****

**PROJET 7 - IMPLÉMENTEZ UN MODÈLE DE SCORING**

NOTE MÉTHODOLOGIQUE

*Fatima-zahraeKARIM – Parcours Data Scientist*

**1 Contexte**

**Description du projet**

"Prêt à dépenser" est une société́ financière qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

L’entreprise souhaite mettre en œuvre un outil de “scoring crédit” pour calculer la probabilité qu’un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. Elle souhaite donc développer un algorithme de classification en s’appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

"Prêt à dépenser" décide aussi de mettre en place un Dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d’octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

**2 Prétraitement des données**

Le prétraitement consiste à nettoyer le jeu de données :

* Suppression des données aberrantes
* Suppression des variables avec un taux élevé des valeurs manquantes
* Numériser les variables catégorielles de manière à rendre interprétables les variables non numériques.
* « Feature Engineering » qui consiste à partir des variables présentes, de trouver des combinaisons de variables plus pertinentes.

**2.1 Valeurs aberrantes « Jeu test / Entrainement »**

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

* Pour certaines variables de type catégorielles (non numériques), 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 correspondants ont donc à chaque fois été supprimés.

**2.2 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.

**2.3 Encodage des variables catégorielles**

* **Variables cycliques**

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

* **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.

**2.4 Feature Engineering**

* **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.

* **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…

**3 Méthodologie d’entrainement du modèle**

Il existe différents modèles de ML performants, complexes et interprétables. Parmi ceux communément utilisés dans le milieu, nous avons sélectionnés les suivantes : **Dummy, Gaussien, Régression Logistique, Forêts aléatoires, 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 équilibrer 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 méthode **SMOTE** (Synthetic Minority Oversampling Technique) qui nous a permis de trouver un équilibre à 80/20%. (https://blog.soat.fr/2019/12/techniques-augmentation-dataset-smote/)

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.

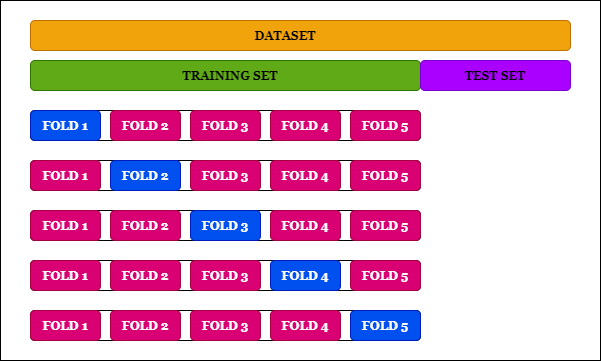
Pour déterminer quelle combinaison d’hyper paramètres donne les meilleurs résultats sur notre jeu de données, l’algorithme d’optimisation «**hyperopt** » a été utilisé.

L’algorithme « **hyperopt**» utilise une approche Bayésienne pour déterminer les meilleurs paramètres d’une fonction. A 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 :

* Génère un point aléatoire initial x\*
* Calcul F(x\*)
* Utilise les valeurs des étapes précédentes pour construire le modèle de probabilité conditionnelle P(F|x)
* Choisir xi telle que P(F|x) donne la plus grande probabilité d’obtenir un meilleur F(xi)
* Calcul la vraie valeur de F(xi)
* Répéter jusqu’à 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 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.

La figure Fig.1 représente le processus de validation croisée implémenté.



(<https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538>)

**4 Fonction coût métier et métrique d’évaluation**

**4.1. Fonction coût : métrique bancaire**

La fonction coût métier spécifique a notre problème est liée à l’estimation du coût subi par l’entreprise et dû aux erreurs de classification faites par notre modèle.

En fait, dans le contexte des prêts pour notre entreprise, le modèle de classification peut effectuer des erreurs de classement du genre :

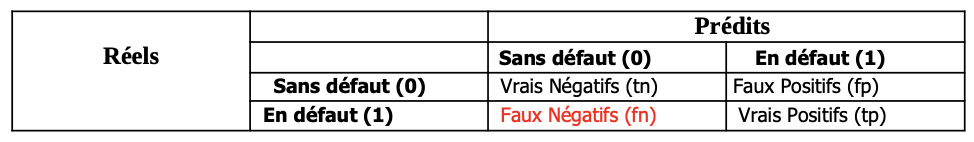
* Classer un bon client (sans ou peu de risque de non remboursement total ou partiel du prêt) comme mauvais client, et donc induire une décision de non-prêt. Cette décision est normalement cause de perte de bénéfice pour l’entreprise.

Ce cas est désigné par **FP (False Positive)** où la prédiction est positive (1), alors qu’en réalité ça doit être négative (0).

* Classer un mauvais client (haut risque de non remboursement total ou partiel du prêt) comme bon client, et donc induire une décision d’autorisation de prêt. Cette décision est surement cause de perte de capital et frais de suivi et d’administration supplémentaires.

Ce cas est désigné par **FN (False Négative)** où la prédiction est négative (0), alors qu’en réalité ça doit être positive (1).

* **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



Le modèle peut donc se tromper de deux manières différentes. 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.

**Fonction = TP\*TP\_Value + TN\*TN\_value + FP\*FP\_Value + FN\*FN\_Value**

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.

* 1. **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.

**5 Interprétabilité du modèle**

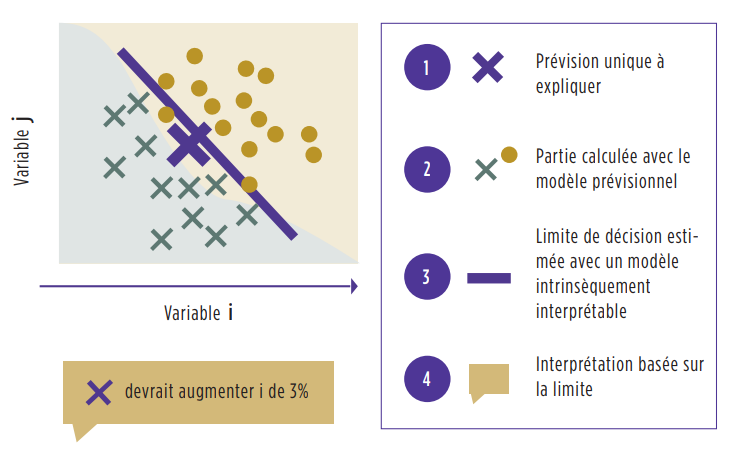
L’Interprétabilité d’un modèle est d’autant plus importante que le modèle est complexe. Le modèle sélectionné étant relativement compliqué, la librairie LIME a donc été utilisée.

**5.1 Principe de fonctionnement**

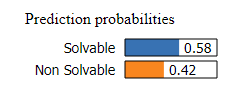
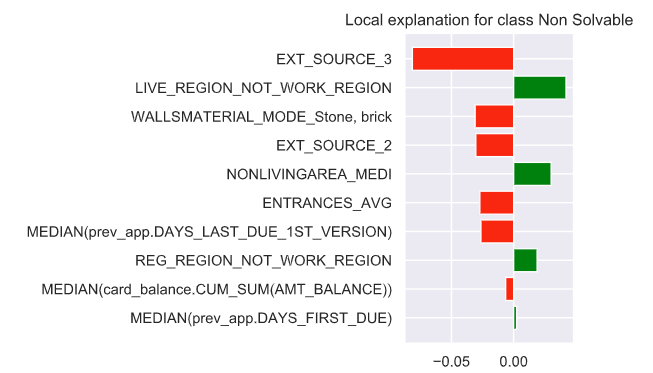
**LIME** permet de déterminer pour chaque observation les « features » qui ont le plus d’impact dans le résultat de la prédiction.

Lime fonctionne en 3 étapes :

* Génère aléatoirement des nouveaux individus fictifs proches de l’individu sélectionné
* Calcule la prédiction en fonction du modèle
* Calcule un modèle linéaire (interprétable) sur ces nouvelles données



**5.2 Exemple pour d’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.

**6 Limites et améliorations du modèle**

**6.1 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.

**6.2 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.

**6.3 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.