Francesco Bianco

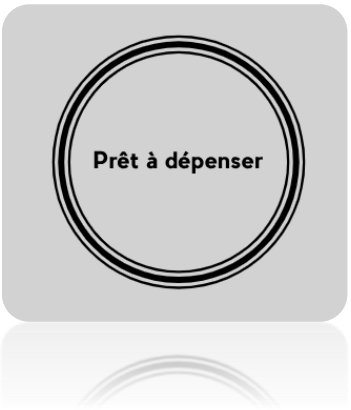
Implémentez un modèle de scoring

(formation Data Scientist – projet # 7)

Note métodologique

1. Préambule

Vous êtes 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. Elle souhaite donc développer un algorithme de classification en s'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. Cette demande de transparence des clients va tout à fait dans le sens des valeurs que l'entreprise veut incarner.

Prêt à dépenser décide donc de 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.

(source : https://openclassrooms.com/fr/paths/164/projects/632/assignment)

2. Méthodologie d'entraînement du modèle

2.1. Jeu de données

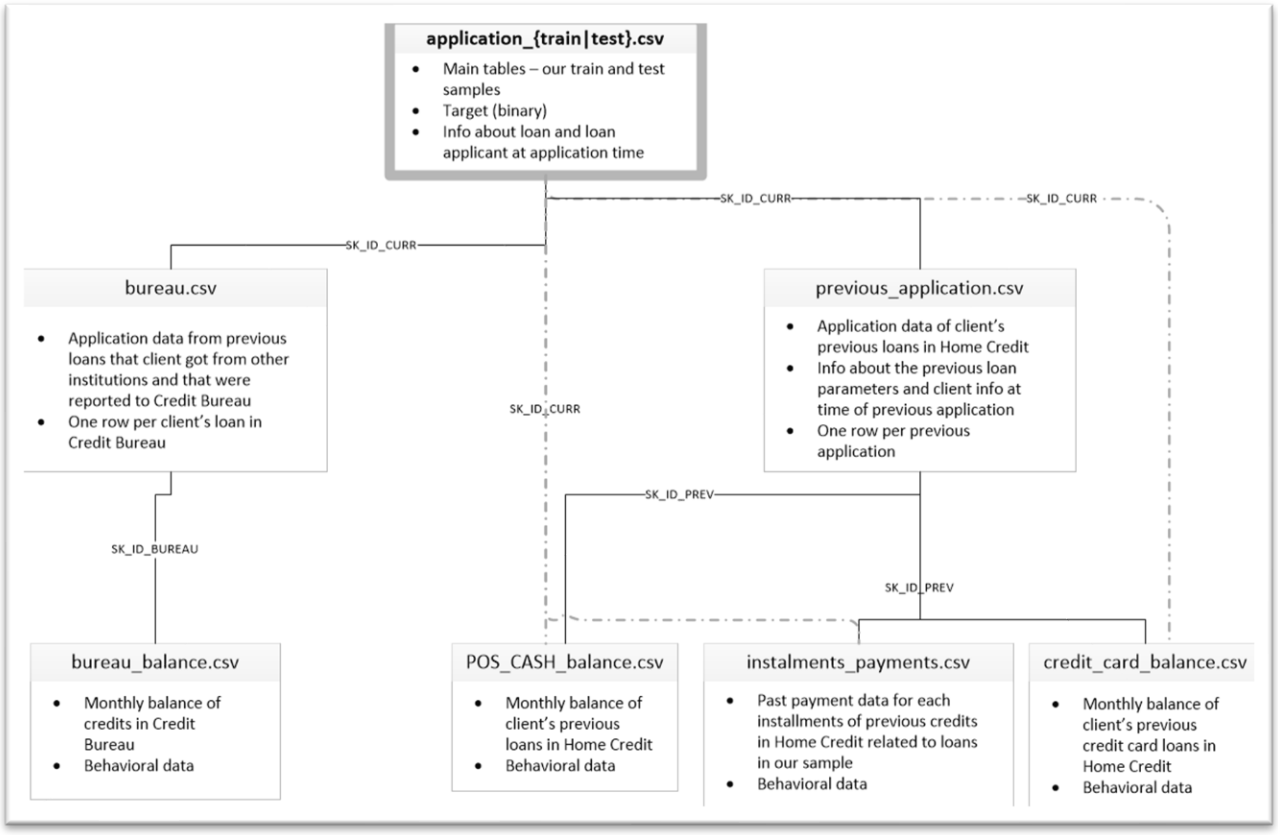
Le jeu de données employé pour l'entraînement et le test du modèle provient de la plateforme Kaggle. Il est téléchargeable de la page https://www.kaggle.com/c/home-credit-default-risk/data, où sont également décrits les contenus des différents tables (sous forme de fichiers .csv) ainsi que leurs rélations:

application_{train|test}.csv est la table principale, dont chaque ligne represènte un client. Les données de cette table sont réparties en deux fichiers, l'un (destiné à l'entraînement/test du modèle en locale) avec, l'autre (destiné à la mesuration de la performance du modèle sur Kaggle, dans le cadre de la competition) sans cible (colonne TARGET).

Autres tables/fichiers, à joindre à la table principale: *bureau.csv*, *bureau_balance.csv*, *POS_CASH_balance.csv*, *credit_card_balance.csv*, *previous_application.csv*, *installments_payments.csv*.

Les colonnes des différentes tables sont décrites dans le fichier *HomeCredit_columns_description.csv*.

Voici le schema des jointures avec lesquelles on a fusionné les différentes tables dans un dataframe/fichier unique (réparti en jeu d'entraînement et jeu de test[1]):



(source : https://www.kaggle.com/c/home-credit-default-risk/data)

[1] Le jeu de test dont on parle ici est celui qui contient les données du fichier *application_train.csv*. Celui-ci étant sans cible (colonne *TARGET*), on l'a utilisé en tant que jeu de données pour l'application web (données de production). L'entrainement et les tests des modèles ont été réalisés, donc, a partir des données du fichier *application_train.csv*, répartis dans un jeu de train et un jeu de test.

2.2. Kernels Kaggle

Pour le nettoyage et l'analyse des données, on s'est inspiré des kernels Kaggle suivants :

- https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction/notebook : notebook très clair et simple, basé sur la table principale. On s' y est inspiré surtout pour concevoir l'analyse des données.
- https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script : script qui utilise toutes les tables. On l'a exploité surtout pour la jointure et le nettoyage des données.

2.3. Nettoyage et exploration

- **Nettoyage** : recherche des doublons, traitement des valeurs manquantes, gestion des valeurs aberrantes (valeurs négatives et infinies, valeurs temporelles, etc.) encodage des variables qualitatives ;
- **Exploration** : corrélation entre variables et cible, creation de variables polynomiales et variables du domaine bancaire.

Pour une description exhaustive des opérations de nettoyage et d'exploration, cf. les kernels Kaggle et le notebook de modelisation).

2.4. Modèlisation

On a essayé les modèls suivants :

- **Dummy Classifier** : classificateur aléatoire (tous les clients sont considérés en tant que fiables), utilisé en tant que base de référence (angl. *baseline*);
- **Regression Logistique** : modèle linéaire classique ;
- **Forêt aléatoire** : modèle basé sur un ensemble d'arbre de décision, dont le vote majoritaire est choisi en tant que résultat de la prediction finale ;
- **Light Gradient Boosting Machine** : modèle basé sur le gradient boosting (donc, lui aussi, sur les arbres de décision), qui combine une qualité des performances élevée avec une exécution assez rapide.

Les modèles ont été essayés sur trois versions du jeu de données :

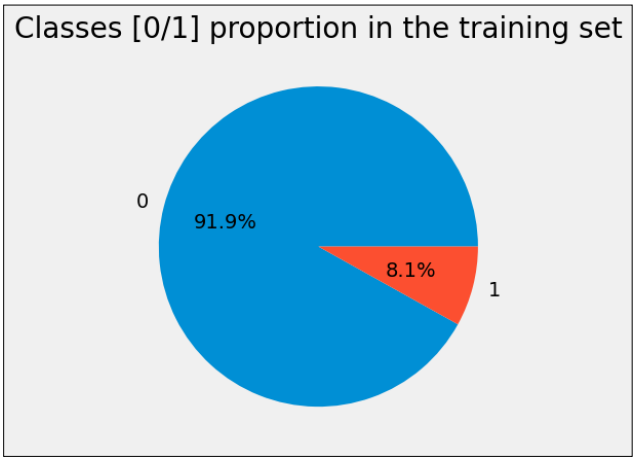
- **Standard** : contenant que les variables des tables d'origine ;
- **Polynomial** : avec les variables polynomiales ;
- **Domain** : avec les variables du domain.

Chaque version du jeu de données a été testée sans ou avec application d'APC au 80% (pour en reduire les dimensions).

3. Traitement du déséquilibre des classes

Lors de la phase d'entraînement des modèles on a du gérer le fort déséquilibre des classes dans le jeu de données, où que le 8% des observations appartiennent à la classe 1 (clients qui n'arrivent pas à rembourser ponctuellement leurs crédits).

Pour cela, on a mise en place des modèles qui gèrent cette déséquilibre : lors de l'instanciation de la regression logistique et du Light GBM, on a choisi *balanced* en tant que valeur de l'hyperparamètre *class_weigth*. Quant à la forêt aléatoire, dont l' implémentation de *ScikitLearn* ne permette pas ce type de choix, on a exploité la classe *BalancedRandomForestClassifier* d'*ImbalancedLearn*, librairie specialisée pour ce type de problematiques.



4. Fonction coût métier, l'algorithme d'optimisation, métrique d'évaluation

4.1. Fonction coût métier

Dans notre domain, les faux négatifs (dorénavant FN) et les faux positifs (dorénavant FP) ont des coûts très différents:

- **FN** (mauvais client prédit bon client) : crédit accordé, qui entraîne une perte en capital ;
- **FP** (bon client prédit mauvais) : crédit refusé, qui entraîne un manque à gagner en marge.

Le rapport entre la coût d'un FN et celui d'un FP est de **10 à 1**. En considération de cela, on a cherché à adopter une métrique qui nous permette d'évaluer les préstations de nos modèles de façon pertinente. Notamment, on a implémenté une métrique métier qu'on a appelé *Petitta score*, d'apres le nom de l'auteur principale d'un article qui aborde l'évaluation des résultats d'analyses de jeux données déséquilibrés provenant de plusieurs domaines, dont l'économie et la médecine (Petitta, F. *et al.*, A., *Tools and metrics for Data-driven Operational Knowledge (DOK)*, in «Central European Journal of Applied Mathematics and Statistics», forthcoming), qui nous a inspiré.

Le principe du Petitta score, dans notre mise en œuvre, est assez simple : le nombre de FN et celui de FP sont multipliés par des coefficients (ou *poids* ; angl. *weights*) différentes (dans notre cas: 10 et 1). Le score finale est ensuite calculé de la façon suivante (où MCP est le max coût possible, c'est-à-dire le coût si tous les clients du jeu de donnés sont des FN : longueur du dataset * 10): MCP – (FN * 10 + FP) / MCP. On obtient une valeur toujours comprise entre 0 et 1, dont la croissance corresponde à l'amélioration de la performance.

4.2. Optimisation des modèles

Une première optimisation des modèles est obtenue par bais d'une grille d'hyperparamètres, dans le cadre d'une validation croisée.

Cette validation croisée étant un processus assez lourd et cronophage, on ne l'a appliqué qu'au test de chaque modèle sur le jeu de donnée standard sans APC. Ensuite, on a retenu les résultats (les paramètres optimales ainsi obtenus) pour les réutiliser, tels quels, lors des tests suivants (avec les autres versions du jeu de donnée, sans ou avec APC).

Ensuite, une fois le modèle définitif choisi, on a testé le meilleur seuil (entre 0.0 et 1.0) de décision entre les deux classes, en rapport au score métier obtenu.

4.3. Métrique d'évaluation

En plus que le score métier (Petitta score), on a considéré, dans la comparaison des modèles, les outils d'évaluation suivants :

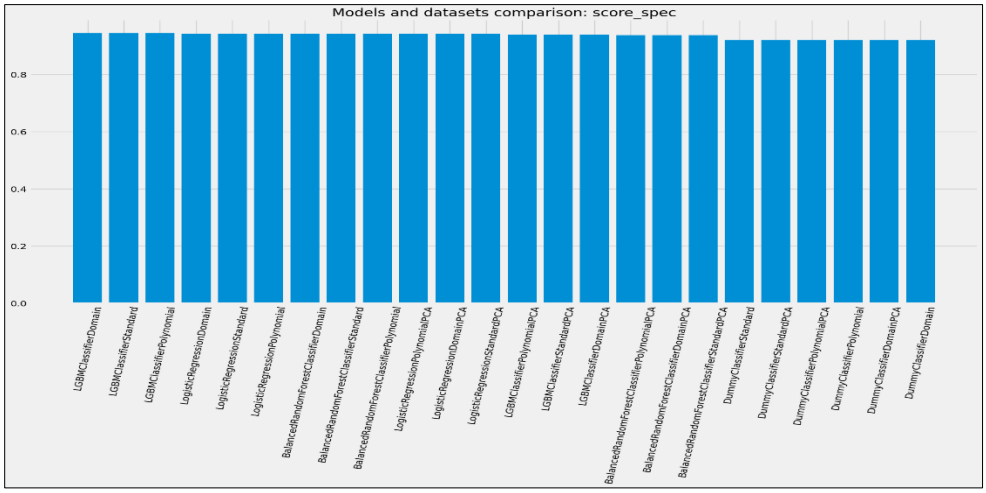
- **exactitude** (angl. *accuracy*) ;
- **matrice de confusion** ;
- **courbe ROC** ;
- **temps d'exécution**.

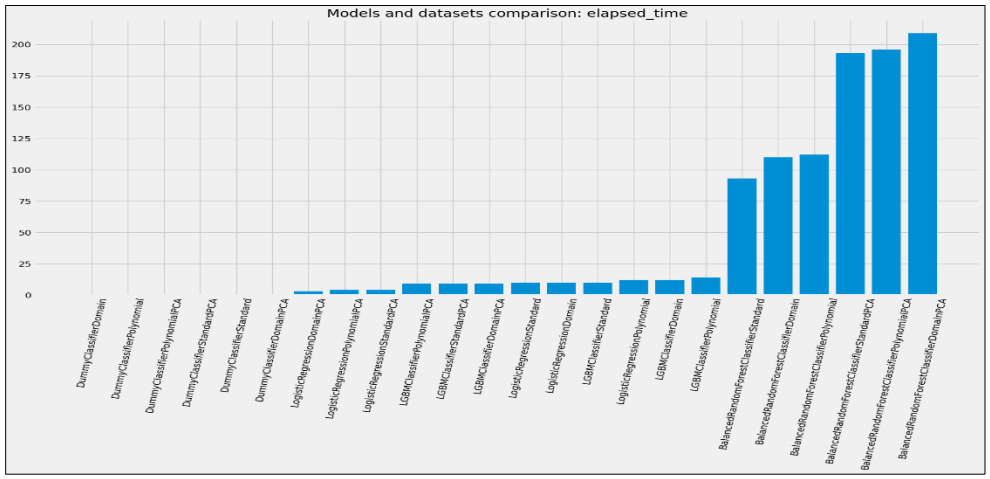
5. Synthèse des résultats

	model_name	dataset_type	pca	score_spec	accuracy	roc_auc	elapsed_time
23	LGBMClassifier	domain	False	0.9452	0.6957	0.7396	12.0
3	LGBMClassifier	standard	False	0.9452	0.6948	0.7403	10.0
21	LGBMClassifier	polynomial	False	0.9450	0.6955	0.7408	14.0
13	LogisticRegression	domain	False	0.9441	0.6724	0.7331	10.0
1	LogisticRegression	standard	False	0.9440	0.6723	0.7323	10.0
11	LogisticRegression	polynomial	False	0.9439	0.6719	0.7323	12.0
18	BalancedRandomForestClassifier	domain	False	0.9426	0.6741	0.7203	110.0
2	BalancedRandomForestClassifier	standard	False	0.9423	0.6738	0.7197	93.0
16	BalancedRandomForestClassifier	polynomial	False	0.9421	0.6689	0.7179	112.0
10	LogisticRegression	polynomial	True	0.9420	0.6578	0.7177	4.0
12	LogisticRegression	domain	True	0.9417	0.6582	0.7166	3.0
9	LogisticRegression	standard	True	0.9416	0.6588	0.7162	4.0
20	LGBMClassifier	polynomial	True	0.9400	0.6787	0.7020	9.0
19	LGBMClassifier	standard	True	0.9397	0.6788	0.7029	9.0
22	LGBMClassifier	domain	True	0.9396	0.6813	0.7024	9.0
15	BalancedRandomForestClassifier	polynomial	True	0.9384	0.6559	0.6892	196.0
17	BalancedRandomForestClassifier	domain	True	0.9384	0.6540	0.6893	209.0
14	BalancedRandomForestClassifier	standard	True	0.9382	0.6528	0.6880	193.0
0	DummyClassifier	standard	False	0.9200	0.9200	0.5000	0.0
4	DummyClassifier	standard	True	0.9200	0.9200	0.5000	0.0
5	DummyClassifier	polynomial	True	0.9200	0.9200	0.5000	0.0
6	DummyClassifier	polynomial	False	0.9200	0.9200	0.5000	0.0
7	DummyClassifier	domain	True	0.9200	0.9200	0.5000	0.0
8	DummyClassifier	domain	False	0.9200	0.9200	0.5000	0.0

Sur la base de ces résultats, on a choisi, en tant que modèle final, la regression logistique : elle nous offre, en effet, une bonne combinaison entre qualité des performance et vitesse, ce qui peut jouer un rôle important en vue du déploiement de l'application de prédiction sur le web.

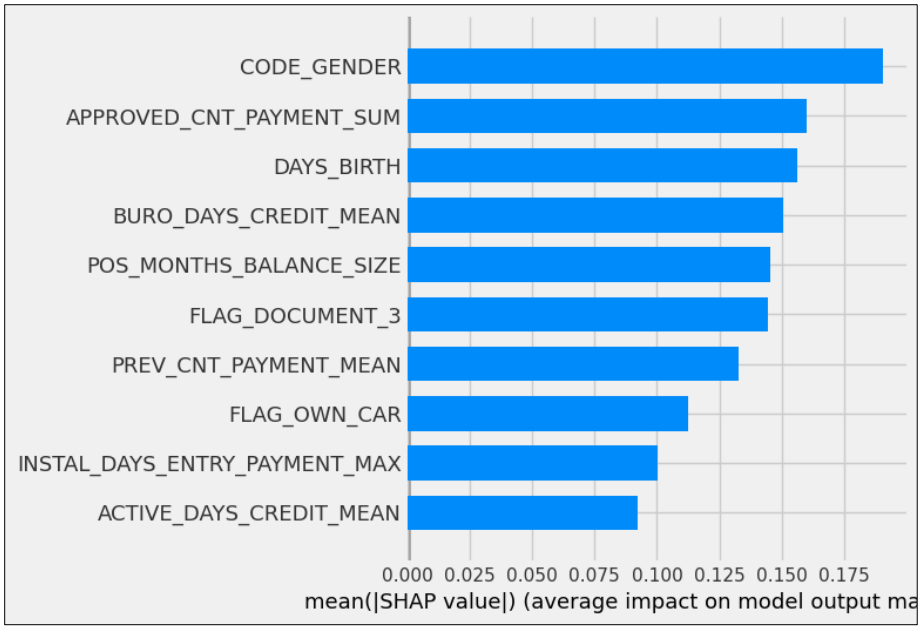
Pour plus d'info sur la comparaison des modèles, cf. le notebook de modelisation.



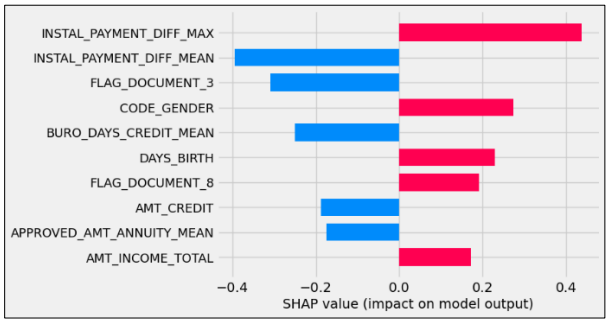


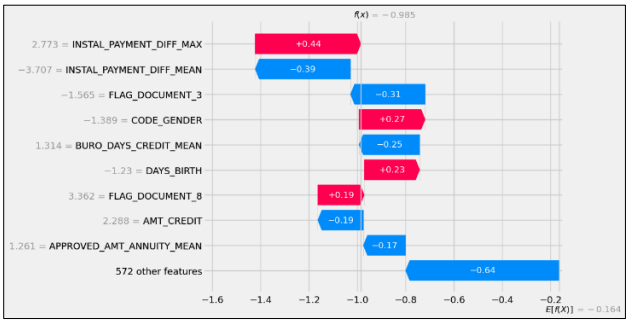
6. Interprétabilité globale et locale du modèle

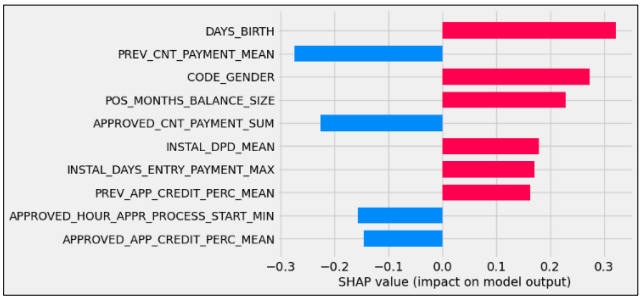
L'importance des variables sur les prédictions a été mesuré par biais de la librairie SHAP. Voici la liste des 10 premiers variables par importance globale:

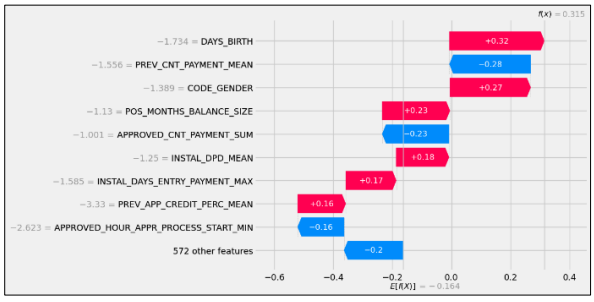


Quant à l'importance locale, on présente les résultats sur deux individus, dont l'un est classifié en tant que 0 (bon client), tandis que l'autre est classifié en tant que 1 (mauvais client):









A première vue, on remarque que certaines variables jouent un rôle important autant au niveau global qu'au niveau local. Parmi eux, le genre (*CODE_GENDER*) et l'age (*DAYS_BIRTH*). La gestion du traitement de ces variables, pour éluder des bias cognitifs et éviter la possible discrimination de catégories de candidats aux prêts, sera l'un des objectifs des prochaines étapes du travail.

7. Analys du Data Drift

Le possible dérive de donnée a été mesurée par biais de la librairie *Evidently* sur un échantillon de 2000 observations. D'après le rapport d'*Evidently* (https://github.com/biancof/oc_bank_scoring/blob/main/data_drift_report.html), aucune dérive de donnée n'a été detectée.

Dataset Drift

Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5

583	50	0.0858
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 8.576% of columns (50 out of 583).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
target	cat			Not Detected	Z-test p_value	0.527089
prediction	cat			Not Detected	Z-test p_value	0.527089
col_0	num			Not Detected	Z-test p_value	1
col_100	num			Not Detected	Z-test p_value	1
col_102	num			Not Detected	Z-test p_value	1
col_104	num			Not Detected	Z-test p_value	1

8. Limites et améliorations possibles

Celui qu'on vient de présenter, ainsi que le deploiement du modèle final sur le web (URL de l'application : http://bankscoring.streamlit.app; URL du dépôt sur GitHub : https://github.com/biancof/oc_bank_scoring/), n'est qu'un travail préliminaire pour la mise en production d'un système de prédiction des payements des clients ayant demandé un prêt.

Nombreuses possibilités d'amélioration sont encore envisageables. Parmi les plus importants, on peut lister les suivantes :

- hérités des deux kernels Kaggle, le **nettoyage** et l'**analyse des données** sont assez profonde et détaillés sur les colonnes de la table principale, alors qu'ils deviennent plus grossièrs sur les données des autres tables. Pousser le nettoyage et l'analyse de ces tables peut nous fournir une connaissance plus profonde de notre jeu de données et mieux nourrir nos modèles afin d'en améliorer les performances ;
- on n'a pas testé la combinaison des **variables de domain et polynomiales** : on pourrait bien le faire ; de même, on pourrait tester plusieurs niveaux de **réduction de dimensions** par biais de l'APC ;
- au niveau d'**optimisation**, on peut (a) optimiser les hyperparamètres des modèles en les testant sur des datasets autres que celui standard et (b) une fois le modèle final choisi, refaire une optimisation plus poussé sur celui-ci, encore par biais de *GridSearchCV* ;
- utiliser les résultats de l'analyse de l'**importance des variables** pour (a) réduire les dimensions du jeu de donnée et (b) prêter une attention particulière aux variables les plus importantes, afin d'éviter erreurs, bias cognitifs, etc. De plus, on n'a pas encore abordé, lors de cette analyse, la correlation entre les variables, qui peut aussi nous aider pour la mise en place du jeu de donné idéal par rapport à notre objectif ;
- ajouter d'**autres métriques** (précision, rappel, score F1, etc.) ;
- mettre en œuvre des **mesures de sécurité** de l'application web notamment quant aux aux échanges de données entre l'API et le tableau de bord.