*L'objectif de la compétition est de nous aider à construire le meilleur modèle possible afin que nous puissions, de manière optimale compte tenu de ces données, établir une relation entre les informations moléculaires et une réponse biologique réelle.*

*Pour le projet concernant le rapport et l’oral : Parler des fonctions utilisées comment elles marchent, quels arguments sont utilisés derrière*

Données au format de valeurs séparées par des virgules (CSV).

**Chaque ligne de cet ensemble de données représente une molécule.**

**La première colonne contient des données expérimentales décrivant une réponse biologique réelle ; la molécule a été observée pour provoquer cette réponse (1) ou non (0).**

**Les colonnes restantes représentent des descripteurs moléculaires (d1 à d1776). Ce sont des propriétés calculées qui peuvent capturer certaines caractéristiques de la molécule, telles que sa taille, sa forme ou sa composition élémentaire.**

**La matrice des descripteurs a été normalisée.**

* [Prédire la réponse biologique grâce au GBM léger | Kaggle](https://www.kaggle.com/code/vernondsouza123/predict-biological-response-through-lightgbm) :

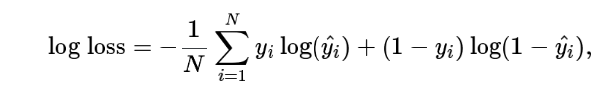
Ce projet vise à prédire la présence ou l’absence de réponse biologique d’une molécule, en se basant sur des descripteurs moléculaires contenues dans chaque colonne. Le développement de médicaments dépend de divers facteurs qui influencent la réponse biologique, et l'apprentissage automatique peut nous aider à évaluer l'impact du médicament sur le traitement. Chaque colonne du jeu de données représente des informations moléculaires, mais étant donné le grand nombre de colonnes, le principal défi réside dans la réduction de la dimensionnalité des caractéristiques, c'est-à-dire dans la réduction de l'espace des caractéristiques.

* Réponse exemple de code : [GitHub - benhamner/BioResponse: Benchmarks for Boehringer-Ingelheim's Predicting a Biological Response Competition, hosted by Kaggle](https://github.com/benhamner/BioResponse)
* “4 ans plus tard” : [Prédire une réponse biologique | Kaggle](https://www.kaggle.com/competitions/bioresponse/discussion/26567) : En 2012, aucun prétraitement des données, aucune transformation logarithmique, aucune normalisation, et aucune validation croisée n'ont été effectués. Ces modifications pourraient améliorer le modèle de prédiction.
* Random forest meilleur que SVM : [Prédire une réponse biologique | Kaggle](https://www.kaggle.com/competitions/bioresponse/discussion/3472)
* Régression logistique
* Code : [bio-response | Kaggle](https://www.kaggle.com/code/ashtcoder/bio-response)

**Evaluation**

Les probabilités prédites que la molécule provoque une réponse sont évaluées à l'aide de la métrique de la perte logarithmique (log loss).

La perte logarithmique est définie comme suit :



où \(N\) est le nombre d'échantillons,

\(\text{log}\) est le logarithme naturel,

\(\hat{y\_i}\) est la probabilité postérieure que le \(i^{ème}\) échantillon provoque une réponse,

et \(y\_i\) est la vérité de terrain (\(y\_i=1\) signifie que la molécule a provoqué une réponse, \(y\_i=0\) signifie qu'elle ne l'a pas fait).

**EXPLICATIONS :**

* Si votre modèle prédit une probabilité très proche de 1 pour un exemple qui est effectivement de la classe 1, la log loss sera très faible, ce qui est bon. Cela signifie que votre modèle a bien prédit cette observation.
* Si votre modèle prédit une probabilité proche de 0 pour un exemple qui est effectivement de la classe 1, la log loss sera très élevée, ce qui est mauvais. Cela signifie que votre modèle a mal prédit cette observation.
* De même, si votre modèle prédit une probabilité proche de 0 pour un exemple qui est effectivement de la classe 0, ou une probabilité proche de 1 pour un exemple qui est effectivement de la classe 0, la log loss sera élevée.

Plus la probabilité prédite par votre modèle est proche de la vraie valeur (0 ou 1), plus la log loss est faible, ce qui est un indicateur de la précision de votre modèle. En revanche, si les prédictions divergent considérablement des valeurs réelles, la log loss augmente, ce qui signifie que votre modèle a du mal à faire des prédictions précises. L'objectif est de minimiser la log loss, car cela indique que votre modèle fait des prédictions précises.

**Description du dataset:**

Les descripteurs d1 à d1776 sont des variables numériques qui représentent diverses propriétés moléculaires calculées à partir de données expérimentales ou de modèles mathématiques. Ces descripteurs peuvent inclure des informations sur la taille, la forme, la charge, la polarité, la composition chimique, la structure moléculaire, et bien d'autres caractéristiques de chaque molécule. Chaque d1, d2, ..., d1776 représente une caractéristique spécifique extraite ou calculée à partir des molécules.

Les molécules elles-mêmes ne sont pas spécifiées dans la description. Les données sont génériques et anonymes, ce qui signifie qu’il s’agit d’informations sur un ensemble des molécules, mais nous ne connaissons pas leur nature exacte ou leur application spécifique. Les molécules peuvent appartenir à divers domaines, tels que la chimie pharmaceutique, la chimie organique, la chimie des matériaux, etc. La nature des molécules dépend de la source des données et de l'objectif de l'analyse.

**Méthode:**

Comprendre les données : Charger les données depuis le fichier CSV. Une colonne de sortie (1 pour une réponse biologique positive, 0 pour une réponse négative) et 1776 descripteurs moléculaires. Comprendre la structure des données et leurs caractéristiques.

Exploration des données (EDA) : Effectuer une analyse exploratoire des données pour visualiser et comprendre la distribution de la réponse biologique et les descripteurs moléculaires. Utiliser des graphiques et des statistiques descriptives pour identifier des tendances, des valeurs manquantes ou des outliers.

Prétraitement des données : Prétraiter les données en normalisant les descripteurs moléculaires si ce n'est pas déjà fait. Utiliser des techniques telles que la Standardisation (Z-score) pour mettre à l'échelle les données.

Séparation des données : Diviser les données en un ensemble d'entraînement (pour former le modèle), un ensemble de validation (pour ajuster les hyperparamètres) et un ensemble de test (pour évaluer la performance finale). Une division courante est 70-80% d'entraînement, 10-15% de validation et 10-15% de test.

Choix du modèle : Travailler sur une tâche de classification (prédire 1 ou 0), alors utiliser des modèles de classification tels que la régression logistique, les arbres de décision, les forêts aléatoires, les SVM, ou des modèles de réseaux de neurones.

Entraînement du modèle : Entraîner le modèle sur l'ensemble d'entraînement en ajustant les hyperparamètres et en surveillant les métriques de performance sur l'ensemble de validation. Pour les réseaux de neurones, Utiliser des bibliothèques comme TensorFlow ou PyTorch.

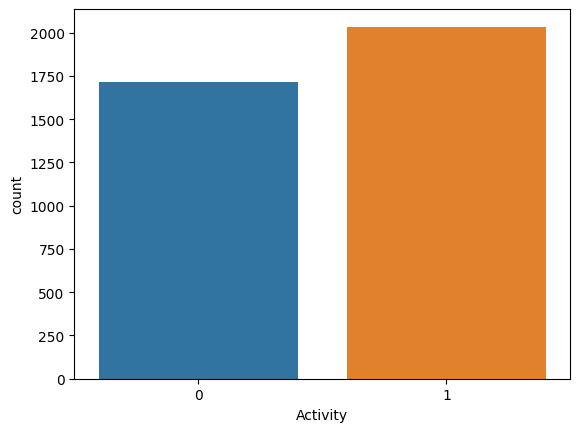
Évaluation du modèle : Évaluer la performance du modèle en utilisant des métriques appropriées telles que la précision, le rappel, la F1-score, l'aire sous la courbe ROC (AUC-ROC), etc. Utiliser l'ensemble de test pour obtenir une évaluation impartiale.

Optimisation : Selon les résultats de l'évaluation, ajuster les hyperparamètres du modèle et répéter les étapes d'entraînement et d'évaluation pour améliorer la performance.

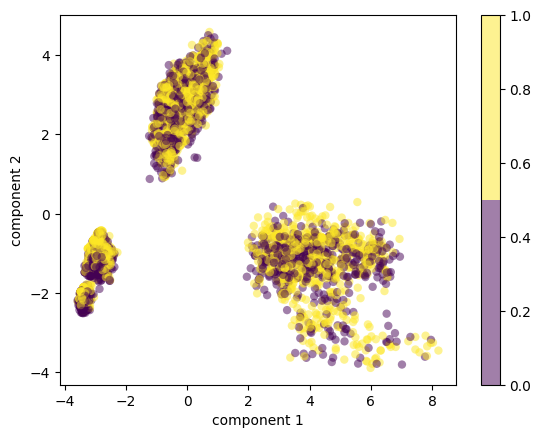
Prédiction : Une fois que satisfait de la performance, utiliser le modèle pour faire des prédictions sur de nouvelles données si nécessaire.

Rapport et soumission : Préparer un rapport détaillé de l'approche, des méthodes utilisées et des résultats obtenus.

**Figures:**

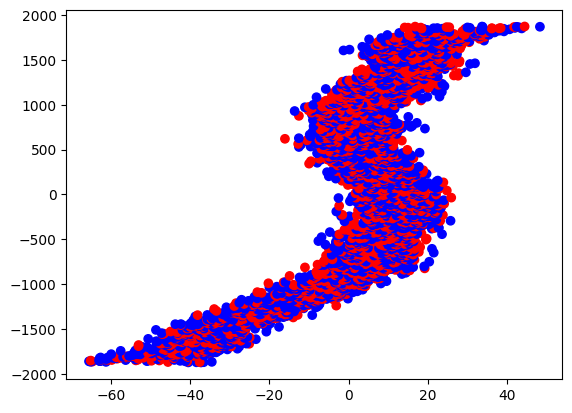


Effectifs des classes de Y du jeu de données complet : faire un lien avec la matrice de confusion



Graphe des individus de l’ACP. Les molécules déclenchant une réponse biologique sont en jaune. Celles n’en déclenchant pas sont en violet.

On distingue 3 groupes distincts mais qui ne semblent en rien être liés avec l'activité ou non de la molécule.

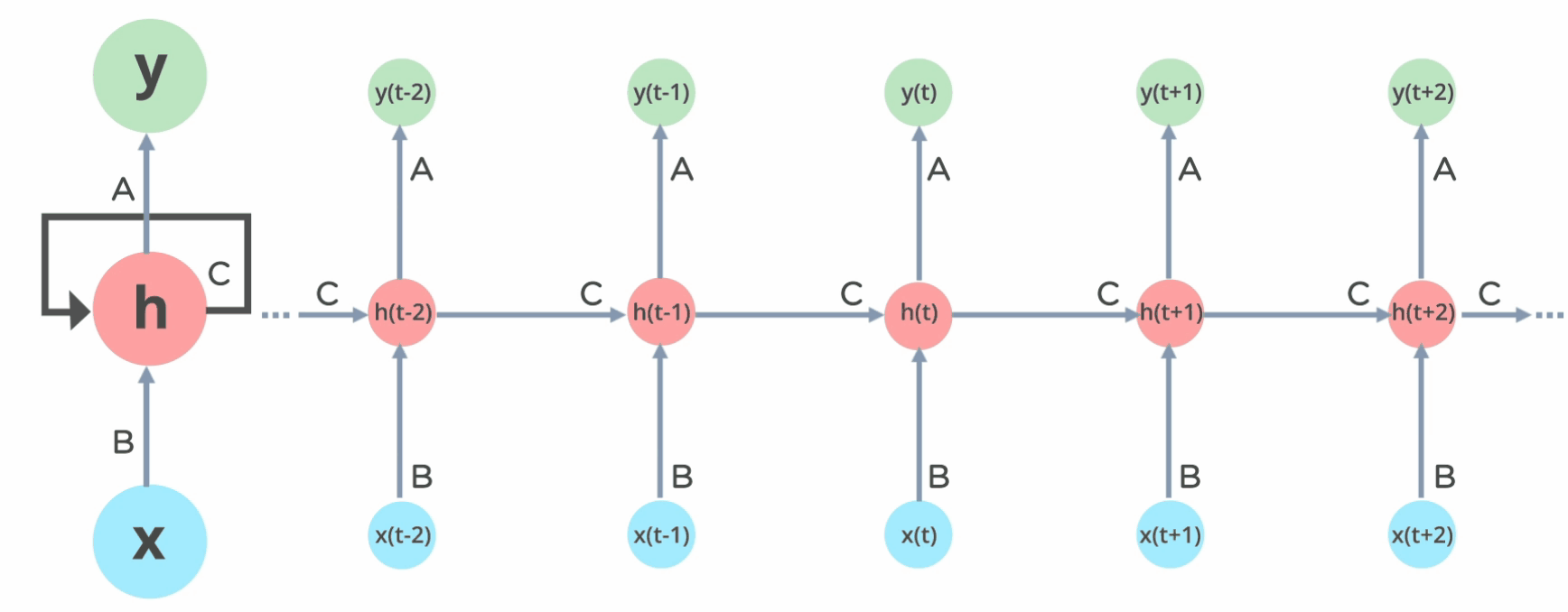


Résultats du MDS. Les molécules déclenchant une réponse biologique sont en bleu. Celles n’en déclenchant pas sont en rouge.

On ne distingue pas du tout de groupes visibles.

RNN :

Dans les réseaux de neurones récurrents, l'information circule à travers une boucle jusqu'à la couche cachée centrale



<https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>

La couche d'entrée 'x' prend l'entrée du réseau neuronal, la traite et la transmet à la couche centrale.

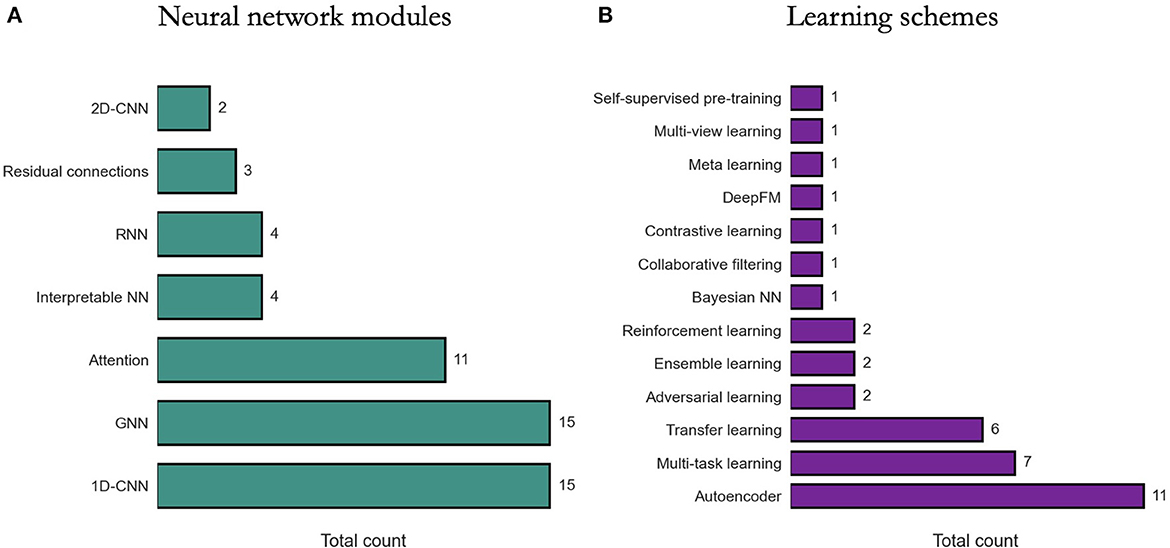
La couche centrale 'h' peut être constituée de plusieurs couches cachées, chacune ayant ses propres fonctions d'activation, poids et biais. Si vous avez un réseau neuronal où les paramètres variés des différentes couches cachées ne sont pas influencés par la couche précédente, c'est-à-dire que le réseau neuronal n'a pas de mémoire, alors vous pouvez utiliser un réseau neuronal récurrent.

Le réseau neuronal récurrent standardisera les différentes fonctions d'activation, poids et biais de manière à ce que chaque couche cachée ait les mêmes paramètres. Ensuite, au lieu de créer plusieurs couches cachées, il en créera une seule et la bouclera autant de fois que nécessaire.

→ il faut utiliser feed forward NN : CNN, MultiLayer Perceptron, …

LIEN ARTICLE :

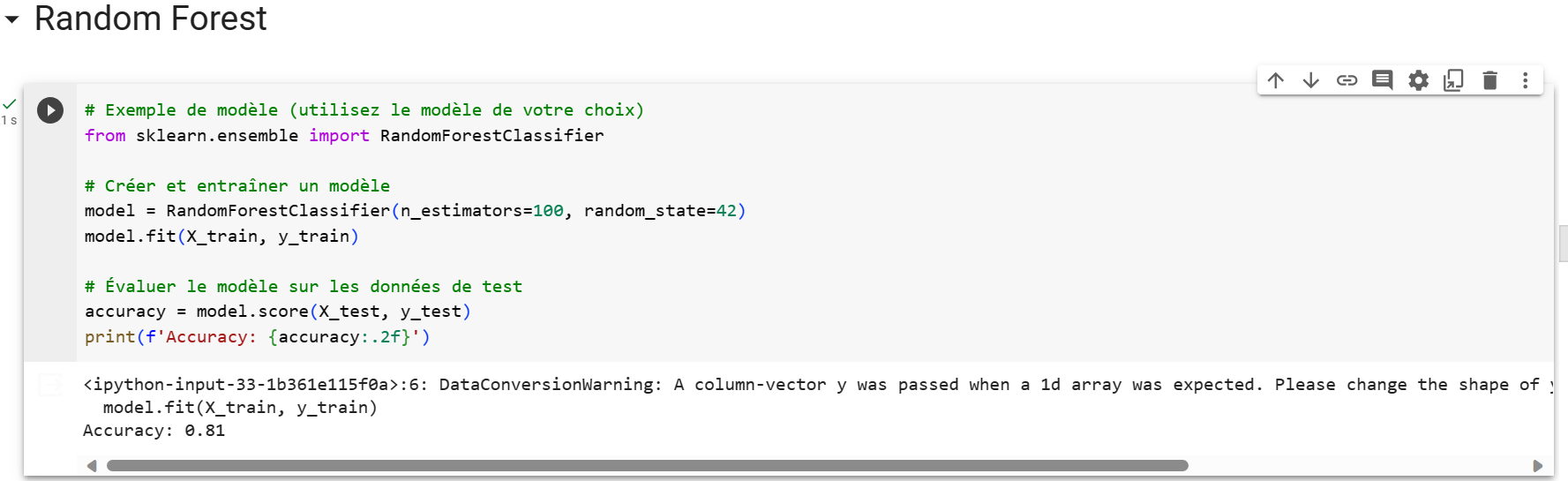
<https://ieeexplore.ieee.org/abstract/document/7004386?casa_token=osVCQO101EgAAAAA:vY09vaSWxZoAlCsddsW2VzfIPPDLYi-aL0nJu9zjkdw-jE3JVfbqZ3Pr-OmqsF8q2cUcckRW_A>



Présentation : 15-20 min

Gridsearshed: lancer différent modèle : nb couche predef (lui donner une liste de choses à tester)

* Les données: nettoyage? → supp variances nulle + scale
* Random Forest -> accuracy = 0.81

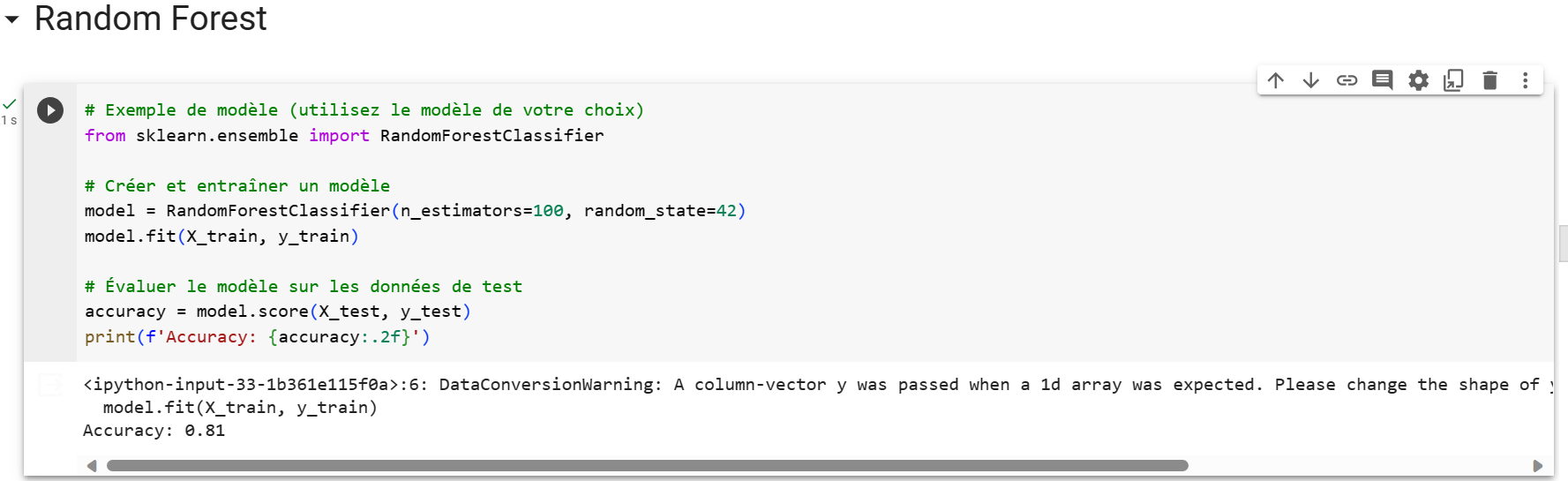


* RNN →pas ouf donc →CNN
* CNN
* Dense : MLP( même para que CNN)
* Feed forward ( ouverture : Radial basis function?)

TODO:

* ~~Refaire random Forest~~

-> accuracy = 0.81



* Retester sur dense et tout refaire avec gridsearch : <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>
* ~~Hyperparamètres et optimisation CNN (en cours) :~~ [~~https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model\_selection~~](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection)
* check la logloss et les plots associé
* Log Loss et graphes : <https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a>
* Courbe ROC

Idées pour des jolis représentations des réseaux: https://github.com/ashishpatel26/Tools-to-Design-or-Visualize-Architecture-of-Neural-Network

Exemple de modèles : SVM (Support Vector Machine), Régression logistique, Gradient Boosting (XGBoost, LightGBM, CatBoost), Réseaux de neurones profonds (Deep Learning)

Rapport :

Introduction:

* Pourquoi étudier la prédiction de réponses de molécules biologique : enjeux

Le développement de médicaments dépend de divers facteurs qui influencent la réponse biologique, et l'apprentissage automatique peut nous aider à évaluer l'impact du médicament sur le traitement.

* Présentation données :

Ce projet vise à prédire la présence ou l’absence de réponse biologique d’une molécule, en se basant sur des descripteurs moléculaires. Chaque colonne du jeu de données représente des informations moléculaires, mais étant donné le grand nombre de colonnes, le principal défi réside dans la réduction de la dimensionnalité des caractéristiques, c'est-à-dire dans la réduction de l'espace des caractéristiques

Preprocessing:

* Exploration préliminaire des données : ACP, voit-on des groupes distincts ?
* Nettoyages basiques : aucune données manquantes, suppression des variables avec peu de variabilité, variables fortement corrélées et colinéaires

A la fin, 1112 descripteurs sur les ? initiaux.

* Normalisation : argumenter le choix, MinScale
* Introduction de la logloss comme critère d'évaluation

Premier Modèle : Random Forest

* Pourquoi
* Ses avantages : prend en compte les variables binaires ?
* Ses limites
* Performances : 0.81 acc sur le test

Second Modèle RNN:

* Pourquoi avoir tester avec un RNN
* Perf et pourquoi ça ne marche pas (type de données non adapté ?)

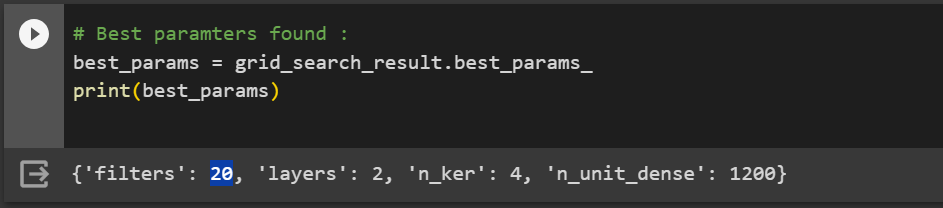
Troisième Modèle Dense (avoir pour l’ordre):

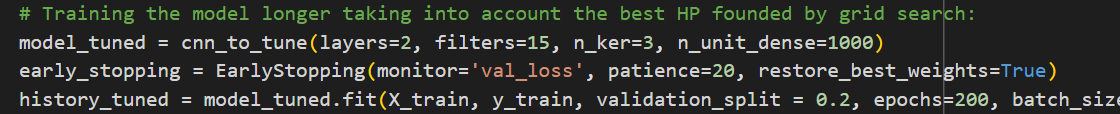
Etant donné que les réseaux de neurones de type récurrent ne semblent pas être adaptés au jeu de données, nous nous sommes par la suite concentrés sur des Feed Forward Neural Networks. Tout d’abord, nous avons étudié un modèle constitué de couches denses. Le modèle utilisé est décrit dans la Figure ?.

* Description du modèle : Modèle assez simple finalement avec “peu” d'hyper paramètres à optimiser
* Pourquoi ?
* Tuning du modèle
* Ses avantages
* Ses limites
* Performances

Quatrième Modèle CNN:

Un autre type de FNN est le CNN.

* Description du modèle : Couches de convolution 1D, adapté pour nos données.
* Pourquoi
* Tuning du modèle : GridSearch/Randomized, : 



choix du learning rate (plot)

* Ses avantages : De même que pour le MLP, le CNN construit est rapide à tourner. Assez simple dans l’architecture.
* Ses limites : au bout d’un moment, toujours du surapprentissage
* Performances : moins bonnes que le RDF jusqu’à présent.

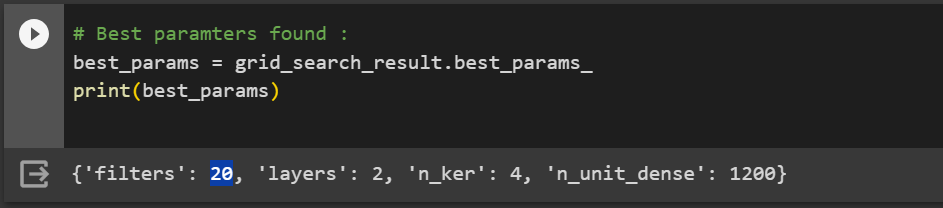
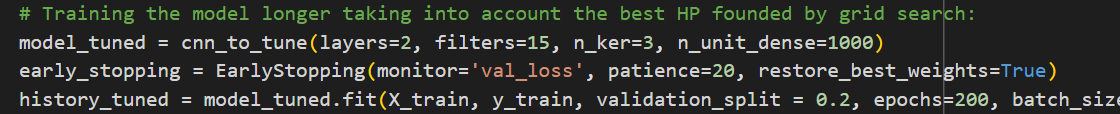
Cinquième Modèle Modèle qui combine CNN + Dense ?

* Est ce mieux que les deux autres séparés ?
* Règle-t-il mieux le surapprentissage ?
* Par rapport au RandomForest
* Soumission à la compétition et note obtenue

Conclusion:

* retour sur analyse : propal : plots des accuracies des différents modèles sur les jeux test, parler de la matrice de confusion en fonction du contexte de la question
* Choix des variables : préprocessing, équilibre entre trop et pas assez de variables
* Les modèles feed forwards semblent mieux fonctionner sur nos données : explications : pas de points de temps ni de données séquentielles ?
* quel fut le meilleur modèle : le RD Forest ?
* perspectives

Bibliogaphie

* 
* 

Mauvaise prédiction :

