Projet scoring crédit

Note méthodologique

La présente note s'inscrit dans le cadre du projet de développement d'un algorithme de scoring crédit pour des clients ayant peu ou pas d'historique de prêt. Elle décrit la méthodologie adoptée lors de ce développement ainsi que le modèle retenu et ses performances.

L'algorithme en question doit non seulement calculer, pour un client donné, la probabilité que celui-ci rembourse le crédit qu'il demande, mais aussi déterminer s'il faut le lui accorder ou non, selon que la probabilité calculée est inférieure ou supérieure à un certain seuil, que l'on aura réglé de manière à minimiser les coûts pour l'entreprise.

Sommaire

La méthodologie de développement du modèle

Métriques d'évaluation & fonctions coût métier

Résultats

Optimisation du seuil

Interprétation des résultats du modèle

Les limites et les améliorations possibles

L'analyse du data drift

La méthodologie de développement du modèle

La plupart des étapes du traitement des données, depuis leur préparation jusqu'au calcul de la probabilité de défaut de paiement, ont été incluses dans un même pipeline, mais certains nettoyages ont été effectués préalablement. À terme, une inclusion de toutes les étapes dans le pipeline rendrait le processus plus automatique.

Pour le moment, dans le pipeline ont été incluse les étapes suivantes :

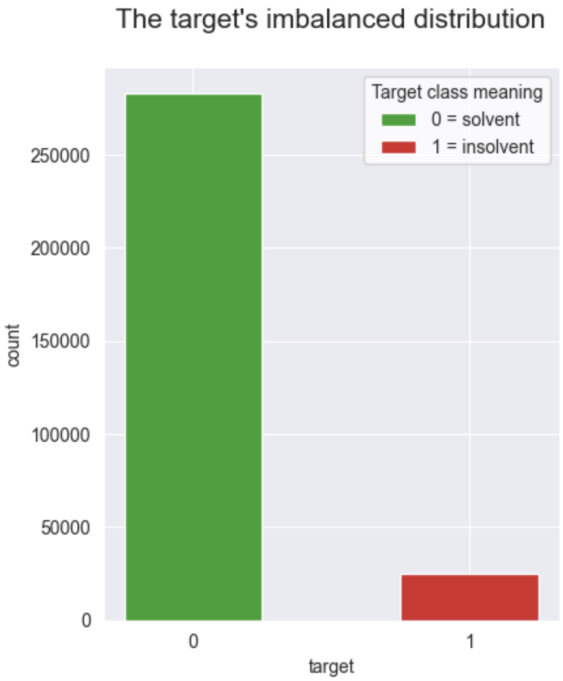
- le rééquilibrage des classes
- l'imputation des valeurs manquantes
- la mise à l'échelle des différentes variables numériques (scaling)
- l'encodage des variables catégorielles
- le calcul de probabilité par le classifieur

Chaque étape a fait l'objet d'une grid search (la dernière, de deux : une pour le choix du classifieur, et une autre pour celui des hyperparamètres) pour tester différentes options.

1. rééquilibrage des données

test avec/sans sous-échantillonnage

Option retenue : effectuer un sous-échantillonage (avec le RandomUnderSampler de la librairie imbalanced-learn)



Le jeu de données contient seulement 8,1 % de positifs, et ce déséquilibre des classes nuit aux performances (ce que l'on a vérifié en mesurant celles-ci lorsque que l'on ne corrige pas le déséquilibre). Mais le nombre de positifs, s'il est faible relativement à celui des négatifs, reste en soi assez élevé puisque l'on compte 24 835 positifs. J'ai jugé ce volume suffisant pour entraîner le modèle, et j'ai donc privilégié un simple sous-échantillonnage comme solution au déséquilibre, pour éviter d'avoir recours à des données fabriquées.

Le sous-échantillonnage améliore grandement les performances par rapport à l'option qui consiste à ne pas rééquilibrer les données.

2. imputer

test de SimpleImputer (médiane et moyenne), IterativeImputer, KNNImputer

Option retenue : SimpleImputer qui remplace les données manquantes par la médiane.

3. scaler

test de StandardScaler, RobustScaler, Normalizer, PowerTransformer, QuantileTransformer, MinMaxScaler

Option retenue : RobustScaler

4. encodeur

test de OneHotEncoder, WOEEncoder, GLMMEncoder, JamesSteinEncoder

Option retenue : JamesSteinEncoder

5. classifieur

LogisticRegression, RandomForestClassifier, LGBMClassifier, GradientBoostingClassifier (+ DummyClassifier comme point de comparaison)

Option retenue : LGBMClassifier

6. hyperparamètres

n_estimators : [100, 200, 300, 400, 500],
learning_rate : [0.15, 0.1, 0.05],
max_depth : [2, 3, 4, 5]

Option retenue : n_estimators=300, learning_rate=0.1, max_depth=4

Métriques d'évaluation & fonctions coût métier

Accorder un prêt à un client qui ne le remboursera pas coûte en général plus cher à l'entreprise que de ne pas accorder de prêt à un client qui aurait pu le rembourser, car dans le second cas, seul les intérêts sont perdus (ils sont « perdus » au sens où ils représentent un manque à gagner). Autrement dit, un faux négatif est plus coûteux qu'un faux positif.

J'ai utilisé trois métriques qui prennent en compte chacune à leur manière cette différence de coût :

- une fonction de perte qui compte 1 pour chaque faux positif, et 10 pour chaque faux négatif
- un score F10 : un score F-beta est la moyenne harmonique pondérée de la précision (precision) et du rappel (recall). Le paramètre beta représente le ratio de l'importance du rappel par rapport à celle de la précision. Avec beta=10, on donne 10 fois plus d'importance au rappel qu'à la précision. Le rappel est ici le taux de clients catégorisés comme insolvables parmi ceux qui le sont effectivement, et la précision, à l'inverse, le taux de clients effectivement insolvables parmi ceux qui sont catégorisés comme tels. Le F10 score donne donc 10 fois plus d'importance à la détection des clients insolvables qu'au fait de ne pas les classer à tort comme insolvables. On en revient ici au même principe que pour la fonction de perte évoquée précédemment, à savoir qu'un faux négatif serait 10 fois plus coûteux qu'un faux positif.
- une fonction Profit & Loss qui calcule la différence entre la somme du profit généré par l'ensemble des vrais négatifs (donc les prêts qui seront remboursés) et la somme des pertes dues aux faux négatifs (les prêts qui ne seront pas remboursés). J'ai fait l'hypothèse qu'en moyenne, un prêt non remboursé est à moitié remboursé, et donc génère une perte égale à la moitié de la somme prêtée ; et en deuxième hypothèse, que le taux d'intérêt d'un prêt est en moyenne de 5 %, et donc génère un profit de 5 % lorsqu'il est remboursé. On a donc gardé ici le même ratio de 1 pour 10 pour le coût estimé des faux positifs par rapport aux faux négatifs. Il s'agit d'hypothèses de travail temporaires, qui pourraient bien sûr être largement améliorées par certaines analyses.

Ces trois métriques ont été utilisées dans un premier temps pour comparer les différentes options testées dans les grid searchs, en plus des métriques ordinaires ci-dessous :

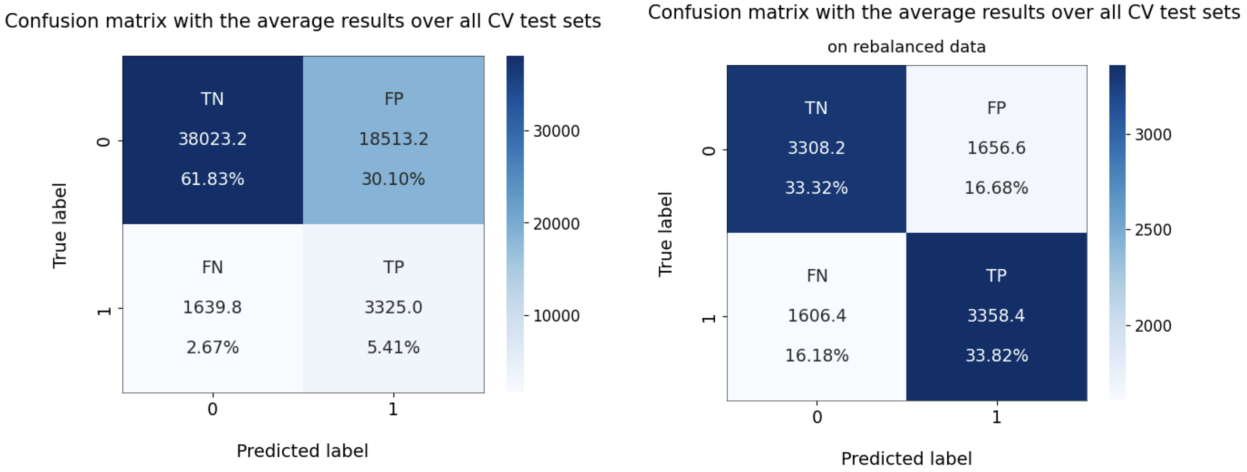
- l'aire sous la courbe ROC
- l'accuracy
- le rappel (recall)
- la précision (precision)
- la log loss, qui mesure l'erreur des probabilités calculées par rapport à la target
- les métriques du temps pris par le modèle (mean_fit_time, std_fit_time, mean_score_time, std_score_time)

Mais je me suis aussi servi de la fonction de perte et de la fonction Profit & Loss pour optimiser le seuil par lequel on détermine si l'on accorde ou refuse le prêt.

Résultats

Les performances du modèle retenu comparées à celles du classifieur idiot (cross-validation 5 splits) :

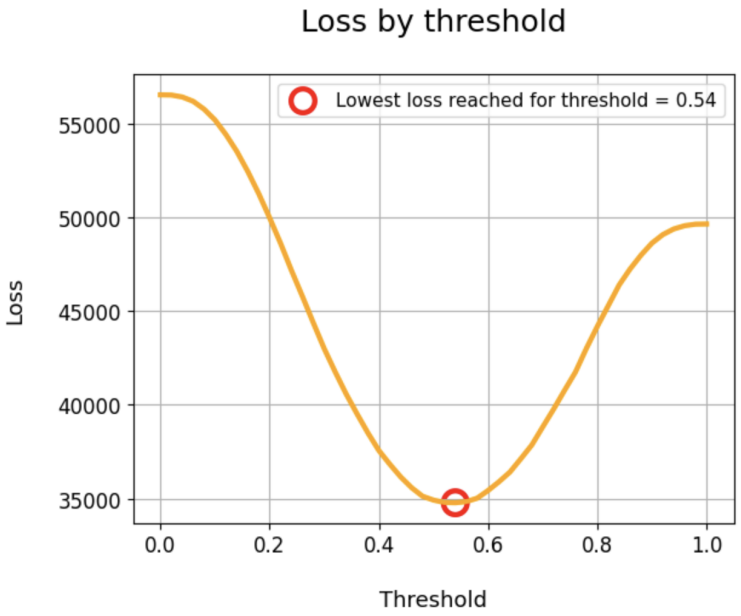
	LGBM	DummyClassifier
mean_test_roc_auc	0.73	0.50
mean_test_accuracy	0.67	0.92
mean_test_recall	0.67	0.00
mean_test_precision	0.15	0.00
mean_test_f10	0.65	0.00
mean_fit_time	5.40	0.35
std_fit_time	0.71	0.02
mean_score_time	1.30	0.51
std_score_time	0.32	0.01

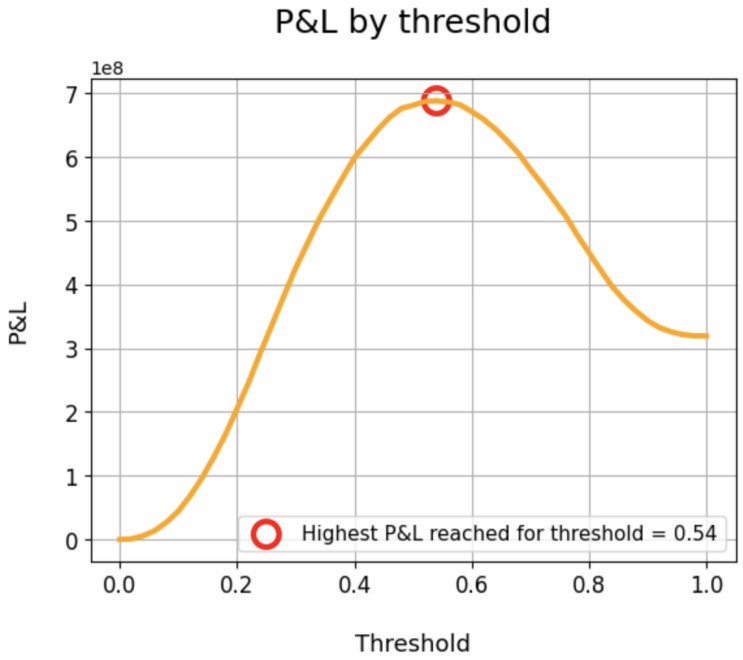


Matrice de confusion des prédictions sur tout le jeu de données (à gauche) et sur le jeu de données rééquilibré par sous-échantillonnage (à droite).

Optimisation du seuil

Toutes les valeurs allant de 0 à 1 avec un pas de 0,02 ont été testées comme seuils potentiels. Que l'on procède au test par ma fonction de perte ou par ma fonction Profit & Loss, on obtient le même résultat : la valeur optimale du seuil est de 0,54. Cela tient au fait que les hypothèses à la base de ces deux fonctions sont très proches (voir plus haut).

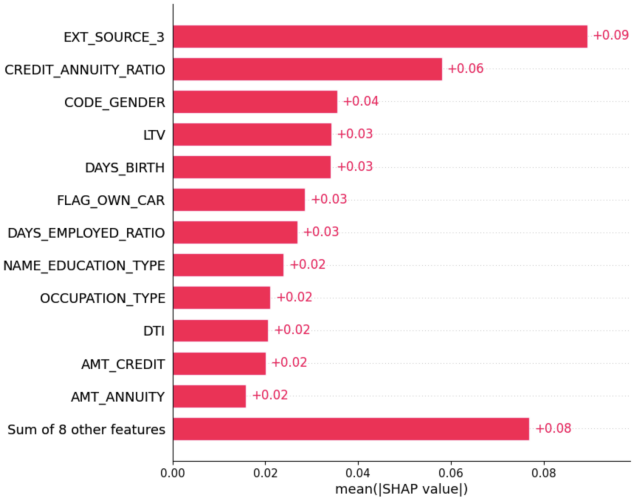


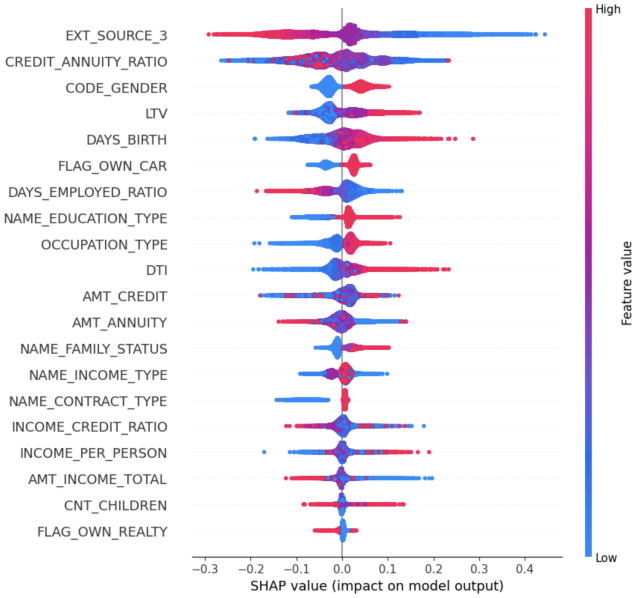


Avec un seuil à 0.54, 70 % des demandes de prêt sont accordées (sur le jeu d'entraînement).

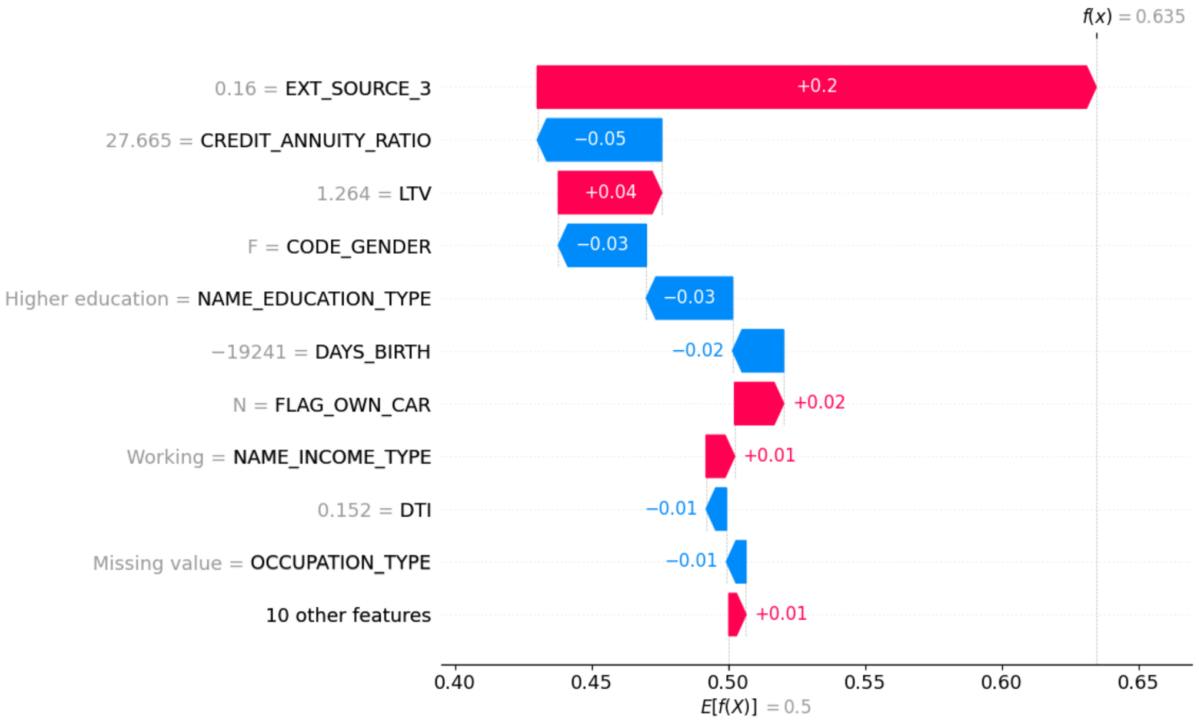
Interprétation des résultats du modèle

La feature importance calculée par les SHAP values, au niveau global :





Exemple d'interprétation au niveau local, pour le client 100001 :



Les limites et les améliorations possibles

1. Je me suis inspiré d'un kernel Kaggle pour le preprocessing (le kernel 'LightGBM with Simple Features'). Je pourrais tirer profit encore d'autres kernels Kaggle.

2. Revenir sur la sélection des features. Une simple sélection par la feature importance a été effectuée, mais d'autres méthodes existent, qui sont probablement plus justes, comme celle proposée dans ces articles Medium, où l'on prend comme critère l'erreur (log loss) plutôt que la feature importance :

   Your Features Are Important? It Doesn't Mean They Are Good | by Samuele Mazzanti | Aug, 2023 | Towards Data Science

   Which Features Are Harmful For Your Classification Model? | by Samuele Mazzanti | Sep, 2023 | Towards Data Science

3. Pour l'optimisation du seuil : améliorer les fonctions coût en affinant leurs hypothèses (coûts relatifs d'un faux positif et d'un faux négatif, etc.).

L'hypothèse qu'un faux négatif est en moyenne dix fois plus préjudiciable qu'un faux positif est assez arbitraire et invite à mener des analyses qui permettraient de trouver une valeur plus juste. De même pour l'hypothèse que le profit de la banque pour un prêt entièrement remboursé est égal en moyenne à 5 % du montant du prêt, ou celle qu'un faux négatif cause une perte en moyenne égale à la moitié du montant du prêt. D'ailleurs, cette dernière hypothèse elle est en fait probablement très éloignée de la réalité, puisque les banques ont la possibilité de saisir les biens du client en défaut de paiement pour se rembourser : le cas où la perte s'élève à 50 % du montant du prêt est probablement un cas relativement rare.

On pourrait aussi faire attribuer un coût aux faux positifs ou aux faux négatifs en fonction de la catégorie du prêt, ou de quelque autre caractéristique dont on remarquerait qu'elle fait varier ce coût.

4. Tester davantage d'hyperparamètres

5. Réviser l'algorithme pour qu'il puisse prendre en compte l'historique de prêt du client quand il y en a un.

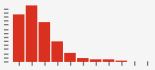
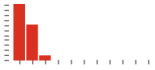
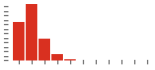
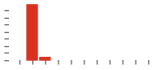
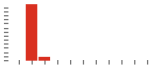
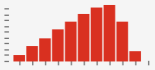
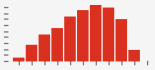
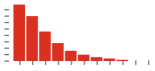
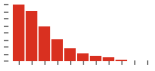
L’analyse du data drift

Il se peut qu’au cours du temps des changements significatifs dans le profil des clients ou d’autres paramètres affectent les performances du modèle.

On observe déjà du data drift sur 5 des 20 variables retenues pour notre modèle entre les données d’entraînement et les données des demandes de prêt en cours. On veillera à réentraîner le modèle sur les données mises à jour au fur et à mesure (lorsque l’on aura la valeur de la target pour celles des demandes en cours qui auront été acceptées). Si le data drift venait à trop augmenter, on envisagerait de reconstruire le modèle.

Drift is detected for 25.0% of columns (5 out of 20).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
CREDIT_ANNUITY_RATIO	num			Detected	Wasserstein distance (normed)	0.567299
INCOME_CREDIT_RATIO	num			Detected	Wasserstein distance (normed)	0.263488
AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207339
AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.161095
NAME_CONTRACT_TYPE	cat			Detected	Jensen-Shannon distance	0.147537
LTV	num			Not Detected	Wasserstein distance (normed)	0.096678
AMT_INCOME_TOTAL	num			Not Detected	Wasserstein distance (normed)	0.094786
INCOME_PER_PERSON	num			Not Detected	Wasserstein distance (normed)	0.071365
EXT_SOURCE_3	num			Not Detected	Wasserstein distance (normed)	0.061818
DAYS_EMPLOYED_RATIO	num			Not Detected	Wasserstein distance (normed)	0.053601

→ ratio montant du prêt/mensualités globalement en hausse, avec changement de la forme de la distribution : plus de ratio faibles et forts, moins de ratio moyens
→ augmentation du ratio revenu / montant du prêt
→ augmentation des montant des prêts
→ augmentation des mensualités
→ chute du taux de crédits renouvelables (revolving loans) par rapport aux prêts personnels (cash loans)