**Méthodologie du modèle IA**

**de conseils à l'attribution de crédits**

**« Home Credit AI advice** **»**

**SOMMMAIRE**

**1. Conception et entraînement du modèle**

**1.1. Problématique**

**1.2. Traitement des données**

**1.3. Preprocessing**

**1.4. Modèles testés, optimisation et évaluation**

**2. Un modèle adapté aux assurances : fonction coût métier**

**3. Interprétation des résultats**

**3.1. Interprétation globale**

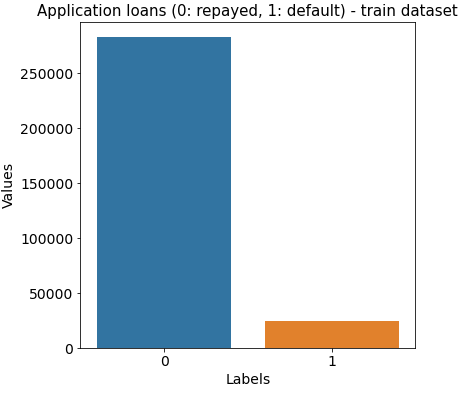
**3.2. Interprétation locale**

**3. Limites et perspectives**

**Annexe : features d'entrée de l'algorithme**

**1. Conception et entraînement du modèle**

**1.1. Problématique**



La problématique posée est de définir si le client va rembourser son prêt ou faire défaut à partir des données de précédents clients. C'est un problème supervisé de classification binaire. Les données sont fortement disproportionnées avec plus de 280000 clients ayant remboursé leur prêt et un peu moins de 25000 ayant fait défaut.

**1.2. Traitement des données**

La base de données de Home Credit comporte initialement plus de 300 000 clients et leur données personnelles récoltées lors de leur demande de prêt, leur historique de remboursement, l'historique de leur précédents crédits à Home Credit et dans d'autres établissement bancaires. Ces données sont réparties sur 7 datasets, plus 1 fichier de description des variables. Les datasets sont liés par les identifiants clients ou les identifiants des demandes de prêts antérieures. L'ensemble des datasets comporte 206 variables. Dans un premier temps, l'analyse exploratoire a permis de déterminer les variables ayant un impact significatif sur la proportion de clients remboursant ou non leur prêt. Les variables catégorielles ayant un impact ont été transformées :

- soit en variables ordinales avec une valeur plus forte lorsque le risque de défaut est plus grand,

- soit en score allant de 0 à 10 proportionnel au pourcentage de clients ayant fait défaut, un score élevé indiquant un risque de défaut plus grand,

- soit laissée sous forme de variable catégorielle pour les variables du fichier "application" qui seront ensuite transformées avec un One Hot Encoder.

La transformation en variables ordinales/numériques à ce stade est surtout nécessaire pour l'agrégation des datasets (i.e. un client peut avoir plusieurs prêts antérieurs à agréger en une seule ligne). Les datasets sont donc agrégés pour obtenir un seul fichier avec une ligne par client, tout en ajoutant des variables par feature engineering. Les colonnes créés sont de type minimum, maximum, moyenne, somme ou écart-type des variables de base.

Les valeurs manquantes sont gérées par suppression des variables avec un taux de remplissage inférieur à 80%, excepté la variable EXT\_SOURCE\_3. Cette variable conservée malgré un taux de remplissage proche de 50% car elle ressort comme une variable importante quelque soit le type de modèle utilisé. Les valeurs manquantes de la variable OCCUPATION\_TYPE (profession) sont remplacées par 'Unknown'. Puis les lignes ayant encore des valeurs manquantes sont supprimées. Cela suppose que les variables conservées sont toutes nécessaires pour la modélisation (pas d'imputation ou de remplacement des NaN).

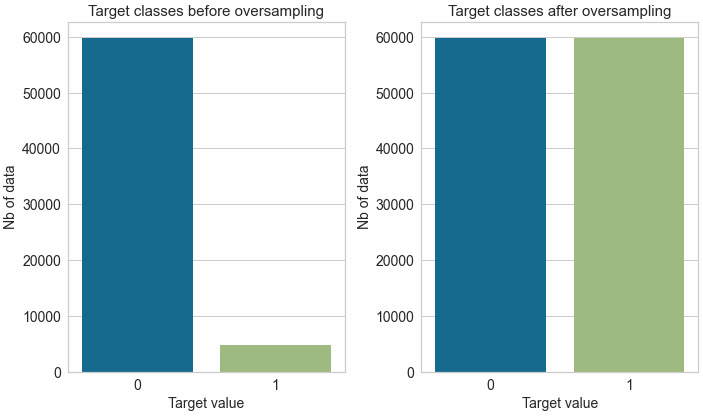
Une étude approfondie de la corrélation entre les variables (matrices de Spearman) mène à supprimer de nombreuses variables trop corrélées entre elles (supprimées si coefficient de corrélation supérieur ou égal à 0,6).

A ce stade, le dataset comporte 129 variables + l'identifiant client + la target.

**1.3. Preprocessing**

Les variables catégorielles restantes sont transformées avec un One Hot Encoder, les variables numériques sont centrées/réduites avec un Standard Scaler. Le dataset comporte alors 229 variables.

Les données sont séparées en train (80%) et test (20%).



Le train set est alors rééquilibré avec SMOTE par oversampling. On préfère cette méthode à l'undersampling ou une combinaison undersampling/oversampling car notre traitement des valeurs manquantes a considérablement réduit le dataset de base. On obtient un train set d'environ 119000 lignes et un test set d'environ 16000 lignes après preprocessing.

**1.4. Modèles testés, optimisation et évaluation**

Plusieurs types de modèles ont été testés : des modèles linéaires (régressions logistiques, SVC linéaire) et ensemblistes (Random Forest, LightGBM). Les réseaux de neurones ont été écartés en raison de leur manque d'explicabilité : les résultats de l'algorithme d'aide à la décision doivent pouvoir être justifiés de façon compréhensible pour les assureurs et les clients.

Plusieurs metrics sont calculés pour l'évaluation : accuracy, balanced accuracy, précision, rappel, F-mesure, spécificité, ROC AUC. La mesure privilégiée dans un premier temps pour classer les modèles est l'aire sous la courbe ROC (ROC AUC).

Un premier run avec les paramètres de base donne un premier aperçu des performances des modèles. Ceux-ci sont entraînés sur le train set, testés sur le test set. Les modèles linéaires donnent des résultats significativement meilleurs que les modèles ensemblistes (AUC sur le test set d'environ 0.68 contre environ 0.50).

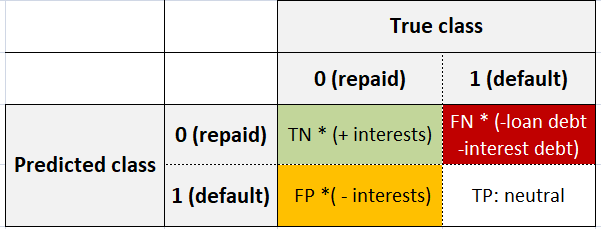
On cherche ensuite à optimiser les hyperparamètres et à tester la fiabilité des résultats avec des cross-validations. Pour la régression logistique, le solver saga ne converge jamais, et sag a des difficultés à converger. Ceci réduit les possibilités aux régularisations L1 ou L2 (elastic net impossible) et aux autres solvers (Newton-cg, LBFGS, Liblinear). Les meilleurs résultats sont obtenus avec une régularisation L1, le solver liblinear et une valeur de C = 21,5 (AUC sur test set = 0,6888). A la cross-validation, les scores sur les sets de train et validation se recoupent et sont aux alentours de 0,8.

Les autres modèles ont des résultats inférieurs, y compris les modèles ensemblistes après réglage fin des hyperparamètres.

**2. Un modèle adapté aux assurances : fonction coût métier**

Afin d'avoir un modèle sur-mesure adapté à la problématique métier, on implémente une fonction coût personnalisée. Cette fonction est utilisée sur la régression logistique pour recalculer les meilleurs hyperparamètres.

Le but est de calculer les gains ou pertes engendrés pour chaque possibilité de la matrice de confusion (vrai négatif (TN), faux négatif (FN), vrai positif (TP), faux positif (FN)) que l'on multiplie par le nombre de clients de chaque catégorie.



Il y a donc 4 possibilités :

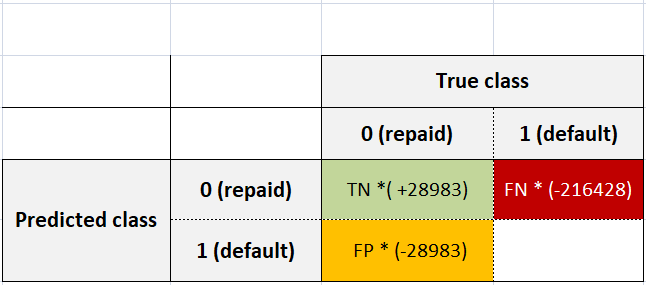
- TN (vrai négatif) : prêt correctement prédit comme remboursé, gain = montant des intérêts du prêt

- FP (faux positif) : prêt prédit comme défaut, mais aurait été en réalité remboursé, perte = montant des intérêts du prêt

- FN (faux négatif) : prêt prédit comme remboursé, en réalité le client fait défaut, perte = partie du prêt non remboursé + partie des intérêts non remboursés

- TP (vrai positif) : prêt prédit comme défaut, le client aurait effectivement fait défaut en réalité, ni gain ni perte car dans ce cas le prêt n'est pas accordé.

A partir des informations du dataset, on peut calculer que les clients faisant défaut sont en moyenne endettés de 38,6% de leur crédit, le crédit moyen est de 602648 $ et le taux d'intérêt moyen est de 4,81%. On obtient :



-

-

-

Ces valeurs sont utilisées comme

coefficients.

On cherchera donc à maximiser :

Avec TN, FP et FN le nombre de clients true negative, false positive et false negative respectivement.

**3. Interprétation des résultats**

**3.1. Interprétation globale**

**3.2. Interprétation locale**

**3. Limites et perspectives**

