Note Méthodologique : Projet 7 – Implémentez un modèle de scoring

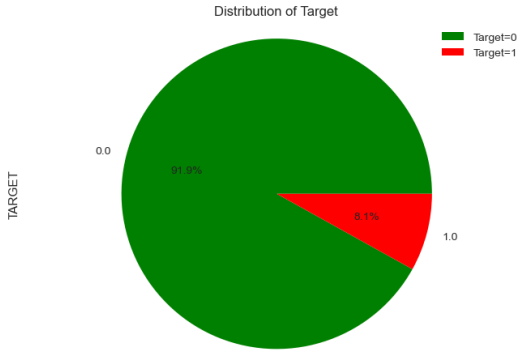
Le but de ce projet est de créer un modèle de classification binaire pour le compte de l'entreprise financière « Prêt à Dépenser » afin de déterminer si on accorde ou pas un prêt bancaire à des personnes qui postulent pour un prêt. La classe 0 signifie que le prêt est accordé tandis que la classe 1 signifie que le prêt n'est pas accordé

1. Méthodologie d'entraînement du modèle

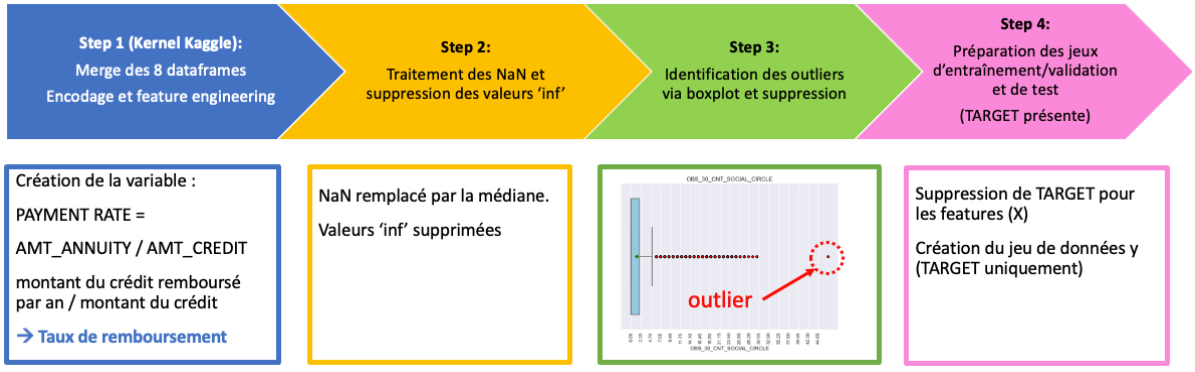
1.1. Données

Les jeux de données sont disponibles ici : https://www.kaggle.com/c/home-credit-default-risk/data. Ces données contiennent 307 000 clients et 1221 features qui renseignent sur le profil de chaque client : revenu, sexe, âge, montant emprunté, etc. Il y a en tout 8 jeux de données à merger.

Il est à noter que les classes de la TARGET sont très déséquilibrées : il n'y a que 8% des observations qui correspondant à la classe 1 (prêt non accordé).



1.2. Traitement des données



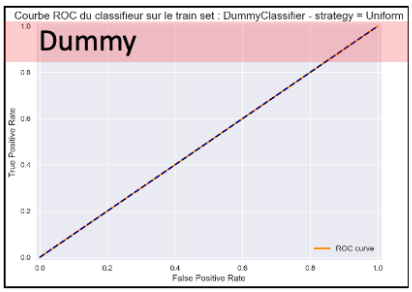
On soumet les données aux étapes mentionnées ci-dessus. On utilise le kernel Kaggle pour merger les 8 jeux de données. Après encodage, on procède à un traitement des NaN (remplacement par la médiane). Afin de diminuer le nombre de features, on supprime les colonnes qui contiennent plus de 20% de NaN.

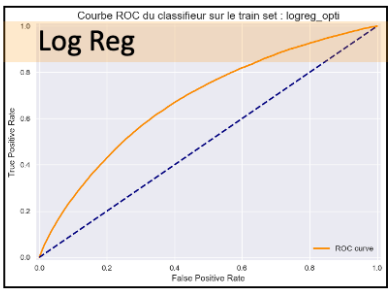
Sur les 193 features restantes, on visualise sur un boxplot la présence d'outliers. Le cas échéant, ils sont supprimés.

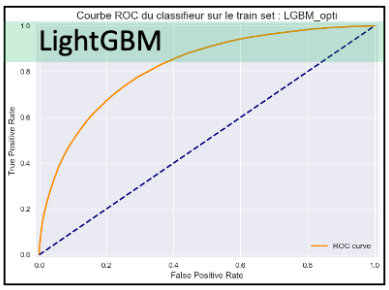
1.3. Entraînement du modèle

Le jeu d'entraînement représente 90% des clients tandis que le jeu de test en représente 10%. Auparavant, on a également sorti un jeu de données de validation non utilisé pendant l'entraînement/test.

Plusieurs modèles sont testés : DummyClassifier, Logistic Regression et LightGBM. Pour les deux derniers, on réalise une optimisation des hyperparamètres sur le jeu d'entraînement/test en faisant une validation croisée avec GridSearchCV. Le modèle retenu est LightGBM :

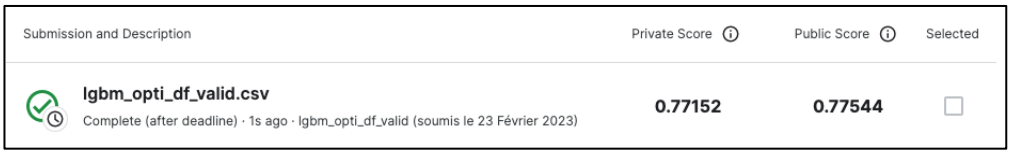






ROC_AUC Dummy = 0.5 -	**ROC_AUC Log Reg = 0.6719** C : 1 penalty = l2 solver : newton-cg	**ROC_AUC LightGBM = 0.7814** learning_rate: 0.1, max_depth: 5, min_data_in_leaf: 50, num_iterations: 200, num_leaves: 20

Une fois ce modèle LightGBM optimisé, on le teste sur le jeu de validation (non vu pendant l'entraînement/test) : on obtient un score ROC_AUC = 0.766. Ce modèle est également soumis à la compétition Kaggle et le score ROC_AUC obtenu est : 0.7715/0.7754 (privé/public), ce qui nous conforte dans la performance du modèle.



1.4. Déséquilibre des classes

On applique deux techniques : optimisation du paramètre class_weight de LightGBM et un undersampling. La technique d'undersampling donnant un score ROC_AUC dégradé de 0.7787, on choisira l'optimisation du paramètre class_weight réalisé avec une validation croisée qui améliore légèrement le score ROC_AUC de 0.7814 à 0.7819.

2. Fonction coût métier

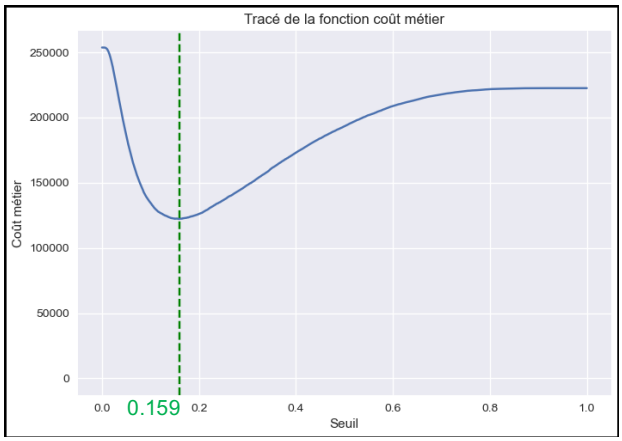
2.1. Définition du coût métier

On définit la fonction : $coût\ métier = FN + 10FP$. On considère que la présence d'un faux négatif (personne non solvable qui s'est vu attribuée un prêt) est 10 fois plus coûteuse que la présence d'un faux positif (personne solvable qui s'est vu refuser un prêt).

2.2. Optimisation (minimisation du coût)

On doit minimiser la quantité $FN + 10FP$. Sur le jeu d'entraînement, on calcule la probabilité de faire défaut (classe = 1) en utilisant la méthode predict_proba du modèle LightGBM. Celle-ci retourne deux colonnes, la 1ère est la probabilité de prédire la classe 0, la 2ème est la probabilité de prédire la classe 1, on va donc prendre la deuxième colonne. Pour chaque valeur de seuil entre 0 et 1, on aura une répartition de FN et de FP différente sur l'ensemble du jeu d'entraînement. En calculant le coût pour chaque valeur de seuil, on obtient une courbe dont on cherchera le minimum.

2.3. Métrique d'évaluation

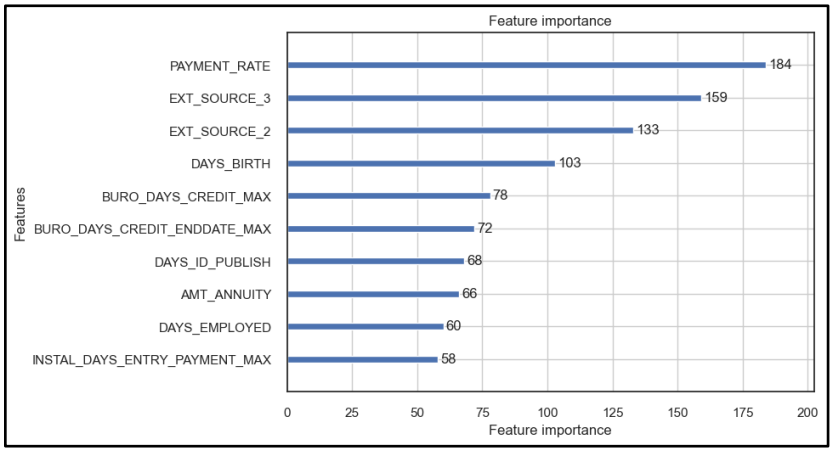


On observe après calcul que le seuil pour minimiser le coût est de 0.159 donc il faudra refuser le prêt si la probabilité de prédire la classe 1 est supérieure à 0.159.

3. Interprétabilité globale et locale

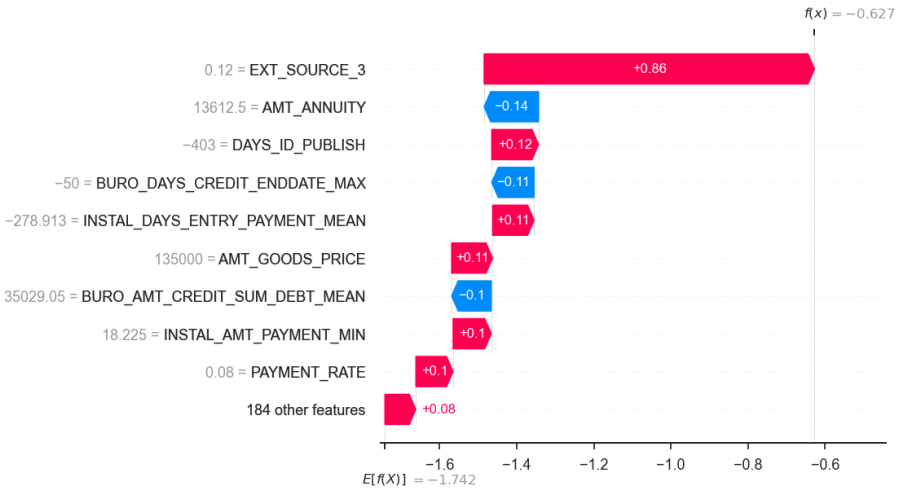
3.1. Interprétabilité globale

LightGBM fournit les features par ordre d'importance, elles sont regroupées dans ce graphique :



3.2. Interprétabilité locale

SHAP permet de déterminer les features qui contribuent à la décision de classification pour chaque client. Ci-dessous, un exemple pour le client 7395, pour lequel le prêt a été refusé. On voit que la feature EXT_SOURCE_3 a été déterminante (on la retrouve aussi dans les features globales) :



4. Limites et améliorations possible

Le fait de faire tourner le modèle LightGBM sur une machine aux performances modestes limite l'optimisation des hyperparamètres. On pourrait optimiser plus d'hyperparamètres ou avec une granulométrie plus fine avec plus de CPU.

On aurait pû également tester d'autres modèles (RandomForrest Classifier, HistGradientBoostingClassifier) ou une fonction coût plus complexe pour améliorer l'optimisation du coût.