# Projet 8: Participez à une compétition Kaggle !

# Novozymes Enzyme Stability Prediction

Page d'accueil de la compétition:

https://www.kaggle.com/competitions/novozymes-enzyme-stability-prediction

Mon kernel kaggle:

https://www.kaggle.com/tonymathieux/eda-and-groups-of-variants-in-training-dataset

Github associé au projet :

https://github.com/MathieuxTony/Projet-8

# 1) Le sujet

### Les enzymes

Les enzymes sont des substances organiques qui catalysent des réactions biochimiques. Ce sont des protéines, c'est-à-dire des molécules constituées d'une succession d'acides aminés. Il y a 22 acides aminés différents qui peuvent apparaître dans la structure d'une protéine, mais deux d'entre eux ne se rencontrent que dans des cas très spécifiques qui ne figurent pas dans ce projet, les 20 autres sont dits standards. On attribut à chaque acide aminé une lettre majuscule et on peut donc écrire une protéine comme une suite de lettres. La taille, ou longueur, des protéines est très variable : de quelques acides aminés à plusieurs milliers.

Quand des protéines ont une structure très proche, on parle de variants d'une même protéine. On peut notamment, et c'est ce que fait la société Novozymes, modifier artificiellement la structure d'une enzyme naturelle (on créé un variant par mutation de l'enzyme de base) afin d'optimiser ses propriétés à des fins industrielles. On peut utiliser des enzymes dans des domaines aussi différents que l'alimentation, la fabrication de détergents ou encore de biocarburants.

## Objectif

Les enzymes se dégradent en fonction de la température. Actuellement on détermine précisément la température, notée 'tm' pour « melting temperature », à laquelle une enzyme se dégrade de façon expérimentale. Ce qui implique qu'il faut déjà fabriquer l'enzyme puis faire une expérience. Le but de cette compétition Kaggle est de prédire la température 'tm' des variants d'une certaine enzyme.

## Prédictions à partir d'une séquence d'acides aminés

Nous ne sommes pas encore parvenu à prédire précisément la stabilité thermique d'une protéine (d'où cette compétition), mais il y a quand même eu des avancées majeures dans les prédictions basées sur les séquences d'acides aminés. Par exemple FoldX permet de prédire l'influence des mutations sur la stabilité, et AlphaFold permet de prédire la structure 3D des protéines (AlphaFold est bien basé sur du machine learning, mais FoldX se base sur des principes physiques).

## Les données fournies par Kaggle

- Un fichier csv, que nous appellerons « train », contenant des données pour entraîner nos modèles (attention, il y a des erreurs dans le fichier de base, les corrections se trouvent

dans un autre fichier et c'est à nous de combiner les deux fichiers). Le fichier contient les séquences d'acides aminés (chaînes de caractères), les pH et les températures de fusion tm  d'environ 29 000 protéines.

- Un fichier csv, que nous appellerons « test », contenant les séquences d'acides aminés et les pH d'environ 2400 variants d'une enzyme, dont on veut prédire tm.
- Un fichier pdb contenant des informations sur l'enzyme de base du fichier test.

# 2) Approches envisagées

Les données fournies ne contiennent en l'état que deux variables donc il va falloir en créer d'autres, soit à partir de la variable 'protein_sequence', soit en téléchargeant d'autre données sur internet. Dans les deux cas, des connaissances spécifiques sur les protéines et leurs mutations constituent un plus non négligeable.

J'ai choisi de traiter les séquences des protéines simplement comme des chaînes de caractères, comme s'il n'y avait pas de contexte biologique derrière, et de me concentré sur des questions qui viennent naturellement au vu des données :

Est-ce qu'il y a dans « train » des variants d'une même enzyme ? Si oui, comment les trouver et les regrouper ? A quel points des variants d'une même protéine peuvent avoir des tm différents ?, des pH différents ?

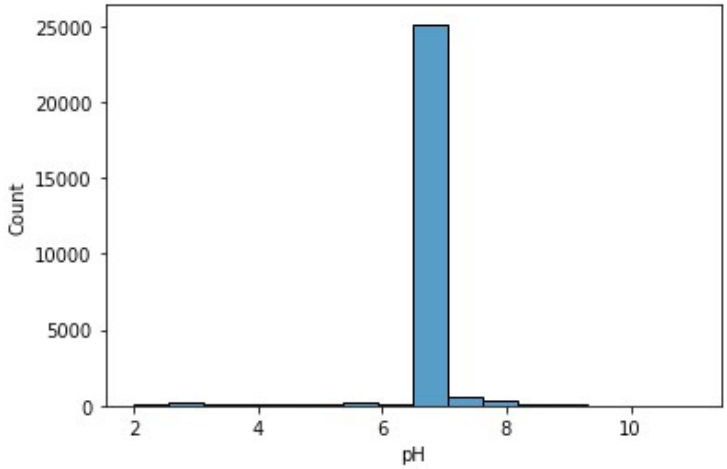
Ainsi l'essentiel de mon travail se trouve dans l'analyse exploratoire des données et c'est sur cela que j'ai basé mon kernel kaggle . Il y aura bien une partie consacré à la modélisation, mais elle sera courte.

# 3) Analyse exploratoire des données

## a) Les variables numériques

### pH

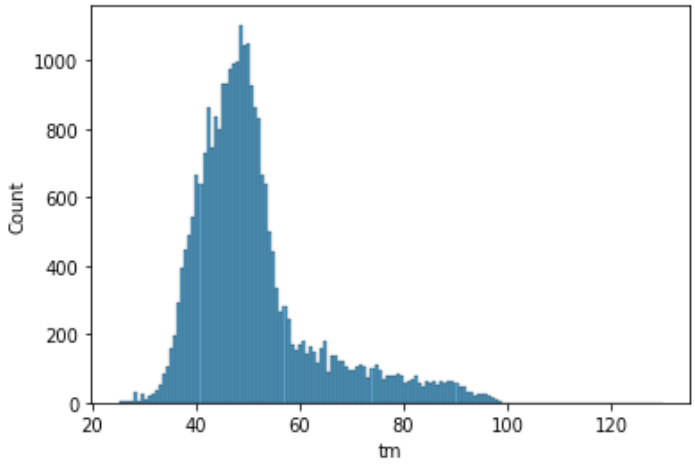
Toutes les protéines de « test » ont un pH de 8 et quasiment toutes les protéines de « train » ont un pH de 7.




*Histogramme des pH de "train"*

### tm

Les températures sont indiqués en degré Celsius avec une précision au dixième de degré. On a une concentration entre environ 40 et 55 degrés, avec un étalement vers des températures plus élevées.

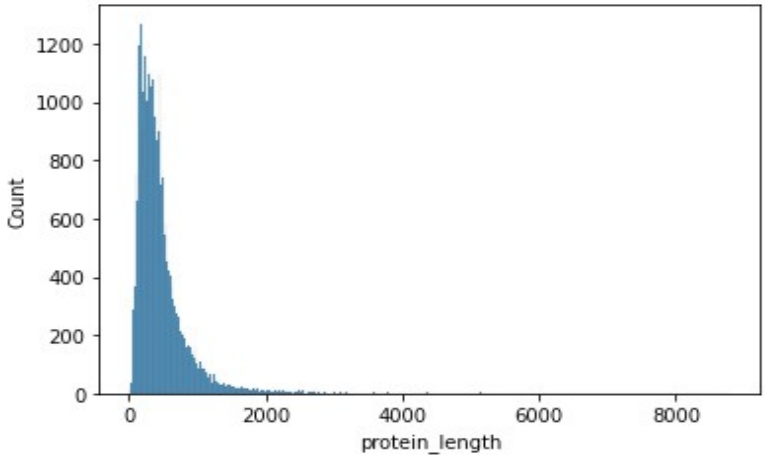



*Histogramme des 'tm' de "train"*

# b) Les séquences des protéines

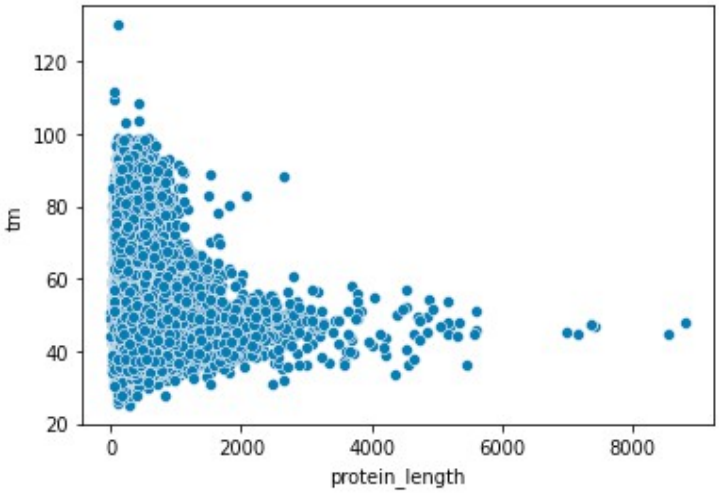
## Longueur des séquences

Les protéines de « train » comportent en générale quelques centaines d'acides aminés. Certaines en ont quelques dizaines et d'autres, plus de mille.




*Histogramme des longueurs des séquences de "train"*

Le pH ne variait pas assez pour tracer 'tm' en fonction du pH, mais 'protein_length' varie suffisamment. Il n'y a pas de corrélation visible entre ces deux varialbes :




*'tm' en fonction de la longueur des séquences*

Les longues protéines vont posé des problèmes pour la suite. D'une parts elles vont considérablement augmenter la taille des données dans l'une des approches de modélisation. D'autre parts le test que nous ferons pour déterminer les variants de « train » est plus long avec des protéines plus longues et ce, alors que des protéines de tailles très différentes ne peuvent pas être des variants d'une même protéine. J'ai donc décider de ne garder que des protéines de moins de milles acides aminés.

Comme les protéines de « test » sont des variants d'une même enzyme, leurs séquences ont bien entendu toutes à peu près la même longueur : 221 en générale et 220 pour quelques dizaines d'entre elles.

### Quantité de chacun des acides aminés

J'ai mis en œuvre deux approches de création de nouvelles variables à partir des séquences des protéines. La première consiste à prendre en compte les quantités de chaque acide aminé dans les protéines, ce qui remplace notre variable 'protein sequence' par 20 variables numériques.

### Position des acides aminés

La deuxième approche consiste à remplacer 'protein sequence' par mille variables (mille est la longueur de la plus longue séquence) indiquant l'acide aminé présent en première position, deuxième position, troisième position, etc...

# 4) Recherche de variants dans « train »

L'idée est d'utiliser une distance qui donne une faible valeur entre les variants, et seulement entre les variants. La bonne distance est la distance de Levenshtein, mais j'ai aussi testé la distance de Hamming parce qu'elle permet quand même de trouver

beaucoup de variants (les variants non détectés sont ceux avec un décalage dans leur séquence) et parce que cette distance est utilisable avec la fonction 'pairwise_distances' de scikit-learn.

J'ai d'abord utilisé ces distances sur le jeu « test » car on sait qu'il n'y a que des variants d'une même enzyme et parce que ce jeu de données est beaucoup plus petit que « train ».

## Le problème des temps de calculs

La méthode brute pour trouver les variants consiste à calculer les distance de Levenshtein entre les protéines de « train » deux à deux, puis de regrouper les protéines qui sont proches.

Le calcul de la matrice des distances de Levenshtein pour le jeu « test », qui contient environ 10 fois moins de protéines que « train », permet d'évaluer le temps de calcul de la matrice des distances de Levenshtein pour le jeu « train » à un peu plus de 40 minutes (avec un processeur 12$^{th}$ Gen intel Core i7-12700K et 8Go de RAM). Cette estimation ne prends pas en compte le temps de regroupement des variants à partir de la matrice des distances. Mon objectif était de faire le regroupement des variants de « train » en beaucoup moins de 40 minutes.

## Méthode employée

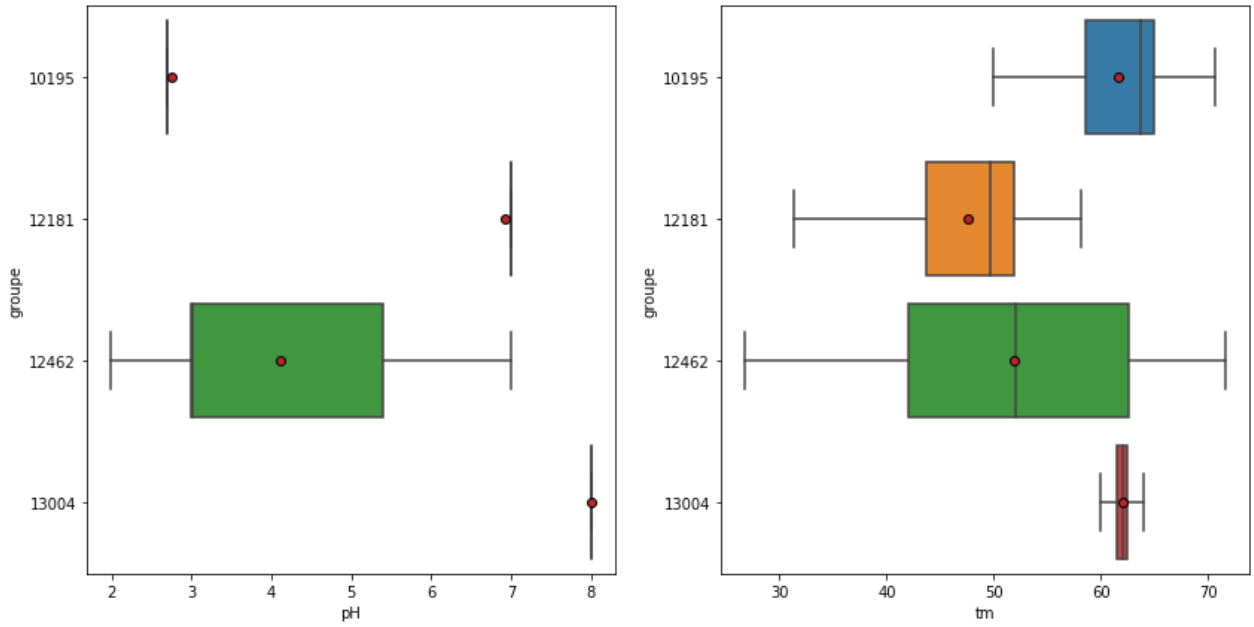
Plusieurs participants à cette compétition ont publié un notebook dans lequel ils cherchent les variants de « train ». Ils ont employé des méthodes assez complexes avec parfois un GPU pour accélérer les calculs. Je ne voulais pas recopier une méthode lu dans un de ces notebooks mais proposé ma propre méthode, d'autant que j'avais plusieurs idées pour aborder le problème.

Finalement j'ai pu regrouper les variants en environ 30 minutes de calculs, en évitant de calculer la matrice des distances entre tous les couples mais en regroupant les variants au fur et à mesure: l'idée est que si 'b' et 'a' sont proches (pour la distance de Levenshtein) et si 'c' et 'a' sont proches, alors 'b' et 'c' sont proches, donc on peut économiser des itérations.

## Résultats

On attribut à chaque protéine de « train » un numéro de groupe. Les variants d'une même protéine ont le même numéro.

Il y a 4 groupes de plus de cent variants dans « train » et l'analyse statistique de ces groupes montre notamment que des variants peuvent avoir une thermostabilité très différente, ce qui implique que pour être performante, une modélisation doit pouvoir donner des résultats très différents à partir de séquences de protéines très proches. On remarque que le pH aussi peut fortement varier entre des variants (groupe 12462 ci-dessous).




*Répartition des variables 'pH' et 'tm' au sein de 4 groupes de variants*

Enfin, précisons qu'il y a beaucoup de protéines isolés ( c'est-à-dire seules dans leur groupe) et qu'il n'y a pas de variant de l'enzyme de « test » dans « train ».

# 5)Modélisation

## Les données

Le jeu « train1 » se compose de 20 variables indiquant les quantités de chacun des 20 acides aminés présent, de la variable pH, ainsi que d'une variable indiquant un numéro de groupe de variant. On va faire des tests avec, et sans cette variable 'groupe' afin de voir si elle améliore les prédictions.

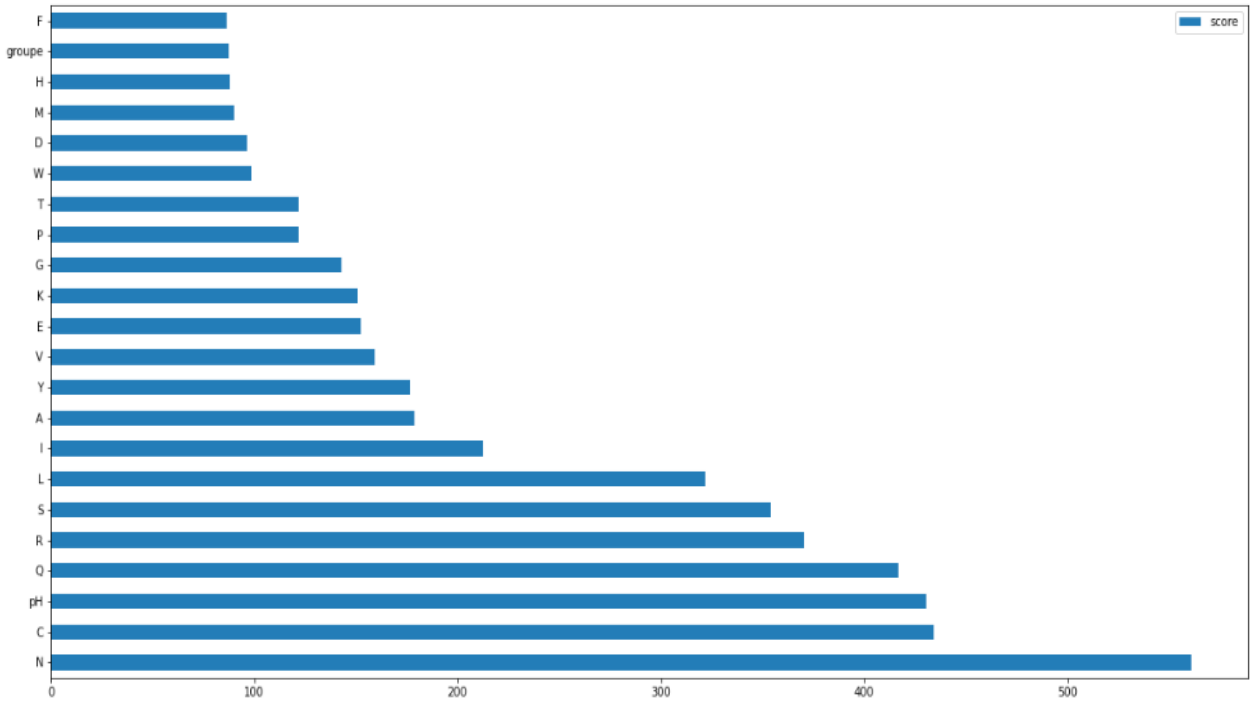
Le jeu « train2 » se compose de mille variables indiquant l'acide aminé présent à chaque position, ainsi que du pH.

## L'algorithme

L'algorithme mis en œuvre est le gradient boosting avec Xgboost. Celui-ci a l'avantage de s'entraîner rapidement, ce qui permet une optimisation des hyperparamètres avec optuna.

## Résultats

La modélisation à partir de « train1 » est clairement meilleure que celle à partir de « train2 », laquelle ne fait d'ailleur pas beaucoup mieux qu'une prédiction par un DummyRegressor. En revanche la variable 'groupe' n'a pas d'influence significative sur les prédictions, ce qui est également confirmé par le diagramme de feature importance du modèle :



Notre meilleur modèle reste cependant mauvais puisque l'erreur en valeur absolue est en moyenne de 5,7 degré pour des données ayant un écart-type d'environ 12 degrés.

La soumission d'une prédiction pour la compétition à donné un mauvais score de 0,106. Notez qu'au moment de l'écriture de ce rapport peu de participant ont réussi à atteindre un score de 0,5.

# Bibliographie


https://en.wikipedia.org/wiki/Amino_acid

https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

https://www.kaggle.com/code/cdeotte/train-data-contains-mutations-like-test-data

https://practicaldatascience.co.uk/machine-learning/how-to-tune-an-xgbregressor-model-with-optuna

https://mljar.com/blog/feature-importance-xgboost/