Note méthodologique

Table des matières

Références

Brownlee, J. (2020, January 17). *SMOTE for Imbalanced Classification with Python*. Récupéré sur machinelearningmastery.com: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Lundberg, S. (s.d.). *SHAP*. Récupéré sur shap.readthedocs.io: https://shap.readthedocs.io/en/latest/index.html

N. V. Chawla, K. W. (2011). SMOTE: Synthetic Minority Over-sampling Technique. *arXiv*, 321-357.

Introduction

Cette note méthodologique est rédigée dans le but d'expliciter la démarche mise en œuvre dans ce projet. Elle ne traitera pas des détails de toute la partie analyse de donnée et ingénierie des différents attributs de la base de données qui est fondamentale dans tout projet de data science.

La méthodologie d'entraînement du modèle

Dans ce projet d'attribution de prêt en fonction des antécédents du client, il est tout de même important de noter que la base de données est très fortement déséquilibrée. On parle ici de la valeur cible, 0 ou 1, qui représente respectivement les clients ayant repayé leur crédit et les clients n'ayant pas (partiellement, avec du délai etc.) ou dans une vision plus manichéenne, les bons et les mauvais clients. Cette valeur cible possède un ratio 0/1 d'environ 8,8 % sur les 307511 individus de la base de données (fig 1), représentant le fait que la majorité des clients honorent leur dette.

```
TARGET repartition in the dataframe :
0    282686
1     24825
Name: TARGET, dtype: int64
----------
pourcentage target 1/0 : 8.781828601345662 %
```

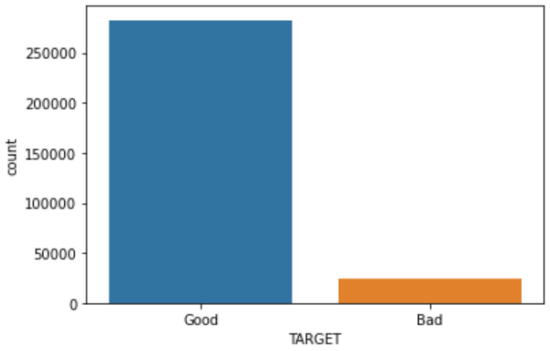

Figure 1 : déséquilibre de la variable cible

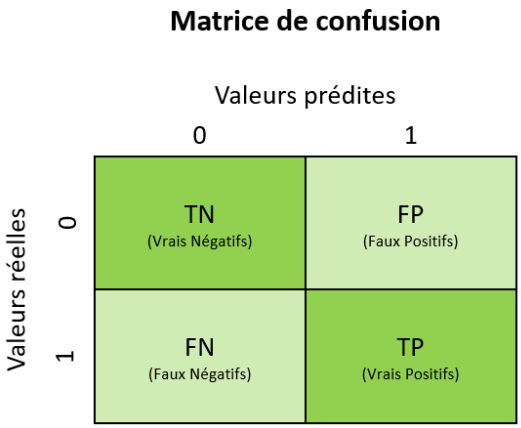

Figure 2 : matrice de confusion

Le traitement de ce déséquilibre va avoir un impact direct sur la méthodologie d'entraînement des modèles et de la métrique dévaluation présentée dans la partie suivante.

Prenons l'exemple de la mesure de précision $= \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ (voir fig.2) Si l'on fait une prédiction de 100 % de « 0 ». On retrouvera une précision globale de 92.8 % avec dans le détail 100 % pour la classe majoritaire et 0 % pour la classe minoritaire. Dans les faits, accorder un prêt à un « mauvais » client que l'on a prédit comme « bon client » entraîne un coût bien plus élevé pour l'organisme de crédit que de refuser un prêt à un « bon client » prédit comme mauvais.

Cela nécessite donc la sélection minutieuse d'une mesure de performance qui favorise à la fois la minimisation des erreurs de classification erronée en général mais favorise surtout la minimisation du type d'erreur de classification erronée de la classe minoritaire par rapport à la classe majoritaire.

Entraînement du modèle

L'entrainement du modèle commence par la préparation des données nécessaire à la bonne utilisation de l'algorithme.

La première étape consiste, en utilisant la bonne métrique (voir partie suivante), à tester de manière succincte (en utilisant les options de configuration par défaut ou légèrement adaptées au problème) plusieurs algorithmes de différent type. On va pouvoir en retirer un ensemble de performances base que l'on va pouvoir comparer pour faire une première sélection.

Un bon cadre d'évaluation de test va impliquer l'utilisation d'une validation croisée stratifiée, (*stratified k-fold* ou estimation de fiabilité du modèle par l'échantillonnage {avec la même répartition que la distribution des classes d'origine} en jeux d'entrainements / validation), pour lequel on s'attardera sur les moyennes et les déviations standard des différents modèles pour les comparer.

J'ai décidé de prendre en compte 4 types d'algorithmes :
Un algorithme naïf (logique inexistante par rapport aux données) pour servir de référence.
Un algorithme linéaire (relation linéaire pondérée avec les entrées).
Un algorithme non linéaire (combinaison non linéaire des entrées).
Des algorithmes ensemblistes qui utilisent la prédiction de 2 modèles ou plus (« bagging / boosting ») car ils sont souvent très performants.

Pour la correction de la disparité des classes, j'opte pour 2 solutions qui sont :
1. Les algorithmes dit « *cost sentitive* » (i.e. algorithmes classiques qui peuvent prendre en compte la disparité lors de l'ajustement sur le jeu d'entrainement)
2. Les techniques d'équilibrage de la base de données (par *Oversampling* et *Undersampling*) tels que SMOTE (*Synthetic Minority Oversampling Technique*) qui dispose de plusieurs stratégies, telles que
- *Adaptive Synthetic Sampling (ADASYN)* : génération d'échantillons inversement proportionnel à la densité de la classe minoritaire (i.e. génération d'échantillons synthétiques dans les « régions » ou la densité de la classe minoritaire est faible et correspondant à un apprentissage plus difficile).
- *Borderline-SMOTE SVM* : Sélection des régions où il y a moins d'exemples de la classe minoritaire et extrapolation vers la limite de la classe.
- D'autres approches que je ne détails pas ici (Brownlee, 2020).
Ces techniques sont combinées avec un sous-échantillonnage pour une performance optimale par rapport à une sous-échantillonnage simple (N. V. Chawla, 2011).

Après cette première étape, je vais sélectionner un ou plusieurs modèles pour passer à une recherche plus approfondie d'optimisation. L'utilisation d'une grille de recherche large, type *GridSearchCV*, permet d'effectuer cette tache de manière automatisée en choisissant des combinaisons d'hyperparamètres adéquats (dans un premier temps, les plus importants pour avoir une meilleure idée de leur précision et de sélectionner le modèle final, puis pour optimiser les performances au maximum). J'opte aussi pour l'utilisation de pipelines pour le prétraitement des données. Cette pratique permet non seulement d'éviter la fuite des données mais aussi de notamment pouvoir intégrer des algorithmes d'équilibrage dans la recherche de paramètres d'optimisation.

La dernière étape consiste en l'optimisation finale du modèle sélectionné avec la fonction de coût métier détaillé dans la prochaine partie.

La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

Métrique d'évaluation

Le choix de la métrique d'évaluation est un des point critique de ce projet. Pour correctement répondre à la problématique, j'ai sélectionné les métriques « *roc_auc* » et le *fbeta* score.
Ces deux scores s'appuient sur les valeurs de la matrice de confusion (fig. 2) et permettent d'attribuer une considération plus importante à la classe minoritaire :
- Courbe *ROC* ou *AUC ROC* {*Compute Area Under the Receiver Operating Characteristic Curve* / calcul de l'air sour la courbe}), soit la False Positive Rate en fonction de la True Posistive Rate (eq. I) en fonction du seuil.
- le fbeta score (avec beta =2) qui s'intéresse à la précision et au rappel (eq. I).

$$FPR = \frac{FP}{FP + TN} \qquad precision\ (TPR) = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \qquad F_\beta = \frac{(1 + \beta^2)\ * precision\ * recall}{(\beta^2 * precision) + recall} \qquad (I)$$

Ces mesures se basant sur des seuils, on peut alors envisager différentes stratégies :
- Une stratégie de validation du plus de client tout en prenant le risque d'avoir plus de mauvais clients et d'augmenter ses pertes.
- Une stratégie plus prudente où l'on va minimiser les clients à risques quitte à laisser passer de bons clients et minimiser ses bénéfices.
- Des stratégies intermédiaires avec par exemples des variations en fonction du montant du crédit.

Pour pouvoir optimiser l'outil de décision sur le choix de stratégie, on va pouvoir s'appuyer sur l'utilisation d'une fonction coût métier.

La fonction coût métier

La mesure roc auc ou le fbeta score ne vont pas forcement parler au client, on peut donc utiliser une fonction dit métier pour mieux coller au cahier des charges (i.e. attentes du client et stratégie). Pour se faire j'ai crée une métrique personnalisée qui va dans ce sens.

Fonction cout métier : $(\frac{TP}{TP+FN} - 33 * \frac{FN}{FN+TP})/32$

Mon raisonnement est le suivant :

Pour un bon client accepté, on gagne les bénéfices du prêt (que j'ai estimé à 3% en moyenne mais que l'on pourrait ajuster en fonction de la valeur du prêt).
Pour un mauvais client, on va perdre les intérêts plus tout ou partie du prêt.

Cette fonction coût métier représente donc une prise en compte de l'impact sur le bénéfice d'un prêt à un mauvais client que j'ai estimer à 33 fois plus grande que le gain d'un prêt à un bon client.

L’interprétabilité globale et locale du modèle

Pour l’interprétabilité du modèle, j’ai opté pour la librairie SHAP (*SHapley Additive exPlanations*), qui explique la restitution de la sortie d’un modèle de *machine learning* par une approche de la théorie du jeu (Lundberg, s.d.).

SHAP permet de présenter de manière graphique les différents poids des *features* d’entrées de manière globale, en donnant l’impact positif ou négatif de manière générale ou en donnant le poids des variables liées à la prédiction d’un individu particulier.

Après le choix d’un *explainer* adapté à notre modèle de *machine learning*, SHAP commence avec des valeurs de base utilisé pour des prédictions basés sur des connaissances préalables. On vient ensuite essayer les *features* une par une pour comprendre l’impact de leur introduction sur les valeurs de base et la prédiction finale. SHAP prend aussi même en compte l’ordre d'introduction des *features* ainsi que l'interaction entre elles, ce qui aide à mieux comprendre les performances du modèle. Au cours de ce processus, SHAP enregistre des valeurs qui seront utilisées ultérieurement pour tracer et expliquer les prédictions.

On peut ensuite représenter différents types de graphiques, tels que :

- Le *summary_plot* : Un graphique de distribution des valeurs de chaque *feature* de la base de données et présente son impact global sur la prédiction. On peut par exemple voir sur la figure 3 que des valeurs élevées de la *feature* « EXT_SOURCE_3 » ont un impact négatif sur la prédiction.

- Le *force_plot* (fig. 4) : les features les plus positive ou négative à l’échelle locale de la prédiction.

Le chargé de clientèle pourra alors en fonction de ces différents graphiques avoir une interprétation globale du modèle pour tous les clients et locale en fonction du client pour lequel il étudiera l’octroi du prêt.

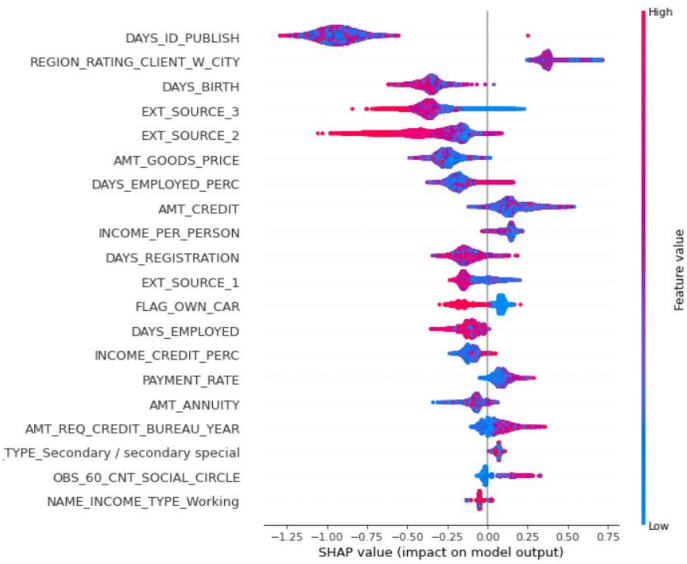

Figure 3 : summary plot (globale)

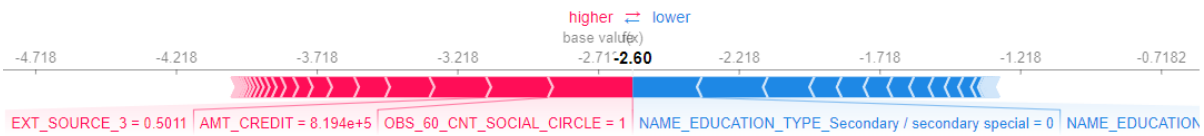

Figure 4 : force plot (local)

Les limites et les améliorations possibles

Pour améliorer la prédiction de ce projet, il y a plusieurs pistes :

• Premièrement d'un point de vue technique :

L'Inclusion de toutes les autres bases de données et pas uniquement l' « applications_train » pour avoir une prédiction plus précise.

Un *feature engineering* plus poussé (création de plus de variables, traitement plus approfondit des valeurs extrêmes etc.).

L'élargissement du nombre de modèle testés en première recherche et recherche approfondie des hyperparamètres.

Le prétraitement des variables spécifiques à chaque modèle. Par exemple retirer des variables trop corrélées entre elles pour les algorithmes linéaires.

Un travail plus approfondi sur les techniques d'*over* et d'*under sampling*.

Travailler avec l'ensemble de la base de données (par manque de puissance de calcul, je me suis limité à un travail sur des portions de cette dernières).

• Deuxièmement, d'un point de vue métier :

Pour pouvoir répondre au mieux aux attentes du client, il est impératif de pouvoir communiquer avec lui pour avoir un retour métier sur la fonction de coût et par exemple, ajuster la fonction de perte en fonction de la valeur du prêt. On peut aussi discuter de la stratégie rechercher en fonction du type de prêt et de clients.