NOTE METHODOLOGIQUE

Victoire Mohebi

Projet 07 : « Implantation d'un modèle de scoring »

Open Classrooms - Parcours *Data scientist*

Table des matières

1. Problématique & Identification du problème

Société financière "Prêt à dépenser" propose des crédits à la consommation pour des personnes ayant peu ou pas d'historique de prêt.
Dans le but de la prise de la décision d'accorder ou ne pas accorder un prêt à un client, l'entreprise souhaite développer un modèle de *scoring* de la probabilité de défaut de paiement du client.
Ce projet est constitué de deux grandes parties :

- **Construire un modèle du scoring** qui en s'appuyant sur les données comportementale et financières de client (les données sont disponibles sur le KAGGLE), donnera d'une façon automatique la prédiction et la probabilité que celui-ci ne rembourse pas le prêt.
- **Construire un dashboard interactif** pour le gestionnaire du service de relation des clients. Ce dashboard permet d'interpréter les prédictions.

Dans ce document, nous allons nous focaliser essentiellement sur la méthodologie de la modélisation en détaillant toutes les étapes de la construction du modèle.
La démarche suivie pour élaborer le dashboard interactif est présenté dans la présentation.
Comprendre les enjeux businesse d'une entreprise et identifier ses problèmes est l'étape la plus importante dans un projet de Data Science.
Dans le cas de l'entreprise « Prêt à dépenser », la fiabilité des prédictions de performance vs les résultats réels est primordial.

Figure 1: cycle de vie d'un projet data science

2. Modélisation

Nous avons récupéré les données dites annotées, pour entrainer notre modèle. Nous avons donc affaire avec un apprentissage supervisé.
On devra prédire le résultat pour le label qui se présentent sous forme de deux catégories. Il s'agit donc d'un problème de classification. Le modèle d'apprentissage automatique classera les clients en deux classes les clients qui remboursent le prêt les clients qui ne remboursement du prêt. Nous devrons donc faire recours aux algorithmes de classification binaire.
Nous avons suivi le cycle de vie d'un projet de data science, illustré dans la figure 1.

3.Compréhension de données & EDA

Le jeu de données est composé de plusieurs tables. (Figure 2). Pour chaque client il existe au total 218 variables comportementales et financières sur l'emploi, le cadre de vie, l'historique de crédit, etc. Plus 300000 demandes de crédit sont enregistrées.
Chaque client est présenté par son identifiant « SK_ID_CURR ». Un label de 0 ou 1 lui est associé suivant qu'il est solvable ou non.
La classe 0 signifie que le client n'a pas difficulté de rembourser le prêt et la classe 1 signifie que le client n'a pas pu rembourser la dette.

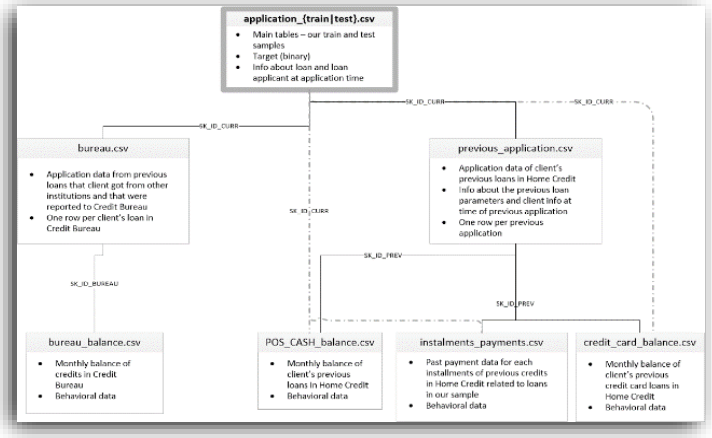

Figure 2 : différents données des client sont reliés avec le clé primaire de leur identifiant

Analyse exploratoire

L'analyse exploratoire nous permet de comprendre mieux l'ensemble de données. Cette étape donne aussi une image claire des variables et de leur relation.
Notre objectif, dans cette étape a été de nous focaliser sur les variables qui peuvent être important dans la prise de décision de l'attribution du crédit. Ceci nous aide aussi de choisir une stratégie pertinente pour gérer les gérer les valeurs manquantes et identifier les valeurs aberrantes.
En se basant sur l'analyse exploratoire, nous avons pu construire la base finale de données nécessaire à l'élaboration du modèle.
L'exploration des données révèle un problème de déséquilibre des classes qu'il faut régler. On a pu constater que 92 % des prêts ont été remboursés et que seulement 8 % des individus ont été non solvables. (Figure 3)

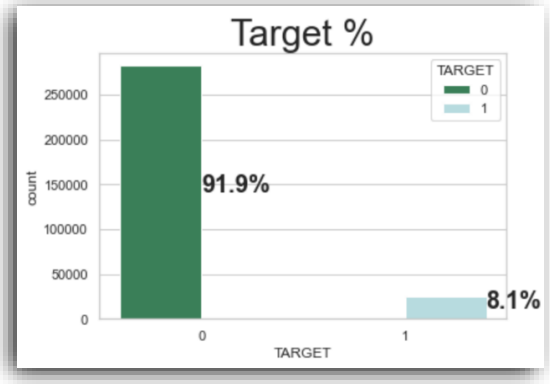

Figure 3 : la distribution de variable

4. Prétraitement des données

Préparer les données avant la modélisation est une étape importante dans un projet de data science. Cela permet d'améliorer la performance de l'algorithme.
Le prétraitement consiste en générale à nettoyer le jeu de données, en traitant les valeurs aberrantes et valeurs manquantes.
Il faudra aussi encoder les données catégorielles pour le rendre utilisable pour l'algorithme.
La préparation de données nécessaires à l'élaboration du modèle de *scoring* a été réalisé en s'inspirant d'un kernel fourni par l'équipe métier.
Nous avons ajouté d'autre étapes de pré_processing des données, dont le *feature scaling.*

4.1. Feature engineering

Cette étape est réalisée par la jointure entre les différentes tables. Les fonctions utilisées agréation des variables numérique sont les suivantes : *min, max, sum, mean*.

- **Création des feature métier :**
  Dans cette étape nous avons aussi créé *features métier* comme la proportion du temps travaillé par rapport à l'âge du client, ou le ratio crédit/revenu, etc.
- **Traitement des valeurs aberrantes :**
  - Pour certaines variables catégorielles, des valeurs aberrantes apparaissent comme « XNA » (comme ''CODE_GENDER''). Ces valeurs sont très peu nombreuse. Elles ont été donc supprimées.
  - Pour le variable numériques, DAYS_EMPLOYED, les valeurs aberrantes > 1000 ans ont été remplacées pap NaN.
- **Encodage des variables catégorielles**
  Pour l'encodage des variables catégorielles on a utilisé la méthode de One-Hot-Encoder. Cette méthode consiste à coder chaque variable catégorielle avec différentes variables booléennes (aussi appelées variables factices) qui prennent les valeurs 0 ou 1, indiquant si une catégorie est présente dans une observation.

4.2. Traitement des valeurs manquantes

Les variables ayant plus de 40% de valeurs manquantes ont été supprimées. Nous avons donc 552 variables. Les autres valeur manquantes ont été remplacées par la valeur *median.*

4.3. Sélection de caractéristiques (Feature selection)

Après l'étape de feature engineering (par la jointure des tables, etc) nous avons obtenu une table avec 798 variables. Afin de sélectionner les variables plus pertinentes à la modélisation, nous avons supprimé les variables suivantes :

- Celles dont la variance a été < 0.0005.
- Les variables ayant une corrélation < 0.01 avec la variable cible (« Target ») ; et les variables fortement corrélées entre elles. (Le coefficient de corrélation > 0.8)
- Celles qui donnent des informations sur le numéro de portable du client et que nous avons estimé non nécessaire à l'élaboration du modèle.

4.4. Feature scaling

Les ordres de grandeurs des variables numériques sont très différents dans notre dataset final. D'où la nécessité de réaliser le **Feature Scaling**.
On a eu recours à la *Standardisation* qui est le processus de transformer une variable en une autre qui répondra à la loi normale.

4.5. Split Train/Test set

Les données sont divisées en données d'entrainement (80%) et de donnée de test (20%) avec la méthode de *train_test_split* de *Scikit-Learn*. Afin de garder les mêmes proportions des différentes classes, on a utilisé l'argument *stratify*.

5. Sélection du modèle et la méthode d'entrainement

Il existe différents algorithmes plus ou moins performants et plus ou moins complexes, donc plus ou moins interprétables pour un problème de classification binaire.
Nous avons sélectionné des algorithmes suivants : *Dummy, Classifier, Régression Logistique, Forêts aléatoire et LGBM.*
Nous les avons testés sur leur paramètre par défaut afin d'avoir un aperçu de leur résultat sur nos données.
Comme on peut constater dans le tableau ci-dessous, le modèle LGBM qui est un algorithme basé sur le « *gradiens boosting* » avaient le meilleurs AUC.

	Model	AUC	Accuracy	Precision	Recall	F1	F5	Time
3	LGBMClassifier	0.773796	0.930654	0.551282	0.025982	0.049625	0.02697	17.276524
1	LogisticRegression	0.755995	0.930373	0.516129	0.01289	0.025152	0.013392	7.034355
2	RandomForestClassifier	0.714623	0.930373	0.833333	0.001007	0.002012	0.001047	742.990809
0	DummyClassifier	0.5	0.930317	0.0	0.0	0.0	0.0	0.448258

5.1 Choix des métriques spécifiques

De nombreux indicateurs sont utilisés pour évaluer la performance d'un algorithme de l'apprentissage automatique.
Avec un jeu de données de deux classes, où la première classe représente 90% des données, si le classifieur prédit que chaque exemple appartient à la première classe, l'exactitude sera de 90%, mais ce classifieur est inutile dans la pratique.
L'utilisation de mesures plus simples comme le "*Precision*" ou "*Accuracy*" peut être donc trompeuse.
D'autres métriques sont plus pertinentes dans le cas du déséquilibre de classes.

Matrice de confusion

La matrice de confusion est en quelque sorte un résumé des résultats de prédiction pour un problème particulier de classification. Elle compare les données réelles pour une variable cible à celles prédites par un modèle. Les prédictions réelles et fausses sont révélées et réparties par classe, ce qui permet de les comparer avec des valeurs définies. (Figure 4)

Métrique technique : AUC_ROC

Il existe deux groupes de mesures qui peuvent être utiles pour la classification déséquilibrée car elles se concentrent sur une classe. Ce sont la *Sensitivity-Specificity* et la *Precision-Rappel.*

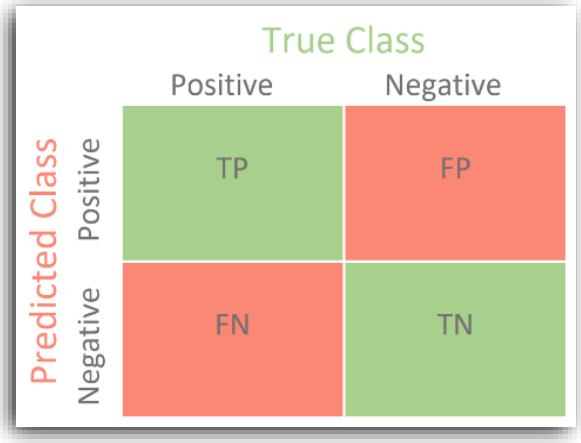

Figure 4 : La Matrice de confusion

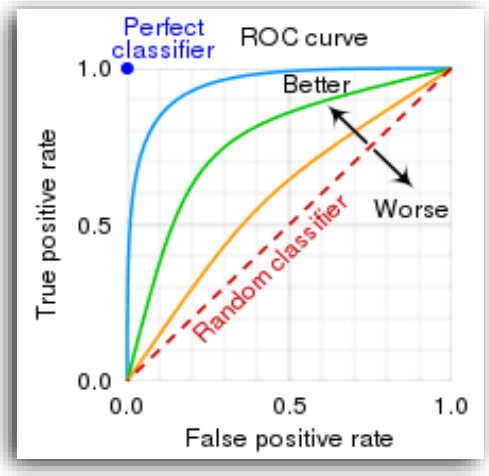

Figure 5: AUC_ROC

Sensitivity-Specificity :

La ***Sensitivity*** fait référence au taux de vrais positifs (TP) et résume à quel point la classe positive a été prédite.

$$Sensitivity = TP /(TP+FN)$$

La ***Specificity*** est le complément de la Sensitivity, ou le taux de vrais négatifs (TN), et résume à quel point la classe négative a été prédite.

$$Specificity = TN /(TN+FP)$$

Pour une classification déséquilibrée, la *Sensitivity* pourrait être plus intéressante que la *Specificity*.

Au lieu d'exactitude (*Accuracy*), nous avons donc utilisé la métrique de *AUC_ROC* (**A**rea **U**nder the **C**urve-**R**eceiver **O**perating **C**haracteristic). La courbe ROC représente la *Sensitivity* en fonction de (1 – *Specificity*)

L'aire sous la courbe ROC (Figure 5) 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.

Métrique métier : F-Score

Le choix des métriques d'évaluation d'un algorithme est lié aussi au contexte et l'objectifs de l'entreprise.

Dans le cas de ce projet le taux d'erreur ainsi que la proportion de faux-positifs et faux-négatifs sont important. En effet, 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).

Rappel- Precision :

Precision-Recal est le second group de métriques utilisées pour une classification déséquilibrée.

La ***Precision*** résume la fraction d'exemples affectés à la classe positive qui appartiennent à la classe positive.

$$Precision = TP /(TP+FP)$$

Tandis que le *Recall* résume à quel point la classe positive a été prédite et correspond au même calcul que la ***Sensitivity***

$$Recall = TP /(TP+FN)$$

La précision et le rappel peuvent être combinés en un seul score appelé le score F ou la mesure F. Ceci cherche à équilibrer les deux préoccupations (Precision & Recall).
La mesure Fβ- est une abstraction de la F-mesure où l'équilibre entre précision et rappel dans le calcul est contrôlé par un coefficient appelé β.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

Pour β >1, nous accordons plus d'importance au *Recall*. Fscore est donc le score qui permet de minimiser les faux-négatifs (client non solvable classifiés comme ceux qui ont repayé des crédits) sans ignorer totalement les faux positifs (crédits repayés classés comme défauts de paiement.

5.2. Comment résoudre le problème de la classification déséquilibrée ?

On a constaté que la majorité des clients repayant leur crédit. Le déséquilibre des classes de la variable cible augmente nettement la difficulté de l'apprentissage par l'algorithme de classification. Cela aboutit au surajustement.

Une technique largement adoptée pour traiter des ensembles de données très déséquilibrés est appelée rééchantillonnage. Il existe plusieurs méthodes de rééchantillonnage :

- ***Sous-échantillonnage des observations majoritaires*** : on retire aléatoirement des observations majoritaires. (Figure 6)
- ***Suréchantillonnage des observations minoritaires*** : on tire au hasard des individus minoritaires que l'on rajoute aux données. (Figure 6)
- ***Hybrides*** : combinaison des deux stratégies : le sous-échantillonnage et le suréchantillonnage pour augmenter l'efficacité de la gestion de la classe déséquilibrée.

Appliquer la méthode d suréchantillonnage à l'ensemble de données volumineuses est parfois couteux en terme du temps de calcul. De nombreux algorithmes de *Scikit-Learn* ont un paramètre *class_weights* qui peut être défini comme "balance" pour déterminer comment classer l'importance des données déséquilibrées. Cette méthode s'apparente au suréchantillonnage. Au lieu de suréchantillonner réellement nous pouvons informer l'estimateur d'ajuster la façon dont il calcule la perte.

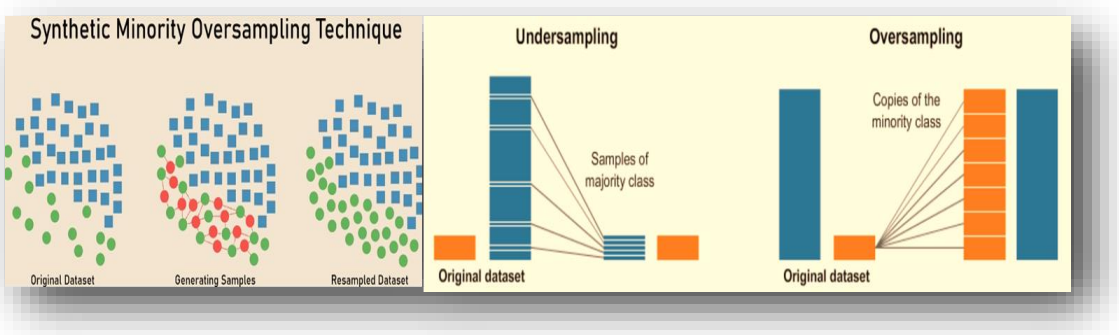

Figure 6 : différentes méthodes de resampling

Pour gérer le déséquilibre des classes, nous avons utilisé les deux approches suivantes :

- **Le sur-échantillonnage synthétique** (SMOTE _**S**ynthetic **M**inority **O**versampling **T**echnique) produit des observations minoritaires ressemblantes mais distinctes de celles déjà existantes. La proportion des deux classes avant et après rééchantillonnage est illustrée dans la figure 7.
- **Le paramètre class_weight** qui force l'algorithme à apprendre en fonction d'importance ("poids") donnée à une classe particulière.

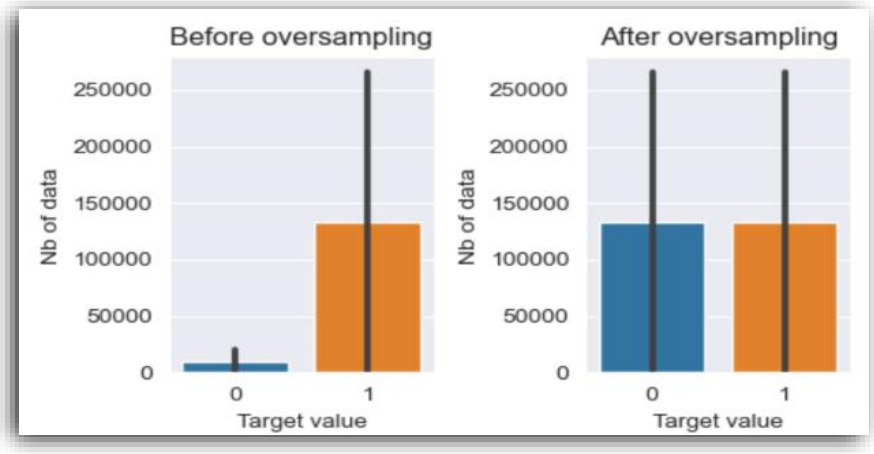

Figure 7: la distribution de variable cible avant et après le rééchantillonnage

5.3 Optimisation des hyperparamètres

Après avoir régler le problème des classe déséquilibrées, nous avons cherché la combinaison des hyperparamètres qui donne le meilleur *F5 score*. Nous avons ajusté «*max_depth*» et «*num_leaves*». Le premier fixe une limite de la profondeur de l'arbre et le second contrôle a complexité du modèle, en spécifiant le nombre de feuilles. 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.

5.4 Enregistrement du modèle

Après avoir entraîné et optimisé le modèle, nous l'avons enregistré dans un fichier « *sav* ». Pour faire cela, nous avons utilisé la librairie *pickle*. Cette démarche est intitulée sérialisation. Le sauvegarde du modèle entraîné nous permets de pouvoir l'utiliser ultérieurement dans les applications. Nous avons ensuite téléchargé (désérialisé) le modèle avec *joblib* afin de l'utiliser dans le API.

6. Interprétabilité du modèle

Les modèles de classification ont deux objectifs principaux. Tout d'abord, ils doivent bien fonctionner, ce qui signifie qu'ils doivent prévoir la sortie pour de nouvelles caractéristiques d'entrée données aussi précisément que possible. Deuxièmement, ils doivent être interprétables, c'est-à-dire fournir une certaine compréhension entre les caractéristiques d'entrée et la sortie. Il existe deux grandes catégories des approches d'interprétation des modèles :

Interprétabilité globale *ou feature importance* :

L'interprétabilité globale consiste à comprendre comment le modèle en fournissant des informations sur l'importance des variables ainsi que sur les données utilisées .

Interprétabilité locale ou explicabilité du modèle :

L'explicabilité ou interprétabilité local du modèle consiste à pouvoir expliquer pourquoi le modèle a donné telle prédiction. Pour l'interprétabilité locale il faut changer d'échelle afin d'extraire des informations locales pour des exemple spécifique du jeu de données.

Nous avons utilisé pour l'interprétabilité locale l'algorithmes SHAP (**SH**apley **A**dditive ex**p**lantions).

6.1. Interprétabilité globale

Nous avons utilisé l'attribut de *feature_importances_* de l'algorithme LightGBM, pour calculer l'importance de chaque caractéristique.

Un score plus élevé signifie que la variable spécifique aura un effet plus important sur le modèle utilisé pour prédire la variable cible. Nous avons créé un plot pour les 20 variables qui ont plus impact sur le modèle. (Figure 8).

Nous avons aussi utilisé le plot « *summary_plot* » de *SHAP* pour combiner feature importance et feature effets. Ce graphique nous a permis d'avoir aussi une interprétation globale du sens de l'impact des variables sur la prédiction 0 ou 1. (Figure 8)

Chaque point représente la *Shapley Value* pour une observation. Le couleur rouge désigne la classe 1 et la couleur bleue indique la classe 0.

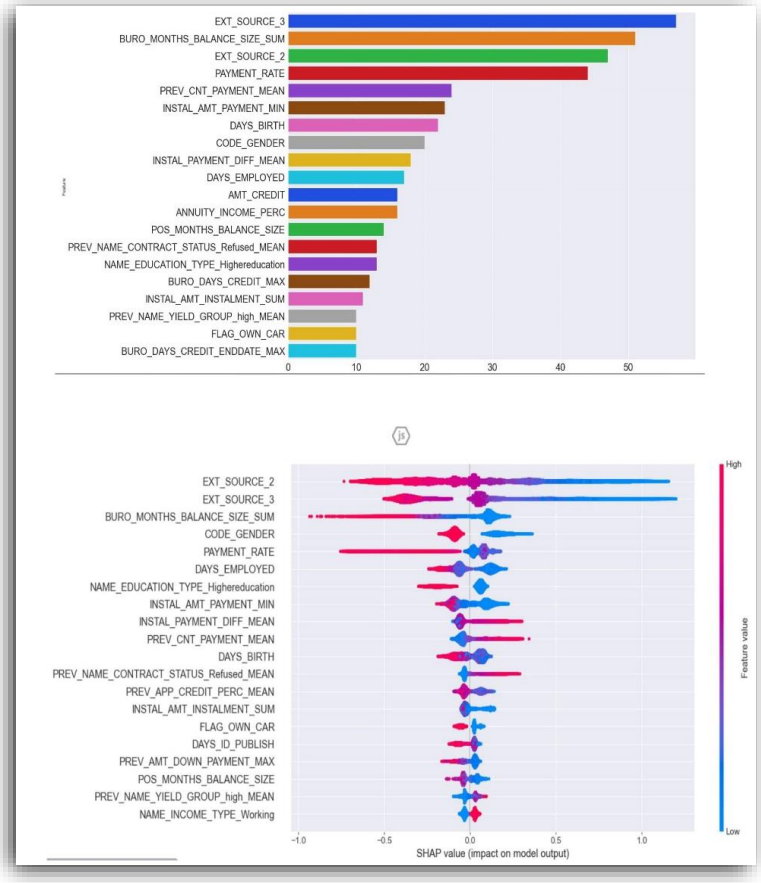

Figure 8 : Le graphique en haut montre les variables qui sont important pour le modèle. Celui en bas représente l'importance globale des variables calculées par les valeurs de Shap

Par exemple « EXTERNAL SOURCE2 » qui est la variable la plus importante, a un impact négatif quand la valeur de cette variable est élevée.

6.2. Interprétabilité locale

La méthode locale est plus utile pour interpréter un modèle de scoring, car elle permet d'interpréter le score d'un individu donné. Nous avons utilisé *Shap force plot* pour montrer quelle variable a plus d'impact sur la prédiction d'un individu.

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.

Le force plot est bon pour voir où se place le « output value » par rapport à la « base value».

Sur la figure 9, deux exemples ont été représentés. La valeur de base du dataset est à 0.4028 (moyenne des valeurs des de la variable cible), la prédiction du score des clients est 0.54 et 0.68.

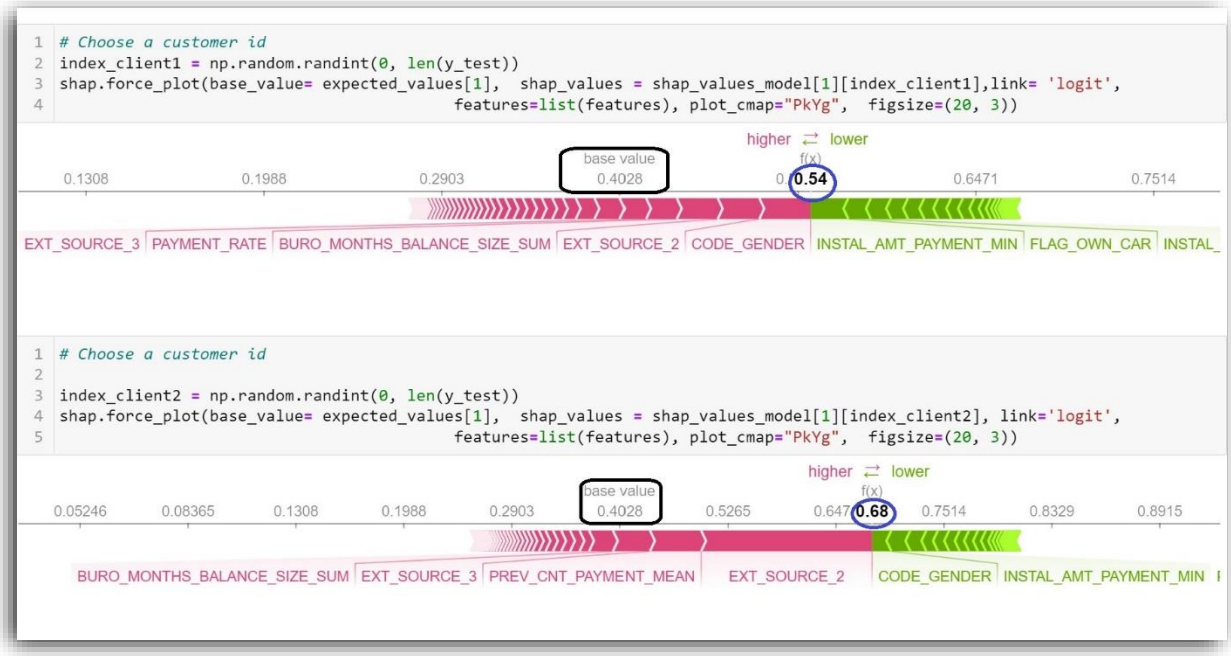

Figure 9 : Compare le « Shap force » pour deux clients non solvables

Les caractéristiques qui étaient importantes pour faire la prédiction de cette observation sont affichées en rose et vert, le rose représentant les caractéristiques qui ont fait augmenter le score du modèle (elle ont donc un impact positif et contribuent à ce que la prédiction soit 1) et le vert, les caractéristiques qui ont fait baisser le score (elle ont un impact négatif et contribuent à ce que la prédiction soit 0)

Les caractéristiques qui ont eu plus d'impact sur le score sont situées plus près de la limite de séparation entre le rose et le vert, et la taille de cet impact est représentée par la taille de la barre.

Ainsi, le crédit de ces deux personnes en particulier a été refusé, car elles sont été poussée plus haut par tous les facteurs indiqués en rose. Nous voyons cependant que les raisons du modèle pour que leur crédit ne soit pas accepté sont différentes : contribution positive de « CODE GENDER » et «EXT_SOURCE » pour la première, et contribution positive de «EXT_SOURCE » et « PREV PAYMENT MEAN » pour le deuxième.

7. Limites et recommandations

Les axes d'amélioration pourraient être faites avant et pendant la modélisation.

7.1. Piste d'amélioration avant la modélisation :

Problème de la classification déséquilibrées : Pour rééquilibrage des données, nous avons essayé le suréchantillonnage avec SMOTE et le paramètre « class_weight » de l'algorithme LightGBM. On pourrait essayer d'autre stratégies comme **cluster de la classe majoritaire.**

Traitement des données manquantes : Nous avons supprimé les variable avec plus de 40% valeur manquantes et imputer les autres avec la médiane. On pourrait essayer d'autre approche comme *knn* pour les imputer

Réduction dimensionnelle pour un meilleur feature extraction : Afin de réduire le nombre des variables utilisées pour l'entraînement du modèle, on pourrait réaliser un PCA ou ACP (Analyse en composantes principales)

Echange avec l'équipe métier : Echanger avec l'équipe métier nous permettra de sélectionner les variables les plus pertinentes

7.2. Piste d'amélioration pendant l'entrainement du modèle

Ajustement des hyperparamètres : Ajuster plus de paramètres de l'algorithme pourrait être un autre axe d'amélioration pour obtenir un modèle plus performant.

Définition une fonction coût adaptée aux besoins de l'entreprise : La modélisation et son évaluation a été effectuée sur la base d'avoir le meilleur score F5 qui donne beaucoup d'importance au *Rappel* et qui diminue le taux de faux négatif. L'axe principal d'amélioration de la modélisation serait de définir avec l'équipe métier, une fonction coût plus adaptée aux besoins de l'entreprise *« Prêt à dépenser ».*