Projet Kaggle : Prédire une réponse biologique des molécules à partir de leurs propriétés chimiques
2023-2024

Apprentissage, Intelligence artificielle et Optimisation

Master 2 Bioinformatique
Années 2023/2024

Dépôt Git :
https://github.com/esoufir/Biological_Response_Prediction

Naïma AMMICHE
Thanina CHABANE
Noura NOUALI
Emma SOUFIR

Introduction

Le développement de médicaments et de traitements dépend largement de notre compréhension de la manière dont les molécules interagissent avec les organismes vivants. L'étude de la prédiction des réactions biomoléculaires est cruciale car elle implique des défis importants. Elle permet le développement rapide de médicaments ciblés et d'optimiser l'efficacité des traitements. Dans cette optique, l'apprentissage automatique peut jouer un rôle clé en nous aidant à évaluer et à anticiper la réponse biologique des molécules aux médicaments. Comprendre et prédire ces réponses est essentiel pour la mise au point de traitements médicaux plus efficaces et personnalisés.

Dans le cadre de ce projet, nous avons utilisé un ensemble de données fourni par Kaggle pour prédire si une molécule provoque ou non une réponse biologique à partir de leurs propriétés chimiques.

Matériel et méthodes

Données brutes

Nous avons récupéré les données de réponses biologiques à partir de la compétition Kaggle : "Predicting a Biological Response[1]". Les données sont au format CSV: chaque ligne représente une molécule et la première colonne correspond à la variable à prédire ('Activity') indiquant si la molécule a provoqué une réponse biologique réelle (y = 1) ou non (y = 0). Les colonnes suivantes décrivent les 1777 descripteurs moléculaires (d1 à d1776), qui sont des propriétés calculées permettant de capturer diverses caractéristiques des 3751 molécules. Parmi ces molécules, 1717 n'ont pas provoqué de réponse biologique (y = 0) contre 2034 qui en ont provoqué une (y = 1).

Prétraitement

L'un des principaux défis consiste dans un premier temps à réduire la dimensionnalité des caractéristiques. Le jeu de données ne présentait aucune valeur manquante. Les variables présentant peu de variabilité (variance faible) ont été supprimées de manière à garder uniquement les variables apportant de l'information. Afin d'éviter les redondances au sein des différents modèles, les variables fortement corrélées ($r > 0.9$) ont été supprimées. A l'issue de ces nettoyages, 1533 descripteurs ont été conservés pour l'apprentissage.

La normalisation permet de mettre toutes les caractéristiques sur une échelle commune, pour un traitement des données plus efficace. Ici, nous avons utilisé la normalisation MinMax car elle permet la mise en échelle des données entre 0 et 1. Ainsi, on s'assure que les valeurs des variables se situent dans une plage limitée, ce qui facilite la convergence du modèle. De plus, elle est moins sensible aux valeurs aberrantes par rapport à d'autres méthodes de normalisation.

A l'issue de ce traitement, le jeu de données a été séparé en deux jeux d'apprentissage (70% du jeu de données initial) et de test (30% du jeu de données initial). Le contrôle de l'homogénéité des échantillons a été effectué à l'aide d'une ACP.

Afin de prédire la réponse biologique d'une molécule, plusieurs méthodes de machine learning sont couramment utilisées[2]. Ici, nous en avons testé quelques-unes afin de déterminer lesquelles étaient les plus adaptées à nos données. L'ensemble de nos analyses et de nos modèles ont été réalisés en Python (version 3.10) à l'aide des packages *scikitlearn, numpy* et *pandas* pour l'analyse des données, *keras* et *tensorflow* pour les modèles de deep learning et *matplotlib* pour la visualisation.

L'ensemble des modèles construits ont été entraînés pour différents paramètres tout en surveillant les métriques liées au surapprentissage. Notamment, pour les méthodes de deep learning, du "Early Stopping" sur la valeur de la perte du jeu de données de validation a été mis en place.

Une fois les différents modèles entraînés, différentes métriques d'évaluation ont été calculées afin de pouvoir comparer les performances des différents modèles entre eux (Figure 1).

La Précision mesure le nombre de vrais positifs parmi toutes les prédictions positives du modèle. Le Recall évalue la capacité du modèle à identifier correctement tous les exemples positifs. Le F1-Score est une métrique de l'équilibre entre la précision et le recall. Un F1-score élevé indique un bon équilibre entre la précision et le rappel, ce qui signifie que le modèle est capable de prédire avec précision les instances positives tout en capturant la plupart des vrais positifs dans l'ensemble de données. Enfin, les courbes ROC et PR ont aussi été générées.

$$Precision = \frac{VP}{VP+FP} \qquad Recall = \frac{VP}{VP+FN} \qquad F1\ score = \frac{2*Precision*Recall}{Precision*Recall}$$

$$Log\ Loss(y,p) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$$

Figure 1 : Principales formules des métriques d'évaluation des modèles de classification binaires.

De plus, dans le cadre de la compétition, les probabilités prédites seront évaluées avec la Log Loss décrite dans la Figure 1.

Modèles

Régression logistique

La régression logistique est un algorithme d'apprentissage supervisé couramment utilisé pour la classification. Son rôle principal est de modéliser la relation entre une variable dépendante binaire (ou catégorielle) et un ensemble de variables indépendantes.

Random Forest

Le Random Forest est un algorithme d'apprentissage automatique puissant et polyvalent qui se prête bien à la classification. De plus, il est robuste aux données bruitées et aux valeurs aberrantes, ce qui est un avantage considérable étant donné la complexité des données moléculaires. Le Random Forest est capable de gérer efficacement les variables binaires. Cet algorithme ensembliste combine plusieurs arbres de décision pour obtenir des prédictions plus précises afin de réduire le surapprentissage et d'augmenter la stabilité du modèle. Les limites du Random Forest sont la complexité: en effet, la production d'un grand nombre d'arbres peut rendre le modèle difficile à interpréter, ainsi que la vitesse d'exécution, liée au temps de génération de chaque arbre.

Ainsi, notre modèle de Random Forest a été construit avec 150 estimateurs, c'est-à-dire en construisant une forêt aléatoire contenant 150 arbres. Aucune profondeur maximale n'a été définie. Nous avons choisi le critère de gini comme critère de qualité afin de sélectionner les variables permettant d'effectuer une séparation efficace des données.

Deep Learning

Nous avons ensuite construit plusieurs types de réseaux de neurones. Etant dans un problème de classification binaire, la fonction de perte Binary Cross Entropy pour mesurer la disparité entre les prédictions et les étiquettes réelles a été utilisée. De plus, la dernière couche des réseaux est toujours une couche Dense avec un seul neurone ainsi qu'une fonction d'activation "sigmoid" pour effectuer une classification binaire. Enfin, la fonction d'activation est toujours une fonction "relu". Dans la majorité des cas, l'optimiseur Adam a été employé. Les différents types de réseaux utilisés sont décrits ci-dessous.

Recurrent Neural Network (RNN)

Le RNN est un type de réseau de neurones capables de traiter des données séquentielles ou temporelles en utilisant des informations passées pour influencer les prédictions actuelles. Ce type de réseau partage des paramètres entre les couches, mais rencontrent des problèmes de gradients explosifs et de vanishing gradients qui affectent l'apprentissage. Pour résoudre ces problèmes, on peut réduire la complexité en réduisant le nombre de couches cachées[3].

Le choix d'un RNN provient d'un article de référence dans le domaine, dont les résultats ont mis en évidence une efficacité de cette architecture pour des tâches similaires de prédiction de réponse à des médicaments[2]. Nous avons donc construit un modèle de RNN simple. En effet, celui-ci utilise une unique couche SimpleRNN avec 256 unités cachées. Nous avons ajouté un Flatten pour aplatir les sorties en un vecteur 1D. Le taux d'apprentissage utilisé est de 0.007, afin d'ajuster les poids du modèle pendant l'entraînement.

Multi Layer Perceptron (MLP)

Nous nous sommes par la suite concentrés sur des Feed Forward Neural Networks (FNN). Les MLP sont un type d'architecture de réseaux capables d'apprendre des fonctions complexes et non linéaires grâce à leurs nombreuses couches, ce qui les rend adaptés à un large éventail de tâches, de la classification à la régression.

Lorsqu'il s'agit de choisir un optimiseur pour l'entraînement de réseaux de neurones, deux options courantes sont la Descente de Gradient Stochastique (SGD) et Adam (Adaptive Moment Estimation). SGD est simple et nécessite moins de mémoire, mais il peut converger lentement. En revanche, Adam est adaptable et converge plus rapidement grâce à l'ajustement automatique du taux d'apprentissage, mais consomme plus de mémoire.

Nous avons créé deux modèles MLP similaires utilisant ces différents optimiseurs. Ils sont constitués de plusieurs couches Dense, chacune ayant un nombre spécifique de neurones (1024,960,256,128,1). Dans le but de réduire le surajustement nous avons intercalé des couches de Dropout entre les couches Dense pour réduire le surajustement.

Convolutional Neural Network (CNN)

Les CNN sont des architectures de réseaux très utilisées dans la prédiction de réponse à une molécule[2]. Un CNN va extraire les "features"[4](les caractéristiques) des données en utilisant des filtres de convolution, permettant ainsi de capturer les motifs locaux au sein des données.

L'architecture du CNN utilisé est présentée dans la Figure 2. Le réseau est composé de trois blocs de convolution, chacun constitué d'une couche de convolution avec un nombre croissant de filtres (de taille 2 x 2) par couche, une couche de Max Pooling (2 x 2) et une couche de Dropout (50%) . Le dernier bloc contient également une couche de Batch

Normalization afin d'éviter le surapprentissage. Les valeurs des 66 573 hyperparamètres ainsi que le taux d'apprentissage de 0.005 ont été optimisées avec la méthode de GridSearch.

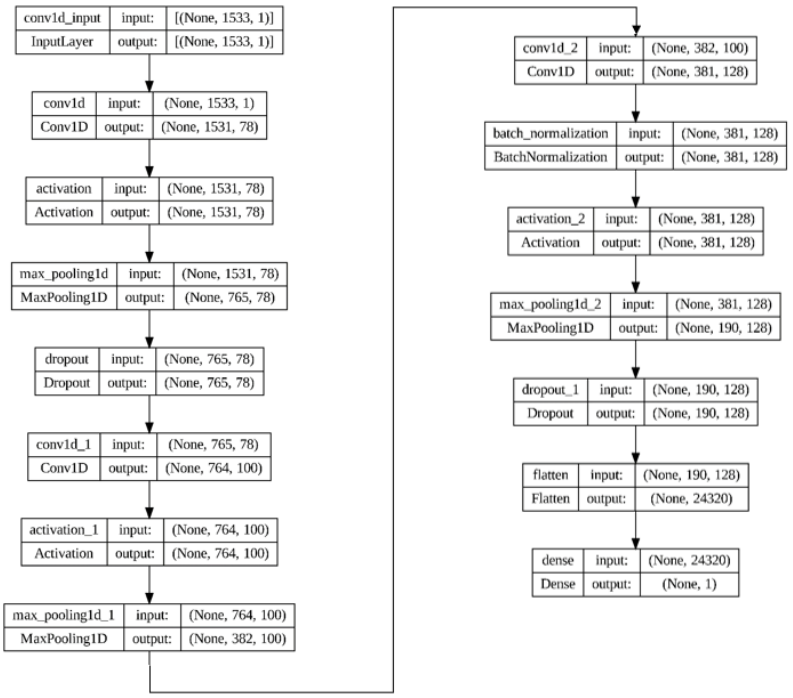

Figure 2 : Représentation du CNN construit.

Le CNN a l'avantage d'avoir une architecture simple. Étant donné qu'il comporte peu de paramètres à ajuster lorsque l'on travaille sur des matrices à deux dimensions, l'apprentissage est d'autant plus rapide.

Multi-Input Neural Network (MINN)

Enfin, sur la base de la proposition d'un participant à la compétition[5] nous avons implémenté un modèle de Multi-Input Neural Network. Ce modèle propose de séparer les variables continues et les variables catégoriques du jeu de données afin de créer deux entrées différentes au réseau de neurones avant de les concaténer.

Ainsi, les variables continues sont traitées à travers une couche d'entrée suivie d'une couche d'Embedding permettant au réseau de trouver les relations entre les variables et de saisir les données significatives. Par la suite, les données sont traitées à travers deux couches de Convolution contenant 256 et 128 filtres respectivement, intercalées par deux couches de Max Pooling, puis une couche de dropout (30%). La transition vers une couche dense contenant 64 neurones s'effectue via une couche permettant d'aplatir les sorties en un vecteur unidimensionnel, suivie d'une seconde douche de Dropout. (50%)

Les variables catégoriques sont traitées à travers deux couches denses suivant l'input, contenant 128 et 64 neurones respectivement, intercalées de couches de Dropout (30% et 50%) afin de limiter le surapprentissage. Par la suite, les deux inputs sont concaténées en une couche de taille 128. Les données passent ensuite par une couche dense de taille 64, une couche de Batch Normalisation permettant de normaliser les différences entre les deux entrées, puis une seconde couche dense.

L'architecture de ce réseau présente l'inconvénient de devoir entraîner plus d'un million de paramètres, le rendant plus complexe qu'une architecture plus classique telle qu'un CNN.

XGBoost

XGBoost est un algorithme d'apprentissage automatique utilisé pour la classification et la régression. Son nom signifie "eXtreme Gradient Boosting", ce qui reflète son utilisation d'arbres de décision boostés pour améliorer la précision des prédictions. L'algorithme fonctionne en construisant une séquence de modèles faibles (généralement des arbres de décision) et en les combinant pour former un modèle fort. Il se distingue par sa capacité à gérer des données complexes, son efficacité en matière de calcul et sa robustesse aux valeurs aberrantes. Pour ajuster les performances de XGBoost, des paramètres importants tels que la profondeur maximale des arbres (max_depth), le taux d'apprentissage (learning_rate), et le nombre d'estimateurs (n_estimators) peuvent être réglés à l'aide de techniques telles que la recherche par grille.

Gradient Boosting

Le Gradient Boosting est une technique d'ensemble en apprentissage automatique qui diffère des réseaux de neurones profonds conçus pour des données complexes et non structurées, telles que des images ou du texte. Au lieu de cela, il se concentre sur la construction d'un modèle robuste à partir de plusieurs modèles plus simples, généralement des arbres de décision peu profonds. L'idée centrale derrière le Gradient Boosting est d'ajuster itérativement ces modèles faibles pour corriger les erreurs commises par les modèles précédents. Cette amélioration continue se fait en minimisant la perte, c'est-à-dire en réduisant les erreurs entre les prédictions du modèle actuel et les vraies valeurs cibles.

Le modèle de Gradient Boosting utilisé comporte 100 estimateurs dans l'ensemble, un taux d'apprentissage de 0.1 et une profondeur maximale des arbres de décision de 3.
Le nombre minimum d'échantillons requis pour effectuer une division à un nœud a été mis à 2 et le nombre minimum d'échantillons requis pour former une feuille de l'arbre a été mis à 1 Le nombre de caractéristiques à considérer lors de la recherche de la meilleure division est déterminé automatiquement par le modèle. La fraction des échantillons utilisés pour l'apprentissage de chaque arbre est de 1.0 (sub_sample).

Modèle final

Enfin, nous avons intégré les résultats des différents modèles étudiés afin de tenter d'améliorer les prédictions. Pour ce faire, pour chaque molécule, nous avons fait la moyenne des probabilités d'appartenance à chaque classe selon chaque modèle. Cette approche a été envisagée dans le but de corriger les erreurs de prédictions faites par certains modèles. En effet, si un modèle prédit mal une molécule alors que les autres modèles la prédisent mieux, alors la prédiction finale sera ajustée. De plus, les modèles avec de meilleures performances, étant considérés comme plus précis, ont été assignés avec plus de poids dans la décision finale. Enfin, à partir des moyennes de probabilités de chaque molécule, l'assignement à chaque classe de prédiction de chaque molécule a été effectué (Figure 3), avec un seuil d'appartenance à la classe 1 à 0.56.

$$Y_{pred\_final} = \frac{\sum_i w_i \times p_i}{\sum_i w_i}$$

Figure 3 : Formule de la prédiction finale en fonction des prédictions des différents modèles. Les *wi* sont les poids donnés au modèle i. Les *pi* sont les probabilités prédites par le modèle i.

Résultats et Discussion

Dans un premier temps, nous avons évalué les performances de chaque modèle en utilisant différentes métriques relatives à la prédiction binaire.

	Random Forest	MLP Adam	MLP SGD	CNN	RNN	MINN	Régression logistique	Gradient boosting	XGBoost
Accuracy	0.82	0.78	0.79	0.79	0.78	0.76	0.76	0.80	0.80
Sensibilité	0.84	0.86	0.84	0.82	0.81	0.79	0.81	0.85	0.84
Spécificité	0.78	0.68	0.74	0.74	0.74	0.75	0.71	0.74	0.74
F1-score	0.83	0.81	0.82	0.81	0.80	0.79	0.79	0.82	0.82

Table 1 : Résumé de performances des différents modèles testés sur le jeu de données test.

Le modèle qui présente la meilleure Accuracy parmi les autres pour notre jeu de données est le Random Forest. Sur notre ensemble de test, nous obtenons une Accuracy de 0.82. La plupart des autres modèles ont des Accuracy comparables (entre 0.76 et 0.80). Les mêmes observations peuvent se faire sur la sensibilité et le F1-score.

Concernant la spécificité (Table 1, Ligne 3), le Random Forest et le Gradient Boosting possèdent une meilleure capacité à minimiser les faux positifs que les autres modèles. Cependant, on observe que, pour l'ensemble de nos modèles, la spécificité est moins robuste que la sensibilité, ce qui signifie que les modèles ont une meilleure capacité à repérer les vrais cas positifs que les vrais cas négatifs, ce qui peut entraîner davantage de faux positifs par rapport aux faux négatifs. Dans le contexte du projet, prédire un faux positif, c'est-à-dire une molécule déclenchant une réponse alors qu'elle n'en engendre pas est plus problématique que de prédire une molécule comme n'engendrant pas de réponse. En effet, il serait plus raisonnable de ne pas faire de recherche sur des molécules mal prédites car elles généreront une perte de temps et d'argent.

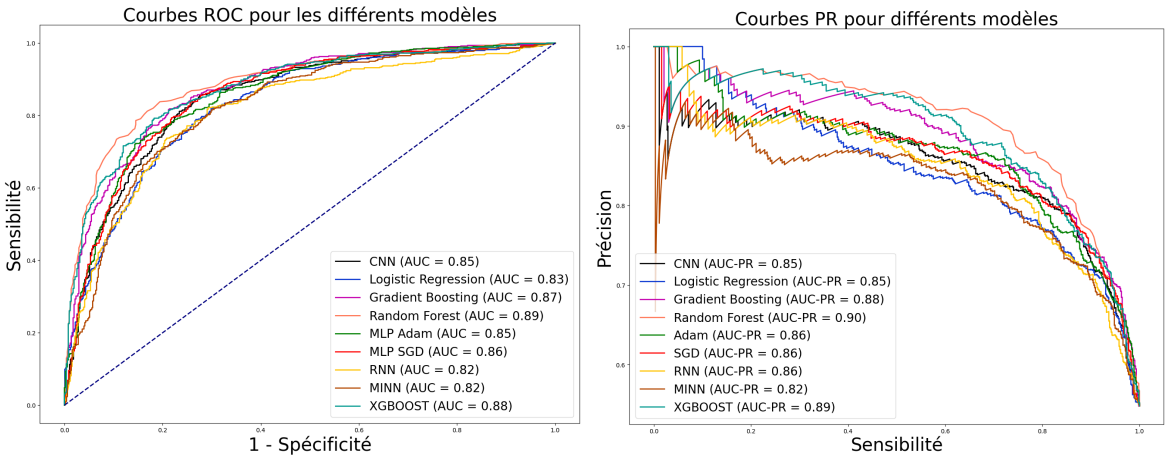

Figure 3: Courbes ROC et PR de différents modèles créés.

L'aire sous la courbe ROC est comprise entre 0.89 et 0.82, ce qui indique une bonne capacité de nos modèles à discriminer les classes. Les meilleures performances ont été observées avec le Random Forest, le XGBoost et le Gradient Boosting. Pour le reste des modèles, les AUC sont comparables entre elles. Le MINN présente de moins bonnes

performances pour la prédiction de la réponse biologique. Les mêmes déductions peuvent se faire sur la PR-Curve.

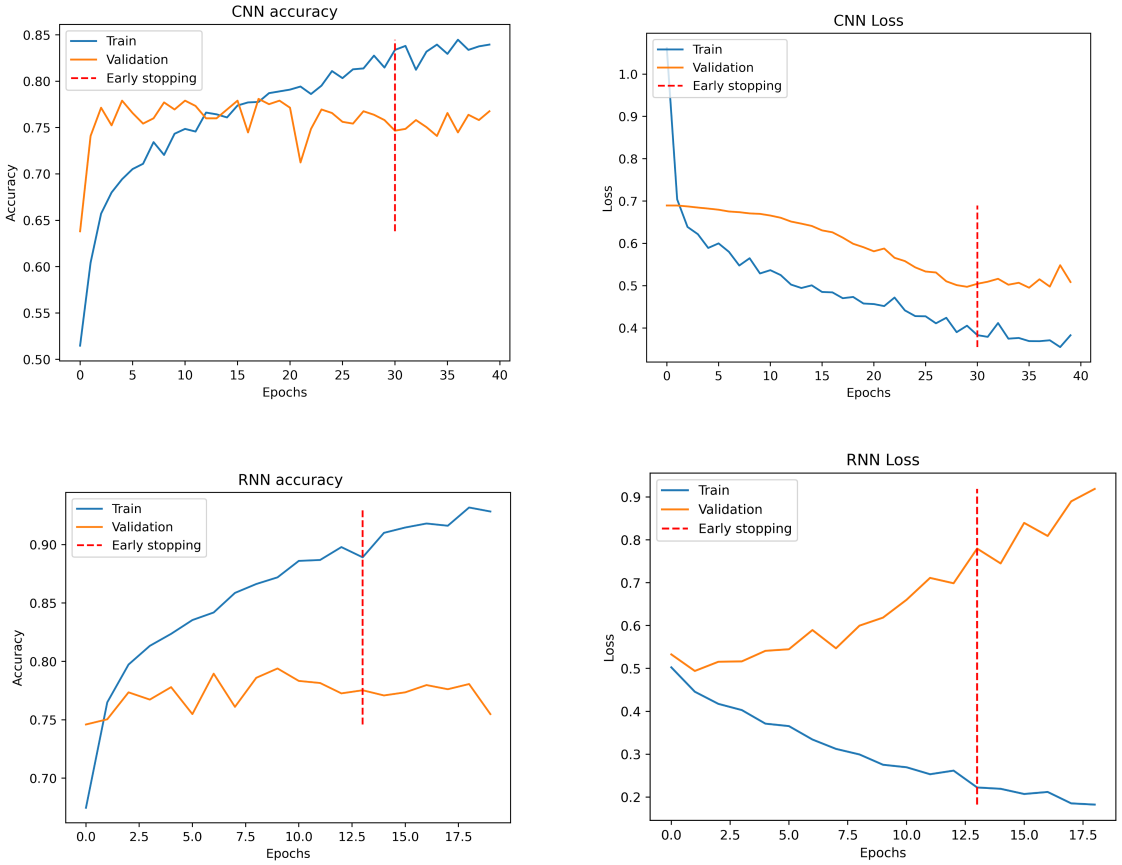

Figure 4: Courbes d'apprentissage des modèles de CNN et de RNN.

Concernant les réseaux de neurones, les MLP présentent les meilleures performances en particulier avec l'optimiseur SGD (Table 1). Les performances des RNN et CNN sont comparables. Le MINN est le réseau présentant les moins bonnes performances. Globalement, les performances des réseaux sont moins bonnes que les autres méthodes utilisées (hormis la régression logistique). Les courbes ROC (Figure 3) des réseaux se superposent, ce qui confirme des performances comparables.

Le surapprentissage est plus présent dans le RNN (Figure 4, bas). Malgré plusieurs combinaisons d'hyper paramètres testés, aucune n'a pu réduire ce dernier. Dans les deux réseaux de neurones présentés, à un moment donné, les valeurs d'Accuracy sur le jeu de validation stagnent autour de 0.75 alors que les valeurs d'Accuracy sur le jeu d'apprentissage continuent d'augmenter. Ceci traduit un sur-apprentissage, au-delà duquel aucun des modèles n'a pu mieux performer.

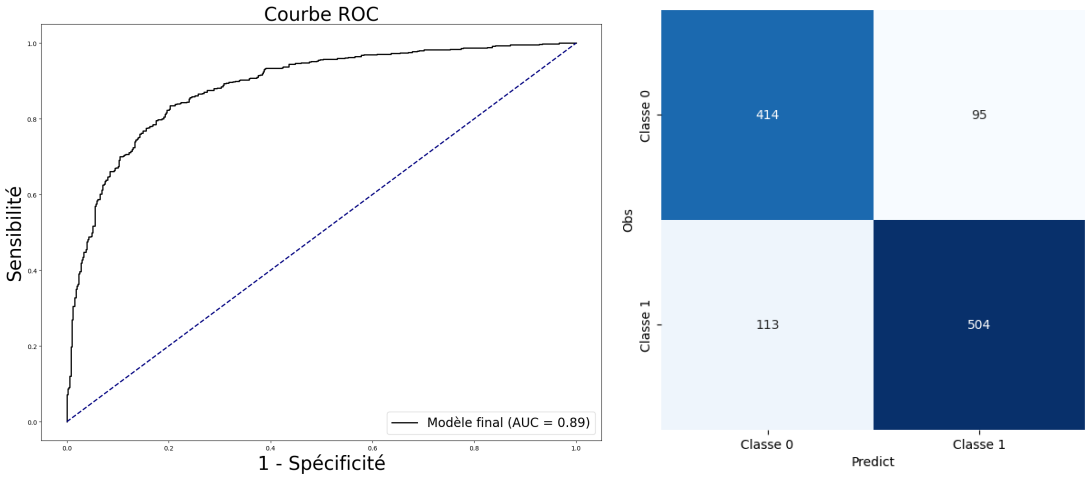

Figure 5: Performances du modèle final sur le jeu test.

Les performances du modèle final (Figure 5) sont comparables à celles du Random Forest, ce qui montre que la prise en compte de l'ensemble des modèles dans le choix final a permis d'améliorer la prédiction finale. Ce modèle prédit moins de faux positifs que le Random Forest, ce qui néanmoins, est fait au prix des vrais positifs. De surcroît, il présente une Accuracy de 0.82, une Sensibilité de 0.82 et une Spécificité de 0.81. Ceci indique que prendre en compte l'ensemble des modèles pour le choix final a permis une hausse significative globale de la spécificité, ce qui l'améliore grandement, étant donné que l'on souhaite avoir une spécificité la plus haute possible.

	RF	**MLP Adam**	**MLP SGD**	**CNN**	**RNN**	**MINN**	**Régression logistique**	**Gradient boosting**	**XGBoost**	**Modèle final**
Notes à la compétition	0.468	0.500	0.440	0.490	0.598	0.529	0.538	0.472	0.466	0.435

Figure 6 : Notes à la compétition Kaggle de nos différents modèles

Les notes à la compétition Kaggle de nos différents modèles ont été décrites dans la Figure 6. Selon la compétition, le meilleur modèle est le modèle final avec un score de 0.435. Dans le classement ce modèle se trouve 426 ème sur 698. De nouveau, il est pertinent d'avoir fusionné les modèles ensembles car le score à la compétition s'est amélioré.

Conclusion

Ce projet visait à prédire la réponse biologique des molécules en utilisant des propriétés chimiques comme descripteurs.

Le modèle final ainsi que le Random Forest et le Gradient Boosting sont les plus performants dans la prédiction de la réponse biologique.

Le modèle Gradient Boosting semble avoir de bonnes performances, avec une bonne capacité à discriminer les classes et une sensibilité élevée pour détecter les cas positifs. Le modèle de régression logistique et MINN montrent des performances similaires avec une précision raisonnable, une sensibilité correcte pour détecter les cas positifs, une spécificité raisonnable pour minimiser les faux positifs, un F1-score équilibré. Cependant, il est possible d'améliorer davantage ces métriques en ajustant les hyperparamètres du modèle ou en explorant d'autres techniques de modélisation.

Les réseaux de neurones de type MLP donnent de meilleures performances que les autres (CNN et RNN). Cependant, pour tous les types de réseaux, on remarque l'apparition de sur-apprentissage qui empêche d'atteindre de meilleures performances. Le RNN et le CNN montrent des performances similaires.

Les moins bonnes performances de nos modèles peuvent s'expliquer par le pré-traitement des données que nous avons effectué. On peut supposer que trop de variables ont été enlevées. On peut aussi supposer que du fait de la disparité du nombre d'individus entre les deux classes à prédire, l'apprentissage est biaisé.

Pour des améliorations futures, nous pourrions explorer davantage la réduction de la dimensionnalité, tester d'autres architectures de réseaux de neurones, et examiner plus en détail les causes du surapprentissage. De plus, le manque d'information sur les descripteurs ne permet pas de comprendre le rôle des variables impactantes dans les mauvaises prédictions, ce qui est un frein dans l'interprétation des erreurs.

Bibliographie

1. Predicting a Biological Response | Kaggle. https://www.kaggle.com/competitions/bioresponse/overview.
2. Deep learning methods for drug response prediction in cancer: Predominant and emerging trends - PMC. https://www-ncbi-nlm-nih-gov.ezproxy.u-paris.fr/pmc/articles/PMC9975164/.
3. What are Recurrent Neural Networks? | IBM. https://www.ibm.com/topics/recurrent-neural-networks.
4. 1D convolutional neural networks and applications: A survey - ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0888327020307846.
5. TPS Oct 2021 - Multi Input Neural Network | Kaggle. https://www.kaggle.com/code/kavehshahhosseini/tps-oct-2021-multi-input-neural-network/notebook.

Annexe:

Support Vector Machine (SVM) *pour le data 15*

SVM est un algorithme d'apprentissage automatique qui est utilisé pour la classification et la régression. SVM tente de trouver la meilleure ligne (ou hyperplan) qui sépare deux groupes de points de données tout en maximisant la distance entre cet hyperplan et les points les plus proches de chaque groupe. Cela permet de classifier de nouvelles données en fonction de leur position par rapport à cet hyperplan. Les meilleurs hyperparamètres ont été déterminés grâce à une recherche par grille (GridSearchCV) pour l'entraînement. Le code débute par l'importation des bibliothèques nécessaires et la définition d'une grille d'hyper paramètres pour la recherche. Ensuite, il crée un modèle SVM en spécifiant certains paramètres clés, tels que la régularisation C, le type de noyau kernel, et le paramètre gamma. Une fois le modèle configuré, il effectue une recherche d'hyper paramètres pour trouver la meilleure combinaison. Après avoir identifié les paramètres optimaux, il entraîne le SVM, effectue des prédictions sur les données de test, évalue la précision et génère diverses métriques de performance. Les valeurs spécifiques de ces paramètres (C=1, gamma=scale, kernel=rbf) indiquent que la régularisation est modérée, que gamma est adapté à l'échelle des données, et que le noyau radial est utilisé pour traiter des données potentiellement non linéaires.

SVM : accuracy=0.79, sensibilité=0.83, spécificité=0.75, F1_scrore=0.82

Globalement, SVM est un modèle qui tente de trouver un hyperplan pour séparer les classes de données, et les résultats obtenus montrent que ce modèle est capable de prédire avec succès si une molécule provoque ou non une réponse biologique dans 78,77 % des cas.

La spécificité du modèle est de 0.75, ce qui signifie qu'il parvient à classifier correctement 75 % des exemples négatifs. Une spécificité élevée est souhaitable pour réduire les faux positifs. Le F1-score est de 0.82, ce qui représente une mesure globale de la précision et du rappel du modèle. Un F1-score élevé indique une performance globale solide. Avec un score de 0.58, le modèle SVM montre une performance raisonnable. En résumé, le modèle SVM démontre des performances globalement solides avec une précision correcte, une sensibilité élevée pour détecter les cas positifs, une spécificité raisonnable pour minimiser les faux positifs et un F1-score équilibré.

Cela suggère que le modèle SVM est compétent pour la tâche de classification, bien qu'il soit toujours possible d'explorer des ajustements d'hyper paramètres ou des techniques de modélisation supplémentaires pour l'améliorer davantage en fonction des besoins spécifiques de l'application.