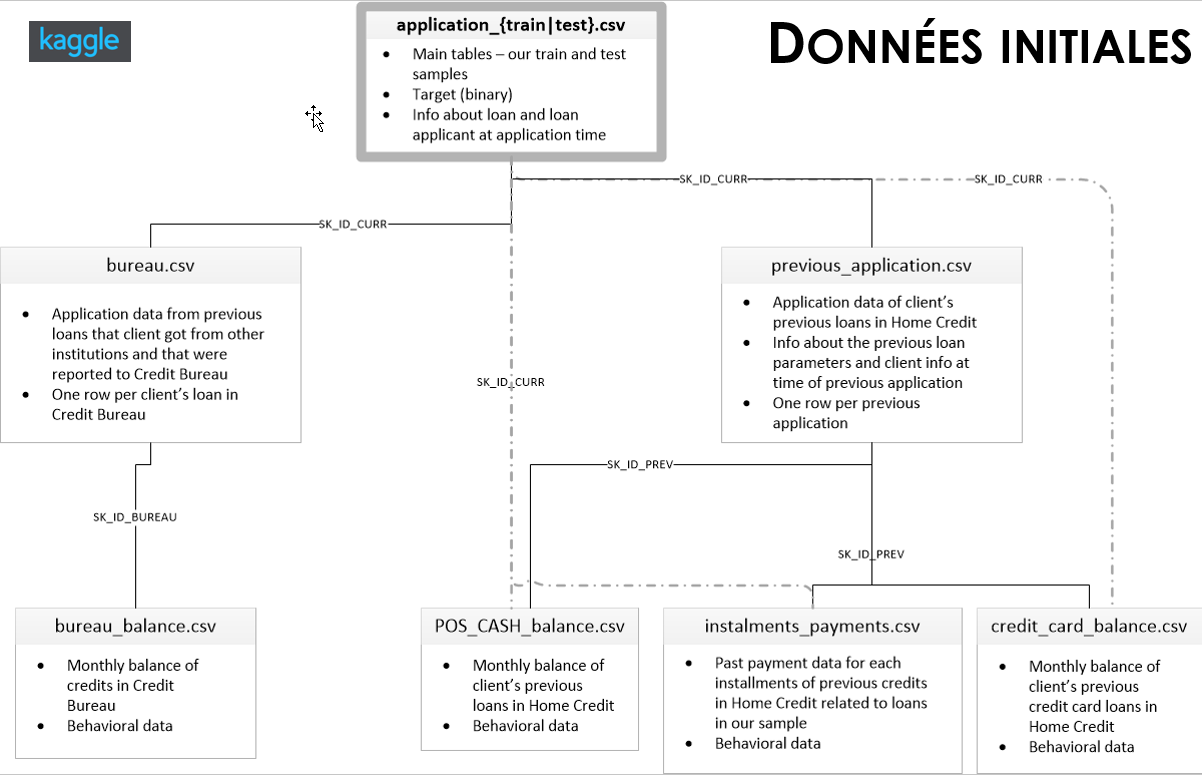
|  |
| --- |
| **PROJET 7 :** NOTE MÉTHODOLOGIQUE |

L'ENTREPRISE "PRÊT À DÉPENSER" propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt. L’entreprise souhaite développer un modèle de Scoring de la probabilité de défaut de paiement du client pour étayer la décision d'accorder ou non un prêt Pour réaliser cette étude on s'appuie sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.). On doit fournir un outil de compréhension et d'explication du résultat car les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d’octroi de crédit .

1. **MÉTHODOLOGIE D’ENTRAÎNEMENT**

Le but est d’expliquer comment on a préparé les données et quels sont les modèles short listés

Le jeu de données initial :



a été nettoyé des lignes et colonnes pour lesquelles le taux de remplissage était inférieur à 80%.

Dans le cas d'une prédiction supervisée, l'entraînement d'un modèle nécessite une étape de division du jeu de données entre le prétraitement des données et la modélisation. Cette division permet de créer un échantillon d'apprentissage (train) qui représente **80% des données et un échantillon test qui représente 20% des données.**

Dans le jeu de données initial (train), nous avons séparé :

* X : matrice des variables
* y : vecteur des cibles

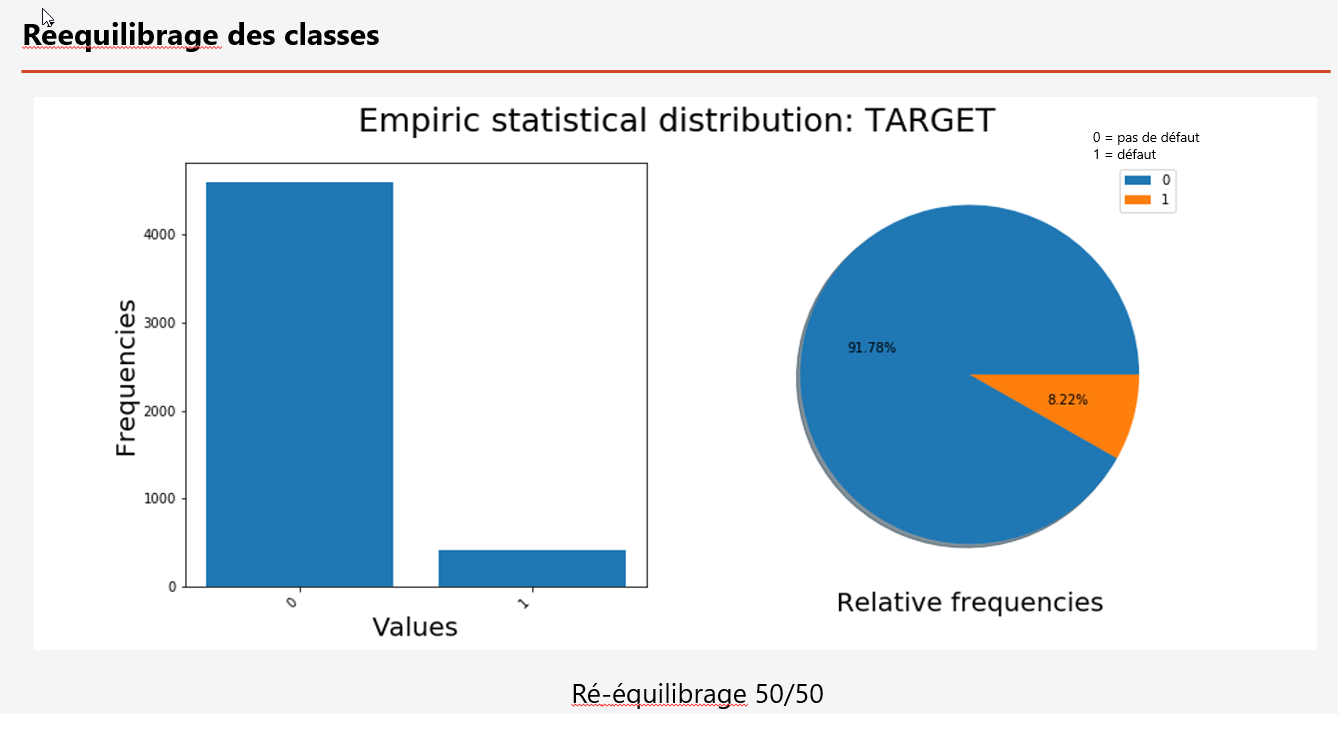
La cible ‘Target’ est = 0 pour ceux qui ne font pas défaut, et 1 pour ceux qui font défaut

Nous avons ensuite séparé en deux :

* X\_fit, y\_fit : jeu de données servant à la sélection du modèle et des hyperparamètres
* X\_eval, y\_eval : jeu de données servant à l’évaluation finale du modèle

Les valeurs manquantes résiduelles ont été imputées avec **simpleimputer « par la médiane »**.

Pour faciliter l’apprentissage du modèle, nous avons procédé à l’équilibrage des classes par sur-échantillonnage (*oversampling*) de la classe minoritaire par utilisation de la librairie **SMOTE** (**Synthetic Minority Oversampling Technique).**



**Sampling** : Nous avons prélevé un aléatoire échantillon de **5000 demandes de crédit** dans le jeu (X\_fit, y\_fit) pour accélérer la recherche d’optimisation des modèles et hyperparamètres.

**La liste de modèles utilisés :**

* Bayésien naïf :

Notre baseline. Point 0 sur lequel on va pouvoir comparer les autres modèles.

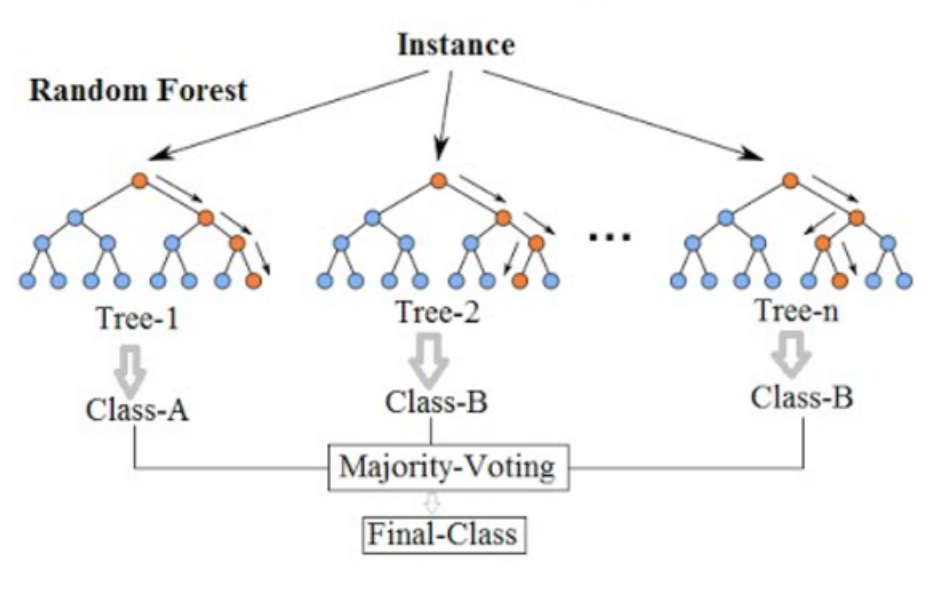
La classification naïve bayésienne est un type de classification bayésienne probabiliste simple basée sur le [théorème de Bayes](https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Bayes) avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des [classifieurs linéaires](https://fr.wikipedia.org/wiki/Classifieur_lin%C3%A9aire).

Un terme plus approprié pour le modèle probabiliste sous-jacent pourrait être « modèle à caractéristiques statistiquement indépendantes ».

En termes simples, un classifieur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. Un fruit peut être considéré comme une pomme s'il est rouge, arrondi, et fait une dizaine de centimètres. Même si ces caractéristiques sont liées dans la réalité, un classifieur bayésien naïf déterminera que le fruit est une pomme en considérant indépendamment ces caractéristiques de couleur, de forme et de taille.

* Forêt aléatoire

Les forêts d'arbres décisionnels (ou forêts aléatoires de l'anglais random forest classifier) font partie des techniques d'[apprentissage automatique](https://fr.wikipedia.org/wiki/Apprentissage_automatique). Cet algorithme combine les concepts de sous-espaces aléatoires et de [bagging](https://fr.wikipedia.org/wiki/Bagging). L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.

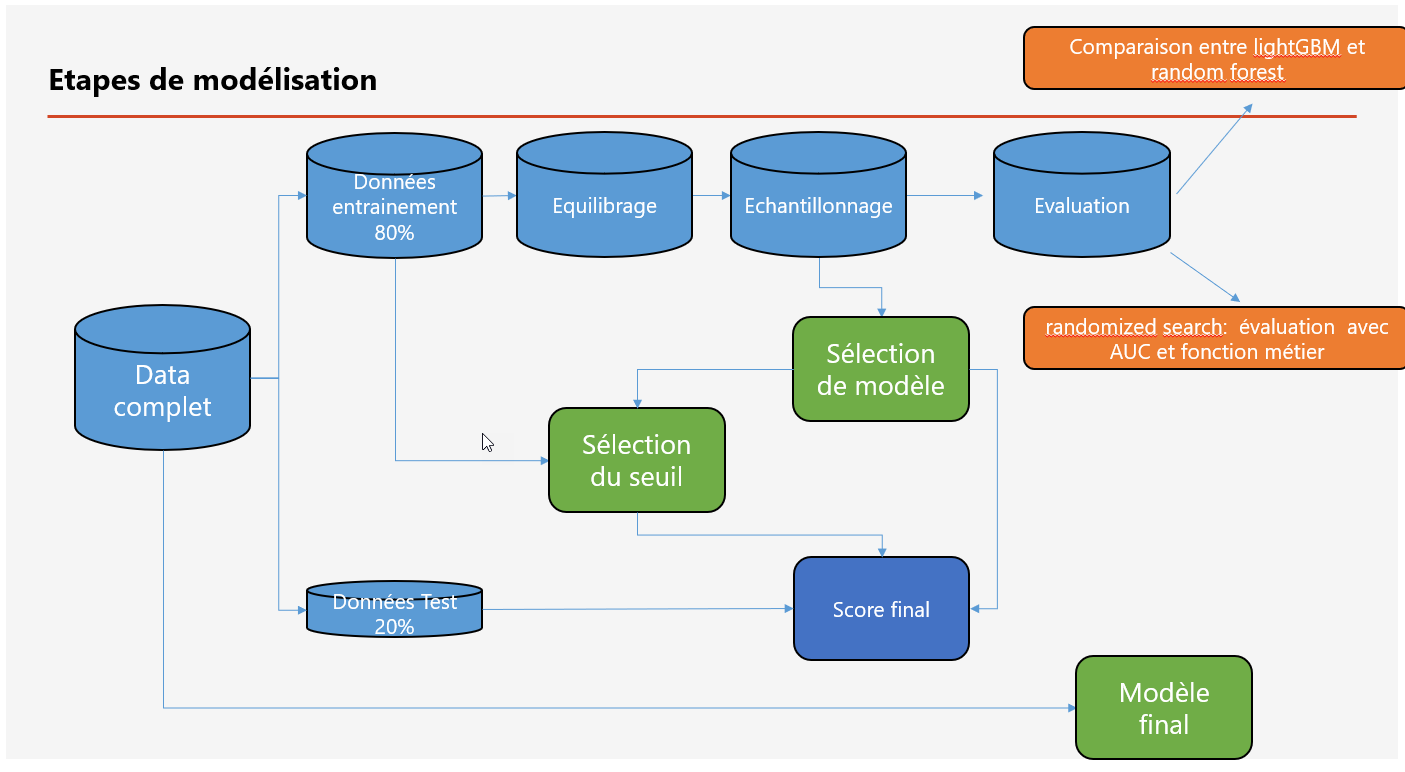


* LightGBM

Light Gradient Boosting Machine, libre et opensource, developpé par Microsoft. Basé sur un arbre de décision LightGBM adopte deux nouvelles techniques GOSS (Gradient-based One-Side Sampling) et EFB (Exclusive Feature Bundling). Avec GOSS, LightGBM peut entraîner chaque arbre avec seulement une petite fraction du jeu de données complet. Avec EFB, LightGBM gère beaucoup plus efficacement les fonctionnalités clairsemées de haute dimension. LightGBM prend également en charge l’entrainement distribué avec un faible coût ce qui réduit les besoins sur les GPU.

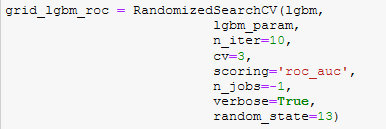
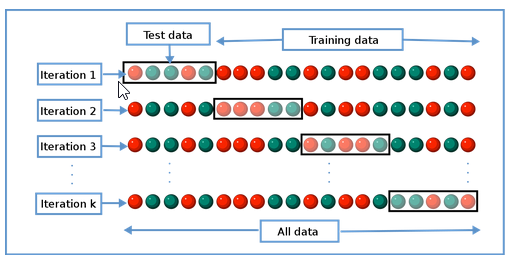
1. **FONCTION COÛT, ALGORITHME D’OPTIMISATION, MÉTRIQUE D’ÉVALUATION**

Le but est de prendre en compte le fait que les clients qui font défaut et qui sont considérés comme solvables font perdre beaucoup plus d’argent que de perdre les clients solvables qu’on a considéré comme non solvables



1. **Fonctions de pertes**

La sélection du modèle est faite en se basant sur l’évaluation, par validation croisée à 3 plis (folds), à l’aide de l’AUC. 3 plis est un choix fournissant assez de précision en évitant un processus de calcul trop long.

1. **Algorithmes d’optimisation**

**RandomizedSearchCV** de sklearn préféré à gridsearch pour sa plus grande rapidité, les algorithmes sélectionnés sont déjà assez performants.

1. **Métrique d’évaluation**

Première phase d’évaluation pour comparer les algorithmes en utilisant un scoring roc\_auc, puis on choisit le meilleur algorithme en fonction de :

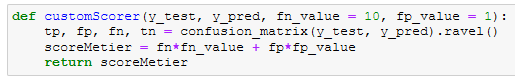
* **Rmse** : l'erreur quadratique moyenne est une mesure fréquemment utilisée des différences entre les valeurs (valeurs d'échantillon ou de population) prédites par un modèle ou un estimateur et les valeurs observées Écart quadratique moyen
* **Roc** : la courbe ROC permet de décrire la performance d’un modèle à travers deux indicateurs : la sensitivity et la specificity. L’aire sous cette courbe, nommée AUC / ROC *(area under curve / receiver operating characteristic curve)*, mesure de façon globale la performance d’un modèle de classification.
* **F1 score** : Le F1-score est une métrique pour évaluer la performance des modèles de classification à 2 classes ou plus. Il est particulièrement utilisé pour les problèmes utilisant des données déséquilibrées comme la détection de fraudes ou la prédiction d’incidents graves. Le F1-score permet de résumer les valeurs de la precision et du recall en une seule métrique.



L’impact pour l’entreprise d’un faux négatif et ceux d’un faux positif ne sont pas les mêmes.

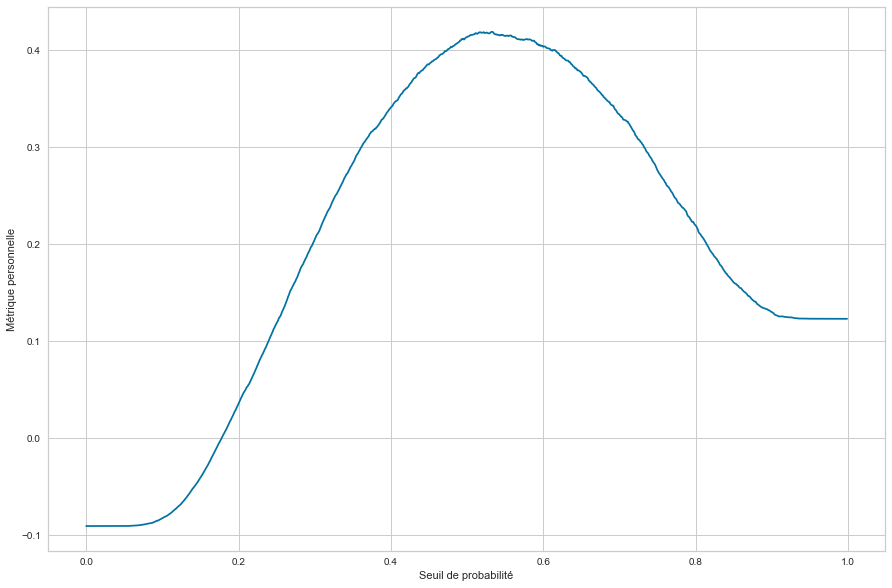
* FN, erreur de 1ère espèce :  client à qui l’on refuse un prêt, mais qui l’aurait honoré
* FP, erreur de 2ème espèce : client à qui l’on accorde un prêt, mais qui fait défaut

Nous avons donc construit une métrique d’évaluation « métier » qui surpondère le poids des faux négatifs, ceux à qui on donne un crédit mais qui feraient défaut. Ces derniers coûteraient **10X plus cher** que de perdre des clients qui ne feraient pas défaut mais à qui on ne donnerait pas de crédit.

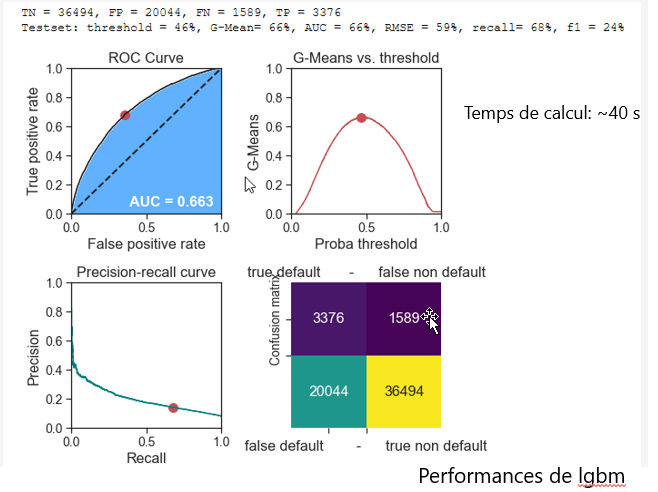


**Seuils :**

La métrique sur le revenu net personnalisé doit être maximisée pour que la banque gagne le plus d'argent et ainsi permettre de déterminer à partir de quel seuil de probabilité un client est considéré comme défaillant ou non. Généralement le seuil de probabilité choisi par les modèles pour classer les individus est de 0,5, mais il ne s'agit pas du seuil optimal pour que la banque maximise ses revenus.



Les résultats montrent un seuil de 0,46 ce qui signifie qu'un client avec une prédiction au-dessus de cette valeur sera catégorisé comme défaillant et un client avec une prédiction en dessous de cette valeur sera considéré comme non défaillant.



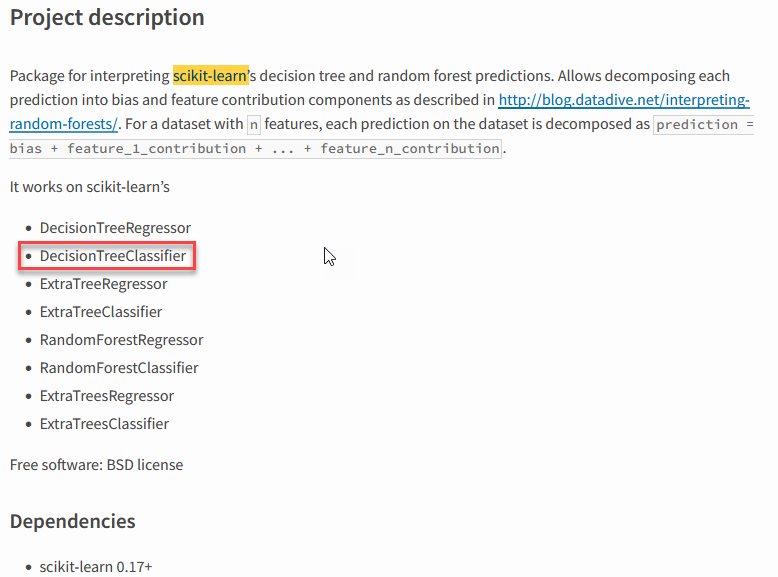
1. **INTERPRÉTABILITÉ DU MODÈLE**

Le but est de permettre au chargé de clientèle d’expliquer pourquoi un prêt à été refusé ou accepté

Lightgbm a été choisi pour ses meilleures performances en particulier sur les faux négatifs. Il est également nettement plus rapide que randomforest (40 secondes vs 290 secondes).

Néanmoins il est important de trouver un juste compromis en machine learning entre un modèle performant versus un modèle interprétable.

Pour répondre au besoin d’interprétabilité et au besoin de rapidité de calcul pour l’API, nous avons donc privilégié la méthode du « *surrogate model* » (modèle de substitution) qui consiste à entraîner un modèle interprétable sur les prédictions du modèle à interpréter (lightgbm choisi). Ce modèle (arbre de décision) est alors interprétable : globalement, avec les « *features importances* », et localement en utilisant le package treeinterpreter de la librairie scikit-learn.



1. **LIMITES & AMÉLIORATIONS**
2. **Modèles et hyperparamètres**
3. Un premier axe d’amélioration est d’intégrer dans l’espace de recherche d’autres familles de modèles et notamment :

* Boosting : XGBoost, CatBoost…
* Réseaux neuronaux : fully-connected avec Keras/TensorFlow

1. On peut aussi essayer d’intégrer davantage d’étapes de pré-traitement (pour, par exemple comparer les méthodes d’équilibrage des classes, ou les méthodes d’imputation).
2. **Interprétabilité**

La prédiction de solvabilité repose sur le modèle entier.  
L’interprétabilité repose sur le modèle surrogate: treeinterpreter sur arbre de décision. Il faut alors faire une mise à jour régulière du modèle surrogate, au fur et à mesure que le nombre de nouveaux clients augmente. Ceci afin que les features globales de la population ne diffèrent pas du modèle complet basé sur lightgbm.