Data Challenge

Guillhem Artis

12/11/2023

1 Introduction

En guise d'introduction, disons simplement qu'il s'agissait d'un problème de prédiction de série temporelle. Un leak été présent dans les données mais il n'a pas été exploité.

2 Pré-traitement des Données

Plusieurs variables posaient problème en vue d'une prédiction. Commençons tout d'abord par les valeurs manquantes pour les variables $p_i q_j$. Au vu de la procédure de récupération des données avec des capteurs, les valeurs manquantes sont très probablement dues au simple fait qu'il n'y avait pas de train pour cet arrêt. Les valeurs manquantes ont donc été remplacées par des 0. Il y'avait aussi des valeurs manquantes parmi les heures. Des méthodes plus complexes auraient pu être utilisé, comme regrouper les heures par train et prendre la moyenne (ou la médiane sur ces groupes), mais j'ai pris le parti de la simplcité et rempli les valeurs manquantes par la médiane. Enfin vient les dates. On peut remarquer que les 2019 est la seule année présente dans les jeux de données, on peut donc la supprimer. Ensuite entre le jeux de train et le jeu de test les mois sont différents, on les enlèves donc également. Finalement la distribution des jours est la même pour les deux jeux, on les enlèves aussi.

3 Sélection du Modèle

Pour sélectionner le modèle, j'ai écrit une fonction évaluation qui estime l'erreur de prédiction pour des modèles classiques non optimisés. Le RandomForestRegressor et le GradientBoostRegressor ont fourni les meilleures prédictions. J'ai donc selectionné le GradientBoostRegressor pour sa précision et son efficicatcité.

- Modèles testés : 'SVR', 'Decision Tree', 'K Neighbors', 'Linear Regression', 'Gradient Boosting', 'Random Forest'
- Critère de sélection La métrique MAE a été utilisée. Elle paraîssait en effet préférable à la MSE car on souhaite pas prende en compte davantage en compte les "grandes" erreurs.

4 Modèle Final et Scores

Dans ma démarche d'optimisation des hyperparamètres, j'ai fait appel à la bibliothèque Optuna pour sa capacité à orchestrer la recherche des paramètres optimaux.

Pour chaque essai, le programme génère une combinaison unique d'hyperparamètres, comme le nombre d'arbres, variant de 100 à 500, le taux d'apprentissage, défini dans un intervalle de 0.01 à 0.3, la profondeur maximale de chaque arbre, choisie entre 3 et 10, et le nombre minimal d'échantillons requis pour fractionner un nœud, situé entre 2 et 10.

J'ai finalement selectionné les hyperparamètres donnant la plus faibles erreurs et les aies réutilisés pour entrainer le modèle sur le jeu de train tout entier Ces hyperparamètres parmi de nombreux autres ont été choisi après des tests car c'est eux qui semblaient avoir le plus d'impact.

5 Conclusion

L'erreur finale était de $0,0115$ elle aurait put être améliorer en modifiant le pré-traitement des données. Peut être en ajoutant des variables (moyenne d'occupation pour chaque train,...), et en considérant d'améliorer aussi la random forest. Cependant le plus optimal, vu les scores, aurait été d'esploiter le leak.