NOTE METHODOLOGIQUE

|  |
| --- |
|  |
|  |

Note Méthodologique

PRÊT À DÉPENSER

Parcours Openclassrooms Data Scientist Projet 7

Catherine BRICE

3 février 2022

## Contexte

Cette note fait partie du livrable du projet « Implémentez un modèle de Scoring » du parcours Data Scientist d’Openclassrooms.

Les processus de modélisation et d’interprétabilité mis en place dans le cadre de ce projet y sont présentés.

Dans ce projet, un modèle de Scoring de la probabilité de défaut de paiement d’un client est développé pour la société « Prêt à Dépenser », une société de crédit de consommation. L’historique de prêt du client ne sera pas pris en compte bien que présent dans la base de données.

Les données de base utilisées pour ce projet sont une base de données de 307 000 clients comportant 121 features (âge, sexe, emploi, logement, revenus, informations relatives au crédit, etc.)

## Pré-traitement des données

Le pré-traitement consiste à nettoyer le jeu de données, en supprimant les valeurs aberrantes et les variables dont les valeurs manquantes sont en trop grande quantité.

Il faut ensuite numériser les variables catégorielles de manière à rendre interprétables les variables non numériques. Ce travail s’accompagne d’une étape de « Feature Engineering » qui consiste à partir des variables présentes, à trouver des combinaisons de variables plus pertinentes.

### Valeurs rares ou aberrantes pour les jeux d’entrainement et de test

Dans un premier temps, les jeux de données d’entrainement et de test ont été nettoyés.

Pour certaines variables de type catégorielles (non numériques), des valeurs apparaissent uniquement et en faible nombre dans le jeu de données d’entrainement (‘XNA’ apparait seulement 4 fois pour la variable ‘CODE GENDER’).

Pour les variables numériques, les valeurs aberrantes (DAYS\_EMPLOYED>366) ont été supprimées. Les emprunts correspondants ont donc à chaque fois été supprimés.

### Feature Engineering

Quelques variables additionnelles qui semblaient importantes ont été construites manuellement comme la proportion du temps travaillé par rapport à l’âge du client, ou le ratio crédit/revenu etc.

### Données Manquantes

Certaines variables ne possèdent pas de valeurs pour certains emprunts.

Or, tout modèle de ML nécessite d’avoir comme entrée des valeurs numériques pour chaque variable qui lui est fournie. Il existe diverses méthodes qui permettent de remplacer les valeurs manquantes mais il est nécessaire que celles-ci ne soient pas en trop grand nombre.

Ainsi, toutes les variables du jeu d’entrainement et de test, dont les données manquantes étaient supérieures à plus de 10% ont été supprimées. Pour chaque variable, les valeurs manquantes ont été remplacées par la valeur médiane de la variable en question.

### Encodage des variables catégorielles

Deux méthodes ont été utilisées :

* Pour les variables dont le nombre de valeurs différentes est égal à deux, ont été binarisées, i.e. codés par des 0 ou des 1.
* Pour les autres, une nouvelle variable a été créée pour chaque valeur différente prise par la variable initiale. Et pour chaque emprunt, la valeur prise est soit 1 ou 0 suivant la valeur de la variable initiale.

## Méthodologie d’entraînement du modèle

Le modèle entraîné dans le cadre de ce projet a été entraîné sur la base du jeu de données après analyse exploratoire et création de nouvelles features. Le notebook utilisé est consultable sur le site Kaggle.

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

* Un jeu de données exemple pour l’API (45 individus)
* Avec le jeu restant :
* un jeu de training (70% des individus) qui a été séparé en plusieurs folds pour entraîner les différents modèles et optimiser les paramètres (cross validation) sans overfitting.
* un jeu de test (30 % des individus) pour l’évaluation finale du modèle

Le problème est un problème de classification binaire avec une classe sous représentée (9 % de clients en défaut contre 91 % de clients sans défaut).

Ce déséquilibre des classes doit être pris en compte dans l’entraînement des modèles puisqu’un modèle « naif » prédisant systématiquement que les clients sont sans défaut aurait une accuracy (justesse) de 0.92 et pourrait être considéré à tort comme un modèle performant alors qu’il ne permettrait pas de détecter les clients à risque. Une approche a été testée pour rééquilibrer les deux classes :

* SMOTE qui permet de créer des données synthétiques à partir des données existantes

Le jeu de données post feature-engineering + SMOTE a été exporté et utilisé pour faire la modélisation.

Différents modèles ont été testés avec recherche d’hyperparamètres et cross validation (5 folds) :

* Régression logistique
* XGBoost
* Light GBM
* Random Forest Classifier

Le choix du meilleur modèle a été effectué en retenant le modèle avec des taux de faux négatifs et faux positifs minimisés, en prenant en compte des mesures comme le score de la métrique personnalisée qui pénalise les faux positifs, l’AUC, le score sur le jeu de validation et le temps de traitement. Le Light GBM

a été choisi. Après entrainement, ce modèle reste moyennement performant, le taux de vrais positifs reste autour de 1%.

## Fonction coût, algorithme d’optimisation et métrique d’évaluation

### Fonctions coût

Les modèles ont été entraînés dans le cadre d’une validation croisée et testés suivant différentes combinaisons d’hyperparamètres. Les fonctions de coût pour les algorithmes entraînés sont les suivantes :

|  |  |
| --- | --- |
| **Algorithme** | **Fonction de coût** |
| Régression logistique  XGBoost Classifier | Régression logistique pour classification binaire  Binary logistic |
| Random Forest Classifier | Minimisation du coefficient de GINI pour chaque noeud |
| LightGBM | Binary logistic |

### Métrique d’évaluation

Choix de la métrique

Dans le jeu de données de base, 92 % des clients sont étiquetés sans défaut de paiement (classe 0), tandis que 8 % des clients sont étiquetés en défaut (classe 1).

La matrice de confusion est la suivante :

|  |  |  |
| --- | --- | --- |
|  | **Prédit en défaut** | **Prédit sans défaut** |
| **Réel en défaut** | Vrais Positifs (VP) | Faux Négatifs (FN) |
| **Réel sans défaut** | Faux Positifs (FP) | Vrais Négatifs (VN) |

Du point de vue d’une banque, on cherchera à éviter de mal classifier un client avec un fort risque de défaut (pertes financières et frais de recouvrements qu’on imagine importants). On cherche donc à minimiser le pourcentage de faux négatifs (erreur de type II) et faux positifs (erreur de type I) et à maximiser le pourcentage de vrais positifs.

On va donc chercher à maximiser le Recall

Par ailleurs on cherche à maximiser le nombre de clients potentiels donc à ne pas tous les classer en défaut. On cherche donc à éviter d'avoir un trop grand nombre de faux positifs (erreur de type I).

On cherche aussi à maximiser la précision

Pour notre problématique métier, le RECALL est plus important que la PRECISION car on préfèrera vraisemblablement limiter un risque de perte financière plutôt qu’un risque de perte de client potentiel.

On cherche donc une fonction qui optimise les 2 critères en donnant plus d'importance au Recall.

La fonction qui permettrait de faire cela est le FβScore avec β, le coefficient d'importance relative de Recall par rapport à Precision.

Nous n’avons pas assez d’information remontée par le métier de la banque pour quantifier l'importance relative entre precision et recall (Beta)

On va donc estimer :

* Le coût moyen d'un défaut
* Le coût d'opportunité d'un client potentiel accidentellement écarté

La consigne du projet ne permet pas de répondre à ce problème, on va prendre des hypothèses « réalistes », qu’on pourrait ajuster avec les interlocuteurs métiers sur un projet réel :

* Que chaque défaut entraîne la dépense de 30 % du montant du crédit en pertes et frais de recouvrement ;
* Qu’un client a 10 % de chance de souscrire au crédit quand il en fait la demande à un conseiller, et donc que le coût d'opportunité pour un client potentiel accidentellement écarté est de 10 % du montant du crédit.

On obtient les coefficients suivants :

* Coefficient RECALL : de 30 % x le montant moyen de crédit des personnes en défaut
* Coefficient PRECISION : 10 % x le montant moyen de crédit des personnes sans défaut

On aurait un coefficient Beta qui tendrait vers 3.

Mise en place de la métrique personnalisée

Une métrique personnalisée a été mise en place pour minimiser les prédictions de faux négatifs et donner plus d’importance au Recall.

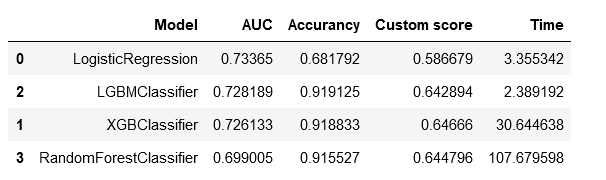
La variable ‘score custom ’ est créée pour être utilisée comme la variable du paramètre ‘scoring’ du modèle.

Cette variable est donnée par la formule suivante :

‘score custom’ = 1 – ((4.5\*FP + FN)/ (TP+TN+FP+FN))

En utilisant cette variable comme la variable du ‘scoring’, nous créons une métrique personnalisée qui pénalise les prédictions de faux positifs.

### Algorithme d’optimisation



Les algorithmes ont été entrainés dans un cadre de validation croisée et la meilleure combinaison d’hyperparamètres a été retenue pour chaque algorithme (fonction GridSearchCV de Scikit-learn).

Le modèle a été retenu en tenant compte des scores en cross validation sur le jeu de training,

les scores de l’AUC (aire sous la courbe ROC), la métrique personnalisée, et le temps de traitement compte tenu des ressources informatiques du projet : le modèle LightGBM.

D’autres mesures comme les taux de faux négatifs et faux positifs minimisés ont également été prises en compte. Les calculs de Recall et Precision ont été faits pour chaque modèle.

Le modèle LightGBM basé sur des algorithmes d’arbres de décision, présente le concept de renforcement de gradient ce qui le rend performant.

## 4. Interprétabilité du modèle

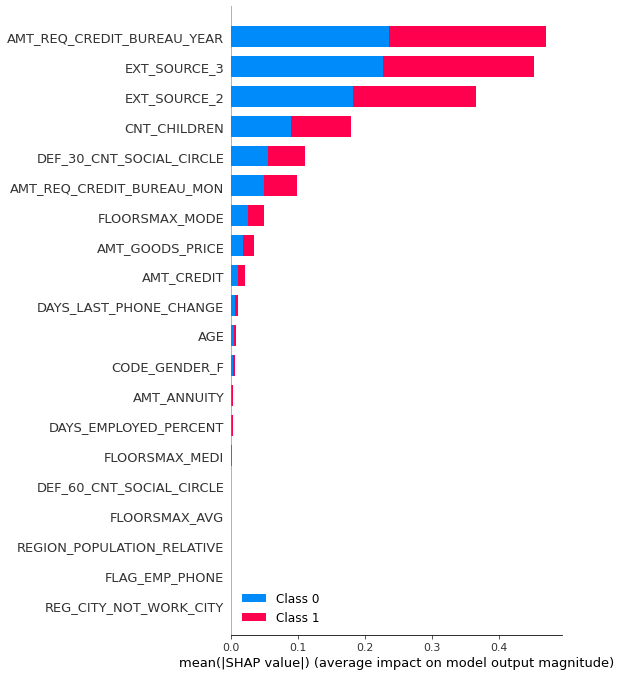
Le modèle est destiné à des équipes opérationnelles qui sont en mesure d’expliquer les décisions de l’algorithme à des clients réels. Un module d’explicabilité a été mis en place.

Pour réaliser ce module, la première perspective envisagée était d’utiliser l’importance des features issues des différents modèles utilisés mais cette approche n’est pas optimale car :

* Les features importances en sortie de modèle sont difficiles à interpréter lorsqu’il y a des variables issues de One Hot Encoding.
* L’alternative de réaliser une méthode par permutation pour déterminer l’importance des variables est lourde en termes de temps de calcul.

L’approche retenue a été d’utiliser SHAP , une librairie s’appliquant à n’importe quel modèle de machine learning . SHAP permet de comprendre comment évolue la prédiction d’un modèle en fonction des variables en entrée du modèle.

SHAP, feature importance du modèle LGBM, variables globales  :



Dans l’API réalisée, une interface comportant quelques features issues des feature importance a été mise en place afin de perturber des valeurs de variables et faire une nouvelle prédiction.

L’API donnera aussi accès aux feature importance locales avec SHAP pour un client choisi.

## 5. Limites et améliorations possibles

La métrique d’évaluation FβScore a été fixée selon des hypothèses non confirmées par le métier. L’axe principal d’amélioration serait de définir plus finement la métrique d’évaluation en collaboration avec les équipes métier.

Le but de cette modélisation était de minimiser les faux négatifs et faux positifs et maximiser les vrais positifs. La maximisation des vrais positifs n’est pas satisfaisante. Ce serait un autre axe d’amélioration possible.

La partie de traitement préalable du jeu de données a été abordée de façon superficielle en réutilisant un notebook issu de Kaggle qui se base uniquement sur une table du jeu de données. Il y a très probablement l’opportunité d’améliorer la modélisation en utilisant d’autres features des données fournies, ainsi qu’en créant de nouvelles features en collaboration avec les équipes métier.

On pourrait aussi améliorer les performances du modèle en retravaillant sur ses hyperparamètres.