Implémentez un modèle de scoring - Note méthodologique

Sommaire :

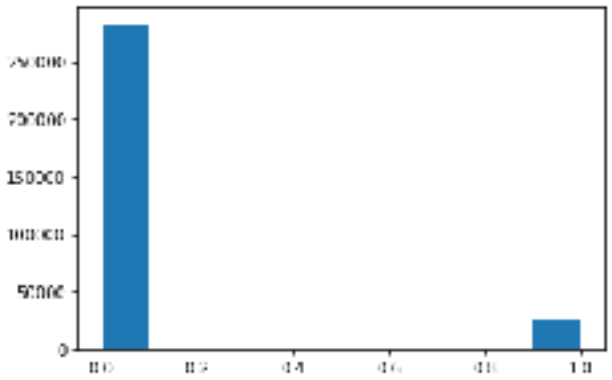
1. La méthodologie d'entraînement du modèle

Pour cet exercice nous disposons d'une base de données très complètes et variées. Au total 7 fichiers comportant comme point commun l'identifiant d'un client, constituent l'ensemble des informations détenues par notre organisme. Parmi ces données certaines sont numériques, continues ou discrètes, ou des chaines de caractères. On peut observer le revenu du client ou le type d'habitation dans lequel il loge par exemple.

Nous cherchons à pouvoir définir à partir d'un algorithme de machin learning supervisé si un client est dans la capacité de lui voir accordé un prêt ou non. Autrement dit, l'algorithme apprendra de données déjà existantes quels clients sont en capacité de contracter un prêt. Mais ce type d'algorithme n'est capable de comprendre que des données numériques, alors que nos données textuelles sont tout autant primordiales. Nous allons alors devoir transformer ces chaines de caractères en valeurs numériques. Il ne s'agit pas de créer une hiérarchie entre chaque élément d'une même variable mais de leur attribuer une simple valeur numérique pour qu'elles soient comprises par l'algorithme. Par exemple :

Homme/Femme -> 0/1
Propriétaire/Locataire/Colocataire -> 0/1/2
Célibataire/Marié/Divorcé/Veuf -> 0/1/2/3

Dans un second temps, pour que l'algorithme apprenne de la manière la plus fiable possible, les données doivent être équilibrées. Cela signifie que la quantité de clients aptes et non-aptes doit être équilibrée. Hors lors de l'exercice nous constaterons un grand déséquilibre : 90% des clients ne sont pas aptes à se voir accorder un prêt :



Quantité d'apparition des valeurs de classe chez les clients

Nous allons alors recréer artificiellement des clients avec des caractéristiques semblables aux clients éligibles pour rétablir l'équilibre entre les classes et ainsi obtenir un résultat le plus juste possible.

Maintenant que les données sont préparées à être utilisées, nous allons chercher à paramétrer l'algorithme. Nous utiliserons le LGBM, il s'agit d'un algorithme utilisé pour de la classification qui utilise des arbres de décision. L'arbre de décision est un algorithme itératif qui, à chaque itération, va séparer les individus en k groupes (généralement k=2 et on parle d'arbre binaire) pour expliquer la variable cible. La première division (on parle aussi de split) est obtenue en choisissant la variable explicative qui permet la meilleure séparation des individus. Cette division donne des sous-populations correspondant au premier nœud de l'arbre. Le processus de split est ensuite répété plusieurs fois pour chaque sous-population (noeuds précédemment calculés) jusqu'à ce que le processus de séparation s'arrête.

Nous allons alors entrainer l'algorithme avec divers paramètres : nombre d'itérations maximum, la valeur de k, la sévérité des séparations. Pour chaque paramètres nous allons essayer une série de valeurs de paramètres et choisir au final la combinaison de paramètres qui aura donné les meilleurs résultats. Cela s'appelle de l'Hyperparamètre searching.

Une fois ces meilleurs paramètres trouvés, nous entrainerons le LGBM pour pouvoir l'utiliser sur nos données.

2. La fonction coût métier et la métrique d'évaluation (1 page max)

La fonction coût métier :

Le but de la fonction coût métier est d'évaluer la fiabilité de notre modèle d'un point de vue opérationnel : pour le métier dans lequel l'outil sera déployé, est-ce que les résultats sont satisfaisants ? Est-ce que les mauvaises prédictions représentent une menace trop importante à la pérennité de l'entreprise ? Il existe 4 cas de figure possible en sortie de l'algorithme :

- Prêt accordé justement
- Prêt refusé justement
- Prêt accordé alors qu'il ne devrais pas
- Prêt refusé alors que le client est éligible.

Ici c'est le troisième scénario qui est le plus dangereux et qui pourrait créer le plus de perte. Le quatrième est également à éviter puisqu'il constitue un manque à gagner. Nous cherchons alors à évaluer la quantité de faux négatifs et faux positifs présents lors de la prédiction grâce à une matrice de confusion :

```
Confusion matrix

 [[20176  6144]
 [11699 38858]]

True Positives(TP) =  20176

True Negatives(TN) =  38858

False Positives(FP) =  6144

False Negatives(FN) =  11699
```

Sur un ensemble de 76 877 testes l'algorithme calcule 6144 faux positifs, donc prêts accordés alors qu'ils ne devraient pas, soit environ 8% des résultats totaux. On considère l'algorithme acceptable avec moins de 10% de faux positifs, donc ces résultats sont recevables.

Métrique d'évaluation :

Pour mesurer l'efficacité du modèle on mesure L'AUC (Area Under the Curve). L'AUC est un indicateur de la capacité prédictive d'un modèle. Plus elle est élevée pour le modèle est capable d'être discriminant entre les deux sorties possibles. C'est en quelque sorte un indice de fermeté du résultat. On utilise également le F-Beta Score qui est une mesure de fiabilité des résultats sortis : il permet d'estimer la quantité d'erreurs par prédiction correcte.

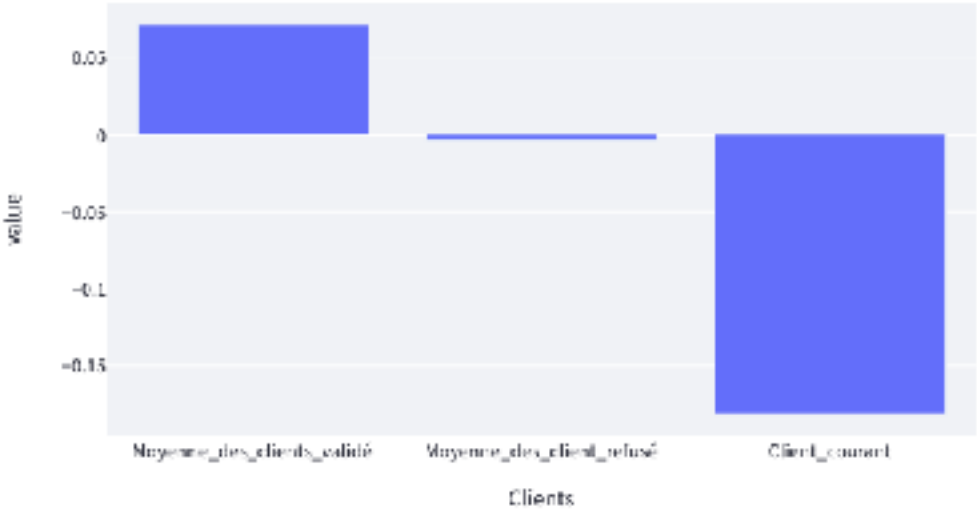
Ici les résultats sont : AUC = 0.79 et F-Beta Score = 0.86

3. L'interprétabilité globale et locale du modèle (1 page max)

Pour mesurer l'interprétabilité du modèle on va utiliser plusieurs mesures qui ont l'avantage d'être visuels, et qui sont donc au moins partiellement compréhensible même pour les moins mathématiciens d'entre nous. Ce sont des mesures comparatives qui permettent d'évaluer la place d'un client par rapport aux autres.

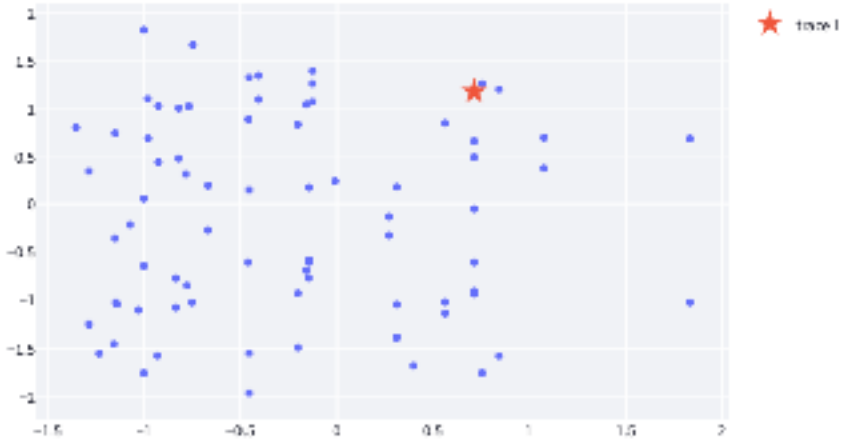
La comparaison entre les classes :

Nous pourrons comparer chaque variable d'un client sélectionné avec les deux classes : Vrai positif et Vrai négatif. Cela permet d'observer à quelle classe ressemble le plus un client sur une variable donnée.



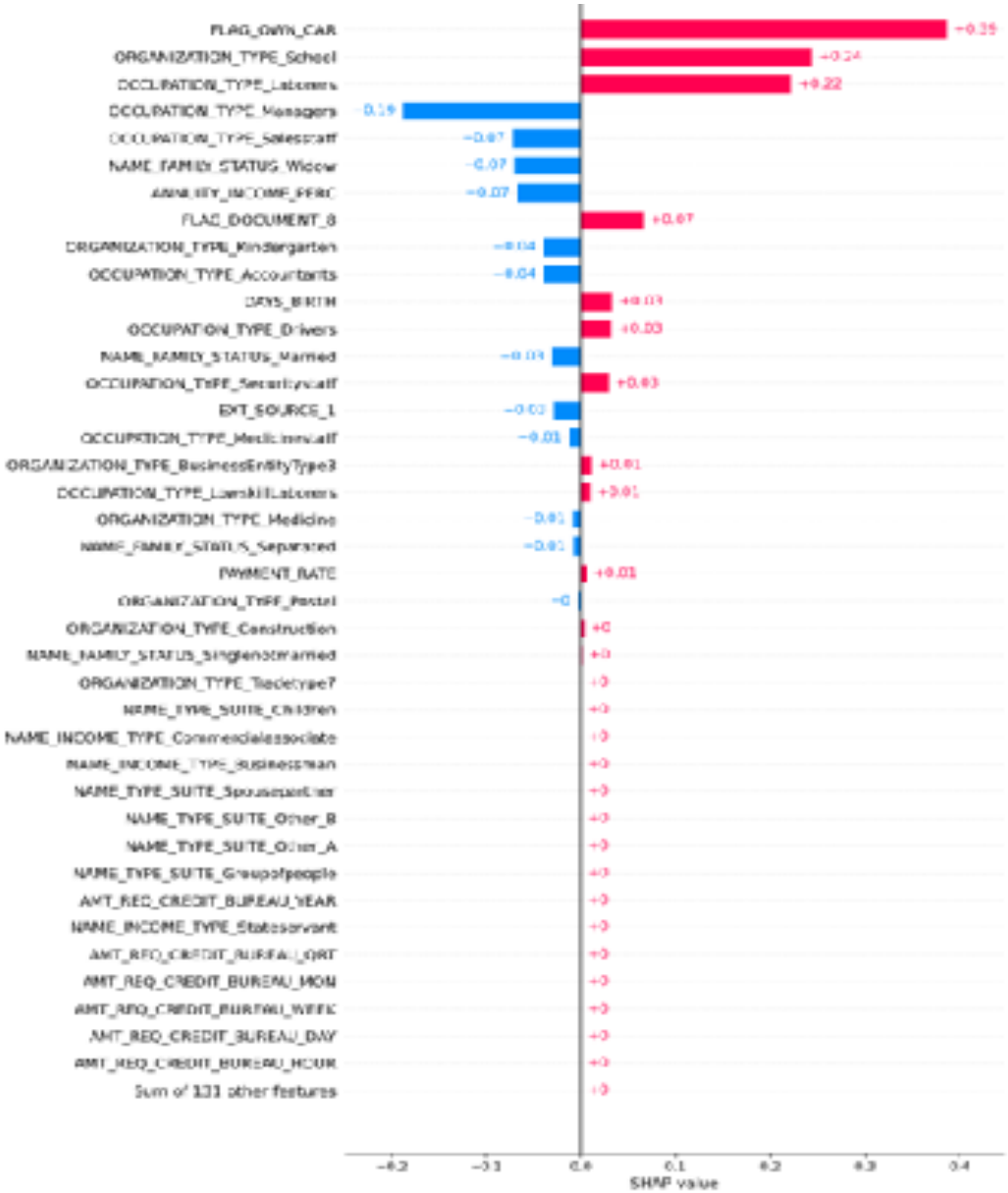
Une analyse bivariée :

Sur le même principe que l'analyse précédente, ici nous pourrons comparer la place du client sélectionné par rapport aux autres, et ainsi estimer sa position par rapport aux autres, mais cette fois-ci sur un scénario composé de deux variables :



La feature importance :

La feature importance représente le poids d'un feature dans le calcule de prédiction. Cela permet d'indiquer quelles variables sont les plus à même de modifier un score final voire une prédiction. La feature importance peut être mesurée de manière globale (sur l'ensemble des clients) ou au cas par cas pour chaque client. Exemple de feature importance locale :



Dans ce cas précis, on remarque que les features « FLAG_OWN_CAR », »ORGANIZATION_TYPE_School » et « OCCUPATION_TYPE_Laborers » sont les plus importantes. Autrement dit ce sont celles qui ont le plus d'impacte sur son score de prédiction.

4. Les limites et les améliorations possibles (1 page max)

Les limitations :

Pour cet exercice j'ai utilisé mon ordinateur personnel. Bien qu'il soit performant, il reste négligeable à cote d'infrastructures professionnelles. Par conséquent j'ai été contraint de limiter les calculs par exemple en limitant le nombre de clients lors de l'entraînement de l'algorithme. Dans le même sens, le LGBM est une version « Light » du Gradiant Boosting, et est par définition moins précis. Une puissance machine bien supérieur permettrait :

- De sélectionner un algorithme plus performant en terme de résultats.
- Améliorer la qualité de ma base de données d'entrainement et par conséquent également augmenter le résultat.

De plus, l'application finale est aujourd'hui déployée en local, donc pas encore accessible universellement sur internet.

Les améliorations :

Il serait intéressant dans un premier temps d'améliorer l'accessibilité de l'application en lui rendant accès depuis internet, en l'associant à un domaine. Ainsi la solution pourrait être déployée à plus grande échelle.

En suite du point de vue de l'utilisation il y a encore des améliorations à apporter d'un point de vue métier :

- Rendre le libellé des variables plus explicite, et les trier par catégories. Ainsi nous pouvons imaginer des filtres qui reproduiraient des scénarios. Par exemple un filtre « Situation professionnelle » qui sélectionnerait toutes les variables liées à la situation professionnelle du client.

- Les variables méritent également un second tri mais cette fois-ci accompagné d'un spécialiste du métier, pour alléger le modèle de potentielles variables parasites.

- Une nouvelle fonction de simulation permettant d'observer les nouveaux résultats d'un clients après modification artificielle d'une des variables.

Liens utiles :

API : https://dry-peak-32185.herokuapp.com/

Application Web : https://rocky-springs-46510.herokuapp.com/

L'application Web dépend du fonctionnement du serveur de l'API. Lancez dans un premier temps l'API jusque'à ce que ce message suivant apparaisse :

```
{"detail":"Not Found »}
```

Ensuite patientez encore une quinzaine de secondes et lancez l'application Web. Le chargement est long à cause de la taille de l'application.

Git interface web : https://github.com/NicolasBoya/projet_7_OC_streamlit.git

Git API : https://github.com/NicolasBoya/projet_7_OC.git

Git NoteBook : https://github.com/NicolasBoya/projet_7_notebook.git