**Note**

**Méthodologique**

**Projet 7 : Dossier Versioning**

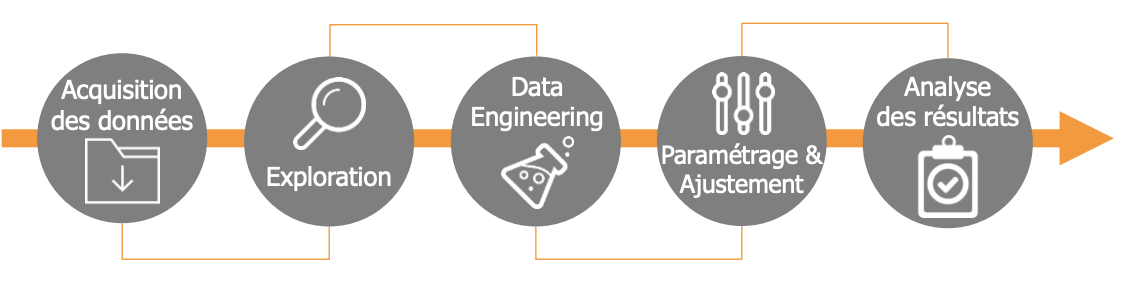
<https://github.com/SCarlevato/p7scoringopenclassrooms>

**Sylvain Carlevato Septembre 2022**

**OpenClassrooms – Centrale Supélec**

**Synthèse :**

Les algorithmes de *Machine Learning* divergent dans leurs méthodes de modélisation mais se basent sur le même principe. Le projet a été traité à l’aide de 3 algorithmes de *gradient boosting* (LightGBM, XGBoost et CatBoost) utilisés en raison de leur simplicité d’interprétation pour certains mais aussi par expérience, face à leurs performances passées. Un projet de *Machine Learning* se construit en plusieurs étapes successives. La mise en oeuvre a été opérée selon une démarche fréquente qui se construit de la manière suivante :



La première étape consiste à récupérer des données cohérentes afin de former la base sur laquelle l’algorithme de *Machine Learning* va apprendre. Des données financières, comportementales, etc, relatives à différents clients sont récupérées sur Kaggle. Avant d’utiliser le jeu de données à des fins de prédiction, le retraitement des données est une tâche importante qui intervient naturellement à la suite de la phase exploratoire. En effet, les données brutes sont souvent bruitées, peu fiables et incomplètes. Leur utilisation peut générer des résultats trompeurs d’où l’intérêt de procéder à certaines modifications ou à un enrichissement de la base. Un kernel Kaggle est utilisé pour faciliter l’exploration des données nécessaires à l’élaboration du *modèle de scoring*. La mise en œuvre de statistiques descriptives découle ensuite et permet de comprendre et d’avoir une vue d’ensemble des données grâce à de la visualisation graphique.

L’objectif final d’un projet de *Machine Learning* est de construire un modèle de prédiction. C’est à partir de la base retraitée que les algorithmes abordés précédemment sont paramétrés puis implémentés de manière optimale.

Pour analyser les résultats et juger de l’efficacité d’un algorithme, donc du modèle qui en découle, il est nécessaire de calculer l’erreur des apprentissages effectués par une métrique d’évaluation, à savoir l’aire sous la courbe *ROC (AUC)*. Enfin, une fonction coût a été implémentée afin de pénaliser l’impact des erreurs sur la décision d’octroi de crédit.

**Table des Matières :**

**Synthèse :**

**1 - Prétraitement des Données :**

**2 - Modélisation de la Probabilité de Défaut de Paiement du**

**Client :**

**2.1 - Notions de Base en Machine Learning**

**2.2 - Algorithmes utilisés pour construire le Modèle de**

**Scoring :**

**2.3 - Fonction Coût Métier :**

**2.4 - Algorithme d’Optimisation HyperOpt :**

**3 - Interprétabilité du Modèle :**

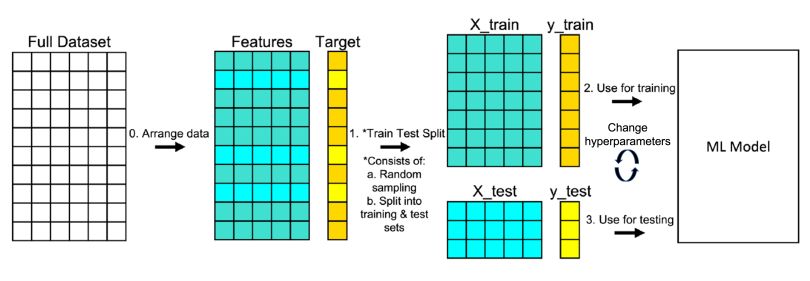
**4 - Limites et Améliorations :**

**1 - Prétraitement des Données :**

L’entraînement d’un modèle nécessite une étape transitoire entre le preprocessing et la modélisation. Ce modèle de prédiction peut être représenté par une fonction qui prend des données en entrée et une décision en sortie. Dans le cas d’*apprentissage supervisé* où nous souhaitons expliquer une variable de sortie, l’échantillon est classiquement subdivisé en 2 parties :

- L’échantillon d’apprentissage correspond à l’échantillon principal où sont appliquées les méthodes, sur lesquelles les algorithmes apprennent. Il sert à ajuster le modèle, et représente dans notre cas 70% des données.

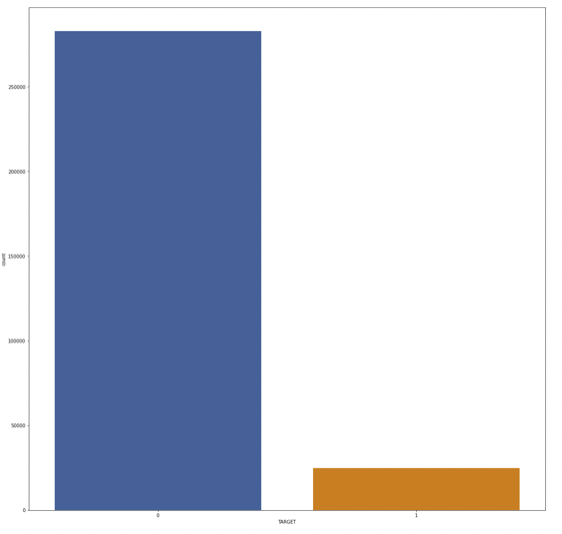
- L’échantillon de test est utilisé pour évaluer le modèle optimal (au sens du résultat de la validation croisée). Il n’a donc pas été utilisé pour l’apprentissage, ce qui fait que le modèle sélectionné est indépendant de cet échantillon test. L’idée est de simuler la réception de nouvelles données (entrée) afin de prédire la variable à expliquer à partir du modèle final et de les comparer aux « vraies » valeurs de la variable à expliquer. Cet échantillon permet d’évaluer objectivement l’erreur réelle. L’échantillon test représente donc 30% des données.



Ce découpage de données dans un projet de *Machine Learning* est une étape très importante qu’il ne faut pas négliger, il existe en effet un risque de surévaluer le modèle (over-fitting) ou tout simplement le contraire (under-fitting). En effet, par nature, un modèle va coller (mais pas trop) à ses données d'entraînement.

*train\_test\_split()* de Scikit-Learn permet d’obtenir facilement une répartition aléatoire des individus en base d’apprentissage et de test. Le problème de *Machine Learning* à résoudre est un problème de *classification binaire*, il faut s’assurer que chaque côté contient une proportion raisonnable des classes 0 et 1.

L’Analyse Exploratoire des Données (EDA) a permis d’identifier un fort déséquilibre entre la précision trouvée entre la classe 0 et la classe 1. L'échantillon de travail est déséquilibré, 92% en classe 0 vs 8% en classe 1. Le traitement *Oversampling* (ou suréchantillonnage en français) permet d’ajuster la distribution de classe de manière à avoir une répartition plus égalitaire.



Pour les techniques de suréchantillonnage, **SMOTE** (Synthetic Minority Oversampling Technique) est considéré comme l'un des algorithmes d'échantillonnage de données les plus populaires et les plus influents dans le *Machine Learning* et l'exploration de données. Avec **SMOTE**, la classe minoritaire est sur-échantillonnée en créant des exemples «synthétiques» plutôt qu'en sur-échantillonnant avec remplacement.

Une image contenant texte

Description générée automatiquement

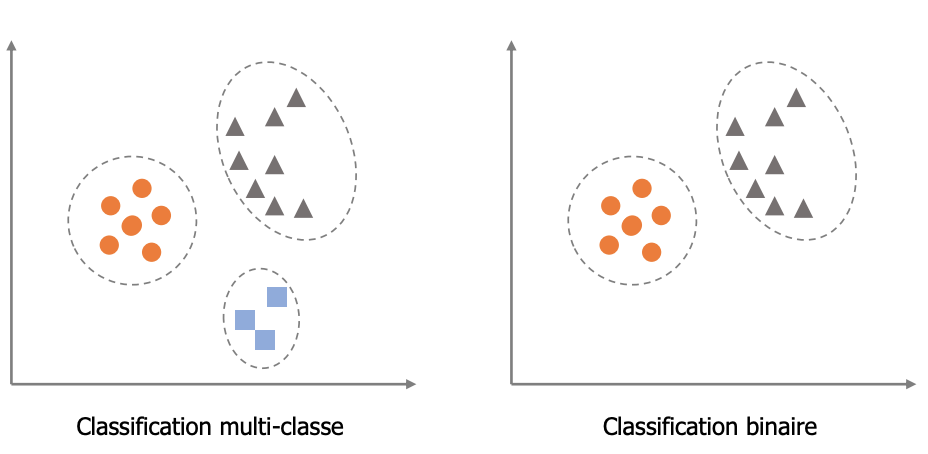
Une image contenant texte

Description générée automatiquement

**2 - Modélisation de la Probabilité de Défaut de Paiement du Client :**

**2.1 - Notions de Base en Machine Learning :**

La modélisation du scoring qui donnera une prédiction sur la probabilité de faillite d'un client a été implémentée par *classification*. La *classification* consiste à identifier les classes d'appartenance de nouveaux objets à partir d'exemples antérieurs connus. Dans le contexte métier du projet, la classification binaire est représentée par une variable de sortie à deux classes, à savoir acceptation du crédit ou refus du crédit. En opposition à un problème multi-classes, c’est-à-dire une variable cible peut-être représentée par plusieurs classes.



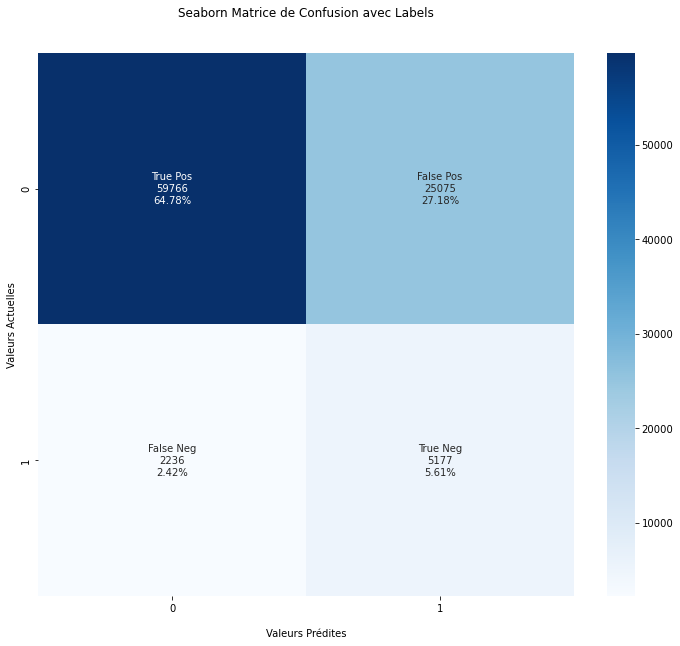
Notons qu’il existe une autre forme de classification appelée *clustering*, dont le principe est de regrouper des données par similarité sans avoir d'information au préalable. La différence avec la classification est que les classes ne sont pas identifiées en amont mais émergent à partir de l'exploration de la structure des données.

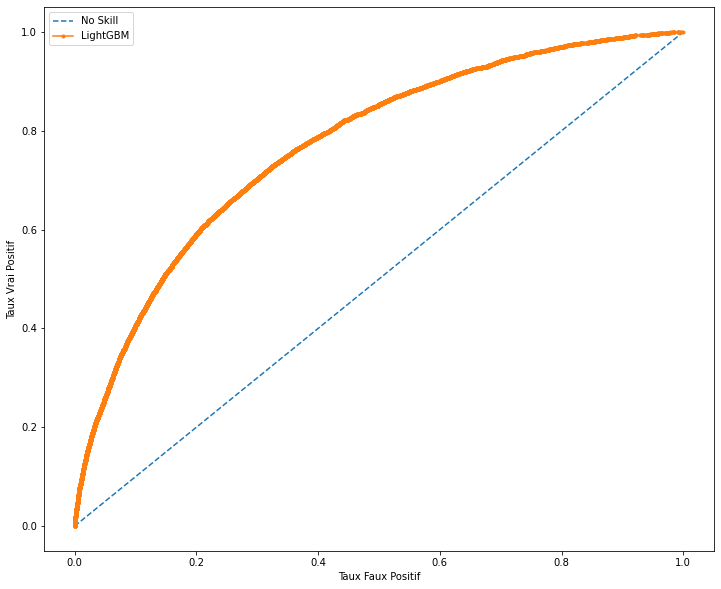
Le *clustering* est souvent utilisé dans des problématiques de réduction de variable et n’a pas été utilisé ici. La méthode utilisée précédemment appartient à la famille des *algorithmes d'apprentissage supervisé***,** dont le but est de généraliser à partir de ce qu'on connaît déjà, tandis que le *clustering* entre dans la catégorie de l'*apprentissage non-supervisé*, où le but est plutôt de chercher à faire émerger de l'information qui n'était pas présentée au départ. Les décisions d’octroi de crédit sont prédéterminées, d’où le cadre de l’*apprentissage supervisé*.

**2.2 - Algorithmes utilisés pour construire le Modèle de**

**Scoring : Exemple LIGHT GBM :**

Toute la force du *Machine Learning* réside dans la diversité des approches utilisées. Plus le nombre de méthodes testées est élevé, plus il sera possible de trouver le meilleur algorithme permettant de répondre à la problématique, ici le modèle de scoring. Contrairement aux modèles statistiques classiques qui doivent vérifier certaines hypothèses sur la distribution des données, le *Machine Learning* a de fortes capacités prédictives compte tenu de son approche non paramétrique et de la faculté de ses algorithmes à apprendre sans a priori, à partir des données. Au vu du large panel de possibilités, trois algorithmes de *Gradient Boosting* ont été testés.



****

**Bilan :**

Une image contenant table

Description générée automatiquement

**2.3 - Fonction Coût  Métier :**

A partir de quel seuil de probabilité allons-nous choisir de classer le client comme un client de confiance ou un client à risques?

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 intêrets.

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

**FN** (Faux Bon Client) : Prêt à tort ===> Perte du capital ou partie du capital ===> Perte de beaucoup d'argent

**FP** (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 10 ans (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.
* 1 x FN = 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.

J'ai créé la fonction findBestThreshold qui automatise cette opération.

Nous lui fournissons en paramètre :

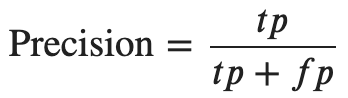
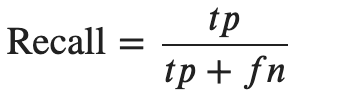
* dictData qui sont les données générées précédemment avec la fonction bestAlgo.
* Algo qui est le nom de l'algorithme à utiliser, ici LGBMClassifier(qui doit être présent dans dictData).
* Formula qui est la formule à calculer et dont nous recherchons le minimum.

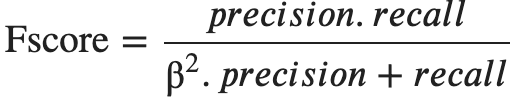
Terminologie :

* FP (*Faux Positifs*) : les cas où la prédiction est positive, mais où la valeur réelle est négative. Perte d'opportunité si le crédit client est refusé à tort, alors qu'il aurait été en mesure d'être remboursé.
* FN (*Faux Négatifs*) : les cas où la prédiction est négative, mais où la valeur réelle est positive. Perte réelle si le crédit client accepté se transforme en défaut de paiement.
* TP (*Vrais Positifs*) : les cas d’acceptation, le crédit client sera remboursé.
* TN (*Vrais Négatifs*) : les cas de refus, le crédit client ne pourra pas être remboursé.

Ainsi, les pertes d'un crédit en raison d'une mauvaise classification dépendent des probabilités *Faux Positifs* et *Faux Négatifs*. L'idée est d'éviter les clients avec un fort risque de défaut. Il est donc nécessaire de pénaliser les FP et FN cités précédemment.

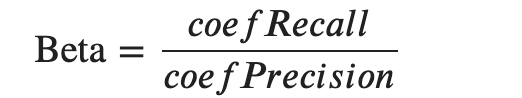
Pour réduire ce risque de perte financière, il faut maximiser deux critères *Recall* et *Precision* :





**L'application de cette métrique métier passe par la quantification de l'importance relative entre *Recall* et *Precision*, à savoir Beta (β).**

Cela revient à estimer le coût moyen d'un défaut, et le coût d'opportunité d'un client refusé par erreur. Cette connaissance métier n'est pas évoquée à ce stade du projet, nous allons donc l'estimer. Cette hypothèse pourra bien entendu être modifiée avec un interlocuteur métier.



**2.4 - Algorithme d’optimisation HyperOpt :**

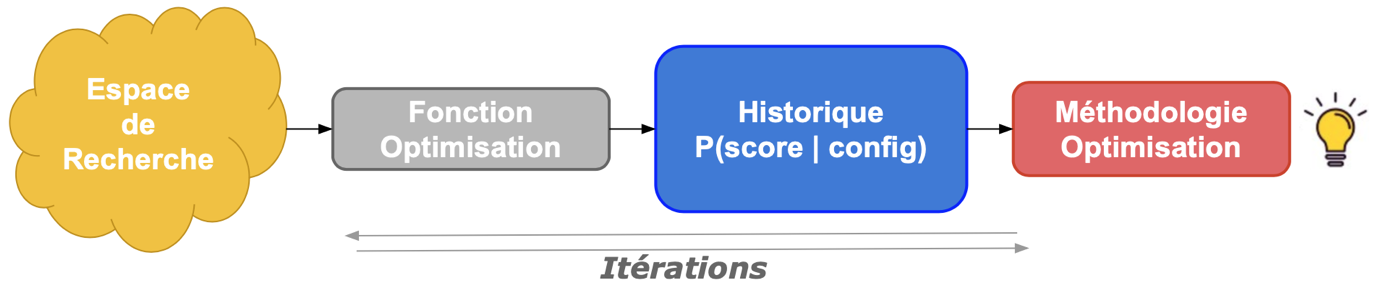
Une alternative pour l'optimisation des fonctions ainsi que des hyperparamètres des pipelines *Machine Learning*. Le réglage des hyperparamètres est basé sur l'**optimisation bayésienne**.

from hyperopt import fmin, tpe, hp, STATUS\_OK, Trials

La configuration optimale des hyperparamètres pour une fonction donnée ne doit pas être entièrement basée sur l'intuition ou l'expérience de certains. La recherche d'une telle configuration optimale doit s'appuyer sur des approches garantissant l’optimalité nécessaire. Parmi les approches les plus utilisées, on retrouve les méthodes basées sur des recherches exhaustives (par exemple *Grid Search* et *Random Search*), ou par *optimisation bayésienne.*

L'approche de l'*optimisation bayésienne* se concentre sur un modèle de probabilité P (score | configuration), qui est mis à jour par un processus itératif d'interrogation d'un historique (score, configuration) dont l'objectif est la maximisation du score donné pour une configuration. HyperOpt prend l'*optimisation bayésienne* comme prémisse en faisant quelques variations dans le processus d'échantillonnage, la définition et la réduction de l'espace de recherche et les algorithmes pour maximiser le modèle de probabilité.

**HyperOpt** nécessite 4 paramètres pour une implémentation de base qui sont: la fonction à optimiser , l' espace de recherche , l' algorithme d'optimisation et le nombre d'itérations.



La mise en œuvre de *XGBOOST* est facile, la seule chose compliquée est le réglage des hyperparamètres. *XGBOOST* couvre plus de 100 hyperparamètres. Dans le contexte du projet l’idée est de pouvoir optimiser quelques hyperparamètres via HyperOpt.

En amont, il est nécessaire d’identifier des hyperparamètres pouvant avoir un impact dans l’amélioration de la métrique d’évaluation.

● **n\_estimators** : nombre d'arbres séquentiels.  
● **max\_depth** : profondeur maximale d'un arbre.  
**● colsample\_bytree** : fraction d'observations à sélectionner pour chaque arbre

Une image contenant texte

Description générée automatiquement

**3 - Interprétabilité du Modèle :**

La réponse au besoin d’interprétabilité est prépondérante, le contexte de prédiction n’est pas uniquement appliqué à des experts de la data science mais au contraire à des experts du crédit. Un chargé de clientèle doit pouvoir utiliser le modèle via l’application mise à disposition, en face à face avec son client, dans le but de lui expliquer le plus simplement possible la décision envisagée dans l’étude de son dossier.

En d’autres termes, **« l’interprétation »** désigne l’évaluation globale du processus de prise de décision. Elle vise à représenter l’importance relative de chaque variable.  
L’idée est donc d’expliciter au mieux le score renvoyé par le modèle. La classe **XGBOOST** permet de mieux comprendre le choix et l’importance de des caractéristiques via un attribut *feature\_importances.*

Une image contenant texte

Description générée automatiquement

**4 - Limites et Améliorations :**

Les résultats obtenus sont basés sur une hypothèse non confirmée par les équipes métier. Le coefficient Beta pourrait être affiné dans ce sens pour éventuellement améliorer la métrique d’évaluation.

L’espace de recherche HyperOpt permet de larges possibilités, le choix des hyperparamètres est évolutif, la question d’élargissement vers d’autres hyperparamètres peut également permettre d’augmenter les performances actuelles.