Note méthodologique

Projet : Implémenter un modèle de scoring.

Réalisé par : Sabrine OUANNES

Cadre : Ce document constitue l'un des livrables du projet 7 « Implémenter un modèle de scoring » du parcours de la formation Data Scientist chez OpenClassrooms. Cette note présente le processus de modélisation du modèle.

Table des matières

1. Compréhension du marché 2
2. Compréhension des données 2
3. Préparation des données 3
4. Modélisation - Evaluation 4
   a. Métrique d'évaluation : AUC-ROC 4
   b. Métrique d'évaluation : Fonction coût métier 5
   c. Déséquilibre des classes 6
   d. Optimisation des hyperparamètres 6
   e. Enregistrer le modèle de ML 6
5. Interprétabilité du modèle 7
   a. Interprétabilité globale 7
   b. Interprétabilité locale 7
   - LIME 8
6. Limite et amélioration du modèle 9

1. Compréhension du marché

Ce projet est divisé en deux grandes parties :

- ***Première partie :*** Construire un modèle de scoring, pour l'entreprise financière « Prêt à porter », qui donnera une prédiction sur la probabilité qu'un client rembourse son crédit de façon automatique en s'appuyant sur des sources de données variées (comportementales, financières, etc.).
- ***Deuxième partie :*** 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.

Dans cette note méthodologique, nous allons nous intéresser essentiellement à la première partie et détailler toutes les étapes aboutissant à la construction du modèle de Machine Learning supervisé de classification des clients. Comme tout projet de Data Science, nous allons suivre les étapes illustrées dans la figure 1.

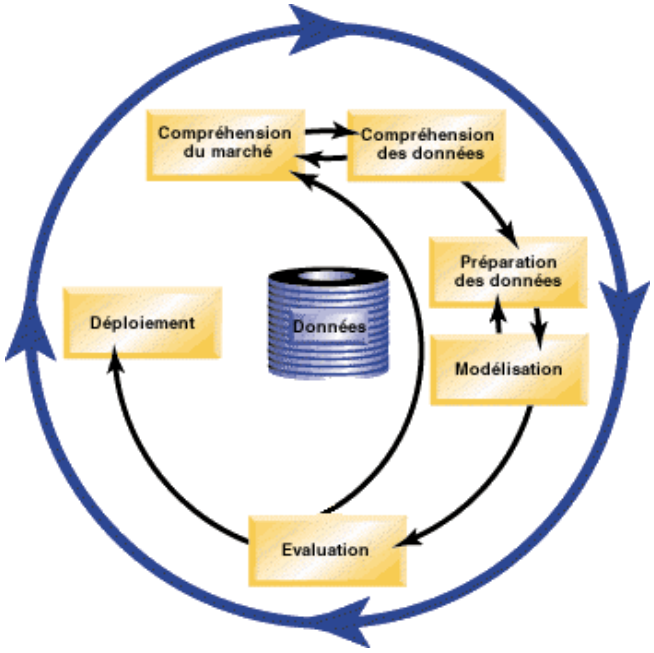


Figure 1 : étapes projet Data Scientist

2. Compréhension des données

Il s'agit de différentes tables (illustrées dans la figure 2) qui résument des données variées (comportementales, financières, etc.) pour un client donné. Chaque client est représenté par son identifiant « SK_ID_CURR ». Les tables seront par la suite jointes pour avoir un seul tableau de taille 307507 lignes × 788 colonnes qui sera ensuite nettoyé et exploité pour la modélisation.
Notre variable cible est « TARGET ». C'est une variable binaire où :

- Classe 0 : le client n'a pas de difficultés de paiement.
- Classe 1 : le client a des difficultés à payer son prêt.

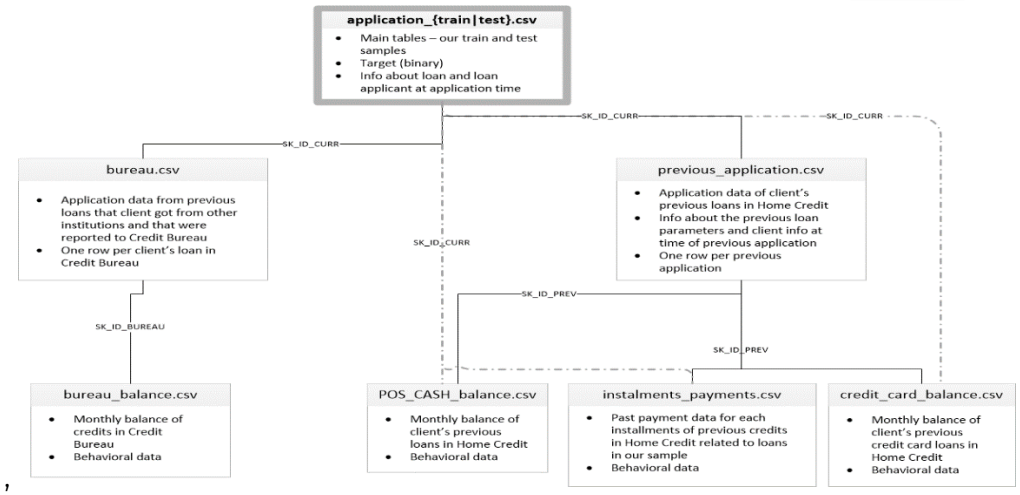


Figure 2 : Différentes données du client

3. Préparation des données

Les données sont au centre des algorithmes de Machine Learning. Par conséquent, préparer au mieux ces données, permettra d'avoir de meilleures performances. L'étape de pré-traitement s'appuie sur un Kernel Kaggle (*https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script*) développé par l'équipe métier pour faciliter la préparation des données nécessaires à l'élaboration du modèle de scoring auquel nous avons ajouté d'autres étapes de pre-processing. Parmi les étapes de préparation des données, on cite :

- ***Gestion des outliers (valeurs aberrantes) :*** Pour certaines variables catégorielles, des valeurs aberrantes apparaissent comme « XNA » pour la variable « CODE GENDER ». Ces valeurs ont été supprimées et nous avons gardé uniquement F pour féminin et M pour masculin.
  Un exemple pour les variables numériques, les valeurs aberrantes (DAYS_EMPLOYED>1000 ans) ont été remplacées pap NaN.
- ***Feature engineering :*** plusieurs variables importantes ont été rajoutées comme la proportion du temps travaillé par rapport à l'âge du client, ou le ratio crédit/revenu etc…
  Ces variables sont données par 'l'équipe métier dans le kernel Kaggle utilisé.
- ***Encodage des variables catégorielles :*** Nous avons utilisé principalement deux méthodes :
  - Les variables ayant deux valeurs différentes comme « Type of contract » de la figure 3, ont été binarisés, i.e. codés par des 0 ou des 1.
  - L'encodage One-Hot a été utilisé pour les variables possédant plus de deux catégories comme « Family status » cet encodage consiste à encoder une variable à *n* états sur *n* bits dont un seul prend la valeur 1, le numéro du bit valant 1 étant le numéro de l'état pris par la variable.

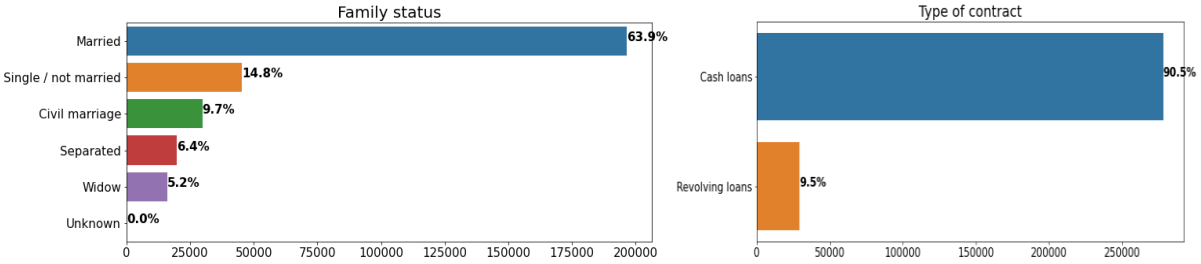
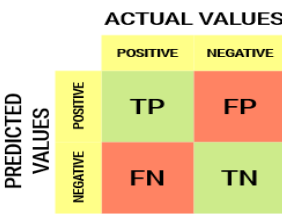


Figure 3 : Exemple de variable catégorielle

- ***Valeurs manquantes:*** Les colonnes ayant plus de 60% de valeurs manquantes ont été supprimées tout en s'assurant que ces variables sont redondantes. Le nombre de variables a été réduit de 788 à 542.
- ***Feature selection:*** Après l'étape de feature engineering, nous avons obtenu 788 variables. Afin de réduire ce nombre. Toutes les factures ayant une corrélation inférieure avec 0.01 avec la variable cible appelée « TARGET » ont été supprimées. Aussi, les variables fortement corrélées entre elles (>0.8) ont été supprimées tout en gardant une parmi elles. Le nombre de variables a été réduit de 542 à 185.
- ***Feature scaling :*** Les ordres de grandeurs des variables numériques dans notre *Data Set* sont très différents. Cette différence d'échelle peut conduire à des performances moindres. Pour pallier cela, nous avons eu recours à la Standardisation qui est le processus de transformer une feature en une autre qui répondra à la loi normale.
- ***Création du Train set et test set :*** Avant de séparer notre jeu de données, nous avons supprimé toutes les lignes ayant des valeurs manquantes. Dans notre cas, nous avons réservé 80% des données du dataset pour l'entraînement et la validation du modèle de Machine Learning. Les 20% restantes seront exploitées pour le test. Cette séparation a été effectué par la méthode *train_test_split* de *scikit-learn* avec l'argument *stratify* afin de garder les mêmes proportions des différentes classes.

4. Modélisation - Evaluation

Un modèle est dit meilleur qu'un autre lorsqu'il donne de meilleurs résultats suivant une métrique de performance bien définie. Notre projet est un problème de classification binaire, l'étiquette est soit égale à 0 (négatif, solvable) soit 1 (positif, non solvable). Il existe donc 4 combinaisons possibles :



- TN : True Negative, le modèle prédit 0 et la valeur réelle est 0
- TP : True Positive, le modèle prédit 1 et la valeur réelle est bien de 1
- FN : False Negative, le modèle prédit 0 alors que la valeur réelle est 0
- FP : False Positive, le modèle prédit 1 alors que la valeur réelle est de 0

Nous avons choisi principalement deux métriques : AUC-ROC et F5 score ainsi que la matrice de confusion.

a. Métrique d'évaluation : AUC-ROC

Afin de classifier les clients de la société financière « Prêt à dépenser » et prédire le score de remboursement du crédit, nous avons testé plusieurs modèles avec leurs

paramètres par défaut, pour avoir un premier aperçu de leur comportement sur nos données. Ces modèles sont illustrés dans le tableau 1.

Tableau 1 : Résultats entrainement différents modèles

Model	AUC	Accuracy	Precision	Recall	F1	F5	Time
GradientBoostingClassifier	0.772361	0.919168	0.56	0.068293	0.121739	0.07068	12.059349
LogisticRegression	0.769105	0.920368	0.6875	0.053659	0.099548	0.055631	0.195451
AdaBoostClassifier	0.753062	0.918367	0.512821	0.097561	0.163934	0.100697	3.313405
LGBMClassifier	0.750303	0.917967	0.5	0.04878	0.088889	0.050534	0.567168
XGBClassifier	0.748804	0.920768	0.62069	0.087805	0.153846	0.090803	4.214271
RandomForestClassifier	0.72055	0.917967	0.0	0.0	0.0	0.0	4.096973
SVC	0.673798	0.917967	0.0	0.0	0.0	0.0	55.861653
KNeighborsClassifier	0.557418	0.913165	0.227273	0.02439	0.044053	0.025257	1.292784
DummyClassifier	0.5	0.917967	0.0	0.0	0.0	0.0	0.022924

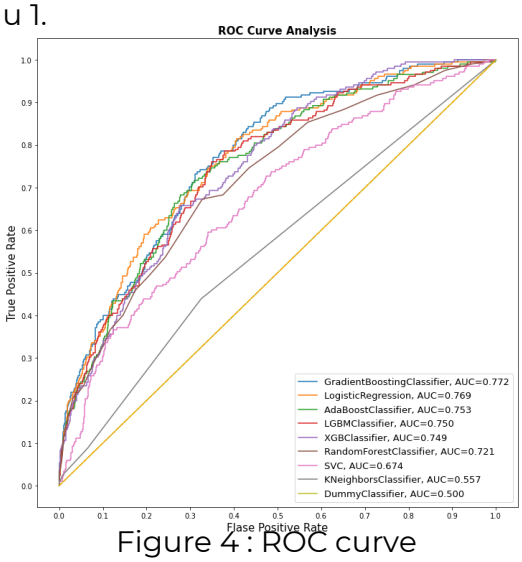


Figure 4 : ROC curve

En cherchant un compromis entre le ROC-AUC score et le temps d'entrainement nous avons opté pour ***LGBM et Logistic regression***. Ces modèles seront optimisés en déterminant la meilleure combinaison d'hyperparamètres qui donne les meilleures performances suivant la fonction coût déterminé par le problème.

- ***AUC-ROC*** :

C'est l'acronyme de **A**rea **U**nder the **C**urve - **R**eceiver **O**perating **C**haracteristic Il s'agit d'une métrique d'évaluation pour les problèmes de classification binaire et mesure la capacité d'un classifieur à distinguer entre les classes positives et négatives. La courbe ROC représente la sensibilité (TPR=$\frac{TP}{TP+FN}$) en fonction de 1 – spécificité (FPR=$\frac{FP}{FP+TN}$) pour toutes les valeurs seuils possibles.

L'aire sous la courbe ROC peut être interprétée comme la probabilité que, parmi deux sujets choisis au hasard, un prêt solvable et un non-solvable, la valeur du marqueur soit plus élevée pour le prêt solvable que pour le non solvable.

Le modèle peut donc se tromper de deux manières différentes, soit en prédisant positif alors que l'individu est négatif (False Positive) soit en prédisant négatif alors que l'individu est positif. En revanche, la perte d'argent est plus conséquente pour un FN (prêt accordé alors que le client n'est pas solvable) qu'un manque à gagner pour un FP (prêt non accordé alors que le client est solvable). Les clients non solvables que notre algorithme ne détectera pas coûteront plus cher à la société financière que le coût d'un client solvable prédit comme non solvable. Dans notre projet, nous nous focalisons sur la minimisation des FN. Pour cela la métrique F5 a été choisie comme fonction coût métier.

b. Métrique d'évaluation : Fonction coût métier

Pour ce projet, une fonction coût choisie dans l'objectif de maximiser les gains obtenus par validation ou non des demandes de prêts bancaires. Le F5 score a été choisi comme fonction coût dans le but de le maximiser.

Ce score a pour formule :

$$\mathrm{F}_\beta\text{-score} = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$$

ou

$$\mathrm{F}_\beta\text{-score} = \frac{TP}{TP+\frac{1}{1+\beta^2}(\beta^2 FN+FP)}$$

Avec β=5

En effet, pour β>1, nous accordons plus d'importance au recall ($\frac{TP}{TP+FN}$), autrement dit aux faux négatifs.

5. Déséquilibre des classes

Notre variable cible comprend 92 % de classes négatives alors que seulement 8% de clients n'ont pas remboursé leur prêt. Ce déséquilibre de classe augmente nettement la difficulté de l'apprentissage par l'algorithme de classification. En effet, l'algorithme n'a que peu d'exemples de la classe minoritaire sur lesquels apprendre.

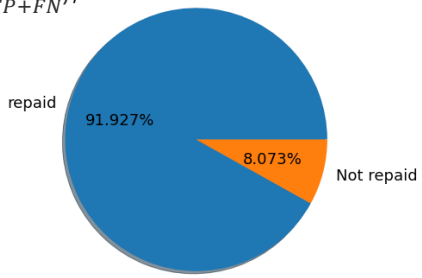


Figure 5 : Répartition des classes

Ainsi, les algorithmes classiques sont souvent biaisés. Le modèle semblera performant mais cela ne sera que le reflet de la surreprésentation de la classe majoritaire. On parle de « précision paradoxe ». Afin de résoudre cette distribution asymétrique de classes, nous avons utilisé l'argument *class_weight = balanced* des modèles LGBM et LogisticRegression. En effet, ceci peut être réalisé en attribuant des poids différents aux classes majoritaires et minoritaires. La différence de poids influencera le classement des classes lors de la phase d'entraînement. Le but est de pénaliser l'erreur de classement faite par la classe minoritaire en fixant un poids de classe plus élevé et en même temps en réduisant le poids de la classe majoritaire.
Dans notre cas, les poids affectés sont : 0.54 pour la classe majoritaire (classe 0) et 6.23 pour la classe minoritaire.

c. Optimisation des hyperparamètres

Après avoir sélectionné LGBM et Logistic regression, il est temps de chercher la meilleure combinaison des hyperparamètres qui donne le meilleur F5 score. Pour ceci, nous avons ajusté quelques paramètres pour les modèles choisis à savoir «*max_depth*» qui fixe une limite sur la profondeur de l'arbre et «*num_leaves*» qui est l'hypermaramètre le plus important pour controler la complexité du modèle. Il spécifie le nombre de feuilles dans un arbre.
L'optimisation des hyperparamètres a été effectué en utilisant GridSearchCV de Scikit-learn qui inclut un système de validation croisée. C'est une méthode d'optimisation qui va nous permettre de tester une série de paramètres et de comparer les performances pour en déduire la meilleure combinaison.

d. Enregistrer le modèle de ML

Après avoir optimisé notre modèle, nous l'avons enregistré dans un fichier *sav* qui sera par la suite téléchargé afin de le réutiliser dans les APIs. Cette sauvegarde s'appelle la *sérialisation* , tandis que la restauration s'appelle la *désérialisation* .

Ceci a été effectué à l'aide des commandes de la librairie pickle : *pickle.dump* pour la sérialisation et *pickle.load* pour la *désérialisation*.

6. Interprétabilité du modèle

Afin de comprendre le mécanisme de prédiction, nous avons opté pour :

- La méthode globale : lorsque la méthode d'interprétation concerne le fonctionnement global de l'algorithme tous inputs confondus.
- La méthode locale : Lorsque la méthode d'interprétation explique une prédiction selon un input donné dans notre cas les données d'un client.

a. Interprétabilité globale

Le Feature Importance fait référence à des techniques qui calculent un score pour toutes les variables d'entrée pour un modèle donné. Les scores représentent simplement l'importance de chaque Feature. Un score plus élevé signifie que la caractéristique aura un effet plus important sur le modèle utilisé pour prédire la variable cible. Nous avons présenté dans la figure 6 les vingt plus importantes variables pour le modèle LGBM.

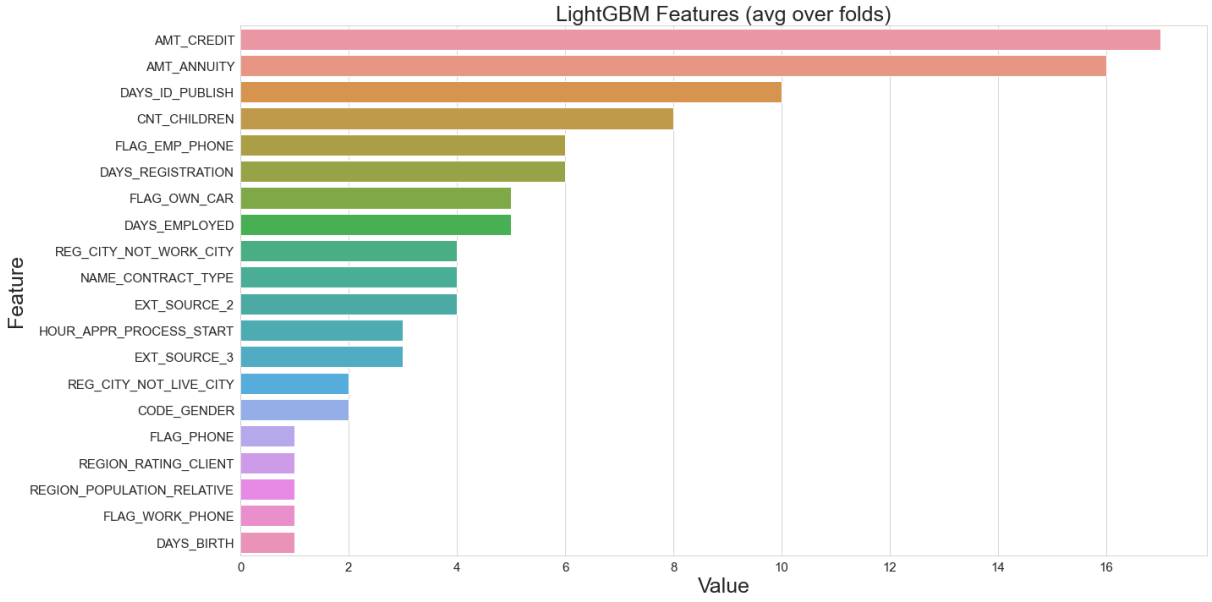


Figure 6 : Feature importance pour le classifieur LGBM

Afin de réduire le nombre de variables, nous avons utilisé l'algorithme *SelectFromModel* pour choisir les vingt les plus importantes variables pour le modèle Puis réentraîné le modèle dessus.

b. Interprétabilité locale

L'interprétabilité se définit comme la capacité pour un humain à comprendre les raisons d'une décision d'un modèle. Nous avons utilisé pour l'interprétabilité locale les algorithmes LIME (**L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations) et SHAP (**SH**apley **A**dditive ex**P**lanations).

- **LIME**

  LIME permet de déterminer pour chaque observation les « features » qui ont plus d'impact dans la prédiction du score. Cette méthode fonctionne en 2 étapes :

  - 1ère étape : l'algorithme LIME génère de nouvelles données fictives, dans un voisinage proche de l'individu à expliquer.
  - 2ème étape : LIME entraîne un modèle transparent sur les prédictions du modèle « boîte noire » complexe qu'on cherche à interpréter. Il apprend ainsi à l'aide d'un modèle simple et donc interprétable (par exemple, une régression linéaire ou un arbre de décision).

  Nous avons essayé d'expliquer les scores de prédiction pour un client donné dans la figure 7.

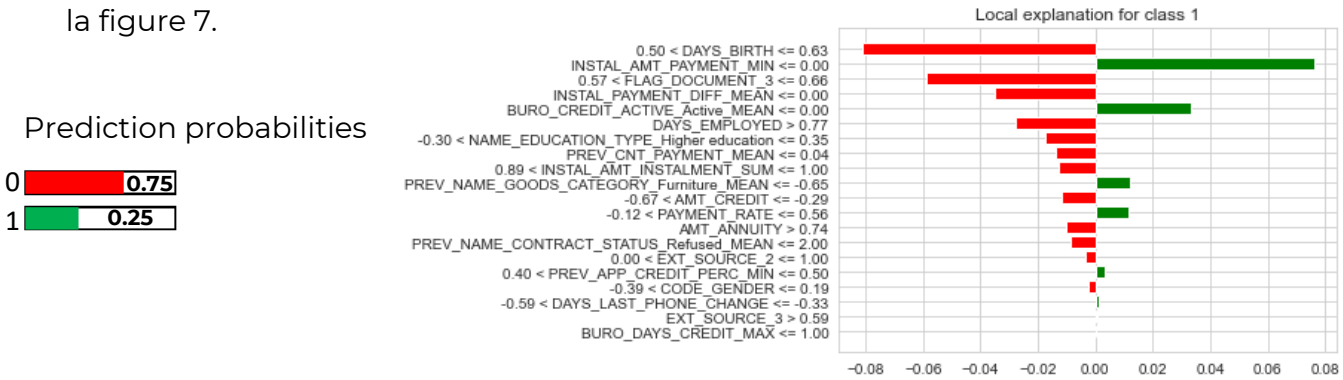


Figure 7 : Explication LIME

La solvabilité du client est supérieur au seuil de solvabilité de 0.5, donc le crédit sera accordé au client. Parmi les variables qui jouent le plus en sa faveur, son âge et son éducation. A l'inverse, la variable « INSTAL_AMT_PAYMENT_MIN » qui représente son historique en crédit joue le plus en sa défaveur.

- **SHAP**

  Pour un individu donné, la valeur de Shapley d'une variable (ou de plusieurs variables) est sa contribution à la différence entre la valeur prédite par le modèle et la moyenne des prédictions de tous les individus. Pour ce faire :

  - 1ère étape : calculer les valeurs de Shapley pour un individu en particulier : simuler différentes combinaisons de valeurs pour les variables d'entrée.
  - 2ème étape : Pour chaque combinaison, calculer la différence entre la valeur prédite et la moyenne des prédictions. La valeur de Shapley d'une variable correspond alors à la moyenne de la contribution de sa valeur en fonction des différentes combinaisons.

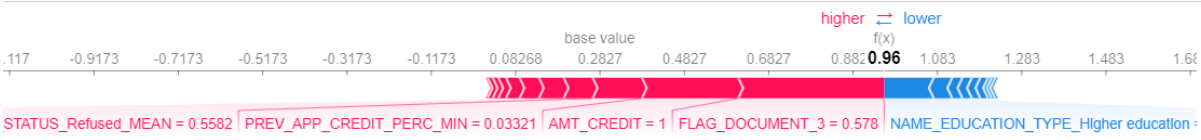


Figure 8 : Explication SHAP

Le force plot est bon pour voir où se place le «output value» par rapport à la «base value». L'analyse de la figure 8 montre une prédiction de 0.96 et la valeur de base du dataset est à 0.2827 (moyenne des valeurs de la variable cible).

En rouge, les variables qui ont un impact positif (contribuent à ce que la prédiction soit 1) et, en bleu, celles ayant un impact négatif (contribuent à ce que la prédiction soit 0).
Les caractéristiques qui étaient importantes pour faire la prédiction de cette observation sont affichées en rouge et bleu, le rouge représentant les caractéristiques qui ont fait augmenter le score du modèle et le bleu les caractéristiques qui ont fait baisser le score. Les caractéristiques qui ont eu plus d'impact sur le score sont situées plus près de la limite de séparation entre le rouge et le bleu, et la taille de cet impact est représentée par la taille de la barre. Ainsi, le crédit de cette personne en particulier a été refusé, car elle a été poussée plus haut par tous les facteurs indiqués en rouge. En effet, la variable «FLAG_DOCUMENT_3» représente la variable la plus importante qui est en défaveur de ce client.

7. Limite et amélioration du modèle

- ***Seuil de solvabilité***

Le modèle retourne un score entre 0 et 1 et par défaut il attribue la classe 1 lorsque le score est supérieur à 0.5 et 0 sinon. Avec ce seuil, la modèle présente un score F5 de l'ordre de 62%. La diminution de ce seuil qui permet de classifier un client s'il est solvable ou pas pourrait améliorer les performances.

- ***Équilibre des classes***

La technique d'over-sampling ou de undersamling a un impact sur les performances du modèle.

- ***Réduction dimensionnelle***

Afin de réduire le nombre de variables sur lesquels le modèle a été entrainé, nous avons sélectionné les vingt plus importantes features pour le modèle. Une autre alternative qui pourrait probablement améliorer la fonction coût et d'utiliser une technique de réduction dimensionnelle (ACP ou t-SNE) bien que l'interprétation des variables soit par la suite difficile.
Nous pouvons aussi utiliser un algorithme de sélection de features de Scikit Learn comme RFE (Recursive Feature Elimination).

- ***Fonction coût***

La modélisation a été effectuée sur la base d'avoir le meilleur F5 score qui donne beaucoup d'importance au rappel (recall). L'axe principal d'amélioration serait de définir plus précisement une fonction coût plus adaptée en collaboration avec l'équipe métier *de « Prêt à dépenser ».*