- Table des Matières -

1. Introduction

La présente **note méthodologique** a pour objectif de décrire :

- la méthodologie d'entraînement d'un modèle de classification,
- la fonction coût, l'algorithme d'optimisation et la métrique d'évaluation,
- l'interprétabilité du modèle et
- les limites et les améliorations possibles

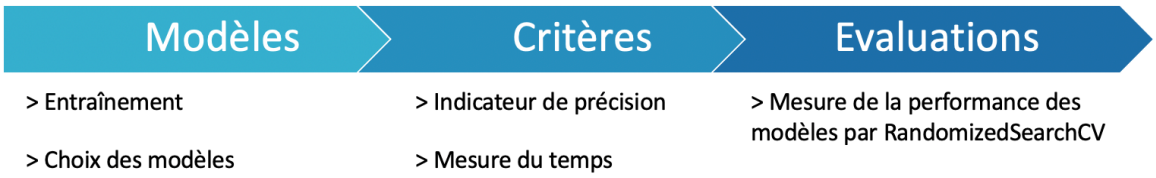
en vue du déploiement de ce modèle dans un tableau de bord interactif.

L'objectif du modèle mentionné tout au long de ce document est de calculer une prédiction sur la probabilité de faillite d'un client de façon automatique.

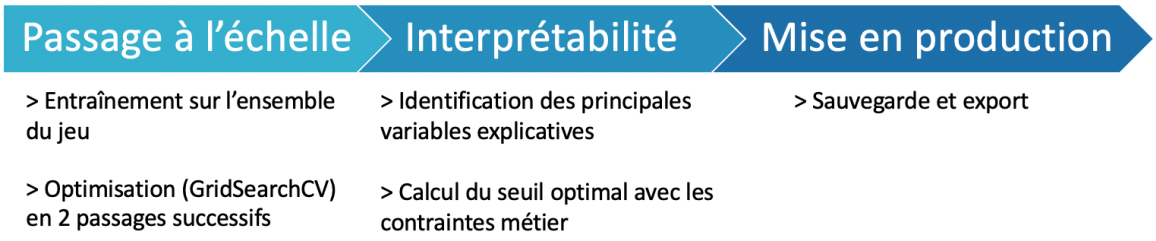
Les données utilisées à cette fin sont disponibles à l'adresse : https://www.kaggle.com/c/home-credit-default-risk/data

Les différentes étapes méthodologiques seront décrites par ordre chronologique de traitement. La méthode retenue s'articule en 2 temps :

- d'abord la sélection du modèle le plus adapté à la problématique traitée,



- puis son optimisation pour être déployé dans un tableau de bord interactif.



2. Modélisation

2.1 Données sources

Le jeu de données est composé de 9 fichiers jeux de données. L'analyse du fichier de descriptions des variables indique que l'ensemble du jeu représente :

- 219 variables agrégées
- 196 variables uniques,
- dont 14 présentes dans plusieurs tables

Une pré-analyse met en avant que les classes sont déséquilibrées :

- classe 0 = 92% des prêts et
- classe 1 = 8% des prêts.

2.2 Prétraitement

Le prétraitement s'appuie sur une adaptation du kernel 'LightGBM with Simple Features' proposé par 'A.Assklou _ Aguiar'. La partie reprise du code consiste principalement en l'agrégation des tables et la création de variables statistiques (moyenne, minimum, maximum, décompte). Il supprime également certaines valeurs aberrantes.

Ce traitement est complété par :

- la suppression des variables n'ayant aucune valeur ('np.nan'),
- la suppression des valeurs infinies ('-np.inf' ou 'np.inf'),
- l'imputation des valeurs manquante par la moyenne pour une variable partiellement renseignée et
- l'échantillonnage des données, seulement pour la phase d'évaluation et de sélection du meilleur modèle (afin de garder une certaine maitrise sur les temps de calcul). Lors de la première étape d'évaluation et de sélection, 10 000 données clients seront ainsi exploitées, contre plus de 92 000 lors de l'optimisation finale.

2.3 Processing

Seules les données d'entrainement sont retenues (c'est-à-dire celles ayant une valeur renseignée pour 'TARGET'). Ces données filtrées sont dissociées en une cible 'y' (la variable 'TARGET') et une source X (toutes les autres variables).

La source est ensuite adaptée puis transformée en X_std avec le StandardScaler de scikit-learn.

Les jeux de test et d'entrainement sont définis avec le 'model_selection.train_test_split' de scikit-learn et les paramètres X = 'X_train_std', Y = 'y', test_size = 0.3 et stratify = 'y')

Ce dernier paramètre permet de corriger le biais de distribution des classes mentionné plus tôt.

2.4 Périmètre de modélisation

Un classifieur simple (Dummy) servira de référence a notre étude. Les données d'entrainements seront appliquées à un ensemble varié d'algorithmes de classification couvrant les différentes familles dont le détail est repris en **annexe 1**.

2.5 Critères d'évaluation

Lors de la phase de sélection, le modèle est évalué d'une part de manière technique sur la base du score AUC. Ceci permet d'estimer la proximité avec un classifieur parfait en traitant séparément la précision et le rappel, qui ont chacun des impacts financiers différents.

Le modèle est évalué d'autre part sur la base de la durée de calcul et d'évaluation des hyperparametres par 10 itérations de RandomizedSearchCV. Ceci permettra de prendre en compte la dimension du jeu complet lorsque l'on passera de l'échantillonnage au jeu complet.

2.6 Résultats

	Model	AUC score	Run Time (in s.)
4	Light GBM	75.8	11.4
3	Random Forest	72.9	16.6
1	Logistic Regression	71.4	9.0
2	Decision Tree	68.0	6.8
0	Dummy	50.0	0.4

On observera que tous les modèles nécessitent plus de temps de calcul que le modèle de référence. A l'exception de ce dernier, les temps mesurés ont le même ordre de grandeur. Nous décidons donc de retenir le modèle ayant la meilleure performance technique, à savoir le Light GBM.

2.7 Optimisation des hyper paramètres

La sélection du modèle ayant été opérée sur un échantillon des données clients, il est nécessaire de l'entrainer sur l'ensemble des données puis de l'optimiser. Le modèle est optimisé de manière plus poussée avec GridSearchCV par itérations sur la recherche des valeurs optimales des 2 hyper paramètres 'max_depth' et 'num_leaves' utilisés avec RandomizedSearchCV.

Pour chacun de ces hyper paramètres, un premier passage avec le GridSearchCV est réalisé avec 5 valeurs étendues. [2, 5, 10, 20, 50, 100]. Le couple des valeurs optimales identifié sert de base a un second passage avec le GridSearchCV et 5 nouvelles valeurs sélectionnées avec la logique suivante :

- la valeur centrale est la valeur optimale,
- les 2 valeurs inférieures sont comprises entre la valeur optimale et la valeur la précédant dans le premier passage et
- les 2 valeurs supérieures sont comprises entre la valeur optimale et la valeur la suivante dans le premier passage.

3. Optimisation métier par la fonction gain

Lors de la phase de sélection, notre parti pris a été d'évaluer de manière technique sur la base du score AUC afin de traiter séparément la précision et le rappel, qui ont chacun des impacts financiers différents. Le tableau ci-dessous reprend les différentes typologies de clients

		Classe réelle	
		0	1
Classe prédite	0	**TN** (Vrai Négatifs)	**FN** (Faux Négatifs)
	1	**FP** (Faux Positifs)	**TP** (Vrais Positifs)

Un client aura contribution financière différente pour l'organisation selon sa classe :

- en classe réelle 0 il contribue positivement aux résultats en amenant une marge,
- en classe réelle 1 il contribue négativement aux résultats en amenant une perte,
- en classe prédite 0 il est validé par le modèle et intègre le portefeuille de clients
- en classe prédite 1 il n'est pas validé par le modèle et ne contribue donc pas aux gains ou perte de l'organisation.

Sur cette base, on peut en déduire une première formulation de la marge du modèle :

Gain = Somme des marges des TN – somme des pertes des FN

La fonction 'gain ' est donc développée Afin de déterminer le seuil de classification permettant de maximiser la marge totale. Pour estimer cette marge, on retient les hypothèses suivantes :

- chaque client génère le même Chiffre d'Affaires (C.A.) moyen de 100 €
- la marge sur un dossier représente 10% du C.A. moyen et
- la perte sur un dossier = 100 % du C.A. moyen.

En intégrant ces hypothèses dans la formule, on obtient :

Gain = [TN x 10% - FN x 100%] x 100 €

= 10 € x TN - 100 € x FN

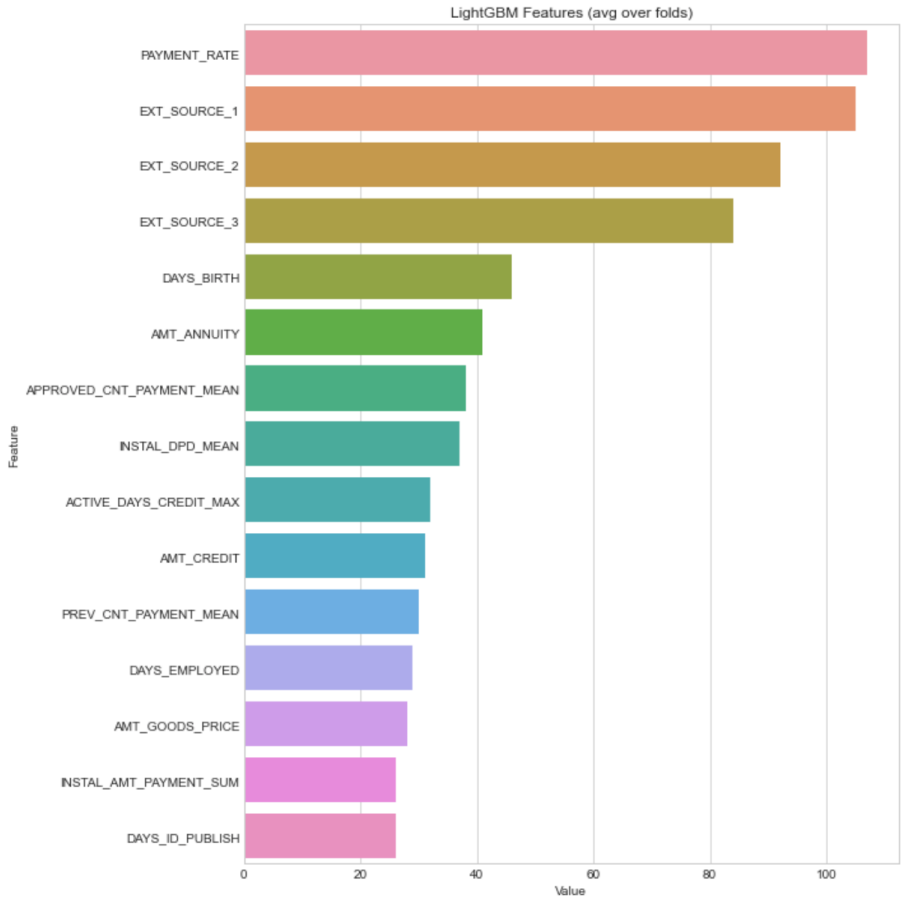
La courbe reprise en

Le seuil correspondant a cet optimal est de 0,086 pour une prédiction positive

4. Interprétabilité du modèle

Le classifieur LGBM possède dans sa bibliothèque la méthode 'feature_importances_'.

En appliquant cette méthode aux jeux de données processé, on obtient le poids de chaque variable dans le calcul du score de probabilité de classe. Le graphique ci-dessous indique par ordre décroissant d'importance les 15 premières variables les plus importantes :



Ce graphique met en avant que les variables :

- PAYMENT_RATE,
- EXT_SOURCE_1,
- EXT_SOURCE_2 et
- EXT_SOURCE_3

se détachent comme étant celles qui contribuent le plus fortement au calcul du score.

5. Limites et améliorations possibles

5.1 Performance intrinsèque

Le score AUC obtenu après la seconde itération pour la recherche des hyper paramètres optimaux de notre modèle est de 0,783.

Ce résultat est inférieur à celui de 0,792 obtenu par l'auteur du kernel 'LightGBM with Simple Features' proposé par 'A.Assklou _ Aguiar' et adapté ici pour la partie prétraitement. Cette différence s'explique par le fait que les auteurs de ce kernel ont optimisé l'ensemble des hyper paramètres en s'appuyant sur une optimisation Bayesienne qui pourrait également être mise en place ici.

D'autres méthodes d'optimisation peuvent être également testées, telles que l'utilisation de la librairie Hyper Opt.

De même, il est à noter que le score obtenu par 'A.Assklou _ Aguiar' reste, inférieur au meilleur résultat obtenu dans la compétition. Au moment de la rédaction du présent document, l'équipe 'Kraków, Lublin i Zhabinka' a obtenu un score de 0,817 avec un notebook public. Il serait donc possible de répliquer leurs méthodes de prétraitement, modélisation et d'optimisation pour dupliquer cette performance.

5.2 Performance métier

Les remarques précédentes étant posées, il convient de rappeler, comme abordé dans la section sur la mesure d'évaluation, que la sélection et l'optimisation du modèle s'appuient sur une mesure technique, le score AUC.

Afin de déterminer le seuil optimal d'un point de vue métier, la fonction 'gain' a été construite. Cette même fonction pourrait être utilisée pour la recherche d'optimisation des hyper paramètres avec la fonction 'make_scorer'.

5.3 Hypothèse métier

La fonction gain utilise une série d'hypothèses globales extérieures faisant appel :

- au chiffre d'affaires moyen par client,
- au gain moyen en l'absence de défaut de paiement et
- à la perte moyenne dans le cas d'un défaut de paiement.

Ces hypothèses peuvent être affinées dans un premier temps en utilisant les valeurs moyennes constatées dans le jeu de données d'entrainement. Dans un second temps, il est possible de d'utiliser des valeurs moyennes qui caractériseraient par exemple des classes de variables pour les 4 variables les plus importantes.

De façon plus absolue, les profits et pertes pourraient être calculés de façons individuelles.

6. Annexes

Annexe 1 : détail des algorithmes et des hyper paramètres étudiés

- algorithme linéaire
  - Régression Logistique
    - 'penalty': ['l2', 'l1'],
    - 'C': np.logspace(-5, 5, 11)
- algorithme non linéaire :
  - Arbre de décision
    - 'max_depth': range(2, 20, 5),
    - 'criterion': ['gini', 'entropy']
- algorithmes ensemblistes :
  - Forêt alléatoire
    - 'max_depth': range(2, 20, 5),
    - 'criterion': ['gini', 'entropy']
  - Light GBM
    - 'max_depth': range(2, 20, 5),
    - 'num_leaves': range(1, 101, 10)

Annexe 2 : Niveaux de marge associés aux différents seuils de prédictions du modèle