**Loan's Attribution Predictor**


# DS Projet 7 : **IMPLEMENTEZ UN MODELE DE SCORING**

## *NOTE METHODOLOGIQUE*

***Patrick TCHOMTE*** | *03 Janvier 2023*

# Table des matières

# Méthodologie d'entraînement du modèle

## *Problématique*

L'entreprise **« Prêt à dépenser »** souhaite mettre en œuvre un outil de **«scoring credit »,** qui permettrait de déterminer la probabilité d'un client à rembourser son crédit.

L'outil de *scoring* recoupera les informations collectées de plusieurs sources de données variées. Afin de déterminer la solvabilité du client.

Il est donc question d'un problème de **classification supervisée binaire**. La variable cible **(TARGET)** prendra la valeur : 0 si le client est solvable et 1 s'il ne l'est pas.

## *Feature engineering*

Un dataset complexe a été considéré.

Il a été créé à partir du jeu de données entièrement inspiré du **kernel Kaggle** disponible sur *https://www.kaggle.com/code/jsaguiar/lightgbm-with-simplefeatures/script*

## *Simplification et Séparation en jeux d'entraînement et de test*

Compte tenu de la taille du dataset résultant du feature engineering **(>300000 lignes** et **>700 variables**), seulement un échantillon de **10%** a été considéré.

Cet échantillon a ensuite été découpé avec la méthode du *« train_test_split »* en 2 jeux avec un ratio entrainement-test de **70-30** pour éviter tout biais.

## *Solution pour l'équilibrage des classes considérées*

Compte tenu du haut "accuracy" du **DummyClassifier** qui s'explique par le déséquilibre de la variable **TARGET** *(9 clients solvables pour 1 non solvable)*.

On constate que notre problème de classification binaire est déséquilibré.

Pour équilibrer les données la méthode **SMOTE** *(répétition des observations de la classe minoritaire)* a été appliquée aux données.

# Algorithme d'optimisation et métrique d'évaluation

## *Algorithmes considérés*

Les algorithmes de classification considérés pour modéliser au mieux notre problème sont de familles différentes à savoir :

**– Régression Logistique** *(un algorithme linéaire)*

**– KNeighborsClassifier** *(un algorithme ensembliste)*

**– Light Gradient Boosting Machine** *(un algorithme gradient boosting)*

## *Métrique d'évaluation priviléqiée :*

**AUC/ROC** : Plus l'AUC est élevé, plus le modèle est capable de prédire les classes correctement.

## *Prétraitement*

Avant d'entraîner les différents modèles et de comparer leur métrique, un pipeline de prétraitement est appliqué.

Ce pipeline permet :

− Suppression des valeurs manquantes pour éviter tout biais

− La distribution plus harmonieuse des variables avec **StandardScaler, RobustScaler, QuantileTransformer, MinMaxScaler, MaxAbsScaler, Normalizer, PowerTransformer**

− L'équilibrage des classes avec la méthode **SMOTE** plus haut.

Ce pipeline est intégré à une fonction de *Gridsearch* qui permet d'identifier pour chaque algorithme, le meilleur pré-traitement en considérant comme *scoring*, la métrique **AUC**.

## *Meilleur modèle*

Les résultats du pipeline sur les modèles ont été classifiés par grande valeur de **AUC**.

Le meilleur modèle est **KNeighborsClassifier** obtenu avec :

− Le prétraitement (**QuantileTransformer**)

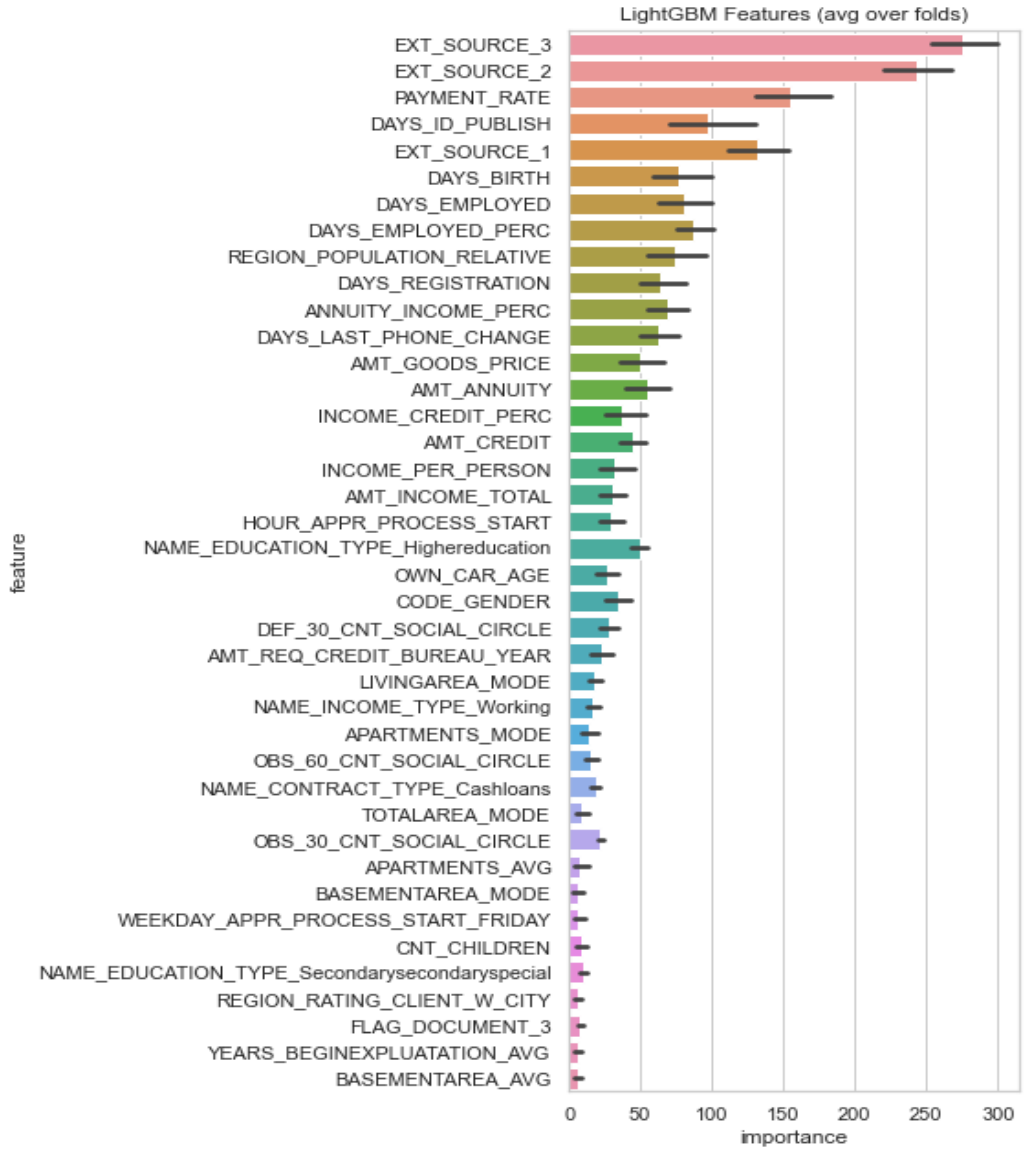
− Les 3 hyperparamètres suivants **KNeighborsClassifier**(*n_neighbors*=**8**, *metric*="**manhattan**", *weights*="**distance**").

## Interprétabilité globale et locale du modèle

*1-ère correction apportée après feedback du Mentor-Evaluateur (1ère soutenance 03/01/2012)*

Dans notre cas le meilleur modèle étant **KNeighborsClassifier**, il n'est pas naturel et intuitif de calculer l'importance des features qui contribuent à l'élaboration du modèle. Parce qu'il n'existe pas d'attribut ***_importance*** pour l'instance de cette estimateur *(nativement, **KNeighbors** ne possède pas l'attribut _importance, contrairement à la plupart des autres estimateurs).*

Compte tenu de ce fait, nous ne pourrons faire qu'une interprétabilité globale sur la base de la méthodologie utilisée dans le *kernel Kaggle*. Qui nous a permis de sélectionner les 30 features les plus importants qui ont permis de construire notre modèle.

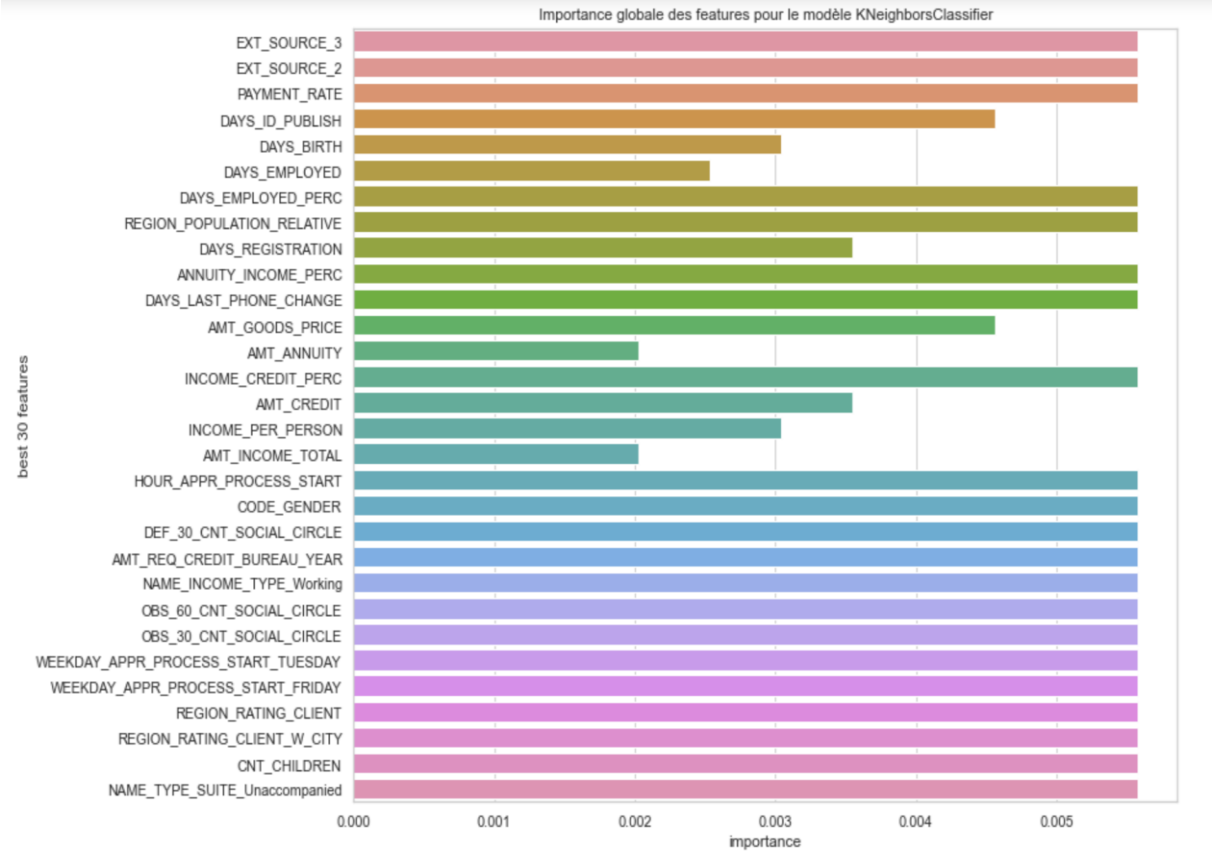


Nous avons tout de même dans nos recherches, essayer de déterminer l'importance globale de features qui ont permis de construire notre meilleur modèle.

Nous nous sommes inspirés des travaux publiés sur ce lien :

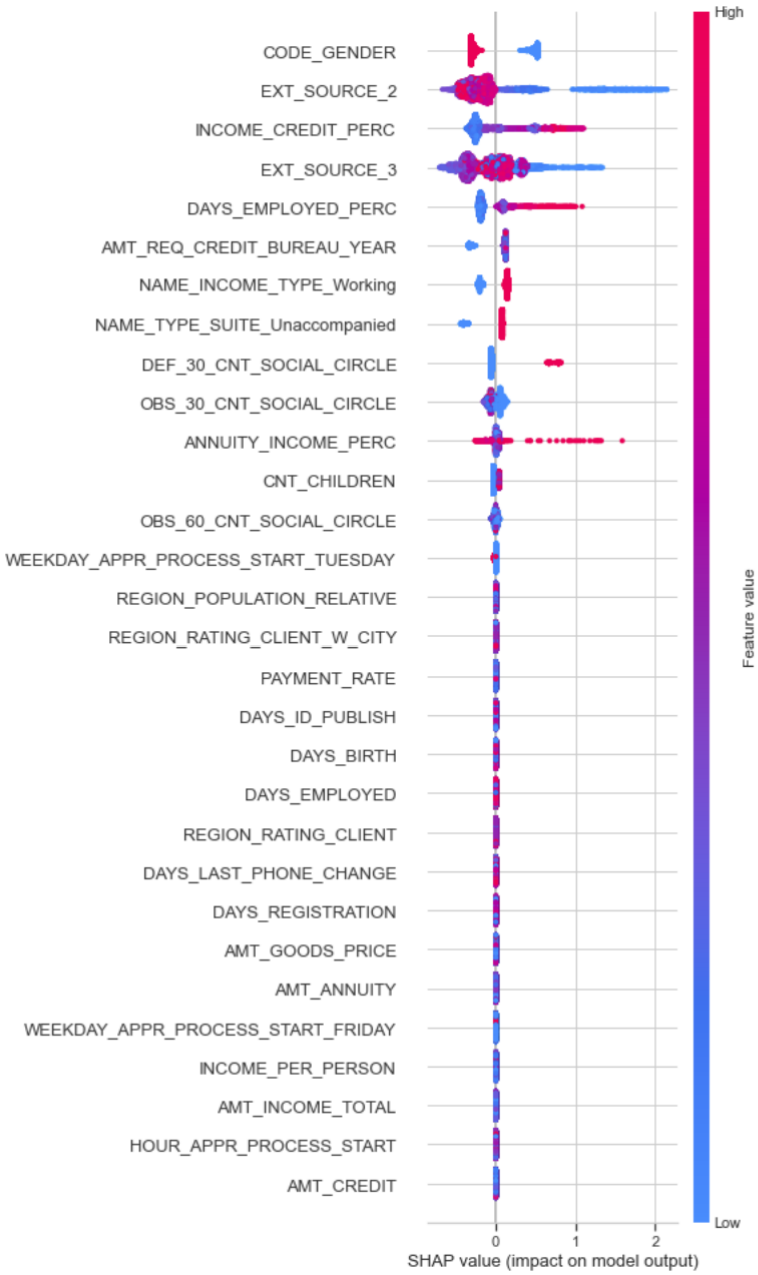
https://github.com/scikit-learn/scikit-learn/issues/8898

Nous avons obtenu, les résultats ci-après après implémentation de la méthodologie :



*2-ème correction apportée après feedback du Mentor-Evaluateur (2ère soutenance 04/01/2023)*

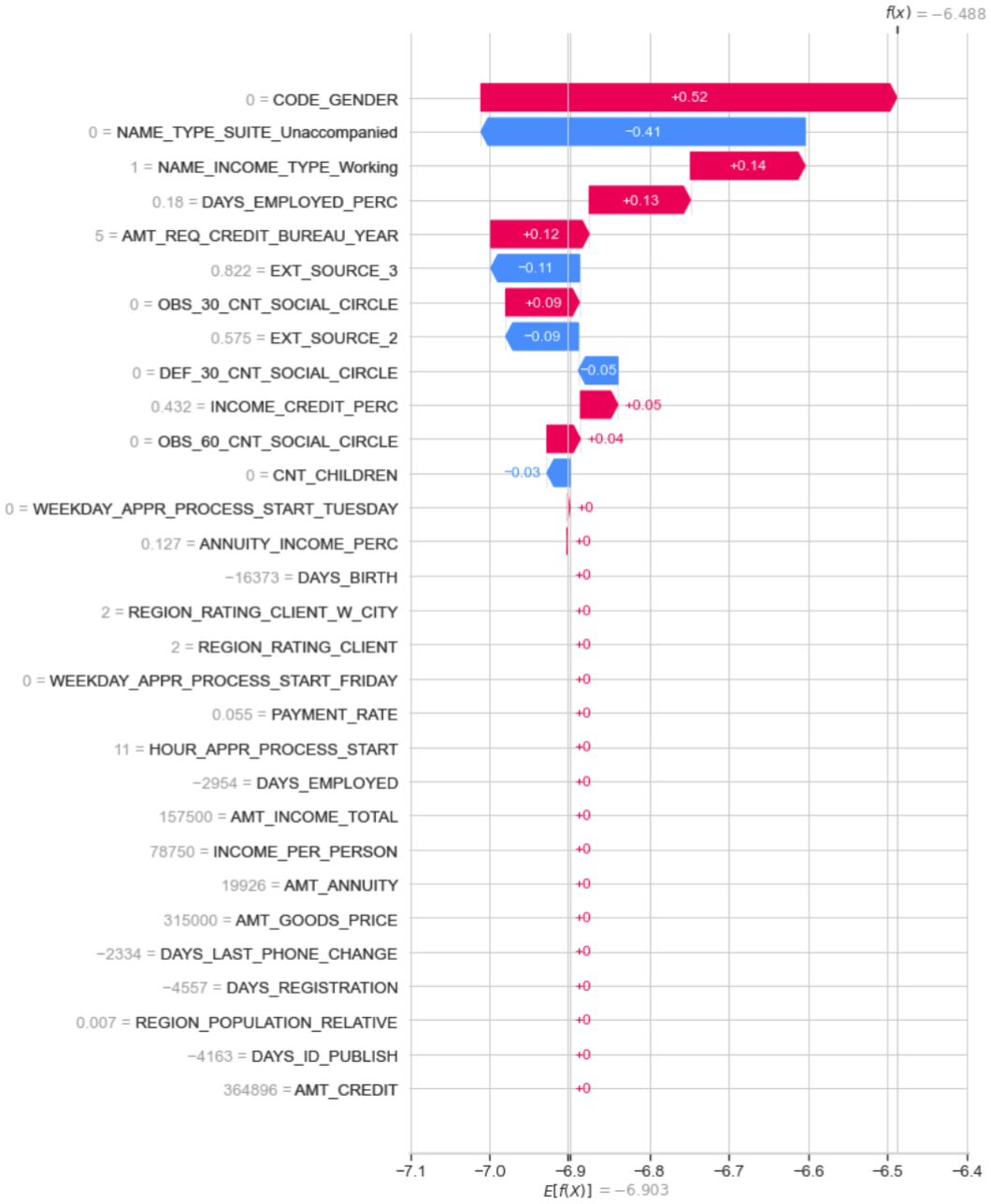
- ***Interprétabilité globale :***




Nous observons que les features qui ont une forte importance dont une forte influence globalement sur le 2-ème meilleur modèle de prédiction **(LGBMClassifier)** sont notamment

ceux qui sont de couleur rouge ou qui tendent vers le rouge. Plus la couleur rouge est foncée pour un feature, plus son influence sur le modèle est importante.

- ***Interprétabilité locale :***




Le graphe ci-dessus représente les impacts des variables sur la prédiction du scoring d'un client en particulier. Les variables rouges contribuent à une prédiction négative *(un client solvable)* et les variables bleues contribuent à une prédiction positive *(un client non solvable)*.

## Limites et améliorations possible

La taille du dataset considéré *(kernel Kaggle)* étant considérable. Pour effectuer nos travaux nous avons considéré un échantillon de 10K individus du dataset résultant du formidable de feature engineering réalisé dans le kernel mis à notre disposition pour ce projet.

La partie feature engineering pourrait éventuellement être optimisée avec la collaboration des spécialistes du métier.

Concernant la partie modélisation, seulement 1 méthode d'équilibrage, 3 algorithmes de 3 familles différentes, 7 « scalers », 3 hyperparamètres ont été considérés pour l'entrainement.

Concernant la fonction coût métier et le calcul du seuil, des hypothèses ont été émises, et demandent une confirmation/correction des équipes spécialisées.

Le Dashboard pourrait être enrichi en graphiques, en widgets, fonctionnalités « métiers », en ergonomie, le design etc.