CentraleSupélec kaggle OPENCLASSROOMS

Participez à une compétition Kaggle !

Note méthodologique

Adrian Rodriguez - Ingénieur Machine Learning

1. Préambule

Mon choix s'est porté sur la compétition [Riiid! Answer Correctness Prediction]. Cette compétition consiste à modéliser les connaissances des élèves au fil du temps dont l'objectif est de prédire avec précision les performances de ces derniers lors des interactions futures.
Cette compétition s'appuie sur les données EdNet de Riiid. Les organisateurs nous fournissent les données d'entraînements comprenant plus de 100 millions d'interactions pour 10 variables.
En approfondissant mes recherches, j'ai trouvé que les données portent sur l'examen du TOEIC. Ce dernier mesure les compétences de compréhension écrite et orale de son utilisateur. Il détermine si une personne peut communiquer en anglais efficacement dans un contexte professionnel avec d'autres personnes.

2. Travail préalable

2.1. Nettoyage du jeu de données

L'idée de départ est de se faire une idée du niveau d'utilisation des utilisateurs. Les données de train comprennent plus de 390 000 utilisateurs. En agrégeant les données utilisateurs, j'ai cherché à connaître le nombre d'interactions, le nombre de jours entre la première et la dernière interaction ainsi que le score moyen de chacun des utilisateurs.
Illustrations en figures suivantes :

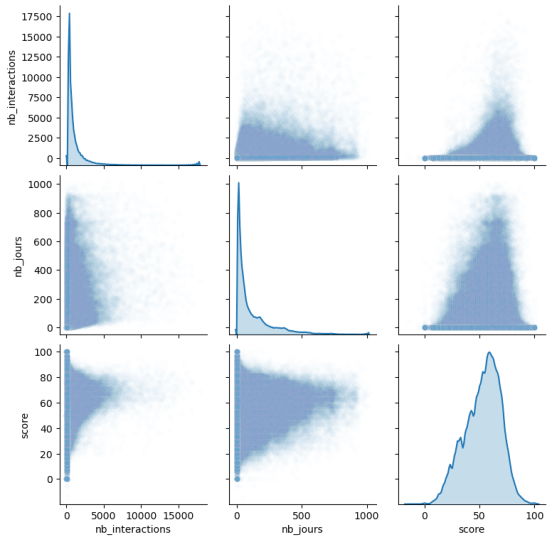

Figure 1 : Pairplot des valeurs agrégées des utilisateurs

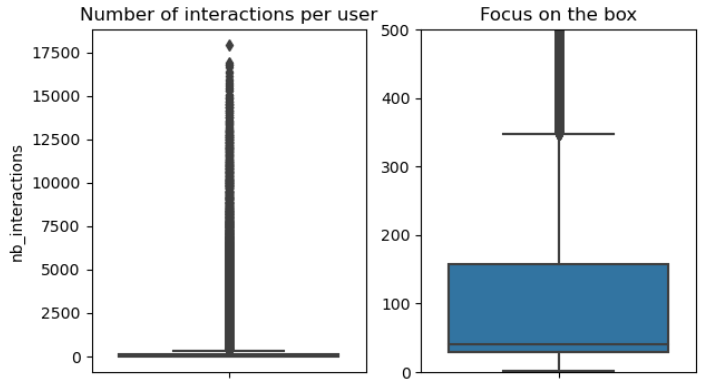

Figure 2 : Distribution du nombre d'interactions des utilisateurs

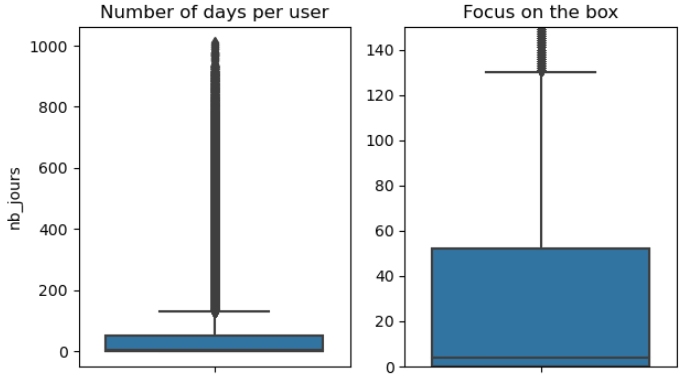

Figure 3 : Distribution nombre de jours des utilisateurs

En observant de manière précise la distribution des nombres de jours, entre la première et la dernière interaction, ainsi que la distribution du nombre d'interactions, je constate de nombreuses valeurs exceptionnelles.

Les informations me donnent une médiane à seulement 4 jours pour le nombre de jours. C'est-à-dire que 50 % des utilisateurs sont présents moins de 4 jours sur le site.
Les informations me donnent également une médiane à seulement 41 interactions, c'est-à-dire que 50 % des utilisateurs ont répondu à moins de 41 questions sur l'examen.
Cela m'indique qu'il y a de très nombreux utilisateurs avec très peu de mouvements, mais en observant de plus près, il y a également des utilisateurs de très longue date. Nous ne pouvons pas laisser ces utilisateurs il convient de les nettoyer.

J'ai décidé de conserver une fenêtre utilisateurs des percentiles 5 à 95, sur les deux variables nombre de jours et nombre d'interactions. De cette manière, cela me permet d'avoir un fichier utilisateurs un peu plus aéré tout en conservant leurs subtilités.

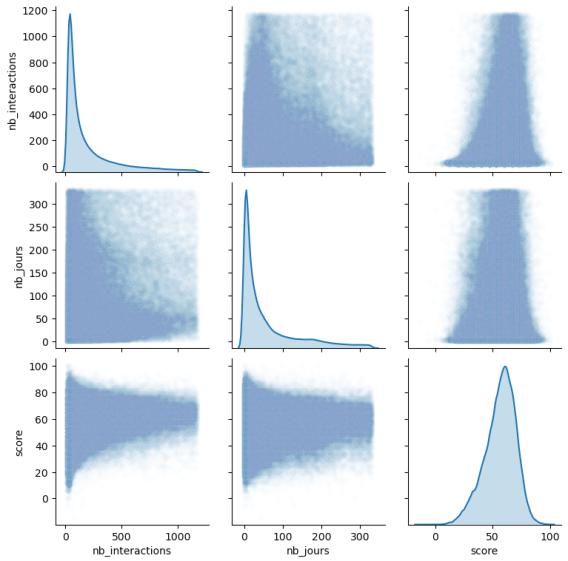

Figure 4 : Pairplot des valeurs agrégées des utilisateurs réduit

De cette manière, les utilisateurs sont réduits de 51% et les interactions sont réduites de 62 %

2.2. Features engineering

2.2.1. Usage des conférences

L'idée ici est de pouvoir se faire une idée du niveau d'utilisation des interactions dites "conference". Ce sont des passages avec explication, sans questions à destination de l'utilisateur.
Ces interactions sont très peu utilisées dans l'ensemble des données d'entrainements, environ 2 %.
En extrayant cette information, je dissocierai les utilisateurs faisant appel à ce type d'interaction, ayant probablement une plus grande progression dans son apprentissage.

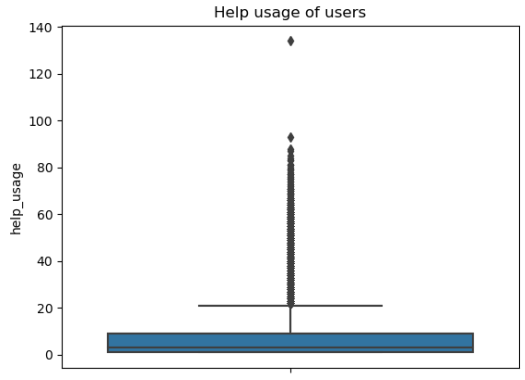

Figure 5 : Distribution des utilisateurs usant des conférences

Parmi les utilisateurs des conférences, 50 % des utilisateurs y font appel au moins 3 fois. Ce qui reste relativement faible.

Après avoir ramené la nouvelle variable sur l'ensemble du jeu de données, je l'ai ensuite binarisé de manière à constituer des catégories d'utilisateurs :

- 0 : Aucune utilisation des conférences dans le parcours utilisateur
- 1 : 1 utilisation des conférences
- 2 : 2 ou 3 utilisations de conférences
- 3 : 4 utilisations et plus des conférences

Les catégories se répartissent selon la catégories suivantes :

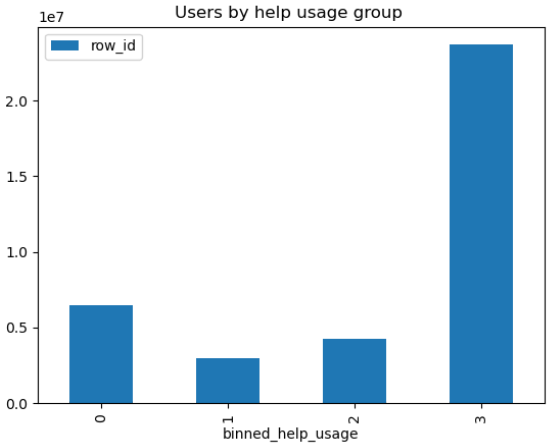

Figure 6 : Répartition des utilisateurs usant des conférences

Dans ce cas, il serait utile d'assembler les groupes 1 et 2 ainsi que diviser le groupe 3.

2.2.2. Difficulté du contenu

L'examen du TOEIC est découpé en deux familles : Listening and Reading. Chaque famille est composée de plusieurs parties ; 4 pour Listening et 3 pour Reading. Ces parties sont progressives dans leur niveau de difficulté.
Dans la figure ci-dessous, plus les couleurs sont faibles, plus le score est bas. De même pour les proportions, plus la surface est grande, plus la partie est usée par les utilisateurs.

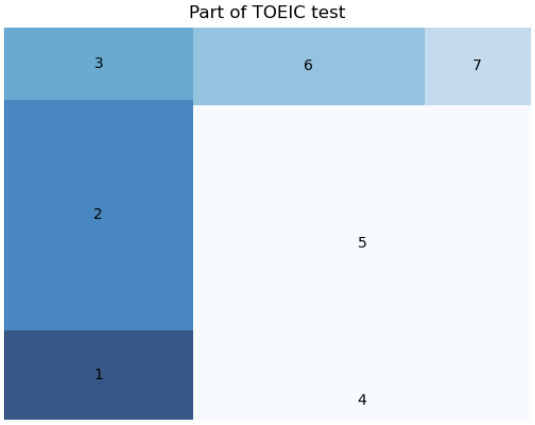

Figure 7 : Répartition des utilisations et des scores par parties de l'examen

La partie 5 est la plus sollicitée parmi les utilisateurs, mais également le moins réussie

2.3. Stratégie de validation

Je pense que la validation doit revêtir d'une stratégie particulière du fait du caractère temporel des données. D'une part par l'ordre d'exécution des interactions, mais aussi du fait que chaque partie est présentée ensemble, est dans un ordre bien précis.

Dans un cas réél, nous ne connaissons pas les résultats suivants, puisque les faits ne se sont pas déroulés. C'est pourquoi il me semble normal de tenir compte de ceci.

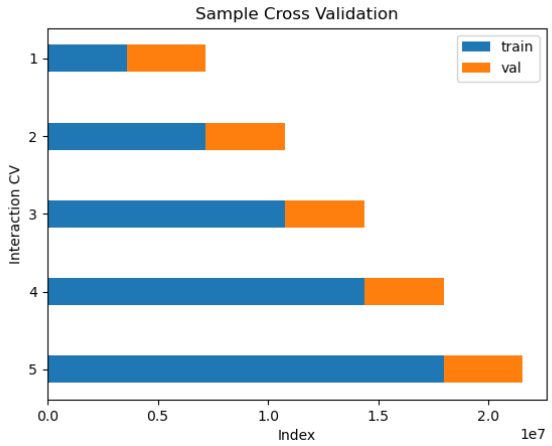

Figure 8 : Stratégie de cross validation envisagée

2.4. Modélisation

A titre d'expérimentation et suite à ces travaux, j'ai entraîné 2 modèles sur ces données, LGBM, et XGBoost.
Les deux entraînements, évalués sur la mesure AUC, ont donné les mêmes résultats, de l'ordre de 0.66. avec optimisation des hyperparamètres dans un grid search CV.

2.5. Retour de la communauté

J'ai soumis à la communauté ce travail préalable, de nettoyage, de features engineering, et de cross validation, au travers 2 notebooks :

- Cleaning users 💥 Features engineering
- Cross Validation Strategy with part of TOEIC Test

Le premier notebook a été approuvé par la communauté, il a été voté plusieurs fois et copié pour réutilisation à plusieurs reprises. Les différents commentaires ont contribué à l'amélioration du notebook.

Le second notebook a été moins bien reçu. Cela signifie que je me dirige certainement vers une moins bonne piste.

3. Amélioration d'un notebook existant

3.1. Compréhension de la démarche

Dans le domaine dans lequel on évolue, il est important de s'appuyer sur la communauté, et sur des utilisateurs qui ont un niveau plus avancé que soi.
De ce fait, j'ai choisi de partir sur un notebook de démarrage proposé par un membre et d'y appliquer mon travail de recherche, afin de l'améliorer.

J'ai choisi le notebook Riiid! LGBM Starter, mis à disposition par Sazuma. Le premier travail a été de comprendre sa démarche.

L'auteur a principalement appliqué de l'encodage cible, le jeu de données étant des variables catégorielles à modalités importantes. On note par exemple presque 400 000 utilisateurs, et environ 13 000 contenus. Un encodage à chaud est ici inadapté.

L'auteur créé 3 features :

- Taux de réussite propre à l'utilisateur,
- Taux de réussite propre au contenu,
- Le nombre de fois qu'un contenu apparaît.

L'auteur a choisi une validation simple, en choisissant les 24 dernières interactions d'un utilisateur, dont les 6 dernières servant de validation.
J'ai cherché pourquoi 24 interactions ? Cela correspond au nombre d'interactions minimale de 80 % des utilisateurs. Je peux facilement penser que l'auteur s'est inspiré de la loi de Pareto.

Lors du test via l'API, l'auteur simule aussi le tuteur IA qui concentrerait ses efforts sur l'utilisateur, en exploitant les données passées de test.

Par défaut, ce notebook part avec un résultat de 0.758 avec LGBM au leaderboard public. L'importance des features sont les suivantes.

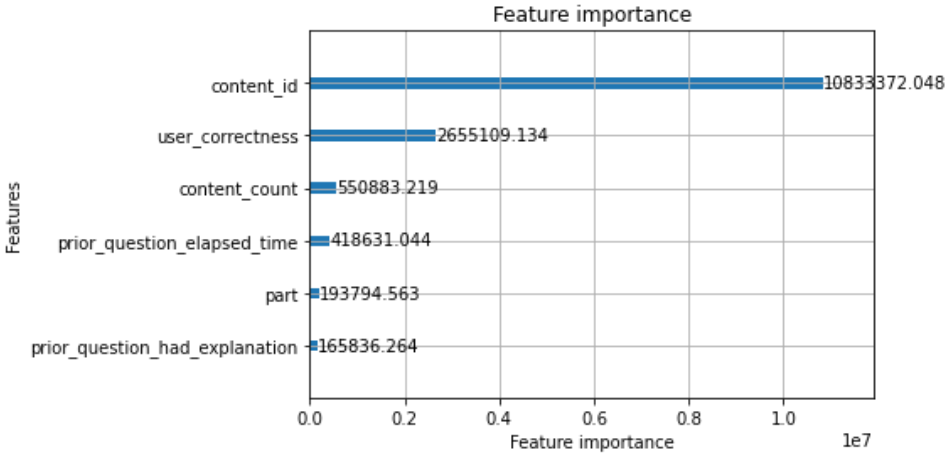

Figure 9 : Features importances sur le notebook original

Le résultat est prédominé par le taux de réussite par contenu et par le taux de réussite par utilisateur.
L'auteur conseille d'optimiser les paramètres du modèle.

3.2. Application du travail préalable

J'ai donc appliqué mon travail préalable à ce notebook public.
Si j'applique mon jeu de données, nettoyé, le résultat au leaderboard public est dévalué. Au-delà de mon nettoyage, l'auteur laisse dans les données les utilisateurs ayant moins de 24 interactions (données d'entrainements), y compris ceux qui ont moins de 6 interactions (données de validation). En les enlevant des données, je dévalue mon résultat au leader board public. Ces données auraient tendance à induire un surapprentissage. Je peux éventuellement les considérer comme des nouveaux utilisateurs auxquels l'algorithme doit s'adapter.
Le feature engineering appliqué à ce notebook n'apporte que très peu d'amélioration à ce modèle au leaderboard public. Je parle d'une évolution de 0.758 à 0.758 en valeur arrondie, que je n'ai pas pu mesurer sur des chiffres, mais sur des évolutions sur le classement.
En revanche, le résultat de validation en local évolue significativement, soit de 0.737 à 0.745. Ce sera une bonne option pour le leaderboard privé, qui concerne 80 % des données.
Je note que le target encoding a toujours une très forte influence sur le modèle, en dépit de mes propres features.

3.3. Modification de la stratégie de validation

L'auteur a décidé d'extraire ses données de validation après feature engineering. C'est une forme d'erreur, car l'auteur crée systématiquement une fuite de données, qui induit un sur apprentissage du modèle final.
Si cela n'est pas visible sur le leaderboard public, cela peut se traduire par des erreurs sur le leaderboard privé qui, je le rappelle, se réserve 80 % des données de test.

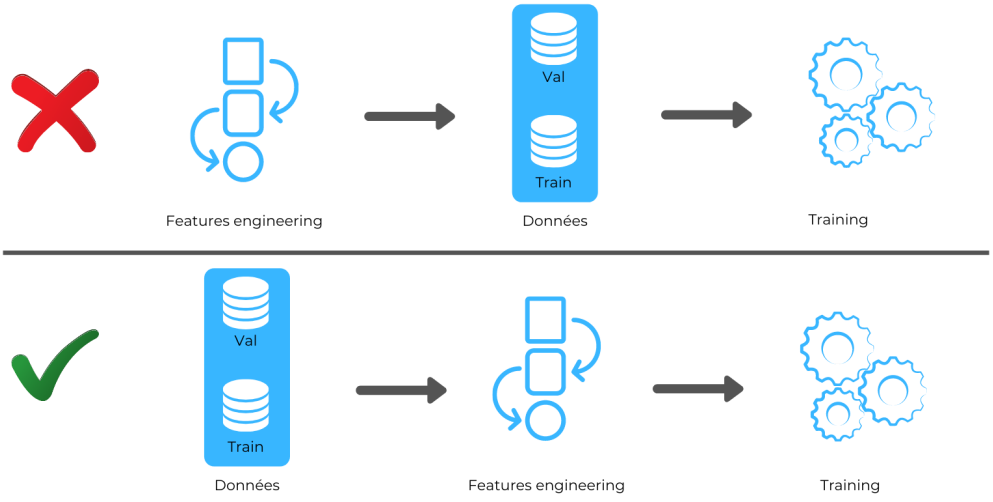

Figure 10 : Modification du split train/val

En procédant à la modification, je perds légèrement en précision sur la validation locale, et sur le leader board public (0.757 au lieu de 0.758). Il reste à savoir si le leaderboard privé me donnera raison.

La démarche est plus saine dans tous les cas.

3.4. Modélisation

Le modèle LGBM sera conservé. Mon étude préalable ne montre quasiment aucune différence entre LGBM et XGBoost sur ce jeu de données.

3.5. Optimisation des hyperparamètres

Les hyperparamètres ci-dessous ont été testés.

learning_rate	max_bin
0.01	127
0.05	800
	1600

Tableau 1 : Valeurs testées pour LGBM

max_bin règle le nombre maximal de regroupement de valeurs. Mettre un petit nombre réduit la précision et accélérer les temps d'entraînements. On réduira le learning_rate pour corriger les erreurs résiduelles, les temps d'entraînements seront plus long.

Après différentes test de combinaison, la paire learning_rate = 0.01 et max_bin = 800 s'avère la plus performante
L'utilisation de max_bin = 127 dégrade considérablement la performance du modèle.

4. Evolution vers un autre modèle

Malgré toutes mes démarches, le score au leaderboard public n'évolue pas significativement. Les scores en haut de podium dépassent les 0.80.

A la recherche d'un autre modèle, un sujet m'interpelle dans les discussions : SAINT+: A Transformer-based model for correctness prediction.

La discussion tourne autour d'un papier sorti mi octobre 2020, et dont les 6 auteurs font partie, Riiid! AI Research. Ce papier est sorti peu de temps après le démarrage de la compétition abordée dans ce document. Je rappelle que Riiid! est la société à l'origine de cette même compétition sur Kaggle.

Ce papier montre que le jeu de donnée Kaggle pour la compétition, est le jeu de données en provenance de ce papier réduit. Le EdNet-KT1 dataset, comporte 600 millions d'interactions pour près de 680 000 utilisateurs. Pour le jeu de données réduit Kaggle, nous sommes à 100 millions d'interactions pour presque 400 0000 utilisateurs.

Cette équipe de recherche obtient un score de 0.78 / 0.79 avec leur modèle SAINT+. Par déduction, il est fort probable que les compétiteurs dépassant ce score utilisent un modèle SAINT+.

On trouve plus d'informations sur le site officiel.

Pour étudier la faisabilité, voici Demystifying Transformers, let's make it public pouvant aider au démarrage de la modélisation sous Keras.