Prêt à dépenser

NOTE METHODOLOGIQUE

TABLE DES MATIERES

Avril 2021

Implémenter un modèle de scoring

La société « Prêt à dépenser » propose des crédits à la consommation.
Notre étude vise à **développer un modèle de scoring de la probabilité de défaut de paiement du client** pour étayer la décision d'accorder ou non un prêt à un client, en s'appuyant sur des sources de données variées.
Dans un souci de transparence, l'entreprise souhaite également **développer un tableau de bord (dashboard) interactif** pour que les chargés de clientèle puissent à la fois expliquer les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

1. PRESENTATION DES DONNEES

Arborescence des 7 tables de données

Sept tables de données sont mises à disposition pour mener à bien notre démarche. Elles sont constituées de données anonymisées d'informations **personnelles et bancaires** des clients.

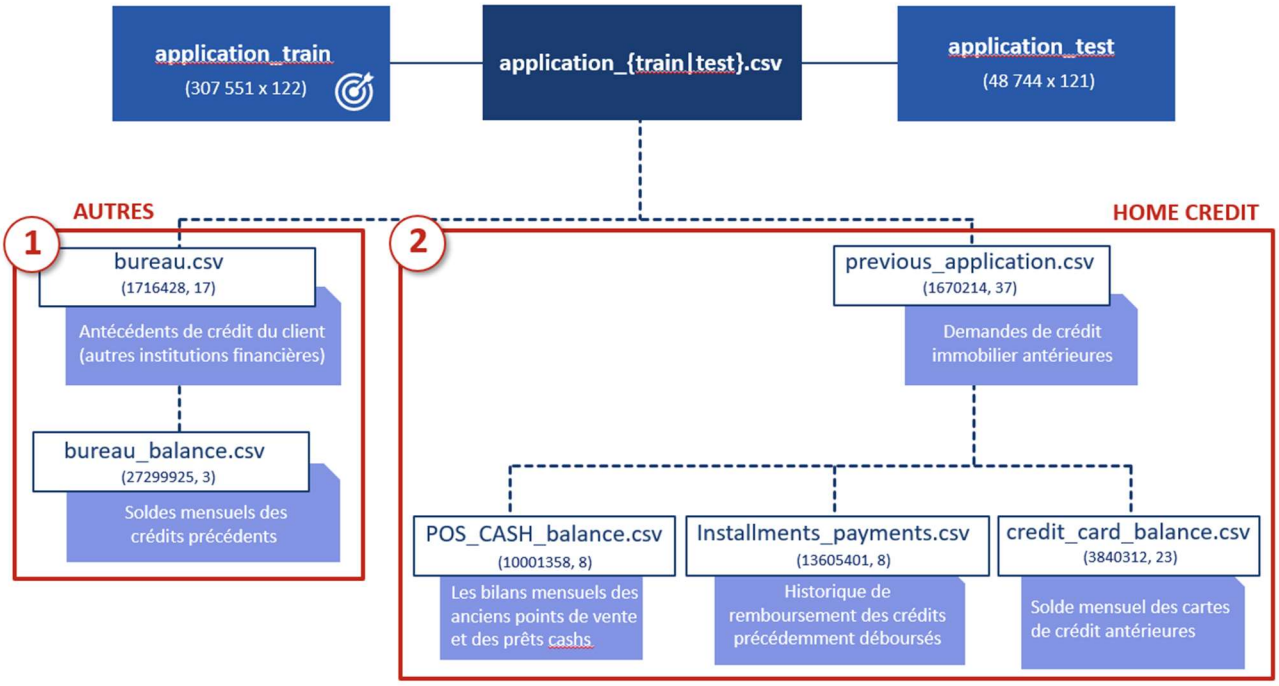


FIGURE 1 – *Arborescence des 7 tables fournies par Home Crédit*

Modèle de classification et cible

La valeur à prédire est contenue dans la variable « **TARGET** ».

Elle peut prendre 2 valeurs :

- **1** qui indique que le client n'a pas remboursé son crédit. (**Défaillant, 8%**)
- **0** qui indique que le client l'a remboursé. (**Non défaillant, 92%**)

*Ce **déséquilibre** devra être pris en compte lors de la construction du modèle, car certains algorithmes sont sensibles au déséquilibre.*

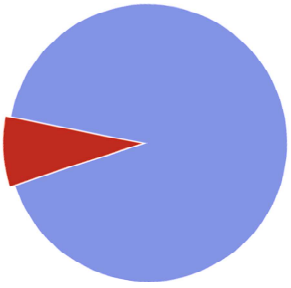


FIGURE 2 – *Cible*

Objectifs : modélisation et interprétabilité
Modèle de scoring de la probabilité de défaut de paiement d'un client
Classification binaire, Cible : **données déséquilibrées**
8 tables de données, 307 511 clients, 218 variables

2. PREPARATION DES DONNEES

Pipeline de pré-traitement appliqué aux tables

Chaque table est traitée indépendamment. Des **anomalies** détectées pendant l'analyse exploratoire des données sont corrigées. Cette analyse a permis de constater **qu'un tiers des variables totales ont 50% de valeurs manquantes**. Une double stratégie est appliquée : suppression de colonnes (NAN > 60%) et imputation selon le type de données.

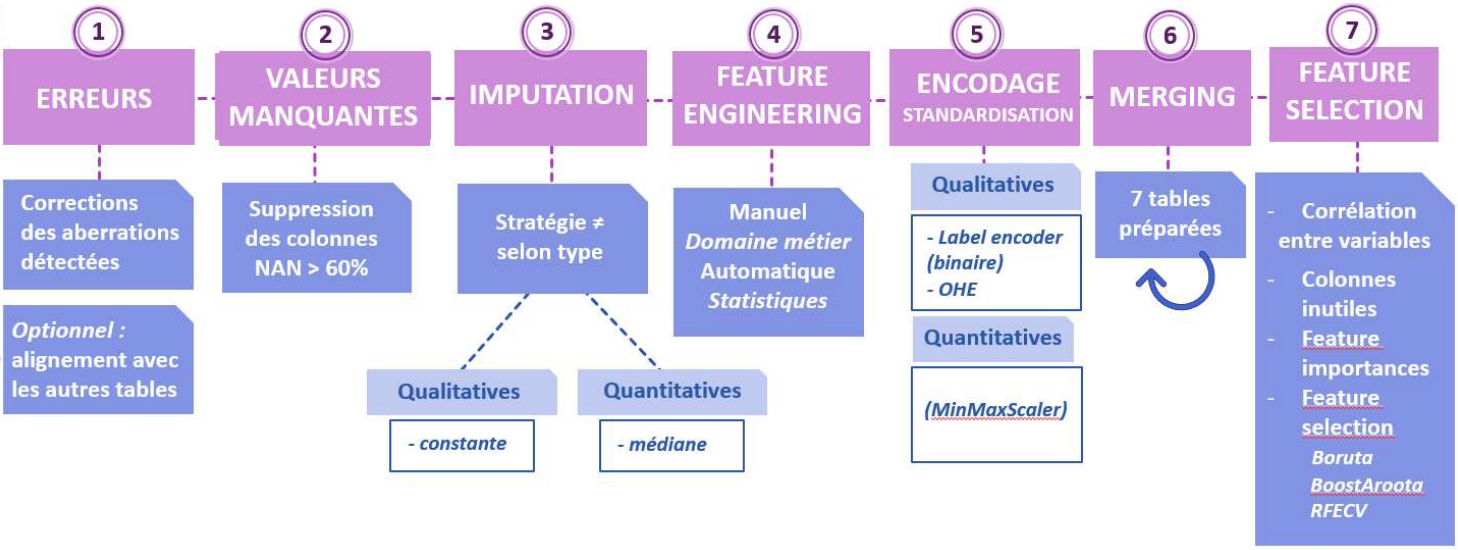


FIGURE 3 – *Pipeline de pré-traitement appliqué aux tables*

Deux processus **de création manuelle de nouvelles variables** (feature enginneering) sont appliqués :

- **Création systématique de variables statistiques** créées à partir des variables existantes : (moyenne, minimum, maximum, écart-type, compte)

- **Création de variables « métier »** basées sur les connaissances du domaine bancaire :
  - **Ratios** : montant total du crédit/revenu du client, montant crédit/ intérêt du prêt, ancienneté au travail/âge, nombre d'enfants/revenu, revenu par tête
  - **Sommes** : des 3 sources extérieures, des docs (flags) à fournir, des contacts client (adresse, téléphones, mail)
  - **Différences** : âge - ancienneté emploi
  - **Booléen** : crédit>somme demandée ? (Credit>GoodPrice)
  - **Variable liée à la cible** : moyenne de la cible pour les **500 voisins les plus proches** de chaque client en fonction de 4 variables : les 3 sources extérieures (EXT_SOURCE 1, 2 et 3) et un ratio créé par nos soins (ratio CREDIT/ANNUITY)

Les données sont ensuite encodées selon leur type et leur signification :

- **Variables qualitatives** : transformation en variables quantitatives pour les rendre exploitables
  - **Label encoder** : encodage binaire (0 ou 1) (modalités < 3)
  - **One Hot Encoder** : pour les autres variables qualitatives
- **Variables quantitatives :** mise à l'échelle : ensemble des valeurs ramenées de 0 à 1
  - **MinMaxScaler** : pour pouvoir tester nos données sur des algorithmes de machine learning sensibles aux variations d'échelle des variables lors de la phase de sélection du meilleur modèle.

Assemblage des tables et sélection de variables

Les 7 tables sont assemblées (merging). Elles contenaient initialement **218 variables** au total. Après la phase de création de variables, nous avons systématiquement supprimé les variables colinéaires à plus de 80%. Nous obtenons alors un jeu de données final de **365 variables**.

> La sélection de variables (feature selection) est un processus de sélection d'un sous-ensemble de variables qui sont les plus pertinentes pour la modélisation et l'objectif commercial du problème.

Plusieurs approches ont été utilisées pour sélectionner les variables les plus importantes :

- **Méthode de filtrage** (suppression des variables colinéaires)
- **Méthodes automatiques** utilisant des packages python : Boruta et BoostAroota
- **Méthode « embedded »** ou intégrées qui effectuent la sélection des fonctionnalités pendant la formation du modèle. Ici le modèle utilisé est LGBM non optimisé.

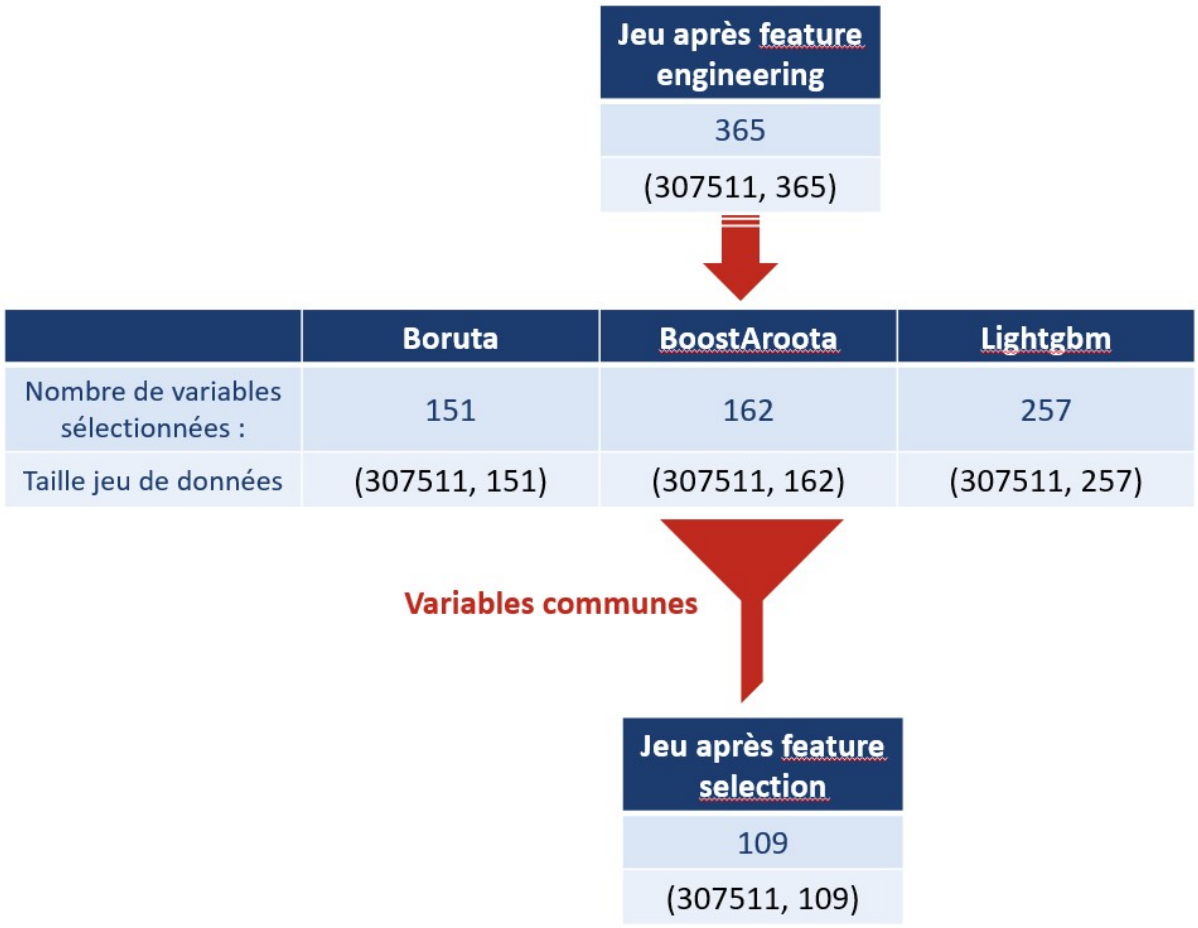


FIGURE 4 – *Sélection de variables*

Après sélection, en ne conservant que les variables communes aux 3 méthodes utilisées, le jeu de données contient **109 variables et une ligne par client.**

3. MODELISATION

Méthodologie de rééquilibrage de la variable cible

L'analyse exploratoire a montré qu'il s'agissait d'un problème de **classification déséquilibré** : le choix d'une métrique de performance est donc essentiel pour évaluer correctement les modèles.
Une modification de l'ensemble de données est possible avant d'entraîner le modèle prédictif afin d'équilibrer les données. Cette stratégie est appelée **rééchantillonnage** (re-sampling) et il existe deux méthodes principales pour égaliser les classes : le **sur-échantillonnage** : Oversampling et le **sous-échantillonnage** : Undersampling.

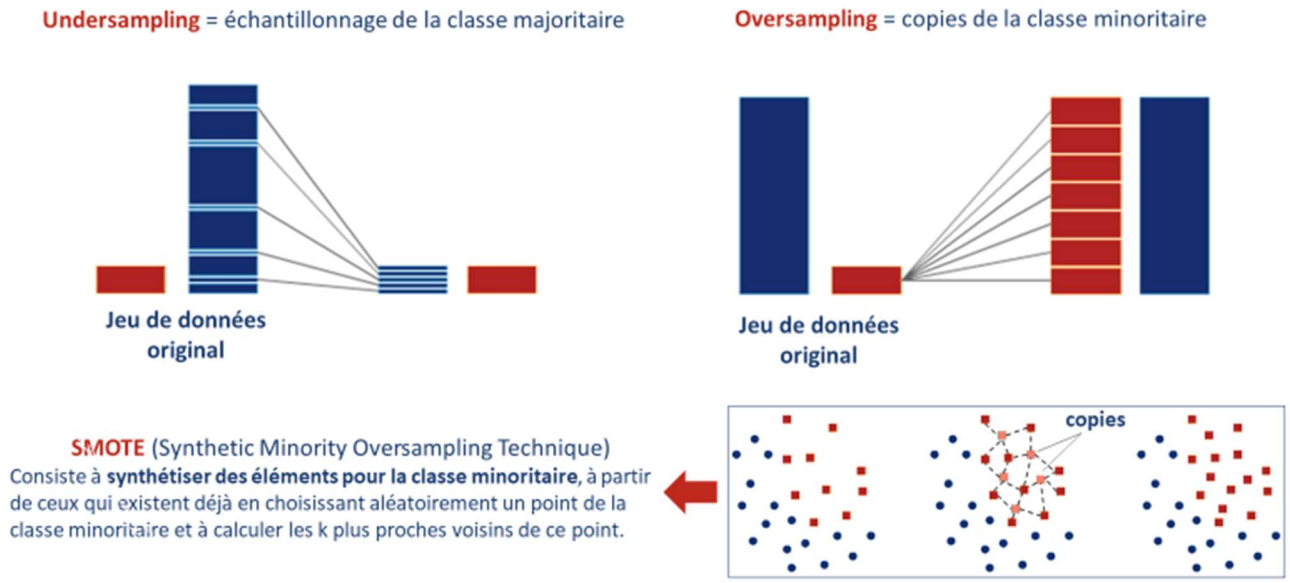


FIGURE 5 – *Stratégies de rééchantillonnage*

Une stratégie de **sur-échantillonnage** appelée **SMOTE** a été utilisée

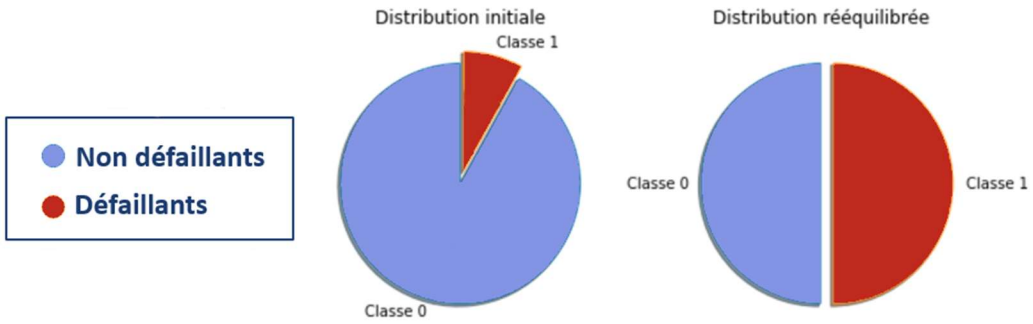


FIGURE 6 – *Sur-échantillonnage SMOTE appliqué au jeu de données*

Une métrique d'évaluation quantifie la performance d'un modèle prédictif.
La qualité d'un modèle de classification dépend directement de la métrique utilisée pour l'évaluer. Elle consiste globalement à comparer les classes réelles aux classes prédites par le modèle ou permettre d'interpréter les probabilités prédites pour ces classes.

Séparation des données

Le jeu de données est séparé en **jeu d'entrainement** (80% des données) et **en jeu de validation** (20%) en veillant à maintenir les proportions de classes 0 et 1 dans les 2 jeux. Le jeu de validation sera utilisé pour mesurer la performance des modèles testés.

Choix de l'algorithme

Nous avons testé un ensemble de modèles grâce à la librairie PyCaret : un modèle linéaire, **régression logistique**, deux modèles ensemblistes, **Random Forest et Adaboost**, et deux modèles ensemblistes « avancés » avec gradient boosting, **Catboost et LightGBM.**

La métrique utilisée est le score AUC ou « aire sous la courbe ROC ». Elle est exploitée en classification binaire pour mesurer la performance d'un modèle : plus il est performant, plus l'aire sous la courbe est maximisée.

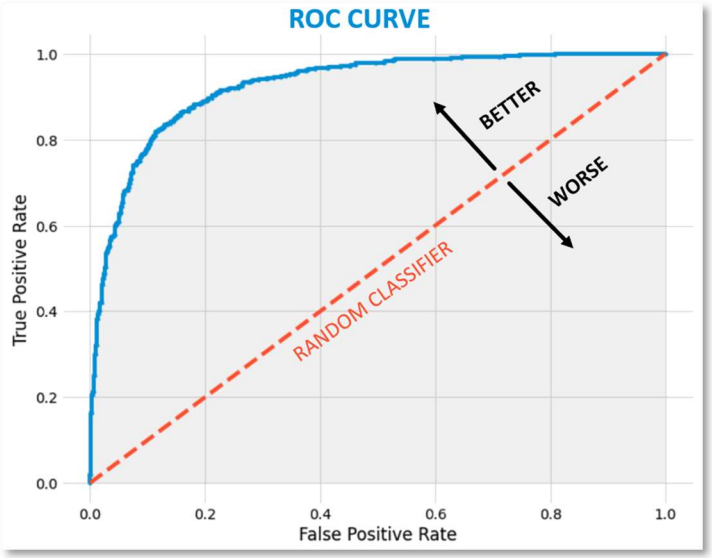


FIGURE 7 – *Courbe ROC*

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.9190	0.7828	0.0558	0.4850	0.1000	0.0848	0.1435	76.660
lightgbm	Light Gradient Boosting Machine	0.9194	0.7808	0.0267	0.5078	0.0507	0.0430	0.1023	9.635
rf	Random Forest Classifier	0.9188	0.7527	0.0102	0.3975	0.0200	0.0159	0.0529	54.897
ada	Ada Boost Classifier	0.9125	0.7406	0.0617	0.2964	0.1018	0.0763	0.1032	45.305
lr	Logistic Regression	0.6130	0.6632	0.6273	0.1243	0.2074	0.0840	0.1325	41.786

FIGURE 8 – *Tableau de comparaison d'algorithmes établi grâce à la librairie Pycaret*

Les deux algorithmes Catboost et LightGBM ont été optimisés.

C'est finalement **LightGBM**, plus rapide, qui a été retenu.

Problématique « métier » : métrique « bancaire » et seuil de solvabilité

Dans notre problème de classification binaire, le coût des **faux positifs** n'est pas le même que celui des **faux négatifs**. (erreurs de type 1 et de type 2)

Matrice de confusion		Classe prédite	
		Classe 0 : non défaillant	Classe 1 : défaillant
Classe réelle	Classe 0 : non défaillant	Vrais négatifs TN	Faux positifs FP
	Classe 1 : défaillant	Faux négatifs FN	Vrais positifs TP

FIGURE 9 – *Matrice de confusion*

Nous pouvons optimiser notre classificateur pour une **valeur seuil de probabilité** afin d'optimiser la fonction de perte personnalisée simplement en définissant le coût des vrais positifs, des vrais négatifs, des faux positifs et des faux négatifs séparément. Pour cela, il faut :

- Créer une **métrique "bancaire"** qui pénalisera les **faux négatifs,** c'est à dire les clients prédits non défaillants par le modèle et qui se révèlent en réalité défaillants. (Mauvais prêts)

Choix arbitraire de pénalisation :

- Mauvais prêts : pénalisation de -10
- Bons prêts : gain de 1

- Chercher et fixer **le seuil de solvabilité optimum** en fonction de notre métrique bancaire.

Méthode d'optimisation des hyperparamètres de LightGBM

La méthode d'optimisation des hyperparamètres choisie est **l'optimisation bayésienne.**

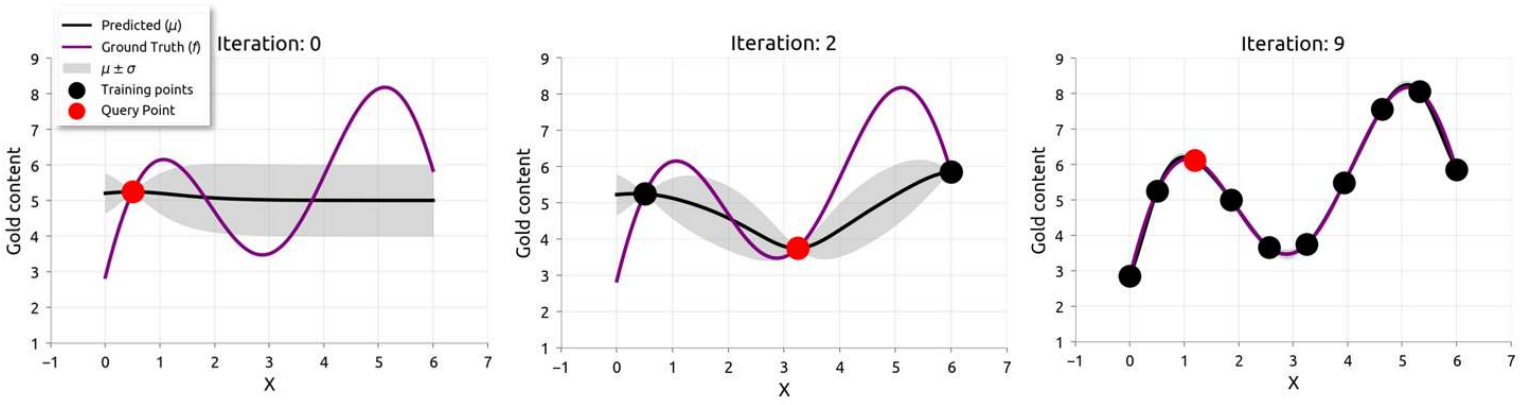


FIGURE 10 – *Optimisation bayésienne (images : https://distill.pub/2020/bayesian-optimization/)*

L'optimisation bayésienne construit un modèle de probabilité de la fonction objectif afin de proposer **des choix plus intelligents pour le prochain ensemble d'hyperparamètres à évaluer.** Au fur et à mesure que le nombre d'observations augmente, la distribution postérieure s'améliore et **l'algorithme devient plus sûr des régions de l'espace des paramètres qui valent la peine d'être explorées et de celles qui ne le sont pas.**

L'optimisation a été faite selon 2 métriques pour LightGBM :

Modèle 1 : sur la métrique **roc_auc**
Modèle 2 : **métrique « bancaire »** créée par nos soins

Et sur 2 jeux de données différents :

Jeu 1 : jeu de données sans rééquilibrage des données et sans standardisation (LGBM est insensible aux variations d'échelle) *Lire note class_weight ci-dessous.*
Jeu 2 : jeu de données obtenu après sur-échantillonnage SMOTE

Un modèle de base, réglé sur les hyperparamètres par défaut de LightGBM sert de comparatif.

1. LightGBM possède un hyperparamètre **« class_weight »** que l'on peut régler sur « balanced » afin qu'il tienne compte du déséquilibre des classes.
2. Un œil attentif sera gardé sur deux métriques : **le rappel et la précision.** Le but dans le cas d'une classification des prêts bons ou mauvais est **de maximiser le rappel** au détriment de la précision : diminuer les faux négatifs (FN) pour augmenter le rappel.

Modèle finalisé

Les deux modèles LightGBM, réglés sur les métriques AUC ou sur la métrique bancaire, sans rééchantillonnage mais avec l'hyperparamètre **class_weight='balanced'** se sont montrés les plus performants.

A noter : ces deux modèles ont un meilleur score de rappel que les modèles de base et SMOTE, mais un moins bon score de précision. Ils détectent mieux les faux négatifs, mais on observe en parallèle une augmentation des faux positifs.

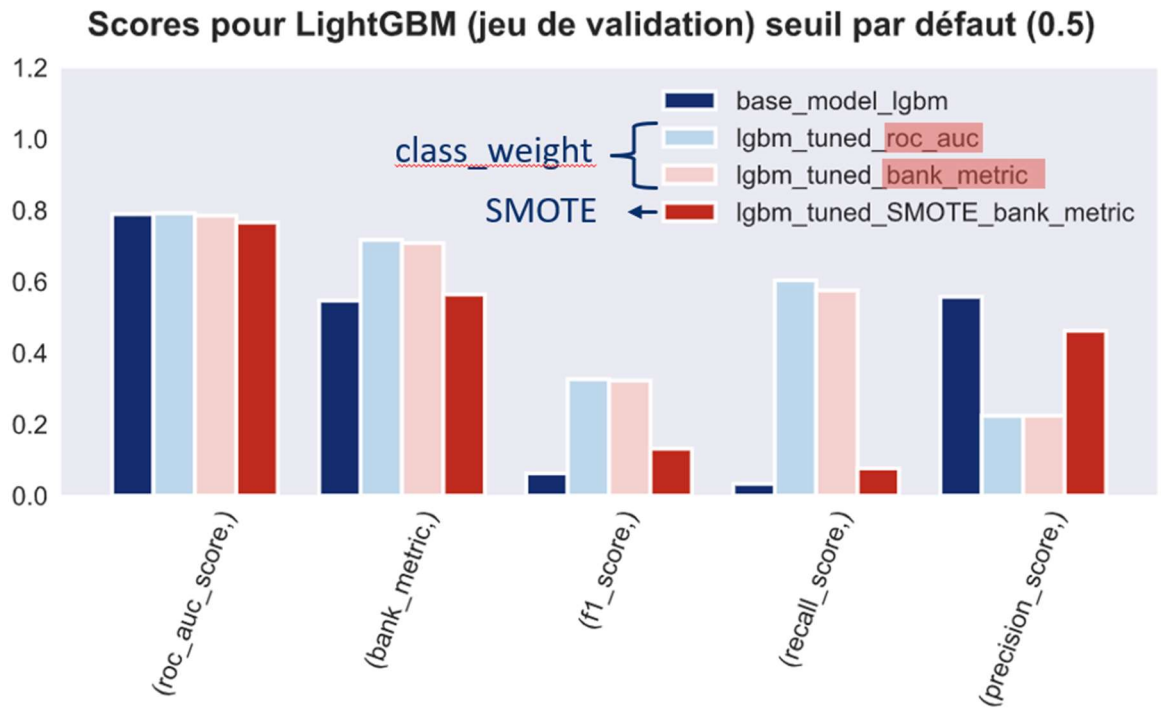


FIGURE 11 – *Scores LGBM class_weight versus SMOTE*

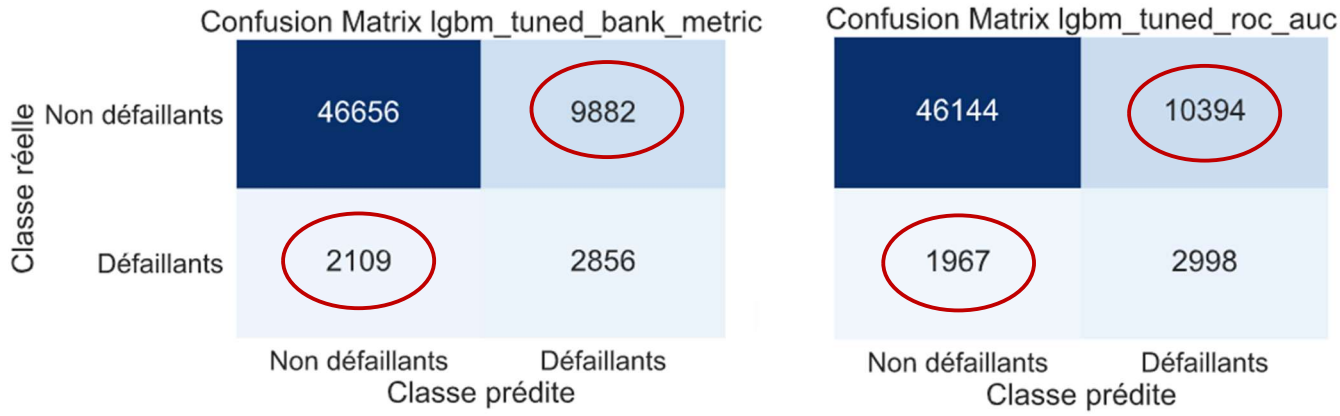


FIGURE 12 – *Matrices de confusion LightGBM class_weight, jeu de validation*

Un arbitrage est à faire, en collaboration avec notre client, pour décider du modèle à privilégier et en particulier quel équilibre optimum on doit trouver entre les faux négatifs et les faux positifs.

Le modèle LightGBM « métrique bancaire » détecte moins de faux négatifs que le modèle LightGBM « ROC_AUC ». Il prédit plus souvent que le client va rembourser un prêt alors qu'il va se révéler défaillant. **L'organisme prêteur perd ainsi la somme prêtée.**

Il est en revanche meilleur sur les faux positifs. En ne prêtant pas à des clients finalement solvables, la banque perd les intérêts que ses clients auraient versés.

Le modèle conservé est donc le modèle LightGBM optimisé sur la métrique roc_auc. La métrique bancaire sera utilisée pour chercher le seuil de solvabilité.

```
LGBMClassifier(
class_weight='balanced',
colsample_bytree=0.6685976111628331,
learning_rate=0.005,
max_depth=11,
min_child_samples=18,
min_child_weight=80,
min_split_gain=0.1,
n_estimators=10000,
num_leaves=48,
objective='binary',
reg_alpha=0.3,
reg_lambda=0.1509731738366173,
subsample=0.7257255326720535,
verbose=0)
```

FIGURE 13 – *Hyperparamètres du modèle final*

	lgbm_tuned_roc_auc
roc_auc_score	0.791848
bank_metric	0.716879
f1_score	0.326633
recall_score	0.603827
precision_score	0.223865

FIGURE 14 – *Performance du modèle final*

Fixation du seuil de solvabilité optimum

Le seuil de solvabilité optimum, en fonction de notre métrique bancaire, est de **0.41**.

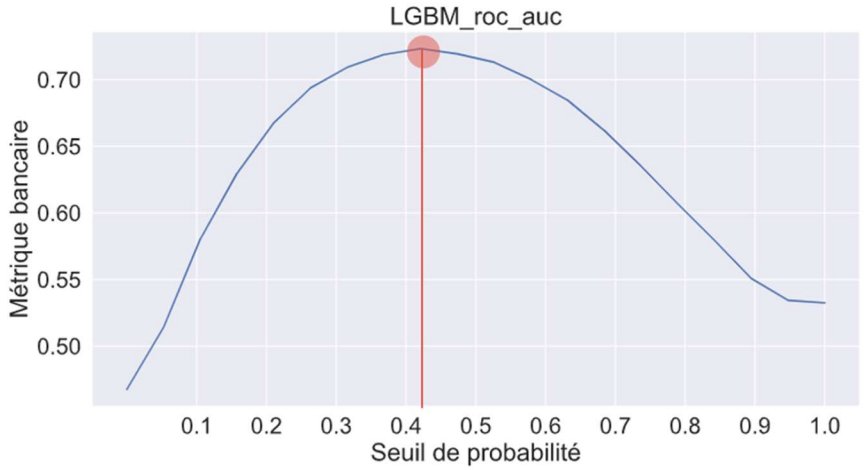


FIGURE 15 – *Seuil de solvabilité*

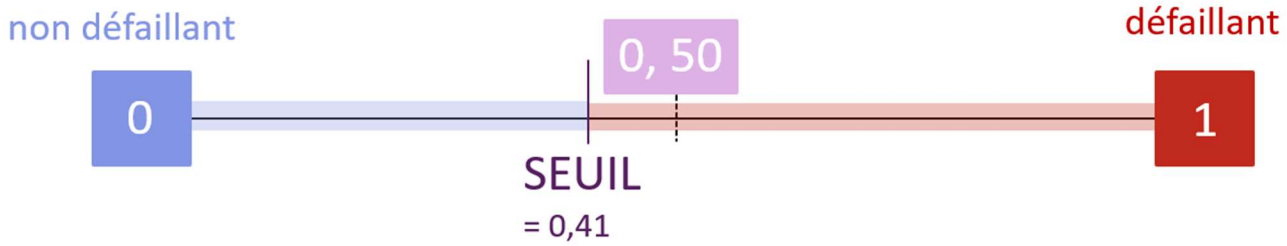


FIGURE 16 – *Exemple explicatif de la notion de seuil*

Notre modèle fait la prédiction que le client demandeur a une probabilité de 50% d'appartenir à la classe 1, c'est-à-dire d'être défaillant. En plaçant le seuil à 0.41, ce client sera prédit défaillant.

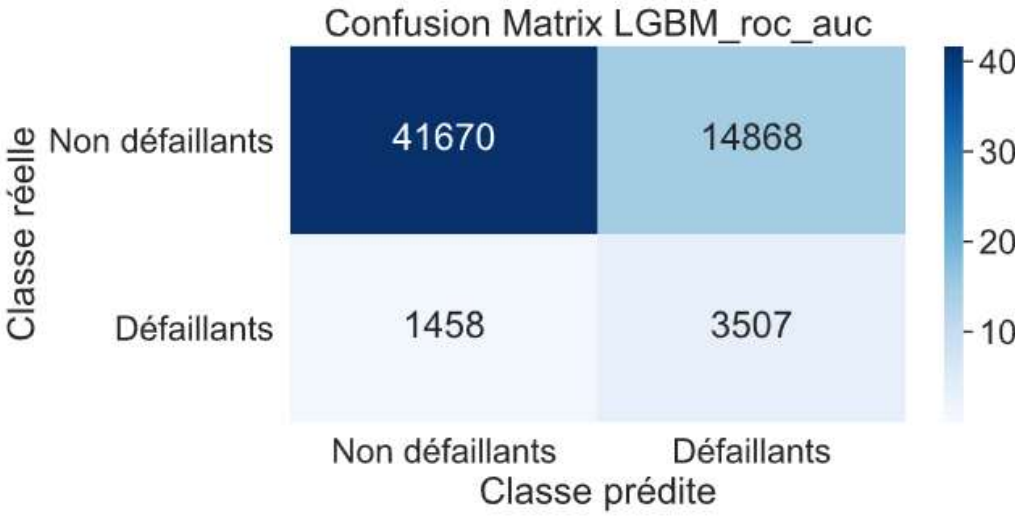


FIGURE 17 – *Matrice de confusion, modèle final, seuil=0.41, jeu de validation*

4. INTERPRETABILITE

Importance relative des variables pour le modèle

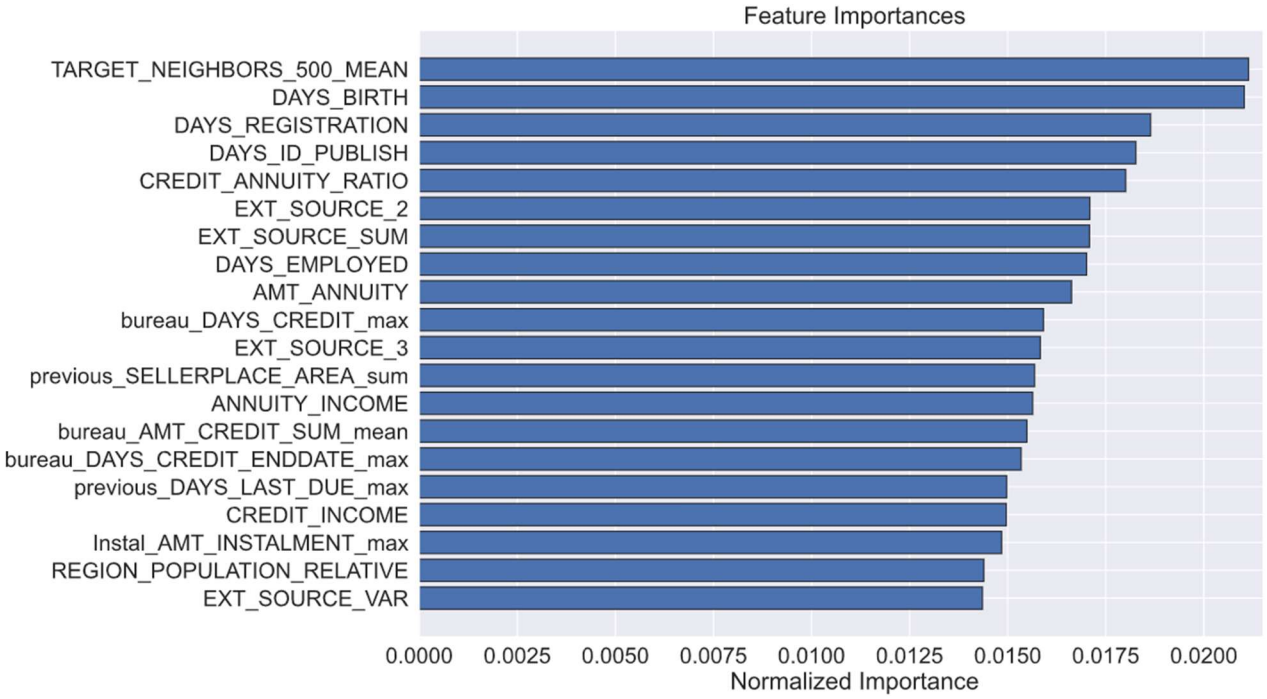


FIGURE 18 – *Les 20 variables les plus importantes pour le modèle final*

Sans surprise, la variable « TARGET_NEIGHBORS_500_MEAN » créée en calculant la moyenne de la cible pour les **500 voisins les plus proches** en fonction de 4 variables du jeu de données arrive en tête. Les variables que nous avions détectées durant l'analyse exploratoire comme fortement corrélées à la cible se trouvent également dans la liste : les variables sources extérieures (EXT_SOURCE_2, 3, VAR), les variables âge (DAYS_BIRTH), l'ancienneté (DAYS_EMPLOYED) ou le ratio montant crédit/annuité (CREDIT_ANNUITY_RATIO).

5. CONCLUSION – LIMITES ET AMELIORATIONS

Notre étude portait sur un problème de **classification binaire présentant un déséquilibre de classe.**

Nous avons mis en œuvre des stratégies propres pour optimiser le meilleur modèle et obtenir une performance maximale. Nous avons testé et comparé **différentes solutions de rééquilibrage de classe.** Nous avons **créé de nouvelles variables** en veillant à ce qu'elles restent facilement explicables et nous conformer ainsi à la demande de notre client.

Nous avons également créé une métrique métier et fixé un seuil de solvabilité optimum. Le modèle final est un LightGBM optimisé sur la métrique ROC_AUC.

Une amélioration du modèle serait envisageable en poursuivant l'optimisation des hyperparamètres et en modifiant, avec l'aide d'un expert métier, la métrique que nous avons créée : la fixation de la pénalisation des faux négatifs est arbitraire, nous pourrions la « doser » différemment. Cet expert pourrait également nous guider dans la création de variables pertinentes auxquelles nous n'avons pas pensé, malgré le soin apporté au feature engineering.

6. PRESENTATION DU DASHBOARD

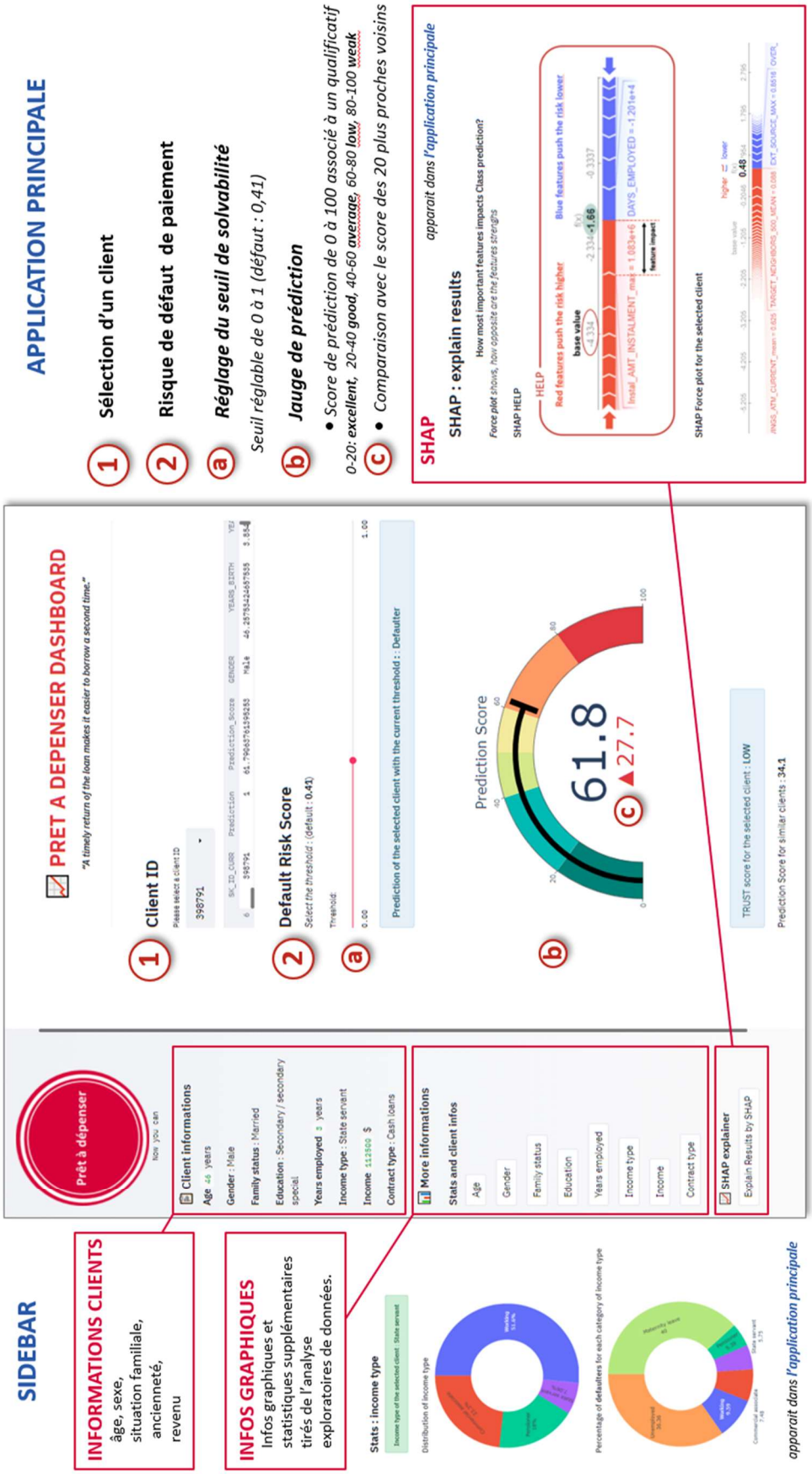


FIGURE 19 – *Tableau de bord : présentation*

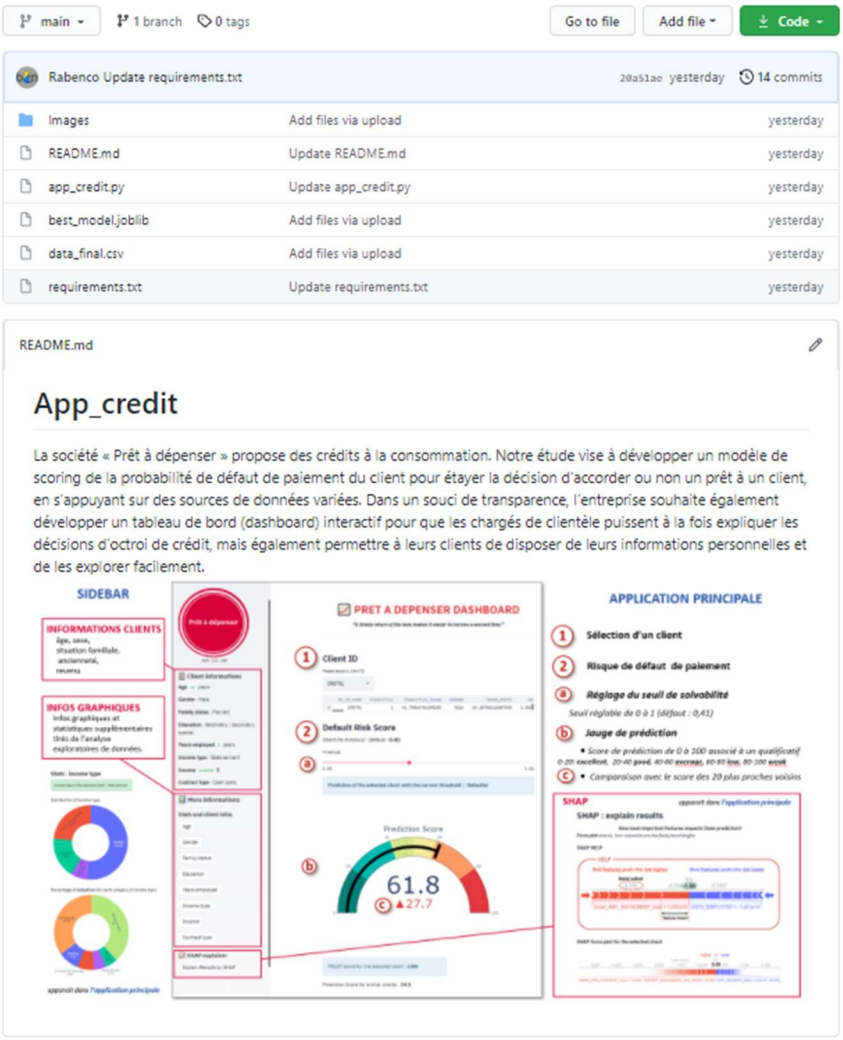
La prédiction est faite sur le jeu de données **application_test** (48744 clients) fourni par Home Credit et qui a été traité systématiquement en parallèle pendant notre étude. Il contient **les mêmes variables que le jeu utilisé pour l'entraînement du modèle,** à l'exception de la variable cible.

7. LIENS

Déploiement du dashboard

Dépôt github :

https://github.com/Rabenco/App_credit



En local :

app_credit.py

A distance :

https://share.streamlit.io/rabenco/app_credit/main/app_credit.py