Note méthodologique

Projet n°7 – Implémentez un modèle de scoring

Développement d'une application de classification pour l'octroi de crédit.

Note méthodologique

Projet n°7 – Implémentez un modèle de scoring

A propos

Une brève note qui permet de comprendre les grands axes empruntés pour aborder ce projet, des liens sont disponibles en fin de note pour approfondir des sujets simplement survolés ici.

I. Présentation du contexte

Dans le cadre du parcours de data scientist d'Openclassrooms, nous nous plaçons dans la peau d'un employé de l'entreprise fictive « Prêt à dépenser », spécialisée dans l'octroi de crédit. Nous sommes chargés de développer une application de classification de client en fonction de leur solvabilité prédite.

II. Traitement des données

a. Présentation des données

Les données sont issues de Kaggle, les deux fichiers principaux, *application_train* et *application_test* compilent des données sur respectivement 307511 et 11 clients.

On y retrouve des données sur le genre, l'âge, l'emploi, les précédents crédits et beaucoup d'autres informations.

Des données externes, récoltées auprès d'organismes extérieur, apparaissent également mais sous forme numérique sans que l'on ait le détail de ce qu'il représente. On dispose également des données du « bureau », dans le domaine du crédit, ces données concernent le passé bancaire du client.

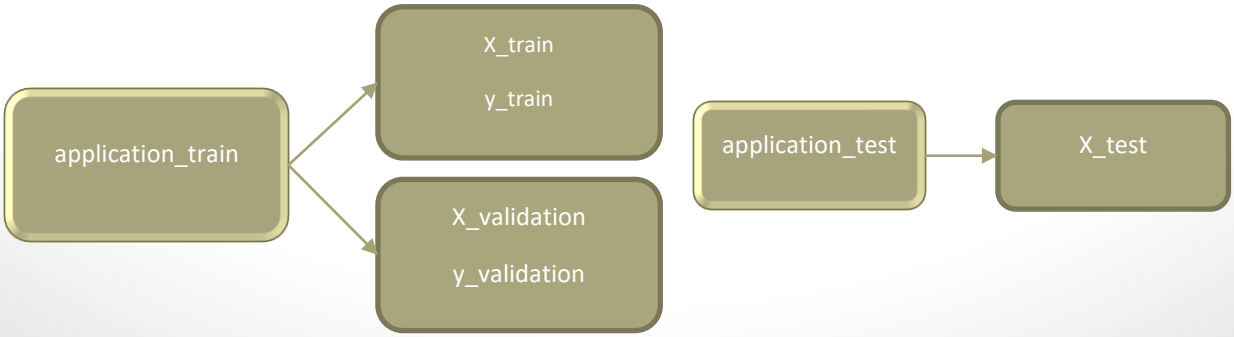
La *target* étudiée ici est une classe binaire, O, indiquant un client solvable, 1, un client non-solvable. La faible représentation de la classe 1, qui constitue seulement 8% de notre population, demandera une attention particulière.

b. Feature engineering

III. Modélisation

a. Méthode d'entraînement

Après avoir effectué une analyse exploratoire et du feature engineering sur les datasets *application_train* et *application_test*, les premiers modèles sont entrainés sur un sample de ces datasets pour optimiser le traitement. Les fichiers sont divisés selon le schéma suivant :



Le déséquilibre de la classe prédite ne doit pas être négligé, en effet, un modèle naïf qui prédirait uniquement des clients solvables (classe 0), obtiendrait un score de 92%, alors qu'il apparaît évident qu'un modèle ne détectant aucun client à risque serait dangereux pour l'entreprise.

Pour pallier ce problème, plusieurs solutions sont testées, notamment SMOTE, via la librairie imblearn, qui possède des méthodes sous-échantillonnage, de suréchantillonnage et des méthodes hybrides[i]. Les modèles de boosting utilisés ultérieurement tels que XGBoost et LightGBM possèdent des paramètres qui gèrent également les déséquilibres de classes.

b. Choix du modèle

Après des essais avec un modèle naïf, et différents modèles de classification, le choix d'utiliser le LightGBMClassifier a été retenu.

L'algorithme se distingue notamment sur sa simplicité d'utilisation, sa rapidité d'exécution, sa capacité à gérer les valeurs manquantes et les variables catégorielles.

c. Hyperparamètres du modèle

Le modèle à été travaillé avec RandomizedSearchCV afin d'optimiser au maximum la performance du LightGBMClassifier.

A noter que dans ce projet, si on observait un déséquilibre très prononcé de la valeur cible, l'utilisation de SMOTE n'a pas apporté d'amélioration assez significative pour être utilisé dans le modèle final. Le rapport temps d'exécution/bénéfice n'étant pas favorable.

IV. Interprétation des résultats

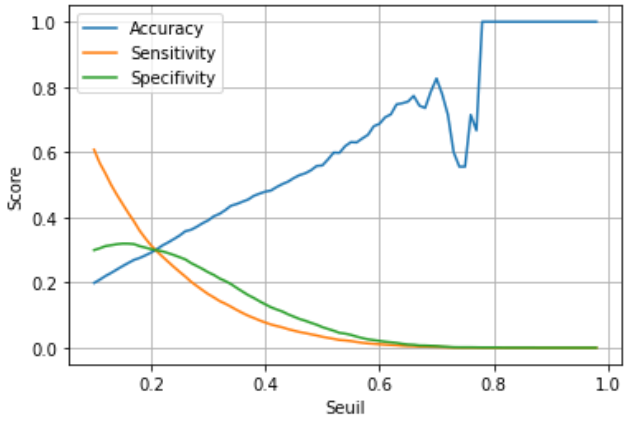
a. Fonction coût métier

Le modèle retourne une classification binaire par sa fonction predict. Le projet demande que l'on privilégie ici l'utilisation de la fonction predict_proba, celle-ci calcule une « probabilité » (il ne s'agit pas réellement d'une probabilité[ii]) pour chaque entrée d'appartenir à la classe 0 ou 1. Par défaut le seuil pour déterminer la classe est de 0,5. Ce seuil arbitraire ne correspond pas la problématique métier posée.

Plusieurs solutions sont envisageables pour affiner le modèle, sachant qu'il n'existe pas de solution parfaite. L'arbitrage se fait entre la perte sèche (somme non recouvrée) lié à un client non-solvable et le manque à gagner (coût d'opportunité) lié à un client peu à risque qui se voit refuser un crédit.

1ère solution : Estimer qu'un client non-solvable non prédit (Faux Négatif) est 10 fois plus couteux pour l'entreprise.

2ème solution : Considérer le taux de 8% de non-solvabilité observer pendant l'analyse exploratoire et d'observer la valeur du predict_proba de la classe 1 pour les 8% les plus haut.



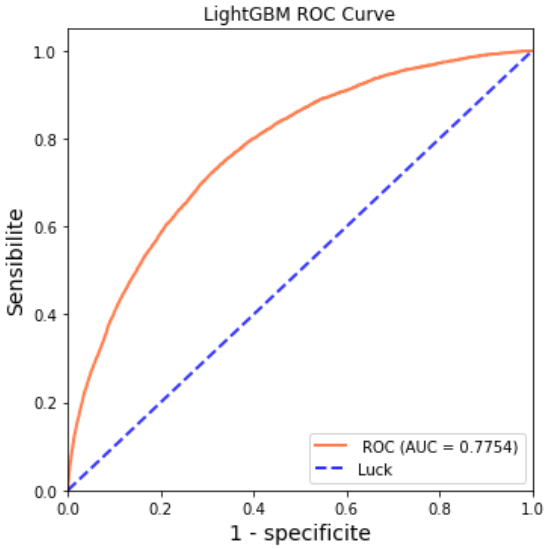
Une recherche de seuil en fonction des résultats du modèle est possible, elle correspond dans ce projet à la 2ème solution envisagée, ici le seuil aura une valeur de 0,20.

b. Métriques

Matrice de confusion métier :

	Clients prédit non-solvables (1)	**Clients prédits solvables (0)**
Clients non-solvables (1)	Vrais Positifs	Faux Négatifs
Clients solvables (0)	Faux Positifs	Vrais Négatifs

Pour évaluer la qualité du modèle, l'utilisation de la courbe ROC, et de l'aire sous cette courbe (AUC) a été privilégié.



c. Importance des variables

Outre l'utilisation de la fonction feature_importances implémentée dans le classifieur LightGBM, qui permet d'avoir un premier aperçu des features qui sont les plus importantes dans le calcul de la prédiction, il a paru nécessaire dans ce projet d'aller un peu plus loin et de tenté d'exploiter les valeurs de Shapley pour une explication de l'influence de chaque feature sur la prédiction.

ILLUSTRATION DE LA FONCTION FORCE_PLOT AVEC LA LIBRAIRIE SHAP

V. Limites et améliorations

Si les résultats permettent d'apporter une plus-value comparée à des modélisations naïves, il reste néanmoins de nombreux axes d'améliorations possibles.

D'abord dans le traitement des données brutes, de nombreuses valeurs manquantes n'ont pas été traitées ou de manière grossière (imputation par moyenne ou pas mode). Il serait également possible de pousser plus loin le feature engineering. Une connaissance plus fine des compétences métier ajouterait une forte plus-value.

Ces mêmes compétences permettraient d'optimiser la classification en affinant le seuil du predict_proba. Il serait possible de calculer le bénéfice/risque des différentes stratégies de l'entreprise.

Faute de puissance de calcul, des options n'ont pas pu être explorées pleinement, notamment des modèles avec plus de features, des variables catégorielles encodées avec One-Hot-Encoder combiné au SMOTE.

Enfin il serait possible d'exploiter de manière plus exhaustive les valeurs de Shapley pour l'interprétabilité du modèle, malgré le fait que features '*SOURCE_EXT*' ne soit pas explicite.

---

[i] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[ii] https://towardsdatascience.com/pythons-predict-proba-doesn-t-actually-predict-probabilities-and-how-to-fix-it-f582c21d63fc