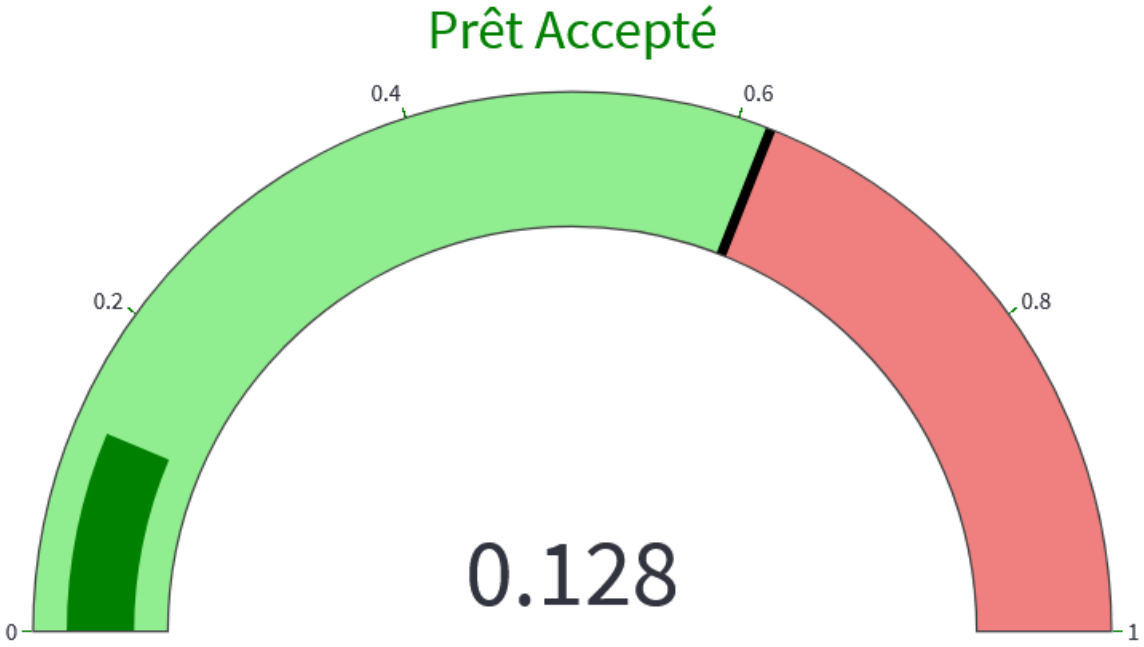
Prêt à dépenser

Mise en place d'un outil de scoring et d'un dashboard interactif



Note méthodologique

Parcours Data Scientist – Projet 7

Github : https://github.com/HDeBoissezon/P7.git

Hélène de BOISSEZON

Juin 2022

TABLE DES MATIERES

CONTEXTE

La société ***Prêt à dépenser*** souhaite mettre en place un outil de scoring crédit afin de calculer la probabilité qu'un client rembourse son crédit. Pour réaliser cela, nous allons nous appuyer sur les données issues de la compétition Kaggle « Home Credit Default Risk ». Cette base de données ayant déjà été travaillée par de nombreuses personnes, nous allons utiliser un kernel kaggle préexistant pour réaliser la préparation des données. Afin de sélectionner et optimiser le modèle de machine learning approprié, une métrique spécifique à notre problématique est créée. Enfin, l'interprétabilité du modèle retenu est explorée, de façon à être en mesure d'expliquer au demandeur de prêt la décision prise par la société ***Prêt à dépenser***.

1 PREPARATION DES DONNEES ET MISE EN PLACE DU PIPELINE

1.1 FEATURE ENGINEERING

L'intégralité des données disponibles dans les 7 fichiers sont traitées. Les différentes features créées sont détaillées dans le fichier *P7_Preprocessing.ipynb*. La démarche générale consiste en :

- des transformations de variables sur la demande de prêt en cours (table application_train) : calcul de pourcentages, revenu moyen
- Aggregation des données concernant les prêts précédant contractés par le demandeur (tables bureau, bureau_balance, previous_application, POS_CASH_balance et installment_payment) et de l'historique des cartes de crédits du client (table credit_card_balance):
    - Données numériques : remontée de min / max / size / mean / var / somme par client
    - Données catégorielles : transformation via un One Hot Encoder et remontée de la moyenne (équivalent à une proportion) par client

Tableau 1 – Détail des données avant et après les différentes étapes de préparation des données (la target est inclue dans le décompte des features)

Données	Nb variables	Nb individus	% valeurs manquantes
Application_train	122	307 511	24.15%
Bureau	17	1 716 428	13.50%
Bureau_balance	3	27 299 925	0.00%
POS_CASH_balance	8	10 001 358	0.07%
Credit_card_balance	23	3 840 312	6.65%
Previous_application	37	1 670 214	17.98%
Installment_payment	8	13 605 401	0.01%
Jeu de données après feature Engineering	678	307 507	30.89%
Jeu de données après nettoyage	354	307 177	5.94%
Jeu de données après feature selection	126	307 177	5.69%
Jeu de données rééquilibré SMOTENC	126	49 710	0.00%

1.2 NETTOYAGE

Certaines variables sont supprimées :

- Les variables correspondant à des valeurs manquantes (37 features)
- Les variables présentant un taux de remplissage inférieur à 75% (287 features)

Et les lignes correspondant à des outliers sont supprimées (valeurs supérieures à Q3 + 5 fois l'IQR ou inférieures à Q1 – 5 fois l'IQR).

1.3 FEATURE SELECTION

Enfin, il est nécessaire de faire le tri dans les différentes variables disponible afin de :

- Eviter des variables redondantes (ie. dont le coefficient de corrélation de Pearson est supérieur à 0.8) : suppression de la variable la moins corrélée à la target (88 features)
- Ne garder que les variables suffisamment corrélées à la target (ie. Dont le coefficient de pearson est supérieur à 0.01 ou, pour les variables catégorielles, dont le coefficient eta² est supérieur à 0.1) : suppression de 140 features

1.4 ENCODAGE, IMPUTATION ET NORMALISATION

Pour la suite du process, il est nécessaire d'encoder les variables qualitatives. Comme il s'agit de variables booleennes, un encodage par label encoder est suffisant.

Après ces différentes étapes il reste des valeurs manquantes (cf. Tableau 1). Elle sont remplacées par une imputation par la moyenne pour les variables quantitatives. Les variables qualitatives sont quant à elles imputées par une valeur constante fixée à -1.

Enfin, les variables quantitatives sont normalisées via un standard scaler.

1.5 SÉPARATION EN TRAIN SET / TEST SET

Les données sont finalement séparées en jeu d'entrainement (80%) et de jeu de test (20%), la variable target étant l'attribution (0) ou non (1) du prêt.

1.6 GESTION DES DONNEES DESEQUILIBREES

La variable cible est fortement déséquilibrée : 92% des prêts sont accordés et seuls 8% sont refusé. Cela pose problème pour l'apprentissage du modèle qui aura tendance, en l'état, à favoriser l'accord de prêt, provocant un nombre trop élevé de faux négatifs (attribution de prêt à des personnes non solvables) faisant encourir un risque financier non négligeable pour ***Prêt à dépenser***. Afin de pallier à cette situation deux approches sont testées.

1.6.1 Pondération

De nombreux algorithmes permettent d'utiliser un argument « class_weight » qui va attribuer un poids plus important à la classe minoritaire de façon à rééquilibrer les classes. Cette méthode revient à attribuer un poids de 11 aux refus de prêt. Cette méthode a l'avantage d'être très simple d'utilisation mais ne va pas permettre de compenser le manque d'information potentiel de la classe sous-représentée.

1.6.2 Undersampling + oversampling

Une autre approche est de générer des individus fictifs à partir des données de la classe sous-représentée (over-sampling, via SMOTENC), ce qui permet d'augmenter la taille de cette classe. Il est par contre fortement conseillé de se limiter à 25% d'individus générés de façon à ne pas risquer de fausser le jeu de données, soit dans notre situation passer d'un ratio 1/0 de 0.088 à 0.11 (création de 5074 individus). Le jeu de données reste donc encore trop déséquilibré (10% de classe minoritaire).

Afin de compléter le ré-équilibrage, un échantillonnage aléatoire est pratiqué (under-sampling) de façon à aboutir à 50% pour chaque classe.

Le jeu de données d'entrainement est donc composé de 24 855 prêts accordés et 24 855 prêts refusés. Cette méthode à comme avantage de constituer un jeu d'entrainement réellement équilibré, par contre elle réduit drastiquement le nombre d'individus. Elle est donc difficilement applicable dans le cas d'une petite base de données (ce qui n'est pas le cas ici).

1.7 MODELES TESTES

Cinq modèles sont testés :

- Dummy classifier : il s'agit de notre baseline, les modèles ne doivent pas performer en dessous de celui-ci.
- Regression logistique : régression linéaire adaptée aux problématiques de classification
- SGD classifier : régression linéaire basé sur l'algorithme de la descente de gradient
- RandomForest classifier : methode de bagging
- LightGBM classifier : gradient boosting

2 EVALUATION ET OPTIMISATION DU MODELE

Afin d'évaluer quelle méthode de rééquilibrage des données est appropriée ainsi que les différents modèles, il est nécessaire de se baser sur une métrique qui s'appuie sur l'analyse métier de la problématique. En l'occurrence, le risque repose sur deux types d'erreurs :

- Erreur de type I (faux positif) : ***Prêt à dépenser*** refuse un prêt à une personne qui est solvable ➔ manque à gagner sur les intérêts que ce client ne va pas payer, mesuré par la précision
- Erreur de type II (faux négatif) : ***Prêt à dépenser*** accorde un prêt à une personne non solvable ➔ perte du montant prêté et des intérêts liés, mesuré par le recall

2.1 METRIQUE D'EVALUATION

En se basant sur les données des prêts précédents (previous_application) et en faisant l'hypothèse (pessimiste) que lorsqu'il y a défaut de paiement, l'intégralité du prêt n'est pas remboursé il est possible de calculer les caractéristiques moyenne d'un prêt (taux, montant, durée et mensualité). Il apparait ainsi que le cout d'un défaut de paiement (erreur de type II) représente 12 fois le manque à gagner généré par une erreur de type I.

Afin de prendre en compte ces risques il est proposé la métrique suivante :

$$score = gain * (VN - FP) - perte * FN$$

Avec ***VN*** : Vrai négatif (accord de prêt à une personne solvable)
FP : Faux positif (accord de prêt à une personne non solvable)
FN : Faux Négatif (refus de prêt à une personne solvable)
Gain : montant des intérêts si le prêt est intégralement remboursé (basé sur les caractéristique moyennes d'un prêt)
perte : montant du prêt et des intérêts afférents si non remboursé (basé sur les caractéristiques moyennes d'un prêt)

Plus le score est grand, plus ***Prêt à dépenser*** gagne de l'argent.

2.2 CHOIX DU MODELE ET OPTIMISATION

Chaque modèle est entrainé avec les paramètres par défaut sur (1) le jeu d'entrainement non rééquilibré, (2) le jeu de données rééquilibré via class_weight et (3) le jeu de données rééquilibré via under et over sampling, soit 15 runs.

Chacun de ces runs est évalué sur le jeu de test et les métriques classiques (AUC, recall, precision, F1), la métrique métier et la durée de calcul sont stockées. Il en ressort, d'un point de vue métrique métier, que les deux meilleurs modèles sont LightGBM et RandomForest à chaque fois sur des données équilibrées.

Ces deux modèles ont donc été optimisés via un GridSearchCV selon les deux méthodes de rééquilibrage (4 runs) sur le jeu d'entrainement et évalués sur le jeu de test. Le modèle choisi au final est un LGBM entrainé sur les données rééquilibrées par under et over sampling, les paramètres détaillés sont disponibles dans le fichier *P7_Modelling.ipynb*.

3 INTERPRETABILITE DU MODELE

Il est très important de pouvoir comprendre et expliquer comment fonctionne le modèle retenu. En effet, la finalité de ce projet est de permettre plus de transparence entre ***Prêt à dépenser*** et le client, ce qui passe par une bonne compréhension des variables importantes de façon générale (interprétabilité globale) et des informations qui ont principalement mené à la décision prise par ***Prêt à dépenser*** pour un client (interprétabilité locale). Pour cela, différentes librairies existe. La plus robuste, SHAP, permet de calculer les valeurs de Shapley pour chaque variable via des permutations (théorie des jeux coopératifs) de façon à distinguer leur rang de contribution au modèle.

3.1 GLOBALE

D'un point de vue fonctionnement général du modèle entrainé on observe sur la Figure 1 que les six premières variables jouant un rôle important sont des variables issues du feature engineering représentant des informations relatives aux prêts précédemment contractés (préfixes PREV_, POS_ et BUREAU_).

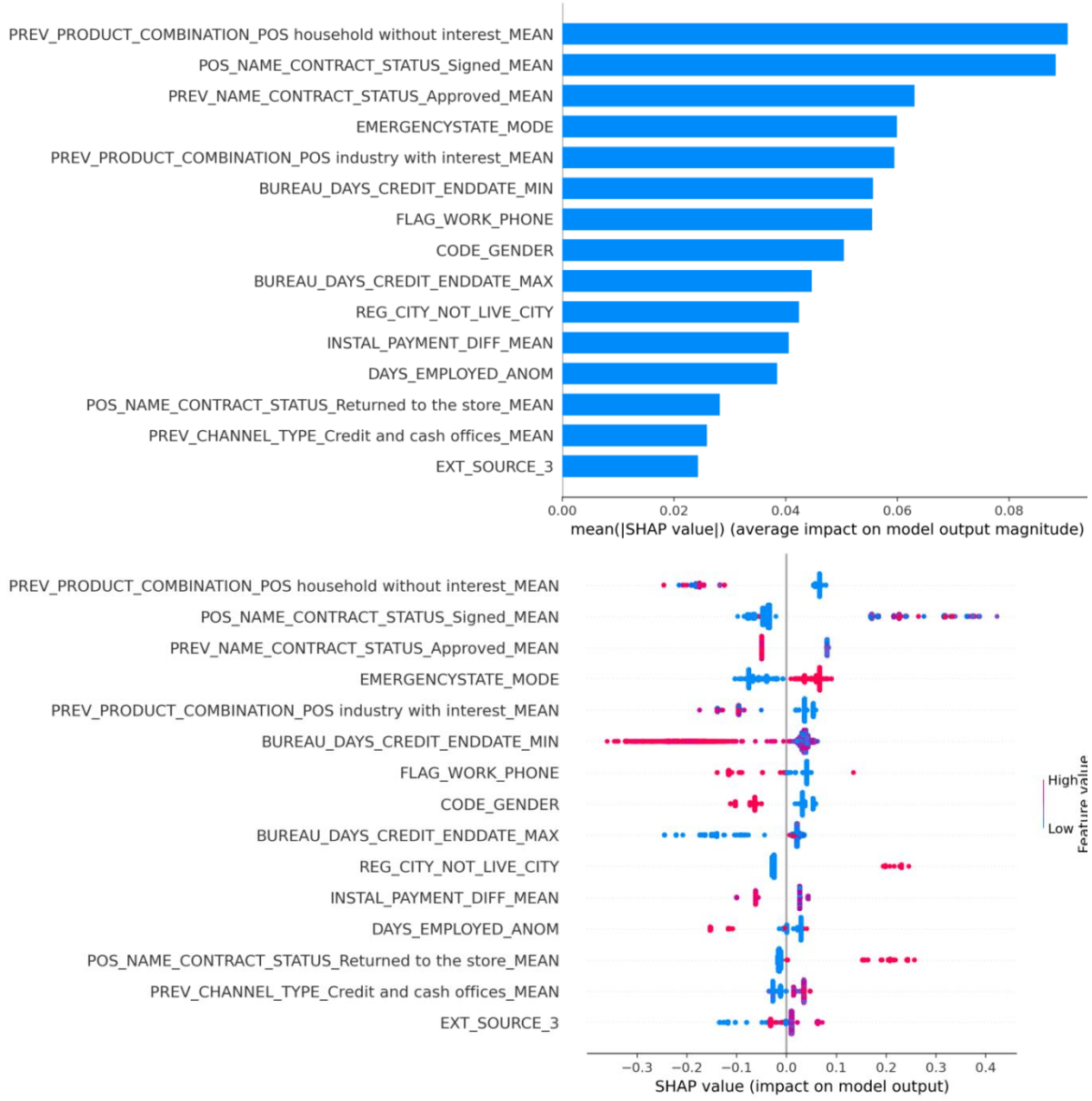


Figure 1 – Feature importance à partir des valeurs de Shapley. En haut, valeur absolue moyenne de shapley pour les 15 première variables. En bas, contribution de chaque individu au score moyen. Exemple de lecture : un CODE_GENDER bas (0=femme) augmente la probabilité de refus de prêt (valeur SHAP positives).

3.2 LOCALE

D'un point de vue local (c'est-à-dire au niveau d'un client donné), il est possible de visualiser les contributions des différentes variables renseignées / calculées à la décision du modèle. En Figure 2 on voit ainsi les différentes variables qui ont contribuées positivement (augmentation de la probabilité de refus de prêt) ou négativement (augmentation de la probabilité d'acceptation du prêt) dans la décision pour le client 100002 (en l'occurrence, le modèle préconise un refus de prêt). Ainsi, la valeur d'EXT_SOURCE_3 (variable calculée en interne) permet de diminuer le risque d'un refus de prêt alors que la valeur d'EMERGENCYSTATE_MODE (variable permettant d'accéder à des informations sur le lieu de vie du client) augmente ce risque.

Figure 2 – Feature importance pour le client 100002 (l'algorithme conclu à un refus de prêt). En haut, diagramme waterfall, en bas diagramme force plot. Ces deux diagrammes affichent la même information, à savoir de quelle manière les variables ont contribuées à la décision de l'algorithme.

4 LIMITES ET AMELIORATIONS

Cette proposition de modèle de scoring en vu de décider de l'attribution ou non d'un prêt à un client repose pour le moment sur un seul jeu de données, très déséquilibré. La technique de rééquilibrage retenue permet de pallier à ce déséquilibre mais conduit à une diminution drastique du nombre d'individus disponibles pour le jeu de données d'entrainement du modèle. Il serait donc intéressant de **collecter d'autres données, notamment de refus de prêt** afin d'étoffer notre jeu de données d'entrainement.

Par ailleurs, la métrique proposée ici pour décider du meilleur modèle mérite d'être discutée avec des expert.e.s métier afin de s'assurer de sa **pertinence et de sa bonne performance** (ie. Est-ce que le nombre de faux positifs et faux négatifs est tolérable ou non). Il en va de même pour les variables construites lors du feature engineering.

Enfin, il semble primordial de discuter avec des personnes du métier quant à **l'interprétation d'attribution** ou non d'un prêt, tant du point de vue du fonctionnement de l'algorithme que de la contribution de chaque variable à la décision pour un client spécifique.