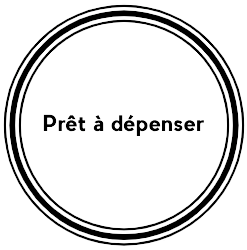
Projet 7



Moustafa ZMERLI
Septembre 2022

NOTE METHODOLOGOGIQUE

Implémentez un modèle de scoring

OPENCLASSROOMS

Table des matières

1. Contexte

La société « **Prêt à dépenser** » souhaite mettre en œuvre un outil de « **scoring crédit** » pour calculer la probabilité qu'un client rembourse son crédit d'après diverses données.

La mission est de développer un algorithme qui classifie la demande en crédit accepté ou rejeté.

Prendre en compte :

- Sélectionner un kernel Kaggle pour faciliter la préparation des données nécessaires à l'élaboration du modèle de scoring.
- Déployer le modèle de scoring comme une API.
- Construire un tableau de bord interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle de scoring.

1.1. Références

[1] Lien détail du projet 7 OPENCLASSROOMS : *https://openclassrooms.com/fr/projects/632/assignment*

[2] Lien Kernel Kaggle : https://www.kaggle.com/willkoehrsen/start-here-a-gentle-introduction

[3] Lien téléchargement des données : https://www.kaggle.com/c/home-credit-default-risk/data

[4] Librairie Python « Imblearn » pour équilibrage des données : https://imbalanced-learn.readthedocs.io/en/stable/user_guide.html

2. Présentation des données

L'entreprise met à disposition sept fichiers (.csv) contenant des données spécifiques à certains paramètres.

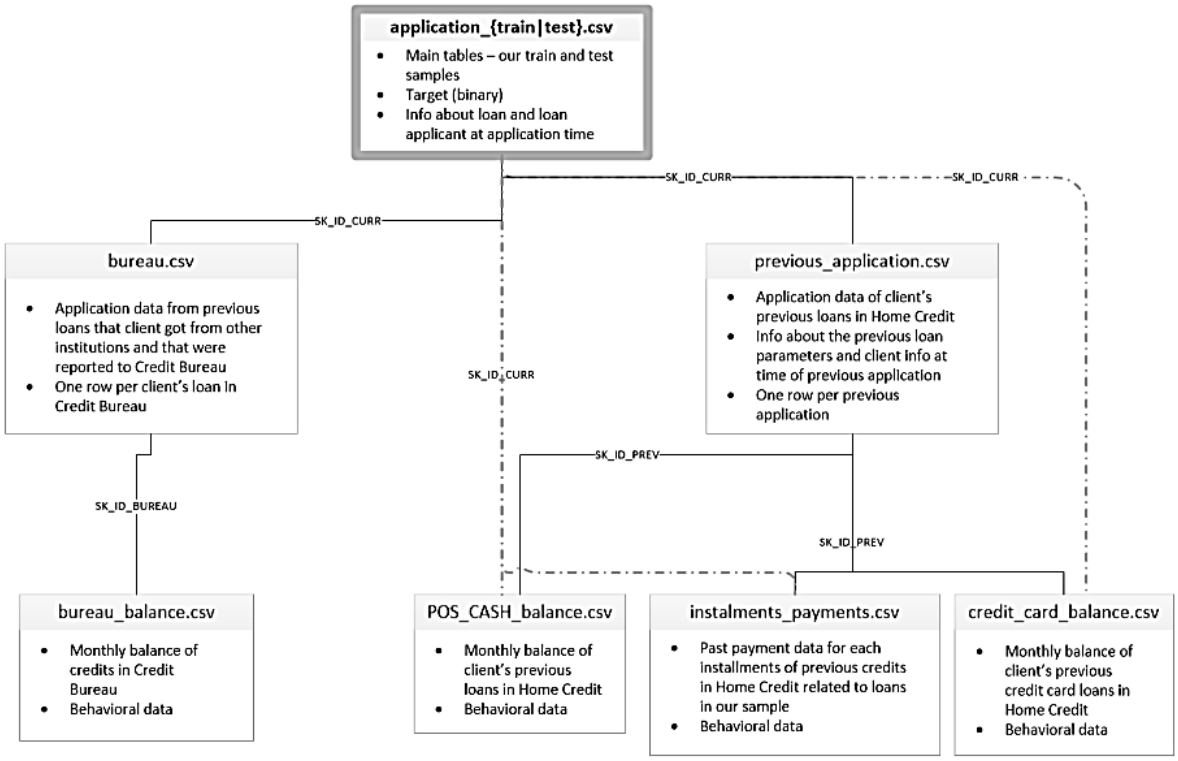


Figure 1. Structure du jeu de données. Source : https://www.kaggle.com/c/home-credit-default-risk/data

Il y a deux tables de données qui viennent **des autres institutions** :

- Bureau : Antécédents de crédit du client (autres institutions financières).
- Bureau_balance : Soldes mensuelles de crédits précédents.

Et quatre tables de données propres de « **Home credit** » :

- Previous_application: Demandes de crédit antérieures.
- Pos_cash_balance: Des bilans mensuels des anciens points de vente et des prêts cash.
- Installements_payments: Historique de remboursement des crédits précédents.
- Credit_card_balance: Solde mensuel des cartes de crédit antérieures.

- Nous disposons d'une base de données nommée « train », qui nous a servi à entraîner notre modèle. Cette base contient une variable « cible ». Nous avons aussi à disposition une base de données « test » que nous utilisons pour simuler un nouveau client dans la base. Toutefois il convient que ces deux datasets aient la même structure à l'issu du feature engineering.

3. Traitements des données

Un kernel Kaggle « LightGBM with Simple Features », a été utilisé et adapté pour faciliter la préparation des données « preprocessing » nécessaires à l'élaboration du modèle de scoring.

La sélection est basée sur

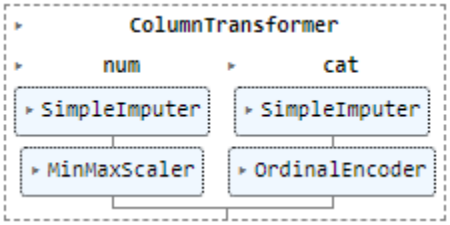
- Un bon score dans la compétition.
- Un Feature Engineering performant.
- Le kernel recommandé dans le projet.

Tout au long du kernel, des analyses exploratoires sont faites afin de déterminer l'influence des différentes variables sur la probabilité qu'un client rembourse ou pas son prêt (Target).

3.1 Traitement initiaux

Tout d'abord, les colonnes ayant un **taux de remplissage de 80 %** ou plus ont été conservées. Ensuite, les **valeurs aberrantes** ont été traitées à l'aide de la méthode Z-score.
Les **valeurs manquantes** ont été traitées en respectant le type de données, qu'elles soient catégorielles ou numériques:

- Données catégorielles :
  - Encodage avec OrdinalEncoder
  - Imputation avec la valeur la plus fréquente.
- Données numériques:
  - Imputation avec la valeur median.
  - Standariser les données avec MinMaxScaler.



L'échantillonnage du dataset a été réalisé avec la méthode « Train_Test_Split » de Scikit-learn. Avec un ratio de 80% pour les données d'entraînement et 20% pour les données de tests.

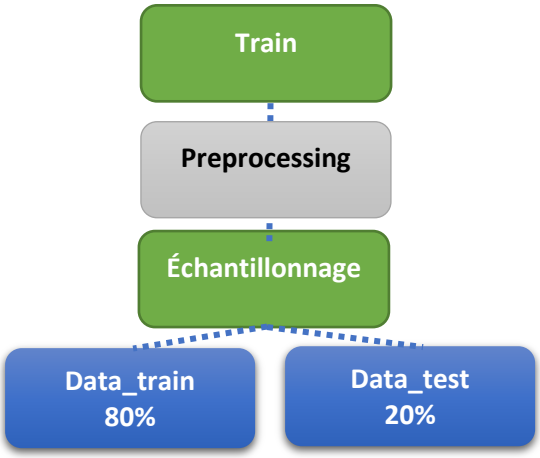


Figure 2. Traitement initiaux

3.2 Données déséquilibrées

L'analyse exploratoire a montré qu'il y a un fort déséquilibre dans la variable « Target », c.-à-d. entre les clients ne présentant pas de risque de défaut de payement et ceux présentant un risque. Ce déséquilibre pourrait avoir un impact sur la performance de l'algorithme et par la suite les prédictions seront faussées.

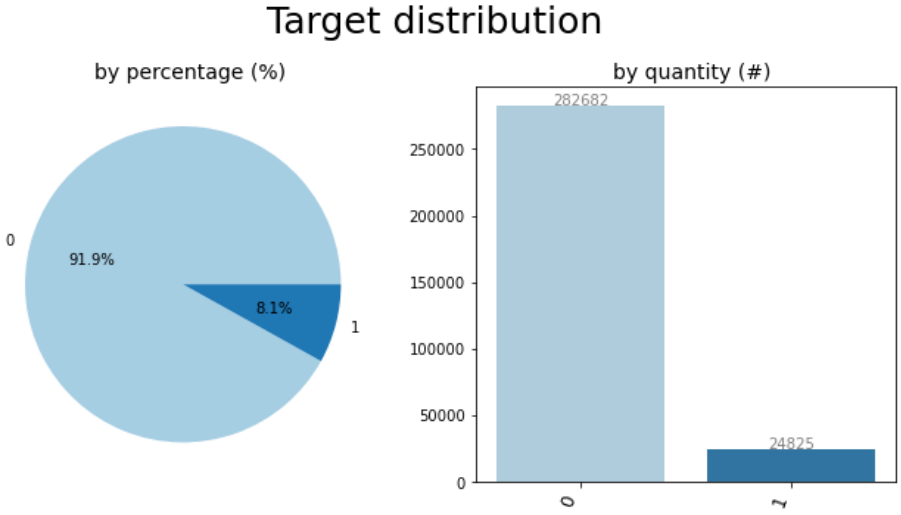


Figure 3. Distribution de la variable « Target ».

La raison en est que le modèle attribuera beaucoup plus fréquemment la classe la plus représentée aux clients. Pour pallier ce déséquilibre, quatre approches peuvent être appliquées sur les données d'entraînement :

• Class weights est une méthode directement gérée par les modèles et qui permet de pénaliser les poids associés aux observations de la classe sur-représentées (une pondération inversement proportionnellement à la fréquence des classes).

• 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.

• SMOTE est une méthode qui va créer de nouvelles données pour la classe sous-représentée à partir des données existantes (et donc de la variété) pour que chaque classe ait le même nombre de données que la classe sur-représentée à l'origine.

• 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.

→ Dans ce projet nous ne testons que deux stratégies : Class-weights et SMOTE *(Synthetic Minority Oversampling Technique).*

4. Modélisation

4.1 Modèles utilisés

Pour déterminer l'algorithme optimal de classification adapté à la problématique, quatre algorithmes ont été testés :

- Dummy Classifier (comme baseline)
- Classificateur de forêt aléatoire
- LightGBM
- Gradient Boosting

4.2 Métriques d'évaluation

4.2.1 Matrice de confusion

PRINCIPE :

Une matrice de confusion est l'un des meilleurs moyens d'évaluer les performances d'un modèle de classification. L'idée de base est de compter le nombre de fois que les

instances d'une classe sont correctement classées ou incorrectement classées comme une autre classe. Les colonnes d'une matrice de confusion représentent la classe réelle, tandis que chaque ligne représente la classe prédite. Un modèle parfait n'aurait que de vrais positifs et de vrais négatifs, ce qui signifie que sa matrice de confusion n'aurait que des valeurs non nulles sur sa diagonale. Dans une matrice de confusion, il existe deux types d'erreurs : les faux positifs (FP) ou les erreurs de type I et les faux négatifs (FN) ou les erreurs de type II. Ces termes proviennent de tests d'hypothèses en statistique et sont utilisés de manière interchangeable avec les problèmes de classification.

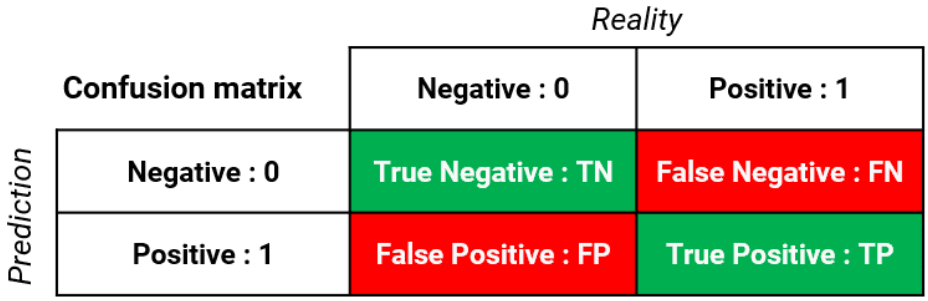


Figure 4 : Matrice de confusion des prédictions d'un modèle de classification binaire.

Pour notre problématique, le résultat attendu le plus important pour un client est la valeur de la probabilité de défaut de payement. En appliquant un seuil à cette valeur, nous pouvons lui affecter une valeur binaire (0 ou 1) suivant que la probabilité est inférieure ou supérieure au seuil.

Si la probabilité est inférieure au seuil, on considère que le crédit sera remboursé, la prédiction est négative (0). Inversement, si la probabilité est supérieure au seuil, on considère que le crédit ne sera pas remboursé, la prédiction est positive (1).

Ainsi :

- Accorder un crédit à un client ne pouvant pas le rembourser par la suite (FN) est synonyme de perte.
- 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.

Ces valeurs peuvent être traduites en des indicateurs caractérisant le modèle, dont voici les plus importants :

- Sensibilité = TP / (TP + FN) - Capacité du modèle à détecter les dossiers de crédit non remboursés (1)
- Spécificité = TN / (TN + FP) - Capacité du modèle à détecter les dossiers de crédit remboursé (0)
- Précision = TP / (TP + FP) - Capacité du modèle à détecter les vrais dossiers non remboursés (1)

4.2.2 Courbe ROC et Score AUC

Nous pouvons représenter les évolutions de la sensibilité et de la spécificité en fonction du seuil de classification avec une courbe ROC.

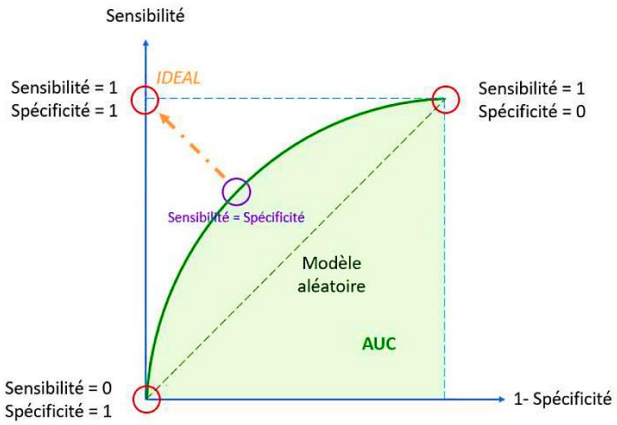


Figure 5 : Courbe ROC d'un modèle de classification binaire.

Une courbe ROC caractérise le classifieur qui a produit les résultats sous forme de probabilités par variation du seuil de classification. Un modèle idéal a une sensibilité et une spécificité = 1. Plus la courbe se rapproche de cet idéal, meilleurs sont les indicateurs.

⇒ On peut résumer la mesure de cette performance par l'aire sous la courbe (Area Under the Curve - AUC).

Ainsi pour le modèle utilisé pour cette étude, nous avons cherché à maximiser le score AUC.

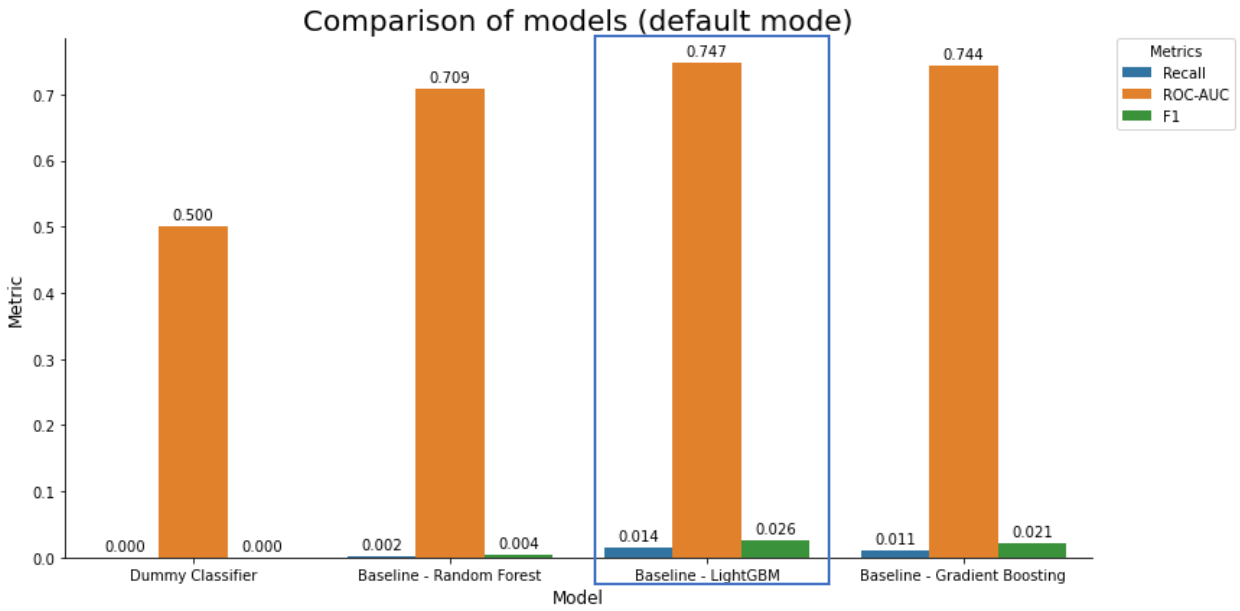


Figure 6. Comparaison des métriques de différents modèles

Au moment de réaliser la modélisation en prenant en compte les valeurs par défaut, le LightGMB a obtenu le meilleur résultat.

4.2.3 Fonction coût (métrique métier)

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).

Pour ce projet, une fonction coût a été développée dans l'objectif de pénaliser des Faux Négatifs.

Un Faux Positif (FP) constitue une perte d'opportunité pour la banque, à la différence d'un Faux Négatif (FN) qui constitue une perte pour créance irrécouvrable.

Ces valeurs de coefficients signifient que les Faux Négatif (FN) engendrent des pertes 10 fois supérieures aux autres.

- Poids/rate :

TN_rate et **TP**_rate	= **1**
FP_rate	= **-1**
FN_rate	= **-10**

Ces poids ont été fixés arbitrairement et il est tout à fait envisageable de les modifier en fonction de l'optique métier.

score = (gain_total - gain_minumun) / (gain_maximun - gain_minumun)

avec :

gain_total = TN*TN_rate + TP*TP_rate + FP*FP_rate + FN*FN_rate
gain_maximun = total_not_default*TN_rate + total_default*TP_rate
gain_minumun = total_not_default*TN_rate + total_default*FN_rate

et:

total_not_default = TN + FP
total_default = TP + FN

4.2.4 Résultats de modélisation

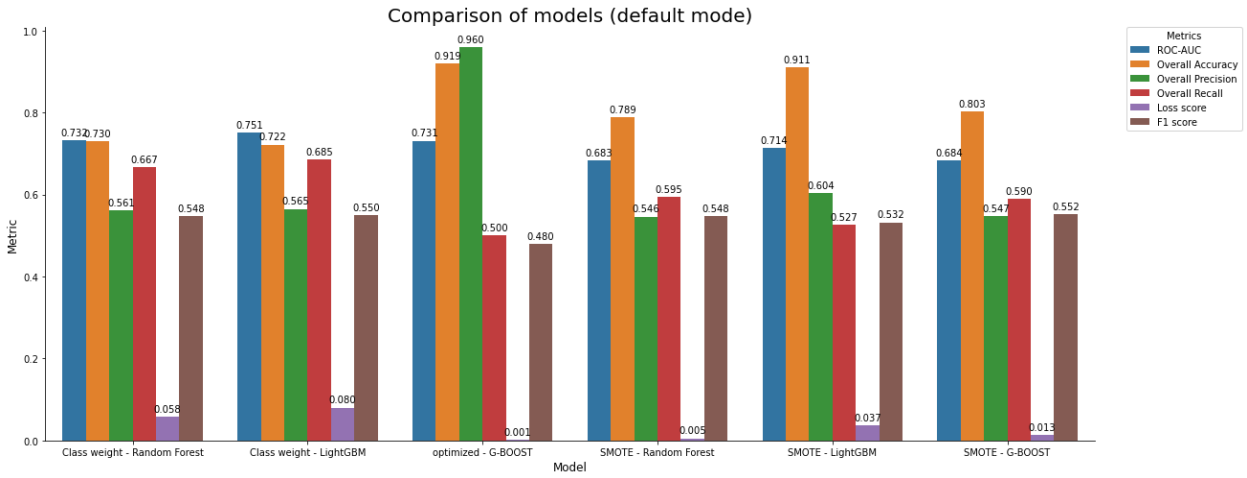


Figure 7. Comparaison des métriques de différents modèles (résultat de modélisation).

Ces résultats sont obtenus après avoir optimiser les hyperparamètres liés à chaque algorithme.

On remarque que la méthode Class-Weight donne des résultats meilleurs que SMOTE. On note que le Gradient Boosting n'a pas le paramètres Class weight.

En se basant sur la métrique ROC-AUC et Loss Score le modèle « Class-Weight LightGBM » a été choisi pour la suite de traitement.

5. Interprétation du modèle

5.1. Feature importance

Le modèle LightGBM optimisé final contient une méthode permettant pour chaque variable de calculer l'importance de cette variable pour le modèle. Une fois normalisées, nous pouvons comparer l'importance relative de chacune des variables et par un simple tri, afficher les 20 premières variables les plus importantes.

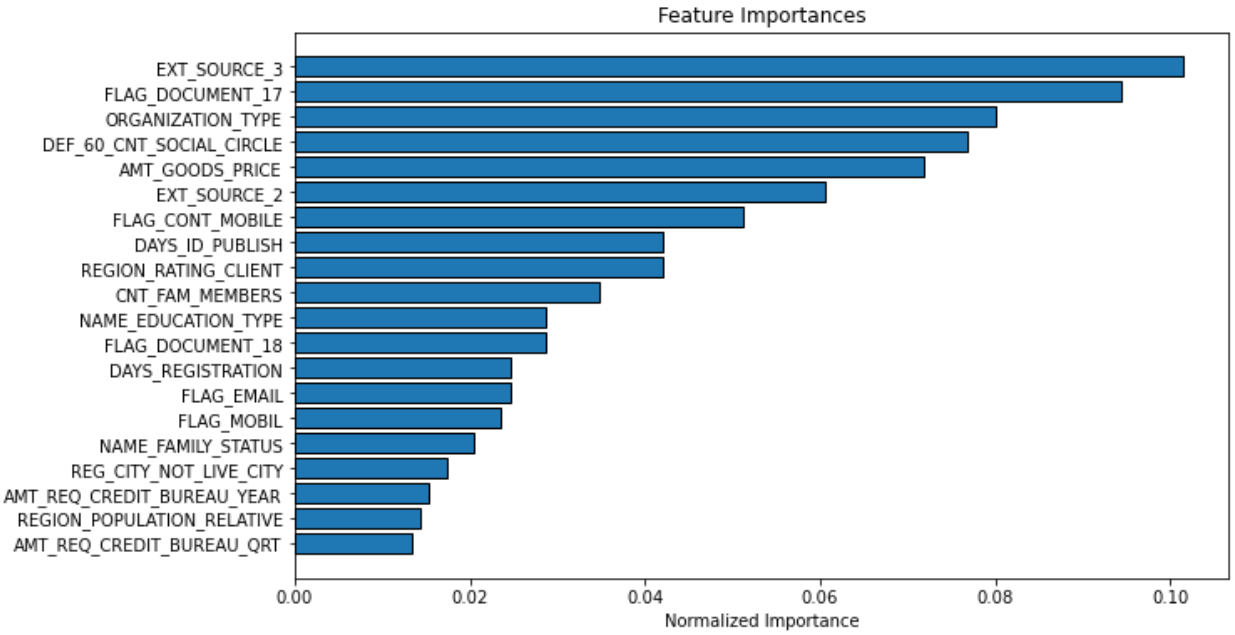


Figure 8. Feature importance

- Les variables les plus importantes proviennent d'une source **externe 3 et 2.** Ce sont des scores normalisés provenant d'autres institutions.

- D'autres variables sont importantes comme : les documents fournis par le client, l'entreprise où il travaille, l'environnement social, le montant du prêt demandé…

- En conclusion le modèle a pris en compte les différents types de variables : ***Variables personnelles***, ***Variables bancaires***, ***Variables externes.***

5.2. Interprétation locale

Pour plus de précision et pour connaître le sens d'influence de chacune des variables localement pour chaque prédiction, il est possible d'utiliser la librairie SHAP (SHapley Additive exPlanations) qui explique la sortie de tout modèle d'apprentissage automatique en utilisant la théorie des jeux.

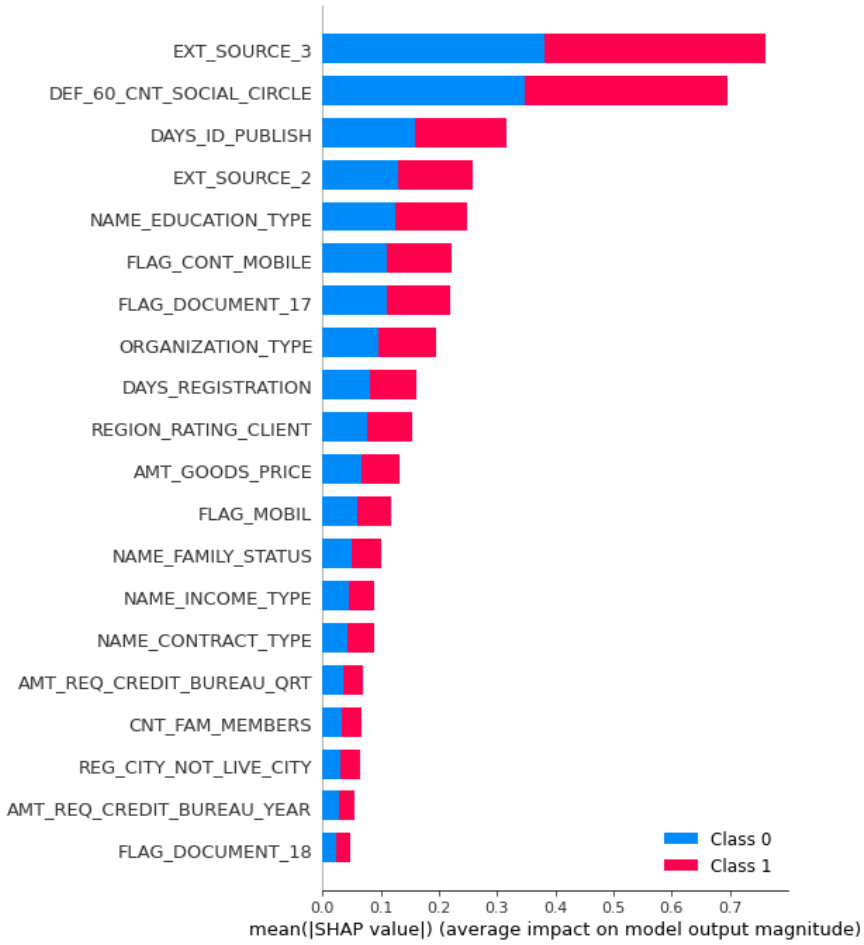


Figure 9. Influence de chacune des variables localement sur la prédiction

6. Limite et améliorations possibles

Limite :

- Ce modèle ne comporte pas un traitement spécifique des valeurs manquantes. Les variables d'entrée sont donc toutes indispensables à son bon fonctionnement.
- Le feature engineering a été effectué sans connaissances réelles du secteur bancaire.
- La modélisation a été effectuée sur la base d'une métrique généraliste AUC ou bien personnellement créée pour répondre au mieux au besoin de gain d'argent d'une banque. Les coefficients de cette métrique ont été choisis arbitrairement selon le bon sens.

L'axe principal d'amélioration :

- Définir plus précisément les coefficients de Loss-Score (Score personnalisé).
- Réduire les données pour éviter le « Curse of Dimensionality ». *Il faut prendre en compte plusieurs méthodes de Feature Selection par Scikit-Learn, PCA, T-SNE, etc.*
- La mise en place d'une étape de traitement des valeurs manquantes, adaptée aux gros jeux de données.
- Un échange avec les experts métier permettrait :
    - D'effectuer un feature engineering plus pertinent (calcul d'indicateurs fiscaux spécifiques).
    - De mieux sélectionner les variables corrélées à enlever ou conserver en fonction des connaissances métier.
    - De réduire le temps de calcul en éliminant les variables qui ont de faibles impacts sur la prédiction (point de vue métier).