASNN (http://www.vcclab.org/lab/asnn/) [81], E-BABEL (http://www.vcclab.org/lab/babel/) qui est une boîte à outils qui vise à mettre en place une plateforme qui est capable de prendre en charge des données chimiques hétérogènes et de les inscrire dans un même cadre d'analyse, E-DRAGON (http://www.vcclab.org/lab/edragon/) [81] qui

permet le calcul de descripteurs moléculaires, ChemSpider (http://www.chemspider.com) [82], SPARC (http://archemcalc.com/sparc-web/calc) [83] et OSIRIS (https://www.organic-chemistry.org/prog/peo/) qui permet de prédire les propriétés physicochimiques ou de toxicité de molécules dessinées [84]. En 2016, Puratchikody *et al.* ont utilisé OSIRIS pour prédire la toxicité structurale de composés anti-inflammatoires. Parmi 55 molécules, seules 19 molécules furent considérées comme de potentiels inhibiteurs de cyclooxygénase-2 (COX-2) [6] En 2019, une équipe de Sanofi-Aventis AG a également réussi à mettre en œuvre un modèle d'apprentissage profond multitâche prédictif dans leurs flux de travail ADMET *in silico* [85]. En appliquant une autre méthode d'apprentissage multitâche pour transférer les caractéristiques entre les ensembles de données, ils ont développé des modèles multitâches pour prédire la clairance, la perméabilité Caco-2 et le $logD_{7.4}$. Ils ont également montré que les méthodes d'apprentissage profond multitâches, bien que surpassant les méthodes d'apprentissage profond monotâches dans de nombreux cas, présentaient des performances inférieures à celles des méthodes monotâches pour certains paramètres.

III.c Prédiction de la bioactivité

Des applications en ligne ont été développés pour mettre à disposition des outils spécifiques (Tableau 3)

Auteurs	Nom	Objectif	URL
Nascimento et al. [86]	KronRLS	Outil qui prédit l'affinité molécule-cible en utilisant les propriétés de la cible et les similarités moléculaires entre la cible et le ligand.	https://bio.tools/KronRLS-MKL
He et al. [87]	SimBoost	Une méthode qui prédit les valeurs continues des affinités de liaison des composés et des protéines et qui incorpore ainsi tout le spectre d'interaction.	
Öztürk et al. [88]	DeepDTA	Un modèle basé sur l'apprentissage profond qui utilise uniquement les informations de séquence des cibles et des médicaments pour prédire les affinités de liaison de l'interaction médicament-cible.	https://github.com/hkmztrk/DeepDTA
Feng et al. [89]	PADME	Un outil qui utilise une technologie de graphe moléculaire convolutif pour prédire les propriétés de bioactivité.	
Matlock et al. [90]	XenoSite	Un outil permettant de visualiser la probabilité que chaque site atomique soit un site de métabolisme pour une variété de cytochromes P450 importants.	https://xenosite.org
Šícho et al. [91]	FAME 3	Cet algorithme prédit les sites de métabolisation de petites molécules à partir d'arbres de décision.	https://nerdd.univie.ac.at
Beck et al. [92]	SMARTCyp	Outil de prédiction du métabolisme par les cytochromes P450	https://smartcyp.sund.ku.dk/mol_to_som

Tableau 3 : Outils et algorithmes visant à prédire la bioactivité de composés.

Beck *et al.* ont utilisé l'outil DeepDTA pour prédire l'activité antivirale de composés commercialisés contre le SARS-Cov-2. Les résultats ont montré que l'atazanavir, le remdesivir et l'association lopinavir-ritonavir, la rapamycine et le tiotropium pourraient être sélectionnés comme candidats médicaments contre l'infection à SARS-Cov-2 [92].

III.d Prédiction de la toxicité et du mécanisme d'action

La toxicité d'un composé est étroitement liée à son mode d'action et à son métabolisme, et sa connaissance ou prédiction est nécessaire pour éviter tout effet indésirable trop important qui compromettrait sa mise sur le marché. Dans le cycle de développement du médicament, les essais de toxicité commencent par les essais précliniques sur des modèles cellulaires et animaux. Ces essais étant coûteux, ils augmentent, de ce fait, le coût de développement. Les avancées dans le domaine de la prédiction de la toxicité pour réduire ces coûts ont amené au développement de plusieurs applications en ligne et logiciels de prédiction (Tableau 4).

Auteurs	Nom	Objectif	URL
Cañada et al. [93]	LimTox	Par une approche de fouille de textes, tente d'extraire des associations entre des composés et un effet toxicologique particulier d'après le contenu des rapports toxicologiques	https://limtox.bioinfo.cnio.es/
Pires et al. [94]	pkCSM	Un outil qui propose de repérer des toxiphores au sein d'une molécule.	https://biosig.lab.uq.edu.au/pkcsm/
Yang et al. [95]	admetSAR2.0	Un service en ligne qui permet de prédire les paramètres ADME de composés et incorporant un module d'évaluation de risque environnemental.	
Patlewicz et al. [96]	Toxtree	Un outil capable d'estimer le risque toxique en appliquant une approche par arbre de décision.	https://toxtree.sourceforge.net/
Wang et al. [97]	SEA	Un outil qui utilise une approche par ensemble de similarité chimique.	https://sea.bkslab.org/
Pu et al. [98]	eToxPred	Un algorithme qui utilise un modèle Deep Belief Network (DBN) et un modèle de forêt d'arbres décisionnels pour évaluer la toxicité de composés à partir de leur empreinte moléculaire.	https://www.brylinski.org/etoxpred-0
Lysenko et al. [99]	TargetTox	Un outil qui exploite les données des cibles de molécules, des résultats d'analyse par ontologie de gènes et par réseaux biologiques pour prédire leur toxicité.	https://github.com/artem-lysenko/TargeTox
Mayr et al. [100]	DeepTox	Outil qui utilise un algorithme de réseau de neurones convolutifs pour prédire la toxicité de nouveaux composés.	http://www.bioinf.jku.at/research/DeepTox/
Gayvert et al. [101]	PrOCTOR	Cet outil calcule un score de probabilité de toxicité d'un composé dans les essais cliniques à partir des propriétés des cibles du composé et du composé lui-même, des données d'expression des tissus cibles et des données génomiques à partir d'un algorithme de forêt d'arbres décisionnels.	https://github.com/kgayvert/PrOCTOR

Jimenez-Carretero et al. [102]	Tox_(R)CNN	Cet algorithme consiste à prédire et d'évaluer la cytotoxicité de molécules en se basant sur des images de cellules colorées au DAPI	
Goh et al. [103]	SMILES2vec	Un outil qui permet de prédire les propriétés chimiques et la toxicité d'un composé à partir de seulement son format SMILES	
Goh et al. [104]	Chemception	Cet outil prédit les propriétés chimiques et la toxicité à partir du dessin 2D d'une molécule.	
Preuer et al. [105]	DeepSynergy	Il cherche à prédire les activités anticancéreuses de composés.	
Xu et al. [106]	deepAOT	Cet outil se concentre sur la prédiction de toxicité orale des composés.	

Tableau 4 : Outils et algorithmes de prédiction de toxicité et de mécanisme d'action.

Ces modèles peuvent être employés pour prédire des toxicités de composés dans le cadre d'une prise en charge thérapeutique comme l'ont fait Srivastava *et al.* en 2020 par l'utilisation d'admetSAR pour évaluer la toxicité de *Withania somnifera* dans le traitement de la COVID-19 [107]. De même Zhang *et al.* ont développé des modèles prédictifs de toxicité hépatique médicamenteuse basés sur les empreintes MACCS et FP4 des molécules. Les résultats montrent une précision globale de 75% [108].

III.e Repositionnement thérapeutique

Le repositionnement thérapeutique consiste en l'étude de molécules déjà commercialisées pour une indication afin de les repositionner sur une autre indication thérapeutique. L'émergence de bases de données hétérogènes de génomique, protéomique et de sensibilité pharmacologique permet de trouver de nouvelles opportunités de cibles thérapeutiques, souvent parce qu'une molécule peut cibler d'autres cibles que celle pour laquelle elle a été développé, et par conséquent, le recours à des méthodes de biologie systémique permet d'évaluer de tels liens (Tableau 5).

Auteurs	Nom	Objectif	URL
Martinez et al. [109]	DrugNet	Cet outil passe par une apparoche en réseau de large échelle connectant des informations pharmacologique, protéique et sur les pathologies.	http://genome.ugr.es:9000/drugnet
Zhang et al. [110]	DRIMC	Un algorithme de repositionnement thérapeutique qui utilise la complétion de matrice inductive bayésienne.	
Luo et al. [111]	DPDR-CPI	Cet outil fait de la prédiction en temps réel basée sur la structure moléculaire et l'interactome composés-protéines.	http://cpi.bio-x.cn/dpdr/
Whirl-Carrillo [112,113]	PHARMGKB	Outil de pharmacogénomique qui compile des informations entre les variations génomiques et la réponse médicamenteuse.	https://www.pharmgkb.org
Gallo et al. [114]	PROMISCUOUS 2.0	Base de données qui fournit une liste prétraitée de candidats médicaments pour des indications thérapeutiques données	https://bioinf-applied.charite.de/promiscuous2/index.php
Luo et al. [115]	DRRS	Un système de recommandation de positionnement thérapeutique par une approche en réseau.	
Yella et al. [116]	MGATRx	Un outil de repositionnement thérapetique qui utilise une approche par réseau de neurones en graphe.	https://github.com/yellajaswanth/MGATRx
Yan et al. [117]	BiRWDDA	Un outil qui fusionne plusieurs mesures de similarité et une marche aléatoire pour découvrir des associations potentielles entre les médicaments et les maladies.	
Fahimian et al. [118]	RepCOOL	Un outil qui utilise une approche par réseau biologique.	
Zheng et al. [119]	DTI-RCNN	Cet outil basé sur une architecture de réseau de neurones profond et de LSTM vise à prédire les interactions molécules-cibles.	

Jarada et al. [120]	SNF-CVAE	Un outil pour prédire les nouvelles interactions médicament-maladie à l'aide d'informations sur la similarité entre les médicaments et les interactions médicament-maladie connues.	
Xu et Wang [121]	PhenoPredict	Un outil spécifique à la schizophrénie qui pour objectif d'effectuer un repositionnement à partir du phénome de pathologies proches	
Wu et al. [122]	SDTNBI	SDTNBI incorpore la chimio-informatique et les réseaux pour combler le fossé entre les nouvelles entités chimiques et le réseau DTI connu pour faire de l'inférence basée sur le réseau substructure-médicament-cible.	
Chen et al. [123]	iDrug	Un outil qui intègre le repositionnement des médicaments et la prédiction de leur cible dans un modèle cohérent via l'intégration de réseaux croisés.	https://github.com/Case-esaC/iDrug
Wang et al. [124]	BiFusion	Un algorithme de réseaux de neurones convolutif sur graphes bipartite qui permet l'intégration de données hétérogènes afin de construire un graph de relation médicament-protéine, pathologie-protéine et d'interactions protéine-protéine	
Zeng et al. [125]	deepDTnet	Une méthode par apprentissage profond qui fait du repositionnement thérapeutique à partir d'un réseau hétérogène regroupant 15 types de données tels que les informations chimiques, moléculaires, pharmacologiques, génomiques, phénotypique et cellulaires	https://github.com/ChengF-Lab/deepDTnet
Liu et al. [126]	HNet-DNN	Un outil qui utilise un réseau neuronal profond (DNN), pour prédire les nouvelles associations médicament-maladie sur la base des caractéristiques extraites du réseau hétérogène médicament-maladie.	https://github.com/hliu2016/HNet-DNN

Jiang et al. [127]	SKCNN	Un outil combinant les techniques du noyau sigmoïde et du réseau neuronal convolutif, qui est capable d'apprendre de nouvelles caractéristiques représentant efficacement les associations médicament-maladie via ses couches cachées.	https://github.com/HanJingJiang/SKCNN
Mazumdar et al. [128]	CBPred	Une méthode basée sur un réseau neuronal convolutif (CNN) et une mémoire bidirectionnelle à long terme (BiLSTM) pour prédire les maladies liées aux médicaments.	https://github.com/csl-iisc/dpPred-cbPred
Jiang et al. [129]	SAEROF	Un modèle de calcul combinant un auto-encodeur clairsemé et une forêt de rotation (SAEROF) pour prédire l'association médicament-maladie.	https://github.com/HanJingJiang/SAEROF
Zeng et al. [130]	DeepDR	Une approche de repositionnement thérapeutique par apprentissage profond qui incorpore 10 réseaux : un réseau médicament-maladie, un réseau médicament-effet secondaire, un réseau médicament-cible et sept réseaux médicament-médicament.	https://github.com/ChengF-Lab/deepDR
Huang et al. [131]	SkipGNN	Une approche de réseau neuronal graphique pour la prédiction des interactions moléculaires	https://github.com/kexinhuang12345/SkipGNN

Tableau 5 : Algorithmes et outils de repositionnement thérapeutiques

Ces algorithmes permettent la découverte de potentiels nouvelles indications là l'image de SNF-CVAE qui, lors de son développement, a correctement prédit de nouvelles molécules en cours d'investigation contre la maladie d'Alzheimer et l'arthrite rhumatoïde juvénile, PhenoPredict qui fait du repositionnement thérapeutique sur la schizophrénie à partir du phénome de pathologies proches, SDTNBI qui propose en preuve de concept un possible repositionnement thérapeutique pour des anti-inflammatoires non stéroïdiens en tant qu'anticancéreux par inhibition d'AKR1C3, C19 ou CA12 et DeepDR a correctement prédit des repositionnement thérapeutiques dans la maladie d'Alzheimer et de Parkinson.

III.f Prédiction protéique

La connaissance de la structure protéique d'une potentielle cible thérapeutique est essentielle à la compréhension de la physiopathologie d'une maladie, du mécanisme d'action d'un médicament et de la détection d'éventuels effets indésirables. Malgré des avancées technologiques dans les techniques d'imagerie par rayons X, par spectroscopie par résonance magnétique nucléaire ou par cristallographie, seuls 35% du protéome humain a été couvert par ces analyses, parfois uniquement sur des portions de domaines protéiques [132]. Une alternative à ces approches expérimentales consiste en l'utilisation d'algorithmes d'intelligence artificielle pour prédire la conformation 2D et 3D des protéines et ainsi prédire les effets qu'aurait une molécule sur une cible bien avant de l'avoir synthétisée ou produite. *Parmi ces technologies, la plus connue est le programme AlphaFold2 développé par DeepMind* [133]*,* entraînée sur 170 000 protéines issues de la base de données Protein Data Bank (PDB), elle utilise les informations d'analyse d'alignement de séquence et calcule les distances entre les paires d'acides aminés pour former la protéine [14]. AlphaFold2 a des capacités de prédiction comparables aux méthodes expérimentales néanmoins la prédiction de la structure 3D sera plus difficile sur des protéines multi-domaines ou des complexes protéiques multimériques. Néanmoins, il est certain que l'intelligence artificielle a une place de choix dans ce domaine. Par ailleurs, AlQurashi a développé une méthode reposant sur un réseau récurrent géométrique pour prédire la structure des protéines plus rapidement qu'AlphaFold2 mais avec une plus faible précision qu'AlphaFold2 [134]. De même, Avdagic *et al.* ont développé un modèle de prédiction de structure 2D de

séquences protéiques en utilisant la base de données CB513 avec une précision de 62.72% [135].

En dehors de la prédiction des structures secondaires et tertiaires des protéines, la prédiction d'interactions protéine-protéine est essentielle pour essayer de comprendre la fonction d'une protéine et son implication dans les processus biologiques. Les études d'interactions protéine-protéine font appel à des approches en réseaux, des études d'ontologie de gènes où l'objectif est de détecter si parmi un ensemble de gènes, des processus biologiques sont sur-représentés ou encore des études de co-expression de gènes. Plusieurs études ont utilisé des réseaux bayésiens pour prédire des interactions protéines-protéines ainsi qu'une approche par analyse en composantes principales [6]. Une approche par apprentissage profond, DeepPPI (https://github.com/gdario/deep_ppi) a réussi à avoir une précision de 92,5%, une sensibilité de 90,56% et une spécificité de 94,49% [136]. Les méthodes de prédiction des site de fixation protéique sont également nombreuses comme HSMM [137], PPI_SVM [138], MULTIPROSPECTOR [139] ou finalement Struct2Net (https://cb.csail.mit.edu/cb/struct2net/webserver/) qui prédit les interactions protéine-protéine à partir de séquences protéiques [140].

III.g Prédiction de voies métaboliques et polypharmacologie

Que ce soit pour la découverte de potentielles nouvelles cibles thérapeutiques par l'identification de voies métaboliques spécifiques impliquées dans les pathologies ou la conception de molécules pouvant interagir avec plusieurs cibles afin d'apporter leur effet sur l'ensemble d'un réseau d'interactions, une molécule à cibles multiples serait plus efficace qu'une molécule à cible unique pour les pathologies complexes comme les cancers ou les pathologies métaboliques [141,142]. L'intelligence artificielle et le machine learning ont produit un panel de méthodes pour répondre à ces besoins dont *Polypharmacology browser* (PPB) (http://gdbtools.unibe.ch:8080/PPB/) [143], TarPred [144], SPiDER [145], TargetHunter [146], PharmMapper (http://www.lilab-ecust.cn/pharmmapper/) [147], ChemMapper [148], SwissTargetPrediction (http://www.swisstargetprediction.ch) [149], KinomeX [150] et Macau [151]. L'outil *Polypharmacology browser* fut utilisé pour identifier des cibles thérapeutiques ciblant l'angiogenèse et des inhibiteurs sélectifs du canal ionique TRPM4 [152,153]. PharmMapper et ChemMapper furent utilisés pour identifier des ligands multicibles dans la maladie d'Alzheimer,

comprendre le mécanisme de synergie entre l'astragale et le coptide chinois dans le diabète de type I, comprendre aussi le mécanisme d'élévation de la glycémie de la décoction *Danggui Buxue Tang*, un remède issu de la médecine chinoise ou encore l'identification de molécules bloquant le trafic du cuivre dans le cancer [6].

L'outil OpenTargets (https://www.opentargets.org) [154] est un outil de machine learning qui offre la possibilité d'identifier des candidats en fonction des voies métaboliques activées. Récemment, l'identification de voies métaboliques activées a permis de sélectionner des molécules efficaces contre des protéines impliquées dans ces voies métaboliques [6].

III.h Design *de novo*

Avec l'accumulation de données biomédicales, l'intelligence artificielle, à travers sa capacité à intégrer des dépendances et des relations complexes au sein des données, offre de nouvelles possibilités dans l'élaboration de nouveaux composés chimiques. On peut noter le développement de GENTRL, un algorithme d'apprentissage par renforcement pour la génération *de novo* de petites molécules (https://github.com/insilicomedicine/GENTRL) [155,156] qui permit la découverte d'un inhibiteur de DDR1 en seulement 21 jours, ainsi que DEL (DNA-Encoded Libraries) [157], Synthia qui est capable de proposer des voies de synthèses [78], Putin *et al.* ont développé un outil entraîné à partir de molécules représentées sous leur forme SMILES qui génère des molécules avec des caractéristiques physicochimiques prédéfinies (logP, masse molaire et surface polaire topologique) [158]. Xiong *et al.* font état de nombreuses méthodes utilisant les réseaux de neurones profonds appliqués sur graphes pour générer automatiquement de nouvelles molécules [159]. Ces techniques furent employées avec succès pour, par exemple, concevoir des modulateurs du récepteur PPAR et RXR [132].

IV Conclusion

La recherche de nouvelles substances thérapeutiques est un domaine de recherche à part entière qui concentre ses attentes sur des molécules promises à un succès thérapeutique. L'informatisation et l'automatisation progressive des domaines de recherche a généré une masse de données dont le volume, la variété, la valeur et la vélocité font de leur traitement un véritable défi à relever.

Les capacités surhumaines de l'informatique et l'habileté de l'intelligence artificielle à discerner les relations latentes au sein de données hétérogènes apportent au processus de découverte du médicament des opportunités inédites à toutes ses étapes. Nous avons vu que lors de l'étape de criblage, l'intelligence artificielle permet de raffiner les molécules pour ne sélectionner que celles qui auront le plus de chance d'être actives. L'intelligence artificielle est aussi capable de prédire les propriétés physico-chimiques des composés ainsi que leur potentielle toxicité au regard de l'état actuel des connaissances. Les techniques de machine learning peuvent aussi détecter de nouvelles opportunités pour des molécules déjà commercialisées, diminuant ainsi le coût de la recherche pour un nouveau traitement mais aussi le coût de la prise en charge du patient. L'intelligence artificielle offre la capacité de découvrir les mécanismes mis en œuvre dans une pathologie afin de pouvoir, plus tard, déterminer les cibles thérapeutiques les plus prometteuses.

Finalement, il est important de rappeler que l'intelligence artificielle est un outil remarquable qu'il faut appliquer avec raison et rigueur car les résultats dépendent fortement de la qualité des données qui lui sont fournies et c'est pour cela que l'intelligence artificielle ne remplacera jamais les techniques traditionnelles mais devra travailler en synergie avec elles pour assurer des résultats toujours plus proches de la réalité biologique.

Partie 2 : Exemple d'étude

I Introduction

Le cancer du poumon est la première cause de mortalité par cancer dans le monde avec 18% des morts estimées soit environ 1,8 millions de décès, c'est aussi le second cancer le plus diagnostiqué avec 11,4% de nouveaux cas estimés en 2020 soit environ 2,2 millions de nouveaux cas [160]. Le taux de survie est assez faible, il est estimé à 20% pour le taux de survie à 5 ans. Les thérapies ciblées sont en train de révolutionner le traitement de nombreux cancers dont le cancer du poumon avec en conséquence une augmentation significative de l'espérance de vie et de qualité de vie face à la maladie. L'efficacité des thérapies ciblées est notamment dépendante d'une sélection fine des patients susceptibles de répondre à la thérapie. Ainsi, les thérapies ciblées antitumorales représentent l'un des exemples les plus représentatifs de la médecine de précision. Cependant, la sélection des patients, basée actuellement sur la présence d'altérations génomiques simples, est imparfaite ce qui conduit soit à la sélection de patients non-répondeurs, soit à un défaut de sélection de patients susceptibles de répondre. Des avancées récentes sur le ciblage du récepteur tyrosine kinase MET sont représentatives de l'espoir et des difficultés rencontrés dans le traitement des cancers par les thérapies ciblées. En effet, près de 3% des cancers du poumon non à petites cellules présentent des mutations de MET, qui ont pour conséquences la délétion d'un domaine régulateur suite au saut de l'exon 14 (METex14). Ainsi, suite à la perte de mécanismes d'atténuation du signal, le récepteur METex14 induit en réponse à son ligand des réponses biologiques non régulées, pouvant conduire à une addiction oncogénique. Ainsi la recherche de nouvelles molécules actives contre le récepteur MET représente des opportunités thérapeutiques à découvrir et des chances de traitements efficaces dans le cancer du poumon.

Cette partie expérimentale a pour objet de présenter un début de pipeline de criblage virtuel utilisant uniquement les données issues de bases de données publiquement disponibles, des algorithmes de prédictions de paramètres d'activité biologique et de toxicité.

II Matériels et méthodes

II.a Données

Les données des molécules retenues sont issues des bases de données ChEMBL via les services web [161,162], et PubChem [163].

La liste des caractéristiques structurales indésirables proviennent d'une étude de Brenk *et al.* qui ont dressé une liste de sous-structures défavorables afin de filtrer leurs bibliothèques de composés utilisées pour les criblages [164].

Baell *et al.* se sont, quant à eux, concentrés sur les sous-structures interférant dans la signalisation des tests, nommés Pan Assay Interference Compounds (PAINS). Ils ont décrit les sous-structures qui peuvent aider à identifier ces PAINS et ont fourni une liste qui peut être utilisée pour le filtrage des sous-structures [165].

II.b Analyses

La première étape consiste à récupérer l'ensemble des composés actifs sur le récepteur MET à partir de la base de donnée ChEMBL ainsi que leurs caractéristiques physico-chimiques et leur pIC50. Parmi ces composés, nous ne garderons que les molécules répondants aux règles de Lipinski (poids moléculaire inférieur à 500 g/mol, nombre d'atomes donneurs de liaison hydrogène inférieur à 5, nombre d'atomes accepteurs de liaison hydrogène inférieur à 10 et un logP inférieur à 5), ainsi que celles dépourvus de motifs structuraux présentant des caractéristiques indésirables telles que les groupes nitro (mutagènes), les sulfates et les phosphates (entraînant probablement des propriétés pharmacocinétiques défavorables), les 2-halopyridines et les thiols (réactifs) et celles ne comportant pas de sous-structure susceptibles de donner des liaisons non spécifiques [165]. Ces analyses furent réalisées avec le module python RDKit (https://rdkit.org).

Afin d'augmenter le nombre de molécules à cribler, nous recherchons des molécules similaires à ce sous-ensemble en interrogeant la base de données PubChem et son web service REST PUG. Les molécules ainsi obtenues sont filtrées selon les mêmes critères que le premier ensemble (caractéristiques physico-chimiques, sous-structures indésirables et PAINS).

Nous prédisons la pIC50 des nouvelles molécules obtenues en entraînant un modèle de deep learning de réseau neuronal à deux couches cachées de 64 et 32 neurones avec la fonction ReLu comme fonction d'activation sur le premier ensemble

de données à partir de la signature moléculaire en utilisant le module python Scikit Learn [166]. Ces molécules sont finalement filtrées d'après leurs valeurs prédites de pIC50 et seules les molécules dont la pIC50 est supérieure à 9 sont conservées.

Finalement la prédiction de toxicité de l'ensemble des composés retenus (ensemble initial et molécules similaires à cet ensemble) fut effectuée avec l'algorithme eToxPred. Selon Pu *et al.*, une valeur de prédiction de toxicité de 0.58 discrimine le mieux les composés les plus toxiques des moins toxiques. Par conséquent, ne seront conservés que les composés ayant une valeur de prédiction de toxicité inférieure à 0.58 [98].

L'ensemble des programmes d'analyses ont été écrits en Python version 3.7.15.

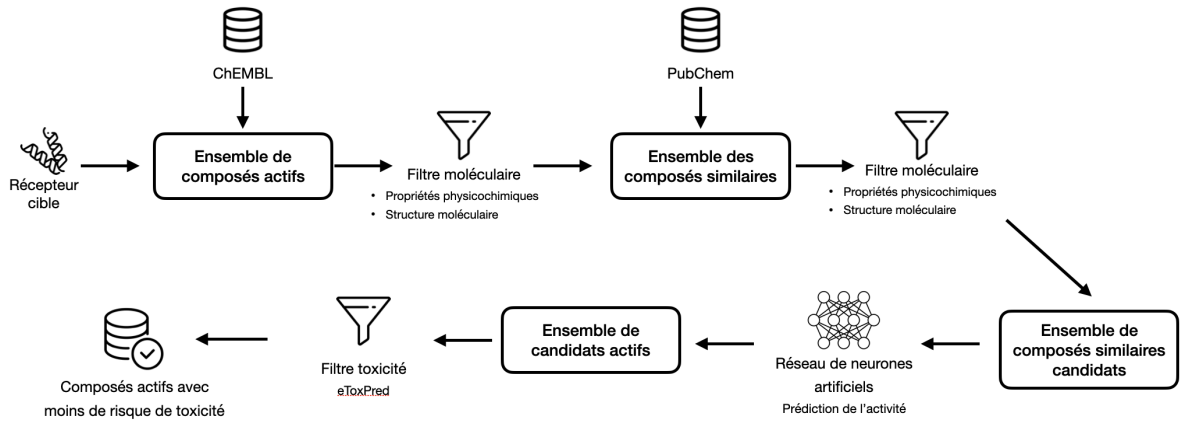


Figure 27 : Schéma de la démarche adoptée

III Résultats

III.a Molécules actives sur le récepteur MET

A l'issue de la recherche de molécules actives sur le récepteur MET, 4206 essais d'IC50 furent récupérés pour 3204 molécules avec des pIC50 allant de 3,585 à 9,88, et une valeur moyenne de 7,25 (Figure 28).

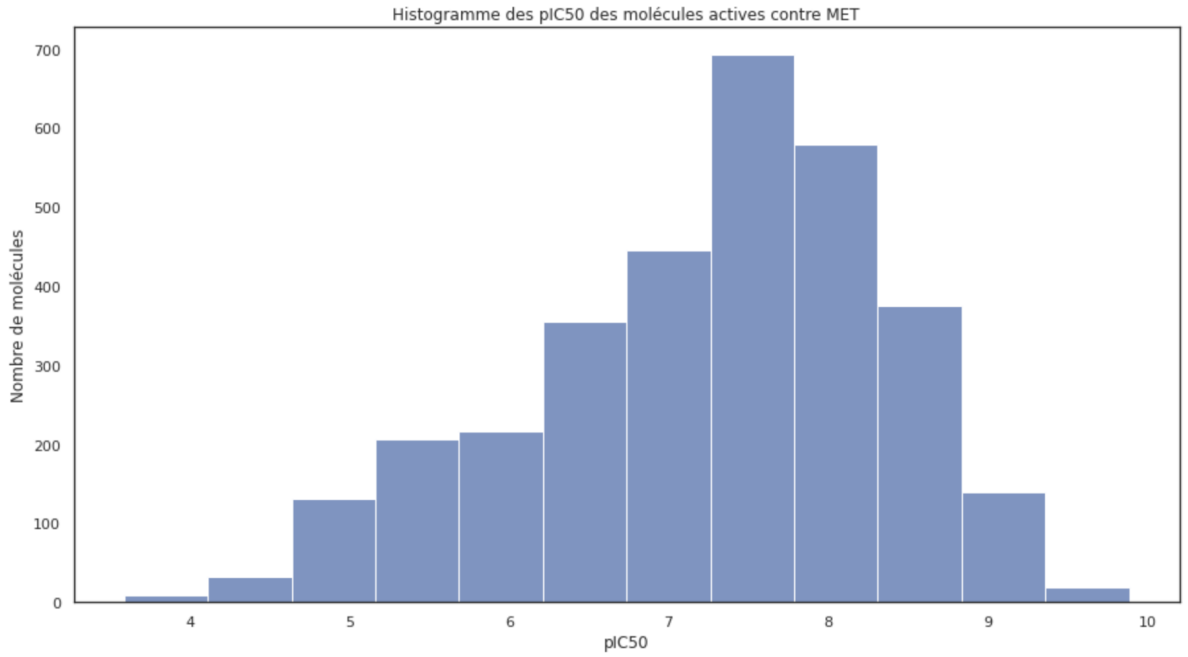


Figure 28 : Histogramme des molécules actives contre le récepteur MET.

A l'issue du filtrage sur l'adéquation aux règles de Lipinski, 1093 molécules furent éliminées et seules 2111 molécules furent conservées (Figure 29).

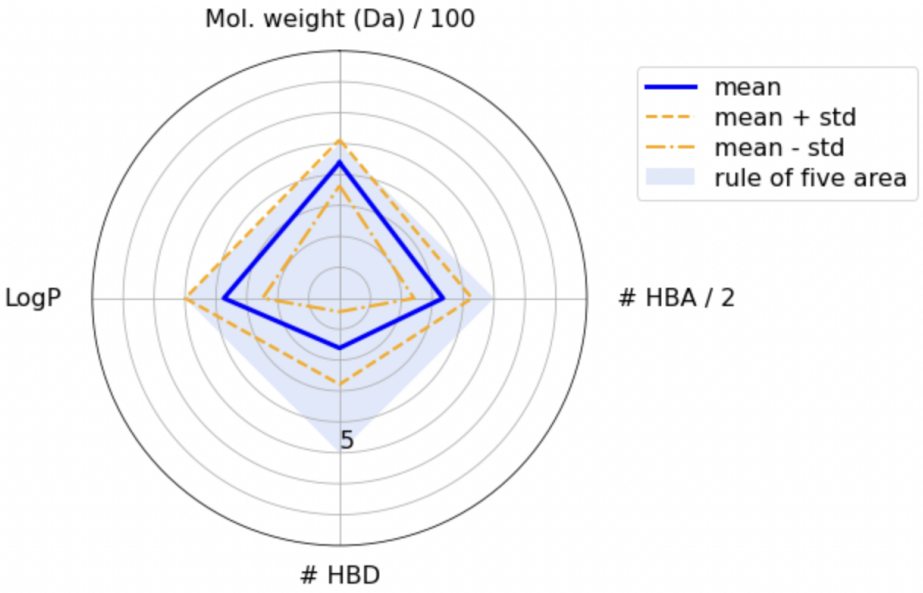


Figure 29 : Radar plot des molécules ayant passé le filtre selon les règles de Lipinski

A l'issue du filtrage portant sur les sous-structures sur-interagissant avec les protéines et sur les sous-structures indésirables, 1308 molécules furent sélectionnées.

III.b Molécules similaires actives sur MET

Sur la base des 1308 molécules sélectionnées, 73505 molécules, ayant une similarité structurale d'au moins 75%, furent extraites de la base de données PubMed. Parmi ces molécules, 66816 étaient en adéquation avec les règles de Lipinski (Figure 30) et l'identification de 13687 molécules comportant des motifs sur-interagissants et indésirables a finalement porté le nombre de molécules filtrées à 53129.

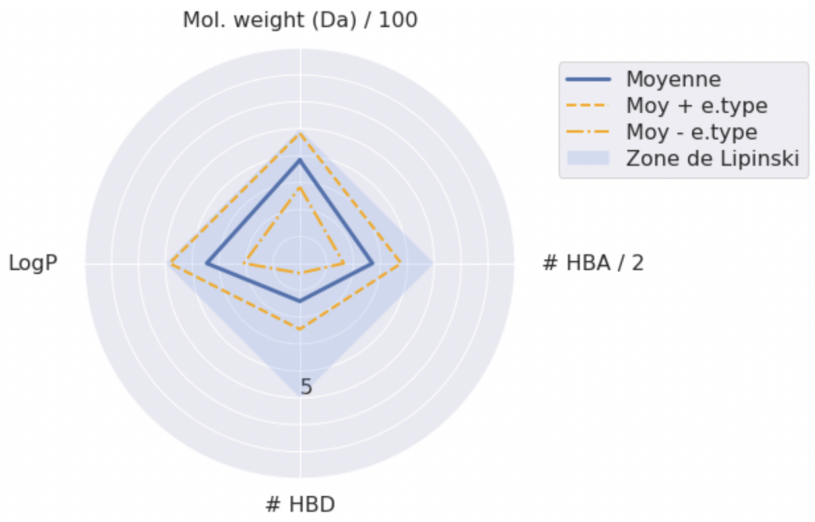


Figure 30 : Radar plot des molécules ayant passé le filtre selon les règles de Lipinski

Sur l'ensemble des molécules obtenues, un modèle deep learning est appliqué pour prédire leurs pIC50. Ce modèle consiste en un réseau neuronal profond à deux couches de cachés de 64 et 32 neurones. Il est entraîné sur le premier ensemble de molécules actives sur MET séparé en un jeu d'entraînement de 915 molécules et un jeu de test de 393 molécules. Le modèle prédit les valeurs de pIC50 avec une erreur absolue à la moyenne de 0,68 et une erreur quadratique moyenne de 0,82. La figure 31 montre le diagramme de dispersion entre les valeurs de pIC50 prédites et les valeurs vraies.

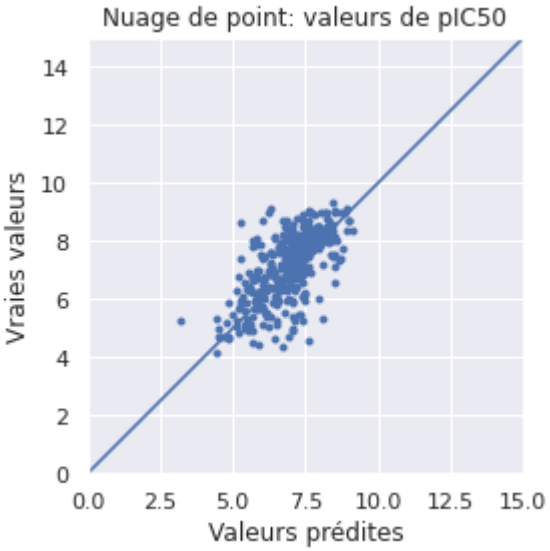


Figure 31 : Diagramme de dispersion entre les valeurs de pIC50 prédites et les valeurs vraies

Ce modèle est appliqué pour prédire les valeurs de pIC50 du deuxième ensemble de molécules sélectionnées (Figure 32)

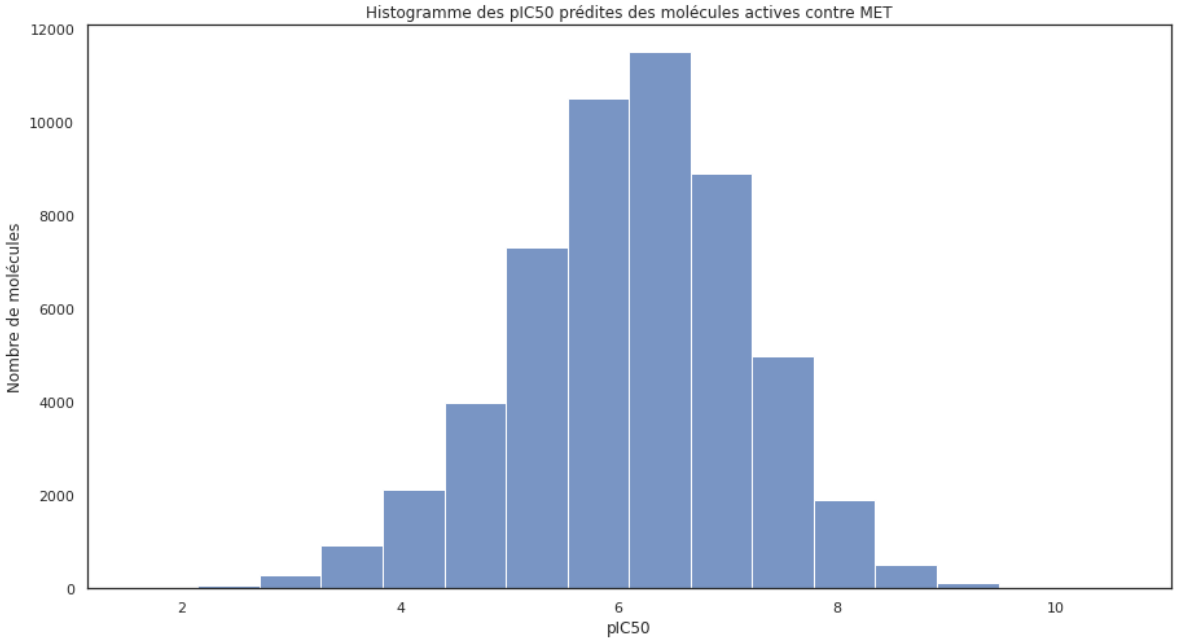


Figure 32 : Histogramme des pIC50 prédites

Afin de sélectionner uniquement les molécules les plus actives sur le récepteur MET nous sélectionnons uniquement les molécules ayant une valeur de pIC50 inférieure à

9. Après ce filtre, 89 molécules sont sélectionnées. La figure 33 montre les 8 molécules les plus actives avec leur pIC50.

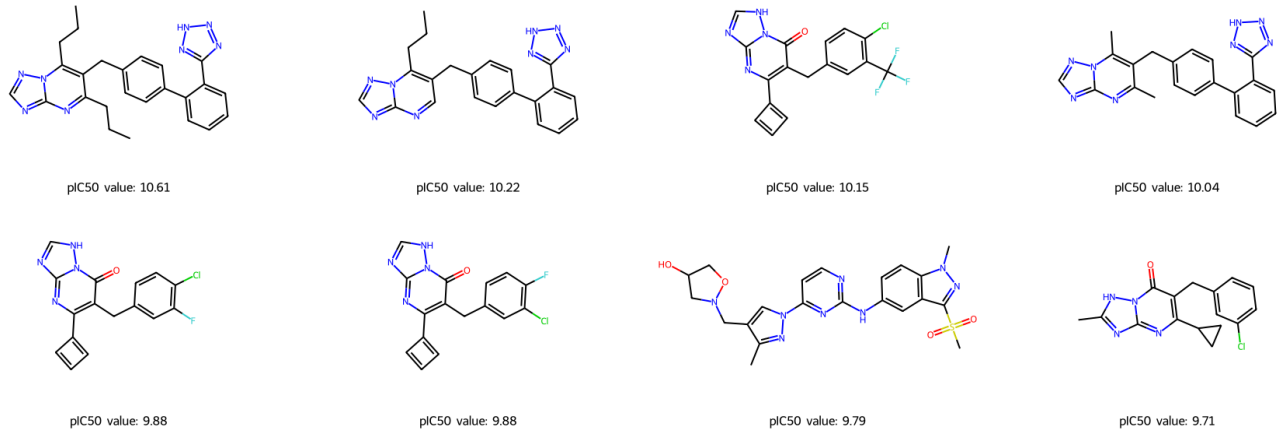
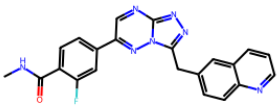


Figure 33 : Les 8 molécules les plus actives sur MET

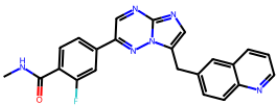
III.c Ensemble final des molécules

L'ensemble final des molécules comporte le premier ensemble obtenu à partir d'essais expérimentaux de mesure de l'IC50 sur le récepteur MET (1308 molécules) et le second ensemble de molécules obtenues par la récupération de molécules ayant une similarité de structure avec le premier ensemble de molécules d'au moins 75% (89 molécules) avec une pIC50 prédite supérieure à 9. Le jeu de données final comporte 1397 molécules.

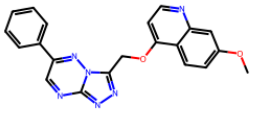
L'algorithme eToxPred est utilisé pour prédire le potentiel toxique des molécules à partir de leur codification SMILES. 1344 molécules ont une valeur de toxicité inférieure à 0.58. La figure 34 montre les 10 molécules prédites comme moins toxiques par l'algorithme eToxPred.



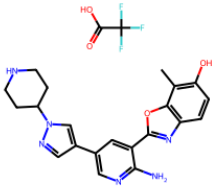
pIC50 value: 9.89



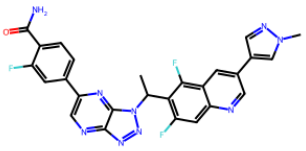
pIC50 value: 9.89



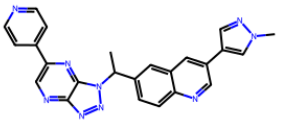
pIC50 value: 9.72



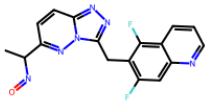
pIC50 value: 9.70



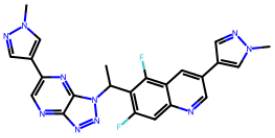
pIC50 value: 9.52



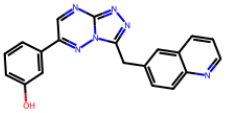
pIC50 value: 9.52



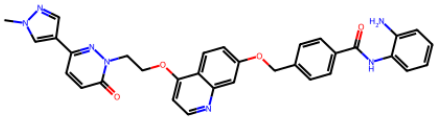
pIC50 value: 9.40



pIC50 value: 9.40



pIC50 value: 9.38



pIC50 value: 9.36

Figure 34 : Top 10 des molécules considérées comme moins toxiques par l'algorithme eToxPred

IV Discussion

L'objectif de cette partie était de proposer un début de pipeline de criblage virtuel sur le récepteur MET en utilisant uniquement les données issues de base de données publiquement disponibles, des algorithmes de prédictions de paramètres d'activité biologique et de toxicité. Pour cela, nous avons eu recours au données d'essais expérimentaux de détermination de concentration inhibitrice médiane de composés testé sur le récepteur MET issus de la base de données ChEMBL, à la base de données PubMed pour récupérer des molécules supplémentaires basées de la similarité structurale et un ensemble d'outil publiquement disponible pour par exemple, déterminer les paramètres physicochimiques à partir de la codification SMILES d'une molécule à l'aide du module python RDKit ou l'algorithme eToxPred pour prédire le potentiel toxique d'une molécule à partir de son code SMILES.

Le recours aux bases de données publiquement accessibles ou facilement accessibles à condition de ne l'utiliser qu'à des fins de recherches académiques tel DrugBank [167,168] constitue une forte opportunité pour accéder à des données de recherches de grande valeur. De nombreuses bases de données existent pour tout type de données biomédicales telles les données génomiques ou de voies métaboliques, phénotypiques, moléculaires et de screening [169–172]. Il est possible de construire comme nous l'avons fait, un flux d'analyses permettant de sélectionner, *in fine*, un nombre restreint de molécules candidates, à tester en laboratoire, en un temps limité et pour un coût limité.

Notre exemple d'application n'est que la première partie d'un flux d'analyse. Au final nous ne sélectionnons uniquement que des molécules sur leur concentration inhibitrice médiane prédite ou vraie sur le récepteur MET ainsi que sur leur potentielle toxicité. Pour compléter cette analyse, il faudrait potentiellement faire du docking moléculaire entre les molécules candidates et le récepteur MET et ses sites d'inhibition, rechercher des effets indésirables connus pour les molécules sélectionnées ou encore chercher à récupérer des données omics de lignées cellulaires tumorales ayant la mutation METex14 afin de faire progresser les analyses vers le modèle cellulaire.

Annexe 1 : Acquisition des données de composés

Objectif : Acquérir des données de composés testés sur une cible thérapeutique spécifique.

Prérequis

Importation des modules nécessaires :

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install rdkit
!pip install chembl_webresource_client
```

```
import math
import numpy as np
import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
import matplotlib.pyplot as plt
import pandas as pd
import pickle
from rdkit.Chem import PandasTools
from chembl_webresource_client.new_client import new_client
from tqdm.auto import tqdm
```

Récupération de la protéine sur la base de données ChEMBL

```
# Préparation de l'api pour accéder à la base de donnée ChEMBL
targets_api = new_client.target
compounds_api = new_client.molecule
bioactivities_api = new_client.activity


# Protéine à étudier : MET
# identifiant uniprot : P08581
#uniprot_id = "P00533"
uniprot_id = "P08581"
# Enrergistrement du code uniprot de la cible dans un fichier texte
with open("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/01_compoundDataAcquisition/uniprot_id.txt",
"wb") as file:
    pickle.dump(uniprot_id, file)
# Récupération des informations sur la protéine à partir de la base de
donnée ChEMBL
targets = targets_api.get(target_components__accession=uniprot_id).only(
    "target_chembl_id", "organism", "pref_name", "target_type"
)
targets = pd.DataFrame.from_records(targets)
# La protéine d'intérêt est la première entrée du dataframe
target = targets.iloc[0]
## Enregistrement de son chemblID
chembl_id = target.target_chembl_id
```

Récupération et traitement des données d'essais d'IC50

```
bioactivities = bioactivities_api.filter(
    target_chembl_id=chembl_id, type="IC50", relation="=", assay_type="B"
).only(
    "activity_id",
    "assay_chembl_id",
    "assay_description",
    "assay_type",
    "molecule_chembl_id",
    "type",
    "standard_units",
    "relation",
    "standard_value",
    "target_chembl_id",
    "target_organism",
)
# Téléchargement
bioactivities_df = pd.DataFrame.from_records(bioactivities)
print(f"Nombre de données de d'essais d'IC50: {bioactivities_df.shape[0]}")
```

```
Nombre de données de d'essais d'IC50: 4206
```

```
# Traitement des données d'essais d'IC50
bioactivities_df.drop(["units", "value"], axis=1, inplace=True)
# Conversition du type de donnée
bioactivities_df = bioactivities_df.astype({"standard_value": "float64"})
# Suppression des lignes avec des données manquantes
bioactivities_df.dropna(axis=0, how="any", inplace=True)
# Sélection uniquement des lignes pour lesquelles l'unité est en nM
bioactivities_df = bioactivities_df[bioactivities_df.standard_units ==
"nM"]
# Suppression des duplicatas dans les molécules
bioactivities_df.drop_duplicates("molecule_chembl_id", keep="first",
inplace=True)
# Remise à zéro des index du jeu de données
bioactivities_df.reset_index(drop=True, inplace=True)
# Renommage de certains noms de colonnes
bioactivities_df.rename(
    columns={"standard_value": "IC50", "standard_units": "units"},
inplace=True
)
print(f"Le jeu de donnée final comporte {bioactivities_df.shape[0]}
composés.")
```

```
Le jeu de donnée final comporte 3204 composés.
```

Récupération d'informations concernant les composés testés

```
# Récupération des informations des molécules
compounds_provider = compounds_api.filter(
    molecule_chembl_id__in=list(bioactivities_df["molecule_chembl_id"])
).only("molecule_chembl_id", "molecule_structures")

compounds = list(tqdm(compounds_provider))
compounds_df = pd.DataFrame.from_records(
    compounds,
)
# Suppression des lignes avec une donnée manquante
compounds_df.dropna(axis=0, how="any", inplace=True)
```

```
# Suppression des lignes dupliquées
compounds_df.drop_duplicates("molecule_chembl_id", keep="first",
inplace=True)
# Récupération uniquement de la strucutre sous la forme SMILES
canonical_smiles = []

for i, compounds in compounds_df.iterrows():
    try:

canonical_smiles.append(compounds["molecule_structures"]["canonical_smiles"
])
    except KeyError:
        canonical_smiles.append(None)

compounds_df["smiles"] = canonical_smiles
compounds_df.drop("molecule_structures", axis=1, inplace=True)
# Suppression des molécules sans structure SMILES
compounds_df.dropna(axis=0, how="any", inplace=True)
```

{"version_major":2,"version_minor":0,"model_id":"022bfff211f14ebf955538be78cfd4b4"}

Fusion des deux jeux de données

```
# Création d'une fonction qui converti l'IC50 en pIC50
def convert_ic50_to_pic50(IC50_value):
    pIC50_value = 9 - math.log10(IC50_value)
    return pIC50_value

output_df = pd.merge(
    bioactivities_df[["molecule_chembl_id", "IC50", "units"]],
    compounds_df,
    on="molecule_chembl_id",
)
```

Enregistrement du jeu de données

```
#output_df.to_csv(f"/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/01_compoundDataAcquisition/{uniprot_id}_com
pounds.csv", index=False)
```

Annexe 2 : Raffinage des données des composés sur leurs caractéristiques ADME

Objectif : Filtrer les molécules obtenues dans la première partie selon leurs caractéristiques physico-chimiques.

Prérequis

```
!pip install rdkit

from google.colab import drive
drive.mount('/content/drive')

from pathlib import Path
import math
import numpy as np
import pandas as pd
import os
from pickle import load
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import matplotlib.patches as mpatches
from rdkit import Chem
from rdkit.Chem import Descriptors, Draw, PandasTools, MolFromSmiles

os.chdir("/content/drive/MyDrive/Colab Notebooks/these_exercice/src")

import fct.molecularFiltering as molFilter
```

Chargement des données de composés

```
# Chargement de l'identifiant uniprot de la protéine étudiée
with open("../output/01_compoundDataAcquisition/uniprot_id.txt", "rb") as
file :
    uniprot_id = load(file)

# Chargement des données de composés
molecules =
pd.read_csv("../output/01_compoundDataAcquisition/"+uniprot_id+"_compounds.
csv")
```

Ajout d'informations

```
molecules["molecular_weight"] = molecules.smiles.apply(lambda x:
Descriptors.ExactMolWt(MolFromSmiles(x)))
molecules["n_hba"] = molecules.smiles.apply(lambda x:
Descriptors.NumHAcceptors(MolFromSmiles(x)))
molecules["n_hbd"] = molecules.smiles.apply(lambda x:
Descriptors.NumHDonors(MolFromSmiles(x)))
molecules["logp"] = molecules.smiles.apply(lambda x:
Descriptors.MolLogP(MolFromSmiles(x)))
```

Vérification de la conformité aux règles de Lipinksi

```
molecules["lipinski"] =
molecules.smiles.apply(molFilter.calculate_ro5_properties)

molecules_ro5_fulfilled = molecules[molecules["lipinski"]]
molecules_ro5_violated = molecules[~molecules["lipinski"]]
```

```
print(f"# compounds in unfiltered data set: {molecules.shape[0]}")
print(f"# compounds in filtered data set:
{molecules_ro5_fulfilled.shape[0]}")
print(f"# compounds not compliant with the Ro5:
{molecules_ro5_violated.shape[0]}")

# compounds in unfiltered data set: 3204
# compounds in filtered data set: 2111
# compounds not compliant with the Ro5: 1093

molecules_ro5_fulfilled_stats = molFilter.calculate_mean_std(
    molecules_ro5_fulfilled[["molecular_weight", "n_hba", "n_hbd", "logp"]]
)

 molecules_ro5_violated_stats = molFilter.calculate_mean_std(
    molecules_ro5_violated[["molecular_weight", "n_hba", "n_hbd", "logp"]]
)
```

Radar Plot

```
thresholds = {"molecular_weight": 500, "n_hba": 10, "n_hbd": 5, "logp": 5}
scaled_threshold = 5
properties_labels = [
    "Mol. weight (Da) / 100",
    "# HBA / 2",
    "# HBD",
    "LogP",
]
y_max = 8
```

Composés obéissant aux règles de Lipinski

```
molFilter.plot_radar(
    molecules_ro5_fulfilled_stats,
    thresholds,
    scaled_threshold,
    properties_labels,
    y_max,
    output_path = "../output/02_molecularFiltering/radarPlots_ro5"
)
```

Composés de respectant pas les régèles de Lipinski

```
molFilter.plot_radar(
    molecules_ro5_violated_stats,
    thresholds,
    scaled_threshold,
    properties_labels,
    y_max,
    output_path = "../output/02_molecularFiltering/radarPlots_nonRo5"
)
```

Enregistrement des composés obéissant aux règles de Lipinksi

```
molecules_ro5_fulfilled.to_csv("../output/02_molecularFiltering/"+uniprot_i
d+"_ro5Compliant.csv",
                               index=False)
```

Annexe 3 : Raffinage des composés selon leur structure moléculaire

Objectif : Filtrer les composés selon la présence de sous-structures non désirées.

Prérequis

```
!pip install rdkit &> /dev/null
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from os import listdir, chdir
from pickle import load

import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
import matplotlib.pyplot as plt

import pandas as pd
from tqdm.auto import tqdm
from rdkit import Chem
from rdkit.Chem import PandasTools
from rdkit.Chem.FilterCatalog import FilterCatalog, FilterCatalogParams
```

Chargement des données filtrés sur la règle de Lipinski

```
chdir("/content/drive/MyDrive/Colab Notebooks/these_exercice/src")

# Récupération de l'identifiant uniprot
with open("../output/01_compoundDataAcquisition/uniprot_id.txt", "rb") as
file:
    uniprot_id = load(file)

# chargement des molécules
moleculesData =
pd.read_csv("../output/02_molecularFiltering/"+uniprot_id+"_ro5Compliant.cs
v")

print(uniprot_id)
```

```
P08581
```

Filtre sur la présence de motifs sur-intéragissant, générateurs de bruit

```
# initialisation du filtre
params = FilterCatalogParams()
params.AddCatalog(FilterCatalogParams.FilterCatalogs.PAINS)
catalog = FilterCatalog(params)

"""
On ne garde que les molécules sans motifs
intéragissant avec toute sorte de protéines
"""

matches = []
clean = []
```

```
for index, row in tqdm(moleculesData.iterrows(),
total=moleculesData.shape[0]):
    molecule = Chem.MolFromSmiles(row.smiles)
    entry = catalog.GetFirstMatch(molecule)  # Get the first matching PAINS
    if entry is not None:
        # store PAINS information
        matches.append(
            {
                "chembl_id": row.molecule_chembl_id,
                "rdkit_molecule": molecule,
                "pains": entry.GetDescription().capitalize(),
            }
        )
    else:
        # collect indices of molecules without PAINS
        clean.append(index)

matches = pd.DataFrame(matches)
moleculesData = moleculesData.loc[clean]
```

{"version_major":2,"version_minor":0,"model_id":"f722236bb71844a9a757c21062c30c4c"}

```
# NBVAL_CHECK_OUTPUT
print(f"Number of compounds with PAINS: {len(matches)}")
print(f"Number of compounds without PAINS: {len(moleculesData)}")
```

```
Number of compounds with PAINS: 188
Number of compounds without PAINS: 1923
```

Filtre sur la présence de motifs indésirables

```
# Chargement des sous-structures non désirées
substructures = pd.read_csv("../input/unwantedCompounds.csv", sep=";")
substructures["rdkit_molecule"] =
substructures.smarts.apply(Chem.MolFromSmarts)
print("Number of unwanted substructures in collection:",
len(substructures))
# NBVAL_CHECK_OUTPUT
```

```
Number of unwanted substructures in collection: 101
```

```
print(moleculesData.shape)
```

```
(1923, 10)
```

```
# search for unwanted substructure
matches = []
clean = []
for index, row in tqdm(moleculesData.iterrows(),
total=moleculesData.shape[0]):
    molecule = Chem.MolFromSmiles(row.smiles)
    match = False
    for _, substructure in substructures.iterrows():
        if molecule.HasSubstructMatch(substructure.rdkit_molecule):
            matches.append(
                {
                    "chembl_id": row.molecule_chembl_id,
                    "rdkit_molecule": molecule,
                    "substructure": substructure.rdkit_molecule,
```

```
                    "substructure_name": substructure["name"],
                }
            )
            match = True
    if not match:
        clean.append(index)
```

{"version_major":2,"version_minor":0,"model_id":"7c787cce1e19466c9e90952c3e7746e5"}

```
print(len(clean))
```

```
1308
```

```
matches = pd.DataFrame(matches)
moleculesData = moleculesData.loc[clean]

# Vérification du nombres de molécules sans sous structures non voules
print(f"Number of found unwanted substructure: {len(matches)}")
print(f"Number of compounds without unwanted substructure:
{len(moleculesData)}")
```

```
Number of found unwanted substructure: 923
Number of compounds without unwanted substructure: 1308
```

Enregistrement des molécules

```
moleculesData.to_csv("../output/03_molecularSubstructureFiltering//"+unipro
t_id+".csv",
                                index=False)
```

Annexe 4 : Recherche de molécules similaires à celles sélectionnées.

Objectif : Questionner la base de données PubChem sur des composés similaires à ceux sélectionnés.

En cas de doublons nous conserverons le couple molécule-candidat avec le maximum de similarité

Pré-requis

```
!pip install rdkit &> /dev/null
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import time
from urllib.parse import quote

import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
import matplotlib.pyplot as plt

from pickle import load, dump
from tqdm import trange

from IPython.display import Markdown, Image
import requests
import pandas as pd

from os import chdir
chdir("/content/drive/MyDrive/Colab Notebooks/these_exercice/src")


from rdkit import Chem
from rdkit.Chem import PandasTools
from rdkit.Chem.Draw import MolsToGridImage

# Fonction de recherche de molécules similaires, seuil fixé par défaut à
75%
def query_pubchem_for_similar_compounds(smiles, threshold=75,
n_records=10):
    """
    Query PubChem for similar compounds and return the job key.

    Parameters
    ----------
    smiles : str
        The canonical SMILES string for the given compound.
    threshold : int
        The threshold of similarity, default 75%. In PubChem, the default
threshold is 90%.
    n_records : int
        The maximum number of feedback records.
```

```
    Returns
    -------
    str
        The job key from the PubChem web service.
    """
    escaped_smiles = quote(smiles).replace("/", ".")
    url =
f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/similarity/smiles/{esc
aped_smiles}/JSON?Threshold={threshold}&MaxRecords={n_records}"
    r = requests.get(url)
    r.raise_for_status()
    key = r.json()["Waiting"]["ListKey"]
    return key

# Fonction de téléchargement des résultats
def check_and_download(key, attempts=30):
    """
    Check job status and download PubChem CIDs when the job finished

    Parameters
    ----------
    key : str
        The job key of the PubChem service.
    attempts : int
        The time waiting for the feedback from the PubChem service.

    Returns
    -------
    list
        The PubChem CIDs of similar compounds.
    """
    url =
f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/listkey/{key}/cids/JSO
N"
    #print(f"Querying for job {key} at URL {url}...", end="")
    while attempts:
        r = requests.get(url)
        r.raise_for_status()
        response = r.json()
        if "IdentifierList" in response:
            cids = response["IdentifierList"]["CID"]
            break
        attempts -= 1
        #print(".", end="")
        time.sleep(10)
    else:
        raise ValueError(f"Could not find matches for job key: {key}")
    return cids

# Récupération de l'ID SMILES à partir du CID PubChem
def smiles_from_pubchem_cids(cids):
    """
    Get the canonical SMILES string from the PubChem CIDs.

    Parameters
    ----------
```

```
    cids : list
        A list of PubChem CIDs.

    Returns
    -------
    list
        The canonical SMILES strings of the PubChem CIDs.
    """
    url =
f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/{','.join(map(str,
cids))}/property/CanonicalSMILES/JSON"
    r = requests.get(url)
    r.raise_for_status()
    return [item["CanonicalSMILES"] for item in
r.json()["PropertyTable"]["Properties"]]
```

Récupération des molécules filtrées

```
# Récupération de l'identifiant uniprot
with open("../output/01_compoundDataAcquisition/uniprot_id.txt", "rb") as
file:
    uniprot_id = load(file)

# Chargement des données
molecules =
pd.read_csv("../output/03_molecularSubstructureFiltering//"+uniprot_id+".cs
v")

print(uniprot_id)
print(molecules.shape)
```

```
P08581
(1308, 10)
```

Recherche de molécumes similaires

```
# Enregistrement des IDs restant à traiter
"""
LIGNE A LANCER QU'UNE SEULE FOIS LORS DE L'INITIALISATION DU PIPELINE!
"""
#molecules.loc[:,["molecule_chembl_id","smiles"]].to_csv("../input/05_findS
imilarCompounfPubChem/idToProcess_" + uniprot_id + ".csv", index=False)

idToProc.head()
```

```
  molecule_chembl_id                                             smiles
0      CHEMBL3582305  CNC(=O)c1ccc(-c2cnc3nnc(Cc4ccc5ncccc5c4)n3n2)cc1F
1      CHEMBL3188267  CNC(=O)c1ccc(-c2cnc3ncc(Cc4ccc5ncccc5c4)n3n2)cc1F
2      CHEMBL3797911       COc1ccc2c(OCc3nnc4ncc(-c5ccccc5)nn34)ccnc2c1
3      CHEMBL2032280  Cc1c(O)ccc2nc(-c3cc(-c4cnn(C5CCNCC5)c4)cnc3N)o...
4      CHEMBL4104884  CC(c1c(F)cc2ncc(-c3cnn(C)c3)cc2c1F)n1nnc2ncc(-...
```

```
# Il faut ajouter des gardes-fous, des moyens de contrôle du bon
déroulement de la boucle.
idToProc =
pd.read_csv("../input/05_findSimilarCompounfPubChem/idToProcess_" +
uniprot_id + ".csv")
listQueries = idToProc.iloc[:,1].to_list()
for i in trange(len(listQueries)):
  #print(f" \ni : {i} \n")
```

```
  #print(f"shape idToProc : {idToProc.shape}")
  #print(f"\n index : {i}")
  query = listQueries[i]
  #print(f"\n query : \n{query} \n")
  #print(idToProc.iloc[i,0:2])
  #print(f"query : {query}")
  #print("\t get similar CID")
  job_key = query_pubchem_for_similar_compounds(query, n_records=400)
  #print("\t Download ...")
  similar_cids = check_and_download(job_key)
  #print("\t Convert SMILES to PubMEd ID")
  similar_smiles = smiles_from_pubchem_cids(similar_cids)

  dataframe = pd.DataFrame(similar_smiles).T
  dataframe = dataframe.set_index(pd.Index([idToProc.iloc[i,0]]))
  dataframe.to_csv("../output/05_findSimilarCompounfPubChem/processedIds_"
+ uniprot_id + ".csv", mode = "a", header=False)

  #print(f"shape processedIds :
{pd.read_csv('../output/05_findSimilarCompounfPubChem/processedIds_' +
uniprot_id + '.csv').shape}")
  #print(f"\t Remove value at index : {idToProc[idToProc.smiles ==
query].index.values}")
  # save remaining ids to process
  if i == 0 :
    tmpIdToProc = idToProc.copy()
  tmpIdToProc.drop(tmpIdToProc[tmpIdToProc.smiles == query].index, axis =
0, inplace = True)
  #idToProc.drop(idToProc[idToProc.smiles == query].index, axis = 0,
inplace = True)
  #print(idToProc.shape)
  #(idToProc.
   #drop(idToProc[idToProc.smiles == query].index).
   #to_csv("../input/05_findSimilarCompounfPubChem/idToProcess_" +
uniprot_id + ".csv", index=False))
  tmpIdToProc.to_csv("../input/05_findSimilarCompounfPubChem/idToProcess_"
+ uniprot_id + ".csv", index=False)
  #print("\n")


100%|██████████| 353/353 [1:34:13<00:00, 16.02s/it]
```

Annexe 5 : Filtrage des composés similaires aux candidats

Objectif : A partir des composés trouvés sur la base de données PubChem, filtrer ces composés sur la base de leurs caractéristiques physico-chimiques et leur respect de la règle de Lipinski.

Prérequis

```
!pip install rdkit &> /dev/null

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
import matplotlib.pyplot as plt

from pickle import load
from os import chdir
chdir("/content/drive/MyDrive/Colab Notebooks/these_exercice/src")

from tqdm import tqdm

from rdkit import RDLogger , Chem
from rdkit.Chem import Descriptors, MolFromSmiles
from rdkit.Chem.FilterCatalog import FilterCatalog, FilterCatalogParams
# Désactivation des xarnings de rdkit
RDLogger.DisableLog('rdApp.*')

import fct.molecularFiltering as molFilter

def find_properties(listSmiles, function) :
    fun = function
    molProp = []
    for i in tqdm(range(len(listSmiles))) :
        mol = molecules[i]
        try :
            molProp.append(fun(MolFromSmiles(mol)))
        except :
            molProp.append(None)
    return(molProp)
```

Importation des molécules récupérées

```
# Récupération de l'identifiant uniprot
with open("../output/01_compoundDataAcquisition/uniprot_id.txt", "rb") as
file:
    uniprot_id = load(file)
print(uniprot_id)
# Chargement des données
molecules =
pd.read_csv("../output/05_findSimilarCompounfPubChem/processedIds_" +
uniprot_id + ".csv", header=None)
# Supression de la première colonne
```

```
molecules = molecules.drop(0, axis=1)
# Conversion de l'ensemble des colonnes en un dataframe ave une seule
colonne : smiles
molecules = [item for i in molecules.values for item in i]
# Supression des doublons
molecules = list(set(molecules))
print(len(molecules))
```

```
P08581
73505
```

Calcul des propriétés physico-chimiques

```
print("Computing molecular weight")
molecules_molecular_weight = find_properties(molecules,
Descriptors.ExactMolWt)
print("HBA")
molecules_n_hba = find_properties(molecules, Descriptors.NumHAcceptors)
print("HBD")
molecules_n_hbd = find_properties(molecules, Descriptors.NumHDonors)
print("LogP")
molecules_logp = find_properties(molecules, Descriptors.MolLogP)
# Create a dataframe
molecules = pd.DataFrame(list(zip(molecules,
                                  molecules_molecular_weight,
                                  molecules_n_hba,
                                  molecules_n_hbd,
                                  molecules_logp)),

columns=["smiles","molecular_weight","n_hba","n_hbd","logp"])
# Supression des données manquantes
molecules = molecules.dropna()
print("Compute Lipinski rule")
molecules["lipinski"] =
molecules.smiles.apply(molFilter.calculate_ro5_properties)
```

```
Computing molecular weight

100%|██████████| 73505/73505 [00:21<00:00, 3418.85it/s]

HBA

100%|██████████| 73505/73505 [00:17<00:00, 4288.90it/s]

HBD

100%|██████████| 73505/73505 [00:13<00:00, 5546.39it/s]

LogP

100%|██████████| 73505/73505 [00:39<00:00, 1878.87it/s]

Compute Lipinski rule
```

```
# Restriction des molécules à celles respectant la règle de Lipinski
molecules = molecules[molecules["lipinski"]]
print(f"{molecules.shape[0]} molecules follows Lipinski rules")
```

```
66816 molecules follows Lipinski rules
```

Radar plot

```
thresholds = {"molecular_weight": 500, "n_hba": 10, "n_hbd": 5, "logp": 5}
scaled_threshold = 5
properties_labels = [
    "Mol. weight (Da) / 100",
    "# HBA / 2",
    "# HBD",
    "LogP",
]
y_max = 8

molecules_stats = molFilter.calculate_mean_std(
    molecules[["molecular_weight", "n_hba", "n_hbd", "logp"]]
)

molFilter.plot_radar(
    molecules_stats,
    thresholds,
    scaled_threshold,
    properties_labels,
    y_max,
    output_path = "../output/06_similarCompoundMolecularFiltering/"
)
```

Filtre sur la présence de motifs sur-intéragissant, générateurs de bruit

```
# initialisation du filtre
params = FilterCatalogParams()
params.AddCatalog(FilterCatalogParams.FilterCatalogs.PAINS)
catalog = FilterCatalog(params)

#molecules.set_index("smiles", inplace = True)
molecules.head()
```

```
                                              smiles  molecular_weight
n_hba  \
1      C1COCCN1CCCCC(=O)NC2=CC=C(C=C2)C3=CN=CC=C3.Cl        375.171355
4.0
2  C1CCC(CC1)CCC(C(=O)N)NC(=O)C2=CC=C(C=C2)CNC(=O...        502.269239
6.0
3  C1CC1C2=CC(=NN2)NC3=NC(=NC=C3)NC4=CC5=C(C=C4)N...        332.149793
6.0
4  CN1C=C(C=N1)C2=C3C4=CC=CC=C4C(C3=CC(=C2)C(=O)N...        443.145676
5.0
5      CS(=O)(=O)CCN1C=CC2=C1C=CC=C2OC3=NC=NC(=C3)CN        346.109961
7.0

   n_hbd    logp  lipinski
1    1.0  3.6114      True
2    4.0  3.3287      True
3    4.0  3.4406      True
4    2.0  2.9300      True
5    1.0  1.7269      True
```

```
"""
On ne garde que les molécules sans motifs
intéragissant avec toute sorte de protéines
"""
```

```
matches = []
clean = []
for index, row in tqdm(molecules.iterrows(), total=molecules.shape[0]):
    molecule = Chem.MolFromSmiles(row.smiles)
    entry = catalog.GetFirstMatch(molecule)  # Get the first matching PAINS
    if entry is not None:
        # store PAINS information
        matches.append(
            {
                "smiles": row.smiles,
                "rdkit_molecule": molecule,
                "pains": entry.GetDescription().capitalize(),
            }
        )
    else:
        # collect indices of molecules without PAINS
        clean.append(index)

matches = pd.DataFrame(matches)
moleculesData = molecules.loc[clean]
```

```
100%|██████████| 66816/66816 [04:03<00:00, 274.82it/s]
```

```
# NBVAL_CHECK_OUTPUT
print(f"Number of compounds with PAINS: {len(matches)}")
print(f"Number of compounds without PAINS: {len(moleculesData)}")
```

```
Number of compounds with PAINS: 2929
Number of compounds without PAINS: 63887
```

Filtre sur la présence de motifs indésirables

```
# Chargement des sous-structures non désirées
substructures = pd.read_csv("../input/unwantedCompounds.csv", sep=";")
substructures["rdkit_molecule"] =
substructures.smarts.apply(Chem.MolFromSmarts)
print("Number of unwanted substructures in collection:",
len(substructures))
# NBVAL_CHECK_OUTPUT
```

```
Number of unwanted substructures in collection: 101
```

```
# search for unwanted substructure
matches = []
clean = []
for index, row in tqdm(moleculesData.iterrows(),
total=moleculesData.shape[0]):
    molecule = Chem.MolFromSmiles(row.smiles)
    match = False
    for _, substructure in substructures.iterrows():
        if molecule.HasSubstructMatch(substructure.rdkit_molecule):
            matches.append(
                {
                    "smiles": row.smiles,
                    "rdkit_molecule": molecule,
                    "substructure": substructure.rdkit_molecule,
                    "substructure_name": substructure["name"],
                }
```

```
            )
            match = True
    if not match:
        clean.append(index)

matches = pd.DataFrame(matches)
moleculesData = moleculesData.loc[clean]
```

```
100%|██████████| 63887/63887 [07:54<00:00, 134.58it/s]
```

```
# Vérification du nombres de molécules sans sous structures non voules
print(f"Number of found unwanted substructure: {len(matches)}")
print(f"Number of compounds without unwanted substructure:
{len(moleculesData)}")
```

```
Number of found unwanted substructure: 14835
Number of compounds without unwanted substructure: 53129
```

```
moleculesData.shape
```

```
(53129, 6)
```

Enregistrement des données

```
moleculesData.to_csv("../output/06_similarCompoundMolecularFiltering/"+unip
rot_id+"_ro5Compliant.csv",
                                index=False)
```

Annexe 6 : Prédiction des IC50

Objectif : Prédire les IC50 des molécules récupérées par similarité de structure sur la protéine d'intérêt

Prérequis

```
!pip install rdkit &> /dev/null

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

from pickle import load

from tqdm import tqdm
# Create new `pandas` methods which use `tqdm` progress
# (can use tqdm_gui, optional kwargs, etc.)
tqdm.pandas()


import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
import matplotlib.pyplot as plt

from warnings import filterwarnings

# Silence some expected warnings
filterwarnings("ignore")

from urllib.parse import quote

import pandas as pd
import numpy as np
import requests
from rdkit import Chem
from rdkit.Chem import MACCSkeys, Draw
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import metrics
import seaborn as sns

# Neural network specific libraries
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import ModelCheckpoint

%matplotlib inline

from matplotlib.axis import YAxis
# Fonction pour calculer l'empreinte d'un molécule à partir de sa structure
smiles
def smiles_to_fp(smiles, method="maccs", n_bits=2048):
    """
    Encode a molecule from a SMILES string into a fingerprint.
```

```
    Parameters
    ----------
    smiles : str
        The SMILES string defining the molecule.

    method : str
        The type of fingerprint to use. Default is MACCS keys.

    n_bits : int
        The length of the fingerprint.

    Returns
    -------
    array
        The fingerprint array.
    """

    # Convert smiles to RDKit mol object
    mol = Chem.MolFromSmiles(smiles)

    if method == "maccs":
        return np.array(MACCSkeys.GenMACCSKeys(mol))
    if method == "morgan2":
        return np.array(GetMorganFingerprintAsBitVect(mol, 2,
nBits=n_bits))
    if method == "morgan3":
        return np.array(GetMorganFingerprintAsBitVect(mol, 3,
nBits=n_bits))
    else:
        print(f"Warning: Wrong method specified: {method}." " Default will
be used instead.")
        return np.array(MACCSkeys.GenMACCSKeys(mol))

# Fonction pour construire le réseau de neurones
def neural_network_model(hidden1, hidden2):
    """
    Creating a neural network from two hidden layers
    using ReLU as activation function in the two hidden layers
    and a linear activation in the output layer.

    Parameters
    ----------
    hidden1 : int
        Number of neurons in first hidden layer.

    hidden2: int
        Number of neurons in second hidden layer.

    Returns
    -------
    model
        Fully connected neural network model with two hidden layers.
    """

    model = Sequential()
    # First hidden layer
```

```
    model.add(Dense(hidden1, activation="relu", name="layer1"))
    # Second hidden layer
    model.add(Dense(hidden2, activation="relu", name="layer2"))
    # Output layer
    model.add(Dense(1, activation="linear", name="layer3"))

    # Compile model
    model.compile(loss="mean_squared_error", optimizer="adam",
metrics=["mse", "mae"])
    return model

# Fonction pour obtenir le CID d'un composé à partir de son identifiant
SMILES
def query_pubmed_cidFromSmiles(smiles):
    """
    Query PubChem for CID of a SMILES sting.

    Parameters
    ----------
    smiles : str
        The canonical SMILES string for the given compound.

    Returns
    -------
    str
        The job key from the PubChem web service.
    """
    escaped_smiles = quote(smiles).replace("/", ".")
    url =
f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/smiles/{escaped_smiles
}/cids/JSON"
    r = requests.get(url)
    r.raise_for_status()
    key = r.json()["IdentifierList"]["CID"][0]
    if key == "0" :
      return("None")
    else :
      return key

# Fonction pour obtenir les références de brevets pour un composé
def query_patent_cid(cid):
    """
    Query PubChem for CID of a SMILES sting.

    Parameters
    ----------
    smiles : str
        The canonical SMILES string for the given compound.

    Returns
    -------
    str
        The job key from the PubChem web service.
    """
    try :
      url =
f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/{cid}/xrefs/Patent
```

```
ID/JSON"
      r = requests.get(url)
      r.raise_for_status()
      y = r.json()["InformationList"]["Information"]
      y = y[0].get("PatentID")
    except :
      y = "None"

    return y
```

Chargement des données

```
# Récupération de l'identifiant uniprot
with open("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/01_compoundDataAcquisition/uniprot_id.txt",
"rb") as file:
    uniprot_id = load(file)
print(uniprot_id)
# Récupération de toutes les molécules ayant des valeurs de pIC50 pour la
protéine d'intérêt
prot_molecules = pd.read_csv(f"/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/01_compoundDataAcquisition/{uniprot_id}_com
pounds.csv")
# Récupération des molécules sélectionnées pour leur interaction avec la
protéine cible
firstMolecules = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/03_molecularSubstructureFiltering/"+uniprot
_id+".csv")
# Récupération des molécules sélectionnées pour leur similarités de
structures avec le premier
# set de molécules
secondMolecules = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/06_similarCompoundMolecularFiltering/"+unip
rot_id+"_ro5Compliant.csv")

print(firstMolecules.shape)
print(secondMolecules.shape)
```

```
P08581
(1308, 10)
(53129, 6)
```

Entrainement du modèle

Encodage de la structure moléculaire

```
firstMolecules["fingerprints_df"] =
firstMolecules["smiles"].apply(smiles_to_fp)

# Look at head
print("Shape of dataframe:", firstMolecules.shape)
```

```
Shape of dataframe: (1308, 11)
```

Préparation des données d'entrainement et de test

```
# Split the data into training and test set
x_train, x_test, y_train, y_test = train_test_split(
    firstMolecules["fingerprints_df"], firstMolecules[["pIC50"]],
test_size=0.3, random_state=42
```

```
)

# Print the shape of training and testing data
print("Shape of training data:", x_train.shape)
print("Shape of test data:", x_test.shape)
# NBVAL_CHECK_OUTPUT

Shape of training data: (915,)
Shape of test data: (393,)
```

Entrainement du modèle

```
# Neural network parameters
batch_sizes = [16, 32, 64]
nb_epoch = 50
layer1_size = 64
layer2_size = 32

# Plot
fig = plt.figure(figsize=(12, 6))
sns.set(color_codes=True)
for index, batch in enumerate(batch_sizes):
    fig.add_subplot(1, len(batch_sizes), index + 1)
    model = neural_network_model(layer1_size, layer2_size)

    # Fit model on x_train, y_train data
    history = model.fit(
        np.array(list((x_train))).astype(float),
        y_train.values,
        batch_size=batch,
        validation_data=(np.array(list((x_test))).astype(float),
y_test.values),
        verbose=0,
        epochs=nb_epoch,
    )
    plt.plot(history.history["loss"], label="train")
    plt.plot(history.history["val_loss"], label="test")
    plt.legend(["train", "test"], loc="upper right")
    plt.ylabel("loss")
    plt.xlabel("epoch")
    plt.ylim((0, 15))
    plt.title(
        f"test loss = {history.history['val_loss'][nb_epoch-1]:.2f}, "
f"batch size = {batch}"
    )
plt.show()

# Save the trained model
filepath = "/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/07_predictIC50/best_weights.hdf5"
checkpoint = ModelCheckpoint(
    str(filepath),
    monitor="loss",
    verbose=0,
    save_best_only=True,
    mode="min",
    save_weights_only=True,
)
callbacks_list = [checkpoint]
```

```
# Fit the model
model.fit(
    np.array(list((x_train))).astype(float),
    y_train.values,
    epochs=nb_epoch,
    batch_size=64,
    callbacks=callbacks_list,
    verbose=0,
)
```

```
<keras.callbacks.History at 0x7f6e751e67d0>
```

Evaluation et prédiction

```
# Evalute the model
print(f"Evaluate the model on the test data")
scores = model.evaluate(np.array(list((x_test))), y_test.values, verbose=0)
print(f" loss: {scores[0]:.2f}")
print(f" mse (same as loss): {scores[1]:.2f}")
print(f" mae: {scores[2]:.2f}")
```

```
Evaluate the model on the test data
 loss: 0.82
 mse (same as loss): 0.82
 mae: 0.68
```

```
# Predict pIC50 values on x_test data
y_pred = model.predict(np.array(list((x_test))))

# Print 5 first pIC50 predicted values
first_5_prediction = [print(f"{value[0]:.2f}") for value in y_pred[0:5]]
```

```
13/13 [==============================] - 0s 3ms/step
5.69
6.42
6.89
6.88
6.89
```

```
# Scatter plot
limits = 0, 15
fig, ax = plt.subplots()
ax.scatter(y_pred, y_test, marker=".")
lin = np.linspace(*limits, 100)
ax.plot(lin, lin)
ax.set_aspect("equal", adjustable="box")
ax.set_xlabel("Valeurs prédites")
ax.set_ylabel("Vraies valeurs")
ax.set_title("Nuage de point: valeurs de pIC50")
ax.set_xlim(limits)
ax.set_ylim(limits)
plt.show()
```

Prédiction de l'IC50 des molécules récupérée sur PubMed

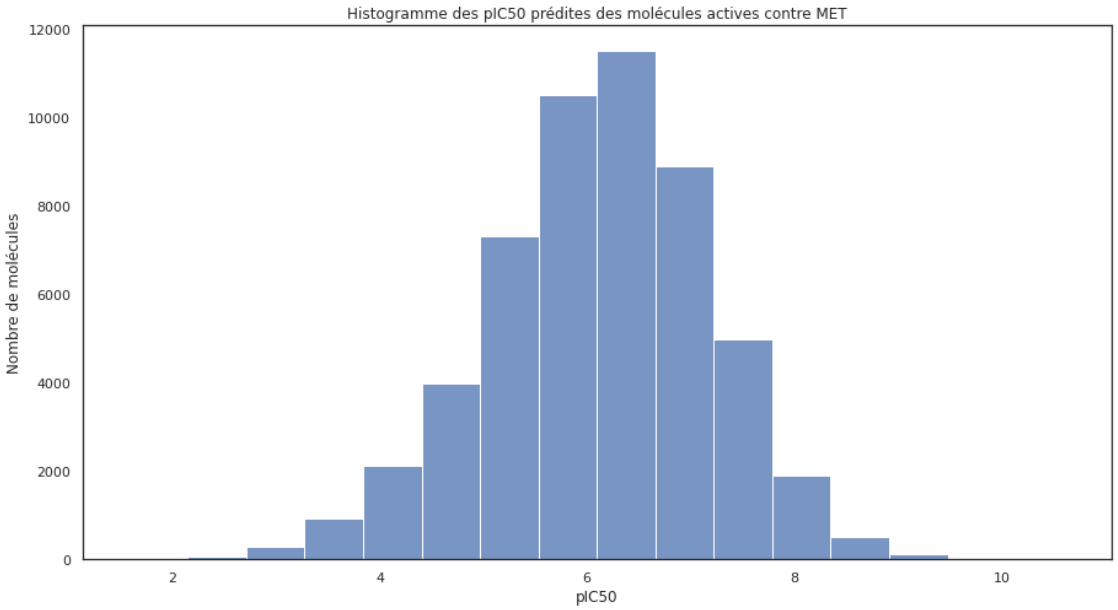
Encodage de la structure moléculaire

```
secondMolecules["fingerprint_df"] =
secondMolecules["smiles"].apply(smiles_to_fp)
```

Prédiction des IC50

```
# Prediction on external/unlabeled data
predictions = model.predict(
    np.array(list((secondMolecules["fingerprint_df"]))).astype(float),
callbacks=callbacks_list)

predicted_pIC50 = pd.DataFrame(predictions, columns=["predicted_pIC50"])
predicted_pIC50_df = secondMolecules.join(predicted_pIC50)
sns.set(rc = {'figure.figsize':(15,8)})
sns.set_style("white")
fig = sns.histplot(predicted_pIC50_df, x="predicted_pIC50",bins =
int(1+3.322*np.log10(predicted_pIC50_df.shape[0])))
fig.set(xlabel="pIC50", ylabel="Nombre de molécules")
plt.title("Histogramme des pIC50 prédites des molécules actives contre
MET")
plt.show()
```



Filtrage des molécules sur la base de leur IC50

On ne garde que les molécules avec une IC50 inférieur au nM

```
predicted_pIC50_df[predicted_pIC50_df.predicted_pIC50 >
9].sort_values("predicted_pIC50", ascending = False).shape
```

```
(89, 8)
```

```
threshold_pIC50 = 9
pIC50_inf = predicted_pIC50_df[predicted_pIC50_df.predicted_pIC50 >
threshold_pIC50].sort_values("predicted_pIC50", ascending = False)
# Affichage des molécules
highest_pIC50 = pIC50_inf["smiles"]

mols_protein = [Chem.MolFromSmiles(smile) for smile in pIC50_inf.smiles]
pIC50_protein = pIC50_inf["predicted_pIC50"].tolist()
pIC50_values = [(f"pIC50 value: {value:.2f}") for value in pIC50_protein]

Draw.MolsToGridImage(mols_protein[0:8], molsPerRow=4, subImgSize=(450,
300), legends=pIC50_values[0:10])
```

Récupération des CID pour les molécules

```
pIC50_inf.shape
```

```
(89, 8)
```

```
pIC50_inf["cid"] =
pIC50_inf["smiles"].progress_apply(query_pubmed_cidFromSmiles)
pIC50_inf["patentID"] = pIC50_inf.cid.progress_apply(query_patent_cid)
```

```
100%|██████████| 89/89 [00:43<00:00,  2.05it/s]
100%|██████████| 89/89 [00:14<00:00,  6.09it/s]
```

```
pIC50_inf.shape
```

```
(89, 10)
```

```
pIC50_inf.to_csv(f"/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/07_predictIC50/mol_pIC50sup{threshold_pIC50
}_{uniprot_id}.csv", index = False)
```

Annexe 7 : Prédiction de la toxicité

Prérequis

```
!python --version
```

```
Python 3.7.15
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from pickle import load

import pandas as pd

from rdkit import Chem
from rdkit.Chem import MACCSkeys, Draw
```

Chargement des données

```
# Récupération de l'identifiant uniprot
with open("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/01_compoundDataAcquisition/uniprot_id.txt",
"rb") as file:
    uniprot_id = load(file)
print(uniprot_id)
```

```
P08581
```

```
# Importation du premier ensemble de données
firstMolecules = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/03_molecularSubstructureFiltering//"+unipro
t_id+".csv")
print(f"shape first molecules : {firstMolecules.shape}")

# Importation du second ensemble de données
secondMolecules = pd.read_csv(f"/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/07_predictIC50/mol_pIC50sup9_{uniprot_id}.c
sv")
print(f"shape second molecules : {secondMolecules.shape}")
```

```
shape first molecules : (1308, 10)
shape second molecules : (89, 10)
```

```
# On ne garde que les colonnes jugées utiles
firstMolecules = firstMolecules.loc[:,["smiles","pIC50",
"molecular_weight","n_hba","n_hbd","logp"]]

# On ne garde que les colonnes jugées utiles
secondMolecules = secondMolecules.loc[:,["smiles","predicted_pIC50",
"molecular_weight","n_hba","n_hbd","logp"]]
secondMolecules.rename(columns = {"predicted_pIC50" : "pIC50"}, inplace =
True)

# Fusion des jeux de données
molecules = pd.concat([firstMolecules, secondMolecules])
print(f"molecules : {molecules.shape}")
```

```
molecules : (1397, 6)

1397
```

Création du fichier regroupant la formule SMILES et le nom/index

```
pd.DataFrame({'smiles' : molecules.smiles, 'index' :
molecules.index}).to_csv(f"/content/drive/MyDrive/Colab
Notebooks/these_exercice/input/09_predictToxicity/{uniprot_id}_moleculesTox
Study.smi", sep = "\t", header = False, index = False)
```

Prédictions

```
!pip install rdkit

Looking in indexes: https://pypi.org/simple,
https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting rdkit
  Downloading
rdkit-2022.9.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(29.5 MB)
ent already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from rdkit) (1.21.6)
Requirement already satisfied: Pillow in
/usr/local/lib/python3.7/dist-packages (from rdkit) (7.1.2)
Installing collected packages: rdkit
Successfully installed rdkit-2022.9.1

from os import chdir
chdir("/content/drive/MyDrive/Colab Notebooks/these_exercice/src/eToxPred")

!python "etoxpred_predict.py" --datafile "/content/drive/MyDrive/Colab
Notebooks/these_exercice/input/09_predictToxicity/P08581_moleculesToxStudy.
smi" --modelfile "/content/drive/MyDrive/Colab
Notebooks/these_exercice/src/eToxPred/etoxpred_best_model.joblib"
--outputfile "/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/09_predictToxicity/P08581_toxResults.csv"

...loading models
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:338: UserWarning:
Trying to unpickle estimator ExtraTreeClassifier from version 0.23.2 when
using version 1.0.2. This might lead to breaking code or invalid results.
Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-mai
ntainability-limitations
  UserWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:338: UserWarning:
Trying to unpickle estimator ExtraTreesClassifier from version 0.23.2 when
using version 1.0.2. This might lead to breaking code or invalid results.
Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-mai
ntainability-limitations
  UserWarning,
...starts prediction
...prediction done!
```

Chargement des prédictions et fusion du dataframe initial et prédictions

```
toxPrediction = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/these_exercice/output/09_predictToxicity/P08581_toxResults.csv")
toxPrediction.drop(["name"], axis = 1, inplace = True)
```

```
print(f"toxPrediction : {toxPrediction.shape}")
```

```
toxPrediction : (1397, 3)
```

Restriction aux molécules ayant la plus forte probabilité d'être non toxiques

D'après l'étude de Pu et al., une valeur de seuil de Tox-pred de 0.58 est la valeur la plus efficace pour disciminer les molécules toxiques des non-toxiques.

```
nonToxicMolecules = molecules[molecules.loc[:,"Tox-score"] < 0.58]
print(nonToxicMolecules.shape)
```

```
(1344, 8)
```

```
# Affichage des molécules

mols_protein = [Chem.MolFromSmiles(smile) for smile in
nonToxicMolecules.smiles]
pIC50_protein = nonToxicMolecules["pIC50"].tolist()
pIC50_values = [(f"pIC50 value: {value:.2f}") for value in pIC50_protein]

Draw.MolsToGridImage(mols_protein[0:10], molsPerRow=3, subImgSize=(450,
300), legends=pIC50_values[0:10])
```

Bibliographie

1. Perrone F, Di Liello R, Gargiulo P, Arenare L, Guizzaro L, Chiodini P, et al. The opportunity of patient-journey studies for academic clinical research in oncology. BMJ Open 2021;11:e052871.
2. Cirillo D, Valencia A. Big data analytics for personalized medicine. Curr. Opin. Biotechnol. 2019;58:161-7.
3. Suwinski P, Ong C, Ling MHT, Poh YM, Khan AM, Ong HS. Advancing Personalized Medicine Through the Application of Whole Exome Sequencing and Big Data Analytics. Front. Genet. 2019
4. Berisha B, Mëziu E, Shabani I. Big data analytics in Cloud computing: an overview. J. Cloud Comput. 2022;11:24.
5. Yang C, Huang Q, Li Z, Liu K, Hu F. Big Data and cloud computing: innovation opportunities and challenges. Int. J. Digit. Earth 2017;10:13-53.
6. Gupta R, Srivastava D, Sahu M, Tiwari S, Ambasta RK, Kumar P. Artificial intelligence to deep learning: machine intelligence approach for drug discovery. Mol. Divers. 2021;25:1315-60.
7. Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016.
8. Shinde PP, Shah S. A Review of Machine Learning and Deep Learning Applications. In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). 2018. page 1-6.
9. Hastie T, Tibshirani R, Friedman J. Overview of Supervised Learning [Internet]. In: Hastie T, Tibshirani R, Friedman J, éditeurs. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York, NY: Springer; 2009 [cité 2022 août 30]. page 9-41.
10. Zou H, Hastie T. Regularization and Variable Selection via the Elastic Net. J. R. Stat. Soc. Ser. B Stat. Methodol. 2005;67:301-20.
11. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. 1958;65:386-408.
12. Chan HP, Samala RK, Hadjiiski LM, Zhou C. Deep Learning in Medical Image Analysis. Adv. Exp. Med. Biol. 2020;1213:3-21.
13. Montesinos-López OA, Montesinos-López A, Pérez-Rodríguez P, Barrón-López JA, Martini JWR, Fajardo-Flores SB, et al. A review of deep learning applications for genomic selection. BMC Genomics 2021;22:19.
14. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al.

Highly accurate protein structure prediction with AlphaFold. Nature 2021;596:583-9.
15. Karimi M, Wu D, Wang Z, Shen Y. DeepAffinity: interpretable deep learning of compound-protein affinity through unified recurrent and convolutional neural networks. Bioinforma. Oxf. Engl. 2019;35:3329-38.
16. Jisna VA, Jayaraj PB. Protein Structure Prediction: Conventional and Deep Learning Perspectives. Protein J. 2021;40:522-44.
17. Yu L, Cheng M, Qiu W, Xiao X, Lin W. idse-HE: Hybrid embedding graph neural network for drug side effects prediction. J. Biomed. Inform. 2022;131:104098.
18. Han Y, Karunasekera S, Leckie C. Graph Neural Networks with Continual Learning for Fake News Detection from Social Media. 2020
19. Torng W, Altman RB. Graph Convolutional Neural Networks for Predicting Drug-Target Interactions. J. Chem. Inf. Model. 2019;59:4131-49.
20. Chen M, Shi X, Zhang Y, Wu D, Guizani M. Deep Feature Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network. IEEE Trans. Big Data 2021;7:750-8.
21. Chi W, Deng M. Sparsity-Penalized Stacked Denoising Autoencoders for Imputing Single-Cell RNA-Seq Data. Genes 2020;11.
22. Emdadi A, Eslahchi C. Auto-HMM-LMF: feature selection based method for prediction of drug response via autoencoder and hidden Markov model. BMC Bioinformatics 2021;22:33.
23. Khayatian F, Nagy Z, Bollinger A. Using generative adversarial networks to evaluate robustness of reinforcement learning agents against uncertainties. Energy Build. 2021;251:111334.
24. Elgammal A, Liu B, Elhoseiny M, Mazzone M. CAN: Creative Adversarial Networks, Generating « Art » by Learning About Styles and Deviating from Style Norms. 2017
25. Rennane S, Baker L, Mulcahy A. Estimating the Cost of Industry Investment in Drug Research and Development: A Review of Methods and Results. Inq.- J. Health Care Organ. Provis. Financ. 2021;58:00469580211059731.
26. Scott TJ, O'Connor AC, Link AN, Beaulieu TJ. Economic analysis of opportunities to accelerate Alzheimer's disease research and development. Ann. N. Y. Acad. Sci. 2014;1313:17-34.
27. DiMasi JA, Grabowski HG, Hansen RW. Innovation in the pharmaceutical industry: New estimates of R&D costs. J. Health Econ. 2016;47:20-33.
28. Koromina M, Pandi MT, Patrinos GP. Rethinking Drug Repositioning and

Development with Artificial Intelligence, Machine Learning, and Omics. OMICS J. Integr. Biol. 2019;23:539-48.
29. Dierynck B, Joos P. Research and Development Costs of New Drugs. Jama-J. Am. Med. Assoc. 2020;324:516-+.
30. Álvarez-Machancoses Ó, Fernández-Martínez JL. Using artificial intelligence methods to speed up drug discovery. Expert Opin. Drug Discov. 2019;14:769-77.
31. Fleming N. How artificial intelligence is changing drug discovery. Nature 2018;557:S55-S55.
32. Yang X, Wang Y, Byrne R, Schneider G, Yang S. Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery. Chem. Rev. 2019;119:10520-94.
33. Zoete V, Daina A, Bovigny C, Michielin O. SwissSimilarity: A Web Tool for Low to Ultra High Throughput Ligand-Based Virtual Screening. J. Chem. Inf. Model. 2016;56:1399-404.
34. Imbernón B, Cecilia JM, Pérez-Sánchez H, Giménez D. METADOCK: A parallel metaheuristic schema for virtual screening methods. Int. J. High Perform. Comput. Appl. 2018;32:789-803.
35. Riniker S, Landrum GA. Open-source platform to benchmark fingerprints for ligand-based virtual screening. J. Cheminformatics 2013;5:26.
36. Li H, Leung KS, Wong MH, Ballester PJ. USR-VS: a web server for large-scale prospective virtual screening using ultrafast shape recognition techniques. Nucleic Acids Res. 2016;44:W436-41.
37. Suzuki SD, Ohue M, Akiyama Y. PKRank: a novel learning-to-rank method for ligand-based virtual screening using pairwise kernel and RankSVM. Artif. Life Robot. 2018;23:205-12.
38. Patel H, Brinkjost T, Koch O. PyGOLD: a python based API for docking based virtual screening workflow generation. Bioinformatics 2017;33:2589-90.
39. Banegas-Luna AJ, Cerón-Carrasco JP, Puertas-Martín S, Pérez-Sánchez H. BRUSELAS: HPC Generic and Customizable Software Architecture for 3D Ligand-Based Virtual Screening of Large Molecular Databases. J. Chem. Inf. Model. 2019;59:2805-17.
40. Wang L, Pang X, Li Y, Zhang Z, Tan W. RADER: a RApid DEcoy Retriever to facilitate decoy based assessment of virtual screening. Bioinformatics 2017;33:1235-7.
41. Mochizuki M, Suzuki SD, Yanagisawa K, Ohue M, Akiyama Y. QEX:

target-specific druglikeness filter enhances ligand-based virtual screening. Mol. Divers. 2019;23:11-8.

42. Zhang H, Liao L, Cai Y, Hu Y, Wang H. IVS2vec: A tool of Inverse Virtual Screening based on word2vec and deep learning techniques. Methods 2019;166:57-65.
43. Arcon JP, Modenutti CP, Avendaño D, Lopez ED, Defelipe LA, Ambrosio FA, et al. AutoDock Bias: improving binding mode prediction and virtual screening using known protein–ligand interactions. Bioinformatics 2019;35:3836-8.
44. Ebejer JP, Finn PW, Wong WK, Deane CM, Morris GM. Ligity: A Non-Superpositional, Knowledge-Based Approach to Virtual Screening. J. Chem. Inf. Model. 2019;59:2600-16.
45. Zhu Z, Wang X, Yang Y, Zhang X, Mu K, Shi Y, et al. D3Similarity: A Ligand-Based Approach for Predicting Drug Targets and for Virtual Screening of Active Compounds Against COVID-19. 2021
46. Liu Z, Du J, Fang J, Yin Y, Xu G, Xie L. DeepScreening: a deep learning-based screening web server for accelerating drug discovery. Database 2019;2019:baz104.
47. Soufan O, Ba-alawi W, Magana-Mora A, Essack M, Bajic VB. DPubChem: a web tool for QSAR modeling and high-throughput virtual screening. Sci. Rep. 2018;8:9110.
48. Chi CT, Lee MH, Weng CF, Leong MK. In Silico Prediction of PAMPA Effective Permeability Using a Two-QSAR Approach. Int. J. Mol. Sci. 2019;20:3170.
49. Xiao T, Qi X, Chen Y, Jiang Y. Development of Ligand-based Big Data Deep Neural Network Models for Virtual Screening of Large Compound Libraries. Mol. Inform. 2018;37:1800031.
50. Choudhary S, Malik YS, Tomar S. Identification of SARS-CoV-2 Cell Entry Inhibitors by Drug Repurposing Using in silico Structure-Based Virtual Screening Approach. Front. Immunol. 2020
51. Amin SkA, Ghosh K, Gayen S, Jha T. Chemical-informatics approach to COVID-19 drug discovery: Monte Carlo based QSAR, virtual screening and molecular docking study of some in-house molecules as papain-like protease (PLpro) inhibitors. J. Biomol. Struct. Dyn. 2021;39:4764-73.
52. Arul Murugan N, Ruba Priya G, Narahari Sastry G, Markidis S. Artificial intelligence in virtual screening: Models versus experiments. Drug Discov. Today 2022;27:1913-23.
53. Bintener T, Pacheco MP, Sauter T. Towards the routine use of in silico

screenings for drug discovery using metabolic modelling. Biochem. Soc. Trans. 2020;48:955-69.

54. Labbé CM, Rey J, Lagorce D, Vavruša M, Becot J, Sperandio O, et al. MTiOpenScreen: a web server for structure-based virtual screening. Nucleic Acids Res. 2015;43:W448-54.
55. Schellhammer I, Rarey M. FlexX-Scan: Fast, structure-based virtual screening. Proteins Struct. Funct. Bioinforma. 2004;57:504-17.
56. Perez-Castillo Y, Sotomayor-Burneo S, Jimenes-Vargas K, Gonzalez-Rodriguez M, Cruz-Monteagudo M, Armijos-Jaramillo V, et al. CompScore: Boosting Structure-Based Virtual Screening Performance by Incorporating Docking Scoring Function Components into Consensus Scoring. J. Chem. Inf. Model. 2019;59:3655-66.
57. Skalic M, Martínez-Rosell G, Jiménez J, De Fabritiis G. PlayMolecule BindScope: large scale CNN-based virtual screening on the web. Bioinformatics 2019;35:1237-8.
58. Fang Y, Ding Y, Feinstein WP, Koppelman DM, Moreno J, Jarrell M, et al. GeauxDock: Accelerating Structure-Based Virtual Screening with Heterogeneous Computing. PLOS ONE 2016;11:e0158898.
59. Pires DEV, Veloso WNP, Myung Y, Rodrigues CHM, Silk M, Rezende PM, et al. EasyVS: a user-friendly web-based tool for molecule library selection and structure-based virtual screening. Bioinformatics 2020;36:4200-2.
60. Ibrahim TM, Bauer MR, Boeckler FM. Applying DEKOIS 2.0 in structure-based virtual screening to probe the impact of preparation procedures and score normalization. J. Cheminformatics 2015;7:21.
61. Shin WH, Christoffer CW, Wang J, Kihara D. PL-PatchSurfer2: Improved Local Surface Matching-Based Virtual Screening Method That Is Tolerant to Target and Ligand Structure Variation. J. Chem. Inf. Model. 2016;56:1676-91.
62. Litfin T, Zhou Y, Yang Y. SPOT-ligand 2: improving structure-based virtual screening by binding-homology search on an expanded structural template library. Bioinformatics 2017;33:1238-40.
63. Ropp PJ, Spiegel JO, Walker JL, Green H, Morales GA, Milliken KA, et al. Gypsum-DL: an open-source program for preparing small-molecule libraries for structure-based virtual screening. J. Cheminformatics 2019;11:34.
64. Akbar R, Jusoh SA, Amaro RE, Helms V. ENRI: A tool for selecting structure-based virtual screening target conformations. Chem. Biol. Drug Des. 2017;89:762-71.

65. Wallach I, Dzamba M, Heifets A. AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery [Internet]. 2015 [cité 2022 oct 22];Available from: http://arxiv.org/abs/1510.02855
66. Bao J, He X, Zhang JZH. DeepBSP—a Machine Learning Method for Accurate Prediction of Protein–Ligand Docking Structures. J. Chem. Inf. Model. 2021;61:2231-40.
67. Adeshina YO, Deeds EJ, Karanicolas J. Machine learning classification can reduce false positives in structure-based virtual screening. Proc. Natl. Acad. Sci. 2020;117:18477-88.
68. Stepniewska-Dziubinska MM, Zielenkiewicz P, Siedlecki P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. Bioinformatics 2018;34:3666-74.
69. Zheng L, Fan J, Mu Y. OnionNet: a Multiple-Layer Intermolecular-Contact-Based Convolutional Neural Network for Protein–Ligand Binding Affinity Prediction. ACS Omega 2019;4:15956-65.
70. Ballester PJ, Mitchell JBO. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. Bioinformatics 2010;26:1169-75.
71. Ashtawy HM, Mahapatra NR. Machine-learning scoring functions for identifying native poses of ligands docked to known and novel proteins. BMC Bioinformatics 2015;16:S3.
72. Durrant JD, McCammon JA. NNScore 2.0: A Neural-Network Receptor–Ligand Scoring Function. J. Chem. Inf. Model. 2011;51:2897-903.
73. Wang C, Zhang Y. Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. J. Comput. Chem. 2017;38:169-77.
74. Wang Y, Dou X, Jiang L, Jin H, Zhang L, Zhang L, et al. Discovery of novel glycogen synthase kinase-3α inhibitors: Structure-based virtual screening, preliminary SAR and biological evaluation for treatment of acute myeloid leukemia. Eur. J. Med. Chem. 2019;171:221-34.
75. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery. Drug Discov. Today 2018;23:1241-50.
76. Schneider P, Walters WP, Plowright AT, Sieroka N, Listgarten J, Goodnow RA, et al. Rethinking drug design in the artificial intelligence era. Nat. Rev. Drug Discov. 2020;19:353-64.
77. Hessler G, Baringhaus KH. Artificial Intelligence in Drug Design. Mol. Basel

Switz. 2018;23:E2520.

78. Paul D, Sanap G, Shenoy S, Kalyane D, Kalia K, Tekade RK. Artificial intelligence in drug discovery and development. Drug Discov. Today 2021;26:80-93.
79. Kumar R, Sharma A, Siddiqui MH, Tiwari RK. Prediction of Human Intestinal Absorption of Compounds Using Artificial Intelligence Techniques. Curr. Drug Discov. Technol. 14:244-54.
80. Tetko IV, Tanchuk VY. Application of associative neural networks for prediction of lipophilicity in ALOGPS 2.1 program. J. Chem. Inf. Comput. Sci. 2002;42:1136-45.
81. Tetko IV, Gasteiger J, Todeschini R, Mauri A, Livingstone D, Ertl P, et al. Virtual Computational Chemistry Laboratory – Design and Description. J. Comput. Aided Mol. Des. 2005;19:453-63.
82. ChemSpider | Search and share chemistry [Internet]. [cité 2022 oct 18] ; Available from: http://www.chemspider.com/
83. Kucukdereli H, Allen NJ, Lee AT, Feng A, Ozlu MI, Conatser LM, et al. Control of excitatory CNS synaptogenesis by astrocyte-secreted proteins Hevin and SPARC. Proc. Natl. Acad. Sci. 2011;108:E440-9.
84. Ayati A, Falahati M, Irannejad H, Emami S. Synthesis, in vitro antifungal evaluation and in silico study of 3-azolyl-4-chromanone phenylhydrazones. DARU J. Pharm. Sci. 2012;20:46.
85. Wenzel J, Matter H, Schmidt F. Predictive Multitask Deep Neural Network Models for ADME-Tox Properties: Learning from Large Data Sets. J. Chem. Inf. Model. 2019;59:1253-68.
86. Nascimento ACA, Prudêncio RBC, Costa IG. A multiple kernel learning algorithm for drug-target interaction prediction. BMC Bioinformatics 2016;17:46.
87. He T, Heidemeyer M, Ban F, Cherkasov A, Ester M. SimBoost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. J. Cheminformatics 2017;9:24.
88. Öztürk H, Özgür A, Ozkirimli E. DeepDTA: deep drug–target binding affinity prediction. Bioinformatics 2018;34:i821-9.
89. Feng Q, Dueva E, Cherkasov A, Ester M. PADME: A Deep Learning-based Framework for Drug-Target Interaction Prediction [Internet]. 2019 [cité 2022 oct 22];Available from: http://arxiv.org/abs/1807.09741
90. Matlock MK, Hughes TB, Swamidass SJ. XenoSite server: a web-available site of metabolism prediction tool. Bioinforma. Oxf. Engl. 2015;31:1136-7.

91. Šícho M, Stork C, Mazzolari A, de Bruyn Kops C, Pedretti A, Testa B, et al. FAME 3: Predicting the Sites of Metabolism in Synthetic Compounds and Natural Products for Phase 1 and Phase 2 Metabolic Enzymes. J. Chem. Inf. Model. 2019;59:3400-12.
92. Beck BR, Shin B, Choi Y, Park S, Kang K. Predicting commercially available antiviral drugs that may act on the novel coronavirus (SARS-CoV-2) through a drug-target interaction deep learning model. Comput. Struct. Biotechnol. J. 2020;18:784-90.
93. Cañada A, Capella-Gutierrez S, Rabal O, Oyarzabal J, Valencia A, Krallinger M. LimTox: a web tool for applied text mining of adverse event and toxicity associations of compounds, drugs and genes. Nucleic Acids Res. 2017;45:W484-9.
94. Pires DEV, Blundell TL, Ascher DB. pkCSM: Predicting Small-Molecule Pharmacokinetic and Toxicity Properties Using Graph-Based Signatures. J. Med. Chem. 2015;58:4066-72.
95. Yang H, Lou C, Sun L, Li J, Cai Y, Wang Z, et al. admetSAR 2.0: web-service for prediction and optimization of chemical ADMET properties. Bioinformatics 2019;35:1067-9.
96. Patlewicz G, Jeliazkova N, Safford RJ, Worth AP, Aleksiev B. An evaluation of the implementation of the Cramer classification scheme in the Toxtree software. SAR QSAR Environ. Res. 2008;19:495-524.
97. Wang Z, Liang L, Yin Z, Lin J. Improving chemical similarity ensemble approach in target prediction. J. Cheminformatics 2016;8:20.
98. Pu L, Naderi M, Liu T, Wu HC, Mukhopadhyay S, Brylinski M. eToxPred: a machine learning-based approach to estimate the toxicity of drug candidates. BMC Pharmacol. Toxicol. 2019;20:2.
99. Lysenko A, Sharma A, Boroevich KA, Tsunoda T. An integrative machine learning approach for prediction of toxicity-related drug safety. Life Sci. Alliance. 2018
100. Mayr A, Klambauer G, Unterthiner T, Hochreiter S. DeepTox: Toxicity Prediction using Deep Learning. Front. Environ. Sci. 2016
101. Gayvert KM, Madhukar NS, Elemento O. A Data-Driven Approach to Predicting Successes and Failures of Clinical Trials. Cell Chem. Biol. 2016;23:1294-301.
102. Jimenez-Carretero D, Abrishami V, Fernández-de-Manuel L, Palacios I, Quílez-Álvarez A, Díez-Sánchez A, et al. Tox_(R)CNN: Deep learning-based nuclei profiling tool for drug toxicity screening. PLoS Comput. Biol.

2018;14:e1006238.
103. Goh GB, Hodas NO, Siegel C, Vishnu A. SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties. 2018
104. Goh GB, Siegel C, Vishnu A, Hodas NO, Baker N. Chemception: A Deep Neural Network with Minimal Chemistry Knowledge Matches the Performance of Expert-developed QSAR/QSPR Models. 2017
105. Preuer K, Lewis RPI, Hochreiter S, Bender A, Bulusu KC, Klambauer G. DeepSynergy: predicting anti-cancer drug synergy with Deep Learning. Bioinformatics 2018;34:1538-46.
106. Xu Y, Pei J, Lai L. Deep Learning Based Regression and Multiclass Models for Acute Oral Toxicity Prediction with Automatic Chemical Feature Extraction. J. Chem. Inf. Model. 2017;57:2672-85.
107. Srivastava A, Siddiqui S, Ahmad R, Mehrotra S, Ahmad B, Srivastava AN. Exploring nature's bounty: identification of Withania somnifera as a promising source of therapeutic agents against COVID-19 by virtual screening and in silico evaluation. J. Biomol. Struct. Dyn. 2022;40:1858-908.
108. Zhang C, Cheng F, Li W, Liu G, Lee PW, Tang Y. In silico Prediction of Drug Induced Liver Toxicity Using Substructure Pattern Recognition Method. Mol. Inform. 2016;35:136-44.
109. Martínez V, Navarro C, Cano C, Fajardo W, Blanco A. DrugNet: Network-based drug–disease prioritization by integrating heterogeneous data. Artif. Intell. Med. 2015;63:41-9.
110. Zhang W, Xu H, Li X, Gao Q, Wang L. DRIMC: an improved drug repositioning approach using Bayesian inductive matrix completion. Bioinformatics 2020;36:2839-47.
111. Luo H, Zhang P, Cao XH, Du D, Ye H, Huang H, et al. DPDR-CPI, a server that predicts Drug Positioning and Drug Repositioning via Chemical-Protein Interactome. Sci. Rep. 2016;6:35996.
112. Whirl-Carrillo M, McDonagh EM, Hebert JM, Gong L, Sangkuhl K, Thorn CF, et al. Pharmacogenomics knowledge for personalized medicine. Clin. Pharmacol. Ther. 2012;92:414-7.
113. Whirl-Carrillo M, Huddart R, Gong L, Sangkuhl K, Thorn CF, Whaley R, et al. An Evidence-Based Framework for Evaluating Pharmacogenomics Knowledge for Personalized Medicine. Clin. Pharmacol. Ther. 2021;110:563-72.
114. Gallo K, Goede A, Eckert A, Moahamed B, Preissner R, Gohlke BO.

PROMISCUOUS 2.0: a resource for drug-repositioning. Nucleic Acids Res. 2021;49:D1373-80.

115. Luo H, Li M, Wang S, Liu Q, Li Y, Wang J. Computational drug repositioning using low-rank matrix approximation and randomized algorithms. Bioinformatics 2018;34:1904-12.
116. Yella JK, Jegga AG. MGATRx: Discovering Drug Repositioning Candidates Using Multi-view Graph Attention. 2020.06.29.171876.
117. Yan CK, Wang WX, Zhang G, Wang JL, Patel A. BiRWDDA: A Novel Drug Repositioning Method Based on Multisimilarity Fusion. J. Comput. Biol. 2019;26:1230-42.
118. Fahimian G, Zahiri J, Arab SS, Sajedi RH. RepCOOL: Computational Drug Repositioning Via Integrating Heterogeneous Biological Networks. 2019;817882.
119. Zheng X, He S, Song X, Zhang Z, Bo X. DTI-RCNN: New Efficient Hybrid Neural Network Model to Predict Drug–Target Interactions. In: Kůrková V, Manolopoulos Y, Hammer B, Iliadis L, Maglogiannis I, éditeurs. Artificial Neural Networks and Machine Learning – ICANN 2018. Cham: Springer International Publishing; 2018. page 104-14.
120. Jarada TN, Rokne JG, Alhajj R. SNF–CVAE: Computational method to predict drug–disease interactions using similarity network fusion and collective variational autoencoder. Knowl.-Based Syst. 2021;212:106585.
121. Xu R, Wang Q. PhenoPredict: A disease phenome-wide drug repositioning approach towards schizophrenia drug discovery. J. Biomed. Inform. 2015;56:348-55.
122. Wu Z, Cheng F, Li J, Li W, Liu G, Tang Y. SDTNBI: an integrated network and chemoinformatics tool for systematic prediction of drug–target interactions and drug repositioning. Brief. Bioinform. 2017;18:333-47.
123. Chen H, Cheng F, Li J. iDrug: Integration of drug repositioning and drug-target prediction via cross-network embedding. PLOS Comput. Biol. 2020;16:e1008040.
124. Wang Z, Zhou M, Arnold C. Toward heterogeneous information fusion: bipartite graph convolutional networks for in silico drug repurposing. Bioinformatics 2020;36:i525-33.
125. Zeng X, Zhu S, Lu W, Liu Z, Huang J, Zhou Y, et al. Target identification among known drugs by deep learning from heterogeneous networks. Chem. Sci. 2020;11:1775-97.

126. Liu H, Zhang W, Song Y, Deng L, Zhou S. HNet-DNN: Inferring New Drug–Disease Associations with Deep Neural Network Based on Heterogeneous Network Features. J. Chem. Inf. Model. 2020;60:2367-76.
127. Jiang HJ, You ZH, Huang YA. Predicting drug−disease associations via sigmoid kernel-based convolutional neural networks. J. Transl. Med. 2019;17.
128. Xuan P, Ye Y, Zhang T, Zhao L, Sun C. Convolutional Neural Network and Bidirectional Long Short-Term Memory-Based Method for Predicting Drug-Disease Associations. Cells 2019;8:E705.
129. Jiang HJ, Huang YA, You ZH. SAEROF: an ensemble approach for large-scale drug-disease association prediction by incorporating rotation forest and sparse autoencoder deep neural network. Sci. Rep. 2020;10:4972.
130. Zeng X, Zhu S, Liu X, Zhou Y, Nussinov R, Cheng F. deepDR: a network-based deep learning approach to in silico drug repositioning. Bioinforma. Oxf. Engl. 2019;35:5191-8.
131. Huang K, Xiao C, Glass LM, Zitnik M, Sun J. SkipGNN: predicting molecular interactions with skip-graph networks. Sci. Rep. 2020;10:21092.
132. Vijayan RSK, Kihlberg J, Cross JB, Poongavanam V. Enhancing preclinical drug discovery with artificial intelligence. Drug Discov. Today 2022;27:967-84.
133. Cramer P. AlphaFold2 and the future of structural biology. Nat. Struct. Mol. Biol. 2021;28:704-5.
134. AlQuraishi M. End-to-End Differentiable Learning of Protein Structure. Cell Syst. 2019;8:292-301.e3.
135. Avdagic Z, Purisevic E, Omanovic S, Coralic Z. Artificial Intelligence in Prediction of Secondary Protein Structure Using CB513 Database. Summit Transl. Bioinforma. 2009;2009:1-5.
136. Du X, Sun S, Hu C, Yao Y, Yan Y, Zhang Y. DeepPPI: Boosting Prediction of Protein–Protein Interactions with Deep Neural Networks. J. Chem. Inf. Model. 2017;57:1499-510.
137. Cunningham JM, Koytiger G, Sorger PK, AlQuraishi M. Biophysical prediction of protein–peptide interactions and signaling networks using machine learning. Nat. Methods 2020;17:175-83.
138. Chatterjee P, Basu S, Kundu M, Nasipuri M, Plewczynski D. PPI_SVM: Prediction of protein-protein interactions using machine learning, domain-domain affinities and frequency tables. Cell. Mol. Biol. Lett. 2011;16:264-78.
139. Lu L, Lu H, Skolnick J. MULTIPROSPECTOR: An algorithm for the prediction of

protein–protein interactions by multimeric threading. Proteins Struct. Funct. Bioinforma. 2002;49:350-64.

140. Singh R, Park D, Xu J, Hosur R, Berger B. Struct2Net: a web service to predict protein–protein interactions using a structure-based approach. Nucleic Acids Res. 2010;38:W508-15.
141. Antolin AA, Workman P, Mestres J, Al-Lazikani B. Polypharmacology in Precision Oncology: Current Applications and Future Prospects. Curr. Pharm. Des. 2016;22:6935-45.
142. Peters JU. Polypharmacology - foe or friend? J. Med. Chem. 2013;56:8955-71.
143. Awale M, Reymond JL. The polypharmacology browser: a web-based multi-fingerprint target prediction tool using ChEMBL bioactivity data. J. Cheminformatics 2017;9:11.
144. Liu X, Gao Y, Peng J, Xu Y, Wang Y, Zhou N, et al. TarPred: a web application for predicting therapeutic and side effect targets of chemical compounds. Bioinformatics 2015;31:2049-51.
145. Reker D, Rodrigues T, Schneider P, Schneider G. Identifying the macromolecular targets of de novo-designed chemical entities through self-organizing map consensus. Proc. Natl. Acad. Sci. 2014;111:4067-72.
146. Wang L, Ma C, Wipf P, Liu H, Su W, Xie XQ. TargetHunter: An In Silico Target Identification Tool for Predicting Therapeutic Potential of Small Organic Molecules Based on Chemogenomic Database. AAPS J. 2013;15:395-406.
147. Wang X, Pan C, Gong J, Liu X, Li H. Enhancing the Enrichment of Pharmacophore-Based Target Prediction for the Polypharmacological Profiles of Drugs. J. Chem. Inf. Model. 2016;56:1175-83.
148. Gong J, Cai C, Liu X, Ku X, Jiang H, Gao D, et al. ChemMapper: a versatile web server for exploring pharmacology and chemical structure association based on molecular 3D similarity method. Bioinformatics 2013;29:1827-9.
149. Gfeller D, Grosdidier A, Wirth M, Daina A, Michielin O, Zoete V. SwissTargetPrediction: a web server for target prediction of bioactive small molecules. Nucleic Acids Res. 2014;42:W32-8.
150. Li Z, Li X, Liu X, Fu Z, Xiong Z, Wu X, et al. KinomeX: a web application for predicting kinome-wide polypharmacology effect of small molecules. Bioinformatics 2019;35:5354-6.
151. Yang M, Simm J, Lam CC, Zakeri P, van Westen GJP, Moreau Y, et al. Linking drug target and pathway activation for effective therapy using multi-task learning. Sci. Rep. 2018;8:8322.

152. Ozhathil LC, Delalande C, Bianchi B, Nemeth G, Kappel S, Thomet U, et al. Identification of potent and selective small molecule inhibitors of the cation channel TRPM4. Br. J. Pharmacol. 2018;175:2504-19.
153. Poirier M, Awale M, Roelli MA, Giuffredi GT, Ruddigkeit L, Evensen L, et al. Identifying Lysophosphatidic Acid Acyltransferase β (LPAAT-β) as the Target of a Nanomolar Angiogenesis Inhibitor from a Phenotypic Screen Using the Polypharmacology Browser PPB2. ChemMedChem 2019;14:224-36.
154. Carvalho-Silva D, Pierleoni A, Pignatelli M, Ong C, Fumis L, Karamanis N, et al. Open Targets Platform: new developments and updates two years on. Nucleic Acids Res. 2019;47:D1056-65.
155. Zhavoronkov A, Ivanenkov YA, Aliper A, Veselov MS, Aladinskiy VA, Aladinskaya AV, et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nat. Biotechnol. 2019;37:1038-40.
156. Elbadawi M, Gaisford S, Basit AW. Advanced machine-learning techniques in drug discovery. Drug Discov. Today 2021;26:769-77.
157. McCloskey K, Sigel EA, Kearnes S, Xue L, Tian X, Moccia D, et al. Machine Learning on DNA-Encoded Libraries: A New Paradigm for Hit Finding. J. Med. Chem. 2020;63:8857-66.
158. Putin E, Asadulaev A, Ivanenkov Y, Aladinskiy V, Sanchez-Lengeling B, Aspuru-Guzik A, et al. Reinforced Adversarial Neural Computer for de Novo Molecular Design. J. Chem. Inf. Model. 2018;58:1194-204.
159. Xiong J, Xiong Z, Chen K, Jiang H, Zheng M. Graph neural networks for automated de novo drug design. Drug Discov. Today 2021;26:1382-93.
160. Sung H, Ferlay J, Siegel RL, Laversanne M, Soerjomataram I, Jemal A, et al. Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. CA. Cancer J. Clin. 2021;71:209-49.
161. Davies M, Nowotka M, Papadatos G, Dedman N, Gaulton A, Atkinson F, et al. ChEMBL web services: streamlining access to drug discovery data and utilities. Nucleic Acids Res. 2015;43:W612-20.
162. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, et al. The ChEMBL database in 2017. Nucleic Acids Res. 2017;45:D945-54.
163. Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, et al. PubChem in 2021: new data content and improved web interfaces. Nucleic Acids Res. 2021;49:D1388-95.
164. Brenk R, Schipani A, James D, Krasowski A, Gilbert IH, Frearson J, et al.

Lessons Learnt from Assembling Screening Libraries for Drug Discovery for Neglected Diseases. ChemMedChem 2008;3:435-44.

165. Baell JB, Holloway GA. New Substructure Filters for Removal of Pan Assay Interference Compounds (PAINS) from Screening Libraries and for Their Exclusion in Bioassays. J. Med. Chem. 2010;53:2719-40.
166. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2011;12:2825-30.
167. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, et al. DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res. 2006;34:D668-72.
168. Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, et al. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res. 2018;46:D1074-82.
169. Li J, Zheng S, Chen B, Butte AJ, Swamidass SJ, Lu Z. A survey of current trends in computational drug repositioning. Brief. Bioinform. 2016;17:2-12.
170. Jin G, Wong STC. Toward better drug repositioning: prioritizing and integrating existing methods into efficient pipelines. Drug Discov. Today 2014;19:637-44.
171. Kumar R, Harilal S, Gupta SV, Jose J, Thomas Parambi DG, Uddin MdS, et al. Exploring the new horizons of drug repurposing: A vital tool for turning hard work into smart work. Eur. J. Med. Chem. 2019;182:111602.
172. Tanoli Z, Seemab U, Scherer A, Wennerberg K, Tang J, Vähä-Koskela M. Exploration of databases and methods supporting drug repurposing: a comprehensive survey. Brief. Bioinform. 2021;22:1656-78.

Université de Lille
FACULTE DE PHARMACIE DE LILLE
DIPLOME D'ETAT DE DOCTEUR EN PHARMACIE
Année Universitaire 2022/2023

Nom : Pawlak
Prénom : Geoffrey

Titre de la thèse : Intelligence artificielle et machine learning dans les stratégies de drug discovery

Mots-clés : Intelligence artificielle, machine learning, AI, Drug discovery

---

Résumé :

Les évolutions des domaines de l'informatique, des techniques d'analyses en biologie et du traitement de l'information ont augmenté de manière exponentielle la quantité de données générée et leur analyse requiert un traitement informatisé et automatisé. Dans le domaine du drug discovery l'usage de l'intelligence artificielle et du machine learning est devenu routinier et nécessaire à la découverte de potentielles cibles thérapeutiques, de nouveaux mécanismes d'action, de la prédiction d'une toxicité, d'un repositionnement thérapeutique possible ou encore de la prédiction de la structure des protéines.

<u>Membres du jury</u> :

Président :
Monsieur Philippe Chavatte, Professeur des
Universités et Assesseur en charge des Relations Internationales à la Faculté de Pharmacie de Lille

Assesseur(s) :
Madame Delphine Allorge, Professeur des Universités, Praticien Hospitalier et Doyen de la faculté de pharmacie de Lille.
Monsieur Thomas Morgenroth, Maître de Conférences, Faculté de pharmacie de Lille.