2021

P7 : Implémentez un modèle de scoring.

Note méthodologique

Ce document présente la méthodologie et les concepts mobilisés lors de l'étude prédictive menée pour la modélisation des accords de prêts en réponse aux demandes faites par les clients de la société Prêt à dépenser. Dans un souci absolu de transparence, nous faisons état de la méthodologie appliquée pour la conception de notre modèle. Nous présentons la fonction coûts métier, l'algorithme d'optimisation ainsi que la métrique score orientée métier. Nous présentons aussi, la méthode d'interprétation des résultats appliquée et enfin nous faisons les remarques, mettons en évidence les limites et proposons des conseils pour améliorer cette méthode.

Steven SON
Prêt à dépenser
23/12/2021

La méthodologie d'entraînement du modèle (2 pages maximum)

L'ensemble des analyses menées a été réalisé à l'aide du langage Python et ont été interprétés par Python version 3.8.5.

Le modèle que nous avons mis en place vise à évaluer la probabilité qu'un client rembourse son crédit et par conséquent à lui accorder ou à lui refuser le prêt qu'il demande. Pour faire cela, nous disposions d'une base de données contenant des informations client. Cette base de données comprenait des informations historiques et des informations ponctuelles sur chaque client. Le recueil des informations ponctuelles de chaque client était présent sous forme de tableaux au format (.csv) sous les noms « application_train.csv » et « application_test.csv ». Le premier contenait un ensemble de 307511 lignes et 122 colonnes dont 9 152 465 cellules étaient vides et le second constitué de 48744 lignes et de 121 colonnes présentait compte à lui un ensemble de 1 404 419 cellules vides. Après nettoyage, exploration et *feature engineering* sur ces données, il a été nécessaire de déterminer le modèle qui pourrait au mieux, permettre de répondre à la problématique métier exigée.

Au cours de la phase de *feature engineering*, nous avons utilisé les données du fichier « application_train.csv » et non celles du fichier « application_test.csv » car ce dernier ne disposait pas de la variable 'Target' qui est l'information binaire selon laquelle le client pourrait ou non rembourser son prêt. C'est de cette variable 'Target' dont nous avions besoin pour calibrer notre prédiction. Nous avons alors séparé les données d'« application_train.csv » en données d'entrainement et en données test grâce à la fonction train_test_split du module model_selection de Scikit-learn. Pour faire cela, nous avons fixé le paramètre test_size à 0.2 de façon à obtenir un jeu d'entrainement à 80% du jeu de départ et un jeu de test à 20% du jeu de départ. Par la suite, c'est sur ce jeu d'entrainement que le modèle a été entrainé.

Pour déterminer quel modèle nous devions utiliser, nous avons étudié les caractéristiques des variables explicatives et de la variable 'Target' à expliquer. En ce qui concerne cette variable 'Target', nous avons observé qu'elle était déséquilibrée dans le jeu de donnée d'entrainement car nous disposions de 282686 clients à qui on n'attribuerait pas le prêt et 24825 clients à qui on l'attribuerait. Cela signifierait que si l'on entraine notre modèle sur ce jeu, il serait plus probable de ne pas attribuer le prêt. Cependant, les clients ayant exigés plus de transparence dans les décisions, il a fallu équilibrer cette variable de façon à ce que notre modèle soit lui aussi plus transparent. Pour faire cela, nous avons employé des méthodes de rééquilibrage des données qui visent à un sur-échantillonnage (ex. : SMOTE) ou un sous-échantillonnage des données (ex. : RandomUndersampler). Ces méthodes ont pour objectif de soit sur-échantillonner la classe sous représentée, soit de sous-échantillonner la classe sur représentée. Il semblerait que dans le papier originel de la méthode SMOTE, il soit conseillé de coupler ces deux méthodes pour optimiser la technique[1]. C'est par conséquent ce que nous avons réalisés, nous avons créé un Pipeline contenant RandomUndersampler et SMOTE.

Nous avons déterminé qu'il était possible d'expliquer ces données en employant soit une Régression logistique, soit un Classifieur Xgboost, soit un LightGBM et soit un Classifieur en forêt aléatoire (Randomforest classifier) puisque nous sommes ici en présence d'un problème de classification. Pour chacun de ces modèles, nous avons alors fait une recherche d'hyperparamètres par validation croisée en employant la méthode de GridSearchCV du module model_selection de Scikit-learn. Le choix s'est alors fait sur le modèle qui présentait le meilleur score obtenu (Tableau 1), ce dernier a été conçu spécialement pour répondre à la problématique métier[2].

[1] https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

[2] Pour plus d'informations, se référer à la section conception du score.

Tableau 1: Paramètres optimisés et scores obtenus pour les modèles entrainés sur le jeu de données.

Modèle	Paramètres optimisés[3]	Score obtenu
Régression logistique	max_iter = 500, solver = 'lbfgs'	0.4518
Classifieur Xgboost	gamma=0, learning_rate=0.01, max_depth=3, scale _pos_weight= 1	0.4726
LightGBM	boosting_type='gbdt', colsample_bytree= 1.0, learning_rate=0.1, n_estimators=200, subsample=0.8	0.1828
Randomforest classifier	max_depth=10, n_estimators=500	0.4027

Nous avons déterminé que le meilleur modèle à sélectionner pour notre jeu d'entrainement correspondait à un classifieur Xgboost (réalisé avec la version 1.3.0. de Xgboost). Un classifieur Xgboost est un algorithme de *Gradient Boosting*[4] qui consiste à assembler plusieurs « apprenants faibles » (qui correspondent à des algorithmes présentant des performances peu élevées) pour en concevoir un « apprenant fort » qui est plus efficace et satisfaisant. La particularité de cette méthode est que contrairement au *Bagging* (où chaque apprenant faible est indépendant du précédent), ici, chaque apprenant faible permet de corriger les résultats de l'apprenant faible précédent. La particularité de Xgboost par rapport aux autres algorithmes de *Gradient Boosting* est qu'il effectue le *pruning*[5]. C'est-à-dire que sur chaque apprenant faible, il y a un élagage réalisé de façon à optimiser les performances de chacun. Si l'apprenant n'est pas suffisamment performant alors il sera supprimé. Ce qui a pour conséquences d'améliorer les performances globales du modèle.

Le modèle Xgboost que nous avons entrainé possédait les paramètres les suivants : base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.5, enable_categorical=False, eval_metric=make_scorer(custom_metric), gamma=0, gpu_id=-1, importance_type=None, learning_rate=0.01, max_delta_step=0, max_depth=3, min_child_weight=1, n_estimators=100, n_jobs=12, num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.8, tree_method='exact', use_label_encoder=False, validate_parameters=1, verbosity=None.

[3] Pour plus d'informations sur les paramètres de modèles, se référer à la documentation du modèle.
[4] Source : Fereirra *et al.*, 2021. Comparison of autoML tools for machine learning, deep learning and XGBoost
[5] Source : https://datascientest.com/algorithmes-de-boosting-adaboost-gradient-boosting-xgboost

La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation (1 page maximum)

Nous avons pu mettre en place un score à visée métier qui évalue à quel point il est possible de se tromper lors de nos prédiction (Figure 1).

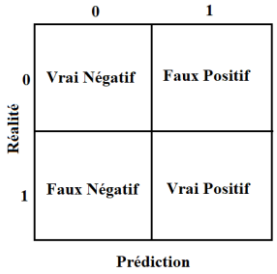


Figure 1 : Table de contingence servant à expliquer les cas de Vrais positifs, Vrais négatifs, Faux positifs et faux négatifs dans toute situation de prédiction.

C'est-à-dire que dans un premier temps, nous avons créé notre score. Pour faire cela, nous avons créé une fonction « cutom_metric » prenant en paramètres les valeurs réelles (correspondant à la variable 'Target') de l'accord ou du refus de prêt pour chaque client, ainsi que les valeurs prédites obtenues par modélisation pour chaque client. Lors de la comparaison de ces valeurs, quatre situations étaient possibles ; deux cas où les valeurs prédites sont égales aux vraies valeurs (situations de vrai négatifs et vrais positifs), et deux cas où les valeurs prédites sont différentes des vraies valeurs (situations de faux positifs et de faux négatifs). Nous avons déterminés pour notre problématique que les Vrais positifs sont doublement bénéfiques ; pour les clients et pour la société « Prêt à dépenser ». C'est le cas car ils permettent à un client qui veut un prêt et qui peut le rembourser de le faire et pour la société d'obtenir un client et de retrouver ses fonds. C'est pour cela que la valeur de '1' a été attribuée aux clients qui étaient dans ce cas. Nous avons déterminés que les Vrais négatifs étaient bénéfiques pour la société car même s'ils ne rapportent pas de nouveaux clients, ils permettent à l'entreprise de ne sélectionner que des clients qui peuvent effectivement rembourser leur prêt. C'est ainsi qu'une valeur de '0.5' a été attribuée à ces clients. Nous avons déterminés qu'une mauvaise prédiction pouvait être plus pénalisante qu'une autre. En effet, un Faux positif est plus grave pour le client et pour la société qu'un Faux négatif. En conséquence, lors de la création de ce score, nous avons attribués moitié moins de points pour les Faux positifs que pour les Faux négatifs. De façon plus concrète, nous avons attribués '0.25'points à un Faux négatif et '0' point pour les Faux positifs (Figure 2).

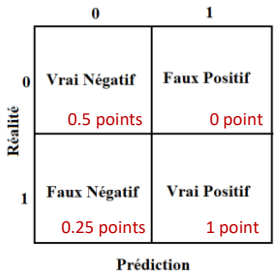


Figure 2: Répartition des points attribués pour la création du score à visée métier.

La conception du score se fait donc en calculant la moyenne des différents cas analysés. Il s'agit d'une métrique qui est comprise entre 0 et 1 sachant qu'il sera meilleur plus on se rapprochera de 1. Au cours de l'analyse des données, nous nous sommes rendu compte que les données étaient déséquilibrées, nous avons tenté de corriger cela à l'aide de méthodes de sur-échantillonnage ou de sous-échantillonnage. Cependant, lors de l'analyse des résultats présentés par le modèle, nous nous sommes rendu comptes que la proportion de clients à qui le prêt serait refusé était toujours importante par rapport à l'ensemble des clients. Pour contre balancer ce fait, nous avons pris en compte cela dans l'évaluation du modèle. À l'aide de la fonction make_scorer du module metrics de Scikit_learn, nous l'avons signifié en entrant dans make_scorer, notre custom_metric et en paramètre greater_is_better nous avons sélectionné la valeur 'True'.

L'interprétabilité globale et locale du modèle (1 page maximum)

À cause du nombre important de variables (63 variables importantes), nous comprenons le modèle présente une certaine complexité. Pour l'interpréter et l'expliciter à tout interlocuteur, nous avons fait appel à la méthode Local Interpretable Model-Agnostic Explanations (LIME) et à la méthode SHapley Additive exPlanation (SHAP). La méthode LIME consiste à expliquer localement *a posteriori* des prédictions de modèles de *machine learning*. Cette méthode commence par la mise en place de l'interpréteur. Ici, il s'agit d'un interpréteur de tableaux car les données sont présentées sous forme de tableau et en paramètres d'entrées, on y intègre les données, on y intègre le mode de prédiction, les données cibles, le nom des variables explicatives et on fixe l'état aléatoire. Puis, on crée l'explication de l'interpréteur pour une prédiction en particulier. Enfin, on montre les résultats explicatifs sous forme graphique (Figure 3).

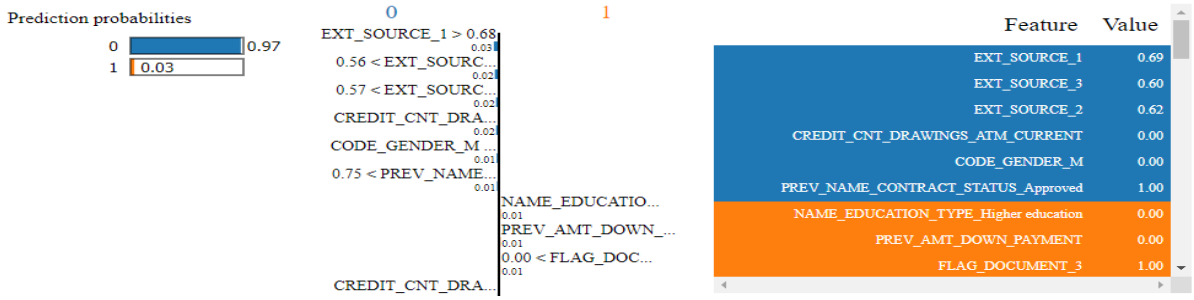


Figure 3: Résultat de la méthode LIME appliquée au client 153551. Les variables sont présentées dans l'ordre décroissant de leur importance. En bleu sont représentées les variables qui influencent la prédiction vers le score 0. En orange sont représentées les variables qui influencent la prédiction vers le score 1. Ici, on observe que la variable EXT_SOURCE_1 est très impliquée dans la prédiction de score 0 pour le client.

Le graphique présente de façon décroissante l'implication des variables par rapport à la probabilité de classification. Elle montre donc les variables qui ont le plus de poids dans la décision, leur valeur et la valeur seuil décisionnelle, ainsi que la prédiction pour l'individu d'intérêt. La méthode SHAP[6] vise à une explication locale des prédictions réalisées mais en agrégeant ces explications locales, il est possible d'obtenir une explication globale valide sur ces prédictions. Cette méthode commence par la mise en place de l'interpréteur qui ici est un interpréteur d'arbre de décisions utilisés pour la classification. À partir de cet interpréteur auquel on intègre le modèle de *machine learning* optimisé ainsi que le jeu de données d'entrainement du modèle, il est possible en y ajoutant les données de tests du modèle, de calculer les valeurs de Shapley qui correspondent à la contribution de chaque individu statistique pour la prédiction globale du modèle. Ces valeurs permettent de voir comment la prédiction est équitablement distribuée entre les différentes entrées du modèle. Grâce à ces valeurs, il est possible de construire différents graphiques explicatifs tels que le « force_plot » ou le « summary_plot » en autres (Figure 4).

Figure 4: Force_plot réalisé sur le Classifieur Xgboost optimisé et sur notre jeu de données.

L'intérêt du « force_plot » est de permettre à l'utilisateur de visualiser localement, c'est-à-dire pour chaque client, l'implication des variables par rapport à la valeur moyenne de la variable. Le « summary_plot » compte à lui permet de visualiser l'impact positif ou négatif d'une variable sur la valeur de prédiction d'un modèle. Il permet de comparer les variables, de quantifier leur impact, de comprendre leur action cumulée ainsi que de différencier les variables continues des variables binaires (Annexe 1).

[6] Source : Lundberg & Lee 2017 . A unified approach to interpreting model predictions

Les limites et les améliorations possibles (1 page maximum)

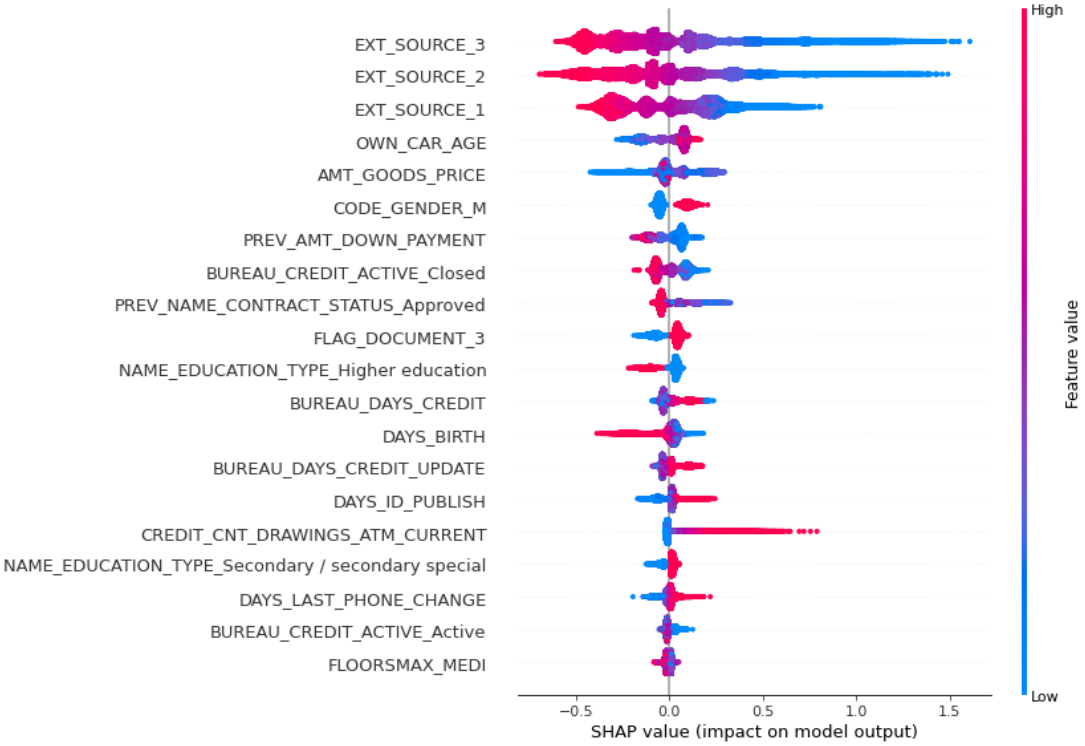
Au cours de la réalisation de notre projet, plusieurs éléments ont pu réduire nos marges d'interprétations telles que l'optimisation des modèles. En effet, par soucis de gain de temps de *processing*, l'optimisation a été réalisée sur un set réduit de modèles et de paramètres, ce qui fait que nous avons testé uniquement quatre modèles prédictifs et qu'un autre modèle soit plus approprié. Le modèle choisi a été optimisé uniquement sur une gamme de paramètres. Pour des résultats plus précis, il faudrait ré-optimiser l'ensemble des hyperparamètres du modèle. Seules 3 cross validations ont été réalisés pour la détermination des hyperparamètres. En effet, il aurait été nécessaire de déterminer le nombre de cross validations à réaliser, mais pour faire cela, il faudrait également disposer de ressources supplémentaires dont on ne disposait pas ici.

Une autre information qui aurait été utile lors de la conception des modèles, c'est une explication plus précise sur certaines variables. En effet, il est possible que lors du remplissage de certains formulaires par les clients que des variables dépendent les unes des autres. En effet, un client qui ne dispose d'aucun bien peut par exemple ne disposer d'aucune information sur un prêt précédent, ce qui peut entrainer ici des biais concernant l'attribution de prêt ou non. Les nouveaux clients seraient potentiellement favorisés ou défavorisés par rapport à d'autres, or par souci de transparence, il est nécessaire que tous les clients soient considérés de la même façon.

En termes d'améliorations possibles, il aurait été utile que dans le fichier « application_test.csv », que la variable 'Target' soit présente de façon à ce que nous n'ayons pas besoin de splitter le fichier « application_train.csv ». En effet, cela aurait pour conséquences que nous travaillions sur un jeu de données plus grand et par conséquent que nos prédictions soient plus fiables. Il aurait été utile d'avoir moins de données manquantes.

Il est nécessaire de rappeler à tous les utilisateurs de notre modèle qu'il ne s'agit que d'un modèle prédictif et il est possible que certains clients se trouvent à la limite de l'intervalle de confiance prédictif de ce dernier. En effet, parmi les informations dont nous disposons, nous n'avons pas le capital épargné par les clients. Il est possible que certains clients puissent bénéficier de prêts mais que le modèle ne puisse les détecter à cause du fait que ces informations ne sont pas prises en compte par ce dernier. Il faut alors que les clients dans cette situation se rapprochent des conseillers pour qu'un modèle plus complet intégrant ces informations leur soit appliqué.

Annexes



Annexe 1: Summary_plot réalisé sur notre Classifier Xgboost et sur nos données. Les variables sont rangées par ordre décroissant d'importance.