Note méthodologique

Projet 7 - Implémentez un modèle de scoring

Plan :

I. Méthodologie d'entraînement du modèle

Un projet de Machine Learning comprend les étapes suivantes :

- Prétraitement des données
- Sélection d'un modèle approprié
- Choix d'une fonction coût et d'une métrique d'évaluation
- Entrainement du modèle
- Prédiction sur des données non apprises

Pré-traitement des données

Nous nous sommes basés sur un notebook issu de Kaggle, et avons donc procédé à ces différents types de traitements :

- Fusion des dataframes, en mergeant les 9 datasets avec leur clé commune
- Détection des anomalies, en supprimant les lignes, pour lesquelles les XNA apparaissaient dans la variable CODE GENDER
- Suppression des colonnes, où nous avions plus de 70% de valeurs manquantes
- Imputation des autres valeurs manquantes, en les remplaçant par 0
- Suppression des colonnes non utiles, Ex : Index …
- Création de nouvelles variables
- Standardisation des données numériques avec MinMax
- Encodage des valeurs catégorielles avec One Hot encoder

Sélection d'un modèle de machine learning

Il existe différents modèles de machine learning plus ou moins performants et plus ou moins complexes. Pour notre projet, nous avons sélectionnés les suivants : Random Forest, XGBoost et LGBM.

Méthode d'entrainement

Séparation des colonnes features (X) de la valeur cible à prédire (y), et découpage du dataset en 2 ; un pour entrainer le modèle (train – 90%) l'autre pour le tester (test – 10%).

Chaque modèle de Machine Learning présente différents paramètres à fixer. Suivant la combinaison des paramètres choisis pour le modèle, les résultats seront plus ou moins performants.

Pour qu'un modèle de classification puisse être entrainé de manière efficace, il faut un ensemble de training (dataset d'entrainement) équilibré, c'est-à-dire comprenant un nombre d'échantillons égal pour chaque classe à prédire. Dans le cas de notre dataset, la valeur cible est distribuée de manière inégale ; 92% de 0 (clients solvables) pour 8% de 1 (clients non solvables).

Alors nous allons appliquer des méthodes pour équilibrer les classes : Smote et resampling.

La première étape a consisté à gonfler artificiellement le jeu de données pour obtenir une plus grande proportion de personnes non solvables. Grace à smote et resampling, nous allons ramener la part des prêts non solvables à 50%.

Pour déterminer quelle combinaison d'hyper paramètres donne les meilleurs résultats sur notre jeu de données, la librairie d'optimisation « hyperopt » a été utilisée.

Hyperopt va utiliser comme point d'entrée un espace de recherche (search space) que nous allons fournir pour chaque algorithme ; afin d'itérer sur des combinaisons possibles des paramètres dans cet espace pour optimiser le coût que l'on lui demandera (en l'occurrence bank cost).

Dans la méthode d'optimisation, nous allons également nous servir de folding du dataset afin de faire la cross validation des résultats obtenus. Chaque pli sera également découpé en sous ensemble de test et entrainement et la moyenne des scores des plis de test sera utilisée comme résultat obtenu.

L'hyperopt a été utilisé pour optimiser l'AUC et la fonction coût métier.

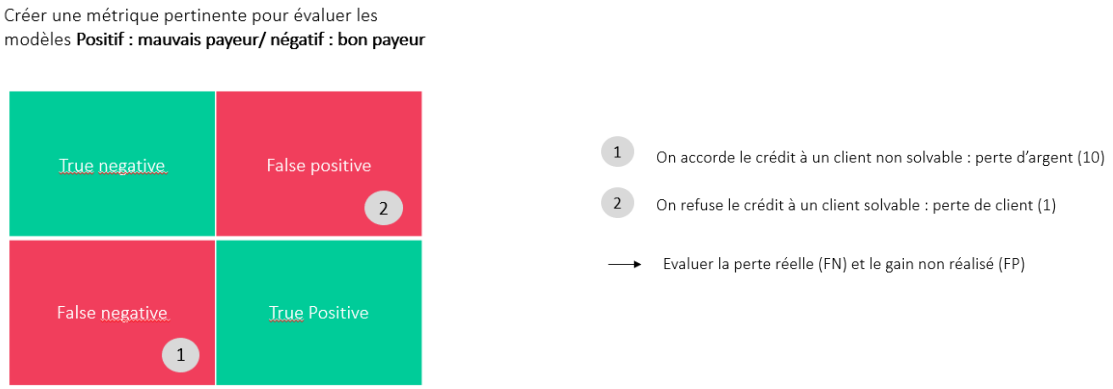
II. La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

Fonction coût : métrique bancaire

Pour ce projet, une fonction coût métier a été développée dans l'objectif de minimiser les pertes par l'octroi ou non des demandes de prêts bancaires. La fonction coût étant à minimiser pour ce problème.

La procédure est la suivante :

Il s'agit d'un problème de classification binaire, l'étiquette est soit égale à 0 (négatif, solvable) soit égale à 1 (positif, non solvable). Il existe donc 4 combinaisons possibles :



Le modèle peut donc se tromper de deux manières différentes, soit en prédisant positif alors que l'individu est négatif (False Positif) 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). Une fonction coût a donc été créée pour tenir compte de l'importance relative de chaque erreur.

fn*fn_val + fp*fp_val

TP, TN, FP et FN étant respectivement le nombre de True Postif, le nombre de True Negatif, le nombre de Faux Positif et le nombre de False Negatif.

Des coefficients ont été posés arbitrairement :

- TP_value : 0
- FN_value : 10
- TN_value : 0
- FP_value : 1

Ces valeurs de coefficients signifient que les Faux Négatifs engendrent des pertes 10 fois plus importantes que les gains des Vrai Négatifs. Ces poids ont été fixé arbitrairement et il est tout à fait envisageable de les modifier à la convenance de l'optique métier.

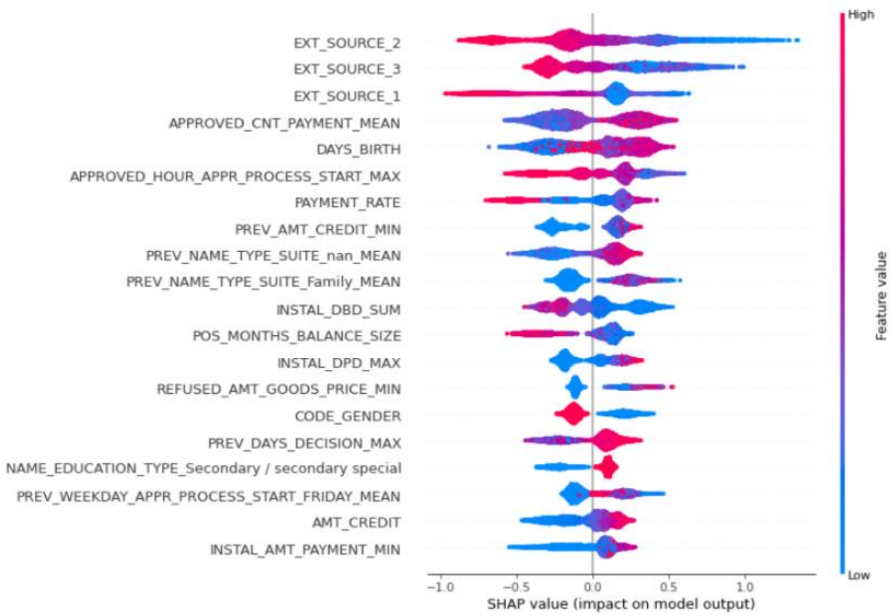
Seuil de solvabilité

Le modèle retourne un score entre 0 et 1 et par défaut il attribue la classe 1 lorsque la probabilité est supérieure à 0.5 et 0 sinon. Il a été décidé d'optimiser ce seuil qui permet de définir si un client est solvable ou non. Le seuil optimal déterminé par « hyperopt » est de 0.03 quand la probabilité de la classe non-solvable (valeur = 1) retournée par le modèle est supérieur à ce seuil optimal, le client est qualifié comme non-solvable.

III. L'interprétabilité globale et locale du modèle

Interprétabilité du modèle

Pour l'interprétabilité d'un modèle nous allons utiliser le modèle Shap. La mise en œuvre de SHAP repose sur une méthode d'estimation des valeurs de Shapley. L'idée est de moyenner l'impact qu'une variable a pour toutes les combinaisons de variables possibles.

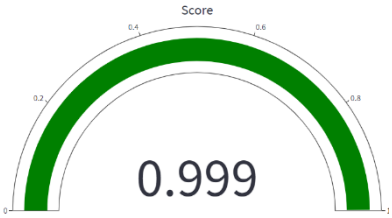


L'importance de chaque caractéristique dans la décision

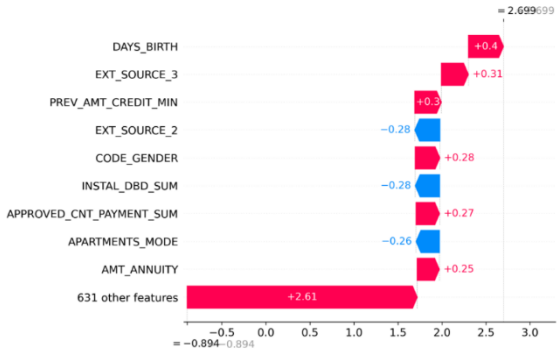
Les variables (SHAP) ayant le plus de poids dans le modèle LightGBM sont :

- Ext Source2 : Normalized score from external data source
- Ext Source3 : Normalized score from external data source
- Ext Source1 : Normalized score from external data source

Exemple pour un client donné



La solvabilité du client ' 358806' est supérieur au seuil de solvabilité de 0.97 (l'inverse probabilistique du seuil de non-solvabilité 0.03), et le client est solvable et a 99,9% de probabilité de remboursement.



La variable qui joue le plus en sa faveur est DAYS_BIRTH qui est un score normalisé par rapport à l'ensemble des données. A l'inverse les variables qui jouent le plus en sa défaveur sont EXT_SOURCE_2 et INSTAL_DBD_SUM.

IV. Les limites et les améliorations possibles

Plus de mémoire et de CPU

La recherche des hyperparamètres prenait beaucoup de temps. Plus de RAM et de CPU m'aurait permis d'entrainer plus de données et d'itérer plus rapidement.

Ajout des informations des variables

Il pourrait être intéressant de rajouter dans le dashboard, les explications des variables pour aider le chargé clientèle de comprendre et d'expliquer plus facilement les variables qui vont impacter les résultats.