Prêt à dépenser

Note méthodologique

Problématique

Nous sommes Data Scientist pour la société "Prêt à dépenser", l'objectif de ce papier est de rendre compte de la méthodologie appliquée pour mener à bien notre mission : calculer la probabilité qu'un client puisse rembourser ou non son crédit.

En quelques mots, le travail consistait à entraîner un modèle de classification binaire performant d'un point du vue métier et de donner la possibilité d'en interpréter les résultats.

Entraînement du modèle

Le modèle est entraîné à partir de l'historique des prêts transmis par notre employeur. L'étape de feature engineering est grandement inspirée d'un kernel Kaggle.

Valeurs manquantes

En fonction des colonnes nous avons décidé de remplir les valeurs manquantes de deux façons

- Par 0 pour les variables qui ont été créées de toutes pièces car leur absence provient du fait que le client n'a pas contracté de crédit dans le passé.
- Par la médiane sinon

Généralités

Plusieurs modèles existent, pour chacun d'entre eux il faut déterminer les meilleurs paramètres. Une première étape est alors de déterminer le modèle qui semble performer le mieux. Ici nous avons décidé d'utiliser une méthode linéaire, une méthode bagging et une méthode de boosting:

- Logistic Regression
- Random Forest
- Xgboost

Ensuite une fois le modèle choisi, nous passons à une étape d'hyper paramétrisation fine en vue d'obtenir un modèle encore plus performant. La recherche des meilleurs paramètres se fait à l'aide du module Hyperopt.

Validation croisée

Tous les tests se font en utilisant le processus de validation croisée. Un jeu de validation est tout de même extrait du jeu de données avant d'effectuer celle-ci. Notons que nous avons utilisé une validation croisée 'manuellement' en construisant nous-même nos folds car nous avions des problèmes pour utiliser proprement les méthodes de resampling. Utiliser ce processus nous a également permis une optimisation du temps de calcul : si le score du modèle sur un des fols est inférieur à un score seuil donné, la validation croisée s'arrête. Ce qui économise du temps sur des combinaisons de modèle/paramètres qui donne de mauvais résultats.

Équilibrage des classes

Suite à l'étape d'analyse exploratoire nous avons constaté que les deux classes n'étaient pas équilibrées. En effet la classe 0 (client ayant remboursé leur prêt) représente 92% des observations. Ce déséquilibre nous amène donc à tester des méthodes visant à pallier ce problème :

- Tomek-link : Supprimer des instances de la classe majoritaire proches des instances la classe minoritaire pour augmenter l'espace entre les deux classes, ce qui facilite le processus de classification.

- SMOTE : Créer des points synthétiques de la classe minoritaire à partir des voisins les plus proches d'une instance choisie au hasard (appartenant tous à la classe minoritaire).

Fonction coût métier et métrique d'évaluation

Aborder le problème d'un point de vue métier

Lorsqu'un prêt est remboursé, une banque gagne relativement peu comparé au montant total de ce qu'elle prête. D'un point de vue technique, nous devons être en mesure de pouvoir détecter avec une grande précision les potentiels clients qui vont faire défaut, ils coûteraient bien plus cher que ce qu'ils pourraient rapporter

Lien entre la fonction coût et la métrique d'évaluation

La fonction coût métier est en lien direct avec la métrique utilisée pour déterminer quel modèle performe le mieux. En effet nous avons le choix entre deux approches :

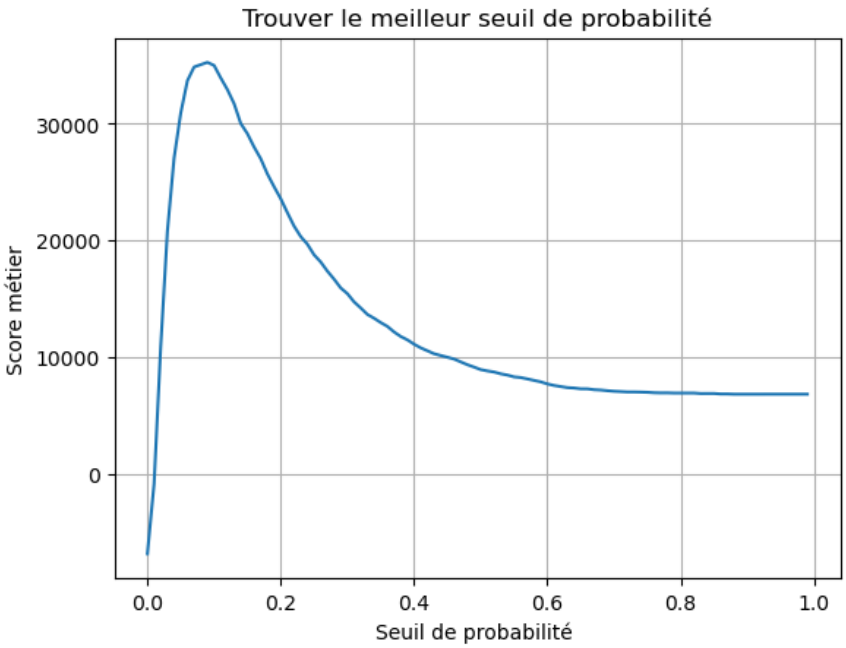
- Utiliser directement la fonction coût comme métrique lors de la l'hyper paramétrisation. Par exemple, nous pouvons utiliser dans notre cas le F-Beta score avec une valeur de Beta élevée pour privilégier le rappel.
- Séparer la fonction coût du modèle et choisir une métrique d'évaluation plus générale. Une fois le modèle obtenu nous déterminerons un seuil d'attribution à la classe 1 ou 0 qui maximise les bénéfices de la banque relativement à la fonction coût métier (par défaut ce seuil est à 0.5).

Nous choisissons la deuxième méthode en utilisant la métrique AUROC qui se prête naturellement au fait de faire varier le seuil d'attribution d'une classe à l'autre. De plus, séparer la fonction coût du modèle permettra une plus grande flexibilité si elle venait à être modifiée, il ne faudrait plus réentraîner le modèle mais simplement calculer un nouveau seuil.

Détail de la fonction coût métier

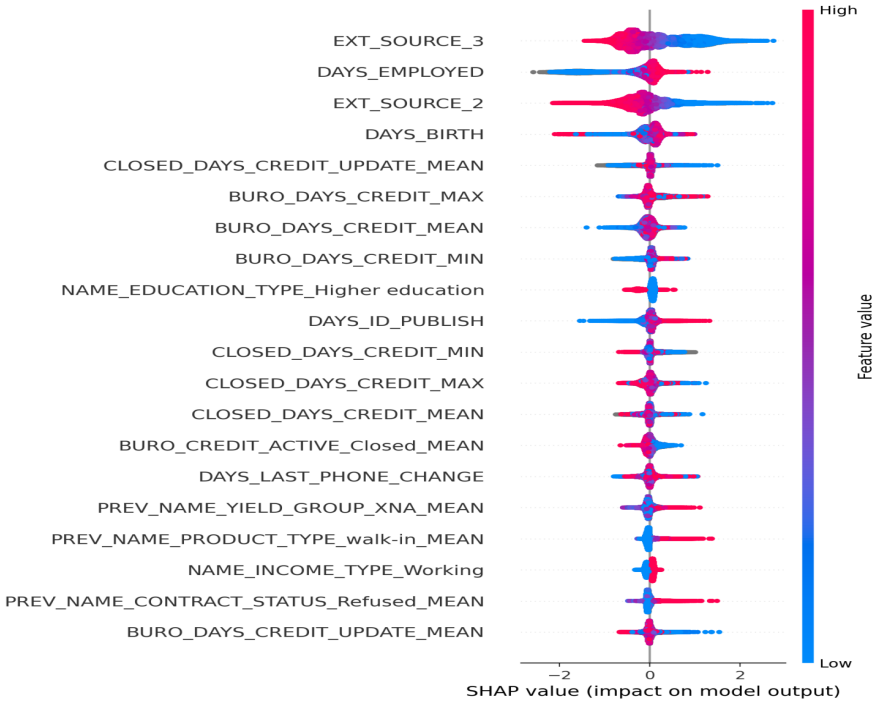
Nous définissons notre fonction coût de la manière suivante :

- + 1 pour les vrais négatifs
- - 1 pour les faux positifs
- + 10 pour les vrais positifs
- - 10 pour les faux négatifs



Constatons que lorsque le seuil d'acceptation est à 0 alors tous les clients sont refusés. A l'inverse, quand il est à 1, tous les clients sont acceptés. Dans cet exemple, le seuil qui maximise la fonction est aux alentours de 0,10 et donc qu'un crédit sera accordé si la probabilité qu'un client fasse défaut est inférieure à 8%.

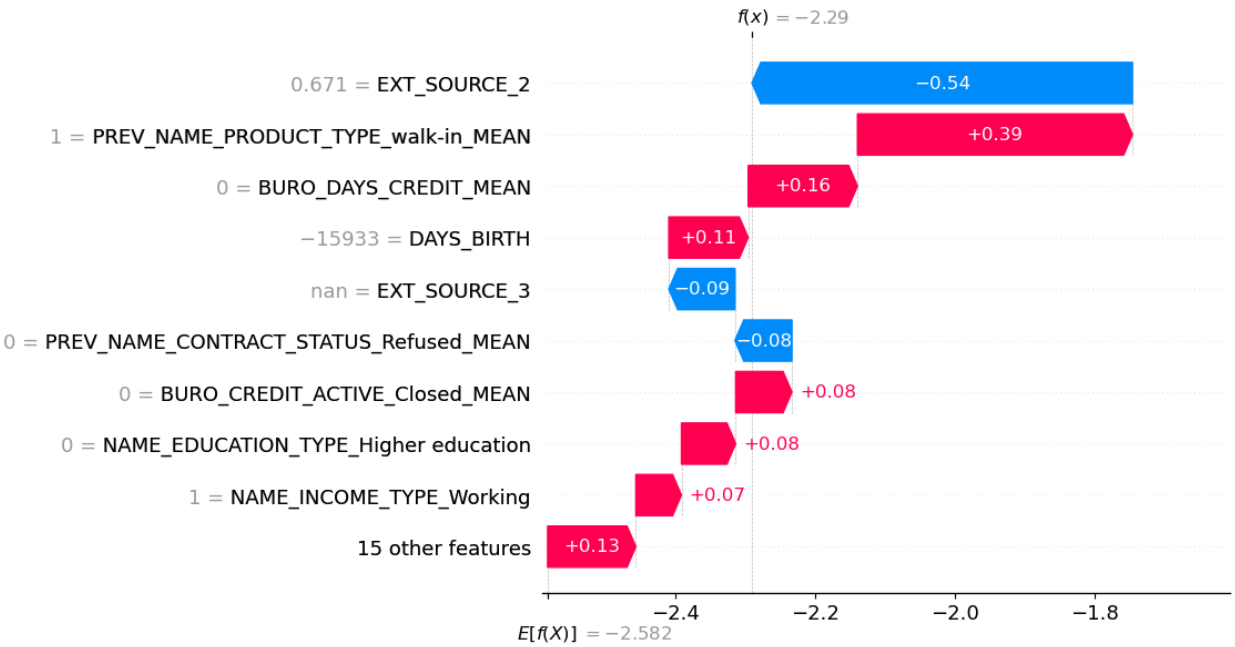
Interprétabilité globale et locale



Interprétabilité globale

Nous utilisons la librairie shap pour déterminer quelles variables ont le plus d'impact positif et négatif sur la classification. Ici nous pouvons par exemple voir qu'une valeur faible de EXT_SOURCE_3 indique que le client a plus de chance de faire défaut.

Interprétabilité locale



Améliorations possibles

Valeurs manquantes

La stratégie pour traiter les valeurs manquantes est très basique. Il serait judicieux de remplir les valeurs par la médiane d'un groupe donnée (même type d'emploie, étude..) ou alors pénaliser ce manque d'information : utiliser la médiane du groupe de client ayant fait défaut.

Fonction coût métier

La fonction coût métier doit être améliorée, il faudra travailler avec un expert du métier. En particulier elle doit varier selon le type de prêt (exemple : la banque peut toujours récupérer un bien si le client ne peut plus payer)

Modèles

D'un point de vu technique il faudrait améliorer les points suivants :

- Utiliser des "class weight" pour ajouter plus de poids à la classe 1
- Faire une hyper optimisation encore plus fine, nous avons été contraint de