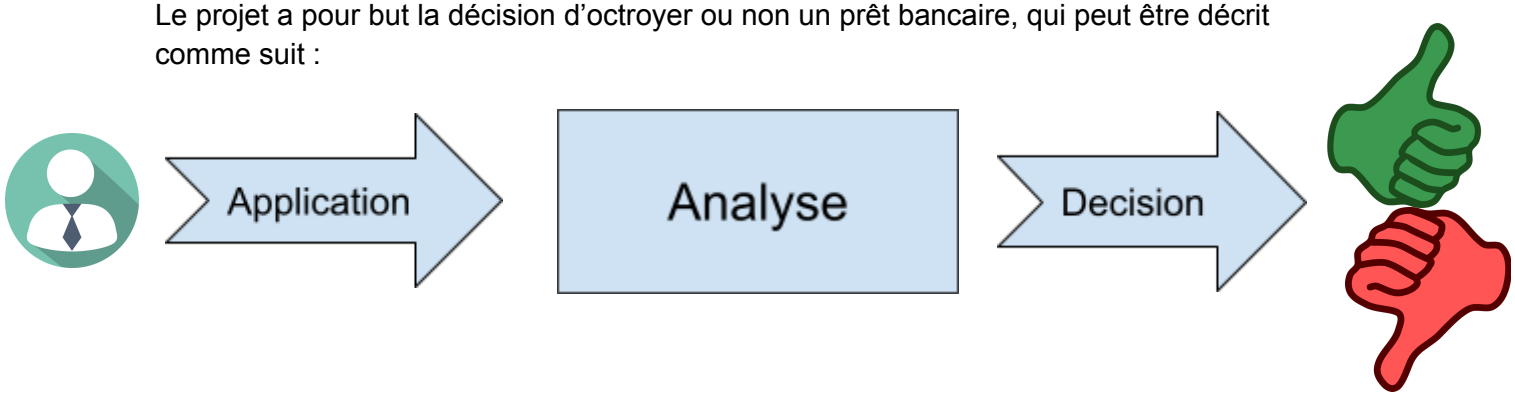
Credit Scoring with Machine Learning

Approche de modélisation

Table des matières

Introduction

Le projet a pour but la décision d'octroyer ou non un prêt bancaire, qui peut être décrit comme suit :



Un client soumet une demande de prêt à l'organisme : la demande est analysée et donne lieu au refus ou à l'accord de prêt.

L'objectif du projet est de construire un modèle permettant de prédire, à partir des données collectées, une future difficulté de remboursement et d'intégrer ce modèle dans le processus d'octroi de prêt de l'organisme par la fourniture d'un tableau de bord interactif.

Les données

La cible est l'information de classification binaire avec Class 1 = risque de défaut de paiement. Les taux de défauts (failure rates) concernent environ **8% des cas**.

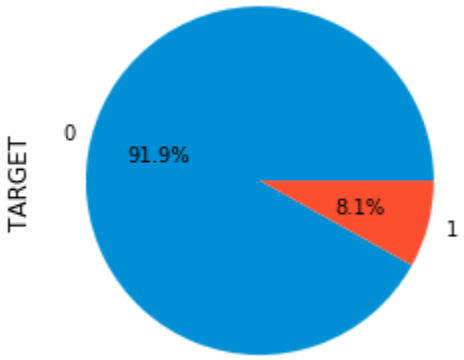


Fig 1 : Pie Chart Target

Nous disposons de 7 jeux de données sources contenant les données disponibles des clients.

A cette étape du projet, il est demandé de réemployer un kernel Kaggle.

Nous avons donc dans cette étape :

- Traité par imputation de la médiane les valeurs manquantes
- Effectué Label encoding pour les variables à 2 catégories et un One Hot Encoding pour les variables à plus de deux catégories.
- Remplacé les outliers par des valeurs nulles. Les valeurs sont ensuite imputées par la médiane dans le Preprocessing. Nous avons aussi ajouté un "flag feature" pour identifier les lignes qui contiennent les outliers.
- Création de 6 variables:

    - 2 variable *"Weighted Features"* : visant à améliorer la corrélation des variables EXT SOURCES avec la target ces deux variables sont créent tel que :

      ```
      app_train['EXT_SOURCES_PROD'] = app_train.EXT_SOURCE_1 *
      app_train.EXT_SOURCE_2 * app_train.EXT_SOURCE_3

      app_train['EXT_SOURCES_WEIGHTED'] = app_train.EXT_SOURCE_1 * 2 +
      app_train.EXT_SOURCE_2 * 1 + app_train.EXT_SOURCE_3 * 3
      ```

    - *6 variable "Domain Features"* : qui sont des variables s'appliquant plus au domaine de la banque :
        - "CREDIT_INCOME_PERCENT" : le pourcentage du montant du crédit par rapport au revenu du client
        - "ANNUITY_INCOME_PERCENT" : le pourcentage de l'annuité du prêt par rapport au revenu du client
        - "CREDIT_TERM" : la durée du paiement en mois
        - "DAYS_EMPLOYED_PERCENT" : le pourcentage des jours d'emploi par rapport à l'âge du client

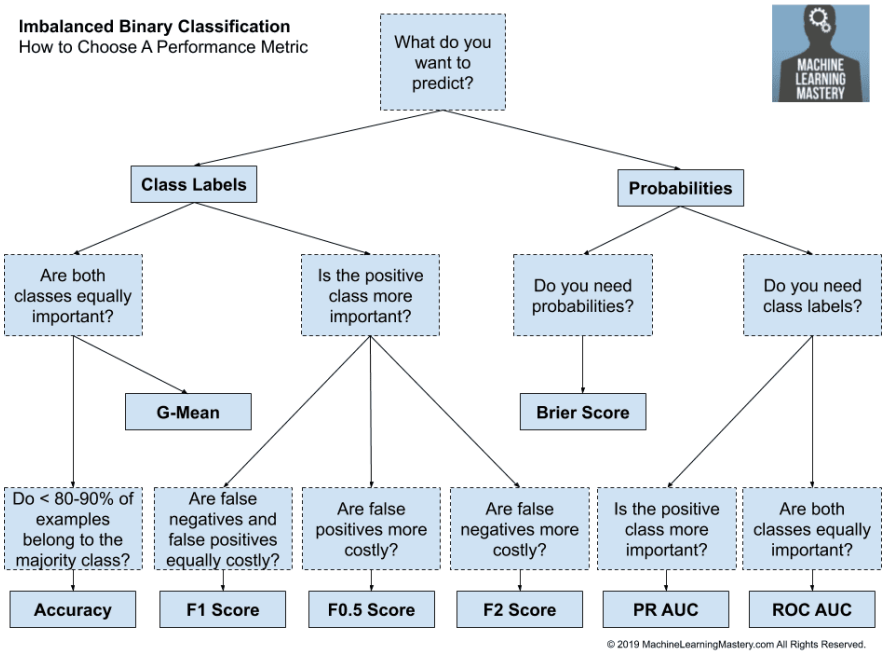
Classificateurs

Nous avons testé 15 algorithmes pour effectuer notre modélisation grâce à pycaret ce qui nous a permis d'obtenir les résultats ci dessous:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.9185	0.6951	0.0042	0.5326	0.0083	0.0070	0.0413	49.7120
lightgbm	Light Gradient Boosting Machine	0.9185	0.7372	0.0123	0.5119	0.0240	0.0202	0.0693	8.3250
rf	Random Forest Classifier	0.9183	0.6943	0.0033	0.4260	0.0066	0.0054	0.0313	61.0790
gbc	Gradient Boosting Classifier	0.9183	0.7050	0.0064	0.4695	0.0127	0.0105	0.0471	232.7220
catboost	CatBoost Classifier	0.9181	0.7378	0.0249	0.4630	0.0472	0.0391	0.0922	117.7640
xgboost	Extreme Gradient Boosting	0.9176	0.7269	0.0304	0.4297	0.0567	0.0465	0.0968	121.2650
ada	Ada Boost Classifier	0.9115	0.6547	0.0312	0.2103	0.0540	0.0340	0.0519	49.6870
dt	Decision Tree Classifier	0.8388	0.5323	0.1659	0.1270	0.1438	0.0566	0.0573	11.6610
knn	K Neighbors Classifier	0.6827	0.5435	0.3508	0.0977	0.1528	0.0289	0.0379	27.0890
ridge	Ridge Classifier	0.6793	0.0000	0.6585	0.1550	0.2510	0.1370	0.1954	2.8370
lda	Linear Discriminant Analysis	0.6793	0.7304	0.6583	0.1550	0.2509	0.1369	0.1953	6.0480
lr	Logistic Regression	0.5910	0.6071	0.5655	0.1099	0.1841	0.0550	0.0881	37.2760
svm	SVM - Linear Kernel	0.5508	0.0000	0.5004	0.0981	0.1292	0.0208	0.0362	12.9220
nb	Naive Bayes	0.2103	0.5935	0.8930	0.0854	0.1558	0.0081	0.0330	2.7240
qda	Quadratic Discriminant Analysis	0.1403	0.5298	0.9590	0.0837	0.1540	0.0046	0.0295	5.0800

Fig 2 : Panel Pycaret

Dans le cas d'un jeu de données non équitables, l'article "Tour of Evaluation Metrics for Imbalanced Classification (Annexe 2)" permet de mieux comprendre les metriques à favoriser selon le jeu de données.



Ainsi les metriques F2 Score sont particulièrement importantes mais elle peut être complétée avec l'accuracy et AUC

Nous avons par la suite décidé de nous éloigner de cette approche, car le nombre de faux positifs, malgrés divers ajustements et ré-entrainements, restent très importants et que la seule approche que nous avons pu mettre en place pour pallier à ce problème a été d'utiiser une métrique "métier", métrique que nous n'avons pas réussi à utiliser via pycaret.

Nous avons donc effectué un entraînement "maison" sur un modèle ensembliste, un modèle de boosting et un modèle linéaire.
Soit un random forest, Light GBM et une régression logistique, que nous avons tous les trois entraînés en utilisant cette fameuse métrique.

Métrique d'évaluation "métier"

Les classes cibles du jeu de données initial sont très déséquilibrées (plus de 90% des crédits sont remboursés sans défaut). Cela rend la métrique *accuracy* peu pertinente.

De plus, on peut supposer que l'impact pour l'entreprise d'un faux négatif et ceux d'un faux positif ne sont pas les mêmes, ce qui rend une *AUC* peu pertinente.

- FN, erreur de 1ère espèce : client à qui l'on refuse un prêt, mais qui l'aurait honoré
- FP, erreur de 2ème espèce : client à qui l'on accorde un prêt, mais qui fait défaut

- $gain = TP \cdot TP\_value + TN \cdot TN\_value + FP \cdot FP\_value + FN \cdot FN\_value$
- $best = (TN + FP) \cdot TN\_value + (TP + FN) \cdot TP\_value$
- $baseline = (TN + FP) \cdot TN\_value + (TP + FN) \cdot FN\_value$

$$=> score = \frac{gain - baseline}{best - baseline} \in [0; 1]$$

$$=> model\_score = \max_{threshold \in [0;1]} [score] \in [0; 1]$$

Cette métrique se base sur 4 paramètres `TP_value`, `TN_value`, `FN_value` et `FP_value` qui sont respectivement les coûts associés aux vrais positifs (TP), vrais négatifs (TN), faux négatifs (FN) et faux positifs (FP). Voici les valeurs de ces paramètres que nous avons utilisés pour la construction du modèle. Ces valeurs peuvent facilement être modifiées pour tenir compte de la « réalité métier ».

```
FN_value = -5
TN_value = 1
TP_value = 0
FP_value = 0
```

Méthodologie d'entraînement

Pour l'entraînement et la validation de nos modèles, nous procédons à une nouvelle séparation en données d'apprentissage et de test, selon une méthode de split stratifié avec une cross-validation sur 5 Folds.

Nous avons testé 4 approches pour chacun de ces algorithmes afin de pallier au problème de déséquilibre.

- Baseline:
- Balanced class_weight :
  Le mode Balanced utilise les valeurs de y pour ajuster automatiquement les poids inversement proportionnels aux fréquences de classe dans les données d'entrée.
- Smote : crée artificiellement plus d'observations dans la classe minoritaire pour rééquilibrer la distribution de la valeur cible
- Random Undersampling : Sélectionne moins de données de la classe majoritaire

Nous avons ensuite systématiquement fait une recherche de threshold afin de sélectionner le seuil ayant le meilleur impact sur la performance du modèle

Smote n'a pas été implémenté avec LightGBM, car notre dataset de training contient des données manquantes.

Résultat

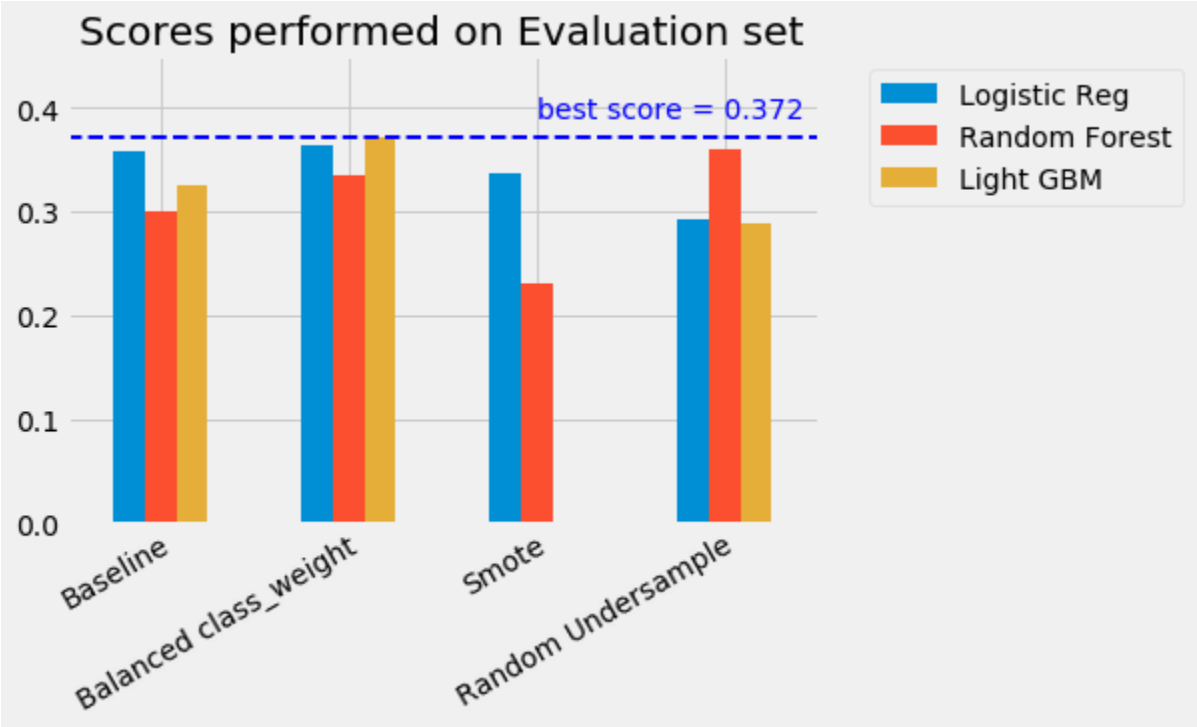


Fig 3 : Score de nos différentes approches

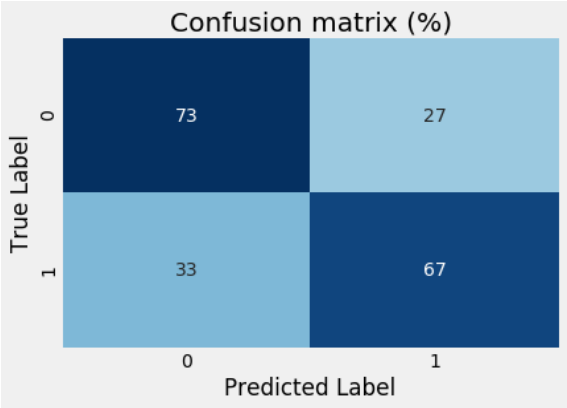


Fig 4 : Matrice de confusion

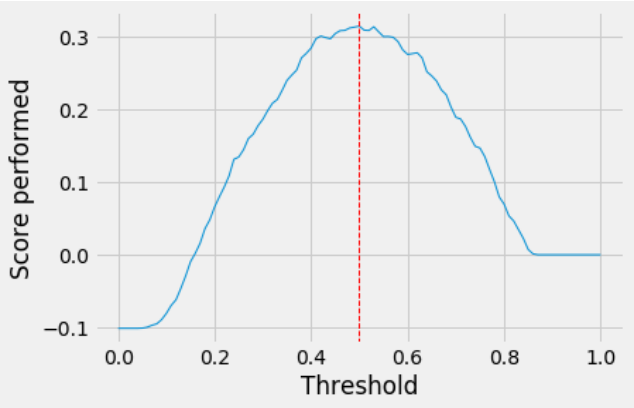


Fig 5 : Score en fonction du seuil

Après l'entraînement des différents modèles, nous pouvons conclure que le meilleur modèle est LGBM en utilisant la méthode Balanced class_weight pour un seuil de 0,5.

Interprétabilité du modèle

Malgré la qualité et la quantité d'informations sources disponibles et tout les efforts que nous avons déployés pour obtenir un modèle de Machine Learning nous obtenons un score médiocre en regard de l'objectif.

Par ailleurs, nous avons observé que l'optimisation d'un objectif métier revient à « sévériser » les décisions de refus.
Faute d'élément complémentaire pour en juger, nous pouvons proposer les actions suivantes :

- Mener une analyse locale, pour sécuriser les accords de prêts.
- Ajuster le seuil de prédiction de façon à s'approcher du taux de refus observé,
- Challenger les règles d'octroi en vigueur par rapport aux modalités de prédiction du modèle.

Ces propositions d'exploitation conduisent naturellement à approfondir la notion d'interprétabilité du modèle.

Feature importance

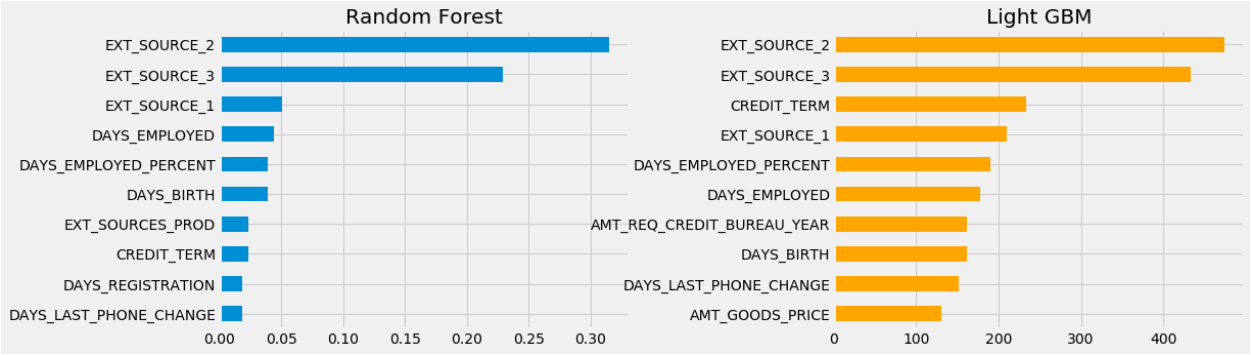


Fig 6 : Feature importance

Dans le cas de modèles d'arbres de décisions, l'importance d'une feature repose sur la Gini Importance ou MDI. Ici, les scores externes (EXT_SOURCE) sont les leviers principaux de notre modèle dans la prédiction de défauts.
On remarque que l'importance de ces variables est plus "équilibré" dans le cas de LightGBM que du random forest, ce qui le rendra plus interprétable.
Le Feature importance ne permet pas d'analyser l'influence des features sur la prédiction.
Cela peut conduire à réduire la confiance des utilisateurs dans le modèle et ses prédictions.

SHAP

La valeur de Shap proposée par Lundberg et al. est la valeur SHapley Additive exPlanation.
L'idée proposée par ces auteurs est de calculer la valeur de Shapley pour toutes les variables à chaque exemple du dataset.
Grâce à la valeur de Shap, on peut déterminer l'effet des différentes variables d'une prédiction pour un modèle qui explique l'écart de cette prédiction par rapport à la valeur de base. La figure suivante décrit les outputs d'une fonction par la somme de ces effets.

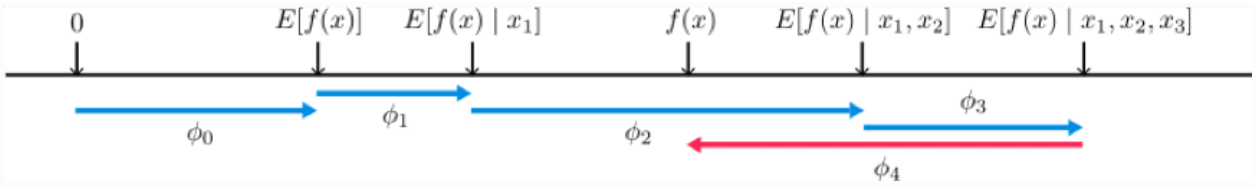


Fig 7: Valeur de Shap. Prédiction f(x) expliquée par la somme des effets de chaque variable

Aussi, en moyennant les valeurs absolues des valeurs de Shap pour chaque variable, nous pouvons remonter à l'importance globale des variables.

Dans notre cas nous avons la figure suivante qui représente l'importance globale des variables calculées par les valeurs de Shap. Grâce au fait que les valeurs sont calculées pour chaque

exemple du dataset, il est possible de représenter chaque exemple par un point (figure de droite) et ainsi avoir une information supplémentaire sur l'impact de la variable en fonction de sa valeur.

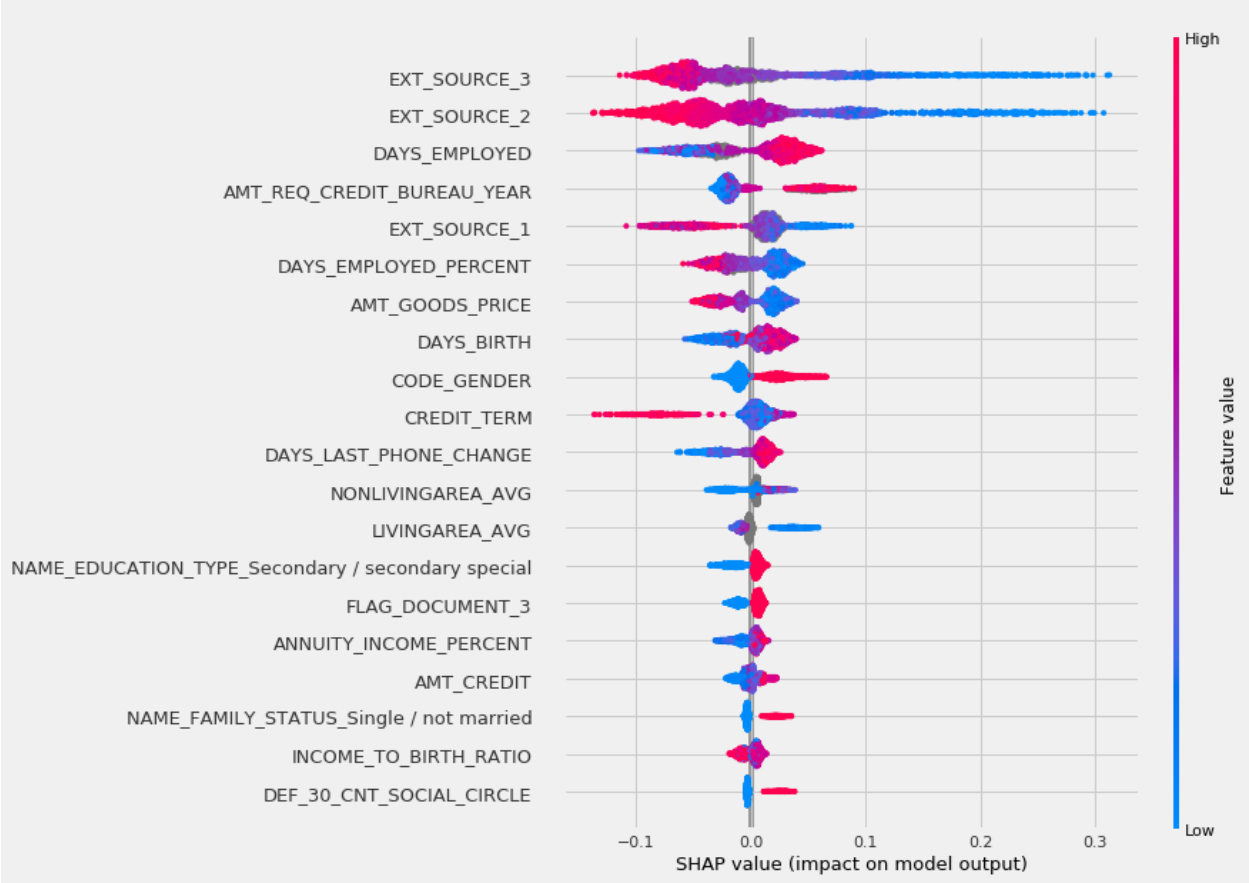


Fig 8: Distribution de Shap global.

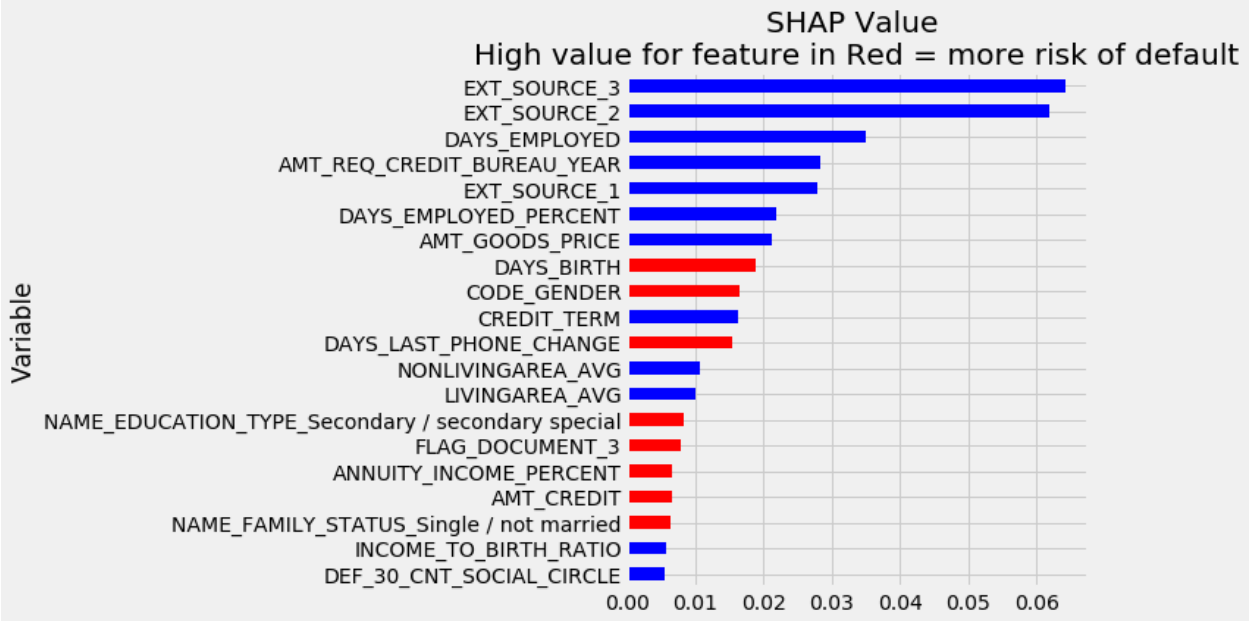


Fig 9: Feature importance Shap.

Shap permet également d'avoir un interprétation "locale".

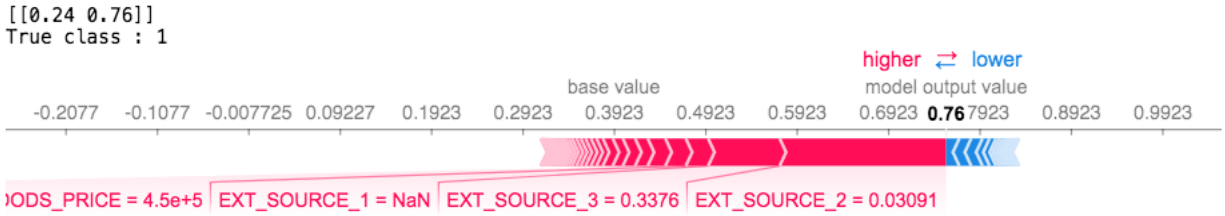


Fig 10: Explication d'impact des variables.

Pour ce client, on a en rouge, les variables qui ont un impact positif (contribuent à ce que la prédiction soit plus élevée que la valeur de base) et, en bleu, celles ayant un impact négatif (contribuent à ce que la prédiction soit plus basse que la valeur de base)

Complexité algorithmique des modèles

L'article "Computational-complexity-learning-algorithms (Annexe 2)" dresse un tableau d'estimation des complexité des différentes familles de modèles.

On a donc si l'on se réfère aux modèles que nous avons entraînés:

Algorithm	Classification/Regression	Training	Prediction
Random Forest	C Breiman implementation	$O(n^2\sqrt{p}n_{trees})$	$O(pn_{trees})$
Linear Regression	R	$O(p^2n + p^3)$	$O(p)$
Gradient Boosting (n_{trees})	C+R	$O(npn_{trees})$	$O(pn_{trees})$

Avec :

- Gradient Boosting = light GBM
- Logistic regression = linear regression
- C Breiman Random Forest = Random forest

En considérant:

- n : le nombre d'échantillons d'entraînement,
- p : le nombre de fonctionnalités,
- n$trees$: le nombre d'arbres (pour les méthodes basées sur différents arbres),
- nsv: le nombre de vecteurs de soutien

- nli : le nombre de neurones à la couche i dans un réseau de neurones, nous avons les approximations suivantes.

On peut donc en conclure que durant la phase d'entraînement Light GBM sera le modèle possédant la plus faible complexité (ce qui s'est vu sur la durée de cette phase).
Par contre les complexités sont équivalentes pour la phase de prédiction avec un léger avantage pour la régression logistique.

Limites & améliorations

Un premier axe d'amélioration est d'intégrer dans l'espace de recherche d'autres familles de modèles et notamment l'implémentation d'un réseau neuronal.

Un second axe d'amélioration est d'utiliser des librairies d'optimisation des hyperparamètres telles qu'Optuna ou Hyperopt afin de chercher à améliorer encore nos modèles. On peut aussi essayer d'intégrer davantage d'étapes de pré-traitement.

Annexe:

Tour of Evaluation Metrics for Imbalanced Classification (Annexe 1)

Computational-complexity-learning-algorithms (Annexe 2)

Kaggle Project

Interpretable-ml-book

Graphics Principles Cheat Sheet

Computational-complexity-learning-algorithms

Dashboard flask

Shapash

Hyperopt

Pycaret