07/06/2021

jeremie.aucher@gmail.com

Jérémie Aucher

Note Méthodologique

Projet 7 du parcours Data Scientist

# Table des matières

[I. Table des matières 1](#_Toc73973056)

[II. Introduction 2](#_Toc73973057)

[III. Rappel de l’objectif et démarche générale : 2](#_Toc73973058)

[IV. La méthodologie d'entraînement du modèle 3](#_Toc73973059)

[A. Précautions prises pour évaluer correctement des algorithmes entre eux 4](#_Toc73973060)

[B. Critère d’évaluation des algorithmes 4](#_Toc73973061)

[C. Pré-Sélection des modèles 5](#_Toc73973062)

[D. Sélection Finale de l’algorithme retenu : LGBMClassifier 6](#_Toc73973063)

[V. La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation 9](#_Toc73973064)

[A. La métrique technique : ROC AUC 9](#_Toc73973065)

[B. Une métrique métier : Définition d’un seuil à partir d’une fonction coût à minimiser 11](#_Toc73973066)

[VI. L’interprétabilité du modèle 13](#_Toc73973067)

[A. Interprétation Globale 13](#_Toc73973068)

[B. Interprétation Locale 14](#_Toc73973069)

[VII. Les limites et les améliorations possibles 15](#_Toc73973070)

[A. Choix du pré-traitement des données 15](#_Toc73973071)

[B. Choix de la mesure : 15](#_Toc73973072)

[C. Choix des hyperparamètres à optimiser 15](#_Toc73973073)

[D. Pour le Dashboard – Possibilité de modifier les valeurs du client et recalculer son score 15](#_Toc73973074)

# Introduction

Dans cette note méthodologique   
nous allons passer en revu :

1. Rappel de l’objectif et démarche générale
2. La méthodologie d'entraînement du modèle
3. La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation
4. L’interprétabilité du modèle
5. Les limites et les améliorations possibles

# Rappel de l’objectif et démarche générale :

L’objectif a été de construire un modèle de scoring qui puisse donner une prédiction   
sur la probabilité de faillite d'un client de façon automatique.

La démarche a été de déterminer :

1. Quel est l'algorithme le plus adapté pour remplir cette tâche
2. Les meilleurs hyperparamètres à utiliser sur cet algorithme

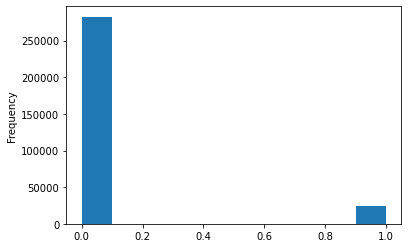
J’ai commencé par évaluer un ensemble de 8 algorithmes   
avec des paramètres prédéfinis.

L'idée a été de pouvoir rapidement mettre de côté les algorithmes   
qui ont montré d'office qu'ils n’étaient pas adaptés à la problématique.

Dans un second temps, je me suis intéressé aux algorithmes toujours en course   
et je suis allé plus loin dans leur sélection en recherchant pour chacun d'eux,   
leurs meilleurs hyperparamètres avec gridSearchCV.

# La méthodologie d'entraînement du modèle

Nous sommes dans problème de classification à deux classes (0 ou 1).  
De plus, les classes sont déséquilibrés.



Pour évaluer correctement l’entrainement des modèles   
il faut suivre une méthodologie précise.  
L’objectif est double :

1. Pouvoir évaluer les modèles sur des critères précis qui permettent objectivement de déterminer l’algorithme le plus adapté à notre situation
2. Eliminer les facteurs qui puissent induire de faux résultats, comme l’influence de l’aléatoire qui peut intervenir à différentes étapes de l’évaluation des algorithmes et qui peut avoir une influence d’un test d’un algorithme à un autre

Voici les étapes qui définissent l’évaluation d’un algorithme :

1. Génération des données d’entrainement et de test  
   Ces mêmes données seront utilisées par l’ensemble des algorithmes testés entre eux
   1. Génération du train-set et du test-set
   2. Génération de l’estimation de la pondération des classes   
      (car notre jeu de données est déséquilibré)
   3. Génération des données de l’étape précédente pour différents pourcentages   
      des données disponibles auquel cas l’entrainement serait trop long) soit :
      1. 100% des données disponibles
      2. 10% des données disponibles
      3. 1% des données disponibles
      4. 0.1% des données disponibles
   4. Génération des StratifiedKFold (5 par défaut)
2. Entrainement du modèle
3. Récupération des scores des différents folds
4. Calcul du score moyen des folds
5. Récupération du score de l’algorithme entraîné sur le test-set
6. Affichage et interprétation des résultats

## Précautions prises pour évaluer correctement des algorithmes entre eux

Comme indiqué au chapitre précèdent, les algorithmes ont tous été testés   
dans scrupuleusement les mêmes conditions, c’est-à-dire les mêmes :

1. données de train/test
2. folds générés
3. poids attribués aux classes

Cette étape très importante nous permet d’exclure des différences de résultats du aux tirages aléatoires présents à chacune de ces étapes et qui peuvent avantager ou désavantager les algorithmes.  
Nous sommes donc absolument sûre que les résultats obtenus pour chaque algorithme le sont dans les mêmes conditions et qu’en définitives, nous pouvons comparer et sélectionner objectivement les algorithmes entre eux sur la base de ces résultats.

## Critère d’évaluation des algorithmes

J'ai évalué les algorithmes sur les critères suivants :

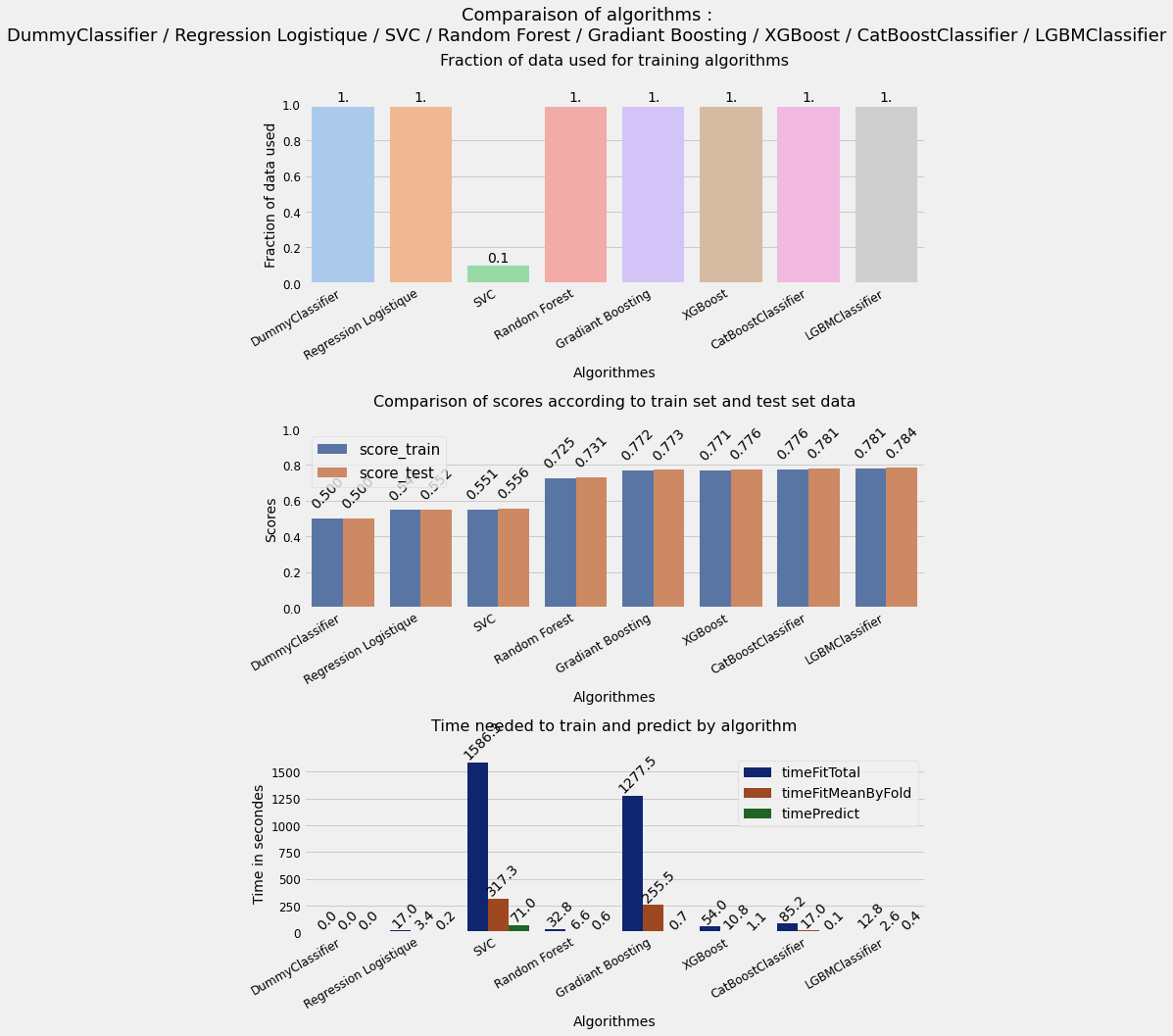
* Rapidité d'entrainement
* Rapidité d'exécution
* Score du train-set (moyen et par fold)
* Score du test-set

Même si les algorithmes ont été testé sur le test-set,   
c’est sur les résultats sur le train-set que les décisions ont-été prises.

J’ai également utilisé l’algorithme **Dummy Classifier** comme baseline.  
Cette algorithme réalise toujours la même prédiction, et les algorithmes qui seront susceptibles d’être sélectionnés pour répondre à notre problématique devront bien-entendu faire mieux que lui.

**Les scores affichés sont issus de la métrique ROC AUC**.  
Je reviendrai sur le choix de cette métrique au **chapitre V**.

## Pré-Sélection des modèles



L’algorithme **SVC** est disqualifié d’office par un temps d’entrainement excessivement long.

Si l’on se concentre sur leurs scores, les 3 meilleurs algorithmes sont respectivement :

1. LGBMClassifier
2. CatBoostClassifier
3. XGBoost

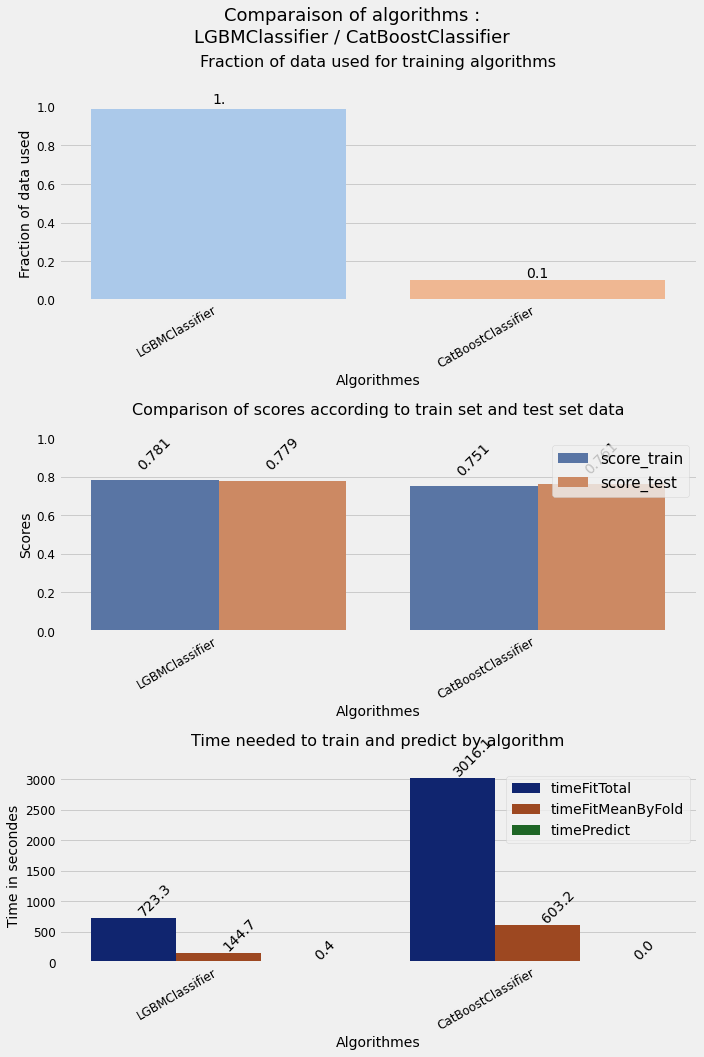
Parmi ces 3 algorithmes, CatBoostClassifier semble le plus lent à entraîner suivi par XGBoost.

LGBMClassifier a l’avantage de prendre en charge l’apprentissage parallèle et l’utilisation des GPU.

L’algorithme Regression Logistique et SVC et même (dans une moindre mesure) ont des scores très en retrait sur nos 3 algorithmes cités précédemment.

## Sélection Finale de l’algorithme retenu : LGBMClassifier

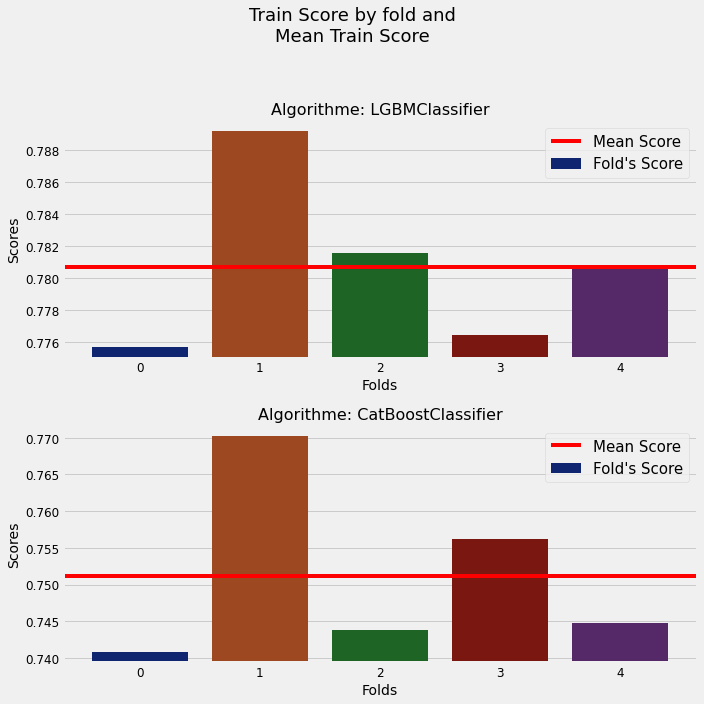
Comme le montre les captures des résultats ci-dessous, l’algorithme LGBMClassifier est le plus performant et l’algorithme retenu dans la résolution de notre problématique.



LGBMClassifier se distingue de CatBoostClassifier aussi bien sur la qualité de ses prédiction que par sa vitesse d’entrainement.

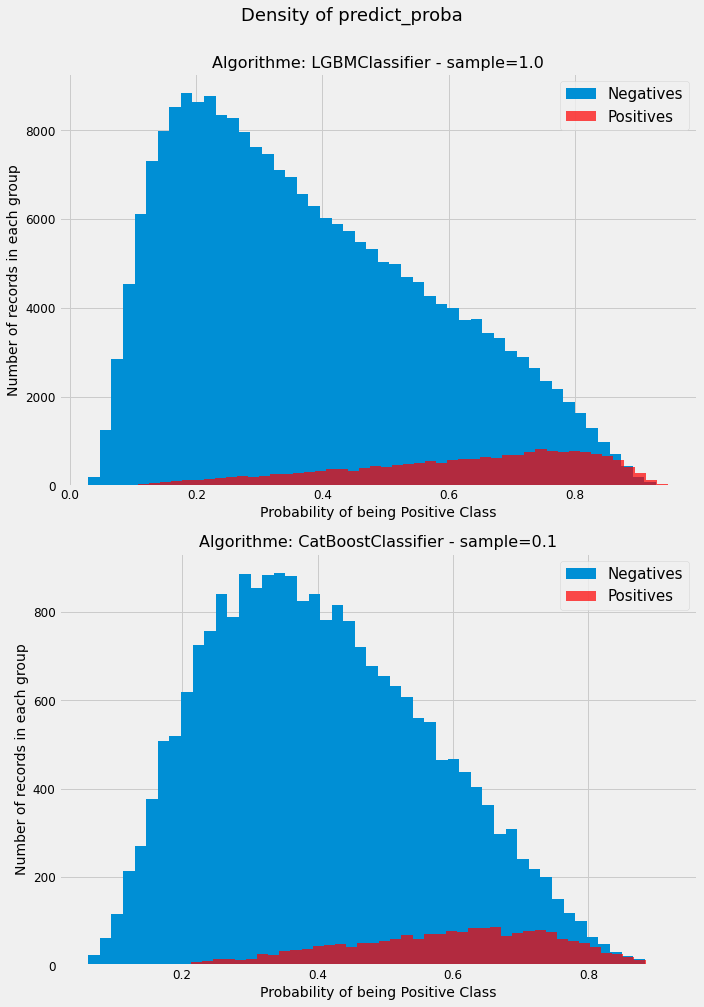
On peut imaginer que l’on prend le temps que l’on veut pour entrainer un modèle de Machine Learning.

Comme leur temps de prédiction sont très rapide pour l’un comme pour l’autre, la sélection se fait ici sur la base de l’étude de leurs scores.



LGBMClassifier est égalemeznt plus stable que son homologue si on compare le delta des scores par fold.

LGBMClassifier a un delta d’environ 0.012 points entre ses deux folds extrêmes quand CatBoostClassifier a un delta d’environ 0.02.



Même si elle n’est pas très intuitive pour départager les deux algorithmes, je préfère LGBMClassifier lors de l’analyse des courbes de densité de probabilité en fonction de la TARGET.

Je trouve la courbe de LGBMClassifier plus propre, mieux séparer que pour CatBoostClassifier mais cela reste subjectif.

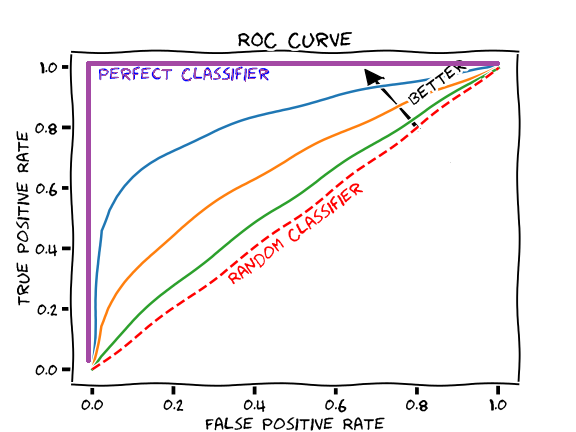
En définitive, le modèle le plus performant retenu est l’algorithme **LGBMClassifier**.

# La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation

Deux métriques ont été utilisés successivement :

* Une métrique technique : **ROC AUC**
* Une métrique métier : **Définition d’un seuil à partir d’une fonction coût à minimiser**.

## La métrique technique : ROC AUC



Cette métrique est adaptée dans les cas où les classes sont déséquilibré comme ici.  
Elle permet une meilleure évaluation d’un modèle que la précision par exemple.

Il s'agit d'une métrique de classification commune connue sous le nom d'aire sous la courbe de la caractéristique d'exploitation du récepteur (ROC AUC, parfois aussi appelée AUROC).

La courbe ROC (Reciever Operating Characteristic) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification.

Cette courbe trace le taux de vrais positifs en fonction du taux de faux positifs

Cette métrique est comprise entre 0 et 1, un meilleur modèle obtenant un score plus élevé.  
Un modèle qui se contente de deviner au hasard aura une AUC ROC de 0,5.

Enfin, lorsque nous mesurons un classificateur selon l'AUC ROC, nous ne générons pas des prédictions 0 ou 1, mais plutôt une probabilité entre 0 et 1.

La métrique ROC AUX est donc adapté pour pouvoir déterminer la performance des modèles et déterminer les plus performant dans notre problématique.

## Une métrique métier : Définition d’un seuil à partir d’une fonction coût à minimiser

Une fois notre meilleur algorithme trouvé et optimisé sur la métrique ROC AUC,   
il nous faut pouvoir l’adapté pour répondre au mieux à notre problématique.

La métrique ROC AUC nous donne une probabilité qu’un échantillon appartienne à une classe. Par défaut le seuil de décision est fixé à 0.5.

Par exemple, si le model prédit qu’un client à une probabilité de 0.6 d’appartenir à la classe des clients qui honore le remboursement de leur prêt (classe 0), alors en fonction du seuil fixé à 0.5, le client sera classé dans la classe 0. Si le seuil avait été fixé à 0.8 alors le client aurait été classé dans la classe 1 et le prêt lui aurait été refusé.

Le choix du seuil est très important.  
C’est lui qui détermine la réussite de l’objectif fixé.

L'idée finale est de maximiser les gains financiers de l'entreprise de prêts (et de minimiser ses pertes). L'entreprise réalise ses bénéfices avec les intérêts des prêts qu'elle octroie. Ne pas prêter d'argent à un client qui aurait remboursé son prêt est un manque à gagner pour l'entreprise. Cependant, lorsque l'entreprise prête à un client qui ne pourra pas rembourser son prêt, il s'agit d'une perte sèche.

On peut aisément comprendre qu'il y a un plus gros risque  
de prêter de l'argent à un client qui ne remboursera  
pas son prêt, qu'il y a de gain à prêter à un  
client qui remboursera et payera ses intérêts.

Il nous faut donc déterminer le coût lorsque l'entreprise  
prête à un mauvais client comparé au manque à gagner de refuser  
de prêter à un bon client.

Calcul posé :

**FN**(Faux Négatif c’est-à-dire Faux Bon Client): Prêt à tort ===> Perte du capital ou partie du capital ===> Perte de beaucoup d'argent  
**FP**(Faux Positif c’est-à-dire Faux Mauvais Client): Refus à tort de prêt ===> Perte de Marge, Manque à gagner

Hypothèse : Imaginons que nous ayons un capital valant **100**.

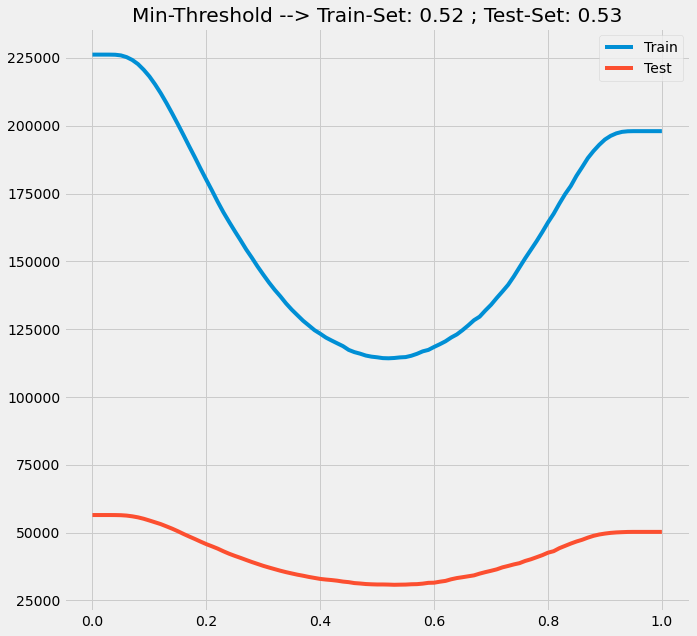
* FN ---> Perte de 50% du capital prêté en moyenne ---> Perte de 50
* FP ---> Manque à gagner de 1%/an sur en moyenne 10ans  
  (capital va de 100 à 0 soit moyenne de 50) ---> 10% de 50 ---> Perte de 5

Rapport de coût entre FN et FP:

* Les FN coûtent 10 fois plus chers que les FP.
* **1xFN = 10FP**

Objectif:

* Optimiser la Matrice de confusion en optimisant la somme de ces 2 coûts.
* Résoudre : Min(10FN + FP)
* Tester tous les seuils et calculer cette fonction



**Nous obtenons donc un seuil optimal fixé à 0.52**.

# L’interprétabilité du modèle

Pouvoir interpréter un modèle nous permet de comprendre comment   
il a été amené à prendre ses décisions.

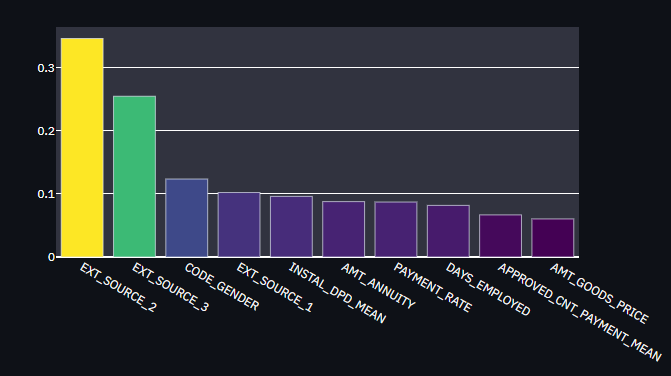
J’ai implémenté l’interprétation des modèles dans le Dashboard.  
J’ai développé le dashboard dans l’idée qu’il puisse être utilisé avec différent modèle.

Pour cette raison, je n’ai pas utilisé les fonctions intégrées aux modèles lorsque cela était possible comme avec LGBMClassifier, mais j’ai opté pour une méthode plus généraliste : SHAP.

Nous pouvons interpréter un modèle à deux niveaux : Globalement puis Localement.

## Interprétation Globale

L’interprétation globale permet de savoir quelles sont les caractéristiques   
qui ont le plus influencés le model dans le calcul du score.

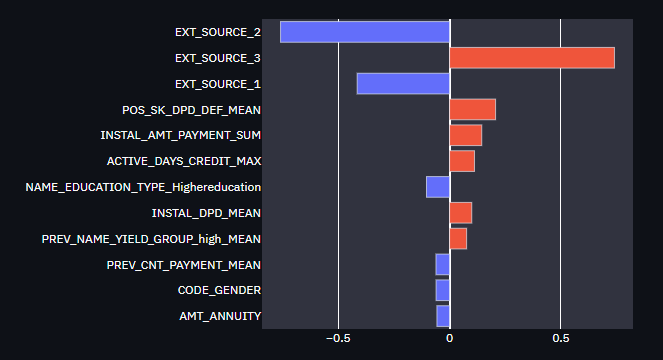


Nous constatons que les principaux critères identifiés sont, dans l’ordre :

1. Les scores obtenus auprès des autres agences de crédit
2. Le genre
3. L’âge
4. L’ancienneté dans l'emploi
5. Le montant du bien acheté
6. Le montant des annuités.

## Interprétation Locale

Les critères retenus peuvent être très variable d’un individu à l’autre.  
Imaginons qu’un client possèdes des valeurs situées dans la moyenne concernant les scores obtenus auprès des agences de crédit, l’âge mais une valeur « extrême » concernant l’ancienneté de l’emploi.  
Alors que les premières séries de variables pèseront peu dans la prise de décision, l’ancienneté de l’emploi jouera un rôle prépondérant dans la prise de décision de l’algorithme concernant ce client en particulier.



J’affiche les 12 premières caractéristiques les plus importantes selon leur ordre de grandeur en valeur absolu.

Une valeur positive signifie que la caractéristique augmente le score du client, et est donc favorable pour lui.

A l’inverse, une valeur négative indique que la caractéristique influence son score vers le bas.

# Les limites et les améliorations possibles

## Choix du pré-traitement des données

Le pré-traitement des données, comme l’imputation, la conversion des données catégorielles en données numérique a été réalisé pour être compatible avec l’ensemble des algorithmes.

Le choix final s’est porté sur l’algorithme LGBMClassifier.  
Cette algorithme supporte les données catégorielles et les données manquante.

Nous aurions peut-être pu améliorer les résultats de l’algorithmes en réalisant un pré-traitement spécifique à cet algorithme.

## Choix de la mesure :

Une amélioration possible aurait été de réaliser la sélection des algorithmes   
non pas sur la mesure technique ROC AUC mais directement sur la mesure métier.

Cette approche n’a pas été retenu car plus complexe à mettre en œuvre.

J’ai considéré l’approche par la mesure technique cohérente compte tenu de   
notre problématique avec un bon ration temps d’implémentation/qualité des résultats obtenus.

## Choix des hyperparamètres à optimiser

Par souci technique, la recherche des meilleurs hyperparamètres ont été réalisés sur un nombre restreint d’hyperparamètres et sur un spectre restreint pour chaque hyperparamètre testé.

Avec plus de temps et de puissance de calcul, il est possible d’améliorer les scores des algorithmes de cette façon.

## Pour le Dashboard – Possibilité de modifier les valeurs du client et recalculer son score

Il aurait été intéressant, pour un client, de calculer son score en modifiant une ou plusieurs de ses valeurs d’entrées actuelles.

Par exemple, qu’elle impact sur le score du client aurait une légère augmentation de son salaire ?

Pour implémenter cela, il aurait fallu pouvoir faire la transformation inverse des données et pouvoir obtenir les valeurs initiales du client et non les valeurs transformées comme c’est le cas ici.

Il suffit ensuite de réappliquer la transformation des données et injecter ce « nouveau client » dans le modèle.