

note méthodologique P7

Parcours Data Scientist « Implémentez un modèle de scoring »

**Table des matières**

[I. Contexte 2](#_Toc89174214)

[II. La méthodologie d'entraînement du modèle 2](#_Toc89174215)

[A. Split jeu d’entrainement et jeu de test 2](#_Toc89174216)

[B. Classification binaire avec déséquilibre des classes 2](#_Toc89174217)

[C. Test de différents modèles. 3](#_Toc89174218)

[III. La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation 3](#_Toc89174219)

[A. Fonctions coût 3](#_Toc89174220)

[B. Evaluation 4](#_Toc89174221)

[1. Choix de la métrique 4](#_Toc89174222)

[2. Quantification de l'importance relative entre PRECISION et RECALL (Beta) 4](#_Toc89174223)

[C. Algorithme d’optimisation 4](#_Toc89174224)

[IV. L’interprétabilité globale et locale du modèle 5](#_Toc89174225)

[V. Les limites et les améliorations possibles 5](#_Toc89174226)

# Contexte

Cette note méthodologique est un des livrables du projet « Implémentez un modèle de scoring » du parcours Data Scientist d’Openclassroom.

Son objet est de présenter le processus de modélisation et d’interprétabilité du modèle mis en place dans le cadre du projet 7.

Le projet consiste à développer pour vous, l’entreprise, « Prêt à Dépenser », société de crédit de consommation, un modèle de scoring de la probabilité de défaut de paiement d’un client avec pas ou peu d’historique de prêt.

La base de données que vous nous avez confiée est constituée de fichiers relatifs à 307 000 clients comportant 121 caractéristiques (âge, sexe, emploi, logement, revenus, informations relatives au crédit, notation externe, etc.)

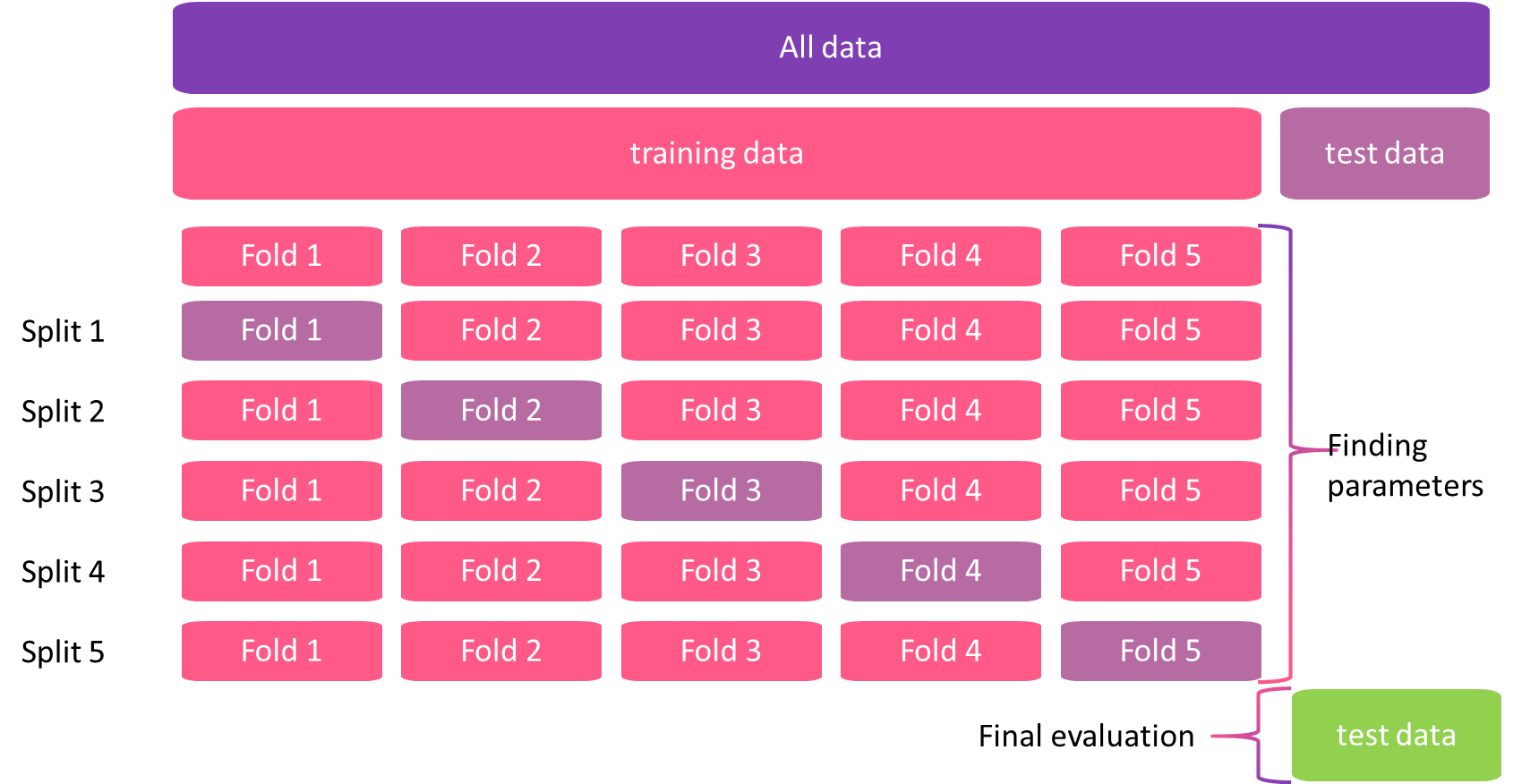
# La méthodologie d'entraînement du modèle

## Split jeu d’entrainement et jeu de test

Nous avons entraîné notre modèle sur la base du jeu de données, après analyse exploratoire et création de nouvelles features. Le notebook utilisé est consultable sur github.

Le jeu de données initial a été séparé en plusieurs parties de façon à disposer :

* d’un jeu de training (75% des individus) qui a été séparé en plusieurs folds pour entraîner les différents modèles et optimiser les paramètres (cross validation) sans overfitting.
* d’un jeu de test (25 % des individus) pour l’évaluation finale du modèle



## Classification binaire avec déséquilibre des classes

Nous sommes face à un problème de classification binaire et une problématique de déséquilibre des classes. En effet, nous avons une classe sous représentée (9 % de clients en défaut contre 91 % de clients sans défaut). Nous avons dû prendre cela en compte dans l’entraînement des modèles. Un modèle « naïf » prédisant systématiquement que les clients sont sans défaut aurait une justesse de 0.92 et pourrait être considéré à tort comme un modèle performant alors qu’il ne permettrait pas de détecter les clients à risque.

Nous avons testé deux approches pour rééquilibrer les deux classes : Sample Weights et Smote :

* SMOTE permet de créer des données synthétiques à partir des données existantes ;
* Sample Weights permet de modifier les poids associés aux observations. Ainsi, une observation mal classée dans la classe minoritaire pénalise davantage la fonction de perte qu’une observation mal classée dans la classe majoritaire.

Le jeu de données post feature-engineering a été exporté et utilisé dans un premier temps sur la base d’un échantillon de 10 % du jeu de données initial, notre objectif étant de comparer avec un temps de calcul réduit les approches SMOTE et Sample Weights.

Pour chaque approche, nous avons testé différents modèles avec recherche d’hyperparamètres et cross validation (5 folds).

## Test de différents modèles.

Voici les différents modèles qu’il nous a paru intéressant de tester :

* Random Forest Classifier ;
* XGBoost ;
* Multi Layer Perceptron Classifier (réseau de neurone peu profond) – avec SMOTE uniquement, Sample Weights n’étant pas implémenté pour ce modèle ;
* Modèle Stacking combinant les 3 modèles précédents (Algorithme XGBoost) : ce modèle prend en entrée la combinaison des probabilités a posteriori issues des algorithmes précisés ci-avant ainsi que le jeu de données de base. Le modèle stacking combine donc les approches de bagging (random forest) et boosting (xgboost).

Le travail sur jeu réduit a montré que les 2 approches présentaient des résultats d’ordre de grandeur comparable sur le jeu de validation. Elles ont donc été conservées pour un entraînement sur le jeu complet. Le processus a ensuite été réalisé sur l’intégralité du jeu de données.

Le choix du meilleur modèle a été effectué en retenant le modèle avec le meilleur score sur le jeu de validation.

Logigramme de méthodologie d’entraînement du modèle

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

## Fonctions coût

Nous avons entrainé les différents modèles dans la fonction de cross validation et testé différentes combinaisons d’hyperparamètres. Les fonctions de coût pour les algorithmes entraînés sont les suivantes :

|  |  |
| --- | --- |
| Algorithme | Fonction de coût |
| XGBoost Classifier  XGBoost Classifier Stacking | Régression logistique pour classification binaire |
| Random Forest Classifier | Minimisation du coefficient de GINI pour chaque nœud |
| Multi Layer Perceptron Classifier | Entropie croisée |

## Evaluation

### Choix de la métrique

Dans le jeu de données de base, il y a une part de 92 % des clients qui n’ont pas d’incident de paiement, tandis que 8 % des clients ont eu des incidents.

La matrice de confusion est la suivante :

|  |  |  |
| --- | --- | --- |
|  | Clients prédits en défaut | Clients prédits sans défaut |
| Clients sans défaut | Faux positifs | Vrais négatifs |
| Clients réellement en défaut | Vrais positifs | **Faux négatifs** |

Dans votre métier, la banque, mieux vaut éviter de mal catégoriser un client avec un fort risque de défaut.

Nous avons donc cherché à minimiser le pourcentage de faux négatifs et à maximiser le pourcentage de vrais positifs. Cela revient à maximiser le « RECALL » de 𝑣𝑟𝑎𝑖𝑠 𝑝𝑜𝑠𝑖𝑡𝑖𝑓𝑠, soit le nombre de *vrais positifs* par rapport à l’ensemble (v𝑟𝑎𝑖𝑠 𝑝𝑜𝑠𝑖𝑡𝑖𝑓𝑠+𝑓𝑎𝑢𝑥 𝑛é𝑔𝑎𝑡𝑖𝑓𝑠).

Par ailleurs, pour étendre votre base de clientèle, nous voulons éviter de tous les classer en défaut. On cherche donc à éviter d'avoir un trop grand nombre de faux positifs.

Nous avons donc cherché à maximiser la PRECISION, soit le soit le nombre de *vrais positifs* par rapport à l’ensemble (v𝑟𝑎𝑖𝑠 𝑝𝑜𝑠𝑖𝑡𝑖𝑓𝑠+𝑓𝑎𝑢𝑥 posi𝑡𝑖𝑓𝑠)

Pour votre problématique métier, le RECALL est plus important que la PRECISION car on préfèrera vraisemblablement limiter un risque de perte financière plutôt qu’un risque de perte de client potentiel.

On cherche donc une fonction qui optimise les 2 critères en donnant plus d'importance au recall. C’est ce que permet la Fonction F Beta Score : https://en.wikipedia.org/wiki/F1\_score) avec Beta le coefficient d'importance relative du recall par rapport à la précision.

On cherchera à **maximiser** cette métrique.

### Quantification de l'importance relative entre PRECISION et RECALL (Beta)

Il faut pour cela estimer :

* le coût moyen d'un défaut
* le coût d'opportunité d'un client potentiel accidentellement écarté

Nous avons pris des hypothèses, que nous pourrons ajuster selon vos directives métiers :

* que chaque défaut entraîne la dépense de 30 % du montant du crédit en pertes et frais de recouvrement ;
* qu'un client a 10 % de chance de souscrire au crédit quand il en fait la demande à un conseiller, et donc que le coût d'opportunité pour un client potentiel accidentellement écarté est de 10 % du montant du crédit.

Nous obtenons les coefficient suivants :

* Coefficient RECALL : de 30 % \* le montant moyen de crédit des personnes en défaut
* Coefficient PRECISION : 10 % \* le montant moyen de crédit des personnes sans défaut

Ce qui nous permet d’établir la valeur de Beta égale au Coefficient RECALL divisé par le Coefficient PRECISION.

## Algorithme d’optimisation

Nous avons retenu la meilleure combinaison d’hyperparamètres pour chaque algorithme. Le modèle ayant le meilleur score en cross validation sur le jeu de training a été retenu : il s’agit du modèle XGBoost de stacking combinant les modèles XGBoost et Random Forest Classifier.

Ce modèle dispose d’une bonne performance en combinant des modèles de machine learning différents : Random Forest Classifier avec des arbres de décisions complets entraînés en parallèle contre XGBoost avec des arbres de décisions faibles (weak learners) mais entraînés de façon itérative.

# L’interprétabilité globale et locale du modèle

Le modèle étant destiné être utilisé par vos conseillers clients, notre objectifs a été de leur donner les moyens d’expliquer les décisions de l’algorithme aux clients réels. C’est pourquoi nous avons un module d’explicabilité.

Pour cela nous avons utilisé la librairie SHAP, librairie adaptable à n’importe quel modèle de machine learning et permettant de comprendre quelles caractéristiques expliquent la décision d’octroi du crédit.

# Les limites et les améliorations possibles

Nous avons effectué notre modélisation sur la base d’une hypothèse forte : la définition d’une métrique d’évaluation : le F Beta Score avec Beta fixé suivant certaines hypothèses non confirmées par le métier.

L’axe principal d’amélioration serait de définir plus finement la métrique d’évaluation en collaboration avec les équipes métier.

L’interprétabilité du modèle pourrait être étoffée en considérant les variables issues du one hot encoding comme une seule et même variable dans la perturbation (un client ne pouvant cumuler plusieurs caractéristiques dans la logique du jeu de données initial.

Par ailleurs, la partie de traitement préalable du jeu de données a été abordée de façon superficielle en réutilisant un notebook issu de Kaggle qui se base uniquement sur une table du jeu de données. Il y a très probablement l’opportunité d’améliorer la modélisation en utilisant d’autres features des données fournies, ainsi qu’en créant de nouvelles features en collaboration avec les équipes métier.