Implémentation d’un modèle de scoring

1. Problématique et données

1.1. Problématique

L'entreprise financière "Prêt à dépenser" propose des crédits à la consommation aux personnes qui ont peu ou pas d'historique de prêt.

Cette dernière veut mettre en place un algorithme de classification à partir de données comportementales ou provenant d'autres institutions financières. Il doit alors lui permettre de calculer la probabilité qu'un client rembourse son crédit et de déterminer si sa demande de crédit est à accepter ou à refuser.

De plus, elle souhaite développer un dashboard interactif à destination de ses chargés de relation client. Avec cet outil, ils auront pour mission d'expliquer aux clients les décisions d'octroi de crédit et de leur rendre disponible leurs informations personnelles.

1.2. Données

Les données disponibles pour ce projet sont variées, nous avons celles provenant directement de l'entreprise :

- application_{train|test}.csv
- previous_application.csv

Celles provenant d'autres institutions financières :

- bureau.csv
- bureau_balance.csv

Et enfin les données comportementales :

- bureau_balance.csv
- credit_card_balance.csv
- installments_payments.csv
- POS_CASH_balance.csv

À noter que dans le cadre de ce projet, seules les demandes de crédit faites chez "Prêt à dépenser" ont été utilisées. Les données de application_train.csv ont servi pour construire le modèle alors que celles de application_test.csv ont été utilisées par l'API et le dashboard.

En vue de la modélisation, les données ont été prétraitées comme suit : gestion des données manquantes, suppression des outliers, retravail de certaines variables, création de nouvelles variables (ou features engineering) et sélection des variables pour l'entraînement du modèle.

2. Méthodologie d'entraînement du modèle

2.1. Préparation des données

Suite au travail effectué sur les données en amont, nous avons pour cette étape un dataset sans donnée manquante de 292813 lignes et 18 colonnes. Cependant les temps de traitement de l'entraînement étaient trop longs sur cette volumétrie, pour cette raison, un échantillon représentatif de la variable cible (*TARGET*) a été tiré à hauteur de 25% du dataset (soit 73203 lignes).

Cet échantillon a donc été utilisé pour entraîner le modèle, et est passé par les dernières étapes classiques de préparation de données : séparation de l'échantillon en train et test sets, encodage des variables qualitatives et standardisation des variables à partir du train set.

Enfin, étant donné que la variable à expliquer (*TARGET*) est déséquilibrée avec 92% de 0 et 8% de 1, l'échantillon est alors rééquilibré pour éviter l'overfitting avec SMOTE.

2.2. Modèles choisis pour l'entraînement

Les modèles de classification supervisés répondent à la problématique rencontrée et parmi eux ont été choisis : la régression logistique, le random forest et le lightGBM qui viendront challenger la baseline.

2.3. Travail préliminaire avant l'entraînement

Pour le random forest, je recherche le nombre d'arbres qui maximise l'accuracy avant de l'entraîner. Ce sera le seul hyperparamètre qui est fixé d'avance parmi les 3 choisis.

2.4. L'entraînement

Il se déroule en deux étapes distinctes : d'abord la sélection de chacun des meilleurs modèles selon leurs hyperparamètres respectifs (voir ci-dessus) avec le gridsearch et puis parmi ces meilleurs modèles, on sélectionne le meilleur d'entre eux selon l'accuracy et le temps de traitement avec la validation croisée.

<u>Liste des hyperparamètres des modèles :</u>

Régression logistique :

- C = [0.001, 0.005, 0.02, 0.1, 0.46, 2.15, 10.0, 46.4, 215.4, 1000.0]
- penalty = ['l2', 'none']

Random forest :

- n_estimators = 300
- max_features = [10, 15, 20, 30, 40, 48]
- max_depth = [10, 15, 20, 25]

LightGBM :

- n_estimators = [300, 400, 500, 600, 700]
- learning_rates = [0.025, 0.05, 0.1]
- max_depth = [1, 2, 3, 4, 5]

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

3.1. Fonction coût métier

L'approche faite pour comparer les modèles est d'utiliser l'accuracy et le temps de traitement puisqu'on les compare sur leur capacité à bien déterminer si un client rembourse son crédit ou non et dans le cas où plusieurs modèles ont des accuracies semblables, le temps de traitement nous permet de choisir le plus rapide d'entre eux.

3.2. Algorithme d'optimisation

Les résultats obtenus entre les modèles sont les suivants :

	accuracy	temps
baseline	0.49998	0.018088
regression_logistique	0.623936	0.639016
random_forest	0.937141	131.817862
light_GBM	0.948468	2.563505

Le lightGBM a un accuracy quasi-équivalent au random forest mais le surclasse en terme de temps de traitement. Pour ces raisons, c'est ce modèle qui a été sélectionné.

3.3. Métriques d'évaluation

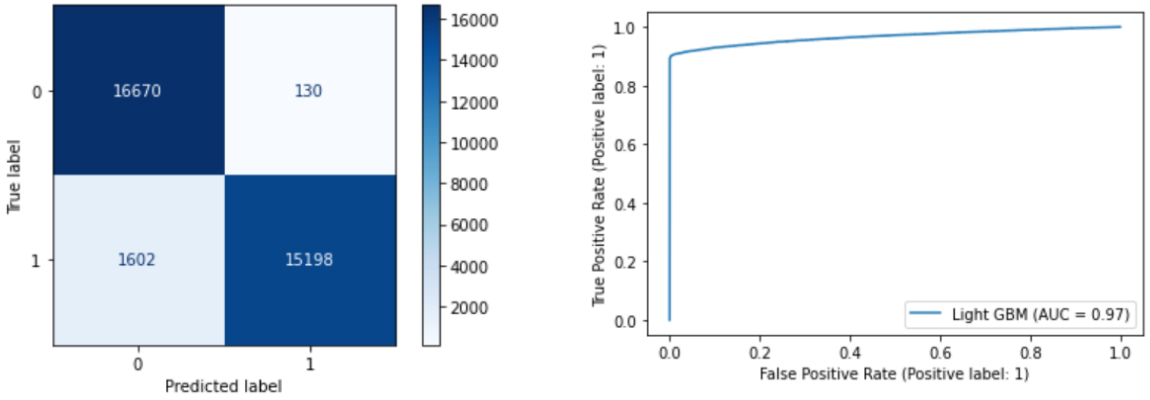
Au vu de la problématique métier, cela risque de coûter plus d'argent à l'entreprise d'accepter un crédit à un client qui ne le remboursera pas. Il vaut donc mieux refuser un crédit qui a des chances d'être remboursé que le contraire. Cela signifie donc qu'il vaut mieux privilégier le recall par rapport à la précision, qui peut être mesuré avec le fβ-score.

En effet, cette mesure est la moyenne harmonique de la précision et du recall dont la formule est la suivante : $(1+\beta^2) \times (\text{precision} * \text{recall}) / [(\beta^2 * \text{precision}) + \text{recall}]$. Afin de donner plus d'importance au recall par rapport à la précision, la valeur de β doit être alors supérieure à 1.

C'est cette approche qui a été utilisée entre autres pour évaluer le meilleur modèle choisi en fixant β à 2. En plus de cette métrique, le lightGBM a été évalué avec la matrice de confusion ainsi que la courbe ROC et AUC.

Nous obtenons les résultats suivants :

```
Résultats sur le jeu de test :
- accuracy = 0.9484523809523809
- precision = 0.9915187891440501
- recall = 0.9046428571428572
- fbeta score = 0.920778402481582
```

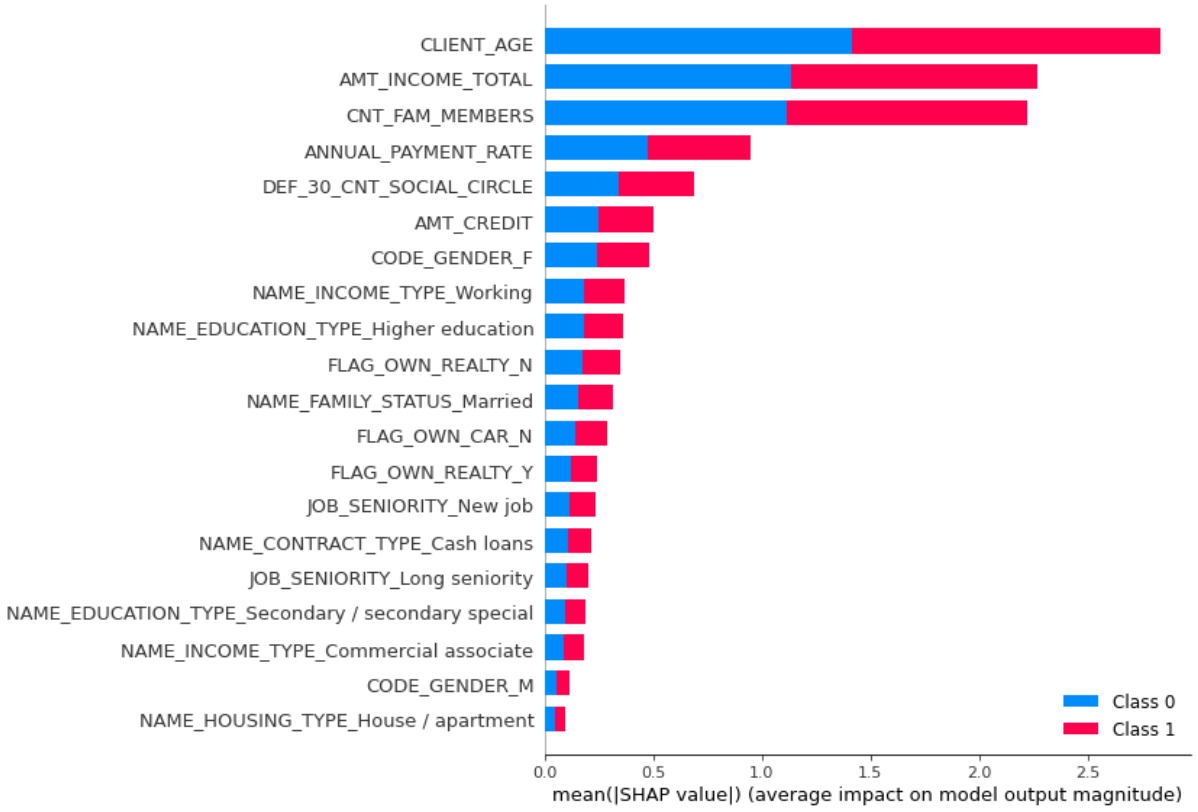


Avec une accuracy à 0,95, un fβ-score à 0,92 et un AUC à 0,97, le modèle fait de très bonnes prédictions que ce soit parmi les clients qui rembourseraient leur crédit ou parmi ceux qui ne le rembourseraient pas.

4. L'interprétabilité globale et locale du modèle

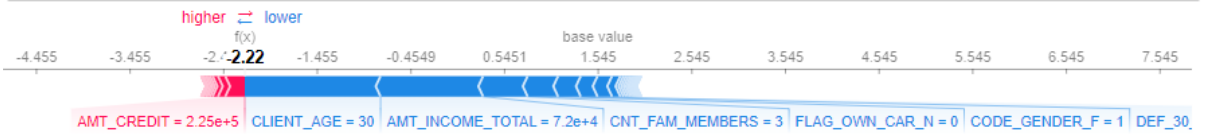
4.1. Interprétabilité globale

Les features qui ont contribué le plus en moyenne à l'élaboration du modèle sont l'âge du client en premier lieu, puis son revenu total et le nombre de membres de sa famille et enfin le taux de remboursement annuel ainsi que le nombre de personnes dans le cercle social du client ayant fait défaut dans les 30 jours.



4.2. Interprétabilité locale

Pour le client avec l'index *50005*, le montant de son crédit est censé pousser la prédiction du modèle à la hausse alors que son âge, le montant de son revenu annuel, le nombre de membres de sa famille, ainsi que le fait que ce soit une femme sans véhicule pousse la prédiction à la baisse.



5. Les limites et améliorations possibles

5.1. Limites

Etant donné les excellents résultats affichés sur le jeu de test que ce soit sur :

- l'accuracy
- le recall
- la précision et par conséquent le fβ-score
- ainsi que l'AUC

Le modèle paraît trop bon et semble overfitter sur les données de test. Aussi, les résultats affichés dans le dashboard ont l'air de confirmer cette hypothèse car sur l'ensemble (entre 20 et 40) des clients testés, aucun n'a vu sa demande de crédit acceptée.

Aussi, les résultats de l'interprétabilité locale sont difficiles à interpréter pour plusieurs raisons. La première est que nous avons deux valeurs de Shapley calculées par client. Et la seconde est que la prédiction ainsi que la valeur de base affichées ne correspondent pas à ce qui est effectivement prédit par le modèle ou présent dans le dataset.

5.2. Améliorations possibles

Afin d'améliorer le modèle, il convient d'analyser la cause de cet overfitting, la première piste à étudier serait une probable fuite de données venant du rééquilibrage effectué avec SMOTE.

Ensuite, pour affiner l'interprétabilité locale du modèle, le travail sur le calcul des valeurs de Shapley ainsi que sur l'affichage de la prédiction et de la valeur de base doit être approfondi.

Annexes

Annexe I : liens repository Github, API & dashboard

Repository Github :

https://github.com/sebastienbourgeois/oc-p7-implementer-modele-scoring

API : https://oc-api-modele-scoring.herokuapp.com/

Dashboard : https://oc-dashboard-scoring.herokuapp.com/

Annexe II : architecture