OPENCLASSROOMS

OpenClassRooms – Parcours Data Scientist

Note Méthodologique

Projet n°7 : Implémentez un modèle de scoring

Rédigé par

Mohamed Sidina SID AHMED

mohamed.sidina@gmail.com

Le 09 février 2022

OPENCLASSROOMS

1. Contexte

Cette note méthodologique s'inscrit dans le cadre du projet n°7 du parcours Data Scientist d'OpenClassrooms.

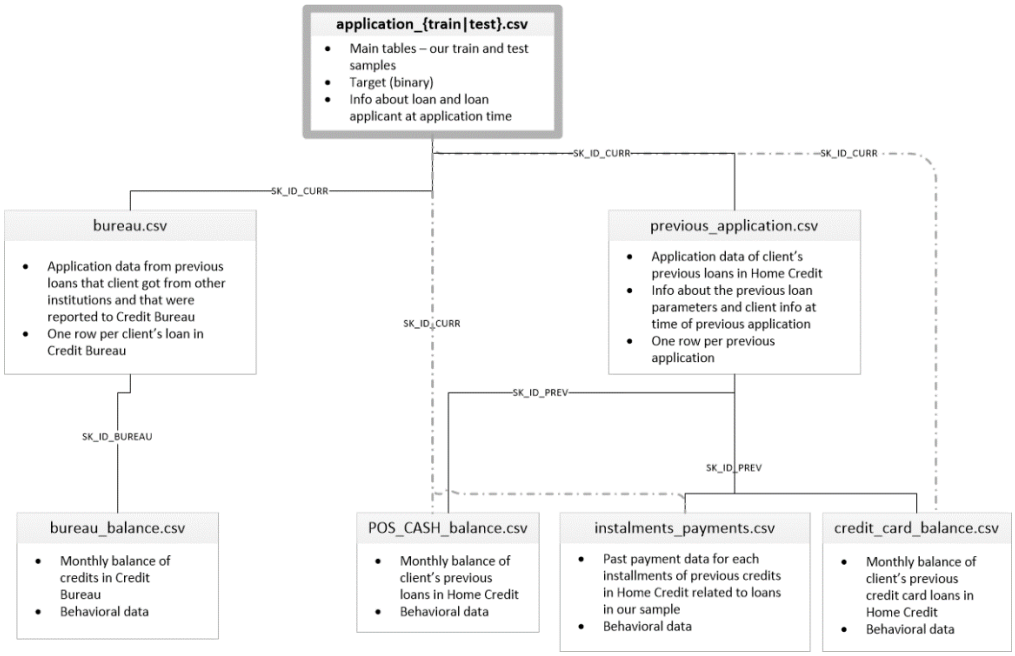
Le projet consiste à développer pour la société « Prêt à Dépenser » :

- Un modèle de scoring de la probabilité de défaut de paiement d'un client, en s'appuyant sur des données variées (données comportementales, données provenant d'autres institutions financières, etc.).
- Tableau de bord interactif pour que les chargés de relation client puissent expliquer de la façon la plus transparente possible les décisions d'octroi de crédit.

Cette note décrit en détail le processus de modélisation et d'interprétabilité du modèle mis en place dans le cadre du projet.

2. Les données

Le jeu de données utilisé pour entrainer le modèle, est accessible sur le site Kaggle.
La structure des données est expliquée dans l'image suivante :



Le dataframe ***application_train*** et ***application_test*** contiennent la liste des prêts et des demandeurs.
Le dataframe ***bureau*** contient les données d'autres demandes de prêts que le client a contractés auprès d'autres établissements de crédit et qui ont été signalés au bureau de crédit.

Le dataframe ***previous_applications*** contient des informations sur l'historique des prêts précédents chez la société « Prêt à Dépenser » par le même client.

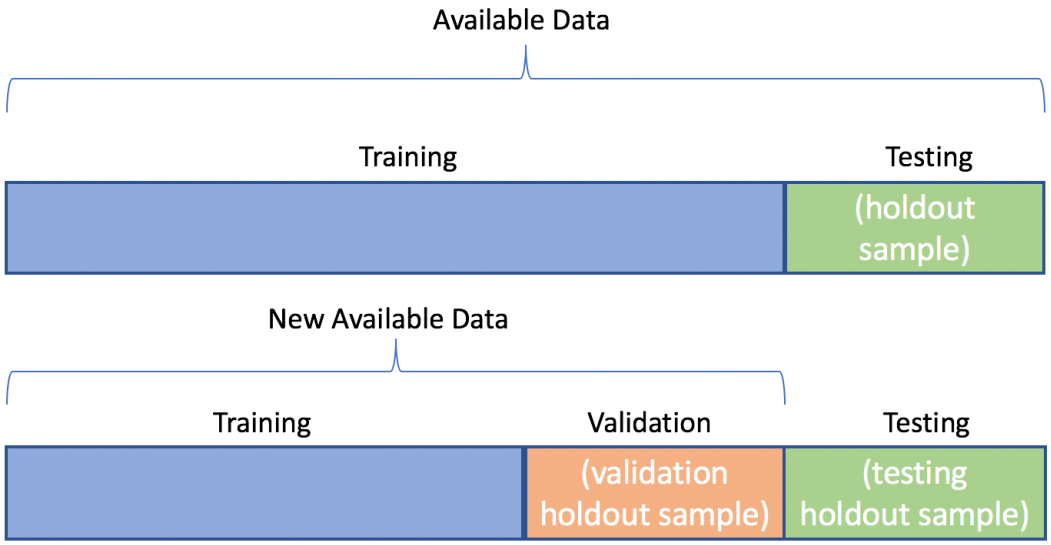
- La clé **SK_ID_CURR** connecte les dataframes ***application_train|test*** avec ***bureau***, ***previous_application*** et aussi avec les dataframes ***POS_CASH_balance***, ***versements_payment*** et ***credit_card_balance***.
- La clé ***SK_ID_PREV*** connecte le dataframe previous_***application*** avec ***POS_CASH_balance***, ***instalements_payment*** et ***credit_card_balance***.
- La clé **SK_ID_BUREAU** connecte le dataframe ***bureau*** avec dataframe ***bureau_balance***.

2. La méthodologie d'entraînement du modèle

2.1 Séparation du jeu de données

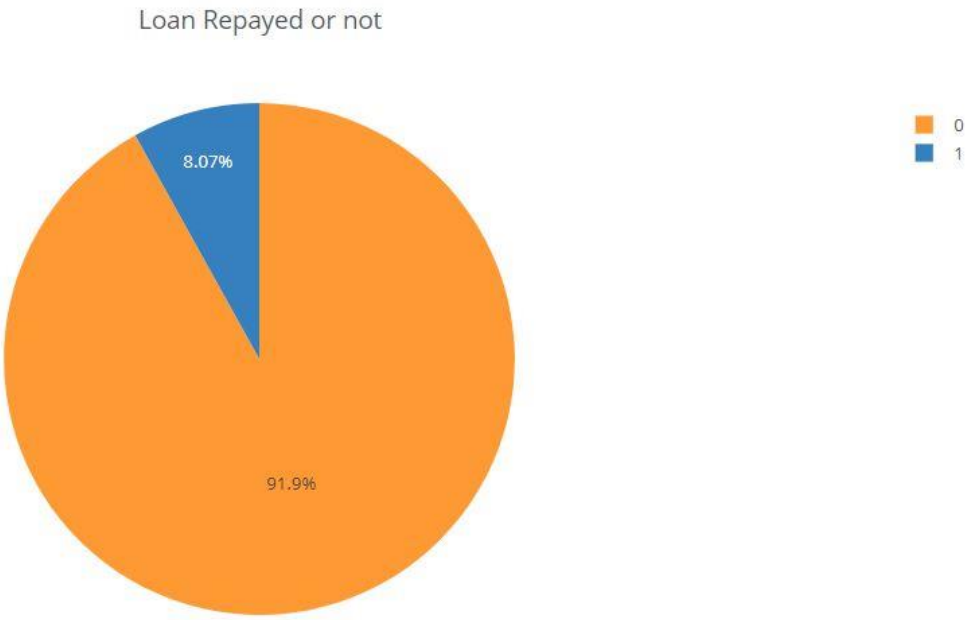
Le jeu de données initial a été séparé en **3 parties** de façon à disposer :

- Un **Training set** qui contient **60%** des individus, pour entraîner les différents modèles.
- Un **Validation set** qui contient **20 %** des individus, pour optimiser les paramètres sans overfitting.
- Un **Test set** qui contient **20 %** des individus, pour l'évaluation finale du modèle.



« Figure n°1 : Séparation du jeu de données »

À la suite de l'analyse exploratoire, il s'avère que nous avons un jeu de données fortement déséquilibré (8 % de clients en défaut de remboursement de crédit contre 92 % de clients sans défaut).



« Figure n°2 : Etude équilibre des classes du jeu de données »

Afin de garantir une meilleure généralisation du modèle, ce déséquilibre des classes doit être pris en compte dans l'entraînement des modèles.

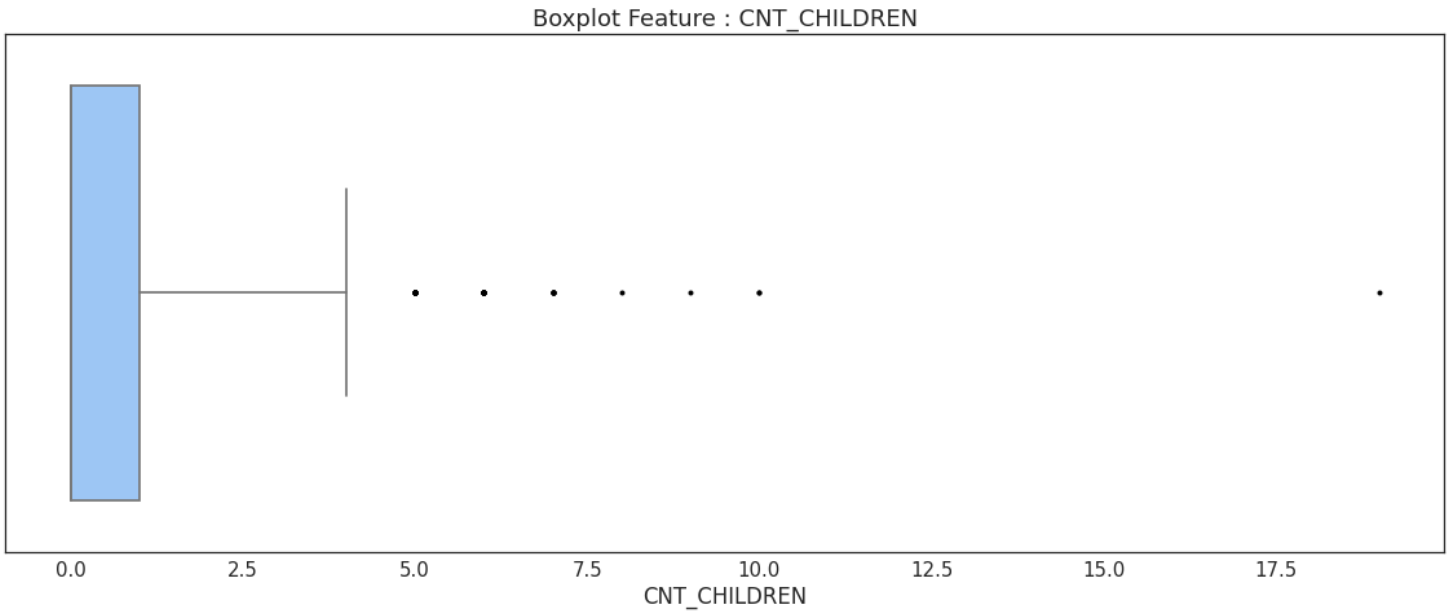
Deux approches pour rééquilibrer les deux classes sont possible à tester : Class Weights et re-sampling avec SMOTE :

- Class Weights : permet de modifier les poids associés aux observations de sorte qu'une observation mal classée dans la classe minoritaire pénalise davantage la fonction de perte qu'une observation mal classée dans la classe majoritaire.
- Re-sampling avec SMOTE : permet de créer des données synthétiques à partir du jeu de données existantes en utilisant différentes techniques (Under-sampling, over-sampling….etc.)

Dans le cadre du projet, nous avons choisi d'utiliser la première approche à savoir (Class Weights) pour sa simplicité de mise en œuvre ainsi qu'un problème de limitation des ressources physique de notre machine.

2.2 Traitement des outliers

Lors de l'analyse exploratoire de jeu de données, des outliers ont été détectés dans certains features.
Les outliers présents dans chaque feature, ont été remplacé par la médiane de celui-ci, car la médiane est plus robuste aux outliers.



« Figure n°3 : Exemple d'outliers détecté »

2.3 Imputation des valeurs manquantes

Pour l'imputation des valeurs manquantes, nous avons utilisé la classe SimpleImputer de Sklearn, car elle fournit des stratégies de base et rapide pour l'imputation des valeurs manquantes :

- Pour les features de type catégoriel ou booléen, les valeurs manquantes ont été remplacés par **le mode** (la valeur la plus fréquente).
- Pour les features de type numérique, les valeurs manquantes ont été remplacés par **la médiane.**

2.4 Sélection du meilleur modèle

Afin de répondre la problématique du projet, nous allons tester et comparer la performance de deux modèles :

- Un modèle linéaire : Logistic Regression Classifier
- Un modèle non-linéaire : LightGBM Classifier

2.5 Feature Engineering

Le processus pour choisir le meilleur modèle qui réponde à la problématique de notre client se compose de deux étapes :

- Etape n° 1 : Entrainer les modèles sans feature engineering utilisant uniquement le jeu de données application_train.
- Etape n° 2 : Créer de nouveaux features avec la bibliothèque **FeatureTools**, qui inclut la possibilité de fusionner tous les datasets de données et d'automatiser la création de features.

3. La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

3.1 La fonction coût métier

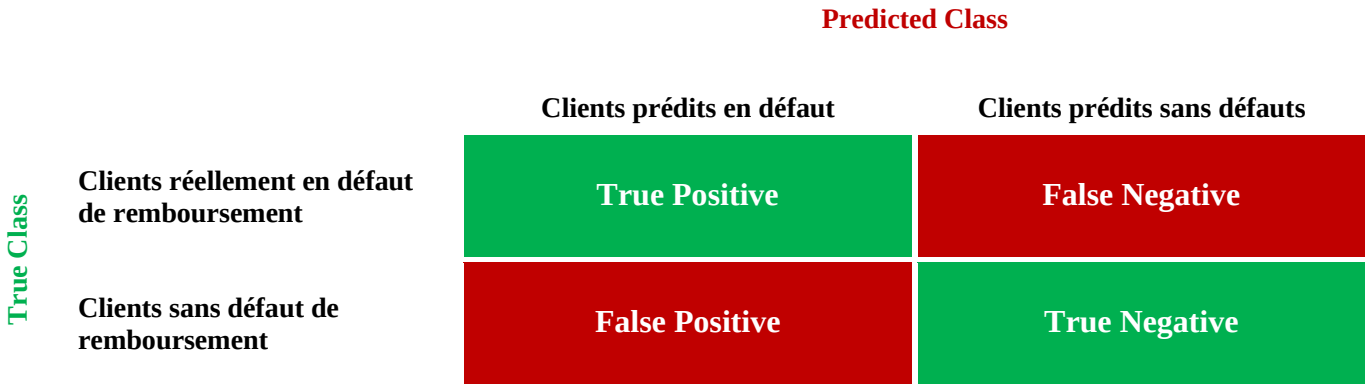
Les deux modèles entrainés utilisent la même fonction de coût métier à savoir :

Modèle	Fonction coût métier	Définition
Logistic Regression Classifier	**Binary Cross Entropy** connue aussi comme **Log Loss**	**Log Loss** est la moyenne négative du **log** des probabilités prédites corrigées pour chaque instance.
LightGBM Classifier		

3.2 Métrique d'évaluation

Sachant que nous disposons d'un ensemble de données extrêmement déséquilibré, l'utilisation d'indicateurs classiques tels que " **Accuracy** " n'est pas appropriée. En effet, ce type d'indicateur n'est pas en mesure de distinguer le nombre de clients correctement classés dans les différentes classes.

L'un des outils permettant de visualiser et d'identifier les classifications des clients dans différentes classes est une matrice confusion :



Pour notre client la société « Prêt à Dépenser », la mauvaise classification d'un client à risque de défaut peut être très problématique, risquant en fait des pertes financières pour l'entreprise et une perte de temps pour les agents, à cause de la gestion de tous les actes administratifs accomplis à cause ce problème de mal classification.
Donc nous avons donc essayé de réduire le pourcentage de « **False Negative** » et d'augmenter le pourcentage de « **True Positive** ».

En d'autres termes, nous essayons de **maximiser** l'indicateur "**True Positive Rate / Recall** " et de **minimiser** l'indicateur "**False Positive Rate** (1 - Recall)" :

- True Positive Rate / Recall = $\frac{TP}{TP + FN}$. Recall nous donne une indication sur la proportion de la classe positive qui a été correctement classifiée.

- False Positive Rate / Specificity = $\frac{FP}{TN + FP} = 1 - Recall$. Specificity nous donne une indication sur la proportion de la classe négative qui a été correctement classifiée.

Un indicateur de performance qui **prend en compte** les deux métriques (**Recall et Specificity**) est l'**AUC** (Area Under the Curve) qui est une **mesure** de la **capacité** d'un **modèle** à **distinguer** la **classe positive** de la classe **négative**.

Plus la valeur de l'AUC est élevée, meilleure est la performance du modèle de distinction entre les classes positives et négatives. Dans la suite du projet, nous cherchons à avoir un AUC score d'une valeur entre 0,5 et 1, car cela indique qu'il y a de fortes chances que le modèle distingue les valeurs de classe positive des valeurs de classe négative. En fait, le modèle est capable d'identifier plus de résultats vraiment positifs et vrais négatifs que de faux négatifs et de faux positifs.

3.3 Algorithme d'optimisation

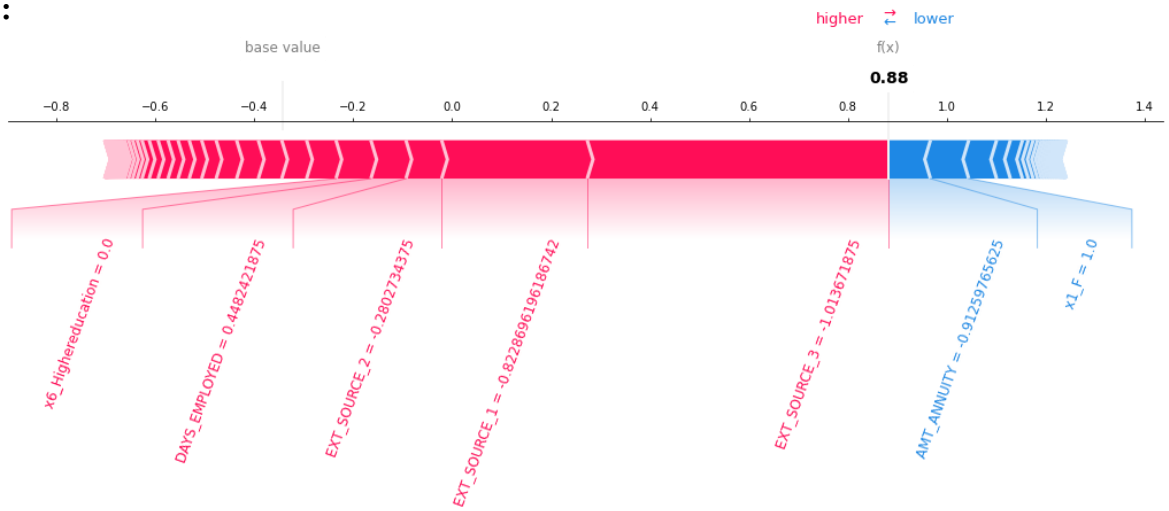
Pour l'optimisation et le choix des meilleurs hyperparamètres pour chaque modèle (Logistic Regression Classifier et LightGBM Classifier), nous avons utilisé la librairie **Optuna** qui est un framework automatisé de pointe pour l'optimisation des hyperparamètres.

Optuna, nous permet d'utiliser la méthode Bayésienne pour l'optimisation. Cette méthode fournit une technique de principe basée sur le théorème bayésien, qui permet de trouver une solution à un problème d'optimisation global de manière plus efficace et plus efficiente qu'une méthode traditionnelle comme **GridSearch** ou **Randomized Search.**

4. L'interprétabilité globale et locale du modèle

Facilité l'interprétation des prédictions du modèle choisi est l'objectif que nous fixe notre client, car il souhaite assurer une plus grande transparence dans la décision d'accorder ou non un crédit à ses clients.
Pour répondre à cette requête, nous avons utilisé la librairie **Shap**, qui permet l'interprétation du sur deux niveaux :

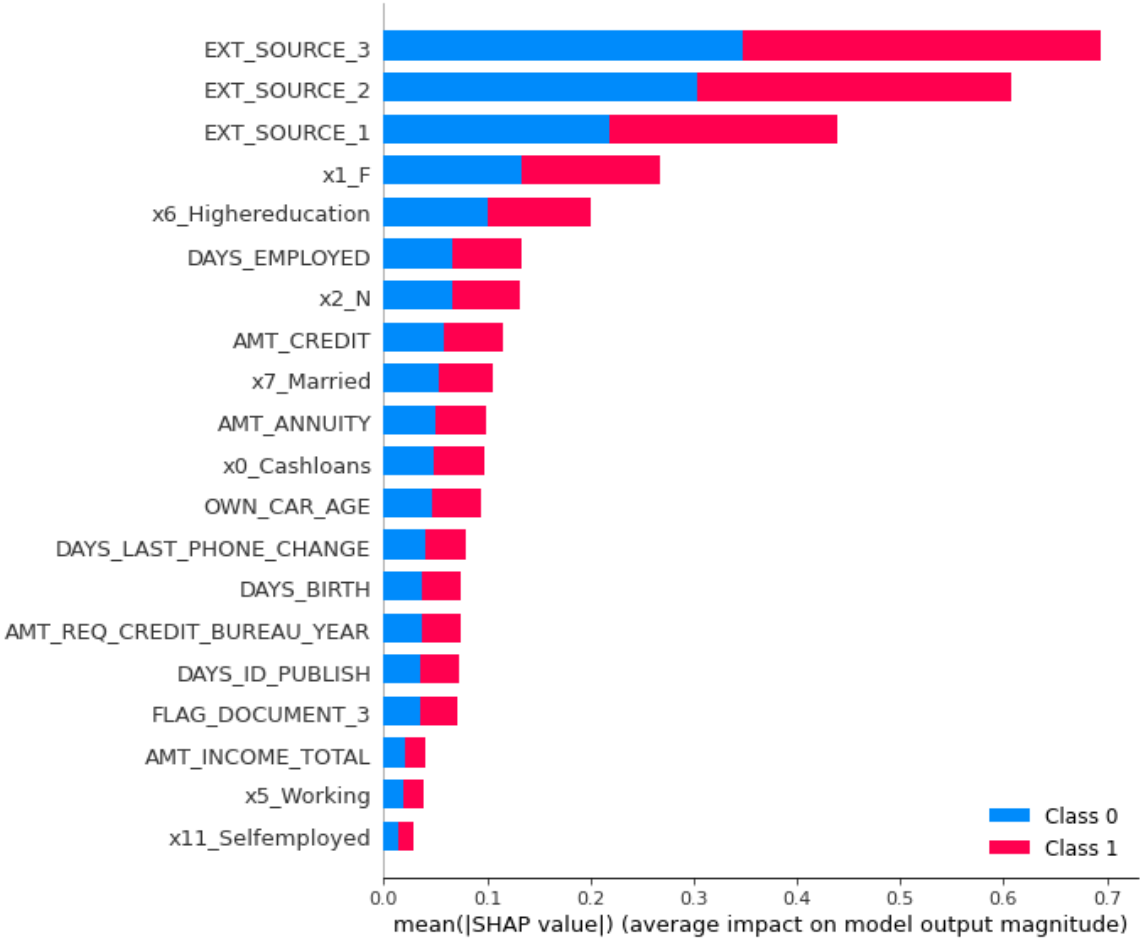
- **Interprétation locale** : comme son nom l'indique, cette méthode se concentre sur une observation / un point de données spécifique. Elle fournit une explication de la prédiction du modèle pour une observation spécifique.
  Exemple :



« Figure n°4 : Exemple d'interprétation locale »

- La prédiction du modèle : **0.88**
- La valeur de base : la moyenne de la prédiction du modèle sur l'ensemble de données d'apprentissage, en d'autres termes, c'est la valeur qui serait prédite si nous ne connaissions aucune feature pour l'instance actuelle.
- Le features en rouge ont un impact positif sur la prédiction du modèle, et les features en bleu ont un impact négatif sur la prédiction du modèle.
- Plus la largeur d'un feature est grande, plus son effet est important sur la prédiction du modèle.

- **Interprétation globale** : Le modèle est examiné dans une perspective plus large. Elle combine l'importance d'un feature avec son effet.
  Exemple :



« Figure n°5 : Exemple d'interprétation globale »

5. Les limites et les améliorations possibles

Le meilleur modèle identifié a un score AUC de 0.699, un modèle avec une meilleure capacité de distinction entre les classes positive et négatives, permettrait à notre client de proposer un meilleur service à sa clientèle et de mieux anticiper les défauts de paiement.

Deux approches pourraient être utilisés pour améliorer la performance du modèle :

- **Un meilleur Feature Engineering :** Lors de la modélisation, nous utilisons la bibliothèque **Featuretools** pour créer de nouveaux features, sauf, qu'ils n'ont apporté aucune information supplémentaire pour le modèle. La collaboration avec les équipes de notre client nous permettra d'identifier et de créer des features plus pertinentes qui pourraient potentiellement fournir plus d'informations au modèle et améliorer sa capacité de distinction entre les classes positives et négatives.

- **Utilisation d'une technique de modèles stacking** : Qui permet de combiner plusieurs apprenants faibles hétérogènes, leur apprend en parallèle et les unifie en formant un méta-modèle pour réaliser une prédiction basée sur les prédictions apprenants faibles.

Références

[1]. Megha Setia, Binary Cross Entropy aka Log Loss-The cost function used in Logistic Regression.
[2]. LightGBM documentation, Applications and Metrics
[3]. Joydwip Mohajon, Accuracy on Imbalanced Datasets and Why, You Need Confusion Matrix!
[4]. Aniruddha Bhandari, AUC-ROC Curve in Machine Learning Clearly Explained
[5]. Sion Chakrabarti, Optimize your optimizations using Optuna
[6]. Joseph Rocca, Ensemble methods: bagging, boosting and stacking