NOTICE MÉTHODOLOGIQUE DU MODÈLE DE PRÉDICTION

PROJET 7

Implémentez un modèle de scoring

Prêt à dépenser

Table des matières

1 – Données et preprocessing : 3

1.1 – Contexte : 3

1.2 – Données et preprocessing : 3

1.3 – Modification apportée au kernel : 4

2 – Modélisation : 5

2.1 – Décision d'utiliser le modèle du Kernel Kaggle : 5

2.2 – Difficultés de l'élaboration du modèle : 5

2.2.1 – Nombre de variables : 5

2.2.2 – Classe minoritaire : 5

2.3 – Modèle et entraînement : 6

2.3.1 – LightGBM : 6

2.3.2 – Entraînement du modèle : 6

2.3.3 – Scores : 6

3 – Interprétabilité du modèle : 7

3.1 – Interpréter un modèle de forêts aléatoires : 7

3.2 – Interprétabilité globale : 7

3.2 – Interprétabilité locale : 8

4 – Limites et améliorations possibles : 9

4.1 – Adapter le modèle à la politique du commanditaire 9

4.2 – Communication et compétence métier 9

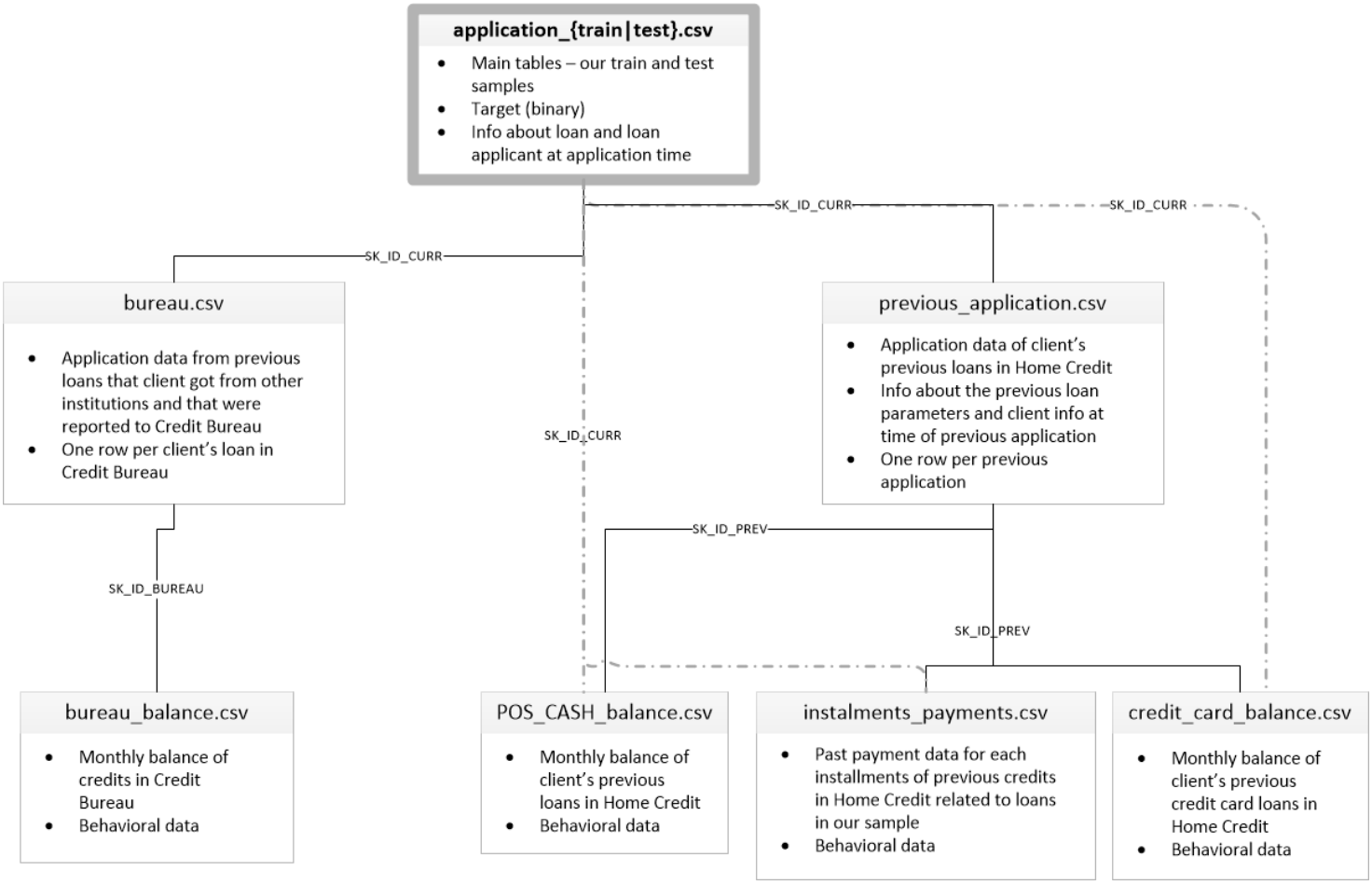
1 – Données et preprocessing :

1.1 – Contexte :

Le jeu de données est issu d'une compétition Kaggle organisée en août 2018 dont l'objectif était de prédire les incidents de payements dans le cadre de l'octroi de prêts bancaires.

1.2 – Données et prétraitement :

Le dataset est complexe et volumineux (356 255 individus et 221 variables). Il est réparti sur 7 tables dont voici le schéma :



Du fait de cette complexité, et afin de gagner du temps, il était recommandé de récupérer un kernel Kaggle de prétraitement des données ; celui utilisé se trouve à l'adresse ci-dessous :

https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script

Ce kernel ne comporte ni nettoyage d'ampleur ni standardisation[1]. En revanche, il présente un énorme travail de *features engineering* avec la création de nombreuses variables synthétiques (dans le but notamment de transposer la temporalité des données en des termes appréciatifs tels que la variance, la moyenne, le minimum et le maximum). Enfin, l'encodage des variables catégorielles porte le nombre de colonnes à 797.

Le dataset obtenu présente des données d'entraînement – qui disposent donc de l'information cible – et des données dites de test dont les meilleures prédictions consacraient le podium du concours Kaggle. Dans notre cas, ces données de test ont servi de base à l'élaboration d'une API et d'un dashboard.

1.3 – Modification apportée au kernel :

Un seul changement a été effectué au sein du kernel : la suppression de la variable « CODE_GENDER ». Il s'agit, en effet, d'un choix à la fois légal et éthique, il est, en effet, inenvisageable pour un organisme de crédit de justifier d'une décision d'octroi sur la base du sexe de la personne.

[1] L'absence de standardisation est possible du fait de l'utilisation d'un modèle de Random Forest.

2 – Modélisation :

2.1 – Utilisation du modèle du Kernel Kaggle comme baseline :

Un modèle était présent dans le kernel Kaggle ; ce dernier a été utilisé comme baseline.

2.2 – Difficultés de l'élaboration du modèle :

La modélisation présente deux difficultés majeures : un nombre de variables très important, et un déséquilibre des classes.

2.2.1 – Nombre de variables :

Après prétraitement et encodage, le dataset compte pas moins de 797 variables. Plutôt que d'effectuer une sélection ou de pratiquer des réductions de dimensions, notre modélisation conserve l'intégralité de ces informations. L'utilisation de LightGBM, un algorithme de forêts aléatoires, permet de pallier les problèmes liés à une telle quantité de variables (colinéarités et temps de calcul notamment). À ce titre, la baseline a été très efficace.

2.2.2 – Déséquilibre des classes :

Dans le jeu de données d'entraînement, les individus coupables d'incidents de payement ne représentent que 8 % de l'ensemble de la population. Un déséquilibre des classes entraîne souvent une mauvaise modélisation ; la fonction de minimisation des erreurs des algorithmes conduit ces derniers, en effet, à éviter la prédiction de la classe minoritaire. Il existe différentes approches à ce problème telles que l'*undersampling* (la réduction du nombre d'individus de la classe majoritaire) ou l'*oversampling* (la création d'individus synthétiques au sein de la classe minoritaire, soit en dupliquant les individus existants, soit en les créant à partir de méthodes statistiques telles que la méthode des plus proches voisins).

Dans notre cas, la baseline ne parvenait pas à capter correctement la classe minoritaire. Or, dans le cadre de l'octroi d'un crédit, la fonction coût métier oblige à orienter nos résultats vers la détection de cette classe. Pour ce faire, il a fallu abandonner l'UAC pour le score F1 et attribuer des poids à la classe minoritaire (poids déterminés par RandomSearchCV).

2.3 – Modèle et entraînement :

2.3.1 – LightGBM :

LightGBM est un algorithme de forêts aléatoires avec optimisation par *gradient boosting* initialement développé par Microsoft. Son fonctionnement est un peu différent de son concurrent XGBoost. Les performances des deux modèles sont assez proches, mais LightGBM a l'avantage d'être beaucoup moins lourd en calculs (d'où son nom), ce qui l'avantage sur les jeux de données massifs.

2.3.2 – Entraînement du modèle :

Le modèle a été entraîné avec une crossvalidation à 10 plis stratifiés afin de garantir la présence de la classe minoritaire dans les testsets, du moins lorsque cela est possible[2].

Les hyperparamètres sont généralement déterminés par *Gridsearch* ou *Randomsearch*. Ici, c'est par une optimisation bayésienne qu'ils ont été obtenus (via la librairie *Hyperopt*). Notons que le retrait de la variable Gender_code (genre) n'a pas influencé les scores et que les hyperparamètres sont restés pertinents. Ils n'ont pas été recalculés après les modifications du modèle (score F1 + poids) ; c'est un axe d'amélioration à noter pour l'avenir.

2.3.3 – Scores :

Compte tenu de la problématique et d'une optimisation qui n'en est qu'à ses débuts, les résultats sont relativement satisfaisants. Le F1 atteint 0.75 sur la classe minoritaire (0.86 au global) avec un rappel de 0.88. Bien sûr, cela implique un nombre important de faux négatifs qui se traduit par une précision à 0,66 ; c'est malheureusement le prix à payer si l'on souhaite capter la classe minoritaire.

[2] Si un pli ne le permet pas, la crossvalidation s'effectue alors sans stratification

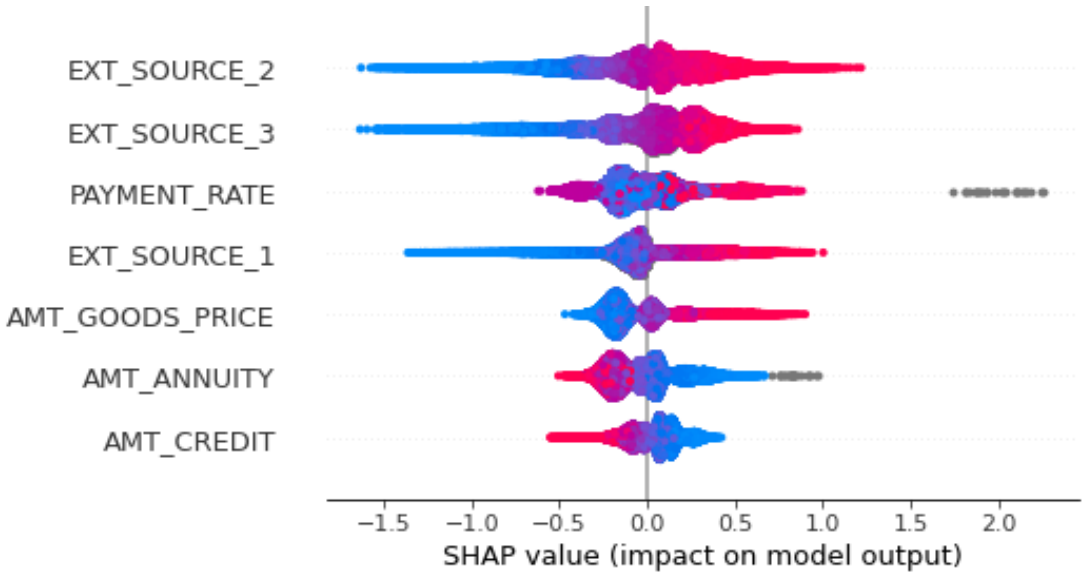
3 – Interprétabilité du modèle :

3.1 – Interpréter un modèle de forêts aléatoires :

Dans le cadre de l'octroi d'un prêt, l'interprétation du modèle est un impératif ; il est en effet inenvisageable, ne serait-ce que légalement, de refuser un prêt à un individu sans pouvoir lui en donner les raisons. Dans la plupart des modèles mathématiques, ces raisons sont facilement récupérables (il s'agit tout simplement des coefficients de la fonction). Interpréter un modèle de forêts aléatoires ou de réseau de données, en revanche, est chose plus complexe et nécessite le recours à un algorithme d'interprétation. Dans notre cas, c'est la librairie Shap qui a été utilisée.

3.2 – Interprétabilité globale :

L'interprétabilité globale montre l'influence décisive de trois variables qui correspondent à des scores de sources externes. Elles sont accompagnées par la variable, « PAYMENT_RATE », qui correspond à la part de l'emprunt remboursé chaque année. Viennent ensuite le montant des annuités, le nombre de jours salariés total, et la moyenne des retards de payements.



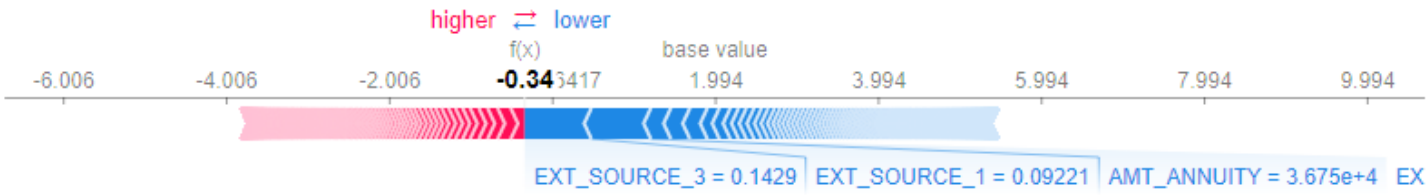
De façon contrintuitive, plus l'individu compte de jours salariés, et plus ses chances de défaut de payement sont importantes. On peut supposer que cela est lié à l'âge (plus l'individu compte de jours salariés et plus il a de chances d'être âgé). Cela peut aussi être

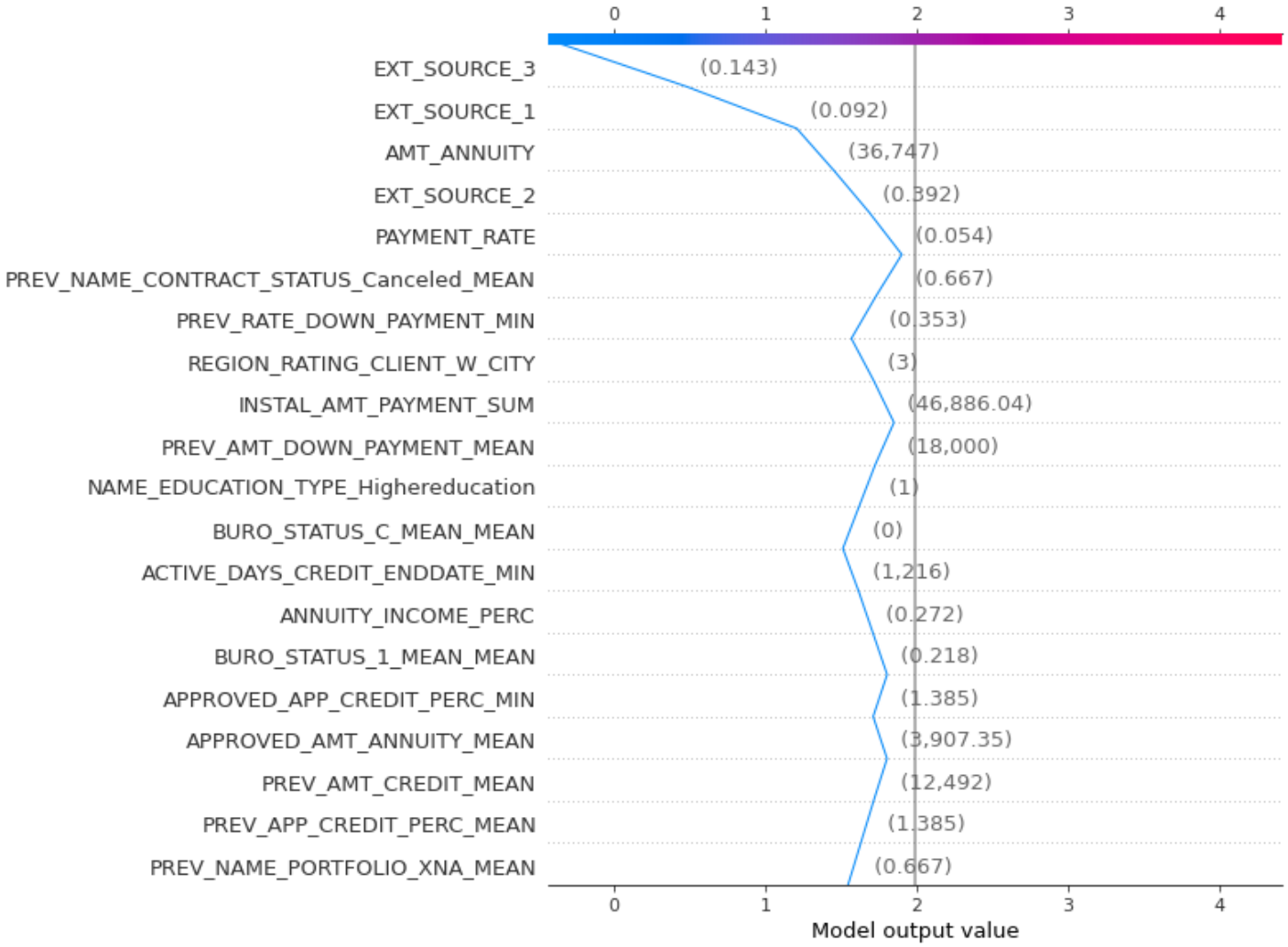
lié à l'évolution des revenus : quelqu'un qui débute dans sa profession ayant évidemment plus de chance d'obtenir des promotions que quelqu'un qui s'approche de la retraite.

3.2 – Interprétabilité locale :

L'interprétabilité locale s'effectue également via la librairie Shap. Les individus testés confirment l'influence importante des scores de sources externes (« EXT_SOURCE »). En revanche, force est de constater que les indicateurs traditionnels utilisés notamment dans l'API (revenus, âge, montant emprunté…) peinent à expliquer les prédictions du modèle.

Voici, pour exemple l'interprétation d'un individu dont la prédiction est un défaut de payement :





4 – Limites et améliorations possibles :

Les limites de ce modèle (et par conséquent, les améliorations possibles) sont principalement liées à l'absence de communication avec les commanditaires de celui-ci :

4.1 – Adapter le modèle à la politique du commanditaire

Notre modèle a été entraîné par maximisation d'un score F1 lesté. Les poids visent bien sûr à capter la classe minoritaire, mais une partie de celle-ci (12%) échappe encore au modèle. Il est possible d'améliorer encore cette captation, mais au prix, certainement de davantage de refus de crédits à de potentiels bons payeurs. Est-ce vraiment ce que souhaite l'organisme de crédit commanditaire ? Malheureusement, il est impossible d'en avoir la certitude.

4.2 – Communication et compétence métier

Les métiers du financement font appel à un savoir-faire et un vocabulaire spécifique. Il est parfois difficile de saisir les subtilités des différentes variables de crédit. Sans aide extérieur, il a fallu très souvent s'en remettre à une compréhension personnelle perfectible.

Exemple symptomatique : trois des variables les plus importantes de notre modèle[3] correspondent à des scores provenant d'organismes étrangers, dont les modalités de calcul nous sont donc totalement inconnues. Encore une fois, il est regrettable de ne pas pouvoir poser la question.

Signalons enfin que le choix des KPI et des graphiques affichés dans l'API aurait été certainement bien plus pertinent en collaboration avec des professionnels du secteur.

Pour toutes ces raisons, l'élaboration du modèle gagnerait à ce que des communications formelles (réunion et cahier des charges) et non formelles (coup de téléphone ou mail pour obtenir une réponse rapide) puissent avoir lieu.

[3] « EXT_SOURCE »