Projet 7 : formation data scientist

2023 | Par : Alain KENFACK

Implémentation d’un modèle de scoring

Note méthodologique

Table des matières

Introduction

Je suis Data Scientist au sein d'une société financière, nommée "Prêt à dépenser", qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt. L'entreprise souhaite mettre en œuvre un outil de “scoring crédit” pour calculer la probabilité qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé.

Pour ce faire, je dois développer un algorithme de classification en m'appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.). De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d’octroi de crédit.

Pour répondre à cette demande, je dois également développer un dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d’octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

Ma mission est donc de :

- Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.
- Construire un Dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.
- Mettre en production le modèle de scoring de prédiction à l’aide d’une API, ainsi que le dashboard interactif qui appelle l’API pour les prédictions.

Je vais procéder de la manière suivante :

- Je vais commencer par collecter et nettoyer les données nécessaires au développement de mon modèle de scoring.
- Ensuite, je vais explorer les données pour identifier les variables les plus pertinentes pour la prédiction de la faillite des clients.
- Je vais ensuite construire plusieurs modèles de scoring et les évaluer sur un jeu de données de validation.
- Enfin, je vais mettre en production le modèle de scoring le plus performant et développer le dashboard interactif.

1. Les données

1.1. Présentation des données

Les données exploitées sont fournies par « Home Credit Group [1]» et elles datent d'il y a 5 ans. Les données sont composées de huit fichiers au format csv. Ces fichiers contiennent 218 informations bancaires et personnelles anonymisées pour 307511 clients recueillies auprès de différentes institutions financières (dont Home Credit Group). La figure ci-dessous présente la relation entre les différents fichiers (base de données).

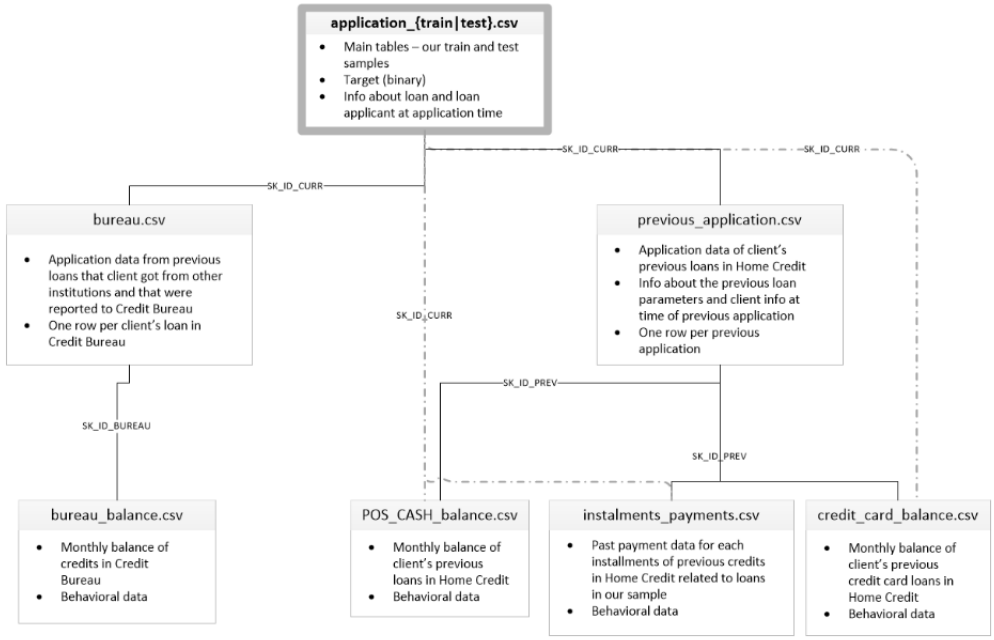
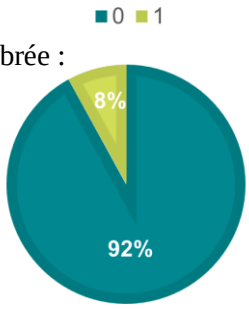


Figure 1 : relation entre les différentes bases de données

- application_train.csv et application_test.csv : Ces fichiers contiennent des données statiques sur tous les prêts, y compris le statut du prêt (accordé ou refusé). La variable Target est absente dans application_test.csv.
- bureau.csv : Ce fichier contient des informations sur les prêts antérieurs des clients auprès d'autres institutions financières.
- bureau_balance.csv : Contient les soldes mensuels des prêts antérieurs des clients auprès d'autres institutions financières.
- POS_CASH_balance.csv : Contient les soldes mensuels des prêts antérieurs des clients auprès de Home Credit.
- credit_card_balance.csv : Contient les soldes mensuels des cartes de crédit antérieures.
- previous_application.csv : Contient des informations sur les demandes de prêt antérieures.
- installments_payments.csv : Contient l'historique des remboursements des prêts antérieurs.
- HomeCredit_columns_description.csv : Ce fichier contient des descriptions des colonnes dans les différents fichiers de données.

1.2. Problématique

La variable cible à prédire (TARGET) prend 2 valeurs et est fortement déséquilibrée :



- Valeur 0 pour non défaillant :
  indique que le client a totalement remboursé son prêt.
- Valeur 1 pour défaillant :
- indique que le client n'a pas remboursé son prêt en totalité ou en partie.

Il s'agit alors d'un problème de classification binaire, dont l'objectif est d'identifier la probabilité qu'un client ne soit pas défaillant. La variable cible étant déséquilibrée, je mettrais en place des techniques de rééquilibrage des classes afin d'éviter les biais.

2. Pré-traitement des données

Chaque fichier est traité individuellement et tous les fichiers traités sont regroupés en un fichier unique ne contenant que les variables jugées les plus pertinentes.

2.1. Nettoyage

La première étape consiste à transformer le type des objets pour les rendre cohérents (ex : Y/N en 0/1) et réduire leur taille de stockage dans la mémoire.

La seconde consiste à corriger les valeurs aberrantes détectées lors de l'analyse exploratoire (EDA) [2] et d'harmoniser les valeurs uniques des données (ex : sexe Masculin, Féminin) pour les informations principales. L'analyse exploratoire a montré que plusieurs variables contiennent plus de 50% de valeurs manquantes. La double stratégie a été :

• De supprimer les variables avec plus de 68% de valeurs manquantes pour conserver les variables importantes détectées lors de l'EDA et de supprimer les variables sans information pertinente.

• De remplacer les variables conservées ayant des valeurs manquantes par la valeur médiane pour toutes les variables numériques et par la valeur la plus utilisée pour les variables qualitatives.

2.2. Feature engineering

Cette étape consiste à créer de nouvelles variables qui peuvent augmenter la performance du modèle :

- Création de nouvelles variables à partir de la moyenne, le minimum, le maximum…
- Division ou soustraction des caractéristiques importantes pour obtenir des taux (comme l'annuité et le revenu).
- Encodage unique pour les variables catégorielles. Deux types d'encodage ont été utilisé :
  - "label_encoder", pour encoder les variables catégorielles binaires en utilisant l'encodage de type label.

- Et "one_hot_encoder", pour l'encodage one-hot des autres variables catégorielles.

Après le feature engineering, tous les fichiers sont joints en utilisant la clé "SK_ID_CURR "(Identifiant du client) et en suivant la relation entre les différents fichiers tel que décrite par la figure 1.

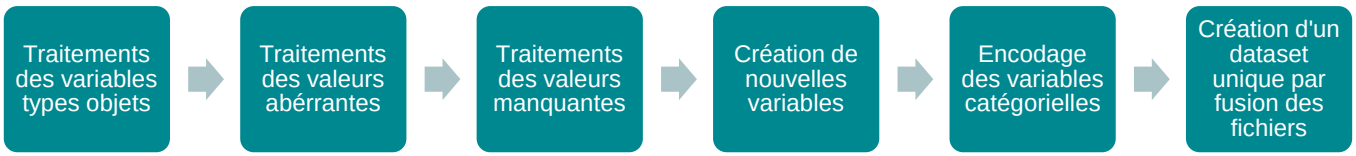


FIGURE 2 : RESUME DU PROCESSUS DE PRE-TRAITEMENT DE

3. Modélisation

3.1. Choix des métriques

Les métriques permettent de comparer les classes réelles aux classes prédites par le modèle. L'objectif de la banque est de réduire au maximum les pertes d'argent. A cet effet, le modèle devra :

- Ne pas prédire un client non-défaillant s'il est défaillant dans la réalité. Ce qui signifie minimiser le nombre de faux négatifs (erreur de type II). Dans ce cas, l'entreprise perdra toute la somme prêtée à l'emprunteur.
- Minimiser le nombre de faux positifs (erreur de type I [client prédit défaillant alors que non-défaillant dans la réalité]). Dans ce cas, les pertes sont minimes, car Home Credit aura seulement perdu les intérêts de la somme qu'il aurait prêté à l'emprunteur.

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

FIGURE 3 : MATRICE DE CONFUSION

L'objectif premier est donc de pénaliser plus fortement les Faux Négatifs. On peut alors créer une métrique métier en attribuant un coefficient à chaque catégorie de la matrice de confusion :

- FN_{coeff} (coefficient des faux négatifs) = -10
- TP_{coeff} (coefficient des vrais positifs) = 0
- TN_{coeff} (coefficient des vrais négatifs) = 1
- FP_{coeff} (coefficient des faux positifs) = 0

Ainsi, la métrique métier est défini par l'équation suivante :

$$gain = \frac{\mathrm{TP} * \mathrm{TP}_{\mathrm{coeff}} + \mathrm{TN} * \mathrm{TN}_{\mathrm{coeff}} + \mathrm{FP} * \mathrm{FP}_{\mathrm{coeff}} + \mathrm{FN} * \mathrm{FN}_{\mathrm{coeff}}}{(\mathrm{TN} + \mathrm{FP} + \mathrm{FN} + \mathrm{TP})}$$

Il sera donc question de maximiser cette métrique pour que l'entreprise puisse faire du bénéfice. De ce fait, les prêts accordés aux individus qui ne sont finalement pas solvables sont dotés d'une pénalisation négative de -10, alors que les prêts accordés aux individus finalement solvables rapportent 1. Ce rapport 10 est totalement arbitraire et il est tout à fait possible de changer ces valeurs à la convenance de l'optique métier. Il faudra cependant relancer l'optimisation des hyperparamètres du modèle.

J'ai utilisé d'autres métriques pour évaluer les performances des modèles, en plus de cette métrique métier, afin de sélectionner le meilleur. Il s'agit de :

- **Fbeta score** : cette métrique reflète l'importance de bien identifier les prêts non remboursés. C'est une généralisation de la F-measure qui permet d'ajuster la balance entre précision et rappel. La précision est le pourcentage de prédictions positives correctes, tandis que le rappel est le pourcentage de vrais positifs correctement identifiés. Une valeur bêta < 1 donne plus de poids à la précision, tandis qu'une valeur bêta > 1 donne plus de poids au rappel. Dans le cas de la prédiction de la probabilité de faillite d'un client, il est important de minimiser les faux négatifs. En effet, il est préférable de refuser un prêt à un client qui est susceptible de le rembourser plutôt que d'accorder un prêt à un client qui est susceptible de le défauter. Par conséquent, nous donnons plus de poids au rappel dans le calcul de l'F-beta score.
- **ROC AUC score** : mesure de façon globale la performance d'un modèle de classification. Il indique à quel point le modèle est capable de faire la distinction entre les classes. C'est cette métrique que j'ai utilisée pour sélectionner le meilleur modèle à optimiser.

3.2. Rééquilibrage de la variable cible (TARGET)

Le déséquilibre entre les classes peut affecter négativement les performances des modèles qui auront tendance à prédire la classe majoritaire (donc client non défaillant). Grâce à la technique de rééchantillonnage (re-sampling), il est possible de corriger ce déséquilibre entre les classes.

Pour rééquilibrer les classes, je me suis servi de la librairie « imbalanced-learn [3] » qui m'a permis de tester et comparer 2 techniques de rééquilibrage : le sur-échantillonnage (Oversampling) et le sous-échantillonnage (Undersampling). La technique de gestion du déséquilibre « class weight (balanced) [4]» a été également testée.

3.3. Séparation des données en jeu d'entraînement et de validation

Le jeu de données est séparé en deux :

- Un jeu d'entraînement, contenant 75% des données et servant à entraîner le modèle,
- Et un jeu de validation (25% des données) permettant d'évaluer la performance des différents modèles testés.

***Remarque** : lors de la séparation, les 2 jeux de données devront conserver la répartition de départ des classes majoritaires (les clients non défaillants) et minoritaires (les clients défaillants).*

3.4. Choix du modèle

Pour modéliser ce problème de classification, j'ai comparé les performances de trois algorithmes :

- Régression logistique : cet algorithme génère une fonction linéaire qui prédit la probabilité d'une observation d'appartenir à une classe donnée.
- Random Forest : cet algorithme est une forêt d'arbres de décision. Il est considéré comme un algorithme robuste et performant, notamment pour les problèmes de classification.
- LightGBM : cet algorithme est une extension de l'algorithme de Gradient Boosting Machine. Il est connu pour sa vitesse et sa précision.

Pour évaluer les performances de ces algorithmes :

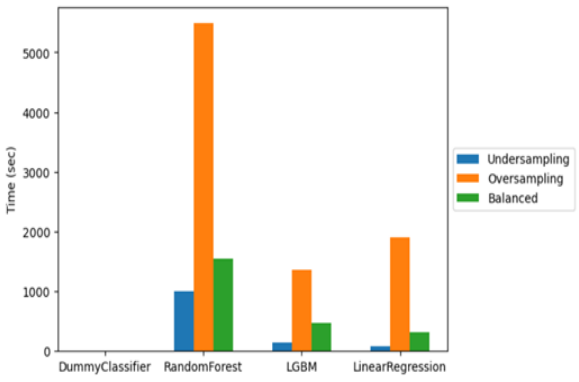
- J'ai utilisé une baseline naïve. Cette baseline est une instance de la classe DummyClassifier de la librairie Pandas, qui prédit systématiquement la classe la plus fréquente. Ainsi un modèle ne peut être performant que si ces résultats sont supérieurs à ceux de cette baseline.
- Les modèles ont été évalué à l'aide d'une validation croisée (5 folds). Cette méthode consiste à diviser le jeu de données en 5 parties, puis à entraîner et évaluer chaque modèle sur 4 parties et à tester sa performance sur la partie restante.
- Enfin, j'ai utilisé la GridSearch pour optimiser les hyperparamètres de chaque modèle. Cette méthode consiste à explorer un espace de paramètres pour trouver les valeurs qui maximisent la performance du modèle.

Comparaison des modèles

Tableau 1 : comparaison des algorithmes

	Algorithm	Balancing_method	AUC	AUC_test	Time
0	DummyClassifier	Undersampling	0.500000	0.500000	0.016010
1	DummyClassifier	Oversampling	0.500000	0.500000	0.031136
2	DummyClassifier	Balanced	0.502000	0.499000	0.020933
3	RandomForest	Undersampling	0.743000	0.746000	996.595219
4	RandomForest	Oversampling	0.827000	0.658000	5484.126693
5	RandomForest	Balanced	0.736000	0.739000	1551.318408
6	LGBM	Undersampling	0.772000	0.774000	150.958268
7	LGBM	Oversampling	0.754000	0.756000	1354.525093
8	LGBM	Balanced	0.773000	0.773000	467.790962
9	LogisticRegression	Undersampling	0.750000	0.752000	80.515491
10	LogisticRegression	Oversampling	0.739000	0.743000	1909.953394
11	LogisticRegression	Balanced	0.751000	0.754000	318.560488

Figure 4 : comparaison du temps d'exécution du fit



Les algorithmes Light Gradient Boosting Machine et RandomForest associé à la stratégie de rééquilibrage undersampling donne les meilleurs résultats sur ce jeu de données. Cependant, Light Gradient Boosting Machine consomme moins de ressource, c'est donc lui le meilleur modèle.

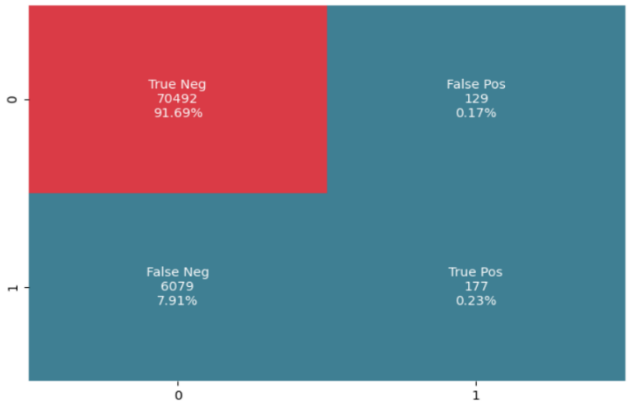


Figure 5 : Matrice de confusion du modèle LGBM

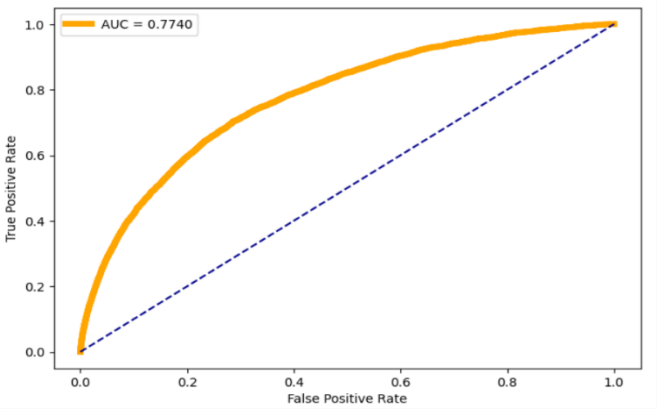


Figure 6 : Courbe AUC du modèle LGBM

4. Optimisation du modèle

L'Optimisation du modèle s'est faite en utilisant la métrique métier défini précédemment pour ajuster les hyperparamètres du modèle LGBM. Le résultat obtenu (présenté ci-dessous) est satisfaisant.

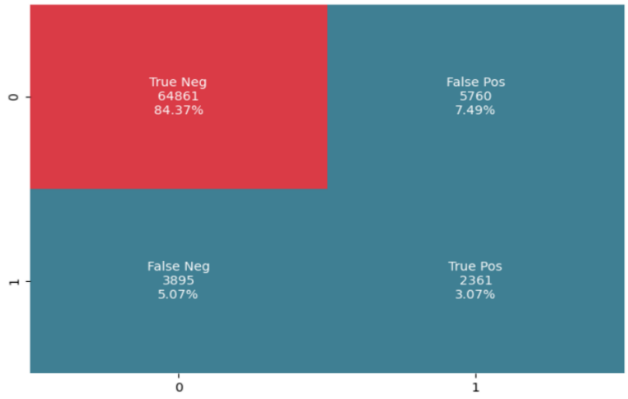


Figure 7 : Matrice de confusion après optimisation

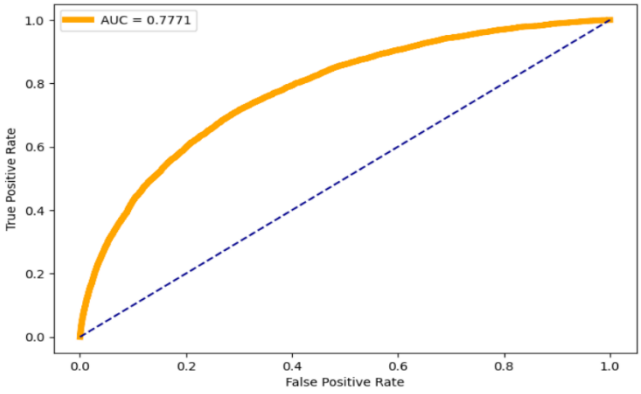


Figure 8 : courbe AUC après optimisation

On peut remarquer une réduction du nombre de « Faux Négatifs » après optimisation.

5. Interprétabilité du modèle

La librairie « shap » a permit d'expliquer le niveau d'impact des variables sur le résultat du modèle LGBM optimisé au niveau global (pour l'ensmble du jeu de données) et au niveau local (pour un client donné)

5.1. Interprétabilité globale

Je commence l'analyse d'importance, d'un point de vue global, en utilisant « summary_plot ». Ce type de tracé agrège les valeurs SHAP pour toutes les variables et l'ensemble des clients. Ensuite, ces valeurs SHAP sont triées, de sorte que la première affichée soit la variable la plus importante.

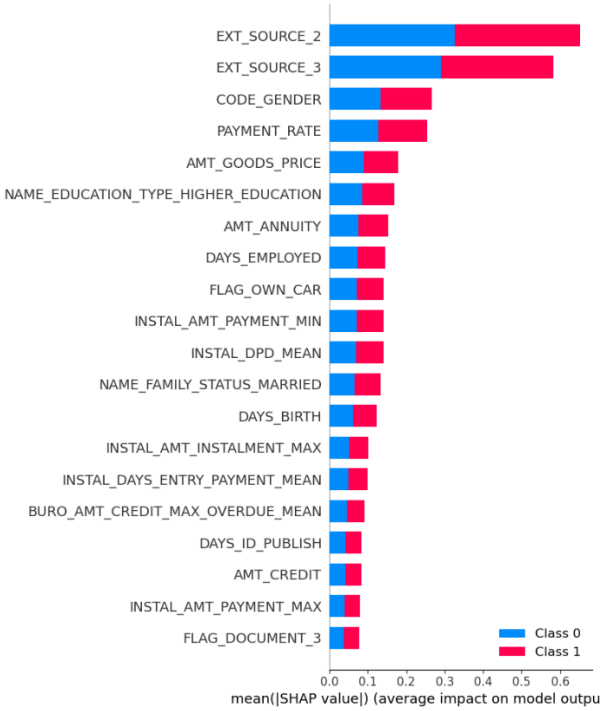


Figure 9 : Valeurs SHAP globale

5.2. Interprétabilité locale

D'un point de vue local, je me suis intéressé à l'impact des variables pour la décision du modèle par rapport à un client choisit. Dans le graphe ci-dessous, on peut voir l'impact de chacune des variables pour le client d'index 4.

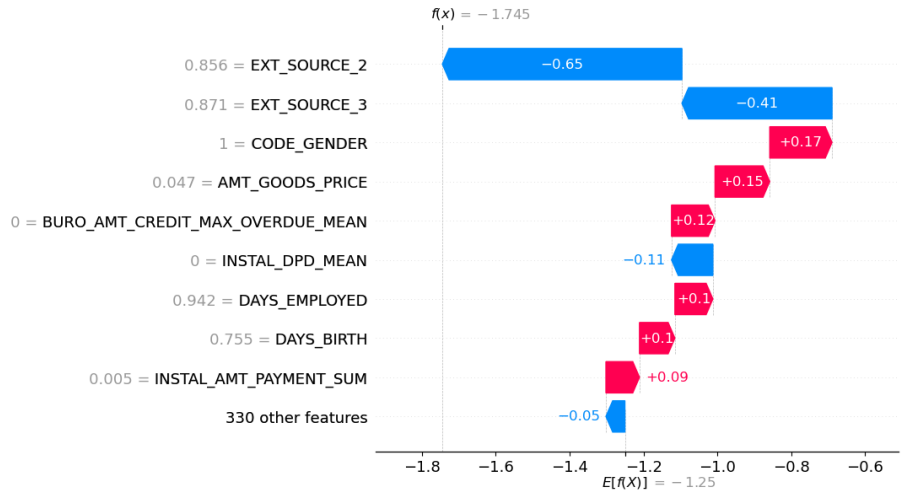


Figure 10 : Valeurs SHAP locale (index 4)

6. Tableau de bord

Après optimisation, le modèle de scoring a été mis en production à l'aide d'une API et d'un tableau de bord interactif. L'API a été développé avec l'outil FastAPI [5] et le tableau de bord avec Dash [6]. Ces deux applications ont été déployés sur la plateforme « Heroku ».

Figure 11 : Interface de l'api

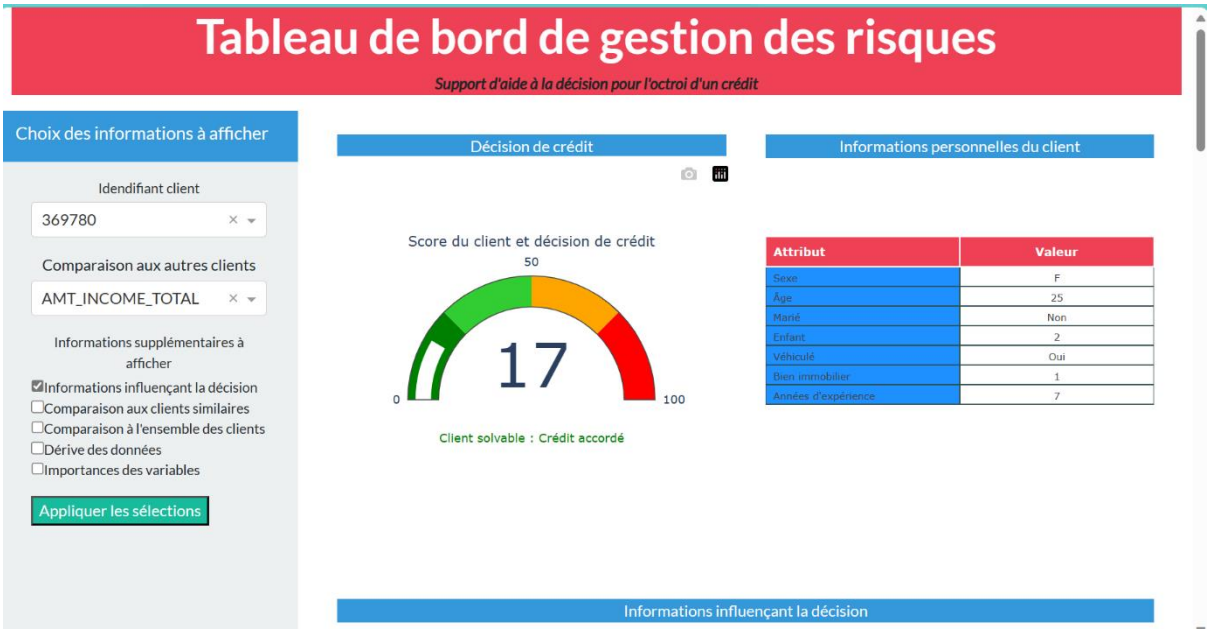


Figure 12 : Aperçu du tableau de bord

7. Axes d’amélioration

La modélisation repose principalement sur la création d'une métrique d'évaluation spécifique au secteur, qui accentue la pénalisation des Faux Négatifs (FN - c'est-à-dire lorsque de bons clients sont incorrectement prédits comme mauvais clients, entraînant ainsi l'octroi de crédits et des pertes en capital). Cependant, il pourrait être plus approprié de travailler en étroite collaboration avec les équipes métier pour concevoir et utiliser une métrique potentiellement plus adaptée. De la même manière, les équipes métier pourraient également participer au processus de création de variables pertinentes dans le contexte du feature engineering.

J'ai fait le choix de ne tester que 3 algorithmes. Cependant, avec des ressources matérielles plus importantes, il serait possible de tester simultanément l'ensemble des algorithmes de classification en utilisant, par exemple, la bibliothèque « Lazy Predict [7]», ce qui permettrait d'identifier l'algorithme le mieux adapté à la problématique.

Références

[1] «Home Credit Group,» 2018. [En ligne]. Available: https://www.kaggle.com/competitions/home-credit-default-risk/data.

[2] R. RAO, «kaggle,» 2020. [En ligne]. Available: https://www.kaggle.com/code/rishabhrao/home-credit-default-risk-extensive-eda/notebook. [Accès le septembre 2023].

[3] The imbalanced-learn developers, «imbalanced-learn documentation,» [En ligne]. Available: https://imbalanced-learn.org/stable/index.html. [Accès le Septembre 2023].

[4] A. Kumar, «Analytics Yogi,» 1 Avril 2023. [En ligne]. Available: https://vitalflux.com/class-imbalance-class-weight-python-sklearn/. [Accès le septembre 2023].

[5] MIT License, «MIT License,» [En ligne]. Available: https://fastapi.tiangolo.com/.

[6] Plotly, «Dash Python User Guide,» [En ligne]. Available: https://dash.plotly.com/.

[7] S. R. Pandala, «Lazy Predict,» 2022, [En ligne]. Available: https://lazypredict.readthedocs.io/en/latest/#.