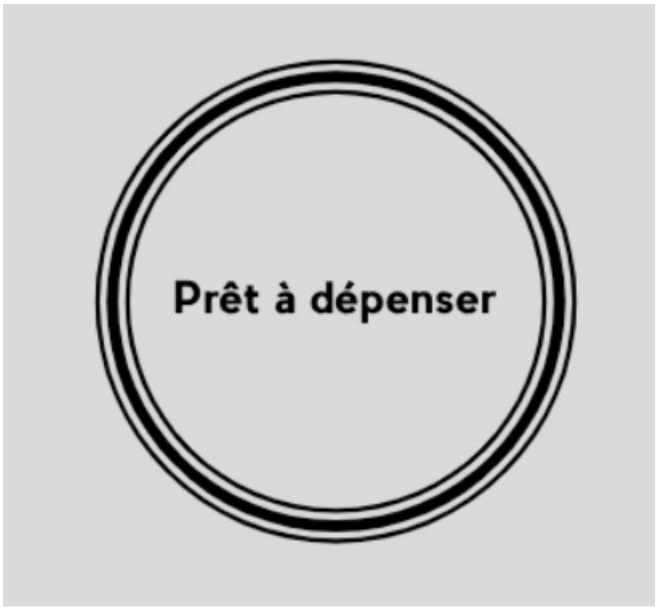
Note Méthodologique

Implémentez un modèle de scoring



Meltem YILDIRIM

Table des matières

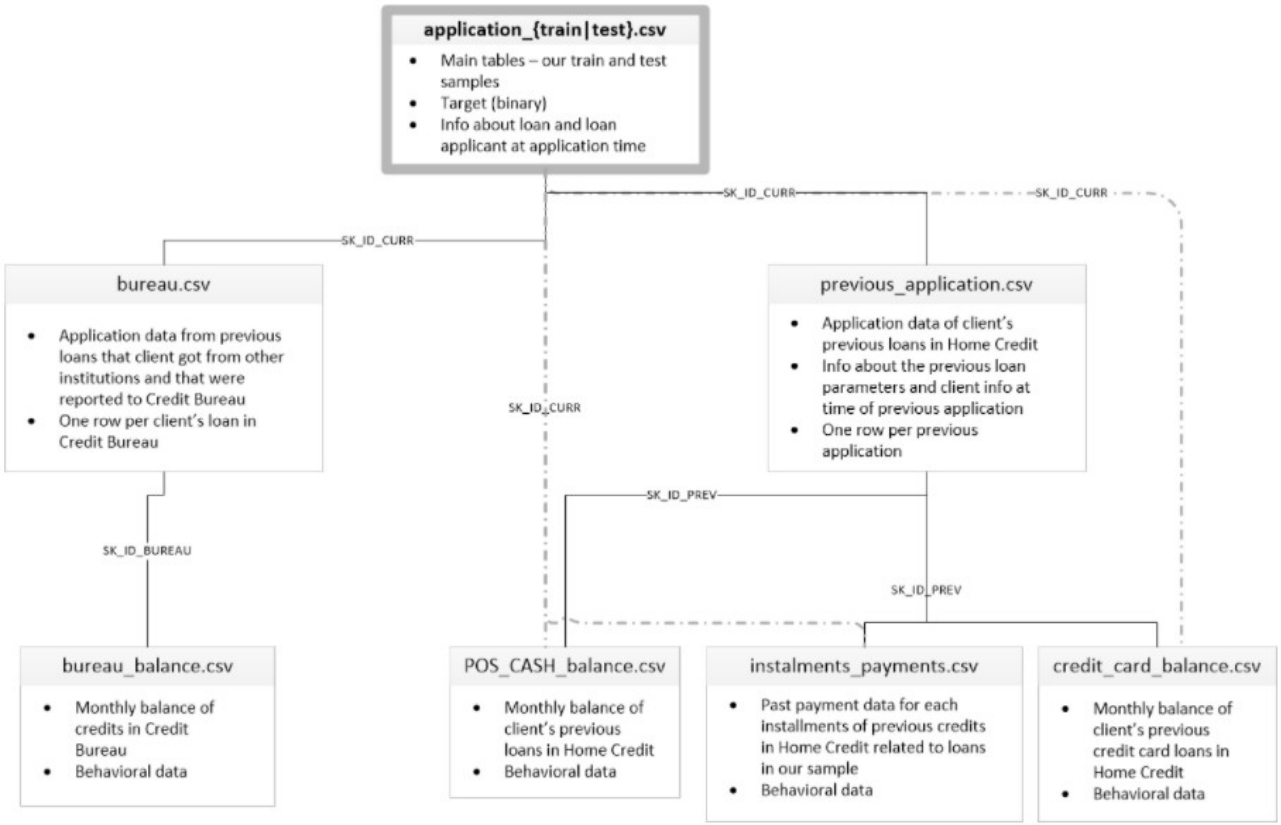
I. Contexte et objectif

La société financière « Prêt à dépenser », qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt, souhaite développer un modèle de scoring de la probabilité de défaut de paiement du client. La classification des clients s'appuiera sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

Par la suite, un dashboard interactif sera développé 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.

II. Jeu de données

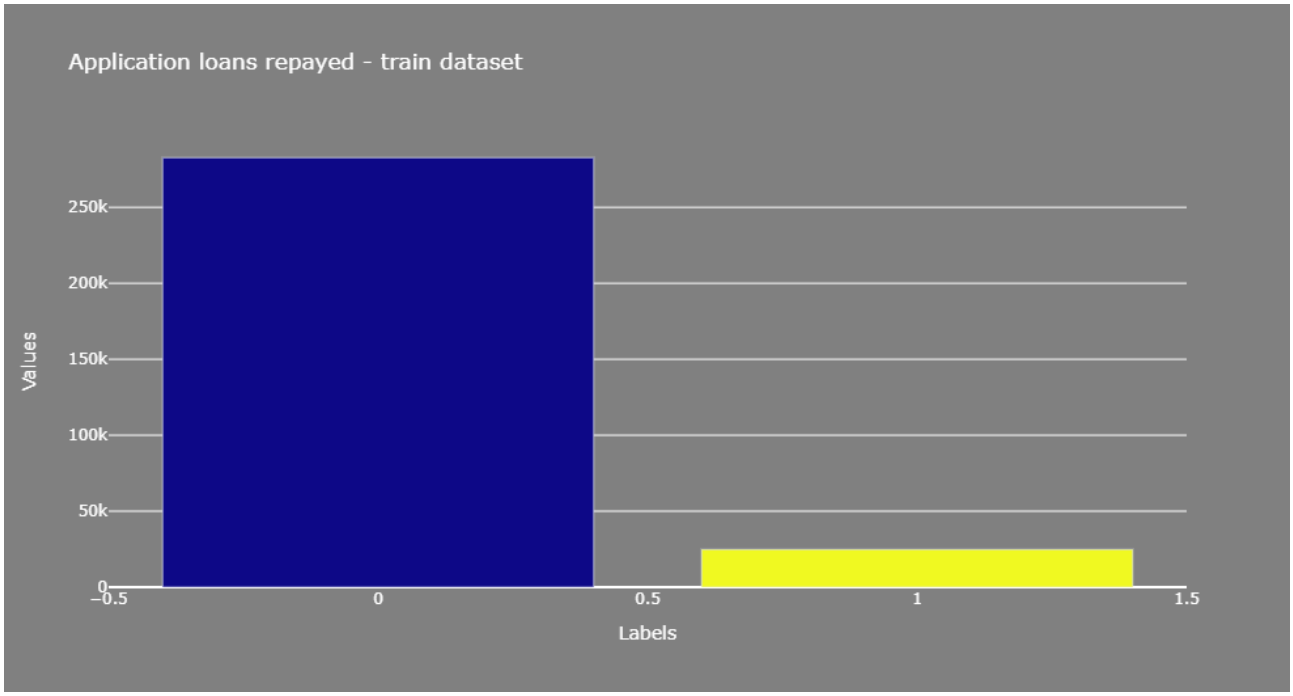
Pour réaliser ce projet, les données fournies par Home Credit, un service dédié à l'octroi de crédits à la population non bancarisée, ont été utilisées.
Il existe 8 sources de données différentes qui peuvent être jointes selon certains features.



La partie exploration se concentre plutôt sur le jeu de données *application_train*.

Le jeu de données *application_train* est constitué de 307 511 clients et de 122 features. Parmi ces features, on retrouve pour chaque client des informations sur le sexe, l'âge, le nombre d'enfants, la situation familiale, le revenu, l'emploi etc.

Ce jeu de données contient également la colonne « TARGET » qui prend la valeur 0 lorsque le client rembourse le prêt à temps et la valeur 1 lorsque le client ne rembourse pas le prêt à temps. En étudiant ce feature, on peut voir qu'il y a un important déséquilibre entre le nombre de clients solvables et le nombre de clients non solvables.



Pour plus d'informations sur la partie exploration des données, voir le notebook *P7_01_exploration*.

Avant de commencer la partie modélisation, plusieurs features engineering ont été effectués sur les jeux de données *application_train* et *application_test*, mais aussi sur les 6 autres jeux de données pour ensuite les joindre avec les données *application_train* et *application_test*.

Pour plus d'information sur la partie feature engineering, voir les notebooks *P7_02_feature_engineering* et *P7_04_merge_and_feature_engineering*.

III. Méthodologie d'entraînement du modèle

1. Prétraitement des données

Avant d'appliquer différents modèles d'apprentissage, un pipeline de prétraitement est appliqué.
Ce pipeline permet :

- l'imputation des valeurs manquantes avec SimpleImputer (par la valeur médiane pour les features numériques et par la « Missing » pour les features catégoriels)
- l'encodage OrdinalEncoder pour les features catégoriels binaires et OneHotEncoder pour les autres features catégoriels
- mise à l'échelle des features numériques avec MinMaxScaler

Ce pipeline est ensuite intégré dans un autre pipeline incluant la méthode SelectFromModel qui permet de sélectionner des features selon leur importance (avec l'estimateur XGBClassifier) et le modèle de classification.

2. Séparation en jeux d'entraînement et de test

Pour ne pas que le modèle ait un problème de sur-apprentissage, il faudra effectuer les prédictions sur de nouvelles données.
Tout d'abord, le jeu de données a été coupé en deux parties : un jeu d'entraînement (80 % du jeu de données) et un jeu de test (20 % du jeu de données). Le jeu de test ne sera pas utilisé quand on entraîne notre modèle, il sera utilisé pour calculer la performance du modèle sur des données inconnues.

3. Validation croisée et hyperparamètres

Dans un second temps, le jeu d'entraînement est découpé en plusieurs sous-ensembles. Ces sous-ensembles sont utilisés tour à tour comme jeu de test (le reste est utilisé pour l'entraînement).
Ceci permet d'éviter un biais potentiel lié à une évaluation unique.
Durant ces validations croisées, on a testé une série de paramètres de chaque modèle dans le but de comparer leurs performances pour déduire les meilleurs paramètres.
La validation croisée est effectuée avec GridSearchCV qui reçoit le pipeline évoqué précédemment et les paramètres a optimiser.

4. Sampling

On a vu précédemment qu'il y avait un important déséquilibre entre les clients solvables et les clients non solvables. Ce déséquilibre peut impacter la performance du modèle et fausser les résultats. En effet, le modèle aura tendance à prédire la classe en majorité, dans notre cas les clients solvables.

Pour remédier à ce problème, trois méthodes ont été testées :

- Undersampling : diminue le nombre d'observations de la classe majoritaire afin d'arriver à un ratio satisfaisant.
- Oversampling : augmente le nombre d'observations de la classe minoritaire afin d'arriver à un ratio statisfaisant.
- Génération d'échantillons synthétiques : crée des échantillons synthétiques à partir de la classe minoritaire.

Ces méthodes sont à appliquer avant l'entraînement du modèle.
Pour le modèle final, la méthode oversampling (avec RandomOverSampling) a été retenuée.

5. Modèle de classification

Quatre modèles de classification ont été testés :

- Logistic Regression
- Random Forest Classifier
- LGBM Classifier
- XGB Classifier

Parmi ces modèles, les meilleurs résultats sont obtenus avec Light Gradient Boosting Machine Classifier qui est un algorithme de gradient boosting basé sur l'arbre de décision.

IV. Métrique d'évaluation

1. Matrice de confusion

La matrice de confusion permet de visualiser de manière claire la qualité du modèle de classification.

		Classe prédite	
		0	1
Classe Réelle	0	True Negatives (TN)	False Positives (FP)
	1	False Negatives (FN)	True Positives (TP)

Dans le cas de la banque :

- True negatives revient à accorder un crédit à un client qui peut le rembourser.
- True positives revient à ne pas accorder un crédit à un client qui ne peut pas le rembourser.
- False negatives revient à accorder un crédit à un client qui ne peut pas le rembourser.
- False positives revient à ne pas accorder un crédit à un client qui peut le rembourser.

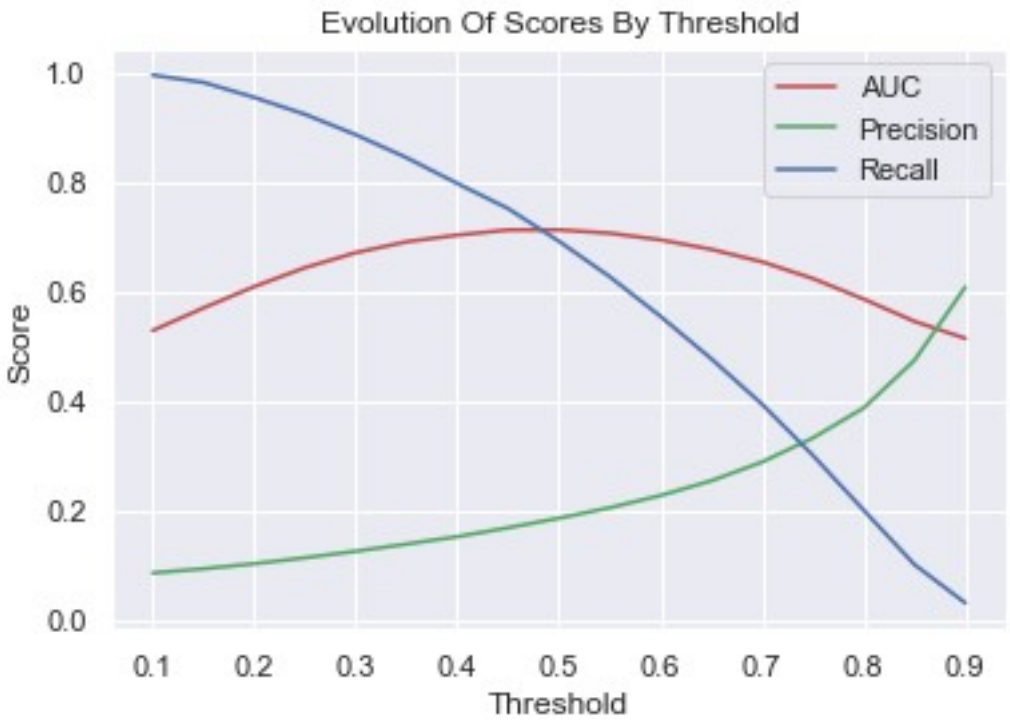
L’idéal sera de maximiser les true negatives et les true positives et de minimiser les erreurs de prédictions, surtout les false negatives.
À partir de la matrice de confusion, d’autres mesures de performances se sont dérivées comme la précision, le rappel et spécificité données respectivement par TP / (TP + FP), TP / (TP + FN) et TN / (FP + TN).

2. Accuracy vs ROC-AUC score

Accuracy mesure le nombre de clients, positifs et négatifs, qui sont classifiés correctement. Dans le cas d’un problème de données déséquilibrées, il est facile d’obtenir un accuracy score élevé juste en classant tous les clients dans la classe majoritaire. Pour cette raison, nous n’utiliserons pas ce métrique d’évaluation.

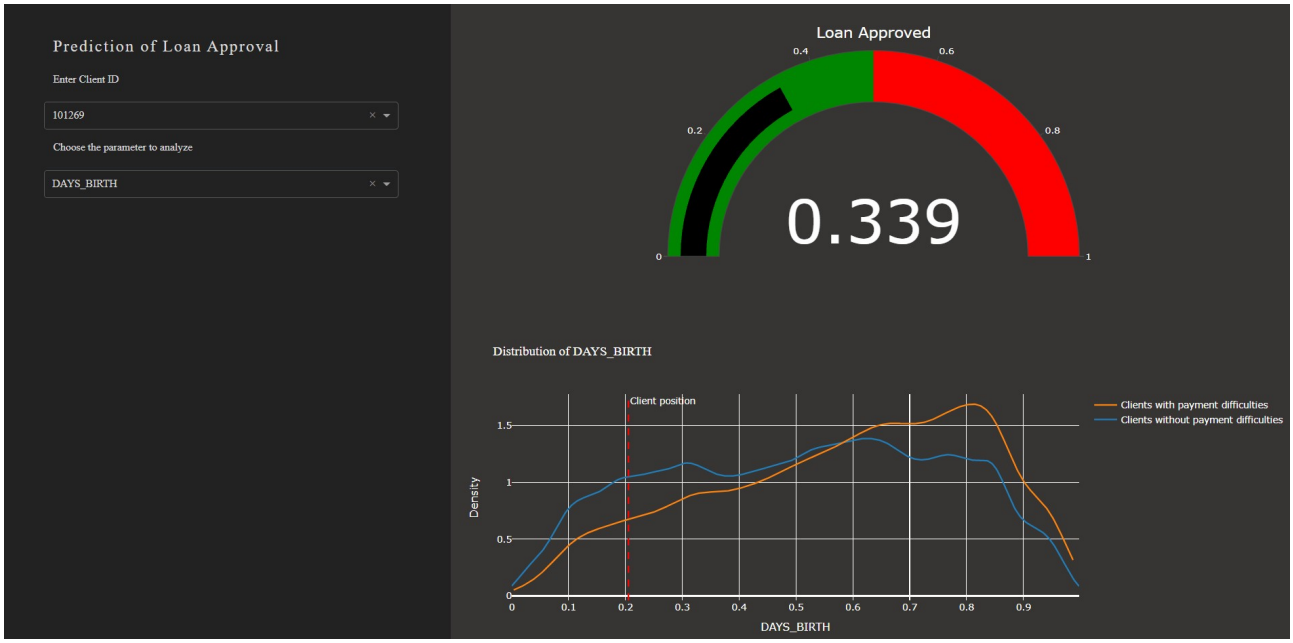
Les algorithmes de classification retournent une probabilité pour chaque client. Plus la probabilité est proche de 1, plus le client est non solvable. Il faudra définir un seuil de probabilité qui permettra de décider si le client est solvable ou non.

La courbe ROC résume les performances d’un classificateur sur tous les seuils de probabilité. Le score ROC-AUC est l’aire sous la courbe de ROC. Le score parfait serait égal à 1, un score égal à 0,5 indiquerait un classificateur aussi efficace qu’un classificateur aléatoire.
Dans ce projet, le score ROC-AUC a été utilisé. Le seuil de probabilité est défini à 0,5.



V. Interprétabilité du modèle

Pour interpréter les résultats, un dashboard a été mis en place. Ce dashboard permet d'afficher le score du client dans un intervalle allant de 0 à 1 et la position du client dans un graphe de densité de distribution des clients solvables et non solvables selon certains features.



VI. Limites et améliorations possibles

La partie feature engineering pourrait être améliorée avec les connaissances et les idées de personnes de ce domaine.

Le modèle d'apprentissage pourrait être amélioré. En effet, la partie GridsearchCV pouvait prendre un temps d'exécution assez important. Ce qui m'a poussé à choisir un nombre de folds faible et une exploration des hyperparamètres assez restreinte.
Le modèle comporte des biais qu'on pourrait améliorer comme les jeunes hommes qui sont discriminés. En effet, le modèle a tendance de ne pas les accorder de crédit.

Le dashboard pourrait être amélioré :

- Le nombre de features sélectionnables est très limité. Étant donné que le nombre de features au total était très élevé, je ne voulais pas tous les mettre dans le dashboard. En effet, cela pourrait être oppressant pour l'utilisateur qui ne saurait pas lesquels choisir. J'ai essayé de choisir les features selon leur importance à la création du modèle, mais je suis sûrement passée à côté d'autres features intéressants.
- On aurait pu mettre des graphiques plus variés, divers pour l'interprétabilité. En effet, la distribution de densité n'est pas toujours évidente à interpréter.