Note

METHODOLOGIQUE

Project 7

« Implémentez un modèle de scoring »

Parcours Data Scientist

Dossier versioning:

Github

Sahel TAHERIAN

Février 2022

1. Contexte

Ce mémoire constitue l'un des livrables du projet « Implémentez un modèle de scoring » du parcours Data Scientist d'Openclassrooms. Il présente le processus de modélisation et d'interprétabilité du modèle mis en place dans le cadre du projet.

Le projet consiste à développer pour la société « Prêt à Dépenser », une société́ de crédit de consommation, un modèle de scoring de la probabilité de défaut de paiement d'un client avec peu ou pas d'historique de prêt.

Les données utilisées pour ce projet sont une base de données de 307 511 clients comportant 122 features (âge, sexe, emploi, logement, revenus, informations relatives au crédit, etc.).

2. Méthodologie d'entraînement du modèle

Le modèle a été entraîné sur la base du jeu de données après l'analyse exploratoire et la création des nouvelles features.

Les données relatives à différents clients sont récupérées sur Kaggle. Un kernel Kaggle a été utilisé pour faciliter l'exploration des données nécessaires à l'élaboration du modèle de scoring.

Le problème à résoudre est un problème de classification binaire avec une classe sous représentée (8 % des clients classé en modalité 1(en défaut) contre 92 % en modalité 0(sans défaut)). Ce déséquilibre des classes doit être pris en compte dans l'entraînement des modèles puisqu'il ne permettrait pas de détecter les clients à risque. Le traitement suréchantillonnage permet d'ajuster la distribution de classe de manière à avoir une répartition plus égalitaire. Dans le cadre de ce projet, l'approche SMOTE a été utilisé pour rééquilibrer les deux classes.

Pour les techniques de suréchantillonnage, **SMOTE** (Synthetic Minority Oversampling Technique) est considéré comme l'un des algorithmes d'échantillonnage de données le plus populaire et le plus influent dans le Machine Learning et l'exploration de données. Avec **SMOTE**, la classe minoritaire est sur-échantillonnée en créant des exemples « synthétiques » plutôt qu'en sur-échantillonnant avec remplacement.

```
Label 1, Before using SMOTE: 17412
Label 0, Before using SMOTE: 197845
```

```
Label 1, After using SMOTE: 197845
Label 0, After using SMOTE: 197845
```

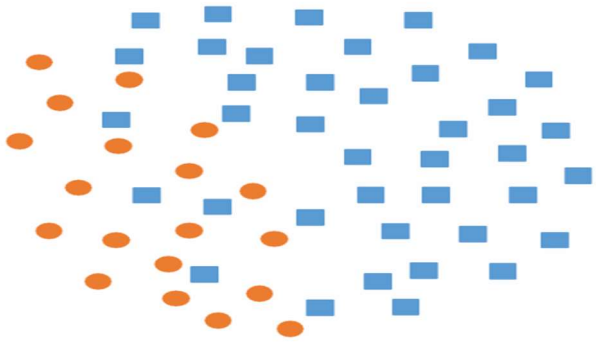

Illustration approche SMOTE (Emmanuel REMY, Conf'ISUP)

Enfin, une fonction coût a été implémentée afin de pénaliser l'impact des erreurs sur la décision d'octroi de crédit.

3. Choix de la métrique

L'entreprise lutte contre les défauts de paiement des clients. Du point de vue d'une banque, on cherchera à éviter de mal catégoriser un client avec un fort risque de défaut (pertes financières, frais de recouvrements, et …). On cherche donc à minimiser le pourcentage de faux négatifs et à maximiser le pourcentage de vrais positifs.

Le modèle ne permettra pas d'éviter totalement ce risque, à titre d'exemple une erreur de prédiction aura pour conséquence soit un défaut de paiement du client, soit un refus de crédit à un client qui pourrait rembourser sa dette sans aucune défaillance. Les erreurs de prédiction doivent donc être minimisées.

Terminologie :

FP (Faux Positifs) : les cas où la prédiction est positive, mais la valeur réelle est négative. Perte d'opportunité si le crédit client est refusé à tort, alors qu'il aurait été en mesure d'être remboursé.

FN (Faux Négatifs) : les cas où la prédiction est négative, mais la valeur réelle est positive. Perte réelle si le crédit client accepté se transforme en défaut de paiement.

TP (Vrais Positifs) : les cas d'acceptation, le crédit client sera remboursé.

TN (Vrais Négatifs) : les cas de refus, le crédit client ne pourra pas être remboursé.

Ainsi, les pertes d'un crédit en raison d'une mauvaise classification dépendent des probabilités Faux Positifs et Faux Négatifs. L'idée est d'éviter les clients avec un fort risque de défaut. Il est donc nécessaire de pénaliser les FP et FN cités précédemment. Pour réduire ce risque de perte financière, il faut maximiser deux critères Recall et Précision.

$$\text{Recall} = \frac{tp}{tp + fn}$$

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.

On cherche donc à maximiser la précision

$$\text{Precision} = \frac{tp}{tp + fp}$$

Pour notre problématique métier, **le Recall** est plus important que la **Precision** car nous préféreront vraisemblablement limiter un risque de perte financière plutôt qu'un risque de perte de client potentiel.

Nous allons donc chercher une fonction qui optimise les 2 critères en donnant plus d'importance au Recall :

$$\text{Fscore} = \frac{precision.recall}{\beta^2.precision + recall}$$

L'application de cette métrique métier passe par la quantification de l'importance relative entre Recall et Precision, à savoir Beta (β). Cela revient à estimer le coût moyen d'un défaut, et le coût d'opportunité d'un client refusé par erreur. Cette connaissance métier n'est pas évoquée à ce stade du projet, nous allons donc l'estimer. Cette hypothèse pourra bien entendu être modifiée avec un interlocuteur métier.

- Défaut de paiement 30% du montant du crédit en pertes et autres recouvrements.
- 10% de chance d'obtenir un crédit pour un client lambda qui souhaite emprunter.

$$\text{Beta} = \frac{coef Recall}{coef Precision}$$

L'hypothèse fixée dans le projet est Beta = 3

3.3. Algorithme d'optimisation

Le projet a été traité à l'aide d'un modèle Baseline (Logistic Régression), et 3 algorithmes (XGBoost, LightGBM, et Random Forest)

La meilleure combinaison d'hyperparamètres a été retenue pour chaque algorithme. Le modèle ayant le meilleur score en cross validation sur le jeu de training a été retenu. Il s'agit du modèle LightGBM. Ce modèle dispose d'une bonne performance en composant l'AUC, l'accuracy, Score custom et le temps de calcul.

Model	AUC	Accurancy	Score custom	Time
LGBMClassifier	0.741376	0.919581	0.602977	43.0968
LogisticRegression	0.73633	0.692729	0.579606	22.4765
RandomForestClassifier	0.704171	0.91789	0.602196	273.956
XGBClassifier	0.703975	0.917185	0.603659	236.391

La **mise en œuvre** de LightGBM est facile, la seule chose compliquée est le réglage des Hyperparamètres. LightGBM couvre plus de 100 hyperparamètres. Dans le contexte du projet l'idée est de pouvoir optimiser quelques hyperparamètres via Bayésien Optimisation algorithme et les mettre dans notre Grid Search de LightGBM.

4.Interprétabilité du modèle

La réponse au besoin d'interprétabilité est prépondérante. Le contexte de prédiction n'est pas uniquement appliqué à des experts de la data science mais au contraire à des experts du crédit. Un chargé de clientèle doit pouvoir utiliser le modèle via l'application mise à disposition, en face à face avec son client, dans le but de lui expliquer le plus simplement possible la décision envisagée dans l'étude de son dossier.
L'idée est donc d'expliciter au mieux le score renvoyé par le modèle. Pour réaliser ce module, la première perspective envisagée était d'utiliser l'importance des features issues des différents modèles utilisés. Puis un LIME qui est une librairie s'appliquant à n'importe quel modèle de machine Learning et permettant de comprendre comment évolue la prédiction d'un modèle et perturbant les variables en entrée du modèle. Ensuite un SHAP, pour une meilleure visualisation.

5. Limites et améliorations possibles

- La modélisation effectuée dans le cadre du projet a été effectuée sur la base d'une hypothèse forte : la définition d'une métrique d'évaluation : le F-Beta Score avec Beta fixé suivant certaines 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.

- L'interprétabilité du modèle pourrait être étoffée en considérant les variables issues du one hot encoding comme une seule et même variable dans la perturbation (un client ne pouvant cumuler plusieurs caractéristiques dans la logique du jeu de données initial.

- Par ailleurs, la partie du 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.