2022

Project 7 Implémentez un modèle de scoring

Note méthodologique

Travail réalisé par Koffi KONAN

I Introduction

« **Prêt à dépenser** » est une société financière qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt. Le jeu de données qu'elle met à notre disposition dans notre rôle de Data Scientist pour l'aider à prédire les clients susceptibles d'être en défaut de paiement comporte 307511 observations ayant chacune 121 caractéristiques dont le nombre de jours travaillés, le sexe, l'âge, le revenu, etc....

Cette présente note s'inscrit dans le cadre de l'explication de la démarche employée pour rechercher un modèle de Machine Learning approprié à ce problème métier ici en l'occurrence la prédiction de défaut de paiement d'un client.

Nous allons dans un premier temps mettre en relief la méthodologie d'entrainement des modèles étudiés puis en évidence l'intérêt de choisir une nouvelle métrique d'évaluation pour se conformer de façon réelle à une approche métier et enfin faire des recommandations orientées métier pouvant aider à optimiser en plus le modèle obtenu.

II La méthodologie d'entraînement du modèle

Les données ont pu être récupérées sur Kaggle (Kaggle, 2018) et comme tout projet de Data science, une étape de prétraitement des données a été réalisée en se servant d'un bout de code (Aguiar, 2018) existant provenant d'un profil ayant déjà traité ce sujet. Notons que ce prétraitement a permis de créer plusieurs autres features conduisant à un nombre total de 795.

Une fois les données prétraitées, en raison de faible capacité mémoire et pour des raisons de réduction du temps de calcul, seules 20000 instances ont fait l'objet d'étude, soient 16000 cas pour le train set et 4000 pour le test set. Il s'agit de choix aléatoire de cas en respectant les proportions de défaut de paiement.

II.1 Approches d'évaluation de modèle de classification

Nous nous sommes penchés sur des modèles tels que KNeighborsClassifier, SVMClassifier(modèle SVC), MLPClassifier, LGBMClassifier, GradientBoostingClassifier, XGBoostClassifier, RandomForestClassifier, LogisticRegression et ainsi que le modèle le plus basique DummyClassifier.

La recherche d'hyperparamètre s'est faite par validation croisée avec un nombre de plis spécifié par la stratégie k-fold stratifiée (ici k =5) et préférentiellement avec l'usage RandomizedSearchCV pour une exécution plus rapide.

Rappelons que le jeu de données est constitué de 8% de clients ayant des défauts de paiement contre 92% de bon clients : on parle de déséquilibre de classe. En général, ce déséquilibre augmente nettement la difficulté de l'apprentissage par les algorithmes de classification (Kengmegni, 2019). Dans ce cas d'espèce, l'évaluation des modèles ne peut être basée uniquement sur l'Accuracy : les algorithmes auraient peu d'exemples de la classe minoritaire sur lesquels apprendre et leurs prédictions seraient donc biaisées vers la population des négatifs. On aurait des prédictions potentiellement moins robustes qu'en l'absence de déséquilibre.

Pour résoudre ce problème deux approches sont explorées :

- Le choix de métrique d'évaluation adaptée ;
- Traitement des données par des techniques de rééchantillonnage et d'ajustement de poids en cas de déséquilibre des classes observées.

II.1.1 Choix de métriques d'évaluation adaptées à la situation

Il existe des métriques d'évaluation de performances de modèle très efficaces à savoir les métriques Kappa de Cohen, la courbe lift Le Brier Score, F1-score, l'aire sous la courbe ROC.

Une courbe ROC (Receiver Operating Characteristic) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification ; Cette dernière est la métrique mise en avant dans ce projet pour définir les meilleurs modèles.

On peut comparer des modèles en affichant leurs courbes ROC. Ensuite, on peut choisir quel modèle performe le mieux. Pour le choisir, il faut se baser sur l'aire sous la courbe (Area Under the Curve). Plus l'aire sous la courbe est grande, meilleur est notre modèle. Ainsi sur la figure ci-contre, D représente la courbe la plus intéressante et la courbe A celle sans intérêt.

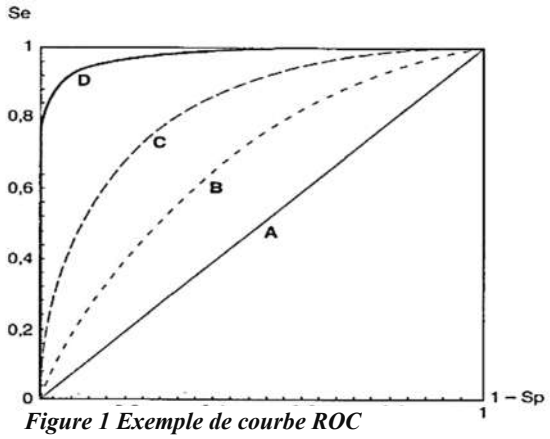

Figure 1 Exemple de courbe ROC

La **Sensibilité *Se*** représente le taux de vrais positifs sur toute la population supposée positive quand la **Spécificité** est la proportion de vrais négatifs, ainsi ***1-Sp*** correspond au taux de faux positifs sur toute la population négative a priori.

On comprend que le seuil idéal est défini par le meilleur compromis entre sensibilité et spécificité : ce qui correspond au modèle au plus fort AUC de ROC : Les seuils les plus intéressants correspondent au coin supérieur gauche de la figure. Un bon test doit être à la fois sensible que spécifique. Dans la pratique ce choix de seuil optimal passe par le modèle qui minimise le plus le coût de l'erreur avec la métrique « roc_auc »

II.1.2 Processing de données : le ré-échantillonnage dans le cadre du déséquilibre de classes et l'ajustement de poids de la classe minoritaire

C'est aussi une étape importante pour la réussite d'un tel projet de classification de classes déséquilibrées. Deux techniques connues de gestion de classes déséquilibrées ont été mises en place :

- **SMOTE** ou la technique d'oversampling : il s'agit d'un suréchantillonnage des données de sorte donc à la création d'observations synthétiques dans la classe minoritaire qui constitueront le jeu d'apprentissage uniquement, puis à évaluer le modèle sur le jeu de test non transformé.

- **L'ajustement de poids de classes via class_weight** uniquement implémenté pour certains modèles tels que LGBMClassifier, LogisticRegression, XGBoostClassifier, RandomForestClassifier : le but étant de pénaliser davantage la fonction de perte aussi bien pour une instance mal classée dans la classe minoritaire qu'une autre mal classée dans la classe majoritaire. Pour ce projet, nous avons obtenu les meilleurs résultats avec class_weight correspondant à « ***balanced »***, de sorte à égaliser les poids des classes de façon automatique donc de rétablir l'équilibre.

II.1.3 Choix du meilleur modèle sur le jeu de validation (validation croisée)

Après entrainement des différents modèles susmentionnés on obtient les aires sous la courbe ROC suivants :

Tableau 1 Récapitulatif de performances selon la technique appliquée

		MODELES						
		KNeighbors	MLP	GradientBoosting	RandomForest	LogisticRegression	XGBoost	LGBM
Techniques	***SMOTE***	0.60	0.69	0.74	0.68	0.72	0.75	0.74
	Ajustement de poids	***X***	***X***	***X***	0.70	0.75	0.68	0.75

NB : Argument ***scale_pos_weight = total_negative_examples / total_positive_example*s** pour XGBoost.à la place de la class_weight.

Quelle que soit la technique de gestion de classes déséquilibrées LogisticRegression et LGBM donnent des valeurs de l'aire sous la courbe ROC intéressantes : ils permettent également d'employer une technique simple moins coûteuse en mémoire et en temps : l'ajustement des poids via le paramètre « class weight ». LogisticRegression et LGBM constituent les modèles choisis pour une optimisation métier.

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

Comme introduit ci-dessus, l'aire sous la courbe ROC ne réponds pas au problème métier, un nombre élevé de faux négatifs ferait perdre le capital investi.

Il convient de s'approprier la matrice de confusion (*Tableau 2*) afin de repérer le paramètre (Vieille, 2016) clé sur lequel agir. Cette matrice permet de décrire les performances d'un modèle de classification sur un ensemble de données de test dont les vraies valeurs sont connues., elle aide à définir les différentes métriques d'évaluation (*Tableau 3*)

Tableau 2 Exemple de Matrice de confusion

Matrice de confusion	Classe négative prédite	Classe positive prédite
Classe négative réelle	Vrais négatifs	Faux positifs
Classe positive réelle	Faux négatifs	Vrais positifs

Tableau 3 Métrique d'évaluation liée aux mauvaises classifications

$$Accuracy = \frac{Vrais\ positifs + Vrais\ négatifs}{Total}$$

➔ Pourcentage de bonnes prédictions

$$Recall = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs}$$

➔ Indication sur les faux négatifs

$$Precision = \frac{Vrais\ positfs}{Vrais\ positifs + Faux\ positifs}$$

➔ Indication sur les faux positifs

Les fonctions coût **Recall** et **Precision** nous permettent d'accéder respectivement aux faux négatifs et aux faux positifs. Par ailleurs il existe des fonctions de type **F*β-score*** de la librairie Scikit-Learn qui permettent de manipuler simultanément les deux fonctions précédentes. La formule (Tremblay, 2021) est :

$$F_\beta\text{-score} = (1+\beta^2)\ \frac{Precision\ Recall}{(\beta^2\ Precision)+Recall} \rightarrow F_\beta\text{-score} = \frac{Vrais\ positifs}{Vrais\ positifs + \frac{1}{1+\beta^2}(\beta^2 Faux\ négatifs + Faux\ positifs)}$$

Ainsi : Pour β≥1, on accorde **plus d'importance au Recall** (autrement dit aux faux négatifs). β≤1, on accorde **plus d'importance à la Precision** (autrement dit aux faux positifs).

Dans notre étude, voulant réduire coûte que coûte le nombre de faux négatifs on choisit $\beta = 3$ conduisant à de bien meilleurs résultats qu'avec $\beta = 2$. À noter que dans la réalité terrain un faux négatif coûte environ 10 fois plus cher à l'entreprise qu'un faux positif prédit. Ci-dessous les matrices de confusion des deux meilleurs modèles que nous avons ressorti pour $\beta^2 \approx 10$ sur une classification de 61502 individus :

LogisticRegression avec $\beta = \sqrt{10}$ LGBMClassifier avec $\beta = 3$

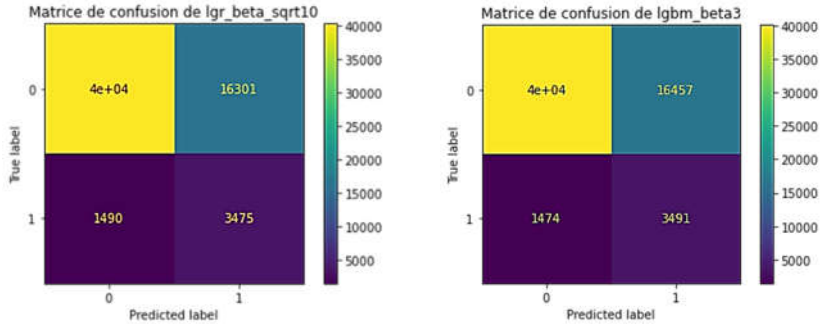

Figure 2 : Matrices de confusion de modèles

LGBMClassifier est de loin le meilleur modèle pour répondant au problème métier : c'est le modèle qui réduit au mieux les faux négatifs source de perte pour l'entreprise. On voit bien les limites de la métrique de l'aire sous la courbe ROC pour deux modèles différents ayant les mêmes aires.

IV L'interprétabilité globale et locale du modèle

Dès lors que nous choisissons notre modèle de prédiction il est aussi important de le comprendre.

Ci-après le modèle et ses hyperparamètres :

Modèle	Hyperparamètres
LighGBMClassifier	class_weight='balanced', max_depth=3, n_estimator=400, num_leaves=127, reg_alpha=0.5

L'interprétabilité (VÉRINE, 2019) de notre modèle répond à la question « comment » ce dernier prend-il une décision. Elle vise à représenter l'importance relative de chaque variable. Depuis 2018, le Règlement Général de l'Union européenne sur la Protection des Données (RGPD) exige que toute décision importante ou de nature juridique puisse être expliquée.

Cela est possible avec la méthode SHAP (**SH**apley **A**dditive ex**P**lanations) qui est une approche théorique des jeux pour expliquer la sortie de tout modèle d'apprentissage automatique. Il relie l'allocation optimale des crédits aux explications locales en utilisant les valeurs classiques de Shapley de la théorie des jeux et leurs extensions associées : on passe d'un modèle **black-box** à un modèle compris :

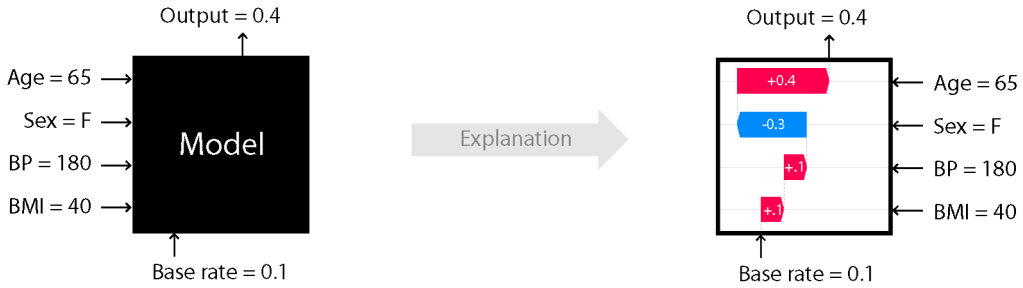

Figure 3 : Exemple d'interprétabilité de modèle (Source : (SHAP, 2018))

SHAP, permet de visualiser l'importance relative globale (ou sur un échantillon) (*Figure 1*) sur tout le jeu de données. On a la possibilité de comprendre la classification faite par le modèle pour une observation locale (*Figure 3*) mais aussi de mettre en évidence les caractéristiques globales importantes mises en avant par ce dernier pour prendre des décisions.

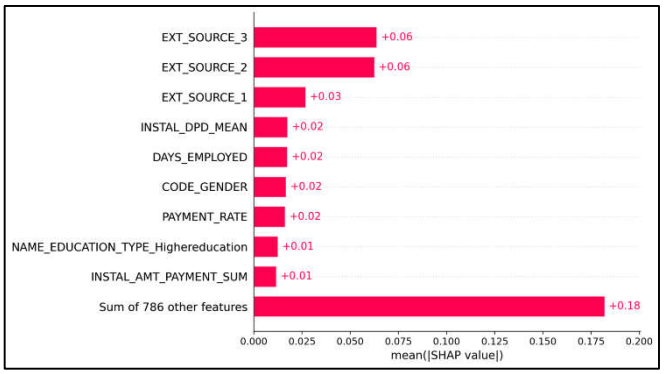

Figure 4 : Feature importance globale

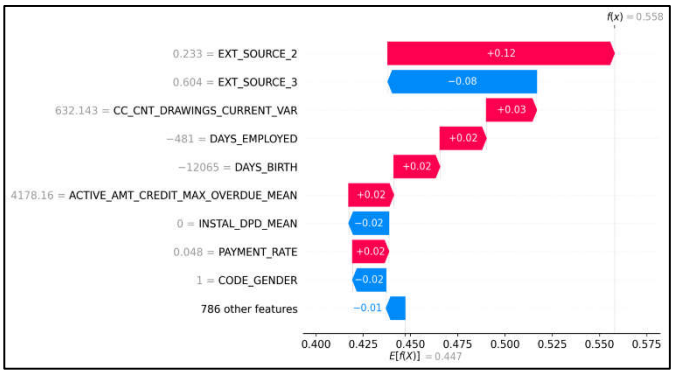

Figure 5 : Feature importance locale

Pour l'exemple du modèle LGBMClassifier optimisé pour nos prédictions, un algorithme appelé **TreeExplainer**, optimisé pour les arbres de décision a permis de l'interpréter sur l'ensemble des données. Ce qui conduit à une attribution d'importance globale plus élevée aux caractéristiques EXT_SOURCE, de la durée de l'emploi DAYS_EMPLOYED, etc... (*Figure 4*).

Par ailleurs, la *Figure 5* a pour objectif d'aider de façon considérable les chargés de relation client à plus de transparence vis-à-vis des décisions d'octroi de crédit pour un client donné. Pour ce cas d'espèce, a priori ce client a une plus faible probabilité (p=0.464) de défaut de paiement avec les features EXT_SOURCE (ici les barres en rouge) qui le pénalisent contrairement aux features telles que DAYS_EMPLOYED, AMT_ANNUITY (ici les barres en bleu), etc...

V Les limites et les améliorations possibles

Ce projet de prédiction de défaut de paiement a pu aboutir grâce à des données clients ayant subi un prétraitement, une modélisation avec bien évidemment le choix de meilleurs mais aussi et pour finir le choix d'une métrique d'évaluation pour répondre au besoin métier.

Il est à noter que le prétraitement s'est réalisé avec un code extérieur issu du site de compétition Kaggle, et le feature engineering associé a fourni des features métiers intéressantes. On pourrait citer en exemple la feature PAYMENT_RATE qui d'ailleurs fait partie du top 10 des meilleures features.

Cependant, fort est de constater lors de cette étude que l'affichage de toutes les features du graphique de features importance globale a montré que plusieurs features n'apportaient rien ou pratiquement rien comme contribution à la prise de décision du modèle. Il serait probablement intéressant de supprimer ces features pouvant créer des bruits perturbateurs de l'apprentissage du modèle et ainsi de réduire ses performances prédictives.

Par ailleurs pour être plus proche des conditions métiers vis-à-vis du profit de « **Prêt à dépenser** » le réalisme auquel est confronté l'entreprise impose de définir une **fonction coût Fβ-score** afin de réduire les faux négatifs qui constituent une perte énorme pour l'entreprise, le paramètre β étant défini aléatoirement au-dessus de 1 conduisant à un nombre de faux négatifs aussi aléatoire. On pourrait peaufiner cette métrique de sorte à atteindre le plus proche possible un objectif d'un nombre de faux négatifs en accord avec le métier pour la rendre plus performante ou d'en créer une autre respectant le plus possible un taux préalablement défini de faux négatifs.

Somme toute, un regard non purement technique du côté des sciences de données mais plutôt qui tient compte de connaissances métiers quel que soit le domaine pourrait permettre non seulement d'arriver à de bien meilleurs résultats de prédiction mais aussi de répondre de façon plus précise à un cahier de charge bien défini.

Références

Aguiar. (2018). Récupéré sur Kaggle: https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script

Kaggle. (2018, 5 18). Récupéré sur Kaggle: https://www.kaggle.com/c/home-credit-default-risk/data

Kengmegni, G. (2019, 12 16). Récupéré sur Quantmetry: https://www.quantmetry.com/blog/classification-et-desequilibre-de-classes/

SHAP. (2018). *SHAP*. Récupéré sur SHAP: https://shap.readthedocs.io/en/latest/index.html

Tremblay, C. (2021, 11 17). *kobia*. Récupéré sur Kobia: https://kobia.fr/classification-metrics-f1-score/

VÉRINE, A. (2019, 09). *Wavestone*. Récupéré sur Wavestone: https://www.wavestone.com/app/uploads/2019/09/Wavestone_Interpretabilite_Machine_learning.pdf

Vieille, M.-J. (2016). Récupéré sur Lovely Analytics: https://www.lovelyanalytics.com/2020/05/26/accuracy-recall-precision/