IMPLEMENTEZ UN MODELE DE SCORING – Saad ZIZI

Le projet :

Je suis Data Scientist au sein d'une société financière, nommée **"Prêt à dépenser"**, 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 donc de **développer 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.

**Ma mission**

* Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.
* Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.

**La note méthodologique** suivante va décrire :

1. Le traitement des données et la méthodologie d'entraînement du modèle
2. L’algorithme d'optimisation et la métrique d'évaluation
3. L’interprétabilité globale et locale du modèle
4. Les limites et les améliorations possibles

**Traitement des données**

***Composition du Dataset :***

La base de données de « Home Credit » (<https://www.kaggle.com/c/home-credit-default-risk/>) comporte plus de 350 000 clients, avec leurs données personnelles, leur historique de remboursement, leur historique de crédits chez les différentes banques.

Ces données sont réparties dans sept jeux de données, plus un fichier de description des variables.

***Analyses graphiques :***

Dans un premier temps, nous avons réalisé des analyses univariées et multivariées afin d’avoir des premières indications sur le profil et le comportement général des clients de la banque.

***Traitement des valeurs manquantes et des outliers :***

Nous avons traité les valeurs manquantes des données numériques en les remplaçant par la médiane.

Les valeurs manquantes ont également été gérées par la suppression des variables avec un taux de remplissage inférieur à 70 %.

Pas de valeurs aberrantes n’ont été détectées pour notre analyse.

***Encodage :***

Nous sommes également passés par une étape d’agrégation sur les jeux de données comportant plusieurs entrées pour un même client. Nous avons également transformé les variables catégorielles avec le One Hot Encoder.

***Séparation jeu de test/jeu d’entrainement :***

Le jeu de données comportait alors plusieurs variables : 356251 lignes et 248 Features. Nous avons séparé notre jeu de données en jeu de données d’entrainement (80% du dataset) et jeu de données de test (20% du dataset). Nous avons enregistré dans un fichier csv le fichier test et avons continué le traitement.

***Travail pré-modélisation :***

Après plusieurs drafts de Notebook nous nous sommes aperçus que les traitements de modélisation prenaient beaucoup trop de temps, ralentissaient les performances et altéraient les résultats. Nous avons certes obtenu des résultats satisfaisants (Disponibles sur le Notebook) mais pour les étapes suivantes, nous ne pouvions pas prendre le risque de répercuter toutes ces données sur nos tableaux de bords et notre API. En effet, le jeu de données est extrêmement important, au niveau des lignes et des colonnes.

**Méthodologie d'entraînement du modèle**

***Traitement valeurs manquantes restantes :***

Il a été nécessaire d’effectuer plusieurs étapes de pré-traitement. L’étape « imputation » qui représente l’imputation ou non des valeurs manquantes restantes pour notre jeu de données « train ». Nous avons pris le parti d’opter pour la stratégie « médiane ».

***Encodage :***

Nous avons effectué ensuite un encodage qui est la transformation des variables explicatives : les variables catégorielles ont été transformées avec un One Hot Encoder et les variables numériques centrées/réduites avec un Standard Scaler.

***Modèles sélectionnés :***

* 1. Régression Logistique (librairie Scikit-Learn)
  2. RandomForestClassifier (librairie Scikit-Learn)
  3. Light Gradient Boosting Machine via la classe LGBMClassifier de la librairie lightgbm.

***Comparaisons des performances :***

Les performances des modèles ont été effectuées et comparées.

***Métrique utilisée :***

La métrique utilisée pour évaluer les performances de chacun des modèles est l’AUC (Area Under the ROC Curve). Plus l'AUC est élevée, plus le modèle est capable de prédire des bons résultats.

***Déséquilibre constaté :***

Nous avons constaté un fort déséquilibre entre les clients présentant un risque de défaut de paiement et ceux ne représentant pas de risque. En effet, nous avons une classe sous représentée (8 % de clients qui ne remboursent pas encodé par la valeur 1, contre 92 % de clients qui remboursent encodés par la valeur 0). Les performances des algorithmes et les prédictions peuvent être faussées car le modèle attribuera beaucoup plus fréquemment la classe la plus représentée aux clients.

Ce déséquilibre des classes doit être pris en compte dans l’entraînement des modèles. Nous avons donc testé 3 approches pour gérer ce déséquilibre.

***Approches testées pour la gestion du déséquilibre du modèle :***

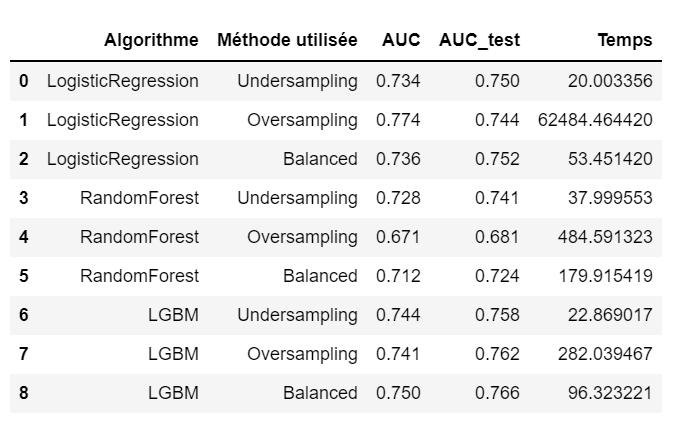
* + 1. Class weight est une méthode directement gérée par les modèles, qui permet de pénaliser les poids associés aux observations de la classe sur-représentées.
    2. L’over-sampling est une méthode qui va dupliquer aléatoirement des données existantes de la classe sous-représentée pour que chaque classe ait le même nombre de données que la classe sur-représentée à l'origine. Nous avons pour cela utilisé la méthode « SMOTE » de la librairie Imbalanced-Learn).
    3. L’under-sampling est une méthode qui va sélectionner une partie des observations sur-représentées pour que chaque classe ait le même nombre de données que la classe sous-représentée à l'origine. Nous avons pour cela utilisé la méthode RandomUnderSampler de la librairie Imbalanced-learn.

Comme dit précédemment, le temps d’entrainement d’un modèle est aussi un facteur de performance. Ayant avant pré-traitement plus de 350 variables explicatives pour plus de 300 000 clients, une étape de sélection de variable afin de diminuer un peu le temps de traitement a été ajouté dans la phase de pré-traitement.

Nous avons donc pris le parti de réduire ces lignes et ces colonnes en effectuant un « sampling » des lignes et des features.

Tous les modèles ont été entrainés avec une validation croisée (séparation du jeu d’entrainement en 5 folds) et leurs hyperparamètres ont été optimisés via GridSearch. Les algorithmes ont été testés avec chacune des méthodes de rééquilibrage des classes présentées plus haut.

***Tableau des résultats des algorithmes testés :***



L’algorithme LGBM associé à la stratégie de rééquilibrage consistant à indiquer "balanced" comme valeur pour " class\_weight" donne les meilleurs résultats : Un bon compromis AUC,AUC test et le temps.

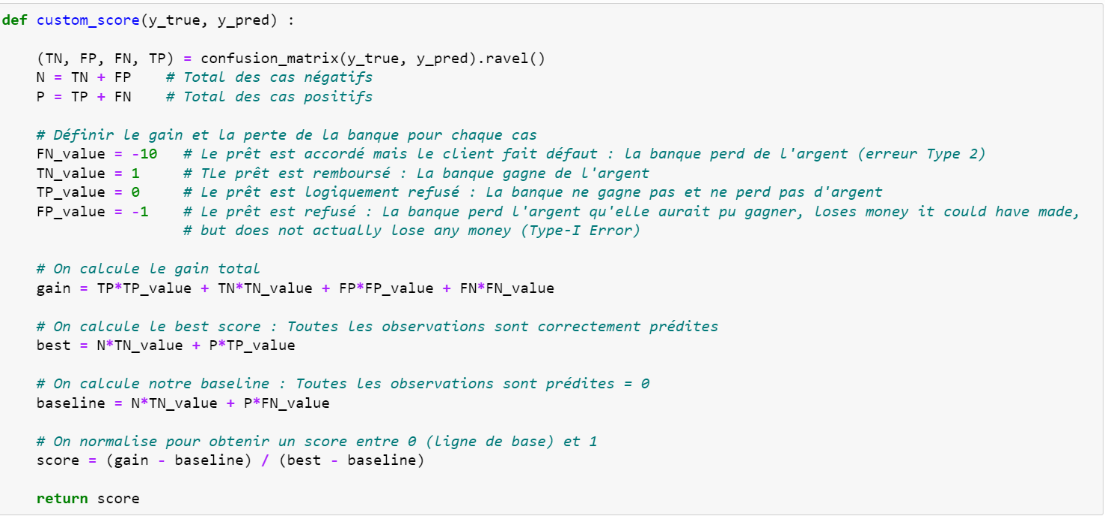
**Fonction coût métier**

***Matrice de confusion***

Une matrice de confusion est l'un des meilleurs moyens d'évaluer les performances d'un modèle de classification. Dans une matrice de confusion, il existe deux types d'erreurs : les faux positifs (FP) et les faux négatifs (FN). Ces termes proviennent de tests d'hypothèses en statistique et sont utilisés de manière interchangeable avec les problèmes de classification.

Pour notre problématique, le résultat attendu le plus important pour un client est la valeur de la probabilité de défaut de payement. Il s’agit de prendre en compte qu'un faux positif (bon client considéré comme mauvais, donc crédit non accordé à tort, donc manque à gagner de la marge pour la banque) n'a pas le même coût qu'un faux négatif (mauvais client à qui on accorde un prêt, donc perte sur le capital non remboursé). Un faux négatif 10 fois plus coûteux qu'un faux positif.

Nous avons défini une fonction métier adaptée au projet qui permet d'attribuer plus de poids à la minimisation des FN. Nous avons défini des coefficients permettant de pondérer les pénalités associées à chaque cas de figure ont été attribués : TP\_value : 0 ; FN\_value : -10 ; TN\_value : 1 ; FP\_value : -1. Nous créons une fonction calculant le score pondéré :



Donc :

* Accorder un crédit à un client ne pouvant pas le rembourser par la suite (FN) est synonyme de perte (10x importante que les FP)
* Accorder un crédit à un client qui le remboursera par la suite (TN) est un gain.
* Ne pas accorder le prêt et que le client ne peut pas rembourser (TP) n'est ni une perte, ni un gain.
* Ne pas accorder le prêt alors que le client pouvait rembourser (FP) est une perte de client donc d'argent.

**Optimisation du modèle :**

Optimiser la valeur de la mesure AUC permet d'améliorer globalement la sensibilité et la spécificité. Cette approche est pertinente si on considère les éléments de la matrice de confusion de même importance. Dans le domaine bancaire, un crédit non remboursé coûte plus cher qu'un dossier de crédit non signé. Il s'agit de trouver le meilleur compromis entre le nombre de crédit qu'on accorde mais qui ne seront in fine pas remboursés (les faux négatifs) et le nombre de crédit qu'on refuse et dont on perd potentiellement le bénéfice sur les intérêts pour les clients solvables (les faux positifs). Ainsi pour le modèle utilisé pour cette étude, nous avons cherché à optimiser le score AUC.

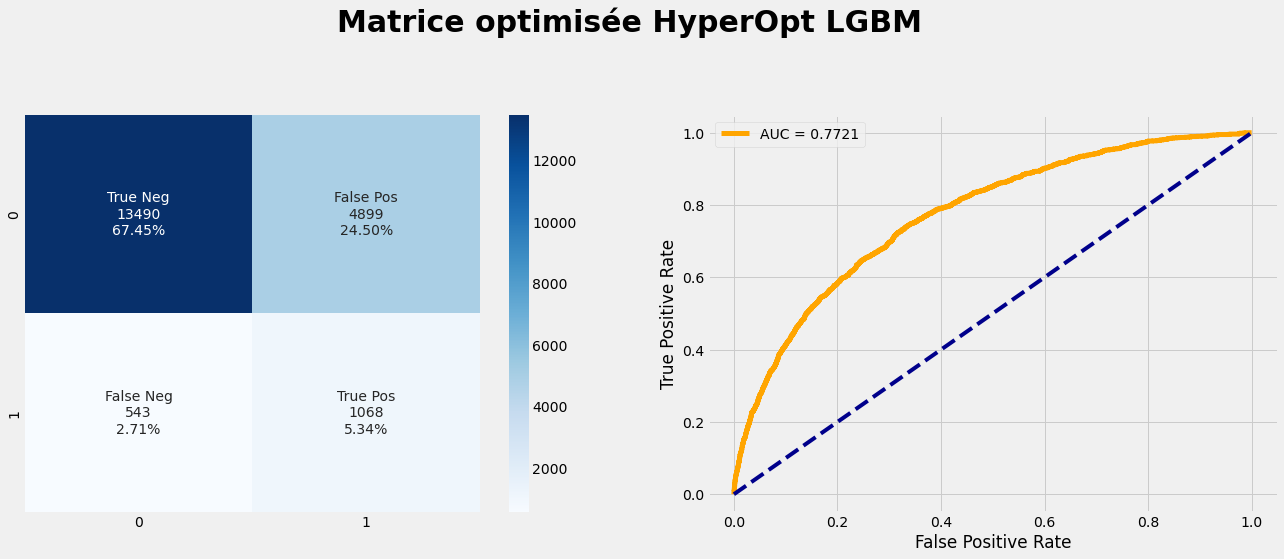
Une image contenant texte

Description générée automatiquement

Nous avons donc effectué une nouvelle recherche des hyperparamètres via HyperOpt se basant sur la fonction métier proposée, de cette façon, ils seront choisis de sorte à minimiser la perte pour l'entreprise. De cette façon, les hyperparamètres optimaux du modèle sont choisis de sorte à minimiser la perte pour l'entreprise. HyperOpt nécessite 4 paramètres pour une implémentation de base à savoir : la fonction objectif (à minimiser loss = 1-score), l'espace de recherche (plages pour les hyperparamètres), l'algorithme d'optimisation et le nombre d'itérations.



Nous obtenons après un nouveau fit sur le LGBM ce résultat pour notre matrice de confusion optimisée avec nos paramètres fixés pour avoir le meilleur résultat possible :



**Interprétabilité du modèle :**

Maintenant que le modèle et l’algorithme ont été définis, il est désormais intéressant de savoir quelles sont les informations qui ont un poids important dans le calcul de la probabilité de solvabilité d'un client. Pour cela nous avons utilisé la librairie « Shap » afin de fournir un calcul de la feature importance globale indépendante de l'algorithme.

Une image contenant table

Description générée automatiquement

Avec cette représentation, on peut dire que les features les plus importantes pour la prédiction d'accord d'un prêt sont le genre, payement rate, les sources extérieures 1, 2 et 3 qui sont des scores normalisés créés à partir de sources de données externes, les revenus annuels et le nombre de jours de travail.

**LIMITES ET AMELIORATIONS POSSIBLES**

Après le feature engineering, les données contenaient pratiquement 800 features au total, ce qui représente un grand nombre. Nous avons dû limiter le nombre de features.

* + *Appliquer la modélisation sur l’intégralité du jeu de données sur une machine avec une RAM extrêmement élevée pourrait être une amélioration*

A cause des soucis d’espace mémoire et de l’execution du Gridsearch qui prend beaucoup trop de temps, nous avons dû travailler avec un échantillon de clients et pas la totalité du dataset

* + *Elargir le nombre d’hyperparamètres pourrait peut-être permettre une amélioration des performances des différents modèles.*

Le modèle ne comporte pas de traitement spécifique des valeurs manquantes. Nous avons pris le parti de les remplacer par la médiane.

* + *Les variables d'entrée sont toutes indispensables au bon fonctionnement des modèles.*

La modélisation a été effectuée sur la base de la métrique AUC. Les coefficients de cette métrique ont été choisis selon le bon sens et la bonne utilisation.

* + *Choisir une métrique plus adaptée au secteur bancaire et après communication avec les équipes métiers ou les conseillers bancaires serait une bonne alternative.*