OPENCLASSROOMS

Prêt à dépenser

Analyse et prédiction de répaiment de debit de client

Sriram Vadlamani

October 2023

Contents

1 Introduction

Ce projet est fait pour mon master en Data Science avec Openclassrooms pour le Projet 7. Cette documentation va décrire la méthodologie principale pour entraîner et mettre en production un modèle pour prédire si un client donné peut repayer son débit et la probabilité de repayment.

2 Entraînement du modèle

Dans cette section, on va explorer le jeu de données en bref et voir comment mettre en place une méthodologie pour entraîner un modèle.

2.1 Données

Les données sont dispatchées dans 7 fichiers liés entre eux selon le schéma ci-dessous. La table "application" regroupe les informations personnelles des clients actuels ainsi que les données relatives au crédit qu'ils demandent. Cette table est séparée en 2 jeux de données : l'application "train" regroupant 307 511 clients dont on connaît la décision de "Prêt à Dépenser" sur l'octroi du crédit (variable "Target") et l'application "test" dont on ne connaît pas cette décision. Les autres fichiers contiennent les données historiques de prêt de ces mêmes clients :

- Les tables "bureau" et "bureau_balance" contiennent les informations des crédits passés dans des institutions autres que "Prêt à Dépenser"

La table "Previous_application" reporte les donnés des crédits passés auprès de "Prêt à dépenser"

Lors de la construction d'un modèle prédictif, nous avons souvent de nombreuses fonctionnalités ou variables dans notre ensemble de données qui peuvent être utilisées pour former notre modèle. Cependant, ce n'est pas parce que la fonctionnalité existe dans notre jeu de données qu'elle est pertinente pour notre modèle et que nous devons l'utiliser. Alors, comment savons-nous quelles fonctionnalités utiliser dans notre modèle ? C'est là qu'intervient la sélection des features. La sélection des features est simplement un processus qui réduit le nombre de variables d'entrée, afin de ne conserver que les plus importantes.

2.2 Selection des features

Pour selectionner les features les plus pertinents, j'ai mis en place les étapes suivantes:

1. **Filter de variance**
   Pour enlêver tous les variables binarisés qui contient pas une changement, ou peu de changements (des variables quasi-constants).

2. **Teste de correlations**
   Enlêvement des variables très correlé entre eux.

3. **Teste d'indépendence**
   Enlêvement des variables sont dépendant de la variable 'target'.

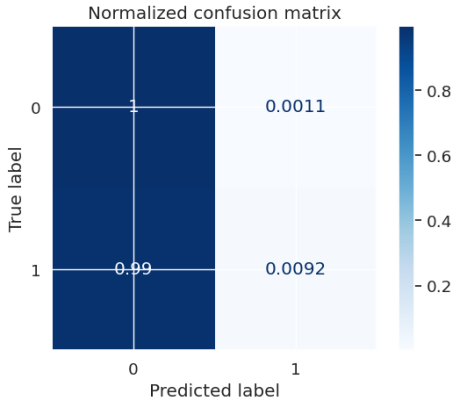
4. **RFE (Recursive Feature elimination)**
   Enlêvement des features récursivement pour éliminer les variables les moins importantes à chaque étape en entraînant un modèle RandomForest.

2.3 Essais avec differents modèles

Après avoir essayé 3 modèles differents:

1. **Regression Logistique**
   Pas très pertinent et, le modèle a planté pendant l'entraînement plusiers fois sur un notebook colab.

2. **XGBoost Classifier**
   Modèle pas très performant avec beaucoup de faux négatifs (dans notre cas, les clients qui sont censé d'avoir un problème de répaiment de débit pas prédi comme le contre).



3. **LightGBM**
   Modèle le plus pertinent entre les 3 choix avec une bonne équilibre entre vrai positifs et faux négatifs.

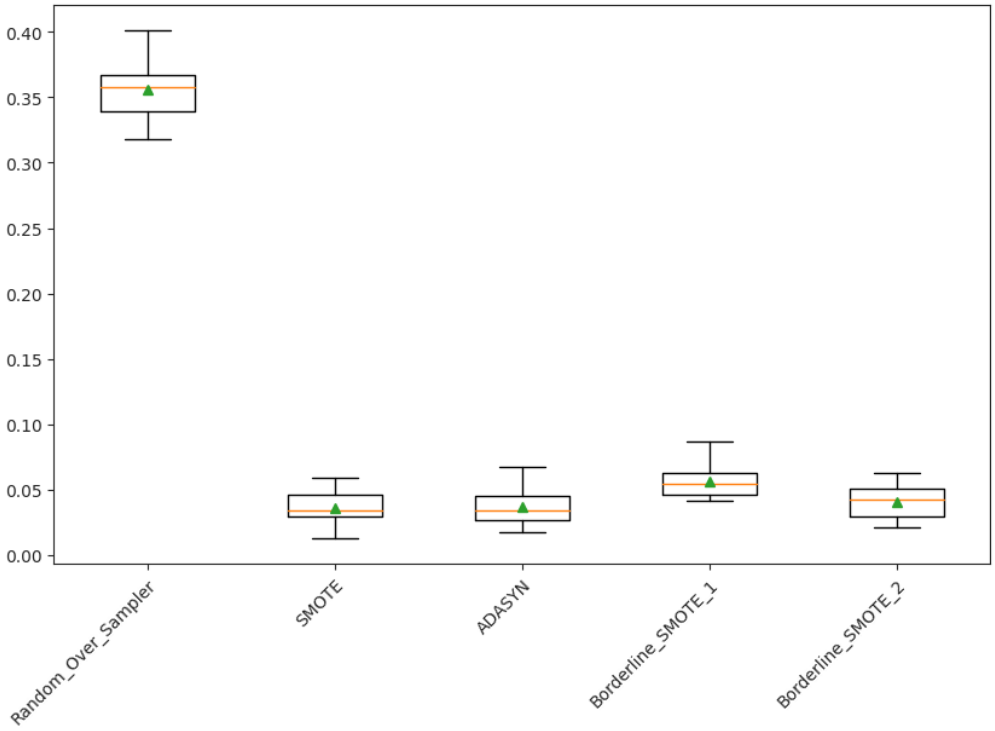
Le synthèse du modèle va se retrouver au dessus dans la section 'Synthèse globale'.

3 Traitement de déséquilibre des classes

Target Name	Value
0	282686
1	24825

À partir du tableau, 0 signifie les clients qui n'avait aucune difficulté de repaiment de débit et 1 signifie le contraire. On peut voir qu'il y a clairement une déséquilibre.

Pour gérer cette déséquilibre des classes, on peut générer des exemples pour la classe en minorité et les utiliser pendant l'entraînement du modèle. Il existe plusiers méthodes pour le faire, avec SMOTE (Synthetic Minotiry Over Sampling) qui est très populaire. J'ai testé les méthodes suivantes avec une entraînement avec le choix de modèle fait ci-dessus, notamment LightGBM Classifier.



On peut voir qu'entre les méthodes SMOTE, le 'Borderline 1' fonctionne le mieux, Qui selectionne, pour une x_i,

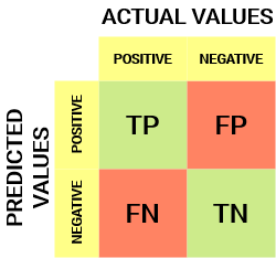
$$x_{new} = x_i + \lambda \cdot (x_{zi} - x_i)$$

Mais, le plus performant entre tous les méthodes est le RandomOversampler, qui fonctionne par dupliquer les examples de minorités existantes et les remplacer, Qui, en effet, nous donne le meilleur score 'F2 (décrit dans la section suivante).'

4 Fonction coût métier et optimization

Dans notre cas d'étude, c'est très importante qu'on prédit précisement les clients qui l'ont une difficulté de repaiment de débit. Dans la terminologie de machine learning, on l'apelle le 'Recall'.

Voici une matrice de confusion quelquonque:



Ici, le taux de True Positives TP par la formule: $\frac{TP}{TP+FN}$ est connu comme le score 'Recall'.

Pour donner plus d'importance pour le recall, on peut utiliser un 'F-beta' score avec 'beta = 2'.

$$F_\beta = \frac{(1+\beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

$$F_2 = \frac{5 \cdot Precision \cdot Recall}{4 \cdot Precision + Recall}$$

En utilisant F2 comme la fonction 'Scorer' pour le modèle, on peut optimiser et trouver la meilleure choix pour Maximiser le Recall.

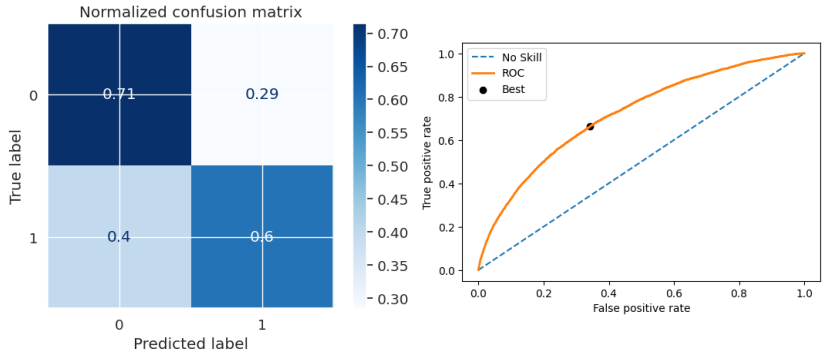
5 Synthèse globale

Voici une table de synthèse globale pour l'entraînement du modèle LightGBM avec un RandomOverSampler sur le jeu de données:

Precision	Recall	F1	F2	Accuracy
0.1556	0.6077	0.2477	0.3843	0.7039

On peut voir que l'accuracy est satisfaisante mais la précision nous manque beaucoup. Ça signifie potentiellement qu'on peut classifier un client ayant aucune difficulté comme le target '1' (la catégorie des clients qui l'ont forcément une difficulté de repaiment). Dans la matrice de confusion au dessus, on peut voir qu'il y a 29% de ce cas.

Avec un score recall de 60% on prédit toujours 40% faux négatifs, notamment les clients sont censé d'avoir les difficultés de repaiment, mais classifiés comme les clients sans difficultés.



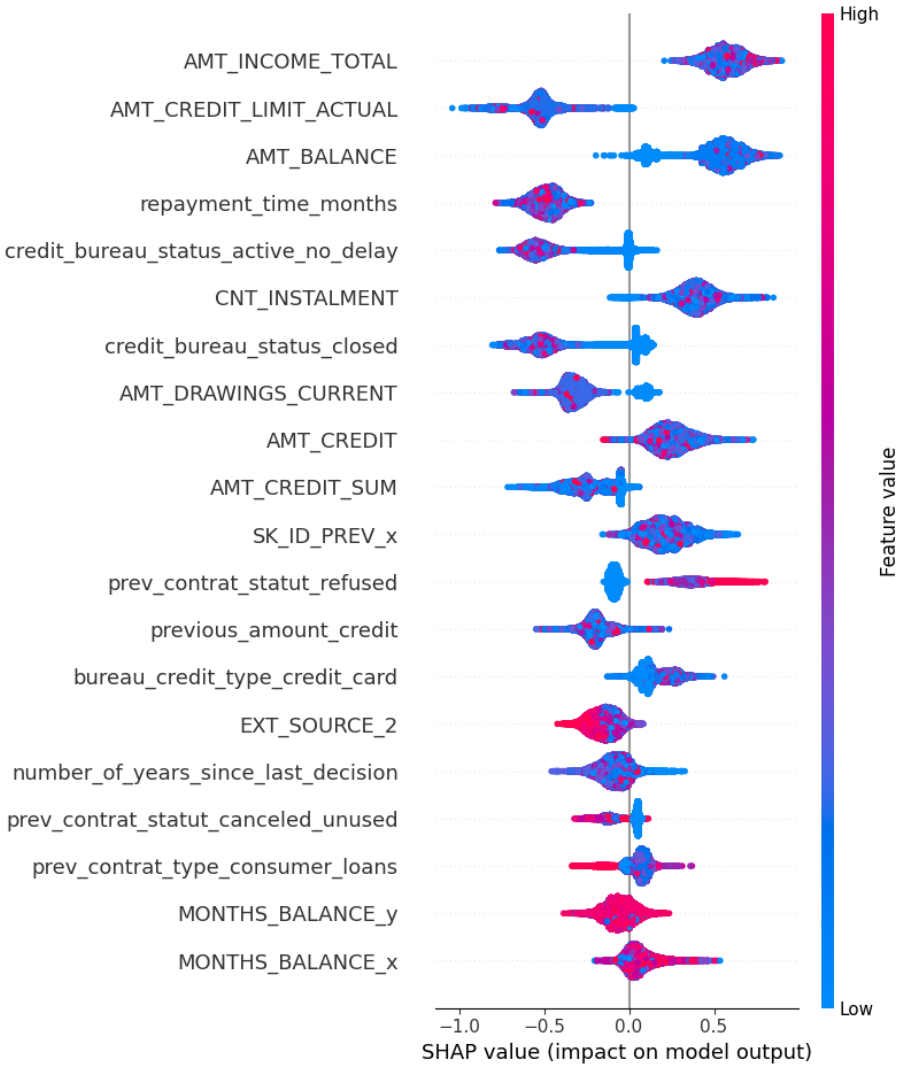
6 Interprétation du modèle

6.1 Interprétation globale

L'interprétation des modèles est importante en machine learning, ne serait-ce que parce que dans de nombreux domaines, il faut expliquer – justifier – la prise de décision induite par le modèle prédictif. La question de l'identification des variables pertinentes reste centrale, ne serait-ce que pour comprendre le mécanisme d'affectation sous-jacent au modèle. L'importance des variables mesure l'impact global de chaque descripteur dans le modèle. Elle peut être estimée en modélisation (sur l'échantillon d'apprentissage) ou en prédiction (sur l'échantillon test). Dans les deux cas, les principales étapes sont les mêmes :

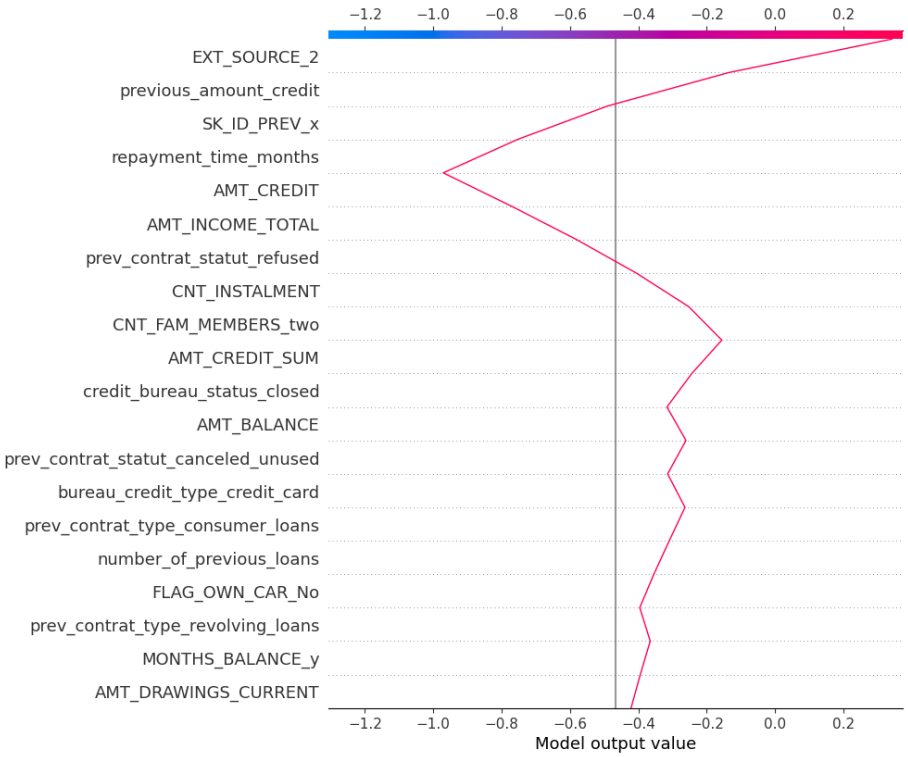
- calculer le taux d'erreur de référence
- calculer ensuite le même indicateur en neutralisant tour à tour chaque variable prédictive
- former le ratio entre les deux valeurs.

Les valeurs de Shapley calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable. Cependant, étant donné que l'ordre dans lequel un modèle voit les variables peut affecter ses prédictions, cela se fait dans tous les ordres possibles, afin que les fonctionnalités soient comparées équitablement. Cette approche est inspirée de la théorie des jeux.



6.2 Interprétation locale

Au contraire, l'intelligibilité locale, consiste à expliquer la prévision f(x) d'un modèle pour un individu x donné. Par exemple, pourquoi la demande de prêt d'un client a-t-elle été approuvée ou rejetée ?



7 Améliorations

Pour améliorer le performance du modèle, on peut potentiellement revoir la partie feature engineering, et de refaire les choix pour les transformations faits sur les données, par exemple: conversion vers les variables catégoriques, détection des outliers multi-dimensionelles, et normalisation des distributions de chaqu'un des variables présents.

Avec peu de ressources disponible, je n'avais pas la possibilité de tester plusiers choix de modèles. Une entraînement avec un modèle deep (comme vu dans le bibliothèque Keras), avec une activation linéaire peut être aussi un choix intéressant comme les réseaux de neurons sont fait pour fonctionner avec les données de pleine volume.

La piste le moins rapide pour améliorer sera de recolter plus de données avec un 'target' = 1 et de re-entraîner le modèle. Cette piste est lente comme c'est un travail extensif de recolter les données de qualité.

8 Data drift

Data drift est une méthodologie mise en place pour détecter les changements dans la distribution des variables. Dans l'image suivante, on voit le rapport généré par 'evidently', pour notre jeu de données. On peut détecter une changement dans les distributions grace à deux méthodes différents:

- **Distance Wasserstein** La distance wasserstein mésure le coût pour transformer une distribution quelconque P à l'autre Q bondé dans une espace métrique. Si cette distance ne dépasse pas un threshold, on peut concluire qu'il n'y a pas de changements.

$$W_p(P,Q) = \left(\sum_{(x,y)\in X} [d(x,y)^p \cdot P(x) \cdot Q(y)]^{1/p} \right)$$

- **Jensen-Shannon distance** Au contraire de la première méthode, on mésure la similarité avec cette distance et pas la distinction. C'est donné par la formule:

$$JSD(P,Q) = \frac{1}{2}\left(D_{KL}(P||M) + D_{KL}(Q||M)\right)$$

où $D_{KL}(P||M)$ est la divergence Kullback-liebler.
et la distribution M est une distribution 'moyenne' calculé par:

$$M(i) = \frac{1}{2}\left(P(i) + Q(i)\right)$$

Par ces formules, on peut facilement détecter fans le futur, s'il y a une changement dans notre jeu de données. Par exemple, avec l'inflation: Le montant de débit peut changer partout et qui peut potentiellement changer les distributions. Dans ces cas, il faut repasser l'entraînement et l'essais des modèles pour raffiner une classifier qui peut prédire si un client donné peut repayer son débit sans problèmes.