

OpenClassrooms - Parcours Data Scientist

Claire-Marie BESNIER

## Projet 7 : Implémenter un modèle de scoring dans le secteur bancaire

# Note méthodologique

## Présentation du projet

Une entreprise souhaite développer un **modèle de scoring** pour prédire la **probabilité de défaut de paiement** d'un client à partir de données diverses. Ce modèle servira d'aide à la décision pour **octroyer ou non un crédit**. L'entreprise souhaite aussi créer un **dashboard interactif** afin de communiquer de manière transparente avec ses clients.

Les objectifs de ce projet sont de :

- **Construire un modèle de scoring** permettant de prédire la probabilité de faillite d'un client
- **Construire un dashboard interactif** à destination des gestionnaires clients permettant d'interpréter les prédictions du modèle et d'améliorer la connaissance du profil client.

## Présentation des données

Le jeu de données est extrait d'une compétition Kaggle proposée en 2018 par l'entreprise Home Credit. Les données sont disponibles ici : <https://www.kaggle.com/c/home-credit-default-risk/data>.

On dispose de 10 tables de données avec des informations sur plus de 307 000 clients, comprenant des informations très diverses : données comportementales, données provenant d'autres institutions financières, etc. Sur ces 10 tables, 8 recensent des informations sur les clients :

### Données principales

- *application\_train.csv* / *application\_test.csv* : données principales concernant la demande de crédit chez Home Credit. Données séparées en jeu d'entraînement et de test. Seul le jeu de test comporte la variable cible.

### Données concernant des crédits ou produits chez Home Credit

- *previous\_application.csv* : données concernant les demandes de crédit précédentes
- *credit\_card\_balance.csv* : relevés mensuels des cartes de crédits
- *POS\_CASH\_balance.csv* : relevés mensuels des crédits
- *installments\_payments.csv* : données concernant les remboursements de crédits

### Données concernant des crédits avec d'autres organismes

- *bureau.csv* : données concernant d'autres crédits reportés au Credit Bureau
- *bureau\_balance.csv* : relevés mensuels des autres crédits reportés au Credit Bureau








## Démarche

Voici les différents points abordés dans cette note méthodologique :

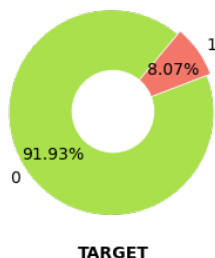
- Analyse exploratoire
- Création et sélection de variables
- Pre-processing
- Métrique d'optimisation
- Modélisation
- Interprétation
- Synthèse et Améliorations possibles
- Dashboard interactif

## Analyse exploratoire

On dispose de 7 fichiers de données concernant les 307 511 clients présents dans le fichier principal `application_train.csv`. Ces clients ne sont pas tous présents dans les autres fichiers. On retrouve notamment une part très faible de ces clients dans les fichiers `credit_card_balance.csv` et `bureau_balance.csv`.

	<i>application_train</i>	<i>credit_card_balance</i>	<i>installment_payment</i>	<i>POS_CASH_balance</i>	<i>previous_application</i>	<i>bureau</i>	<i>bureau_balance</i>
<b>Lignes</b>	307 511	3 840 312	13 605 401	10 001 358	1 670 214	1 716 428	27 299 925
<b>Colonnes</b>	122	23	8	8	37	17	3
<b>Var. quanti</b>	65	22	8	7	21	14	2
<b>Var. quali</b>	57	1	0	1	16	3	1
<b>Col. Nan</b>	67/122	9/23	2/8	2/7	16/37	7/17	0/3
<b>Taux max Nan</b>	70%	20%	<1%	<1%	100%	70%	-
<b>% clients sur 307 511 clients dans application (SK_ID_CURR)</b>	 100%	 28 %	 95 %	 94 %	 95 %	 86 %	 30 %

→ Zoom sur le fichier principal : `application_train.csv`



Ce fichier comporte des informations sur 307 511 clients qui ont déposé une demande de crédit, ainsi que la variable cible TARGET qui vaut 0 pour les « Non Defaulters », les clients qui ont remboursé leurs crédits, et 0 pour les « Defaulters », les clients qui n'ont pas remboursé leurs crédits.

On observe un déséquilibre très marqué des classes : 92% des clients ont remboursé leurs crédits (TARGET = 0).

### Valeurs manquantes

On observe 67 colonnes avec des taux de valeurs manquantes importants de 50 à 70%. Ces variables correspondent principalement à des données normalisées concernant le logement actuel du client, ainsi que des indicateurs statistiques associés (moyenne, médiane, mode)

### Variables les plus corrélées avec la variable cible

Variable quantitatives	Corr.
EXT_SOURCE_3	0.247
EXT_SOURCE_1	0.217
EXT_SOURCE_2	0.213
DAYS_BIRTH	0.102
DAYS_LAST_PHONE_CHANGE	0.073
DAYS_EMPLOYED	0.072
DAYS_ID_PUBLISH	0.067

Variables qualitatives	Corr.
OCCUPATION_TYPE	0.102
ORGANIZATION_TYPE	0.089
NAME_INCOME_TYPE	0.084
REG_CITY_NOT_WORK_CITY	0.079
FLAG_EMP_PHONE	0.072
REG_CITY_NOT_LIVE_CITY	0.069
FLAG_DOCUMENT_3	0.069

## Création de variables

Dans le but de mieux interpréter le comportement des clients avec un risque de défaut de paiement, on crée un ensemble de nouvelles variables :

- **Variables complémentaires** : durée du crédit en mois, nombre d'enfants, etc.
- **Ratios** : entre les différents montants crédit, revenus, remboursement, etc.
- **Variables agrégées** : à partir des données sur les autres crédits engagés par un même client, par exemple : nombre de crédits, pourcentage de crédits actifs, caractéristique de la dernière demande de crédit refusée, etc.

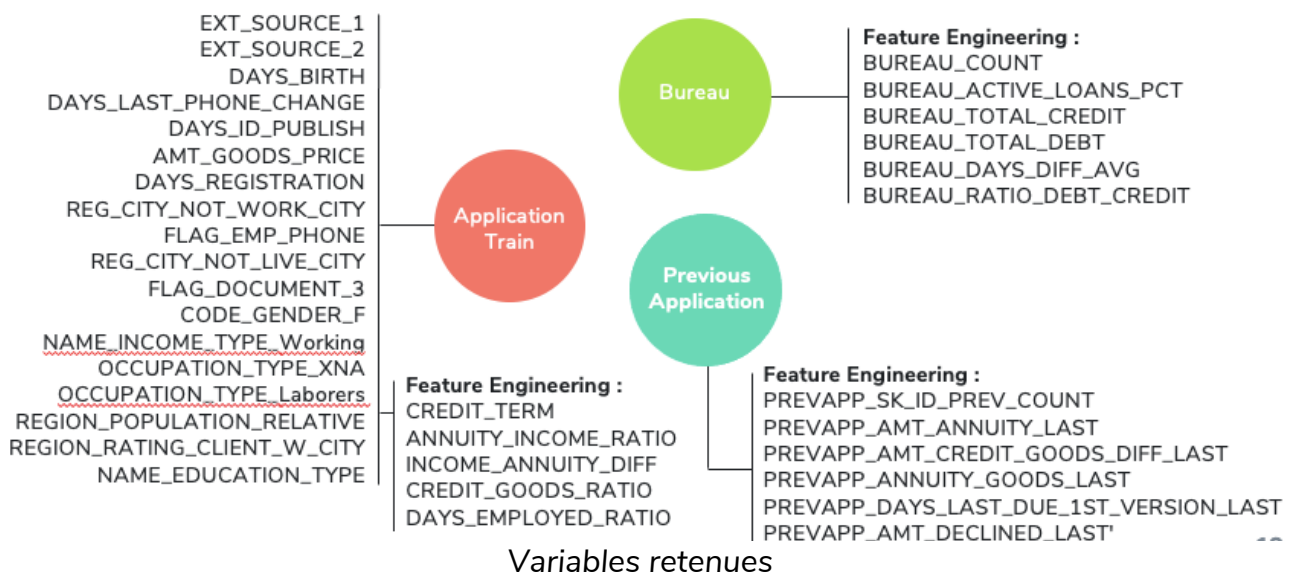
## Sélection de variables

On choisit de retenir seulement 3 fichiers de données : application\_train.csv, bureau.csv et previous\_application.csv. En effet, les fichiers credit\_card\_balance.csv et bureau\_balance.csv contiennent une très faible part des clients étudiés et il faudrait alors imputer un grand nombre de valeurs et les variables issues des fichiers installment\_payment.csv et POS\_CASH\_balance.csv ont peu d'intérêt selon les tests effectués.

Pour sélectionner les meilleures variables, on utilise la librairie Pycaret en créant une pipeline avec les paramètres suivants :

- Suppression de variables à faible variance
- Suppression des variables corrélées avec un seuil de 0,8
- Sélection de 15% des variables initiales

**On passe ainsi de 115 variables à 35 variables.**



## Pre-processing

Hypothèses testées et retenues pour le preprocessing

Étapes	Paramètres testés	Paramètre retenu
Séparation train/test	70/30, 80/20	80/20
Imputation	Moyenne / médiane	Médiane
Normalisation	Standard / MinMax / MaxAbs / Robust Scaler	Standard Scaler
Transformation	Power/ Quantile Transformer	PowerTransformer
Déséquilibre des classes	Oversampling (SMOTE), Pondération	Pondération

## → Zoom sur la gestion du déséquilibre des classes

Ce jeu de données présente un déséquilibre très marqué des classes (92% des clients sont « Non Defaulters »). Cette particularité rend d'apprentissage plus difficile pour un algorithme de classification car il existe un biais lié à la classe majoritaire. Il existe des méthodes qui permettent de remédier à ce problème :

- **le sur-échantillonnage ou « oversampling »**, qui consiste à rééquilibrer le jeu de données en augmentant artificiellement le nombre d'observations de la classe minoritaire.
- **le sous-échantillonnage ou « undersampling »**, qui consiste à rééquilibrer le jeu de données en diminuant le nombre d'observations de la classe majoritaire.
- **la pondération (class weight = balanced)**, qui consiste à attribuer aux observations un poids qui est inversement proportionnel à la fréquence observée dans les données d'entrées.

Dans ce projet, la méthode de sur-échantillonnage a été testée avec l'algorithme **SMOTE** (Synthetic Minority Over-sampling Technique), qui ne duplique pas les observations existantes mais qui crée de nouvelles observations synthétiques proches des observations de la classe minoritaire.

**La méthode de pondération a finalement été retenue.**

## Métrique d'optimisation

Les métriques comme l'accuracy ne sont pas adaptées à un jeu de données présentant un déséquilibre. Certaines métriques sont plus intéressantes :

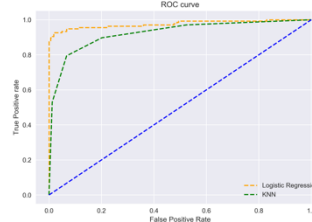
- **Précision** : taux d'observations positives parmi les observations prédites positives, permet de minimiser les erreurs sur les positifs prédits
- **Sensibilité (recall)** : taux de vrais positifs, permet de détecter un maximum de positifs et limiter les faux négatifs
- **ROC-AUC** : permet de maximiser l'aire sous la courbe ROC (sensibilité / spécificité)

TN	FP
FN	TP

Matrice de confusion

$$\text{precision} = \frac{TP}{TP+FP}$$

$$\text{recall} = \frac{TP}{TP+FN}$$



Courbe ROC

## Définition d'une métrique spécifique

Afin d'optimiser au mieux le modèle, on définit une fonction coût et un score associé dont les paramètres sont les suivants :

- **Vrais Négatifs (TN)** : Clients qui remboursent leurs crédits et qui sont prédits comme tels  
**Coût : + 25 000**, correspond à la médiane des remboursements annuels (annuity)
- **Vrais Positifs (TP)** : Clients qui ne remboursent pas leurs crédits et qui sont prédits comme tels  
**Coût : 0**, aucun coût associé
- **Faux Négatifs (FN)** : Clients qui ne remboursent pas mais prédit comme « Non Defaulters »  
**Coût : -250 000**, correspond à 50% de la médiane du montant des crédits sur le jeu de données
- **Faux Positifs (FP)** : Clients qui remboursent leurs crédits mais prédit comme « Defaulters »  
**Coût : -250**, pénalité symbolique car il s'agit en fait d'un manque à gagner

$$\text{cost} = 25\,000\,TN - 250\,FP - 250\,000\,FN \quad \text{score} = \frac{\text{cost} - \text{baseline}}{\text{best} - \text{baseline}}$$

**best** : coût associé à un modèle parfait, sans FN et FP

**baseline** : coût associé à un modèle naïf classant tous les clients comme « Non Defaulters »

## Modélisation

Après comparaison de plusieurs modèles avec la librairie Pycaret, on retient deux modèles pour optimisation des hyper-paramètres :

- **Régression logistique**
- **LightGBM** (Light Gradient Boosting Machine)

Résultats après optimisation des hyper-paramètres par validation croisée sur 10 échantillons.



## Essais de « Blending » et « Stacking »

Il s'agit de méthodes ensemblistes qui consistent à agréger des modèles différents :

- Le « Blending » combine plusieurs algorithmes de machine learning et utilise le vote à la majorité pour faire la prédiction finale.
- Le « Stacking » permet de créer un méta-modèle qui génère la prédiction finale à partir des prédictions d'autres modèles.

Blending		Stacking															
		Meta-modèle : Rég. logistique	Meta-modèle : Light GBM														
<table><tr><td>39 736</td><td>16 811</td></tr><tr><td>1 663</td><td>3 292</td></tr></table>	39 736	16 811	1 663	3 292		<table><tr><td>38 471</td><td>18 076</td></tr><tr><td>1 516</td><td>3 439</td></tr></table>	38 471	18 076	1 516	3 439		<table><tr><td>40 694</td><td>15 853</td></tr><tr><td>1 734</td><td>3 221</td></tr></table>	40 694	15 853	1 734	3 221	
39 736	16 811																
1 663	3 292																
38 471	18 076																
1 516	3 439																
40 694	15 853																
1 734	3 221																
Score : 0,3217 ROC-AUC : 0,7468		Score : 0,3256 ROC-AUC : 0,7520		Score : 0,3269 ROC-AUC : 0,7505													

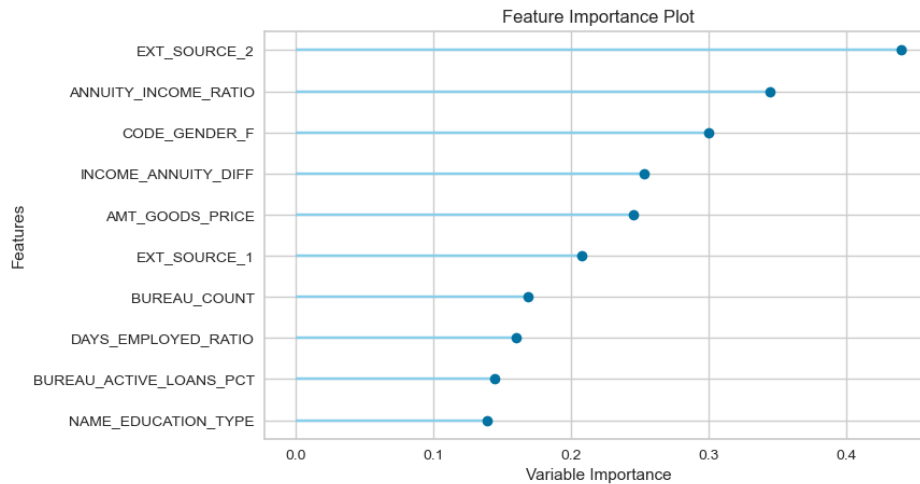
## Modèle retenu : Light GBM

On choisit de retenir le modèle optimisé Light GBM pour la suite. C'est le modèle qui donne les meilleurs résultats pour le score défini et également pour la métrique ROC-AUC.

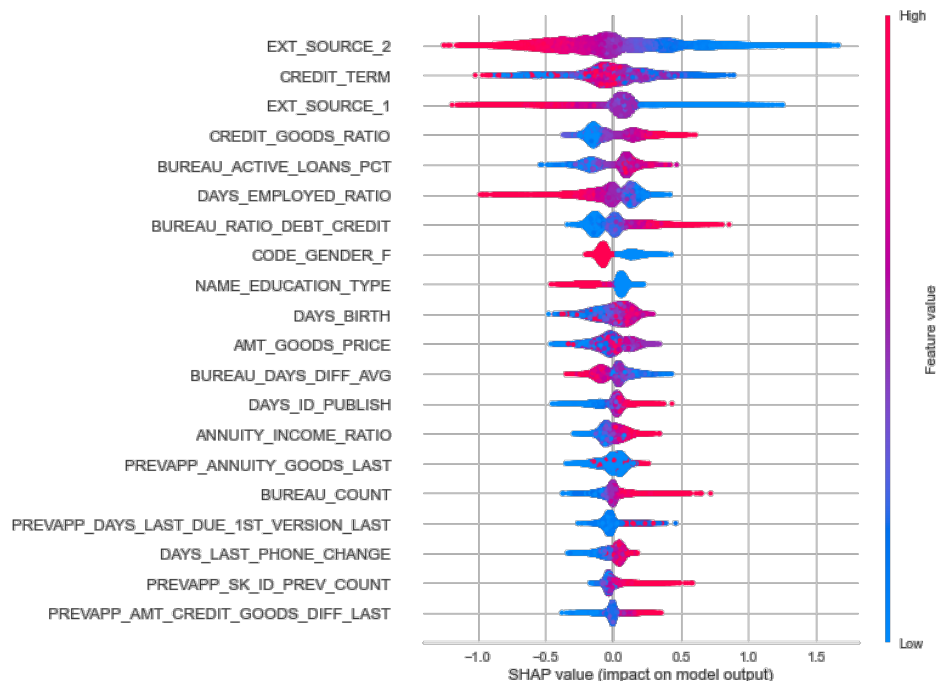
À noter que les modèles de régression logistique et stacking avec méta-modèle régression logistique permettent d'obtenir le moins de Faux-Négatifs, les clients les plus coûteux, mais cela se fait au détriment des Vrais Positifs. La fonction coût et le score défini permettent de trouver un équilibre optimal entre les deux.

## Interprétation

### Importance des variables – Régression Logistique



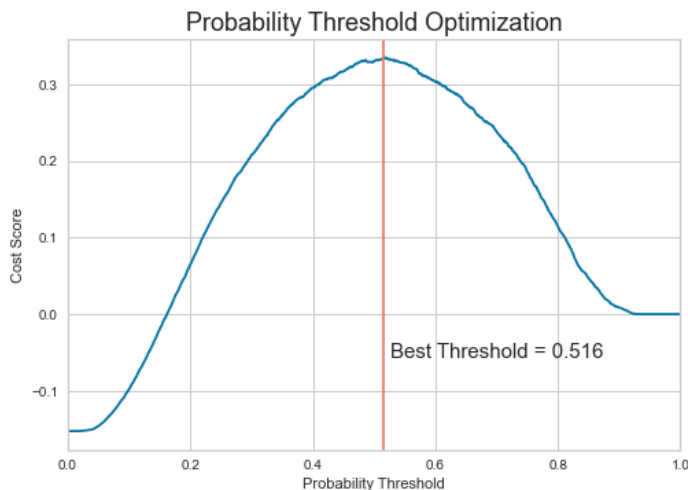
### Importance des variables – Light GBM



Dans les variables les plus significatives, on retrouve :

- **les variables identifiées dans l'analyse exploratoire**, notamment EXT\_SOURCE\_1, EXT\_SOURCE\_2, AMT\_GOODS\_PRICE et DAYS\_BIRTH
- **les variables créées**, avec notamment les ratios et différence de variables : ANNUITY\_INCOME\_RATIO, INCOME\_ANNUITY\_DIFF, CREDIT\_GOODS\_RATIO

## Optimisation de seuil



L'optimisation du seuil de prédiction entre « Defaulters » et « Non-Defaulters » permet encore d'optimiser les résultats du modèle.

Le seuil associé au meilleur score est de 0,516 avec un score de 0,3348.

42 248	14 299
1 852	3 103

Score : 0,3348

ROC-AUC : 0,7520

## Synthèse et améliorations possibles

Le modèle final s'appuie sur un algorithme Light GBM avec 35 variables d'entrées issues de 3 fichiers de données. Les performances sur le jeu de test sont de 0,7520 pour la métrique ROC-AUC et 0,3348 pour le score financier spécifique défini pour ce projet.

Voici quelques pistes d'améliorations possibles de ce modèle :

- Inclure un plus grand nombre de variables, notamment en utilisant l'ensemble des fichiers de données et en travaillant sur la partie création de variables
- Optimiser la gestion et l'imputation des valeurs manquantes
- Optimiser la fonction coût et le score en choisissant les paramètres de coût avec un expert métier (coûts des FN, TP)

## Dashboard interactif

Un dashboard interactif intégrant le modèle retenu est disponible ici :

<https://share.streamlit.io/cmbesnier/credit-dashboard/main/main.py>

