Using Machine learning for Riiid Answer Correctness Prediction With LGBM Algorithm

Use news analytics to predict Answer Correctness of Riiid Database

ARJANE Khadija
Étudiante à l'école nationale des sciences appliquées d'El Jadida-2ITE
arjane.khadija97@gmail.com

CHARHABIL Sanaa
Étudiante à l'école nationale des sciences appliquées d'El Jadida-2ITE
charhabil.sanaa98@gmail.com

Mr. ERRATAHI
Professeur à l'école Nationale des sciences appliquées d'El Jadida et Encadrant du Projet
errattahi.r@ucd.ac.ma

Résumé- **Riid Answer correctness ou La prévision des réponses justes et correctes est un sujet important et florissant en éducation. En effet, l'éducation était déjà dans une situation difficile lorsque le COVID-19 a forcé la plupart des pays à fermer temporairement des écoles. Cela a encore retardé les opportunités d'apprentissage et le développement intellectuel. Nous devons repenser le système éducatif actuel en termes d'assiduité, d'engagement et d'attention individualisée. Notre travail en tant que des data scientistes est de prévoir la capacité que chaque élève de répondre correctement à une série de questions en se basant sur un ensemble de dataset de Riiid en utilisant des algorithmes de classification des données supervisées tel que LGBMClassifier qui donne un meilleur score de précision et les soumissions dans cette compétition sont évaluées sur la zone sous la courbe ROC entre la probabilité prédite et la cible observée.**

Mots clés: Riid, covid-19, LGBMClassifier, ROC

I. INTRODUCTION ET CONTEXT :

Le Machine Learning est une technique de programmation informatique qui utilise des probabilités statistiques pour donner aux ordinateurs la capacité d'apprendre par eux-mêmes sans programmation explicite. Pour apprendre et se développer, les ordinateurs ont toutefois besoin de données à analyser sur lesquelles ils vont s'entraîner. De fait, le Big Data est l'essence du Machine Learning, et c'est la technologie qui permet d'exploiter pleinement le potentiel du Big Data.

Dans ce projet, nous allons participer à une compétition sur la plateforme kaggle qui consiste à prévoir la capacité que chaque élève de répondre correctement à une série de questions, en se basant sur un ensemble d'input (lectures.csv, questions.csv, train.csv…).

A. Objectif :

Notre objectif est de prédire si les élèves sont capables de répondre correctement à leurs prochaines questions selon leurs niveaux de compétences et en suivant leurs progressions dans les formations. Donc nous recevrons les mêmes types d'informations qu'une application éducative complète aurait : la performance historique de cet élève, la performance des autres élèves sur la même question, des métadonnées sur la question elle-même, et plus encore. Le problème réside dans la quantité de données à traiter, qui occupe un grand espace mémoire, environ 101230332 enregistrements par dataset et le traitement concerne 393656 d'utilisateurs uniques.

Donc on aura toujours des difficultés au niveau de traitement, ainsi que les requêtes lancées seront assez lentes et ça pourra générer également des prédictions non précisées. Par conséquent, on propose comme solution de traiter juste une partie de notre dataset à cause de la limite de RAM sur le noyau Kaggle. Quel est le pourcentage à utiliser ?

Test du dataset avec une taille équitable :

POURCENTAGE	NOMBRE D'ENREGISTREMENTS CORRESPONDANTES
100%	101230332 enregistrements.
90 %	91 107 299 enregistrements
80 %	80 984 265 enregistrements.
70%	70 861 232 enregistrements.
60%	60 738 199 enregistrements
50%	50 615 166 enregistrements
40%	40 492 133 enregistrements.
30%	30 369 100 enregistrements.
20%	20 246 066 enregistrements.
10%	10 123 033 enregistrements
5%	5 061 516 enregistrements.

Table 1 : Différent pourcentage des enregistrements

Pour 100% de dataset jusqu'à 10%, Nous n'arrivons pas à exécuter le notebook parce qu'il alloue plus de mémoire que disponible. On se contentera d'utiliser juste 10% de training set. c.-à-d. 10123033 enregistrements.

Réalisons nos prédictions en affectant au paramètres le nombre d'enregistrements à traiter, la sélection de ce nombre sera faite d'une manière aléatoire.

Concernant les données, on va utiliser la famille des algorithmes de classification, parce que nous avons le supervised Learning avec des données discrètes non continues.

II. EXPLORATION ET COLLECTION DES DONNÉES

La phase d'exploration des données est très importante, elle permet d'obtenir une vue sur l'état des données. Plusieurs problèmes peuvent être détectés pendant cette étape et les méthodes de correction correspondantes, appliquées.

Les données de dataset sont des données numériques et alphanumériques partitionnées sous des fichiers csv.
Notre dataset de training contient :

- row_id: (int64) ID de la ligne.
- timestamp: (int64) le temps en millisecondes entre cette interaction de l'utilisateur et la fin du premier événement de cet utilisateur.
- user_id: (int32) code d'identification (int32) de l'utilisateur.
- content_id: (int16) code d'identification (int16) pour l'interaction utilisateur.
- content_type_id: (int8) 0 si l'événement était une question posée à l'utilisateur, 1 si l'événement était l'utilisateur qui regardait une conférence.
- task_container_id: Code d'identification pour le lot de questions ou de conférences. Par exemple, un utilisateur peut voir trois questions d'affilée avant de voir les explications de l'une d'entre elles. Ces trois éléments partageaient tous un task_container_id.
- user_answer: (int8) la réponse de l'utilisateur à la question, le cas échéant. Lisez -1 comme nul, pour les conférences.
- answered_correctly: (int8) si l'utilisateur a répondu correctement. Lisez -1 comme nul, pour les conférences.
- prior_question_elapsed_time: (float32) Le temps moyen en millisecondes nécessaire à un utilisateur pour répondre à chaque question du groupe de questions précédent, en ignorant toutes les conférences intermédiaires. S'il est nul pour la première série de questions ou la première conférence d'un utilisateur. Notez que le temps est le temps moyen mis par un utilisateur pour résoudre chaque question du groupe précédent.
- prior_question_had_explanation: (bool) Indique si l'utilisateur a vu ou non une explication et la ou les bonnes réponses après avoir répondu à la série de questions précédente, en ignorant toutes les conférences intermédiaires. La valeur est partagée dans un seul groupe de questions et est nulle pour le premier groupe de questions ou la première conférence d'un utilisateur. En règle générale, les premières questions qu'un utilisateur voit faisaient partie d'un test de diagnostic d'intégration dans lequel il n'a obtenu aucun commentaire.

Rappelons que notre objectif est de prédire answered_correctly de chaque user_answer. Commençons d'abord de faire quelques visualisations au niveau de ces deux colonnes et quelle est la relation entre eux, On pourra par exemple visualiser :

a - la fréquence de *answered_correctly*:

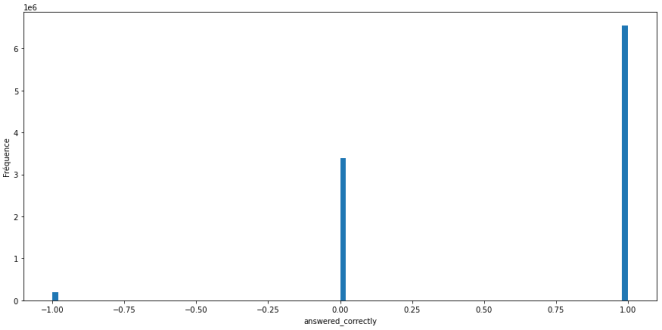

Figure 1 : La fréquence de la colonne answer_correctly

b - la fréquence de *user_answer* :

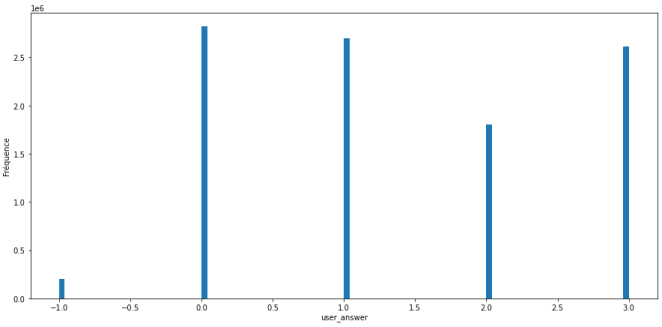

Figure 2 : La fréquence de la colonne user_answer

c - la relation entre *user answer* et *answered correctly*:

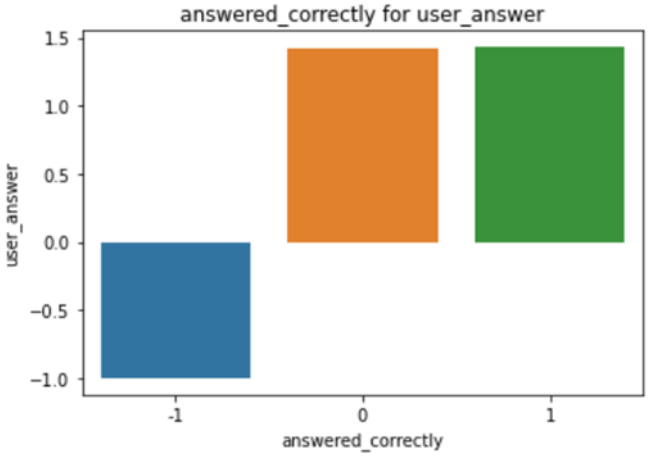

Figure 3 : La relation entre user_answer et answered_correctly

D'après ce plot, on peut dire qu'il existe une relation entre *answer_correctly et user_answer* et qu'ils sont dépendants.

III. PRÉTRAITEMENT ET NETTOYAGE DES DONNÉES

Le prétraitement et le nettoyage des données sont des tâches importantes qui doivent intervenir avant d'utiliser un jeu de données pour la formation de modèles. Les données brutes sont souvent bruyantes, peu fiables et incomplètes :

- *Caractère incomplet :* des valeurs ou des attributs sont manquants.
- *Bruit :* les données contiennent des enregistrements erronés ou des aberrations.
- *Incohérence* : les données contiennent des enregistrements en conflit ou des contradictions.

Par conséquent, leur utilisation pour la modélisation peut générer des résultats trompeurs. Ces tâches font partie du processus TDSP (Team Data Science Process) et suivent généralement l'exploration initiale d'un jeu de données utilisé pour découvrir et planifier le traitement préliminaire requis. Parmi les opérations qu'on peut appliquer lors du prétraitement des données :

- *Nettoyage de données :* recherche et traitement des données manquantes et brouillées.
- *Transformation des données :* L'objectif est de normaliser les données pour réduire le volume et le bruit.
- *Réduction des données :* Il faut échantillonner les enregistrements de données ou les attribuer pour faciliter la manipulation des données.

Mais il faut faire attention aux changements qui seront attribués par suite aux datasets, car cette étape sera réalisée une seule fois et n'importe quelle faute appliquée sur les données input influencera l'output ou le résultat final.

La première étape qu'on va appliquer sur l'ensemble des enregistrements qu'on possède c'est la recherche des données NAN, ou les données manquantes via la fonction **isnull**():

Quelles seront ces colonnes qui contiennent ce de valeurs ?

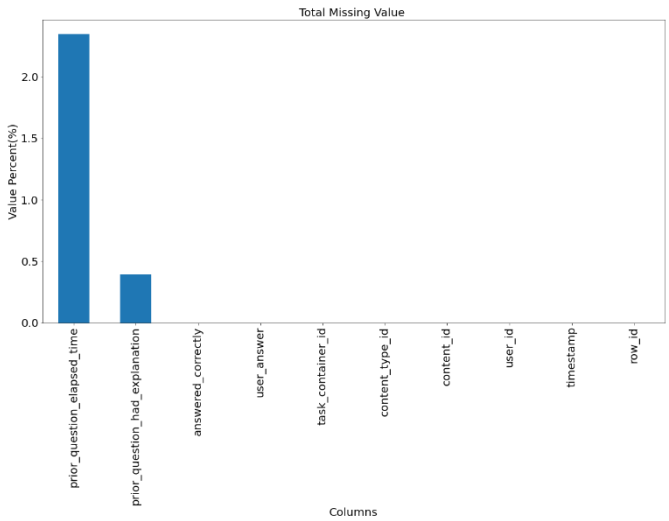

Figure 4 : Le pourcentage des valeurs nulles

Le nombre de valeurs manquantes par colonne :

La colonne	Le nombre de valeurs nulles correspondant
row_id	0
timestamp	0
user_id	0
content_id	0
content_type_id	0
task_container_id	0
task_container_id	0
user_answer	0
answered_correctly	0
prior_question_elapsed_time	237438
prior_question_had_explanation	39821

Figure 5 : Nombre de valeurs nulles

Puisqu'on a des valeurs manquantes au niveau des deux dernières colonnes, la première chose à faire est d'en remplacer ces valeurs en examinant les méthodes les plus courantes de traitement des valeurs manquantes :

A- suppression des données manquantes :

Supprimer les lignes qui contiennent des valeurs nulles avec la fonction **dropna()** qui permet aussi de réduire la taille du dataset en gros ce qui sera un avantage.

B - Remplacement par une valeur factice :

Remplacer des valeurs manquantes par une valeur factice. Par exemple, inconnu pour les valeurs catégorielles ou 0 pour les valeurs numériques. Le remplacement des données numériques manquantes par 0, ça peut affecter nos prédictions.

C - Remplacement des données manquantes par la valeur la plus proche en avant ou en arrière.

Cette méthode est efficace quand il s'agit des données numériques qui sont plus proches, c.-à-d. qui sont dans un intervalle précis.

D - Remplacement par la moyenne :

Si les données manquantes sont numériques, les remplacer par la valeur moyenne.

E - Remplacement par l'élément le plus fréquent :

Si les données manquantes sont catégorielles, remplacer les valeurs manquantes par l'élément le plus fréquent. Il faut par exemple compter le nombre d'occurrence pour chaque catégorie par la fonction **value_counts()** :

Figure 6 : Recherche de la valeur dominante

Et remplacera donc les champs nuls par la valeur True avec **fillna()**.

Cette méthode n'est pas pratique car il y une corrélation entre les deux colonnes qui contient des données vides.

F - Remplacement par la valeur de régression :

Remplacer les valeurs manquantes par des valeurs de régression. Dans ce sens, il faudra supprimer d'abord la colonne cible ou bien la rendre numérique afin d'exécuter la commande **interpolate**().

Suivi par l'étape de la normalisation qui consiste à restreindre les valeurs numériques à une plage spécifiée. Les méthodes de normalisation les plus courantes sont les suivantes :

- *Normalisation min-max :* adapter linéairement les données à une plage comprise, par exemple, entre 0 et 1. La valeur minimale est 0 et la valeur maximale est 1.
- *Normalisation par le test Z :* mettre les données à l'échelle en fonction de la moyenne et de l'écart standard : diviser la différence entre les données et la moyenne par l'écart standard.
- *Mise à l'échelle décimale :* mettre les données à l'échelle en déplaçant le séparateur décimal de la valeur de l'attribut.

Mais dans notre cas, les données qu'on a sont normalisées, en effet la majorité des colonnes qu'on est censée traiter se trouve dans une plage donnée et précise on peut clairement visualiser ceci via la fonction suivante :

	row_id	timestamp	user_id	content_id	content_type_id	task_container_id
count	9885595.000000	9885595.000000	9885595.000000	9885595.000000	9885595.000000	9885595.000000
mean	5061711.677613	7569828177.572103	109339453.072281	4977.612804	0.000000	887.767714
std	2922337.001732	11404930292.454985	63881722.815432	3286.962608	0.000000	1348.642885
min	1.000000	0.000000	115.000000	0.000000	0.000000	0.000000
25%	2531587.500000	516199262.000000	54713154.000000	1987.000000	0.000000	103.000000
50%	5061647.000000	2636836859.000000	108451478.000000	4978.000000	0.000000	375.000000
75%	7592639.500000	9842526710.000000	165342103.000000	7217.000000	0.000000	1066.000000
max	10123032.000000	84707934191.000000	219543270.000000	13522.000000	0.000000	9999.000000

Figure 7 : Description du dataset.

Et finalement, il faut réduire la taille des données pour en faciliter la manipulation. Selon la taille et le domaine, les méthodes applicables sont les suivantes :

- *Échantillonnage des enregistrements :* échantillonner les enregistrements de données et ne choisir que le sous-ensemble représentatif.
- *Échantillonnage des attributs :* ne sélectionner que les attributs importants dans les données.
- *Agrégation* : diviser les données en groupes et stocker les nombres de chaque groupe. Par exemple, le chiffre d'affaires quotidien d'une chaîne de restaurants sur les 20 dernières années peut être agrégé en un chiffre d'affaires mensuel pour réduire la taille des données.

Après le test de tous les méthodes qu'on a appliquée comme solution pour remplacer/supprimer les données manquantes on trouve que la fonction **dropna()** est la méthode la plus efficace en tant de performance, en effet il y a une corrélation entre les deux colonnes qui contiennent les données nulles, , donc on ne peut pas remplacer les champs nulles de la colonne par n'importe quelle valeur car il va influencer le résultat final de nos prédictions.

IV. FEATURES ENGINEERING

Features engineering ou L'ingénierie des fonctionnalités est le processus d'utilisation des connaissances du domaine pour extraire des fonctionnalités à partir de données brutes via des techniques d'exploration de données. Ces fonctionnalités peuvent être utilisées pour améliorer les performances des algorithmes d'apprentissage automatique.

Quelles sont les attributs qui sont en corrélation avec la cible et qu'on peut les utiliser dans les modèles ?

Dans ce sens, on peut calculer la corrélation de toutes les colonnes avec la colonne cible en utilisant la fonction **corr**() en ordre descendant comme suit:

La colonne	La valeur de corrélation avec la colonne cible
row_id	0.005673
timestamp	0.031453
user_id	0.005571
content_id	-0.023037
content_type_id	-0.439740
task_container_id	0.062918
user_answer	0.005889
answered_correctly	1
prior_question_elapsed_time	0.0070016
prior_question_had_explanation	0.263946

Figure 8 : valeur de corrélation.

On peut aussi utiliser la même fonction et tracer un plot :

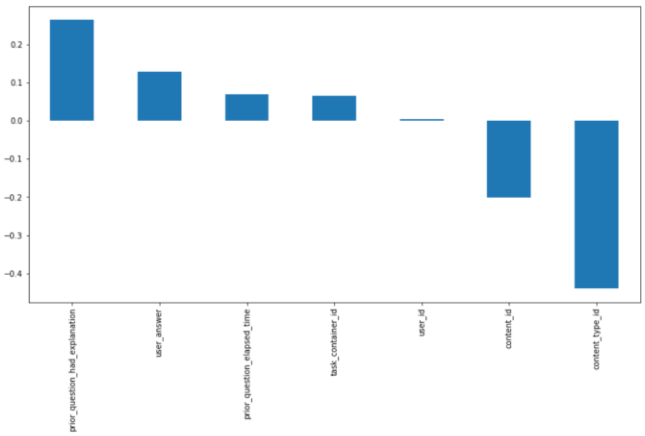

Figure 9 : histogramme de corrélation.

Les colonnes qui sont en corrélation avec **answer_correctly** sont ceux qui ont une valeur positive dans le schéma : *'answered_correctly','user_id','content_id','task_container_id','user_answer','prior_question_elapsed_time','prior_question_had_explanation'.*

On reteste la corrélation entre ces features et la colonne cible via la matrice de corrélation

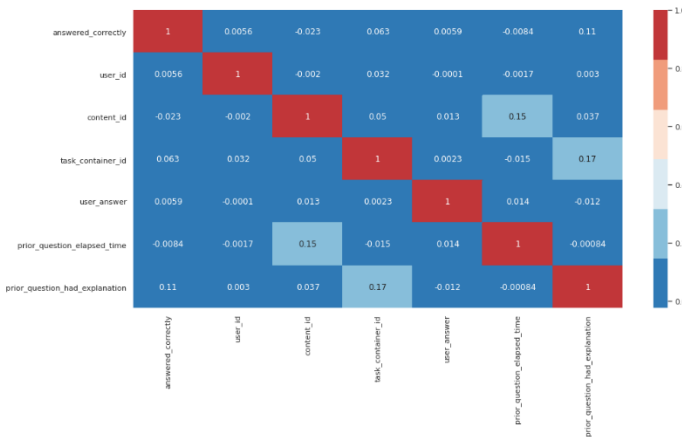

Figure 10 : Matrice de corrélation

Dans cette approche Features Engineering, on a essayé de garder les caractéristiques les plus pertinentes et les plus utiles dans via une étude de corrélation pour les données séparées, puis on va ajouter d'autres colonne et fonctionnalités qu'on trouve assez pertinentes puis on va appliquer des modèles afin de savoir l'importance des Caractéristiques.

Pour chaque *user_id*, on va calculer la moyenne, la médiane, écart-type, l'asymétrie et le nombre de valeurs, en utilisant les fonctions suivantes :

- **mean():** pour calculer la moyenne.
- **median():** pour le calcul de la médiane.
- **std():** utilisée pour calculer l'écart type le long de l'axe spécifié. Cette fonction renvoie l'écart type des éléments du tableau. La racine carrée de l'écart carré moyen (calculé à partir de la moyenne) est appelée écart type.
- **skew():** fonction pour mesurer l'asymétrie. Lorsqu'une distribution est asymétrique, la queue de la distribution est inclinée d'un côté vers la droite ou vers la gauche. Lorsque la valeur de l'asymétrie est négative, la queue de la distribution est plus longue vers le côté gauche de la courbe.
- **count():** pour compter un nombre précis.

On a réalisé cette étape afin de visualiser et mettre en relation les caractéristiques de la colonne cible pour chaque utilisateur, puis on a enregistré le résultat dans un nouveau dataframe ***user_characteristics.***

De même, mais cette fois-ci, on va regrouper les caractéristiques et les infos de la colonne cible par ***task_container_id***.

Maintenant, on va effectuer une jointure entre le dataset qu'on avait avant train et nouveaux datasets qu'on a créé, en utilisant la fonction **merge()**.

Features
prior_question_elapsed_time
prior_question_had_explanation
mean_user_acc
std_user_acc
skew_user_acc
number_of_answered_q
mean_task_acc
median_task_acc
std_task_acc
skew_task_acc
number_of_asked_task_containers
mean_acc
median_acc
std_acc
skew_acc
number_of_asked_q

Figure 11 : liste des features utilisés

V. ML MODELS

Donc on vient de terminer la collection, la préparation des données et features engineering. Mais quels sont donc les modèles à utiliser en machine Learning afin de réaliser les prédictions ? et quels sont les algorithmes de classification à appliquer ? et le modèle le plus performant ?

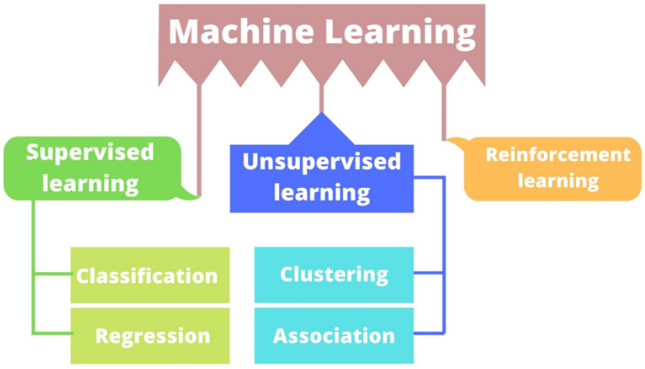

Figure 12 : Classification des ML algorithmes

Puisque notre travail consiste à guider l'algorithme sur la voie de l'apprentissage en lui fournissant des exemples des réponses anciennes qu'il estime probants après les avoir préalablement étiquetés des résultats attendus. Donc en résumé on sera basé sur l'apprentissage supervisé. Et puisqu'on a des données non continues, par conséquent tous les modèles de machine Learning qu'on va appliquer par suite seront des modèles de classification, en effet :

- On dispose d'un ensemble de données connues qui sont déjà classées (answer_correctly, answer ...).
- On souhaite, à partir de cette première classification, dite connaissance, classer de nouveaux éléments.

La première étape à faire avant l'application des algorithmes et modèles de classification est le splitting des données aléatoirement en utilisant la fonction **train_test_split()**. Nous avons respecté la notion de 70% pour le training et 30% pour le test.

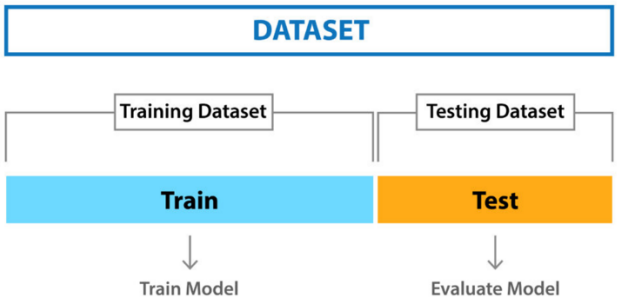

Figure 13 : Split de dataset

Les modèles de machine Learning utilisés sur ce dataset sont :

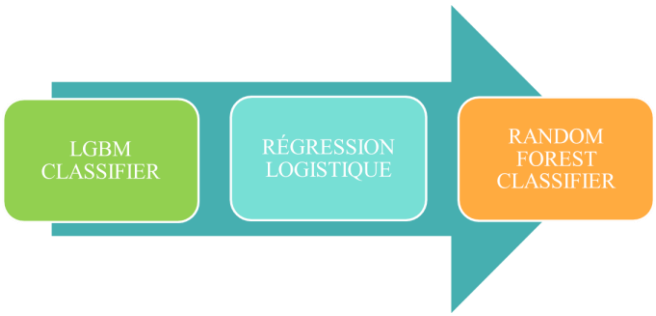

Figure 14 : les algorithmes utilisés

a) MODÈLE 1 : LGBM CLASSIFIER

LightGBM est un cadre de renforcement de gradient rapide, distribué et haute performance basée sur des algorithmes d'arbre de décision, utilisé pour le classement, la classification et de nombreuses autres tâches d'apprentissage automatique pour les avantages suivant :

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

Après avoir réalisé le Paramètres Tuning, nous obtenons les listes suivantes, comme étant des valeurs des paramètres propres à l'algorithme, qui sont définis dans l'étape précédente. Ce qui nous permet d'obtenir le dictionnaire de paramètre suivant d'après l'utilisation de **randomized Grid search**.

Hyperparamètre de LGBM Classifier
num_leaves: 30
n_estimators: 300
min_data_in_leaf: 100
max_depth: 5
lambda: 0.0
feature_fraction: 1.0

Figure 15 : les hyperparamètres de LGBMClassifier

On va utiliser ELI5 pour visualiser pour chaque Features le poids correspondant en employant la fonction **show_weights().** Eli5 est un package Python qui aide à déboguer les classificateurs d'apprentissage automatique et à expliquer leurs prédictions. Il prend en charge les frameworks et packages d'apprentissage automatique suivants : scikit-learn.

Weight	Feature
0.5617	mean_acc
0.2045	skew_user_acc
0.0731	skew_acc
0.0586	std_acc
0.0377	mean_user_acc
0.0123	std_user_acc
0.0112	number_of_answered_q
0.0108	number_of_asked_q
0.0104	median_acc
0.0080	prior_question_elapsed_time
0.0039	std_task_acc
0.0028	number_of_asked_task_containers
0.0024	mean_task_acc
0.0014	skew_task_acc
0.0008	prior_question_had_explanation
0.0006	median_user_acc
0.0000	median_task_acc

Figure 16 : Le poids des features

Visualiser l'importance de chaque feature en appliquant la fonction **plot_importance()**:

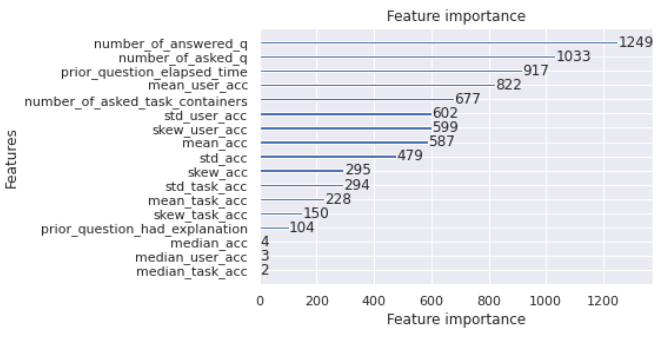

Figure 17 : importance des features avec plot importance

Ceci montre l'utilité des colonnes qu'on a ajouté, elles occupent toujours les grandes valeurs d'importance.

b) MODÈLE 2 : RÉGRESSION LOGISTIQUE

La régression logistique est semblable à la régression linéaire, mais elle est utilisée pour modéliser la probabilité d'un nombre fini de résultats, généralement deux. Il y a plusieurs raisons pour lesquelles la régression logistique est utilisée par rapport à la régression linéaire lors de la modélisation des probabilités de résultats. La régression logistique ou modèle logit est un modèle de régression binomiale. Comme pour tous les modèles de régression binomiale, il s'agit de modéliser au mieux un modèle mathématique simple à des observations réelles nombreuses.

Appliquons cet algorithme sur nos données, en utilisant la fonction ***LogisticRegression***() pour la création du modèle et la méthode fit() qui consiste à réaliser un processus automatique qui garantit que le modèle d'apprentissage automatique a les paramètres individuels les mieux adaptés pour résoudre le problème spécifique de prédiction avec un haut niveau de précision.

Par la suite, nous avons créés la matrice de confusion qui va nous permettre de mesurer les performances d'un modèle de Machine Learning en vérifiant notamment à quelle fréquence ses prédictions sont exactes par rapport à la réalité dans des problèmes de classification.

La précision du Logistic Régression sur l'ensemble de test est de 0,61 et de 69.2% prédictions correctes.

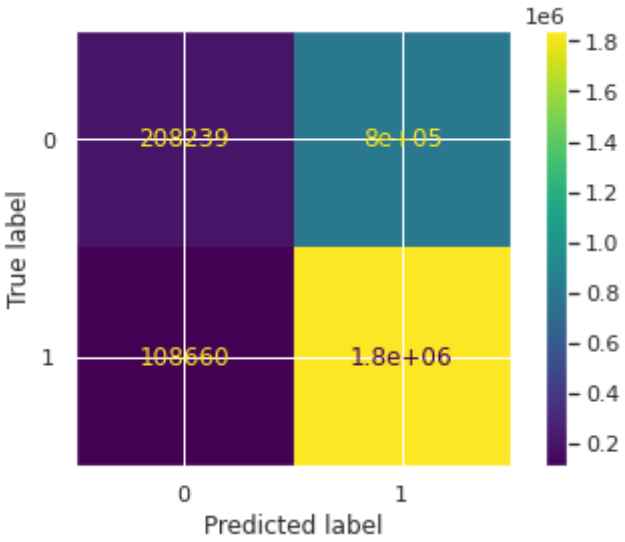

Figure 18 : Matrice de confusion LogisticRegression

La fonction d'efficacité du récepteur, plus fréquemment désignée sous le terme « courbe ROC » est une mesure de la performance d'un classificateur binaire, c'est-à-dire d'un système qui a pour objectif de catégoriser des éléments en deux groupes distincts sur la base d'une ou plusieurs des caractéristiques de chacun de ces éléments.

Figure 19 : La courbe ROC LogisticRegression

a) MODÈLE 3 : RANDOM FOREST CLASSIFIER

Random Forest est un algorithme d'apprentissage automatique robuste qui peut être utilisé pour une variété de tâches, y compris la régression et la classification. Il s'agit d'une méthode d'ensemble, ce qui signifie qu'un modèle forestier aléatoire est composé d'un grand nombre de petits arbres de décision, appelés estimateurs, qui produisent chacun leurs propres prédictions

La précision du Random Forest classifier sur l'ensemble de test est de 0,71 et de 71% prédictions correctes.

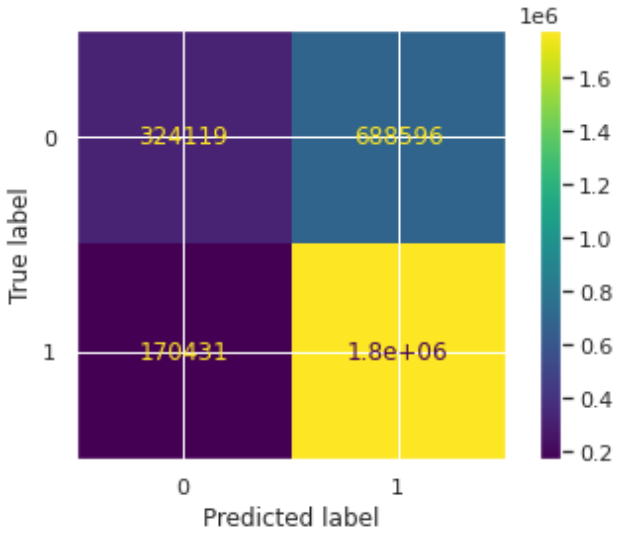

Figure 20 : Matrice de confusion Random Forest

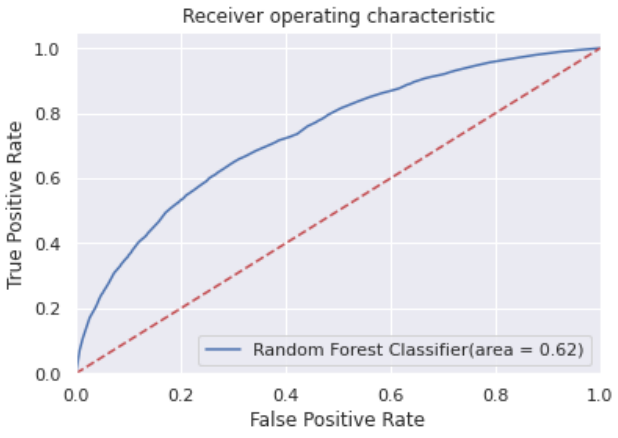

Figure 21 : La courbe ROC Random Forest

VI. RESULTATS

Suite à notre analyse de données et à nos expérimentations vis-à-vis les algorithmes susceptibles de donner de bons résultats, nous avons pu réaliser une soumission sur Kaggle avec LGBMClassifier où nous avons utilisé trois modèles différents, ce qui nous a permis d'avoir les scores représentés dans le tableau suivant :

Algorithme et Méthodes	Score
LGBMClassifier avec Hyperparametre	**0.76**
Logistic Regression	**061**
Random Forest	**071**

Table 2 : Score pour chaque soumission

Le code de notre soumission réalisée se trouve sur le lien suivant :
https://www.kaggle.com/khadijaarjane/notebookriiid-arjane-charhabil

L'algorithme que nous avons utilisé est également utilisé par la plupart des compétiteurs, mais grâce au rassemblement des résultats de plusieurs Paramètres Tuning pour notre modèle, ajoutant à cela la prise en considération de 10% de Dataset utilisée pour la modélisation et nous avons pu obtenir de bons résultats.

VII. CONCLUSION

La prévision des réponses correctes est une tâche ardue car les cours des actions sont influencés par de nombreux facteurs. Cet article présente les étapes suivies afin de réaliser des prédictions dans le domaine éducatif, en se basant sur des données. La contribution de cette étude peut être résumée comme suit.

Premièrement nous avons appliqué des méthodes d'exploration de données (EDA) pour avoir une idée sur les données et les préparer pour l'étape suivante en essayant de les nettoyer et remplacer les données manquantes de plusieurs manières.

Deuxièmement, on a réalisé le Features Engineering pour ne garder que les caractéristiques importantes pour notre étude, et nous avons appliqué les algorithmes de classification comme Régression Logistique, LGBM classifier, Random Forest Classifier..., et comme l'illustre la figure précédente le meilleur score 0.76 correspond au Light Gradient Boosting et cela après avoir appliqué le Paramètres Tuning.

En guise, la participation à cette compétition avait plusieurs avantages, en fait, on a découvert en tant que binôme la plateforme kaggle, ainsi qu'on a appris des méthodes à appliquer en machine Learning ce qui a été vraiment bénéfique, et ça sera la clé d'ouverture vers la participation vers d'autres compétition à l'avenir.

RECONNAISSACE

Nous sommes reconnaissants parce que nous avons réussi à compléter notre sujet intitulé « Using Machine Learning to Predict Answer Correctness of Riiid With LGBM Algorithm » dans les délais impartis. Cette tâche ne peut être accomplie sans l'effort et la coopération des membres de notre groupe : Arjane Khadija et Charhabil Sanaa.

Nous remercions également notre encadrant professeur Rahhal ERRATTAHI pour les conseils et les encouragements ainsi que le partage de son expérience et ses compétences. Et notre professeur Fahd Kalloubi pour le soutien, les connaissances partagées et la coordination de ce module au choix.

Nous adressons aussi nos sincères remerciements à toute l'équipe pédagogique de l'Ecole Nationale des Sciences Appliquées d'El Jadida (ENSAJ) et les intervenants professionnels responsables de la filière Ingénierie Informatique et Technologies Emergentes (2ITE).

RÉFÉRENCES

[1] https://www.kaggle.com/c/riiid-test-answer-prediction

[2] https://www.kaggle.com/khadijaarjane/notebookriiid-arjane-charhabil

[3] https://www.kaggle.com/c/riiid-test-answer-prediction/data

[4]https://github.com/KhadijaArjane97/Riiid-Answer-Correctness-Prediction-Kaggle