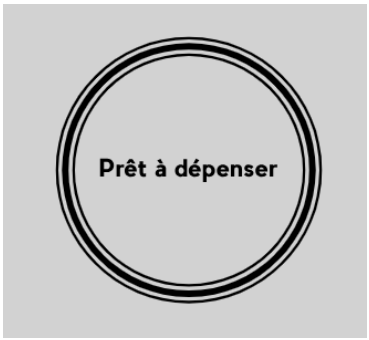
LECLERCQ Lancelot Mai 2022

Note méthodologique



Sommaire

1 Introduction

L'entreprise Prêt à dépenser souhaite utiliser un outil de "scoring" afin de calculer la probabilité qu'un client fasse ou non défaut lors du remboursement de son crédit. Pour cela nous devons entrainer un modèle de classification sur des données variées (comportementales, autres institutions financières, etc).

2 Classification

2.1 Jeu de donées

Afin de mieux comprendre les données nous avons procédé à une analyse exploratoire des données sur le jeu application_train.csv. Pour l'entrainement du modèle nous utiliserons uniquement ces données car l'utilisation des fichiers supplémentaires demande beaucoup de ressources tant en temps d'analyse et d'exploration des données qu'en capacitées de calculs. Nous utiliserons le fichier application_test.csv pour le dashboard.

2.2 "Feature engeneering"

Calcul de variables polynomiales à partir des meilleurs variables (EXT_SOURCES). Il n'y a pas d'amélioration significative des résultats (Fig. 1)

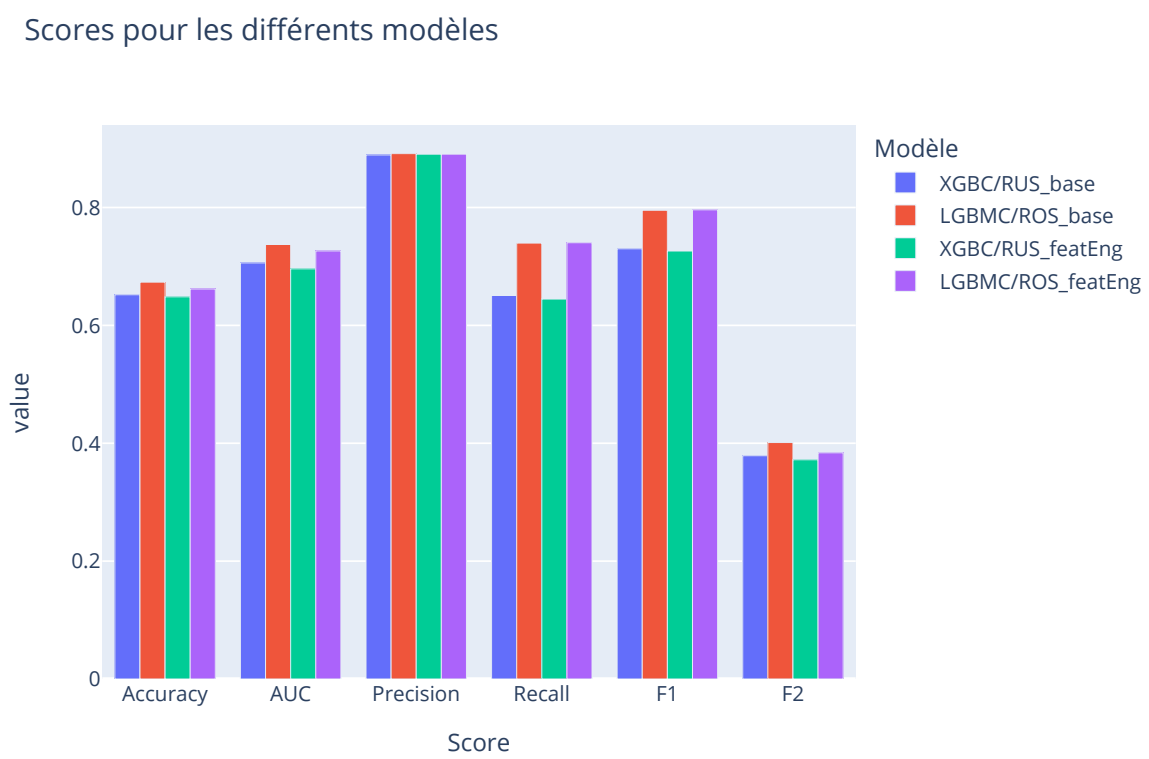


FIG. 1 : Comparaisons des différents scores avec (_featEng) et sans (_base) variables polynomiales

2.3 Déséquilibre des valeurs cible

Du fait d'un déséquilibre dans les valeurs cible il est difficile pour le modèle de classer efficacement les clients (Fig. 2).

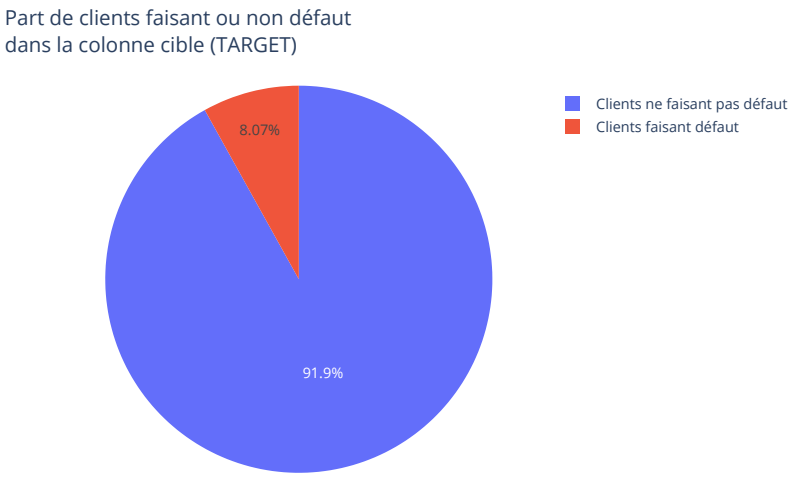


FIG. 2 : Diagramme circulaire illustrant le déséquilibre des types de clients dans la colonne cible

Lors de l'optimisation du score AUC le modèle va préférer classer tout les clients comme ne faisant pas défaut car cela amériorera sont score. Nous avons donc dû rééquilibrer la part des valeurs cible grace à la librairie imblearn. Cette librairie propose des outils de sur- et de sous-échantillonnage.

Le sur-échantillonnage (over-sampling) permet d'augmenter le nombre de cibles faisant défaut. Soit en dédoublant aléatoirements des cibles faisant défaut (random over sampling). Soit en créant des données synthétiques (SMOTE) à partir de cibles voisines (KNN)

Le sous-échantillonnage (under-sampling) permet de diminuer le nombre de cibles ne faisant pas défaut. Soit en échantillonant aléatoirement un nombre de cible ne faisant pas défaut égale au nombre de cible faisant défaut (random under sampling) Soit en créant des données (Tomek links) à partir de groupe de cibles ne faisant pas défaut (KNN)

Un mélange de sur- et de sous-échantillonnage avec SMOTEENN et SMOTETomek qui combinent les effets de SMOTE avec une réduction à partir des plus proches voisins

2.4 Modèles

Nous avons utilisé deux modèles XGBoost et LightGBM qui ont l'avantage d'être résistants au valeurs manquantes. Ces deux modèles reposent sur le boosting de gradient qui consiste en une agrégations d'arbres de décisions simples afin d'optenir un meilleur résultat

3 Évaluation, coût métier et optimisation

3.1 Courbe ROC et aire sous la courbe (AUC)

La courbe ROC représente les vrais positifs en fonction des faux positifs (Fig. 3). Plus la courbe est proche du coin supérieur gauche meilleur est le modèle. L'aire sous la courbe (AUC) nous donne une valeur numérique pour comparer ces modèles.

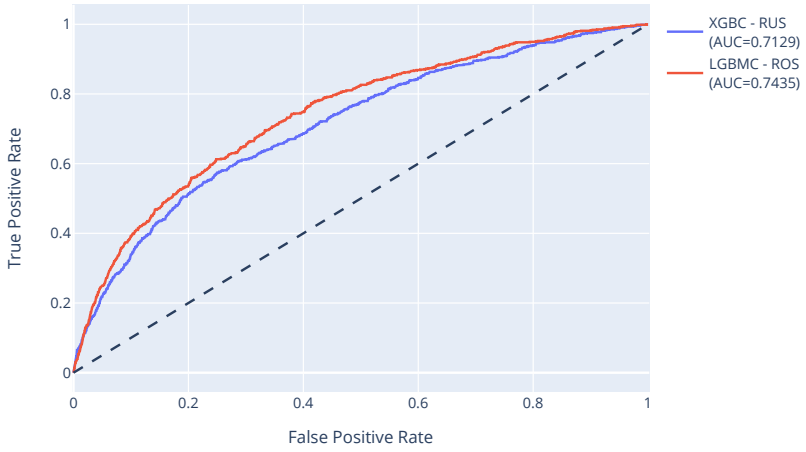


Fig. 3 : Courbes ROC pour les deux modèles dont nous avons testé les hyperparamètres

3.2 Coût métier

Afin d'utiliser une méthode d'évaluation qui soit mathématiquement fondée nous nous sommes basé sur une métrique déjà existante du F-score. Ce score calcul la moyenne harmonique de la précision et du rappel (Fig. 4). L'utilisation du F_β-score permet d'ajouter du poid respectivement au recall lorsque le facteur β est >1 ou à la precision lorsque le facteur β est <1. Nous avons utilisé une valeur de β=2

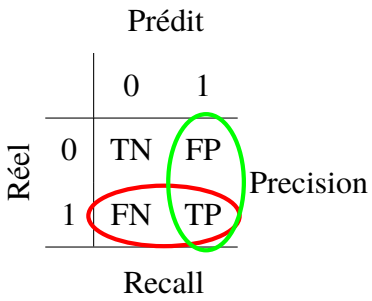


Fig. 4 : Matrice de confusion rapellant ce que sont la précision et le rappel

3.3 Optimisation

L'optimisation a été effectuée par GridSearch. Pour chaque solution de rééquilibrage nous avons fait une GridSearch pour les deux modèles. Cependant pour la classification avec XGBoost nous avons testé une seul solution avec un nombre de boosting de 100. Pour LightGBM nous avons testé différents algorithmes : GBDT, DART, RF et GOSS et différents nombre de boosting : 100, 500 et 1000. Nous avons retenu comme meilleur modèle LightGBM avec un sur-échantillonage aléatoire (random over sampling). Les paramètres retenu sont l'algorithme DART et un nombre de boosting de 100

4 Interprétabilité

Dans un soucis de transparence nous souhaitons pouvoir expliquer comment fonctionnent nos modèles

4.1 Globale

L'étude des principales variables utilisées par le modèle permet de mieux comprendre son fonctionnement global (Fig. 5)

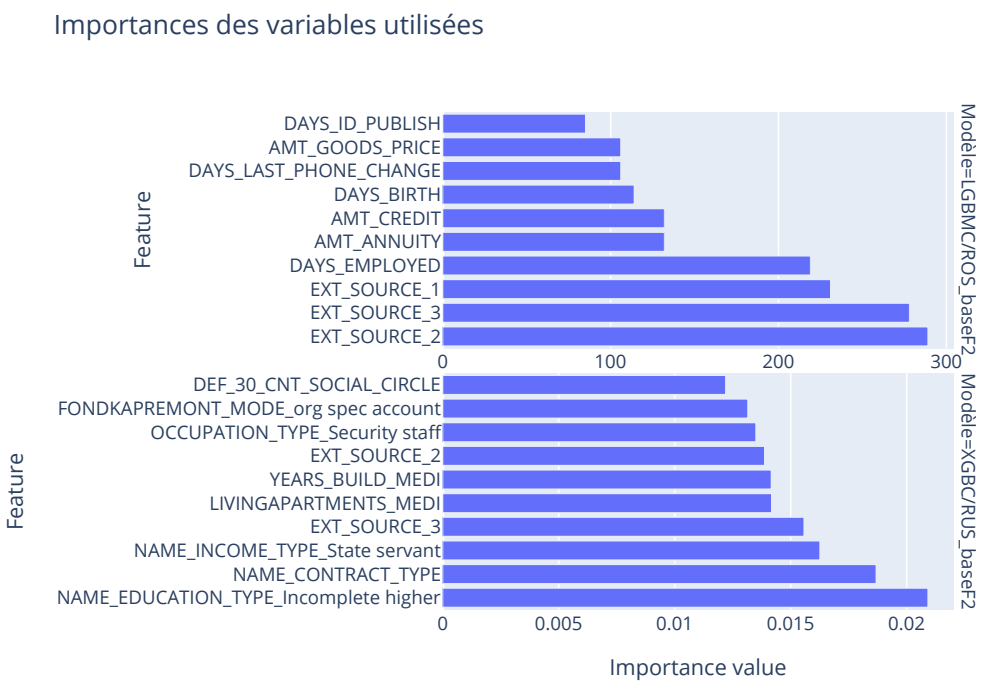


Fig. 5 : Principales variables selon leur importance dans l'entrainement des modèles

Nous avons utilisé la librairie SHAP afin d'expliquer le modèle retenu (LightGBM). Celle-ci nous permet d'observer la part des différentes variables dans la classification dans l'une ou l'autre des classes (Fig. 6).

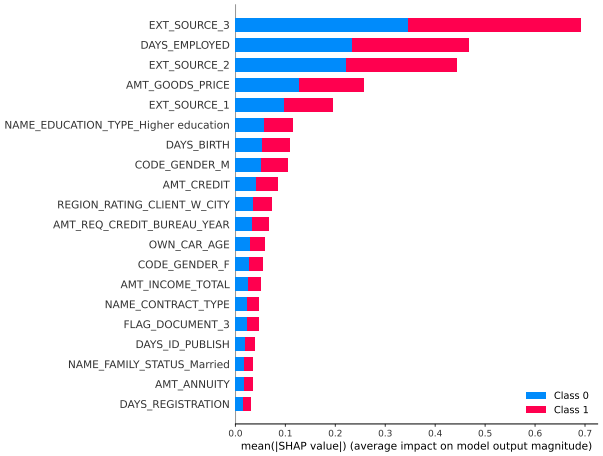


FIG. 6 : Part des principales variables dans la classification

4.2 Locale

La librairie SHAP nous permet d'en apprendre un peu plus sur la part des variables dans le classement d'un client en particulier (Fig. 7).

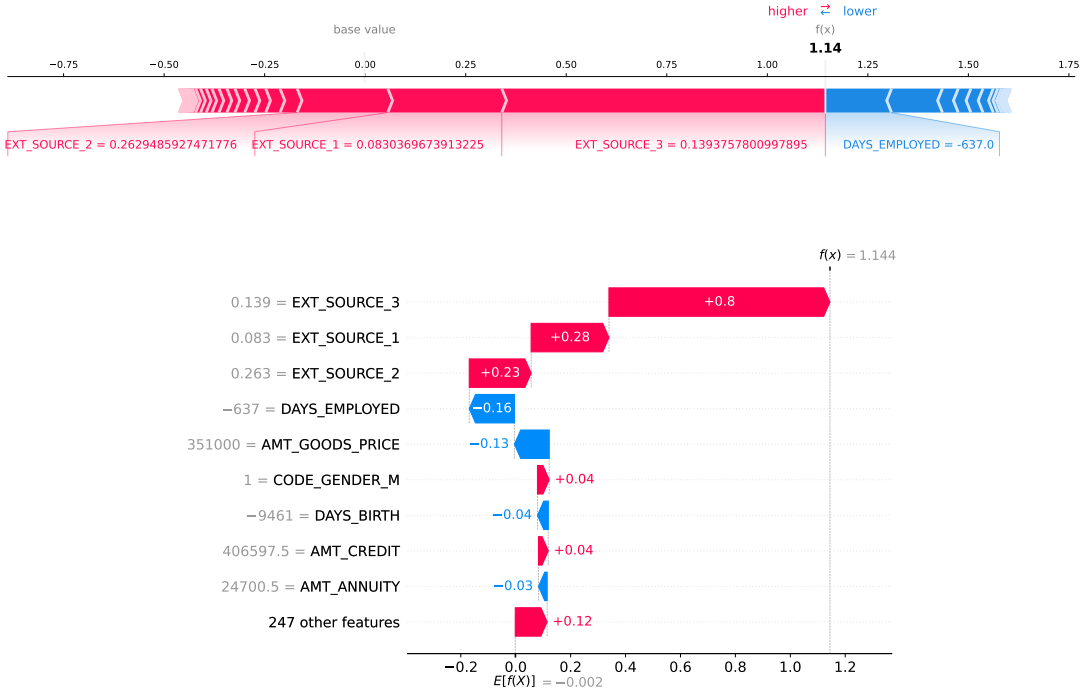


FIG. 7 : Part des variables dans la classification d'un client

5 Limites et améliorations

Nous pourrions essayer d'optimiser plus d'hyperparamètres dans les différents modèles afin d'améliorer encore les résultats