|  |
| --- |
| OpenClassrooms |
| Implémentez un modèle de scoring |
| Projet 7 du parcours data scientist |

|  |
| --- |
| Fayez Hadji  8-23-2022 |

**Sommaire**

**I.Introduction …………………………………………………………………….. 5**

**II.Rappel de l’objectif et démarche générale …………………….….………..….. 5**

**III.La méthodologie d'entraînement du modèle ………………………………..….... 6**

**III.1.Précautions prises pour évaluer correctement les algorithmes …...… 7**

**III.2.Critère d’évaluation des algorithmes ……………………...……… 7**

**III.3.Pré-Sélection des modèles ……………………………………….….… 7**

**III.4.Sélection Finale de l’algorithme …………………………….………. 8**

**IV.La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation ……... 8**

**IV.1.La métrique technique ……………………………………………...8**

**IV.2.La métrique métier …………………………………………..………… 9**

**V.L’interprétabilité du modèle …………………………………………….………. 10**

**V.1.Interprétation Globale …………………………………..……………….... 11**

**V.2.Interprétation Locale …………………………………………………..… 11**

**VI.Les limites et les améliorations possibles …………………………….………. 11**

**VI.1.Choix du pré-traitement des données ……………………………………. 11**

**VI.2.Choix de la mesure …………………………………………………….. 12**

**VI.3.Choix des hyperparamètres à optimiser ……………………...……… 12**

**VI.4.Le tableau de bord …………………………………………….………. 12**

**VII.Annexe ……………………………………………………………………………. 13**

**I.Introduction**

Dans cette note méthodologique nous allons passer en revue les différents points traités dans le projet 7 du parcours data *scientist* intitulé : Implémentez un modèle de *scoring*. Ainsi, les points abordés sont :

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

**II.Rappel de l’objectif et démarche générale**

L’objectif se résume en grande deux étapes. La première était de construire un modèle de notation (*scoring*) qui puisse donner une prédiction sur la probabilité de faillite d'un client de façon automatique. De ce fait, la démarche était de déterminer:

1.Quel algorithme est le plus adapté pour remplir cette tâche ;

2.Les meilleurs hyperparamètres à utiliser pour cet algorithme.

L’un des points les plus importants de la première étape du projet était d’évaluer un ensemble de huit algorithmes avec des paramètres prédéfinis. L'idée était d’être en mesure d’éliminer les algorithmes qui ont montré dès le début une inadaptation à la problématique posée.

Dans un second temps, nous nous sommes intéressés aux algorithmes non éliminés et nous sommes allés plus loin dans leur sélection en recherchant pour chacun d'eux, leurs meilleurs hyperparamètres avec ‘*gridSearchCV*’.

Enfin, la deuxième étape était de construire un tableau de bord (*dashboard*) interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.

**III.La méthodologie d'entraînement du modèle**

Dans ce projet, nous faisons face à un problème de classification binaires (0 ou 1). De plus, les classes sont déséquilibrés. Pour évaluer correctement l’entrainement des modèles, il fallait suivre une méthodologie adaptée et précise. L’objectif était, donc, 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 d’un effet aléatoire qui peut intervenir à différentes étapes de l’évaluation des algorithmes et qui peut avoir un impact sur le test d’un algorithme et par conséquences sa capacité à être sélectionné.

De là, les étapes qui définissent l’évaluation d’un algorithme sont, alors, les suivantes :

1.Génération des données d’entrainement et de test. Ces mêmes données seront utilisées par l’ensemble des algorithmes testés entre eux :

A.Génération du ‘*Train\_set*’ et du ‘*Test\_set*’ ;

B.Génération de l’estimation de la pondération des classes car le jeu de données utilisé durant ce projet est déséquilibré (**Figure 1**) ;

C.Génération des données de l’étape précédente pour différents pourcentages des données disponibles auquel cas l’entrainement serait trop long soit :

a.100% des données disponibles ;

b.10% des données disponibles ;

c.1% des données disponibles ;

d.0.1% des données disponibles.

D.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.

**III.1.Précautions prises pour évaluer correctement les algorithmes**

Comme indiqué au chapitre précèdent, les algorithmes ont tous été testés scrupuleusement dans 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 due aux tirages aléatoires présents à chacune de ces étapes et qui peuvent avantager ou désavantager les algorithmes. Cela, nous permet, donc, d’être absolument sûres que les résultats obtenus pour chaque algorithme le sont dans les mêmes conditions et qu’en définitive, nous pouvons comparer et sélectionner en toute quiétude les algorithmes entre eux sur la base de ces résultats.

**III.2.Critère d’évaluation des algorithmes**

Les algorithmes étaient tous évalués 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 étaient testés sur le ‘*Test\_set*’, c’est sur les résultats de ‘*Train\_set*’ que les décisions ont été prises. Il est important de noter l’utilisation de l’algorithme ‘*Dummy Classifier*’ comme point de repère. Cet 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***’(métrique qui sera revu plus en détail au chapitre V).

**III.3.Pré-Sélection des modèles**

Dès le début de sélection, l’algorithme ‘*SVC*’ était disqualifié à cause d’un temps d’entrainement excessivement long (**Figure 2**). Si l’on se concentre sur les scores obtenus, les trois meilleurs algorithmes qui répondent aux mieux à notre problématique sont dans l’ordre:

-‘*LGBMClassifier*’ ;

-‘*CatBoostClassifier*’ ;

-‘*XGBoost*’.

Parmi ces trois-là, ‘*CatBoostClassifier*’ semble le plus lent à entraîner suivi par ‘*XGBoost*’. L’algorithme ‘*LGBMClassifier*’ a l’avantage de prendre en charge l’apprentissage parallèle et l’utilisation des GPU. Les algorithmes ‘Régression Logistique’ et ‘*SVC*’ ont des scores très en retrait par rapport aux trois algorithmes cités ci-dessus.

**III.4.Sélection Finale de l’algorithme**

L’algorithme ‘*LGBMClassifier*’ est celui qui offre le plus de performance et est ainsi, retenu dans la résolution de notre problématique. ‘*LGBMClassifier*’ se distingue de ‘*CatBoostClassifier*’ aussi bien sur la qualité de ses prédictions que par sa vitesse d’entrainement (**Figure 3**). Même si leur temps de prédiction est très rapide pour l’un comme pour l’autre, la sélection se fait ici sur la base de l’étude de leurs scores (**Figure 4**).

‘*LGBMClassifier*’ est également plus stable que son homologue si on compare le delta des scores par *fold*. Il a un delta d’environ 0.012 points entre ses deux *folds* extrêmes quand ‘*CatBoostClassifier*’ affiche une différence d’environ 0.02.

Même si cela ne semble pas être très intuitive pour départager les deux algorithmes, ‘*LGBMClassifier*’ est préféré lors de l’analyse des courbes de densité de probabilité en fonction de la variable ‘*TARGET*’. La courbe de ‘*LGBMClassifier*’ est plus propre, mieux séparé que pour celle de ‘*CatBoostClassifier*’ (**Figure 5**). En définitive, le modèle retenu et qui nous semble le plus performant pour ce type de situation est l’algorithme ‘*LGBMClassifier*’.

**IV.La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation**

Pour répondre au mieux à la problématique du projet, deux métriques ont été utilisés successivement :

-Une métrique technique défini par la courbe ‘*ROC AUC*’ ;

-Une métrique métier qui prend en compte la définition d’un seuil à partir d’une fonction coût à minimiser.

**IV.1.La métrique technique**

Cette métrique est adaptée pour les cas où les classes sont déséquilibré comme ici. Elle permet une meilleure évaluation d’un modèle que la précision. 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*’ ou parfois appelé ‘*AUROC*’).

La courbe ‘*ROC*’ pour *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 (**Figure 6**).

Cette métrique est comprise entre 0 et 1, un bon modèle obtenant un score élevé. Un modèle qui se contente de deviner au hasard aura un ‘*ROC AUC*’ de 0,5. Enfin, lorsque nous mesurons un algorithme de classification selon la métrique ‘*ROC AUC*’, nous ne générons pas des prédictions 0 ou 1, mais plutôt une probabilité entre 0 et 1.

Cette grandeur mathématique est, donc, adaptée pour pouvoir déterminer la performance des modèles et en déterminer les plus performant dans notre problématique.

**IV.2.La métrique métier**

Une fois notre meilleur algorithme trouvé et optimisé par la métrique ‘*ROC AUC*’, il nous faut, par la suite, pouvoir l’adapté pour répondre au mieux à notre objectif.

La notion de ‘*ROC AUC*’ nous donne une probabilité qu’un échantillon appartienne à une classe. Ainsi et par défaut, le seuil de décision est fixé à 0.5. Par exemple, si le modèle prédit qu’un client à une probabilité de 0.6 d’appartenir à la classe des clients qui honorent 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. En effet, 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.

Nous pouvons 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.

Posons le calcul suivant :

FN (Faux Négatif c’est-à-dire Faux Bon client) : prêt à tort qui correspond à une perte du capital ou partie du capital et donc une perte de beaucoup d'argent.

FP (Faux Positif c’est-à-dire Faux Mauvais client) : refus à tort de prêt qui correspond à une perte de Marge et donc un manque à gagner.

Hypothèse : imaginons que nous ayons un capital valant 100 unités.

-Un FN avec une perte de 50% du capital prêté en moyenne revient à une perte de 50 unités.

-Un FP avec un manque à gagner de 1%/an sur en moyenne 10ans revient à :

Le capital va de 100 unités à 0 unités soit une moyenne de 50 unités et donc 10% de 50 unités représente une perte de 5%

Au final, le rapport de coût entre un FN et un FP est le suivant :

-Les FN coûtent 10 fois plus chers que les FP (1 x FN = 10FP)

Ce qui nous mène aux Objectifs suivants :

-Optimiser la Matrice de confusion en optimisant la somme de ces deux coûts.

-Résoudre : Min(10FN + FP)

-Tester tous les seuils et calculer cette fonction

Nous obtenons donc un seuil optimal fixé à 0.52 (**Figure 7**).

**V.L’interprétabilité du modèle**

Pouvoir interpréter un modèle nous permet de comprendre comment il a été amené à prendre ses décisions. Ainsi, il fallait implémenter l’interprétation des modèles dans le *dashboard*. De ce fait, le tableau de bord a été développé dans l’idée qu’il puisse être utilisé avec différent modèle. Pour cette raison, il était plus judicieux de ne pas utiliser les fonctions intégrées aux modèles lorsque cela était possible comme avec ‘*LGBMClassifier*’, mais d’utiliser plutôt une méthode plus généraliste avec ‘*SHAP*’.

Il est possible d’interpréter un modèle à deux niveaux : globalement puis localement.

**V.1.Interprétation Globale**

L’interprétation globale (**Figure 8**) 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 :

-Les scores obtenus auprès des autres agences de crédit ;

-Le genre ;

-L’âge ;

-L’ancienneté dans l'emploi ;

-Le montant du bien acheté ;

-Le montant des annuités.

**V.2.Interprétation Locale**

Les critères retenus peuvent être très variable d’un individu à l’autre. Imaginons qu’un client possède des valeurs situées dans la moyenne concernant les scores obtenus auprès des agences de crédit tel que 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.

De là, les 12 premières caractéristiques les plus importantes sont affichées selon leur ordre de grandeur en valeur absolu (**Figure 9**). 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.

**VI.Les limites et les améliorations possibles**

**VI.1.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*’. Cet 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.

**VI.2.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. Il est préférable de considérer l’approche par la mesure technique cohérente compte tenu de notre problématique avec un bon rapport temps d’implémentation/qualité des résultats obtenus.

**VI.3.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.

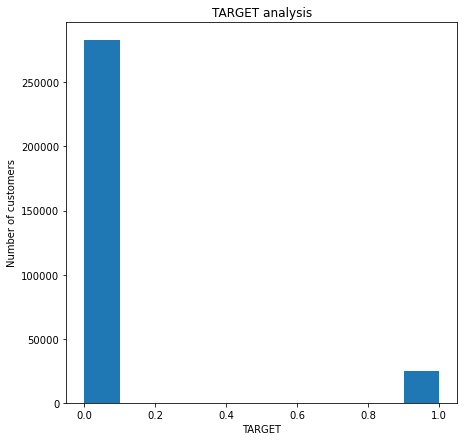
**VI.4.Le tableau de bord**

Il y a la possibilité de modifier les valeurs du client et recalculer son score. Ainsi, 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 aurait une légère augmentation de salaire sur le score du client ?

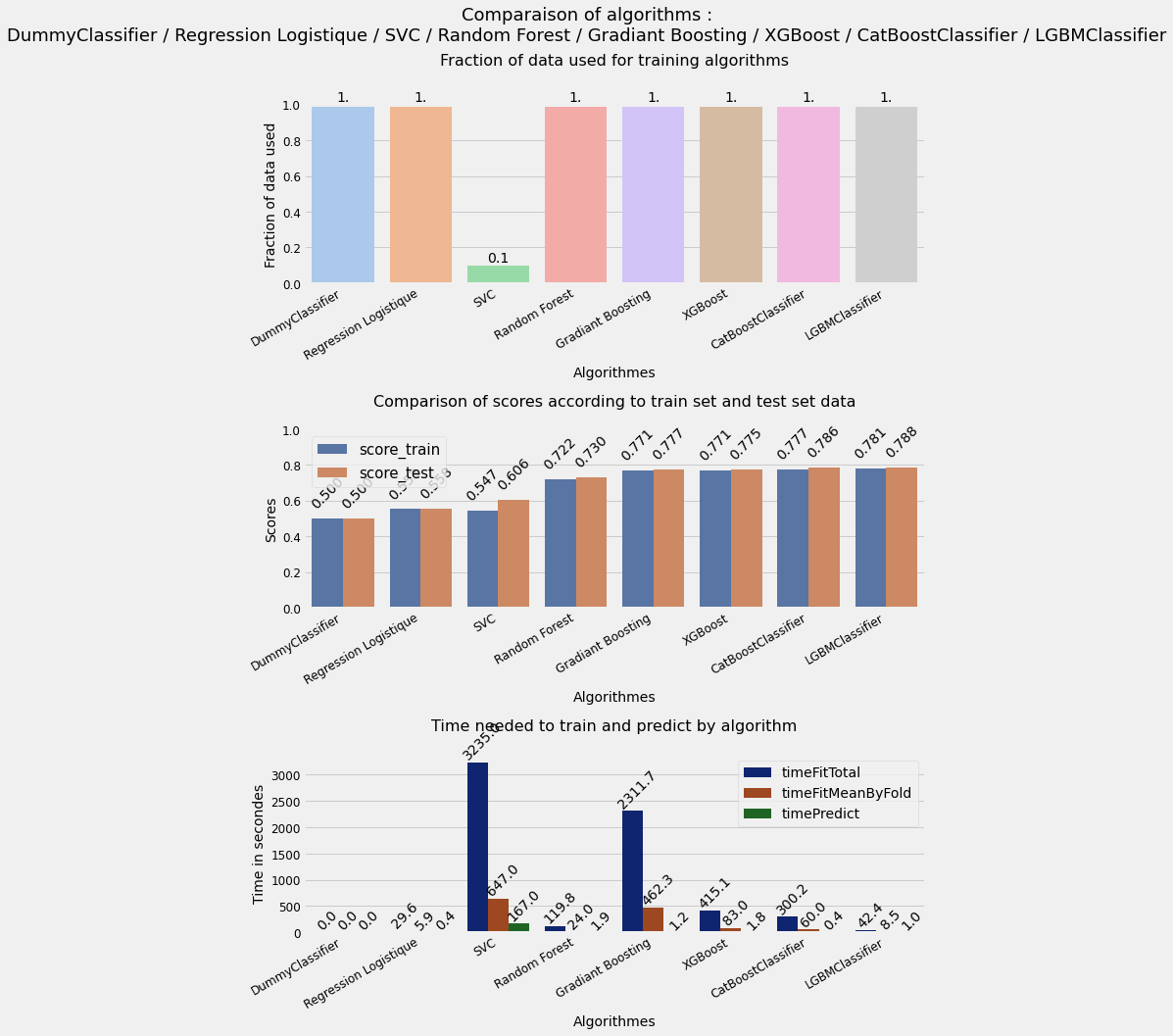
Pour implémenter cela, il aurait fallu pouvoir faire la transformation inverse des données et pouvoir obtenir les valeurs initiales du client et non les valeurs transformées comme c’est le cas ici. Il suffit ensuite de réappliquer la transformation des données et injecter ce « nouveau client » dans le modèle.

**VII.Annexe**

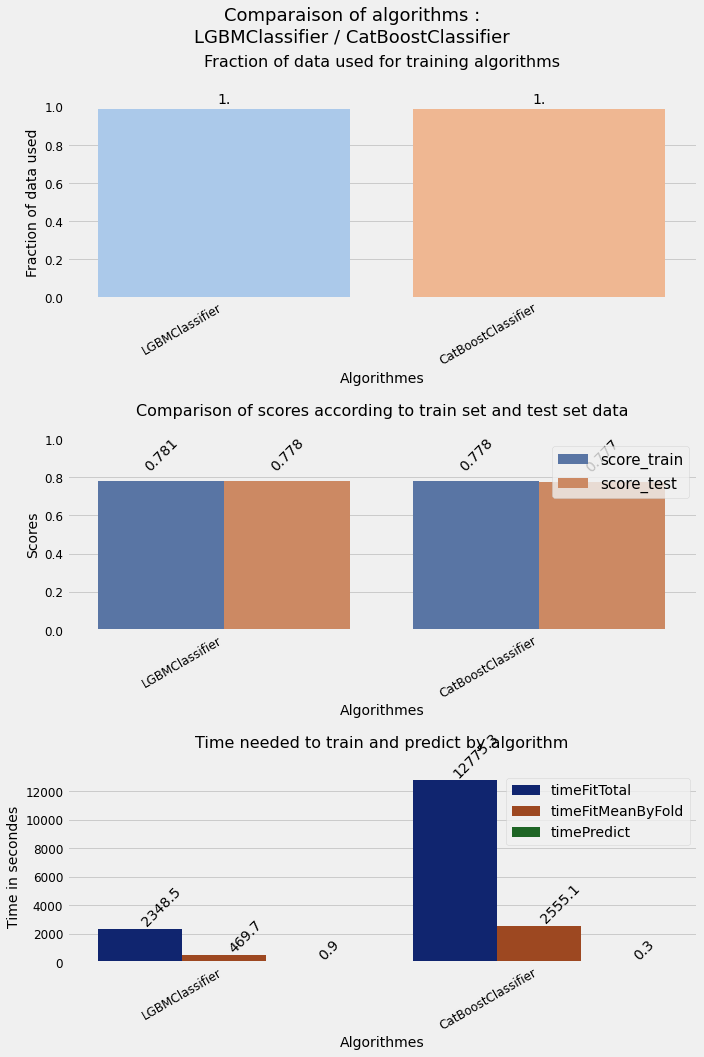
Cette partie de la présente note, regroupe l’ensemble des figures utilisées pour appuyer les arguments cités précédemment.



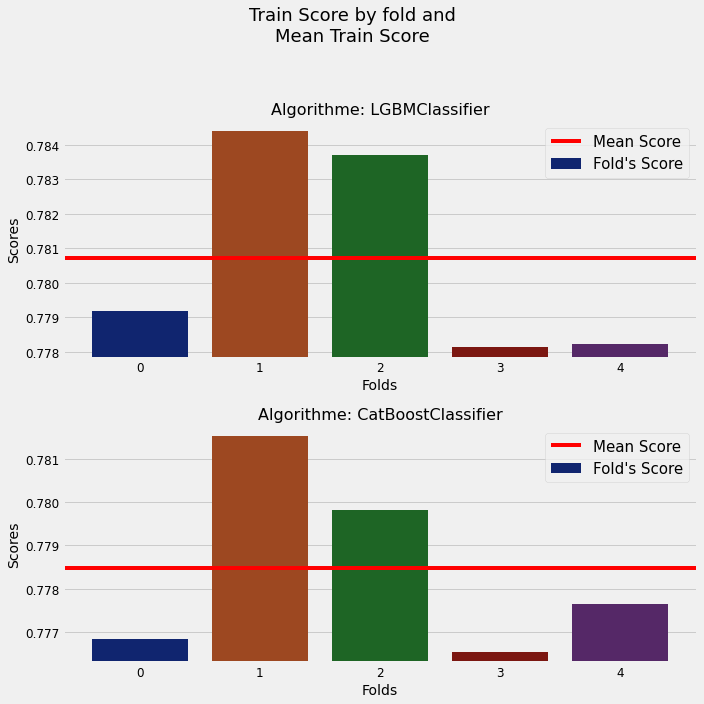
**Figure 1** : représentation du déséquilibre en termes de quantité de données entre les deux classes (0 et 1)



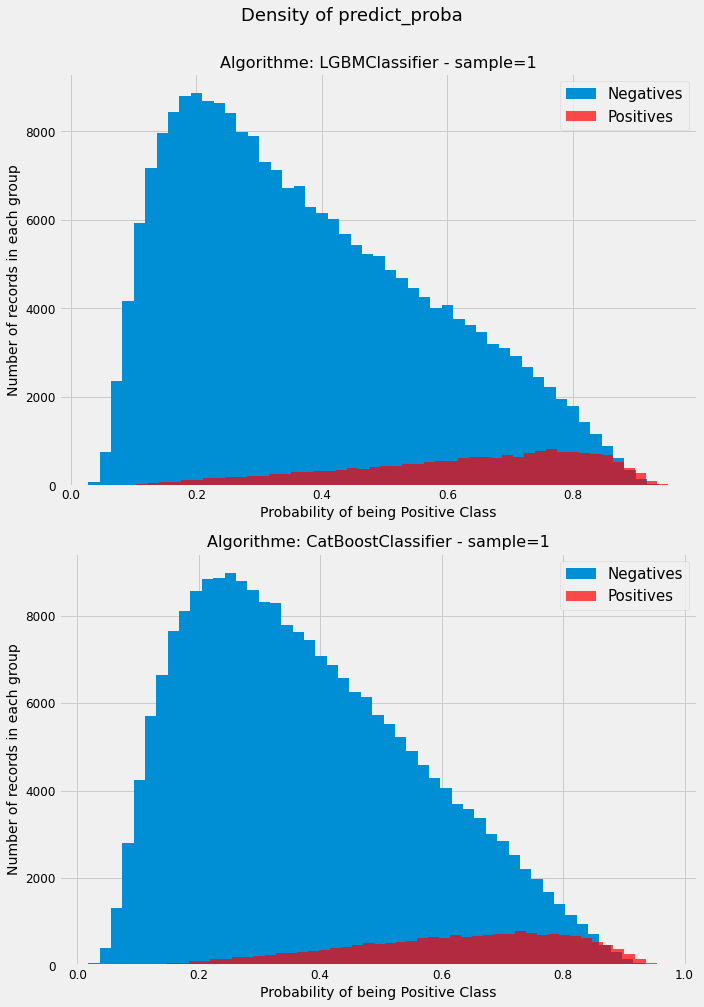
**Figure 2**: représentation de la comparaison des performances des différents algorithmes. Cette comparaison nous permet de choisir l’algorithme le plus adapté à notre situation.



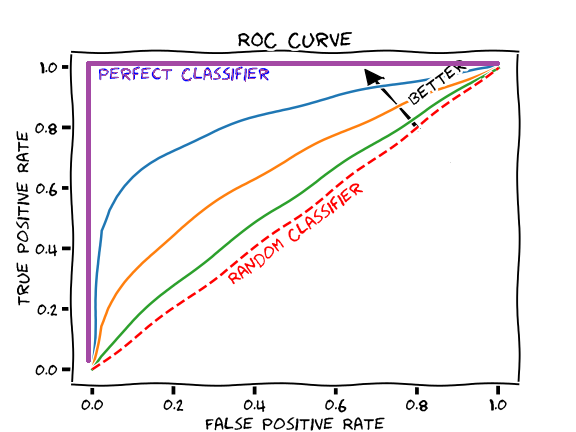
**Figure 3**: représentation de la comparaison des performances des deux algorithmes ‘*LGBMClassifier*’ et ‘*CatBoosClassifier*’.



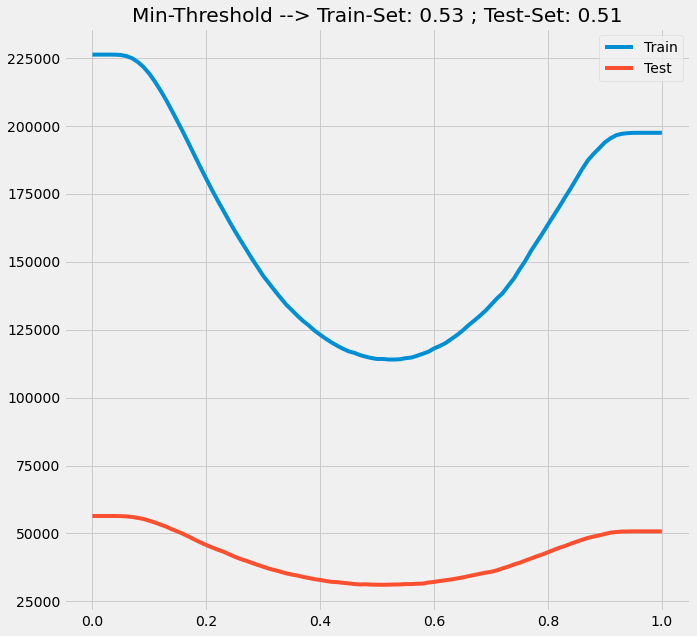
**Figure 4**: représentation du score lors de l’apprentissage pour chacun des deux algorithmes ‘*LGBMClassifier*’ et ‘*CatBoosClassifier*’.



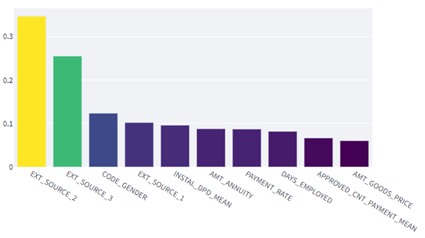
**Figure 5**: représentation de la courbe de densité de prédiction des deux algorithmes ‘*LGBMClassifier*’ et ‘*CatBoosClassifier*’.



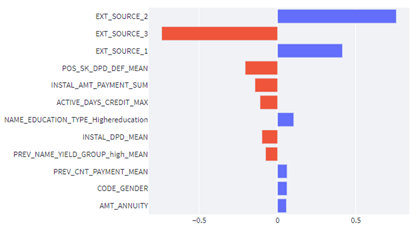
**Figure 6** : Illustration schématique de la notion mathématique de la courbe ‘*ROC AUC*’.



**Figure 7** : représentation du seuil défini pour la métrique métier. Le seuil est établi à partir d’une fonction coût à minimiser.



**Figure 8** : représentation graphique de l’interprétation globale des *features importances* qui permettent de savoir quelles sont les caractéristiques qui ont le plus influencés le model dans le calcul du score.



**Figure 9** : représentation graphique de l’interprétation locale des *features importance* qui permettent de savoir quelles sont les caractéristiques qui ont le plus influencés le model dans le calcul du score pour un individu en particulier. Il est important de noter que ce score est indépendant des autres individus.