CHEVENET Jules
MEUNIER François

Rapport concours Kaggle :

Introduction :

Dans le cadre de l'unité **4301A - Apprentissage automatique 2**, nous avons effectué un concours Kaggle (https://www.kaggle.com/c/dsia-spring2021). L'objectif est de **prédire la note** sur dossier, obtenue par un candidat pour l'entrée dans une école d'ingénieur, à l'aide des données déjà fournies (moyennes, appréciations, ...). Un **ensemble d'entraînement** était à notre disposition, il contenait le score à prédire et les données d'entrée des candidats, son but est de permettre de prédire les notes d'un **ensemble de test** en fonction des données fournies.
Le classement Kaggle est effectué sur le **RSME** entre les scores prédits de l'ensemble de test et les scores réels (inconnus pour nous).

Afin de mettre en place un environnement de travail collaboratif entre le binôme, nous avons travaillé sur Google Colab à partir d'un Google Drive partagé.
Adresse du drive :https://drive.google.com/drive/u/0/folders/0ACXNIhf8Is6nUk9PVA
Le notebook contenu dans le drive explique plus en détail l'environnement utilisé. Ce rapport explique uniquement les moyens mis en place pour effectuer notre meilleure prédiction.

I/ Prétraitement :

Les données ont été manipulées sur Python via les packages pandas et numpy.
Dans un premier temps, nous avons retiré les colonnes du dataframe qui étaient intégralement vides (soit 102 colonnes) dans la base d'entraînement ou dans la base de test, en effet ces colonnes n'avaient aucune utilité.
La phase de prétraitement des données à été mise en place, car elle est nécessaire pour le futur traitement puis pour les futures prédictions sur les dataframes. A la fin de la phase de pré-traitement, le dataframe d'entraînement et le dataframe de test ont donc exactement les mêmes colonnes.

II/ Traitement :

1/ Premier nettoyage :

Le nettoyage est le même pour les deux dataframes.
Pour commencer le nettoyage, les **variables ordinales** se sont vu attribuer une valeur entière. En fonction de la variable, les valeurs étaient comprises entre 0 et le nombre de valeurs possible - 1. Si la valeur n'est pas connue, elle est remplacée par 0.

Exemple pour la variable 'Méthode de travail' :

Nan	0
Peu démontrée	0
Assez satisfaisante	1
Satisfaisante	2
Très satisfaisante	3

De plus, les **variables nominales** ("Genre" et "Bac") sont transformées, en effet, les deux variables étant binaires (bac S/bac STI2D et Homme/Femme), "Genre" devient "Genre_H" et "Bac" devient "Bac_scientific". Les valeurs dans les colonnes deviennent alors 1 pour oui ou 0 pour non.
Les variables inexploitables ("id", "Nom", "Prénom") sont supprimées des dataframes.

2/ Gestion des variables numériques :

Il reste alors à gérer le cas des variables numériques. On peut les diviser en deux parties : d'une part les notes aux épreuves anticipées de français et d'autre part les moyennes.
Dans un premier temps, il était obligatoire de remplir les valeurs manquantes. En effet, comment gérer ces valeurs ? Après différents tests, les meilleurs résultats étaient obtenus lorsque les valeurs manquantes étaient remplacées par zéro.
Nous avons donc pris la décision de **remplacer toutes les valeurs manquantes par zéros**.
Aucun traitement supplémentaire n'a été effectué sur les notes aux épreuves anticipées de français.

Cependant, la gestion des moyennes pose de réelles questions. Pour nous éclairer un peu, nous avons essayé différents traitements et exploité les résultats obtenus avec l'utilisation de matplotlib.

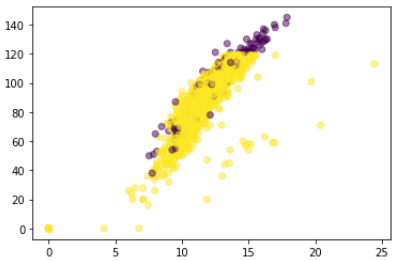


figure 1

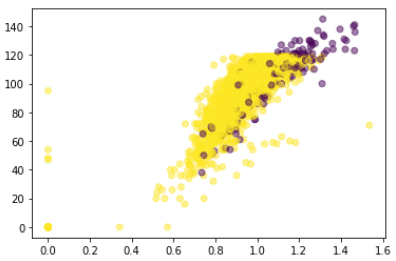


figure 2

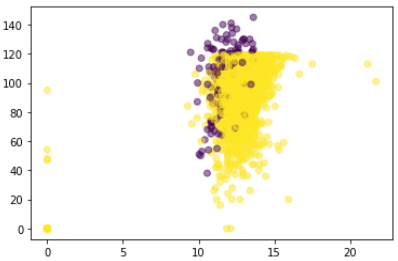
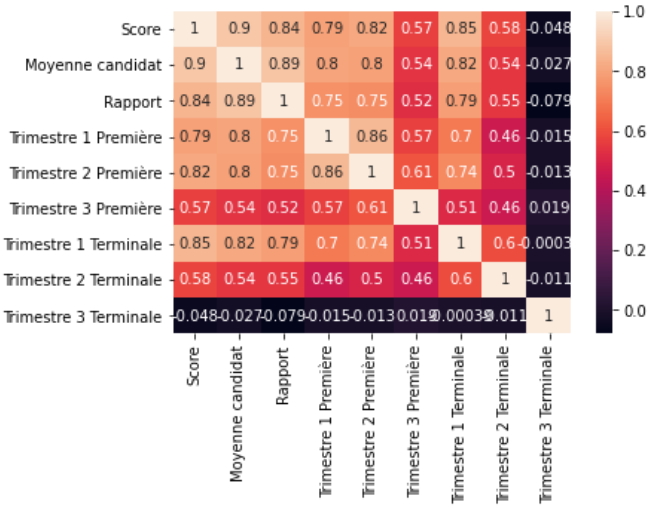


figure 3

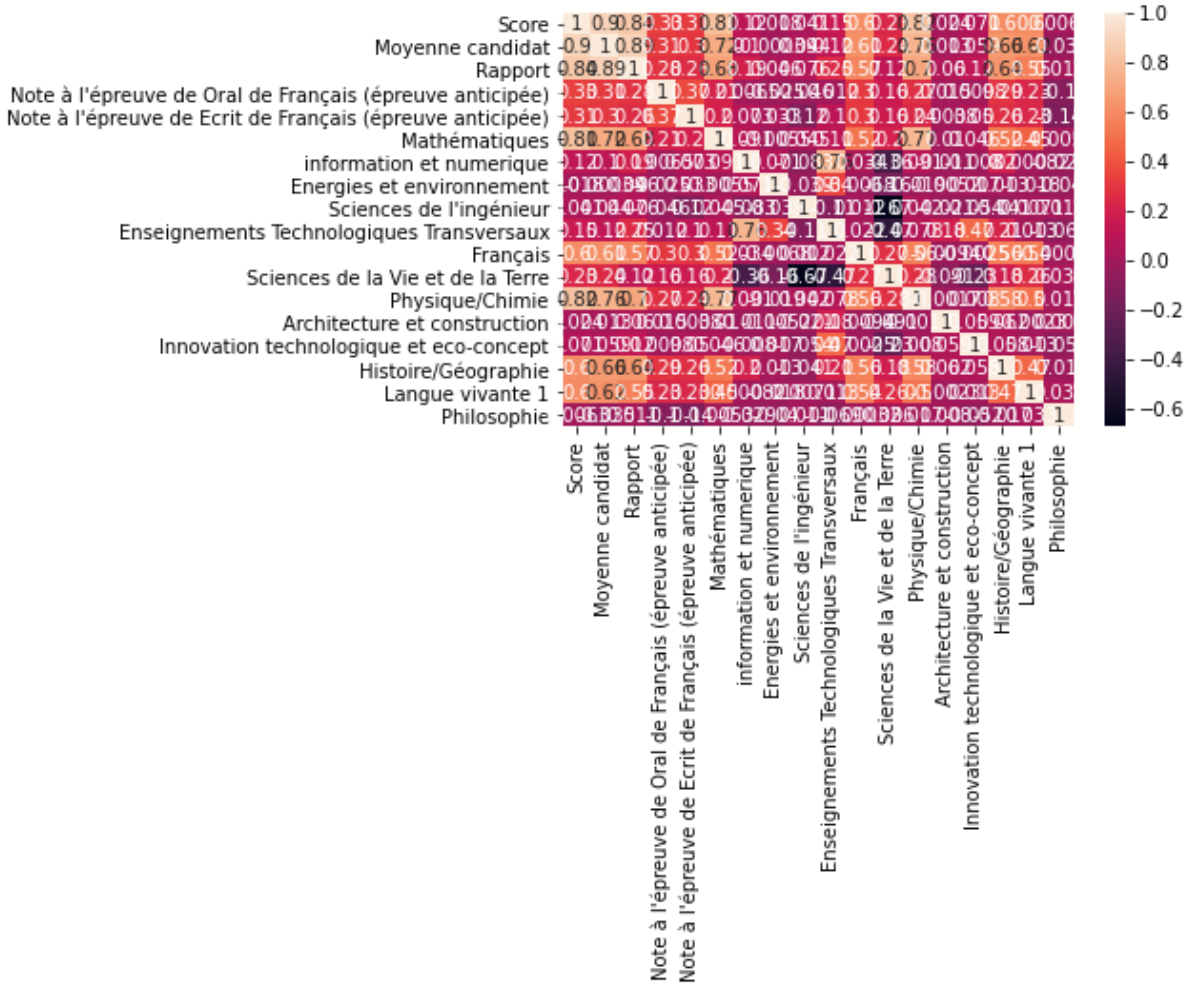
Ces graphes ont été tracés à l'aide de l'ensemble d'entraînement. Sur les trois figures ci-dessus, l'axe des ordonnées représente le score du candidat. En jaune les candidats de bac S et en violet les candidats de bac STI2D.
Pour la figure 1, l'axe des abscisses correspond à la moyenne du candidat.
Pour la figure 2, il correspond à la moyenne du candidat/ moyenne de la classe.
Pour la figure 3, il correspond à la moyenne de la classe.

En effet de nouvelles colonnes ont été mises en place selon un principe simple, on fait la moyenne des moyennes non nulles du candidat.
On peut alors obtenir une "Moyenne candidat" et une "Moyenne classe".
On remarque que la moyenne de classe n'est pas significative du score obtenu par le candidat, le "Rapport" suivant a alors été mis en place :
"Moyenne candidat"/"Moyenne classe"
Seulement les colonnes **"Moyenne candidat"** et **"Rapport"** ont été ajoutées aux dataframes.
De plus, toutes les colonnes correspondant à des moyennes de classe ont été supprimées.
Enfin, on observe que les candidats de bac S ne peuvent avoir un score supérieur à 120 et qu'aucun score n'est inférieur à 0. Ces observations sont exploitables pour les futures fonctions de prédiction.

De la même manière, les moyennes du candidat ont été triées par période ou par matière:



Pour les périodes, on remarque l'impact nul du 3ème trimestre de terminal. En effet, la majorité des valeurs n'étaient pas présentes donc remplacées par 0. Cela vient du fait que la base de données provient probablement d'une plateforme type Parcoursup, les données sont donc collectées avant la fin du 3ème trimestre.
Le choix a été fait de ne pas prendre en compte les moyennes périodiques des candidats pour les fonctions de prédictions.



En ce qui concerne les moyennes, on remarque l'importance des Mathématiques, de la Physique-Chimie, du Français, de l'Histoire-Géographie et de la LV1 sur le score.
Plusieurs réponses sont possibles, premièrement, ces 5 matières sont communes au bac S et au bac STI2D, elles sont donc mises en avant, car elles étaient composées de peu de valeurs manquantes (qui ont été remplacées par 0).
De plus, les Mathématiques et la Physique-Chimie sont des matières scientifiques, il est possible que la fonction de score importe plus d'importance à ces matières.
De même que pour les moyennes périodiques, les moyennes par matière n'ont pas été prises en compte pour les fonctions de prédictions.

Ces deux décisions sont le résultat de différents tests, concluant que les prédictions sont moins bonnes en les incluant plutôt qu'en les excluant.

III/ Prédiction :

Le nettoyage est maintenant défini, il est alors nécessaire d'effectuer les fonctions de prédiction. Les résultats indiqués pour les fonctions de prédictions proviennent d'une simulation en divisant la base d'entraînement en une sous base d'entraînement et une sous base de test. Un calcul de RSME est alors effectué pour avoir une **estimation du RSME** obtenu sur Kaggle.

1/ Régression linéaire :

Dans un premier temps, une **régression linéaire** a été effectuée sur la base de données d'entraînement, puis appliquée à la base de données de test pour obtenir la prédiction à fournir au Kaggle.
Résultat obtenu : `RSME : 6.09`

Afin d'améliorer ce score, la fonction de prédiction a été **divisée en deux parties** :

- une régression pour les bac S
- une régression pour les bac STI2D

Résultat obtenu : `RSME Mean : 5.74`

Dans le même objectif, une nouvelle fonction de prédiction a été créée, elle effectue une régression linéaire sur les différents résultats de la première fonction de régression linéaire sur différentes sous-bases d'entraînement afin d'éviter au maximum l'over-fitting.
Résultat obtenu pour 10 régressions différentes : `RSME Mean: 6.11`

2/ XGBoost :

Le **gradient boosting**, comme nous l'avons vu en cours, est une des techniques les plus puissantes pour apprendre un ensemble de modèles simples. Nous avons choisi d'utiliser l'implémentation de cette technique de prédiction fournie par XGBoost (rapide et pratique).

Avec les paramètres par défaut, le résultat obtenu est le suivant : `RSME : 4.54`

Afin d'améliorer ce score, nous avons utilisé les paramètres que XGBoost fournit pour réduire l'over-fitting.
Le nombre d'arbres dans l'ensemble est l'un des hyper-paramètres qui contrôlent le compromis entre l'under-fitting et l'over-fitting.
L'hyper-paramètre *max_depth* contrôle la profondeur de chaque arbre individuel dans l'ensemble. Une grande valeur de *max_depth* augmente la variance , alors qu'une petite valeur augmente le biais.
L'hyper-paramètre *learning_rate* met à l'échelle l'effet de chaque arbre sur la prédiction globale.

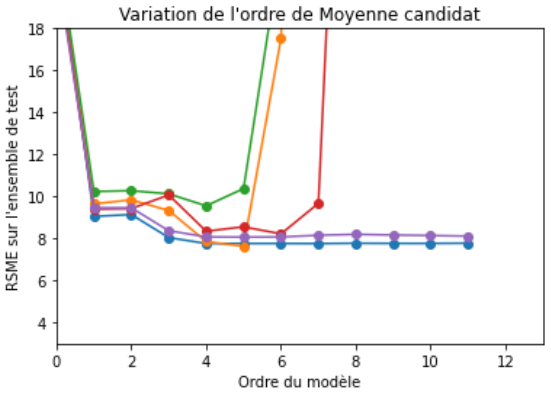
Un grid search afin de déterminer les meilleurs paramètres pour XGBoost n'a pas été concluant, nous avons donc manipulé à la main les paramètres.

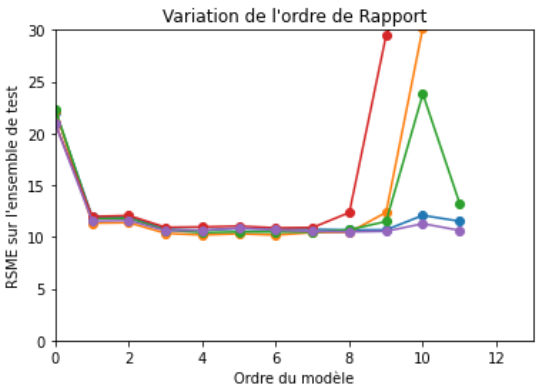
Avec comme paramètres :
n_estimators = 500, learning_rate = 0.1
Nous obtenons `RSME Mean : 4.44`

Puis avec :
n_estimators = 500, learning_rate = 0.1, max_depth= 2
Nous obtenons `RSME Mean : 4.37`

Cependant, le meilleur résultat obtenu sur la compétition Kaggle correspond à notre RSME mean de 4.44.

3/ Pistes supplémentaires :
Plusieurs pistes ont été évoquées durant le concours Kaggle pour améliorer le RSME obtenu. Une décomposition des colonnes en **polynômes de différents degrés** avait commencé à être effectuée.





De plus, la troisième fonction évoquée dans la partie II/1 est applicable aux fonctions de prédictions de type XGBoost. Cependant, encore une fois les résultats sont inférieurs à la simple fonction de prédiction. En nommant fonction finale la fonction permettant la meilleure prédiction obtenue sur le Kaggle voici le résultat.
Résultat obtenu pour 10 fonctions finales différentes : `RSME Mean: 5.66`

Pour XGBoost, nous aurions pu persister concernant le grid search, afin d'obtenir les paramètres délivrant le meilleur score. En effet, le grid search que nous avons implémenté n'a peut-être pas retourné un score se basant sur le RSME.

Conclusion :

Pour conclure, le résultat obtenu sur le Kaggle est le fruit de longues recherches, à la fois provenant du cours “Apprentissage automatique 2”, mais également des recherches et des idées originales venant de notre propre réflexion. Nous avons travaillé pour obtenir un RSME le plus faible possible. Le RSME final obtenu sur Kaggle est de 4.37763 ce qui nous positionne 3ème du concours (1er : 3.94498, 2ème : 4.18090).

Cette unité a été pour nous enrichissante, elle nous a motivés à travailler et à faire preuve d’ingéniosité.

Nous remercions J-F Berchet et toute l’équipe pédagogique de la filière DSIA de l’ESIEE pour leur enseignement dans ce cours et tout au long de l’année.