Projet 7 – Data Scientist

Adrien Gasnault – Fevrier 2023

OpenClassRooms – Implémentez un modèle de scoring

Dossier de versioning : <https://github.com/AdrienGsnlt/P7-Scoring-Model>

## Rappel du contexte de la mission

L’entreprise « Prêt à dépenser » souhaite mettre en œuvre un outil de scoring crédit pour calculer la probabilité qu’un client rembourse son crédit. L’outils de scoring doit s’appuyer sur un algorithme de classification construit sur des sources de données variées. L’objectif de la mission est donc de créer un dashboard interactif afin de permettre :

* Aux chargés de relation client d’expliquer de façon transparente les décisions d’octroi de crédit.
* Aux clients d’explorer facilement leurs informations personnelles

# Méthodologie d’entrainement du modèle

## Preprocessing des données

La base de données fournie est conséquente, elle comprend 10 dataframes combinant plusieurs centaines de variables et plus de 200 000 clients. La structure de la base de données est consultable [ici](https://storage.googleapis.com/kaggle-media/competitions/home-credit/home_credit.png).

Avant d’entrainer notre modèle il est nécessaire d’effectuer un preprocessing des données. Pour ce faire nous avons appliqué [ce kernel kaggle](https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script), en l’adaptant légèrement. Ce kernel nous a permis d’extraire un échantillon de la base de données (n=30 000) issue du training set.

#### Spliting des données

Dans le cas de figure présent : un entrainement de modèle supervisé, il est nécessaire de diviser notre échantillon de données en deux sets :

* Un set d’apprentissage (ou train), comprenant ici 80% des données, sur lequel nous allons entrainer notre modèle.
* Un set de test utilisé pour évaluer notre modèle.

Cette division va nous permettre de simuler une situation où nous appliquerions notre modèle sur de nouvelles données. Par exemple : de nouveaux clients dont on veut prédire la solvabilité via notre modèle. L’application du modèle nous donnera un score de solvabilité (prédit) que nous allons comparer au score connu (vrai). C’est cette comparaison qui nous permettra d’évaluer la qualité du modèle entrainé.

#### Resampling

L’exploration des données a permis d’identifier un fort déséquilibre entre le nombre de clients solvables et non-solvables, ce qui risque de poser différents [problèmes](https://kobia.fr/imbalanced-data-et-machine-learning/) lors de la modélisation. Pour pallier à ce déséquilibre nous avons appliqué la méthode SMOTE (**S**ynthetic **M**inority **O**versampling **TE**chnique) afin d’ajuster la distribution dans notre set d’entrainement (pour plus d’infos voir [ici](https://towardsdatascience.com/the-right-way-of-using-smote-with-cross-validation-92a8d09d00c7)).

## Entrainement du modèle

Pour entrainer notre modèle de scoring, nous allons comparer différentes les méthodes de classifications suivantes : dummyClassifier, RandomForrestClassifier, LightGBMClassifier.

Le dummyClassifier nous permettra d’ancre de comparaison pour les modèles plus complexes. L’entrainement des différents modèles candidats ont suivi la démarche suivante :

#### Tuning des hyperparamètres

Nous avons utilisé **l’algorithme d’optimisation** GridSearchCV, associé à une cross-validation (cv=3) afin de trouver les valeurs les plus nos algorithmes de classification. GridSearchCV permet d’appliquer notre algorithme de classification en trouvant les meilleures valeurs des hyperparamètres pour notre modèle parmi les différentes combinaisons de valeur possibles. La cross-validation sépare lors de l’entrainement notre échantillon en différents feuillets afin de réduire les risques d’overfiting.

Les algorithmes comme RandomForrestClassifier et LightGBM comprennent un nombre d’hyperparamètres très importants. Afin de simplifier cette étape (GridSearch pouvant être relativement long à exécuter), il a fallu au préalable identifier les hyperparamètres les plus susceptibles d’améliorer la qualité du modèle.

#### Comparaison des modèles candidats

Les modèles candidats sont appliqués ensuite aux sets de données test afin de prédire la solvabilité du client (prédite) et de la comparer à la valeur réelle. Dans cette logique plusieurs métriques, qui seront détaillées après, peuvent être appliquées pour retenir le meilleur des modèles candidats.

#### Importances des features

Afin des réduire la complexité du modèle retenu et d’alléger le dashboard nous avons ensuite étudier l’importances des features dans la prédiction de solvabilité bancaire en utilisant [SHAP](https://coderzcolumn.com/tutorials/machine-learning/shap-explain-machine-learning-model-predictions-using-game-theoretic-approach).

SHAP (SHapley Additive exPlanations) est une approche se basant sur la théorie des jeux pour expliquer les modèles de prédictions. L’application de SHAP sur les prédictions de notre modèle pour évaluer l’impact (shap\_value) de l’introduction de chacune des features qui le compose sur la prédiction (en tenant compte de leur ordre et de leur interaction). On peut ensuite visualiser l’ordre d’importances de ces features et ne conserver que les plus importantes pour la suite de la démarche.

## Sélection du modèle final

Après les étapes suivantes nous avons donc une base de données filtré sur laquelle nous allons pouvoir répéter la démarche d’entraînement décrites ci-dessous. In fine nous pouvons sauvegarder notre modèle (avec pickle5) de scoring et notre base de données filtré pour la création du dashboard interactif qui sera créé en utilisant Streamlite et hébergé sur Heroku.

# Métriques

L’objectif de « Prêt à dépenser » est : d’une part, d’éviter les défauts de paiements, accorder un crédit à un client qui ne sera pas en capacité de le rembourser. Et d’autre part, d’éviter ne pas accorder un crédit aux clients qui pourront le rembourser. Les classifications réalisées par notre modèle se distinguent de la façon suivante :

* TP (Vrai positifs) : clients réellement solvables prédits par le modèle comme étant solvables.
* TN (Vrais négatifs) : clients réellement non-solvables prédits par le modèle comme étant non-solvables.
* FP (Faux positifs) : clients réellement non-solvables prédits par le modèle comme étant solvables. Ce sont les clients qui ne pourront pas rembourser le crédit qui leur est accordé.
* FN (Faux négatifs) : clients réellement solvables prédits par le modèle comme étant non-solvables. Ce sont les clients qui se sont vu refuser un crédit qu’ils auraient pu rembourser.

## Métriques d’évaluation

Afin d’évaluer la qualité de notre modèle il existe différentes [métriques d'évaluation](https://blog.nillsf.com/index.php/2020/05/23/confusion-matrix-accuracy-recall-precision-false-positive-rate-and-f-scores-explained/) :

* L’accuracy mesure le nombre de prédictions correctes réalisés par le modèle.
* Le Recall (true positive rate) mesure le nombre le taux de TP prédits sur l’ensemble de l’ensemble de positifs du dataset (TP+FN).
* La Précision mesure la confiance que l’on peut avoir dans la prédiction des positifs, le taux de TP sur l’ensemble des prédictions positives (TP + FP).
* Le False Positive Rate : taux de prédiction positives eronnées (FP) sur l’ensemble des cas négatifs (TN+FP)

## Fonction coût métier

Le modèle doit éviter au maximum les classifications erronés des clients et en particulier les faux positifs qui représente le principal risque de perte financière. Dans cette logique, notre fonction de coût comptabilise le nombre de clients correctement classifiés par le modèle auxquels on retire le nombre de clients mal classifiés en pondérant le poids de ces classifications (un poids de 1 pour les classifications justes, de 5 pour les FN, et de 10 pour les FN).

# Interprétabilité globale et locale du modèle

L’utilisation du modèle de scoring n’est pas d’être analysé sous toutes ces coutures par des experts en data-science, mais d’être utilisé par l’entreprise « Prêt à dépenser » et ses clients. L’interprétation du modèle de scoring et du dashboard doivent donc être rendu lisibles pour des non-expert en data-science. Dans cette objectif, le modèle final retenu se base sur une sélection des 7 features les plus importantes.

Ces 7 features importantes ont été identifiés à un niveau global : via l’utilisation de SHAP, on a pu identifier sur l’ensemble de la base de données les features qui avaient le plus de poids dans les prédictions réalisées par notre modèle.

En complémentarité on peut se placer à un niveau local, pour comprendre pour un client donné : quelles caractéristiques propres au client ont le plus influencé la prédiction de sa solvabilité.

# Limites et améliorations

## Limitations techniques

Nous ne disposions pas d’un ordinateur avec suffisamment de performances pour réaliser l’entrainement du modèle avec l’ensemble des données et un tunning des hyperparamètres plus poussées en termes de nombre de feuillet dans la cross-validation ou le nombre d’hyperparamètres pris en compte par GridSearchCV. Notre modèle s’est donc basé sur un échantillon de la base de données et sur une sélection réduite de features.

## Amélioration

Pour pallier à ses limitations techniques une autre démarche d’optimisation moins couteuse en termes de ressources aurait pu être suivie, par exemple l’utilisation [d’HyperOpt](https://www.kaggle.com/code/ilialar/hyperparameters-tunning-with-hyperopt/notebook), ou de MLFLOW.

Pour la sélection des variables nous aurions également pu utiliser en complément [RFECV](https://www.datatechnotes.com/2022/10/feature-selection-example-with-rfecv-in.html) qui utilise emploi une méthode de sélection récursive des features associé à une cross-validation.