Projet 7 - Note Méthodologique

LOAN SCORING & DASHBOARD

Agustin Bunader
OPENCLASSROOMS | DATA SCIENTIST | AUGUST 2021

Index

Introduction 2
Source de données 2
Contexte 2
Méthodologie d'entraînement du modèle 3
Feature engineering 3
Resampling 4
Preprocessing 4
Construction du modèle 5
Métriques et AUC 5
Importance des features 6
Modèle réduit 7
Interprétabilité du modèle 7
SHAP (Shapley Additive exPlanations) 7
LIME (Local Interpretable Model-agnostic Explanations) 9
Limites et améliorations possibles 10

Introduction

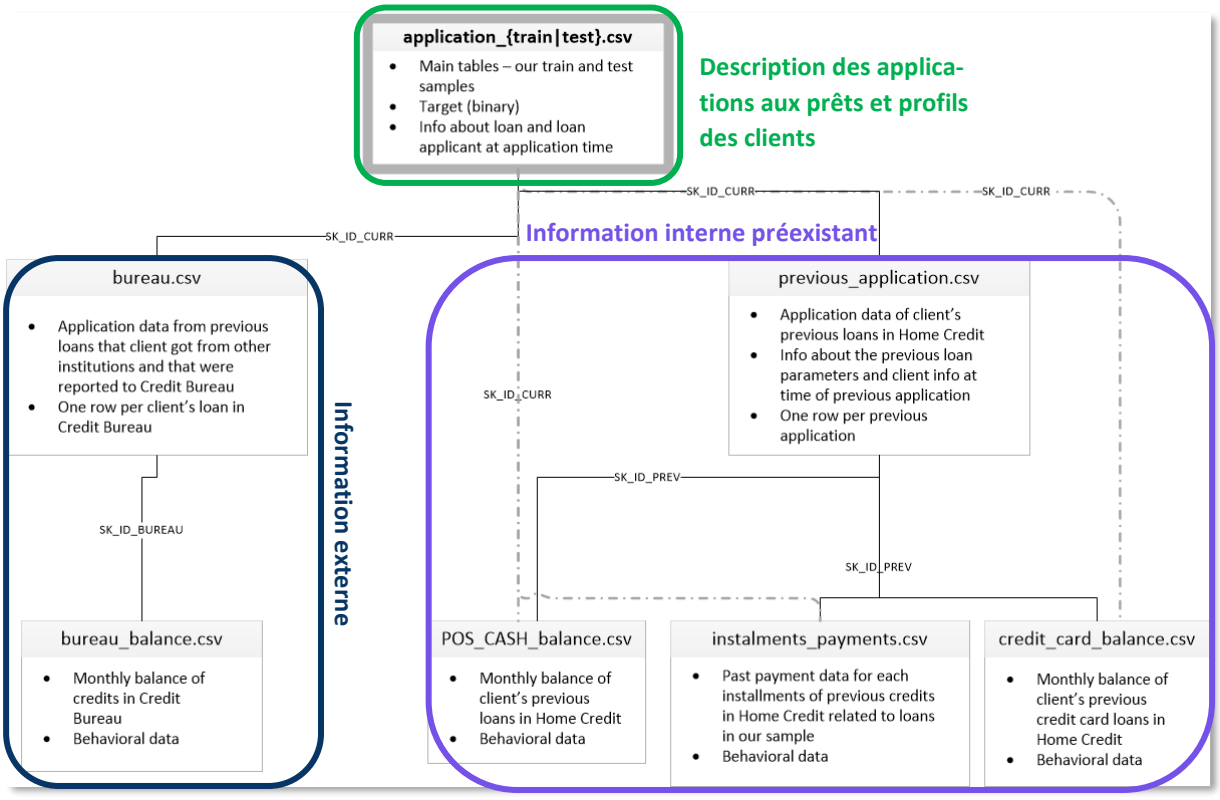
L'entreprise Prêt à dépenser souhaite développer un modèle de Scoring de la probabilité de défaut de paiement du client pour étayer la décision d'octroi ou non d'un prêt à la consommation à un client potentiel en s'appuyant sur des diverses sources des données.

Les potentielles clients soumettent leurs demandes de prêts à l'organisme Prêt à dépenser. Une fois analysé la demande, l'appliquant est informé de l'octroi ou refus du prêt.

L'entreprise collecte les motifs des refus des demandes de prêt et l'ensemble des données de remboursement en cas d'octroi. Ainsi que l'information personnelle basique des clients et demandeurs de prêt. De plus, la société dispose d'informations financières externes des registres.

Source de données

La société nous a proportionnée sept fichiers CSV contentant des données relatives aux clients et aux dossiers des demandeurs des prêts. Le fichier « application_train » contenant la variable cible TARGET, a été utilisé pour l'entraînement du modèle. D'autre, le fichier « application_test » est l'échantillon à prédire leur variable TARGET.



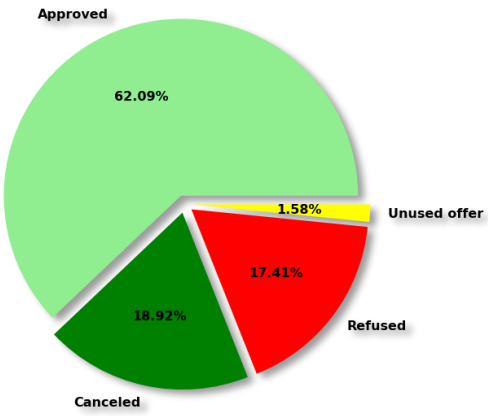
Contexte

Grace aux données cumulées, la société a créée l'indicateur binaire TARGET, qui identifie la difficulté ou non de rembourser un prêt selon le profil du client. La variable cible est connu pour 307511 dossiers où 1 (24825 dossiers) signifie défaut du paiement et 0 (282686 dossiers) tous les autres cas.

L'entreprise dispose des données concernant le profil du client ainsi que les informations financières internes préexistants du 95.1% des dossiers et aussi des détails financiers externes pour 85.8% des dossiers.

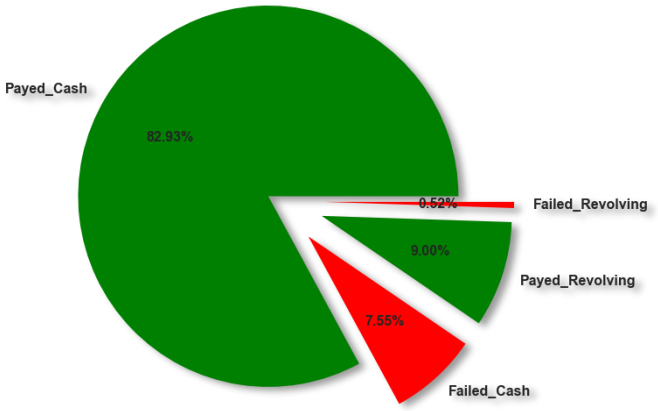
Le but du projet est d'assigner à chaque dossier qui appartient le set de test une prédiction de TARGET et le détail d'interprétabilité de la prédiction.

Il faut signaler que 62.09% des dossiers avec des antécédents internes sont catalogues comme approuves et seulement 17.41% appartient à la catégorie refusée.



D'autre, en regardant la distribution du TARGET on peut apprécier que seulement 8% des prêts sont tombés dans la situation d'impayé.

Deux fois plus des dossiers sont refusés que impayés. Cela parle de la sévérité des refus et les paramètres du risque avec les quelles la société travaille.



Méthodologie d'entraînement du modèle

Le modèle a été entraîné sur la base du jeu obtenu après l'analyse exploratoire et la création des nouveaux features. Le notebook avec les détails est disponible sur le repositoire Github cliquant ici.

Feature engineering

L'entraînement du modèle sera fait sur trois set des données construits avec plus ou moins détails et analyse des variables présents dans les fichiers source.

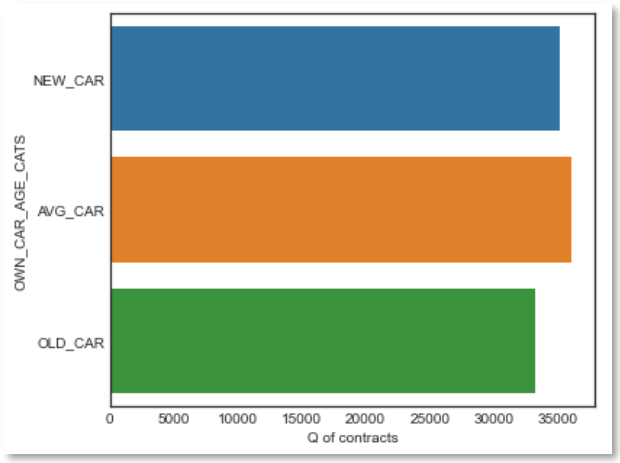
Dans une première étape, après de l'analyse exploratoire de chaque fichier source, on crée des variables dummy pour les features du type catégoriel. Ensuite, on fait une détection des anomalies dans les variables du type date et catégoriels non-interprétables.

	DAYS_BIRTH	DAYS_EMPLOYED
count	307511.000000	307511.000000
mean	43.936973	-174.835742
std	11.956133	387.056895
min	20.517808	-1000.665753
25%	34.008219	0.791781
50%	43.150685	3.323288
75%	53.923288	7.561644
max	69.120548	49.073973

Un exemple d'anomalie trouvé c'est le cas des « Gilgamesh » dans la variable « DAYS_EMPLOYED » du fichier « application_train » où il y a des clients qui ont travaillé pour leur actuel employeur pendant plus de 1000 ans.

Une fois les anomalies corrigées, on commence avec la création des nouvelles variables (ratios, catégories basées sur des données discrètes, durée du crédit et agrégations des données).

La dernière étape du c'est la suppression des features avec plus de 20% des valeurs manquants et on fait un traitement par imputation de la médiane sur les colonnes restantes.

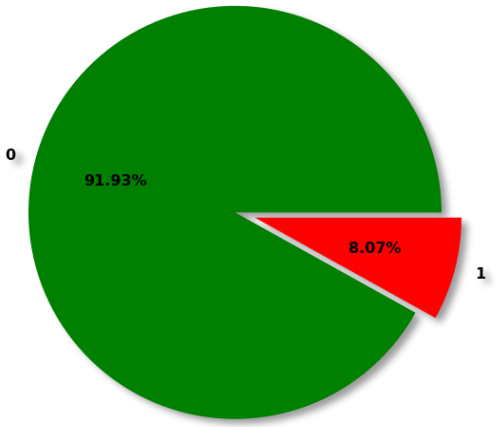


Finalement, ont agrégé les dataframes sur la colonne d'identifiant du dossier/client, ensuite on aligne les jeux des données d'entraînement et du test pour conserver des structures identiques.

Resampling

Le principal problème dans le projet c'est le déséquilibre des catégories de TARGET. La catégorie 0 (contentant les clients sans problèmes de remboursement du prêt) est fortement plus lourde que la catégorie 1 (contenant les clients ayant rencontré des problèmes pour rembourser leur prêt).

Si on décide de continuer sans résoudre cette situation, on obtiendra un modèle prédisant que la classe majoritaire 0. Pourtant, il convient d'entraîner notre modèle sur un jeu de donnés équilibré.



Afin d'éviter de construire un modèle faux, nous allons utiliser la méthode « RandomUnderSampler » de la librairie « Imblearn ». Cette méthode supprime la classe la plus lourde en prélevant au hasard des échantillons avec ou sans remplacement (sans remplacement dans notre cas).

Ils existent plusieurs techniques de resampling des données. Dans le des « under sampling » on peut trouver aussi la méthode « Tomek Links » et dans les « over sampling » on trouve le fameux « SMOTE » qui consiste à synthétiser des éléments de la classe minoritaire à partir de ceux déjà existants en choisissant au hasard un point de la classe minoritaire et en calculant les k plus proches voisins de ce point ajoutant les points synthétiques entre le point choisi et ses voisins.

La méthode RandomUnderSampler été choisi suite a des soucis d'optimisation de temps de calcul car elle s'avère de la méthode la plus rapide.

Preprocessing

Avant de passer à l'entraînement du modèle, on fait les suivantes transformations aux jeux des données :

1. Imputation par la médiane des valeurs manquantes
2. Scaling entre 0 et 1 avec la méthode « MinMaxScaler » car les distributions des features ne sont pas du type Gaussienne
3. Application de la méthode « RandomUderSampler » pour équilibrer les catégories du TARGET
4. Echantillonnage avec la méthode du « train_test_split » avec un ratio entrainement-test du 80-20

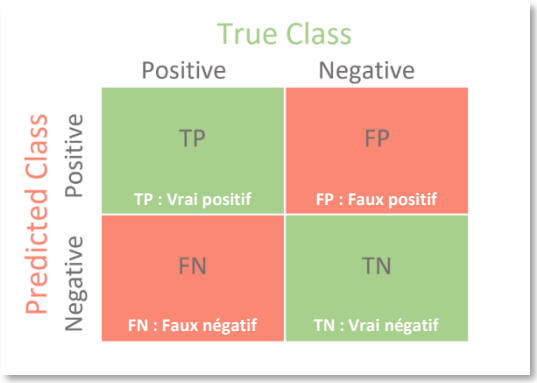
Construction du modèle

Pour chaque jeu de données, différents modèles ont été testés (Logistic Regression, Random Forest Classifier, XGBClassifier et LGBMClassifier) et notre choix final s'est porté sur le LGBMClassifier.

Métriques et AUC

Comme du point de vue de l'entreprise on cherche à éviter de mal catégoriser les clients avec un fort risque de défaut, notre objectif est de minimiser le pourcentage des Faux Négatifs. Autrement dit, on cherche à maximiser le Recall[1] (Erreur de Type II).

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



Par ailleurs, on cherche à maximiser le nombre de clients potentielles, notre objectif est de minimiser le pourcentage des Faux Positifs. Autrement dit, on cherche à maximiser la Precision[2] (Erreur de Type I).

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

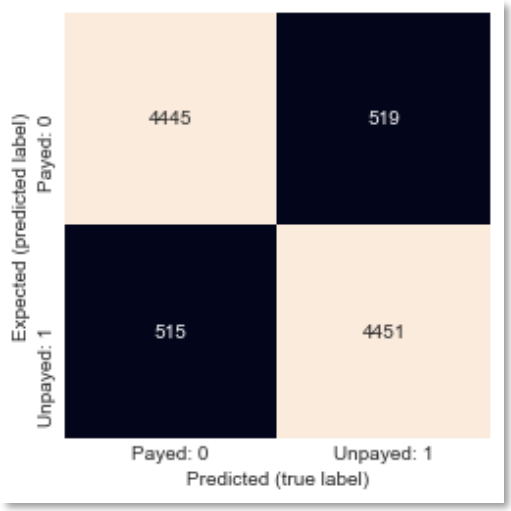
On cherche donc une fonction qui optimise les deux critères. Cette fonction c'est la mesure du F1-Score, calculée comme la moyenne harmonique de Precision et Recall, qui fournis un score unique qui équilibre à la fois les préoccupations de l'Erreur de Type I et Type II en un seul valeur avec un score maximum de 1 (précision et rappel parfaits) et 0. Globalement, c'est une mesure de la précision et de la robustesse du modèle.

$$\text{F1} = \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + TP + FP + FN}$$

Résultats obtenus :

1:	Accuracy	Precision	Recall	F1	Time
Logistic Regression	0.71	0.71	0.71	0.71	00:00:19
RandomForestClassifier	1.0	1.0	1.0	1.0	00:00:40
XGBClassifier	0.85	0.85	0.84	0.85	00:00:27
LGBMClassifier	0.9	0.9	0.9	0.9	00:00:30,

Avec la suivant matrice de confusion pour le classifieur LGBMClassifier :



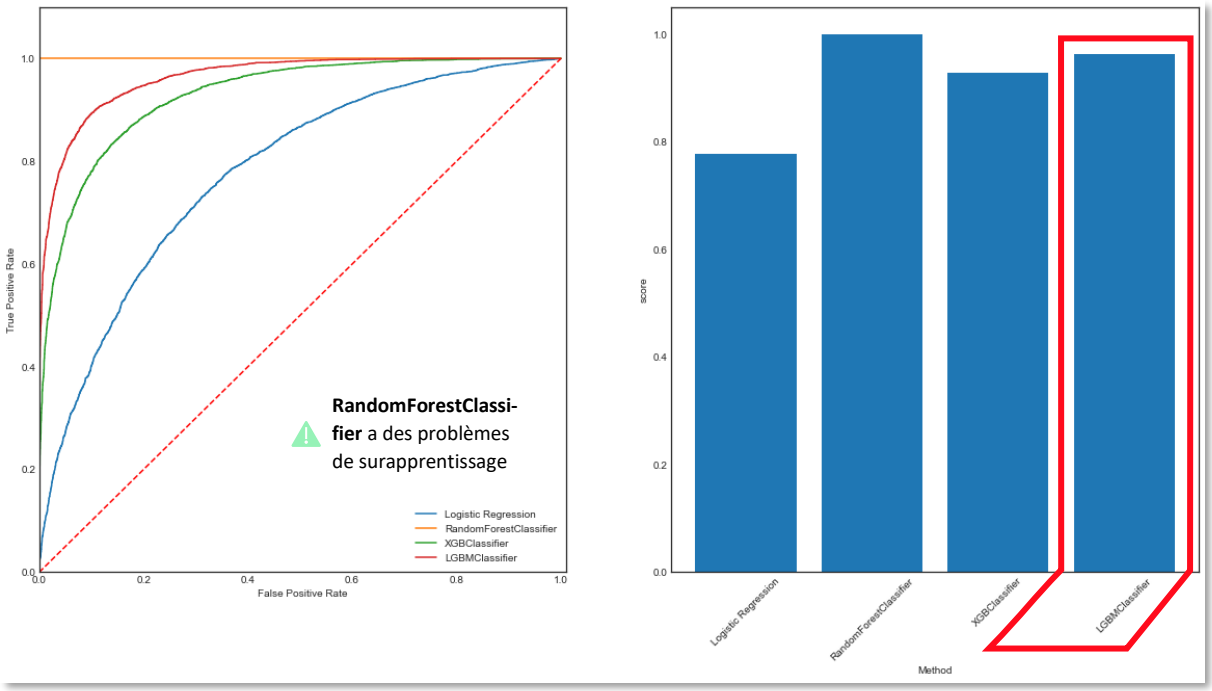
[1] Recall : Quelle proportion de positifs réels a été identifiée correctement ?
[2] Precision : Quelle proportion d'identifications positives était réellement correcte ?

Nous pouvons représenter les évolutions de Recall et Precision en fonction du seuil de classification avec une courbe ROC (Receiver Operating Characteristic). Cette courbe est un outil communément utilisé avec les classifieurs binaires (comme notre cas) parce qu'elle caractérise le classifieur qui a produit les résultats sous forme de probabilités par variation du seuil de classification. Un modèle idéal a un Recall et une Precision égale à 1. Plus, la courbe se rapproche de cet idéal, meilleurs sont les indicateurs.

La mesure de cette performance peut être résumée par le Score AUC (Area Under the Curve). Cette mesure est donc utilisée pour l'évaluation des modèles et représente à ce titre un critère de sélection du classifieur.

Dans notre cas, on doit choisir un classifieur capable de traiter un nombre important de données avec une vitesse de calcul raisonnable et il doit aussi proportionner des fonctionnalités permettant son interprétation.

Le classifier répondant le mieux à l'ensemble des critères est le LGBMClassifier avec un Score AUC de 0.96.

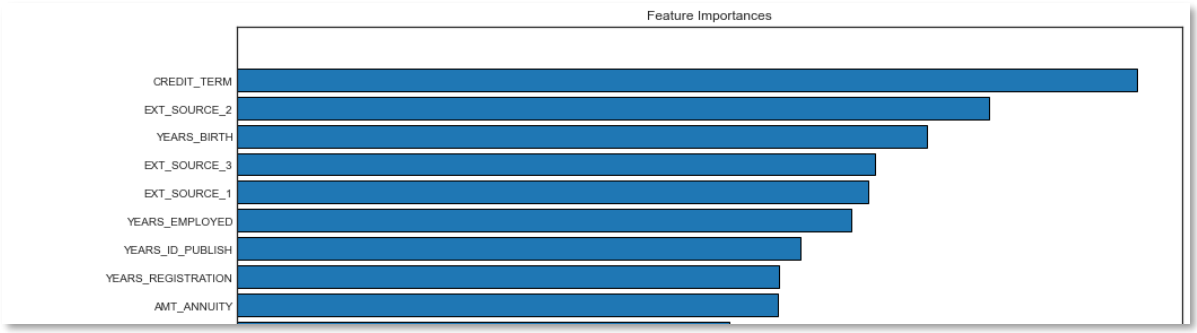


Importance des features

L'analyse de l'importance des variables est la première étape dans l'interprétabilité d'un modèle car elle nous permet de visualiser sur quelles variables s'appuie le modèle pour effectuer ses prédictions.

Dans le cas des modèles d'arbres de décision (comme tous les modèles testés), l'importance d'une variable repose sur la Gini Importance ou MDI (Mean Decrease in Impurity) additionné sur la longueur des nœuds des arbres de décision.

Ici, la durée du crédit, représentée par la variable « CREDIT_TERM » est le principal levier du modèle choisi lors de la prédiction du TARGET et par conséquence dans décision d'octroi du prêt. De toute façon, l'importance globale d'une variable peut ne pas être la même dans le contexte local (Et vice versa). Cette phénomène s'appelé « Local Fidelity » et sera traité lors de l'analyse de l'interprétabilité du modèle

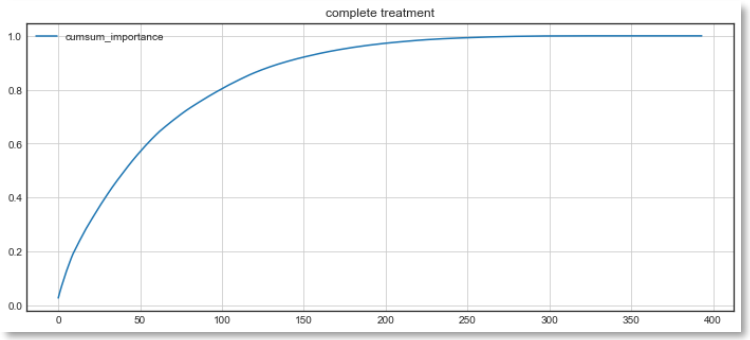


1: Extrait des variables importantes

Modèle réduit

Même si notre modèle a réussi à réduire les Faux Négatifs et Faux Positifs, il garde presque 400 variables, situation pas optimale en termes du temps de calcul et ressources utilisés au moment de la mise en place du dashboard. Par conséquent, nous devons réduire le nombre de variables dans le modèle tout en essayant de maintenir notre score.

Si on regarde la courbe des variances cumulés du modèle choisi, c'est possible d'apprécier que la plupart des features n'apportent pas d'information important à la prédiction.



On a décidé de garder 40 variables de notre modèle, avec un F1-Score du 0.86 et un Score AUC de 0.94.

Interprétabilité du modèle

Comme indiqué précédemment, l'analyse de l'importance des features n'est que le début de l'interprétabilité du modèle. Le jeu de données à presque 400 variables a été utilisé pour l'optimisation du classifieur LightGBM avec pour objectif la maximisation du gain. Nous avons utilisé cette fonctionnalité pour la réduction de dimensions de notre modèle lors de son optimisation de vitesse de calcul.

Entant que pour l'interprétabilité locale, ils existent plusieurs façons de dévoiler les secrets des modèles de machine learning ; les plus populaires sont : « SHAP » et « LIME »

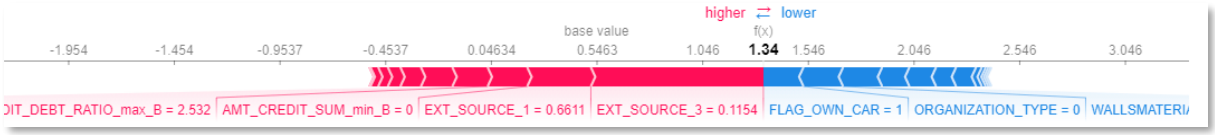
SHAP (Shapley Additive exPlanations)

Il faut signaler que même si SHAP est disponible sur le Notebook d'interprétation (disponible cliquant ici) son calcul est particulièrement lourd donnant comme résultat un fichier au-dessus de 100MB qui manque de réactivité lors de l'utilisation du dashboard.

La méthode « SHAP » consiste à calculer la valeur de Shapley de chaque variable pour chaque dossier dans notre jeu de données. C'est-à-dire, la moyenne de l'impact d'une variable (sur la prédiction) pour toutes les permutations des variables possibles. Donc, la somme des effets de chaque variable explique alors la prédiction.

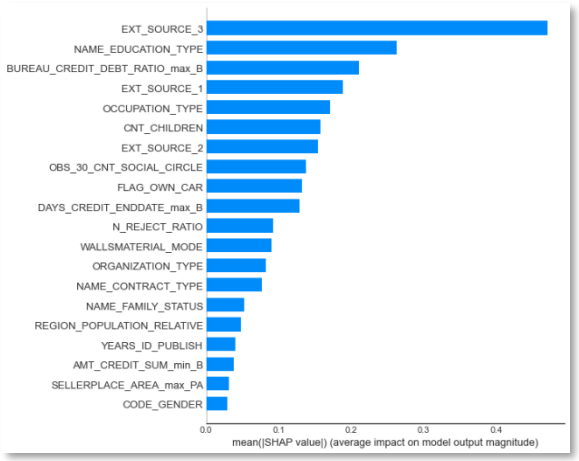
Le graphe ci-dessous représente les impacts des valeurs des features sur la prédiction. Les variables

en corail contribuent à une prédiction positive (soi-disant un client à haut risque) et les variables bleu-ciel appuient une prédiction négative (autrement dit, un client à bas risque).



2: Dans ce dossier d'exemple on peut apprécier que la valeur SHAP est positive. Donc, il s'agit d'un client à haut risque.

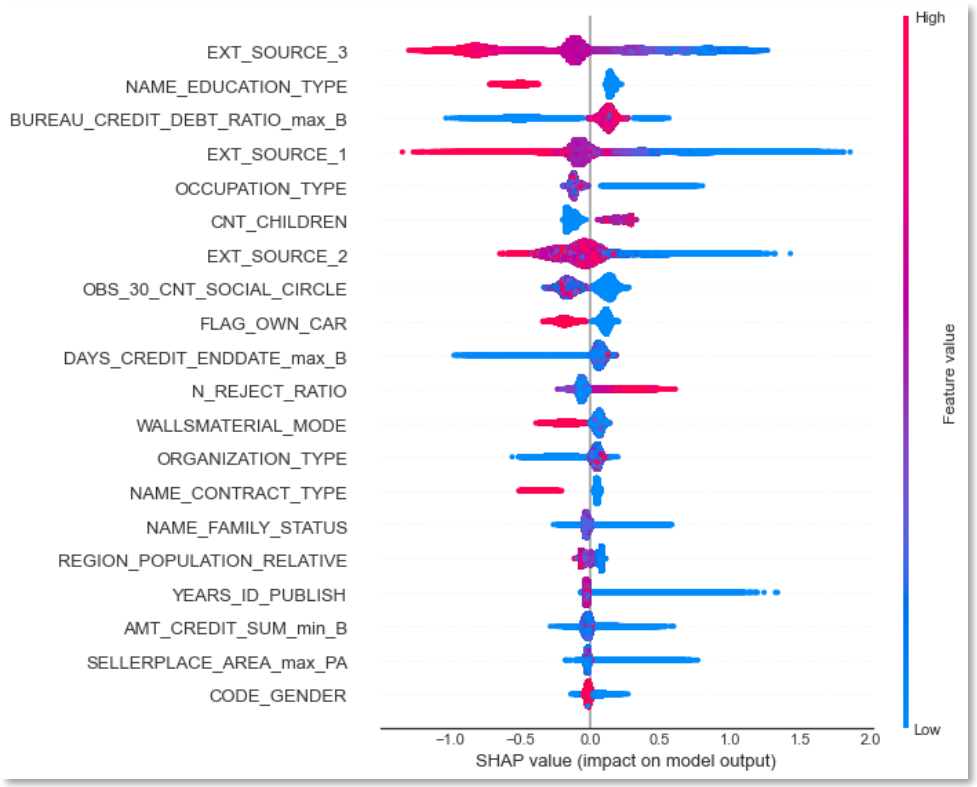
« SHAP » également proportionne une analyse globale des variables (Figure 3). Autre visualisation plus détaillée de l'analyse globale est la Figure 4 où les dossiers sont représentés par des points dont la couleur varie du corail (valeur élevée) à bleu-ciel (valeur petite) en fonction des valeurs du Shapley ce qui caractérise l'impact du feature sur la prédiction.



3: Importance globale des variables selon SHAP

Un exemple de comme lire la Figure 4 c'est la variable « NAME_EDUCATION_TYPE » où on peut apprécier que le plus haute la catégorie d'éducation (corail), plus faible la valeur du Shapley à niveau local (ils sont placés sur la gauche) et contrairement le plus bas est la catégorie d'éducation (bleu-ciel) le plus le feature contribue à que la prédiction soit de haut risque.

Sur le dashboard est uniquement disponible l'analyse globale des features car l'analyse local demande substantiellement plus du temps de calcul que la méthode « LIME » détaillée ci-dessous.

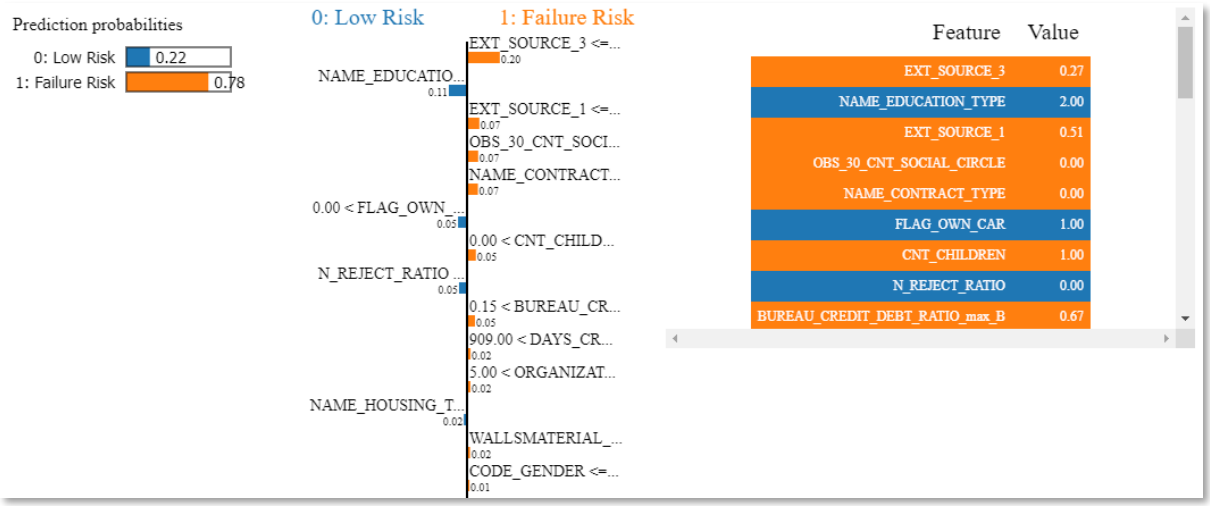


4: Impacte global des features sur la prédiction

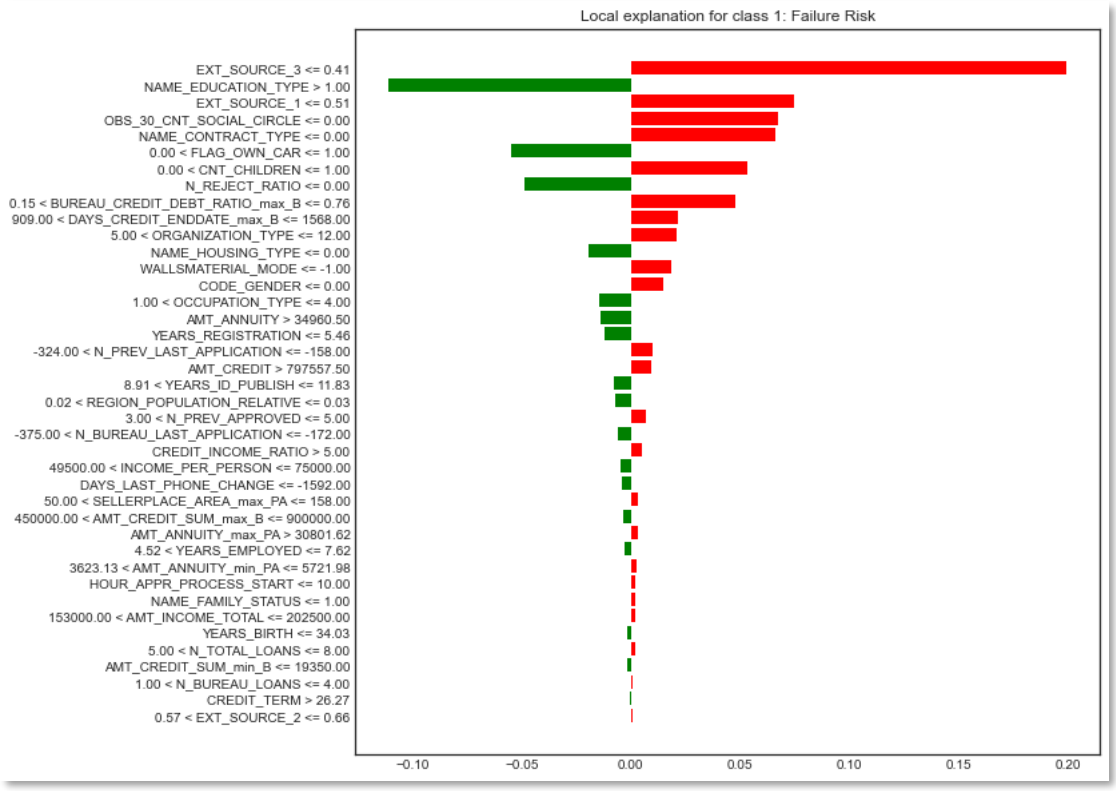
LIME (Local Interpretable Model-agnostic Explanations)

Dans le cas de « LIME », la méthode « LimeTabularExplainer » fourni une mesure de la contribution positive ou négative de chaque feature à la prédiction. Cette mesure est calculée via la distance et le poids des voisins de l'élément choisi à expliquer.

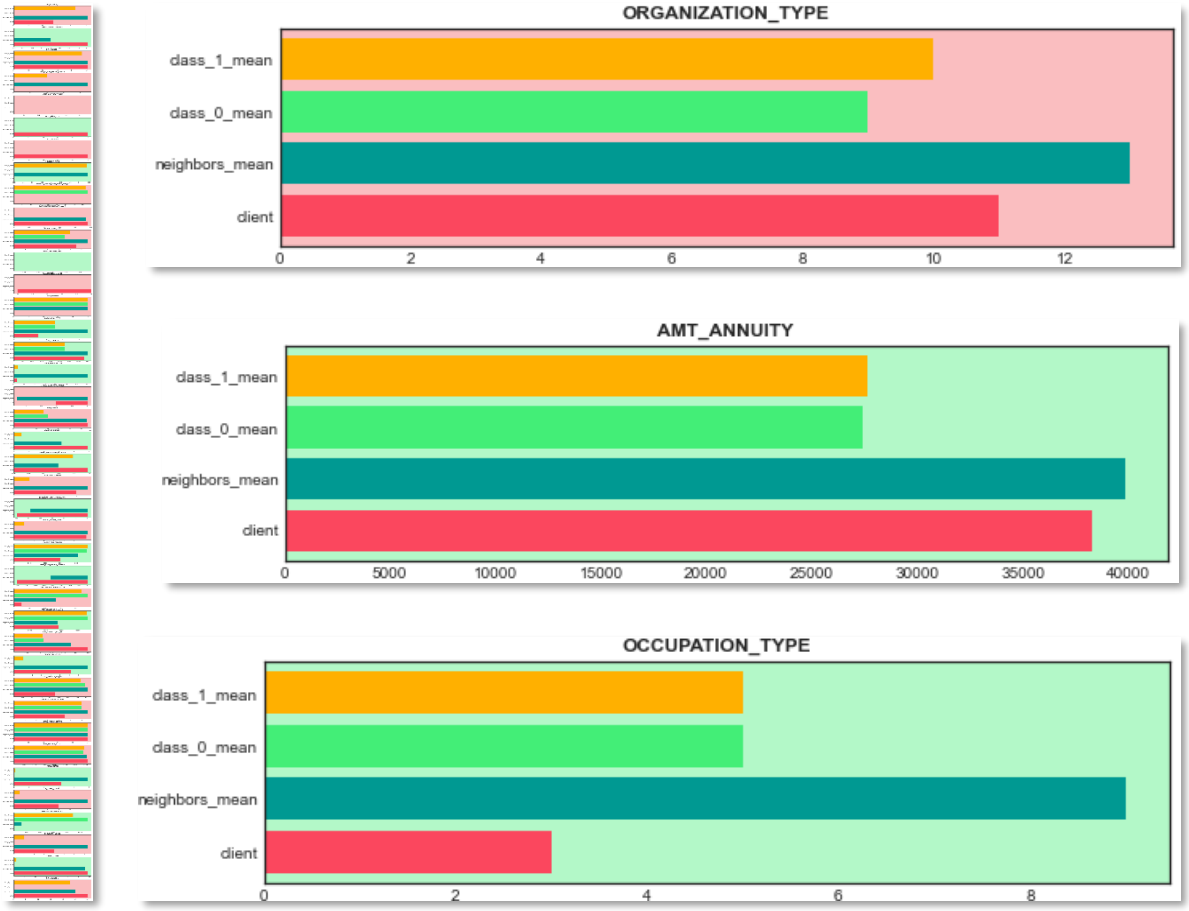
La figure ci-dessous résume la prédiction pour un dossier exemple choisi au hasard. De même façon que « SHAP », on peut voir le détail de la contribution de chaque feature à la prédiction. Mais cette vue n'est pas disponible dans le dashboard parce que on a privilégié des diagrammes à barres plus simples à lire.



Le principal diagramme à barres disponible dans le dashboard est telle comme le montre la figure ci-dessous, plus similaire au sommaire de « SHAP » mais à niveau local. Car il s'agit de la prédiction de la catégorie 1 ce que nous intéresse le plus, nous avons décidé d'inverser les couleurs utilisés par « LIME » : la couleur rouge supporte la prédiction de la catégorie 1 (haut risque) et la couleur verte supporte la prédiction de la catégorie 0 (faible risque).



Un autre diagramme à barres est disponible sur le dashboard, son intérêt est de faciliter la comparaison entre le dossier choisi, la moyenne des dossiers similaires (choisis avec un algorithme de voisinage) ainsi que les moyennes des catégories 0 et 1. Cette diagramme nous permet de détecter des valeurs en écart, soi-disant des anomalies, pour expliquer la prédiction d'une catégorie.



Finalement, comme déjà détaillé, le calcul par client est seulement disponible sur la méthode « LIME » car ses temps de calcul sont plus faibles que les ressources demandées par la méthode « SHAP »

Limites et améliorations possibles

Tout au long de ce Note Méthodologique nous avons pu constate que la réalisation du modèle a demandé la conception de nombreuses transformations et multiples traitements des données. En fait, les résultats sont dépendants des paramétrages choisis au moment du création du modèle. Ce choix été fait toujours avec l'objectif principale de diminuer le plus possible le temps de calcul et les ressources outillés car le modèle est hébergé dans le servie gratuite de « Github » et la solution dashboard fonctionne sur « Streamlit » aussi gratuite.

Cette situation impose ainsi des limitations sur les performances de prédiction. Même si le score du modèle choisi semble assez haut, en réalité on n'a pas mis en place aucun type de méthode de « K-Fold cross-validation ». C'est globalement accepté que « K-Fold » génère des résultats moins baisés et moins optimistes que notre simple train-test split.

La modélisation effectuée a été effectuée sur la base d'une forte hypothèse : la définition du « F1-Score ». Le « F-Beta Score » offre la possibilité de paramétrer le poids relatif de bascule entre le « Recall » et la « Precision ». L'avantage face à le « F-1 Score » est au moment de diminuer les Faux Négatifs

car on peut assigner un « beta value » mayeur à 1 ; de cette façon le « Recall » serait plus important que la « Precision » et varier le seuil de probabilité.

De plus, dans l'équilibrage des données (resampling) on a réduit l'échantillon disponible pour l'entrainement du modèle car nous avons utilisé une méthode d'under sampling. Une évidente amélioration possible pour notre modèle serait la mise en place du méthode « K-Fold ». Autre possible amélioration à envisager, si les temps de calcul sont moins importants, serait la mise en place des autres méthodes de resampling comme « Tomek Links » (under sampling) ou « SMOTE » (over sampling).

Concernant l'interprétation du modèle, la méthode « SHAP » offre des résultats plus intéressants (à niveau local) et interactifs que « LIME ». Cependant, son implémentation dans une application web (comme « Streamlit ») reste encore difficile. Autre amélioration possible serait la restructuration du jeu des données pour trouver un kernel « SHAP » moins lourde. En ce qui concerne à « LIME », ce serait possible d'optimiser la fonction de similarité basée sur la méthode des plus proches voisins.

En tant que pour le dashboard, une claire amélioration serait de le développer depuis zéro avec Flask à la place du « Streamlit » pour optimiser le plus possible les ressources utilisées. L'addition des graphiques interactifs comme les proportionnes par « SHAP » serait un plus au moment de l'interprétation de la prédiction du dossier. Finalement, le développement d'une application web permettant l'évolution du scoring en direct en même temps que les variables du dossier choisi sont modifiées, autrement dit, l'interactivité des dossiers.