Prédiction de l'obtention d'un brevet

Samuel Guilhem-Ducléon

J'ai utilisé Python pour faire ce Data Challenge en utilisant les bibliothèques scikit-learn et xgboost. Vous trouverez ci-joint mon notebook ipython.

Après avoir essayé de la régression logistique et des forêts aléatoire, mon modèle final utilise du Gradient Boosting.

Preprocessing

J'ai d'abord utilisé Pandas pour extraire les deux fichiers csv de train et de test.

Ensuite j'ai séparé mes variables explicatives en trois : les variables catégorielles, les variables numériques et les variables de dates.

Dates

Pour traiter les variables de date, j'ai d'abord essayé de créer deux nouvelles variables explicatives pour séparer numériquement le mois de la date, mais j'ai finalement converti chaque valeur textuelle « mois / années » en nombre : mois + 12 * année après avoir vérifié que cette dernière option améliorait mon score.

Variables numériques

Pour chaque variable explicative de type numérique, j'ai remplacé les valeurs manquantes par la moyenne de la colonne.

Puis j'ai normalisé chacune de ces variables explicatives.

Variables catégorielles

L'idée est la suivante : pour chaque variable explicative textuelle, on cherche à retrouver les catégories les plus importantes. Pour ce faire, value_counts de pandas nous fournit la liste de chaque catégorie accompagnée du nombre d'occurrences.

On définit ensuite un ratio, et on garde les catégories dont le nombre d'occurrences est supérieure au nombre d'occurrence de la variable la plus importante divisée par ce ratio.

Comme les variables FIRST_CLASSE et MAIN_IPC contiennent beaucoup trop de catégories différentes, on fournira un ratio inférieur pour ces deux classes.

En jouant sur ces ratios, qui sont des hyperparamètres de notre modèle, on pourra déterminer le ratio le plus adéquat.

J'ai finalement pris un ratio de 20 pour FIRST_CLASSE et MAIN_IPC, et de 1000 pour les autres variables explicatives catégorielles.

J'ai ensuite modifié les catégories qui ne figurent pas parmi les plus importantes avec la même valeur « x », qui correspond en quelque sorte à la catégorie « autre ».

La fonction get_dummies de Pandas permet de créer une variable explicative par catégorie importante.

Dans mon modèle final, j'obtiens 767 variables explicatives dans mon modèle.

Enfin on réunit toutes les nouvelles variables explicatives ensemble.

On mélange les données et on sépare nos données provenant de train.csv en un ensemble de test et un ensemble de train.

Random Forest

Au début du challenge, n'ayant pas encore vu les forêts aléatoires en cours, j'avais essayé d'utiliser de la régression logistique. Après avoir optimisé les hyperparamètres avec de la validation croisée, je n'ai pas réussi à dépasser 0.69, j'en ai conclu que je devais changer de classifieur.

Dans un premier temps j'ai utilisé les forêts aléatoires en utilisant Scikit-learn.

Une cross-validation en jouant sur le nombre d'arbres et max_depth m'a permis d'obtenir un score de 0.709. J'ai observé qu'en augmentant le nombre d'arbres, le score s'améliorait. J'ai parallélisé le script sur les machines de l'école puis j'ai moyenné les fichiers de prédiction.

Après avoir vu une courbe dans le cours qui montrait que le Gradient Boosting était plus performant parfois, j'ai décidé de changer de classifieur.

Gradient Boosting

Après quelques recherches sur internet, j'ai vu que le classifieur gradient boosting était mieux implémenté en python sur la bibliothèque xgboost car il permettait de faire de la parallélisation et de faire de la régularisation L1 et L2.

Le Gradient Boosting de xgboost possède de nombreux hyperparamètres. Le multi-threading ne marchant pas sur Windows, je n'ai pas pu jouer avec tous ces hyperparamètres et j'ai sélectionné ceux qui me semblaient le plus pertinent. J'ai d'abord cherché à optimiser la profondeur maximale (max_depth) et le nombre d'itérations (num_rounds). Puis j'ai trouvé le poids minimum par enfant (min_child_weight) idéal. J'ai ensuite cross-validé le coefficient gamma. Enfin j'ai joué sur les coefficients de régularisation L1 et L2 (alpha, lambda).

J'obtiens finalement un score de 0.71513 sur l'ensemble de test en ligne.