PROJET 7

IMPLÉMENTEZ UN MODÈLE DE SCORING

Chafaa BENHACINE | Parcours : Data Scientist | 19/12/2021

NOTE METHODOLOGIQUE

I. Introduction

Prêt à dépenser est une société financière d'offre de crédit à la consommation pour la clientèle ayant peu ou pas d'historique de prêt.

Notre mission est de **développer un modèle de scoring de la probabilité de défaut de paiement du client** pour étayer la décision d'accorder ou non un prêt à un client potentiel en s'appuyant sur des sources de données variées.

Le **développement d'un dashboard interactif** permettra aux chargés de clientèles d'expliquer avec transparence la décision d'octroi ou non du prêt et de mettre à disposition des clients l'exploration de leurs informations personnelles.

1. Prétraitement des données

Le prétraitement consiste à nettoyer le jeu de données, en supprimant les valeurs aberrantes et les variables dont les valeurs manquantes sont en trop grande quantité. Puis il faut ensuite numériser les variables catégorielles de manière à rendre interprétables les variables non numériques. Ce travail s'accompagne d'une étape de « **Feature Engineering** » qui consiste à partir des variables présentes, de trouver des combinaisons de variables plus pertinentes.

1.1. Valeurs aberrantes pour le test set et le train set

Dans un premier temps, les jeux de données d'entrainement et de test ont été nettoyés.

Pour certaines variables de type catégorielles, des valeurs apparaissent uniquement et en faible nombre dans le jeu de données d'entrainement. (« **XNA** » apparait seulement 4 fois pour la variable « **CODE GENDER** »)

Pour les variables numériques, les valeurs aberrantes (DAYS_EMPLOYED > 366) ont été supprimés

Les emprunts correspondant ont donc à chaque fois été supprimés.

1.2. Feature Engineering

a) Automatique Feature Engineering

Cette étape s'est accompagnée de la jointure des autres tables (anciens prêts) aux tables d'entrainement et de test. Les fonctions utilisées sont les suivantes : cumsum, sum, mean.

b) Manuel Feature Engineering

Quelques variables additionnelles qui semblaient importantes ont été construites manuellement comme la proportion du temps travaillé par rapport à l'âge du client, ou le ratio crédit/revenu etc…

c) Données manquantes

Certaines variables ne possèdent pas de valeurs pour certains emprunts. Or, tout modèle de ML nécessite d'avoir comme entrée des valeurs numériques pour chaque variable qui lui est fourni. Il existe diverses méthodes qui permettent de remplacer les valeurs manquantes mais il est nécessaire que celles-ci ne soient pas en trop grand nombre. Ainsi, toutes les variables du jeu d'entrainement et de test, dont les données manquantes étaient supérieures à plus de 10% ont été supprimés. Pour chaque variable, les valeurs manquantes ont été remplacées par la valeur médiane de la variable en question.

d) Encodage des variables catégorielles

i) Variables cycliques

Les variables cycliques comme les jours de la semaine ont été encodés en deux dimensions par un système de coordonnées polaires.

ii) Autres variables

Pour les autres variables catégorielles, deux méthodes ont été utilisés :

- Pour les variables dont le nombre de valeurs différentes est égal à deux, ont été binarisés, i.e. codés par des 0 ou des 1.
- Pour les autres, une nouvelle variable a été créée pour chaque valeur différente prise par la variable initiale. Et pour chaque emprunt, la valeur prise est soit 1 ou 0 suivant la valeur de la variable initiale.

2. Sélection d'un modèle ML et méthode d'entrainement

Il existe différents modèles de ML plus ou moins performants et plus ou moins complexes, donc plus ou moins interprétables. Parmi ceux communément utilisés dans le milieu, nous avons sélectionnés les suivantes : Dummy, Gaussien, Régression Logistique, Forêts aléatoires, Perceptron Multi Couches, XGBoost et LGBM.

Nous les avons tous testés avec leurs paramètres par défaut, pour avoir un premier aperçu de leur comportement sur nos données. Il en est sorti que XGBoost et LGBM avaient de loin les meilleures performances. Nous avons donc sélectionné ces deux modèles dont nous avons cherché à déterminer la meilleure combinaison d'hyper-paramètres suivant la fonction coût déterminé par le problème.

Démarche de modélisation

La première étape a consisté à gonfler artificiellement le jeu de données pour obtenir une plus grande proportion de personnes non solvables. En effet, environ 90% des prêts existants dans

le jeu de données d'entrainement ont été remboursés. Nous avons donc utilisé la librairie SMOTE qui nous a permis de trouver un équilibre à 80/20%.

Comme expliqué précédemment, chaque modèle de Machine Learning présente différents paramètres à fixer. Suivant la combinaison des paramètres choisis pour le modèle, les résultats seront plus ou moins performants.

Pour déterminer quelle combinaison d'hyper paramètres donne les meilleurs résultats sur notre jeu de données, l'algorithme d'optimisation « hyperopt » a été utilisé.

L'algorithme « hyperopt » utilise une approche Bayésienne pour déterminer les meilleurs paramètres d'une fonction. À chaque étape, il essaye de construire un model probabiliste de la fonction et choisi les paramètres les plus promettant pour la prochaine étape. Généralement, le procédé est le suivant :

1. Génère un point aléatoire initial $\boldsymbol{x}^*$
2. Calcul $\boldsymbol{F}(\boldsymbol{x}^*)$
3. Utilise les valeurs des étapes précédentes pour construire le modèle de probabilité conditionnelle $\boldsymbol{P}(\boldsymbol{F}|\boldsymbol{x})$
4. Choisir $\boldsymbol{x_i}$ telle que $\boldsymbol{P}(\boldsymbol{F}|\boldsymbol{x})$ donne la plus grande probabilité d'obtenir un meilleur $\boldsymbol{F}(\boldsymbol{x_i})$
5. Calcul la vraie valeur de $\boldsymbol{F}(\boldsymbol{x_i})$
6. Répéter jusqu'à ce que le critère d'arrêt soit satisfait

Couplé à l'algorithme d'optimisation « hyperopt », un système de validation croisée a été mis en place. L'échantillon original (données d'entrainement) est divisé en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les k-1 autres échantillons constitueront l'ensemble d'apprentissage. On calcule comme dans la première méthode le score de performance, puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des erreurs obtenues sur les k-passes est enfin calculée pour estimer l'erreur de prédiction. Cette technique permet d'éviter ce qu'on appelle le sur-apprentissage.

3. Fonction coût : métrique bancaire

Un modèle est dit meilleur qu'un autre lorsqu'il donne des meilleurs résultats suivant une métrique de performance bien définie. Pour ce projet, une fonction coût bien précise a été

développée dans l'objectif de maximiser les gains obtenus par validation ou non des demandes de prêts bancaires. La fonction coût étant à maximiser pour ce problème.

La procédure est la suivante :

Il s'agit d'un problème de classification binaire, l'étiquette est soit égale à 0 (négatif, solvable) soit égale à 1 (positif, non solvable). Il existe donc 4 combinaisons possibles :

- **TN** : True Negatif, le modèle prédit 0 et la valeur réelle est 0
- **TP** : True Positif, le modèle prédit 1 et la valeur réelle est bien de 1
- **FN** : False Negatif, le modèle prédit 0 alors que la valeur réelle est bien de 0
- **FP** : False Positif, le modèle prédit 1 alors que la valeur réelle est de 0

		Classe réelle	
		0	1
Classe prédite	0	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	1	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

Le modèle peut donc se tromper de deux manières différentes, soit en prédisant positif alors que l'individu est négatif (False Positif) soit en prédisant négatif alors que l'individu est positif. En revanche, la perte d'argent est plus conséquente pour un FN (prêt accordé alors que le client n'est pas solvable) qu'un manque à gagner pour un FP (prêt non accordé alors que le client est solvable). Une fonction coût a donc été créée pour tenir compte de l'importance relative de chaque erreur.

J = TP*TP_Value + TN*TN_value + FP*FP_Value + FN*FN_Value

TP, TN, FP et FN étant respetivement le nombre de True Postif, le nombre de True Negatif, le nombre de Faux Positif et le nombre de False Negatif.

Des coefficients ont été posés arbitrairement :

- TP_value : 0
- FN_value : -10
- TN_value : 1
- FP_value : 0

Ces valeurs de coefficients signifient que les Faux Négatifs engendrent des pertes 10 fois plus importantes que les gains des Vrai Négatifs. Ces poids ont été fixé arbitrairement et il est tout à fait envisageable de les modifier à la convenance de l'optique métier.

Seuil de solvabilité

Le modèle retourne un score entre 0 et 1 et par défaut il attribue la classe 1 lorsque le score est supérieur à 0.5 et 0 sinon. Il a été décidé d'optimiser ce seuil qui permet de définir si un client est solvable ou non. Le seuil optimal déterminé par « hyperopt » est de 0.1, i.e quand le score retourné par le modèle est supérieur au seuil optimal, le client est dit non solvable.

4. Interprétabilité du modèle

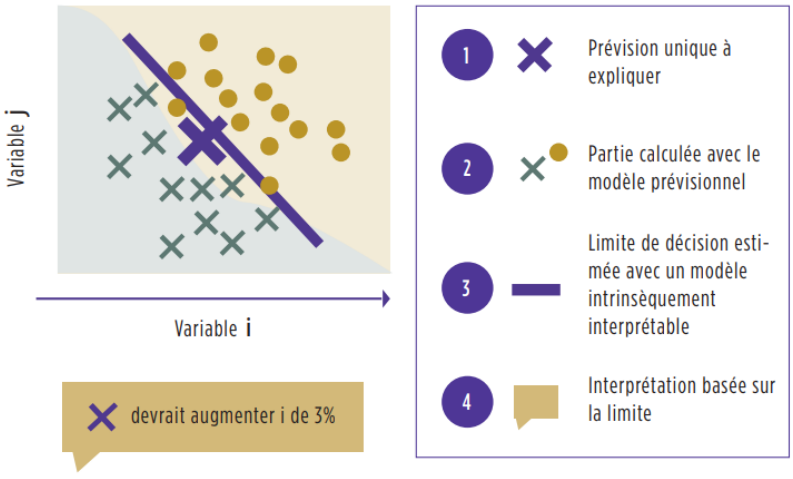
L'interprétabilité d'un modèle est d'autant plus importante que le modèle est complexe. Le modèle sélectionné étant relativement compliqué, la librairie LIME a donc été utilisée.

i) Principe de fonctionnement

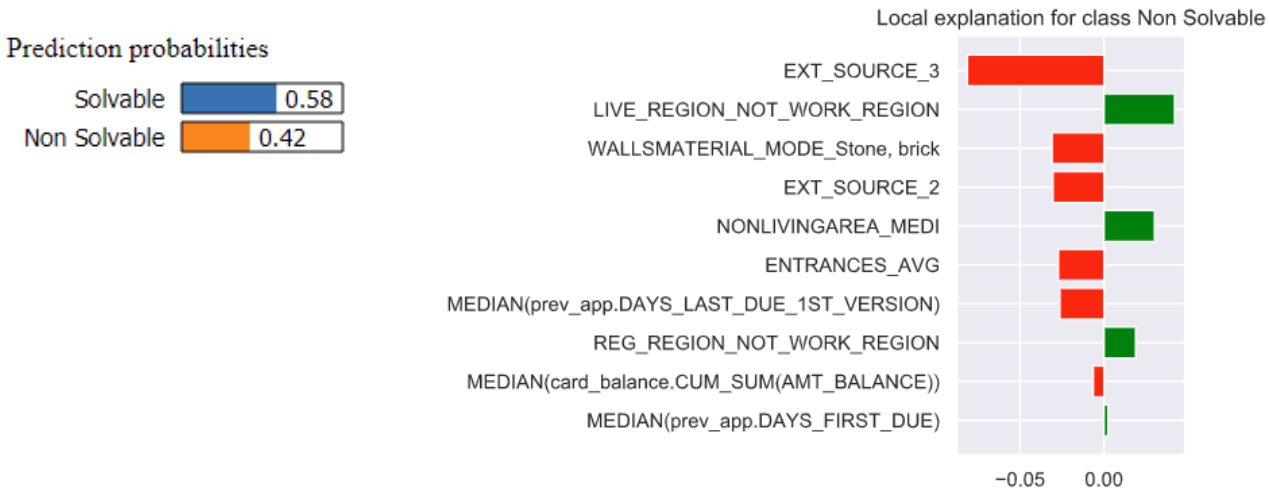
LIME permet de déterminer pour chaque observation les « **features** » qui ont le plus d'impact dans le résultat de la prédiction.

Lime fonctionne en 3 étapes :

- Lime génère aléatoirement des nouveaux individus fictifs proches de l'individu sélectionné
- Lime calcule la prédiction en fonction du modèle
- Lime calcule un modèle linéaire (interprétable) sur ces nouvelles données



ii) Exemple pour un client donné



La non solvabilité du client '100763' est supérieur au seuil de solvabilité de 0.1, donc le client est non solvable. La variable qui joue le plus en sa défaveur est 'EXT_SOURCE_3' qui est un score normalisé provenant d'autres données. A l'inverse la variable qui joue le plus en sa faveur est 'LIVE_REGION_NOT_WORK_REGION', il ne vit pas dans la même région qu'il travaille.

5. Limites et améliorations du modèle

i) Cross validation

Il pourrait être envisageable de diviser le jeu de données en n partitions puis d'entrainer un modèle sur chacune, puis d'utiliser un 'blender' sur les n modèles obtenus. Un 'blender' consiste à agréger les prédictions de chacun des classificateurs et de prédire la classe qui obtient le plus grand nombre de votes. Ce classificateur par vote majoritaire obtient souvent une exactitude plus élevée que le meilleur classificateur de l'ensemble.

ii) Sélection des variables et feature engineering

Il serait judicieux d'effectuer une sélection des variables à partir de « Backward » ou de « Rfecv » de « scikit-learn ».

Quant au « Feature Engineering », il serait judicieux de discuter avec les équipes métiers du domaine. Cela permettrait de déterminer quelles compositions de variables sont les plus pertinentes au vu de leur expérience dans leur choix d'accorder un prêt ou non à un client donné.

iii) Performance du serveur

Un serveur plus conséquent en taille mémoire et en CPU permettrait d'afficher plus de contenu sur le Dashboard. Éventuellement, l'ajout de GPU pourrait permettre de tester le modèle « CatBoost » qui est très prometteur, notamment pour le traitement des variables catégorielles.

iv) Interprétabilité du modèle

La méthode SHAP qui repose entre autres sur la théorie des jeux compense les défauts de la méthode LIME comme son instabilité et le fait qu'un modèle local simple n'est pas une bonne approximation du modèle global.