06/10/2022

Note méthodologique

Projet OpenClassRoom N°7
Parcours Data Scientist

Bouchra MEKHALDI

Table des matières

1- CONTEXTE

Le présent document fait partie des livrables du projet 7 intitulé « Implémentez un modèle de scoring » du parcours Data Scientist d'Openclassrooms. Ce projet consiste à développer pour la société « Prêt à Dépenser » (une société de crédit de consommation) un modèle de scoring afin de prédire la probabilité de défaut de paiement d'un client avec peu ou pas d'historique de prêt. Il s'agit donc d'un problème de classification binaire supervisée dans lequel la variable cible (TARGET) à prédire vaudra 0 si le client est solvable et 1 s'il ne l'est pas.

Ce document décrit de façon détaillée la méthodologie d'entraînement du modèle ; la fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation ; l'interprétabilité globale et locale du modèle puis se termine en exposant les limites et les améliorations possibles du travail effectué.

2- PRÉSENTATION ET TRAITEMENT DES JEUX DE DONNÉES

2.1 Présentation des jeux de données

La base de données de « Home Credit » (https://www.kaggle.com/c/home-credit-default-risk/) comporte plus de 350 000 clients, avec leurs données personnelles, leur historique de remboursement, leur historique de crédits chez les différentes banques.

Ces données sont réparties dans sept jeux de données, plus un fichier de description des variables.

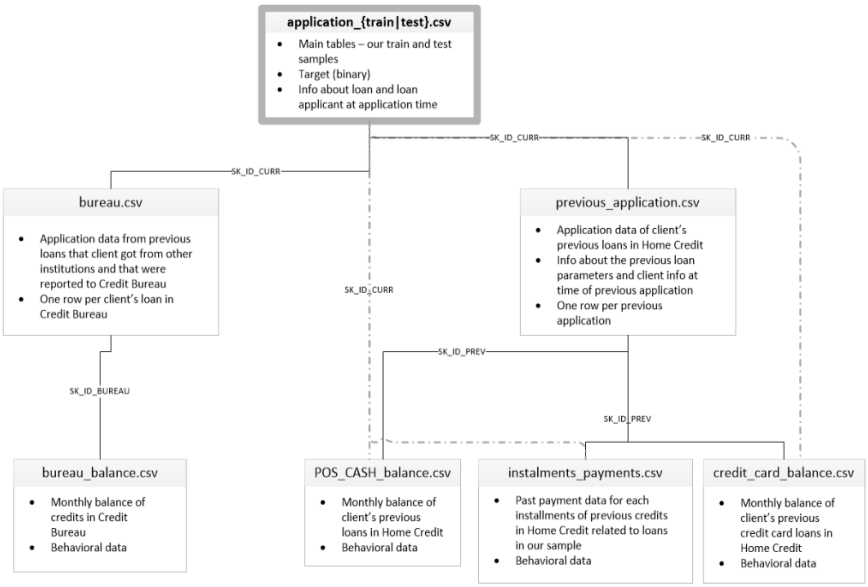

Figure 1 Arborescence du jeu de données

La variable cible à prédire prend 2 valeurs et est fortement déséquilibrée (8/92) :

0 – positive – non défaillant : indique que le client a totalement remboursé son prêt

1 – négative – défaillant : indique que le client n'a pas remboursé son prêt en totalité ou en partie.

2.2 - Traitement des jeux de données

a- Analyses graphiques

Dans un premier temps, nous avons réalisé des analyses univariées et multivariées afin d'avoir des premières indications sur le profil et le comportement général des clients de la banque.

b- Nettoyage

Étape 1 : consiste à transformer les types des objets pour les rendre cohérents (ex : Y/N en 0/1) et réduire leur taille de stockage dans la mémoire.

Etape 2 : consiste à corriger les valeurs aberrantes détectées lors de l'analyse exploratoire (EDA) et d'harmoniser les valeurs uniques des données (ex : sexe Masculin, Féminin) pour les informations principales.

Étape 3 : de supprimer les variables avec plus de 50% de valeurs manquantes pour conserver les variables importantes détectées lors de l'EDA et de supprimer les variables sans information pour le modèle.

Étape 4 : d'imputer donc de remplacer les variables conservées ayant des valeurs manquantes par la valeur médiane pour toutes les variables numériques et par la valeur la plus utilisée pour les variables qualitatives.

c- Feature engineering

Étape 5 : consiste à créer de nouvelles variables qui peuvent augmenter la performance du modèle. La création des features pertinents pour la modélisation a été entièrement inspirée du notebook Kaggle suivant https://www.kaggle.com/jsaguiar/lightgbm-with-simple-features/.

d- Encodage des variables

Étape 6 : consiste à encoder les variables qualitatives pour les transformer en numériques seulement consommables par le modèle. Deux types d'encodage ont été utilisés :

- **LabelEncoder** : pour les variables contenant peu de valeurs différentes (ex : sexe : Féminin transformé en 0 et Masculin en 1)
- **OneHotEncoder** : pour les variables contenant moins plusieurs valeurs différentes

3- METHODOLOGIE D'ENTRAINEMENT DU MODELE

Pour résoudre notre problématique de scoring différents modèles ont été testés avec la recherche d'hyperparamètres grâce à **la cross validation** (5 folds)

Le jeu de données initial a été séparé en plusieurs parties de façon à disposer :

• D'un jeu de training (80% des individus) qui a été séparé en plusieurs folds pour entraîner les différents modèles et optimiser les paramètres (cross validation)

• D'un jeu de test (20 % des individus) pour l'évaluation finale du modèle

Nous avons un problème de classification binaire avec une classe sous représentée (8 % de clients qui ne remboursent pas encodé par la valeur 1, contre 91 % de clients qui remboursent encodé par la valeur 0). Ce déséquilibre des classes doit être pris en compte dans l'entraînement des modèles

3.1- Rééquilibrage des classes

Ce déséquilibre des classes doit être pris en compte dans l'entraînement des modèles. Pour gérer ce déséquilibre, nous avons testé 3 approches :

Undersampling (RandomUnderSampler de la librairie Imbalanced-Learn) : suppression des observations de la classe majoritaire afin de rééquilibrer le jeu de données.

Oversampling (SMOTE de la librairie Imbalanced-Learn) : répétition des observations de la classe minoritaire afin de rééquilibrer le jeu de données.

Class_weight="balanced" : argument qui indique le déséquilibre à l'algorithme afin qu'il en tienne compte directement. 3

3.2- Choix du meilleur modèle

Afin de modéliser au mieux le problème, les performances les algorithmes comparés sont :

Régression Logistique (famille des algorithmes linéaires) via la classe LogisticRegression de la librairie Scikit-Learn

RandomForestClassifier (famille des algorithmes ensemblistes) via la classe RandomForestClassifier de la librairie Scikit-Learn

Light Gradient Boosting Machine (famille des algorithmes gradient boosting) via la classe LGBMClassifier de la librairie lightgbm. Leurs performances ont été comparées à celles d'une baseline naïve, une instance de la classe

DummyClassifier de la librairie Pandas, instanciée avec la stratégie "most_frequent", c'est-à-dire prédisant systématiquement la classe la plus fréquente.

a- Métrique utilisée

La métrique utilisée pour évaluer les performances de chacun des modèles est l'AUC (Area Under the ROC Curve). Plus l'AUC est élevée, plus le modèle est capable de prédire des bons résultats. Puis on a utilisé la métrique F1-score pour le choix final de notre model.

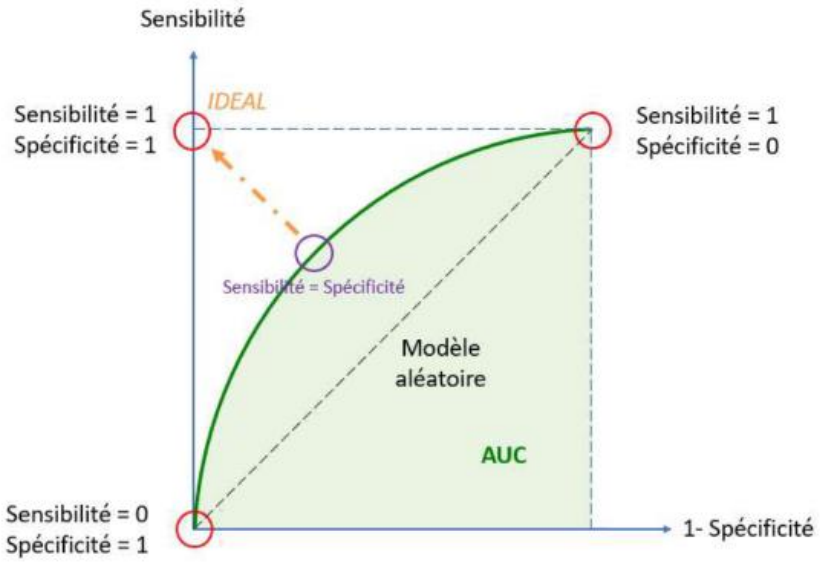

Figure 2 La courbe ROC

Le meilleur modèle a été choisi sur la base de la valeur du score AUC moyen et scoref1 obtenu sur l'ensemble des splits de la validation croisée (GridSearchCv). Le score AUC obtenu sur le jeu de test a également été calculé. L'algorithme LGBM associé à la stratégie de rééquilibrage consistant à indiquer " **Undersampling** " comme sampling méthode donne les meilleurs résultats.

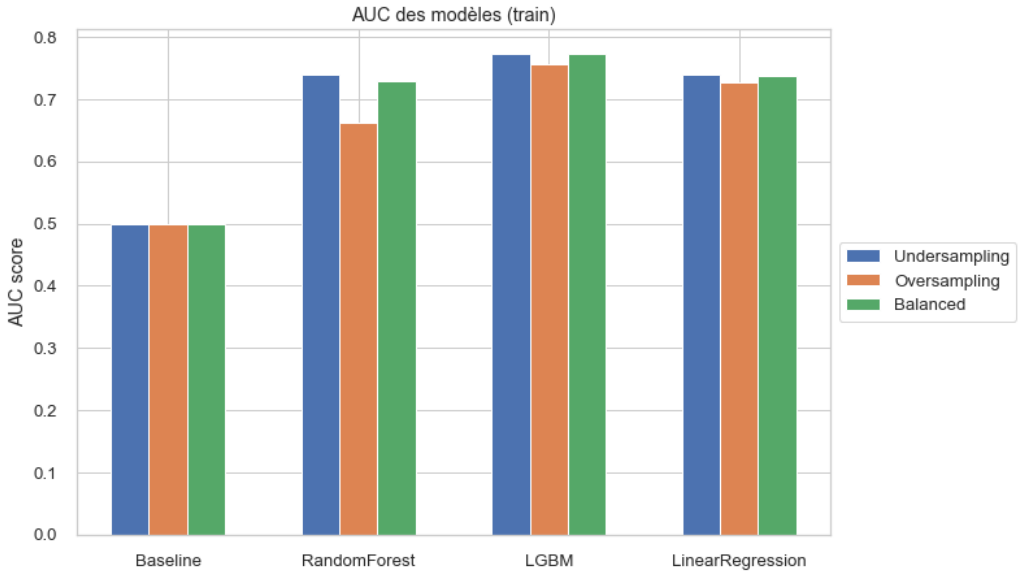

Figure 3 AUC pour chaque model

3.3- fonction cout métier et algorithme d'optimisation

a- Fonction coût métier

Définitions des terminologies de la matrice de confusion

- FP (*Faux Positifs*) : la classe a été prédite comme "1", mais la classe réelle est "0" => le modèle a prédit que le client ne rembourserait pas, mais il a bien remboursé son crédit.

- FN (*Faux Négatifs*) : la classe a été prédite comme "0", mais la classe réelle est "1" => le modèle a prédit que le client rembourserait, mais il a été en défaut.

- TP (*Vrais Positifs*) : la classe a été prédite comme "1", et la classe réelle est bien "1" => le modèle a prédit que le client ne rembourserait pas, et il a bien été en défaut.

- TN (*Vrais Négatifs*) : la classe a été prédite comme "0", et la classe réelle est bien "0"=> le modèle a prédit que le client rembourserait, et il l'a bien fait.

La problématique « métier » est de prendre en compte qu'un faux positif (bon client considéré comme mauvais = crédit non accordé à tort, donc manque à gagner de la marge pour la banque) n'a pas le même coût qu'un faux négatif (mauvais client à qui on accorde un prêt, donc perte sur le capital non remboursé). Un faux négatif est en effet 10 fois plus coûteux qu'un faux positif. Nous avons ainsi défini une fonction coût métier adaptée au projet qui permet d'attribuer plus de poids à la minimisation des FN.

Des coefficients permettant de pondérer les pénalités associées à chaque cas de figure ont été attribués : TP_value : 0 ; FN_value : -10 ; TN_value : 1 ; FP_value : -1

Ces valeurs de coefficients signifient que les Faux Négatifs engendrent des pertes 10 fois plus importantes que les pertes engendrées par les Faux Positifs. Les Vrais positifs n'entrainement aucun gain ni perte pour l'entreprise, tandis que les Vrais Négatifs permettent à l'entreprise de gagner en prêtant à de bons prospects.
Ces poids ont été fixé arbitrairement et il est tout à fait envisageable de les modifier à la convenance de l'optique métier.

```python
def bank_score(y_true, y_pred, fn_value=-10, fp_value=0, tp_value=0, tn_value=1):

    # Matrice de Confusion
    mat_conf = confusion_matrix(y_true, y_pred)

    # Nombre de True Negatifs
    tn = mat_conf[0, 0]
    # Nombre de Faux Négatifs
    fn = mat_conf[1, 0]
    # Nombre de Faux Positifs
    fp = mat_conf[0, 1]
    # Nombre de True Positifs
    tp = mat_conf[1, 1]

    # Gain total
    gain  = tp*tp_value + tn*tn_value + fp*fp_value + fn*fn_value

    # Gain maximum
    max_gain  = (fp + tn)*tn_value + (fn + tp)*tp_value

    # Gain minimum
    min_gain  = (fp + tn)*fp_value + (fn + tp)*fn_value

    # Gain normalisé entre 0 et 1
    gain_normalized = ( gain  - min_gain )/(max_gain - min_gain )

    return gain_normalized   # Retourne la fonction d'évaluation
```

Figure 4 Fonction coût métier

Donc :

- Accorder un crédit à un client ne pouvant pas le rembourser par la suite (FN) est synonyme de perte (10x importante que les FP)
- Accorder un crédit à un client qui le remboursera par la suite (TN) est un gain.
- Ne pas accorder le prêt et que le client ne peut pas rembourser (TP) n'est ni une perte, ni un gain.
- Ne pas accorder le prêt alors que le client pouvait rembourser (FP) est une perte de client donc d'argent.

b- Algorithme d'optimisation

Une fois le modèle choisi (LGBM) et ses hyperparamètres optimisés d'un point de vue technique (via l'AUC avec GridSearch), nous avons à nouveau effectué une nouvelle recherche des hyperparamètres via **HyperOpt** (algorithme qui utilise l'optimisation bayésienne) en se basant sur la fonction coût métier proposée. De cette façon, les hyperparamètres optimaux du modèle sont choisis de sorte à minimiser la perte pour l'entreprise. **HyperOpt** nécessite 4 paramètres pour une implémentation de base à savoir : la fonction objectif (à minimiser **loss = 1-score**), l'espace de recherche (plages pour les hyperparamètres), l'algorithme d'optimisation et le nombre d'itérations.

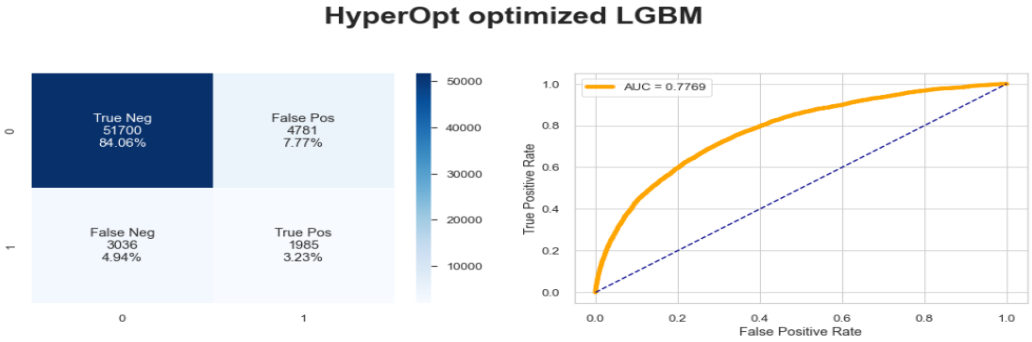

Figure 5 Matrice de confusion et courbe ROC aprés l'optimisation de model

c- Fixation du seuil de solvabilité optimum

Le seuil de solvabilité optimum, en fonction de notre métrique bancaire, est de **0.35**

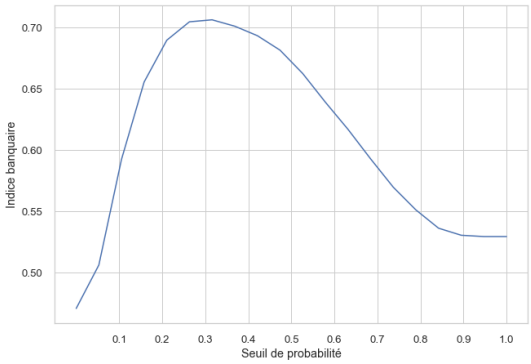

Figure 6 Seuil de solvabilité

4- INTERPRETABILITE

L'objectif métier est de communiquer des informations pertinentes d'analyse au chargé d'étude, afin qu'il comprenne pourquoi un client donné est considéré comme bon ou mauvais prospect et quelles sont ses données/features qui expliquent son évaluation. Il est donc nécessaire à la fois, de connaître d'une manière générale les principales features qui contribuent à l'élaboration du modèle, et de manière spécifique pour le client qu'elle est l'influence de chaque feature dans le calcul de son propre score (feature importance locale).

Pour cela nous avons utilisé une librairie spécialisée de type **Shap** afin de fournir un calcul de la feature importance globale indépendante de l'algorithme (car en effet, de nombreux algorithmes proposent des fonctionnalités de feature importance globale, mais spécifiques à l'algorithme).

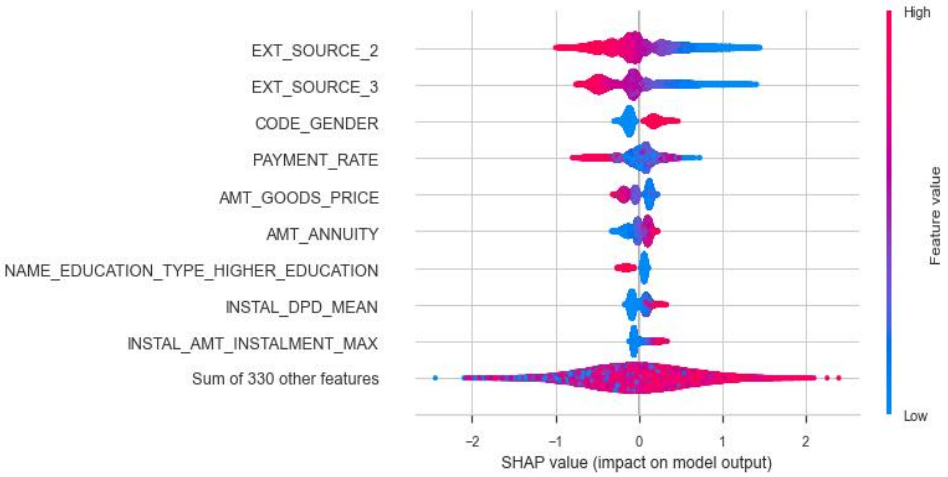

Figure 7 représentation d'impact des variables sur un graphe SHAP

5- AMELIORATION DU MODELE

- Après le feature engineering inspiré du Kaggle mentionné plus haut, les données contiennent 795 features au total. Pour la modélisation, nous aurions pu ajouter une étape de sélection des features (en utilisant par exemple la technique d'élimination récursive des features avec validation croisée (RFECV)) afin de ne retenir que les features les plus importants et réduire ce nombre.

- A cause des soucis d'espace mémoire, nous avons dû travailler avec un échantillon de 100000 clients et pas la totalité du dataset. Appliquer la modélisation sur l'intégralité du jeu de données sur une machine avec plus de RAM serait certainement intéressant à explorer

- Chacun des modèles entrainés a été optimisé en testant plusieurs valeurs des 2 ou 3 principaux hyperparamètres (car l'exécution de GridSearch prend beaucoup de temps). Elargir le nombre d'hyperparamètres pourrait peut-être permettre une amélioration des performances des différents modèles.

- La fonction coût métier définie utilise des poids de pénalités arbitrairement fixés. Il est donc certainement possible d'affiner les prédictions du modèle en fixant ces poids de façon plus adéquate par des experts métier.