

Note Méthodologique

Projet 7

Anantharajah Anojan



EFREI

Table des matières :

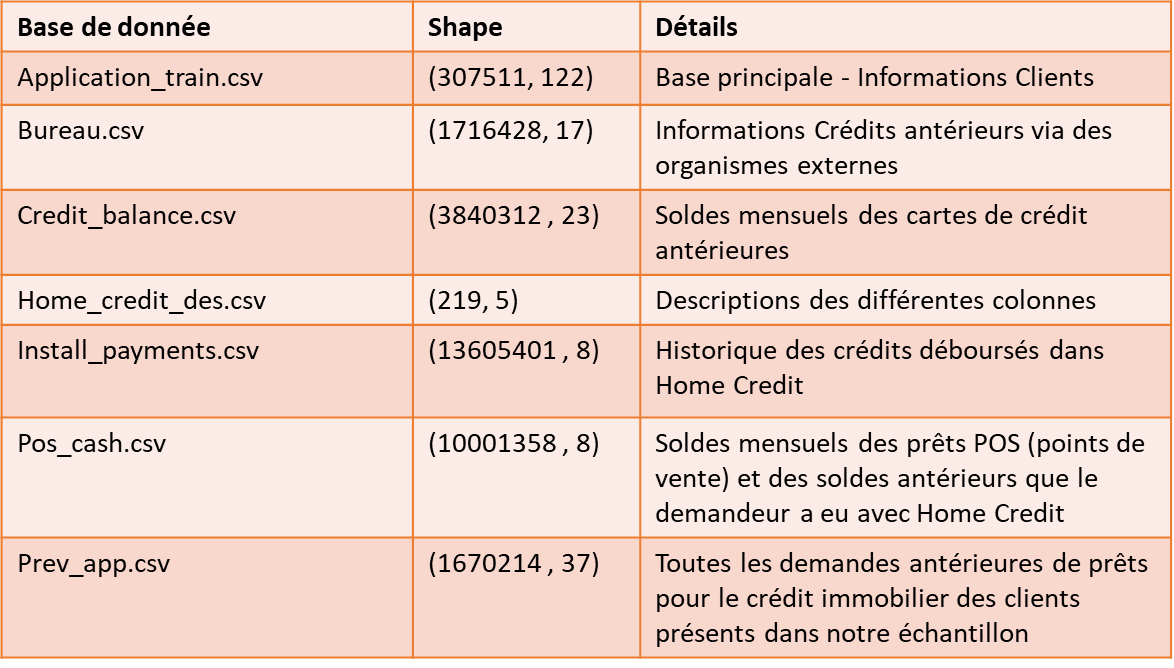
* Rappel du contexte
* Méthodologie d’entraînement du modèle
  + Contrôle sur la variable Target
  + Modélisation avant optimisation
  + Modélisation après optimisation
* Métrique d’évaluation personnalisée
* Interprétabilité du modèle
  + Shap Summary
  + Shap Individual
  + Détermination du seuil de défaut
* Limites et améliorations possibles
* Présentation du Dashboard

Rappel du contexte :

« Prêt à dépenser » est une société française proposant des prêts à la consommation ayant peu ou pas d’historique de prêt.

Pour faciliter l’expérience de la société, ils souhaitent développer un modèle de scoring prédisant la probabilité de défaut de paiement du client. De plus, la société souhaiterait mettre en place une transparence vis-à-vis des clients à propos de la décision d’accorder ou non un prêt. Dans cette optique, il nous a été demandé de mettre en place un Dashboard interactif afin que les décisions d’octroi de prêt soient justifiées.

Pour cela, nous allons d’abord parcourir les jeux de données fournis par l’équipe   
(présents ci-dessous) en réalisant un prétraitement & une préanalyse exploratoire. Puis nous allons proposer une modélisation optimale que nous implémenterons dans le Dashboard interactif accompagnée d’interprétations pertinentes.



Méthodologie d’entraînement du modèle :

**Réalisation d’un split Train / Test (80% Train & 20% Test)**

* Shape Train = (246 008, 93)
* Shape Test = (61 503, 93)

**Contrôle sur la variable Target**

Target à 0 représente 92% du jeu de données alors que les Target à 1 seulement 8%.

On constate donc un déséquilibre au sein de nos classes (Imbalanced data).

Mise en place d’un rééquilibrage des classes via 2 méthodes : Undersampling & OverSampling

* X\_train undersamp : (39 916, 93)
* y\_train undersamp : (39 916,)
* X\_train oversamp : (452 100, 93)
* Y\_train oversamp : (452 100,)

**Standardisation des valeurs**

Standardisation des valeurs réalisée via la fonction StandarScaler() sur l’ensemble des données d’entrainements et de test.

**Modèles avant optimisation**

Utilisation des modèles suivants sur les Datasets UnderSampling & OverSampling:

* Dummy Classifier
* Gaussian Naive Bayes
* LGBM Classifier
* Random Forest Classifier
* XGB Classifier

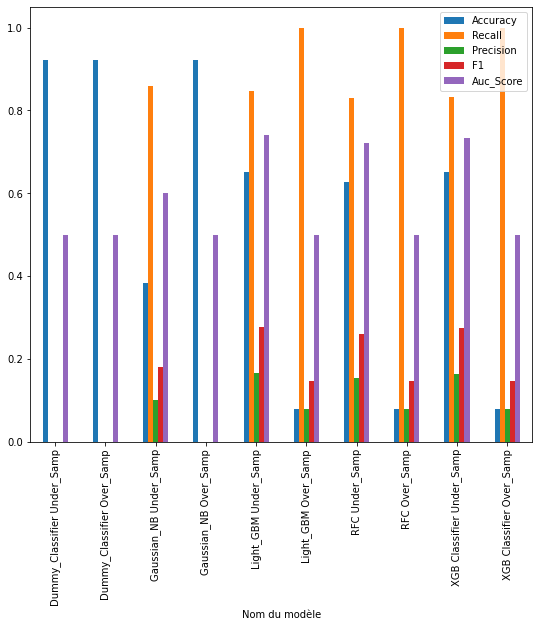
Une image contenant table

Description générée automatiquement

On constate que les modèles encadrés en rouge présentent un Scoring\_weights

(Score personnalisé détaillé par la suite) très élevé. Cela signifie qu’il y a une prédiction de faux négatifs (clients non conformes prédits à conforme) trop importante. Il ne faut donc pas les prendre en compte par la suite.

Pour le choix du modèle optimal nous pouvons nous baser en plus de ce score personnalisé, des autres scores :



Les modèles retenus sont donc :

* Light GBM Under\_Samp
* RFC Under\_Samp
* XGB Under\_Samp

**Modèles après optimisation**

Nous allons optimiser les modèles retenus en appliquant un GridSearchCv en cherchant à minimiser la métrique personnalisée.

Une image contenant texte

Description générée automatiquement

Une image contenant table

Description générée automatiquement

**Le meilleur modèle est donc le XGB Under\_Sampling**

Métrique d’évaluation personnalisée :

L’objectif de cette métrique est de mettre un poids aux prédictions des différents modèles afin de mesurer leur véracité. En domaine bancaire (et dans d’autres) une mauvaise prediction peut entrainer des conséquences importantes (perte business, défaut médical…). C’est donc pour cela que nous allons mettre un poids fort aux prédictions positives alors qu’elles ont été vérifiées comme négatives. L’objectif étant d’avoir un modèle avec le score le plus bas possible.

Voici les étapes pour chaque modèle :

* Récupération des labels prédits (y\_pred) et des labels test (y\_test)
* Récupération de la matrice de confusion
* Scoring suivant le schéma suivant :

Une image contenant table

Description générée automatiquement

Faux positif : Perte de clients => Clients prédits négatifs alors qu’ils sont positifs

Faux négatif : Prêt accordé à des clients pas conformes

Scoring\_weights = (VN \* 1 + FP\*3 + FN\*10 + VP\*1) / 15

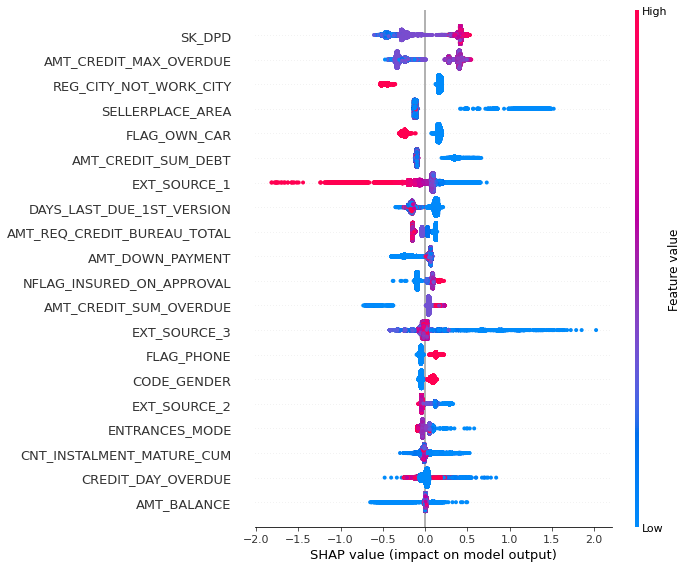
*Les valeurs poids ont été stockées dans des variables (a, b, c, d) de sorte à modifier les poids rapidement.*

Interprétabilité du modèle :

Pour interpreter le modèle, nous allons utiliser la fonction Shap. Pour cela nous allons passer en entrée le modèle final ainsi qu’un jeu d’entrainement (ici le test) afin de récupérer les Shap values.

A partir de cela nous allons pouvoir faire 2 interpretations importantes :

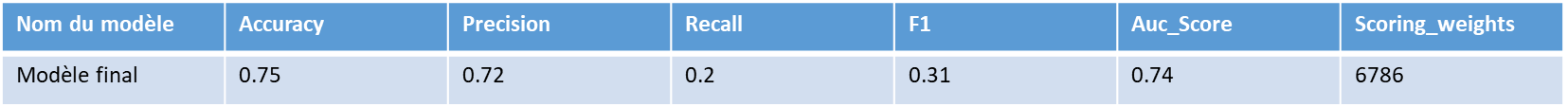
**Shap Summary**



Ce graphique nous indique les features qui impactent le plus le modèle en fonction de leur valeurs. En bleu nous avons les valeurs faibles et en rouge les valeurs fortes. Lorsque le modèle est impacté positivement par un feature les valeurs sont vers la droite et inversement lorsque le modèle est impacté négativement.

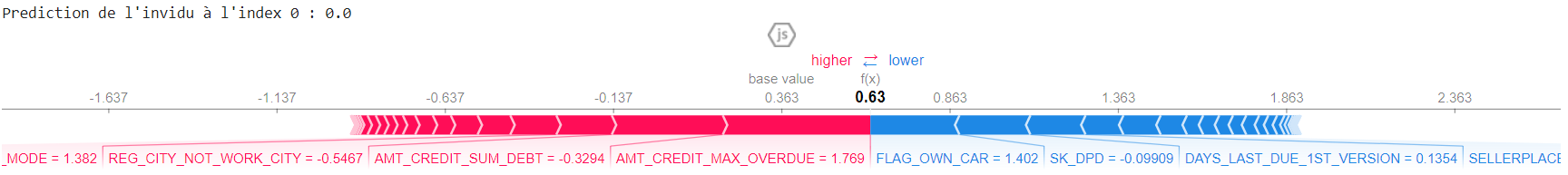
Ce graphique est très utile car il nous permet de faire un Feature selection supplémentaire. Nous allons réduire l’ensemble du jeu de données à ces colonnes et appliquer le modèle final.

Nous pouvons voir que le modèle s’améliore encore :



C’est ce modèle que nous allons utiliser dans notre Dashboard car les valeurs d’entrées ont été réduites tout en améliorant le modèle.

**Shap Individual**



Ce graphique nous permet d’avoir les mêmes informations que le Shap Summary mais cette fois pour un individu en particulier.

Les caractéristiques rouges conduisent notre prédiction à être 0 : client pas en défaut  
Les 3 caractéristiques qui contribuent le plus sont les suivantes : AMT\_CREDIT\_MAX\_OVERDUE / AMT\_CREDIT\_SUM\_DEBT / REG\_CITY

Les caractéristiques bleues indiquent les caractéristiques réduisant la probabilité que le client soit à 0.

**Determination du seuil à partir duquel un client est considéré comme défaut**

Pour déterminer si un client est en défaut de paiement ou non, il est préférable d’utiliser la probabilité de prédiction plutôt que la prédiction. Cela nous permet de connaître le % de précision et ainsi permettre à la Banque de prendre une décision. Pour cela, il faut déterminer un seuil à partir du quel il est considéré comme défaut.

Les étapes :

* Récupération des listes de prédictions à 1 (défaut) pour un seuil allant de 0 à 1
* Custom\_metric entre chacune des listes et le y\_test initial pour récupérer le scoring\_weights
* Scoring\_weight le plus faible lorsque le seuil est à 0.7

Une image contenant table

Description générée automatiquement

Si le client est prédit à 1 avec une probabilité supérieure ou égale à 70%, il sera considéré comme un client à défaut.

Limites et améliorations possibles :

Comme le montre les interprétations Shap, de nombreux features impactent positivement ou négativement le modèle. La réduction du Dataset à ces features seulement a permis d’améliorer les résultats de sortie. Cela signifie que certains features perturbaient le modèle mais il faudrait comprendre pourquoi.

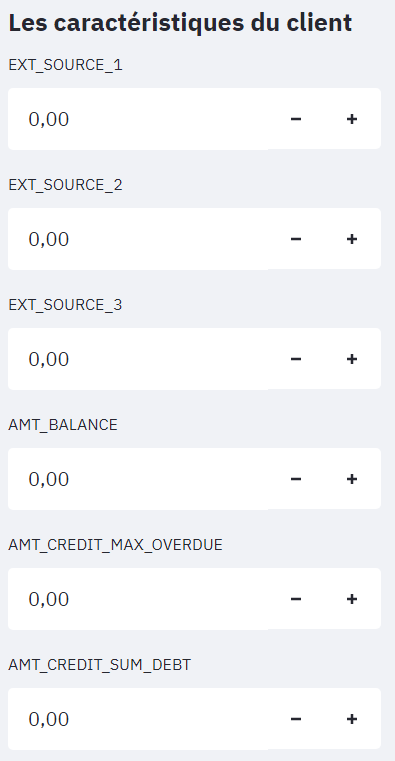
Malgré le scoring weights bas par rapport aux autres modèles, il reste tout de même présent. Cela veut dire que le taux de faux négatifs est toujours présent et peut impacter négativement la banque.

Les paramètres utilisés dans le GridSearchCv peuvent être grandement améliorés.

Les technologies utilisées ne permettent pas d’avoir le modèle le plus optimal possible

Pour améliorer le modèle nous pourrions donc utiliser des plages de valeurs plus grandes pour le paramétrage du GridSearchCv. Cela prendrait plus de temps et demanderait plus de ressources mais nous pourrions avoir un modèle beaucoup plus performant.

Présentation du Dashboard :



Une image contenant table

Description générée automatiquement

