Modèle de scoring de crédit

August 16, 2022

Abstract

L'objectif de cette note technique est d'expliquer notre algorithme de scoring qui modélise le risque d'un individu à être en défaut sur un crédit . Notre modèle se base sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.) qui sont ensuite utilisées pour construire un modèle logistique. Nous obtenons un recall et un AUC proche de 0.8.

Démarche de modélisation

Notre objectif est d'utiliser un maximum d'information sur les individus. Ainsi nous synthétisons la base de données qui est relationnelle en un fichier unique. Ce fichier unique a été créée via un kernel kaggle déjà existant avec des features métier reconnues comme étant intéressante pour prédire la capacité de défaut de crédit. Des traitements exclusifs ont été ajoutés au kernel. Par exemple, les colonnes avec plus de 30% de valeurs manquantes ont été supprimées et les valeurs aberrantes/manquantes (infini) ont été remplacée par la valeur la plus fréquente dans la colonne.Ci-dessous la base de données initiale.

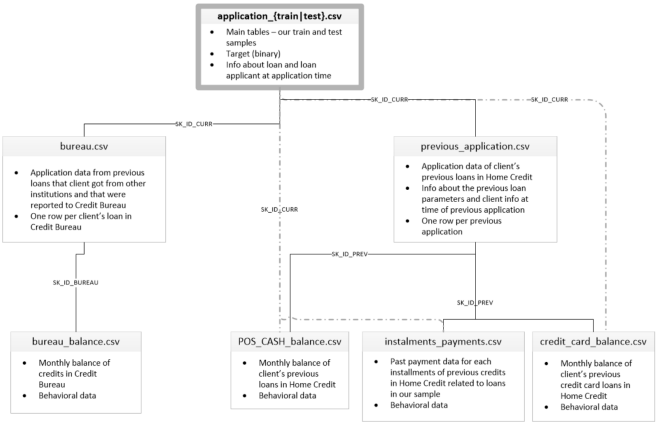

Figure 1: BDD de l'ancienne application

La pipeline de modélisation retenue se divise en 3 parties. Dans un premier temps, nous standardisons nos données. Ensuite nous rééquilibrons notre dataset avec SMOTE. Enfin, nous terminons par une régression logistique. Ci-dessous notre schéma de pipeline.

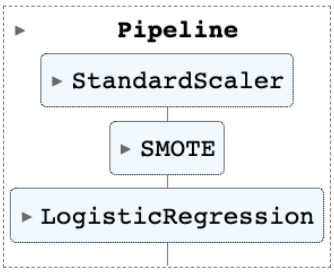

Figure 2: Oversampling

Etape 1 :
La standardisation ou normalisation standard consiste à soustraire pour toutes les valeurs numériques la moyenne de sa colonne et à la diviser par l'écart type. Chaque valeur reflète la distance par rapport à l'écart type. L'application d'une « commune mesure » permet d'améliorer la qualité de notre prediction.

Etape 2 :
La répartition de notre target (le défaut de crédit) est déséquilibrée dans notre dataset. En effet,

moins de 10% de notre population a un risque de défaut. Notre idée est d'uliser l'oversampling avec SMOTE pour pallier à cette faiblesse. L'oversampling consiste à augmenter le nombre d'individus minoritaires afin qu'ils aient moins d'importance à l'étape de la modélisation. SMOTE ou Synthetic Minority Oversampling TEchnique est un algorithme qui suréchantillonne ces observations minoritaires. Le principe est de générér des observations synthétiques qui ressemblent aux individus minoritaires sans être strictement identiques. Le résultat sera d'avoir une distribution de la target plus homogène. Ci-dessous, un schéma illustrant l'oversampling.

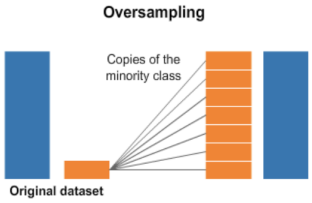

Figure 3: Oversampling

Etape 3 :
Un modèle de régression logistique est un modèle qui permet de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Nous avons choisi la régression logistique régularisée car c'est le modèle qui nous a assuré une meilleure performance par rapport à une régression logistique simple ou un randomforest. La régression logistique correspond à :

$$S(X^{(i)}) = \theta_0 + x_1\theta_1 + x_2\theta_2 + ... + x_n\theta_n \tag{1}$$

- $X^{(i)}$ est une observation modélisé sous la forme d'un vercteur x_1, x_2,...,x_n
- x_i : est une feature d'un individu
- θ_i : est un poids/paramètre de la régression logistique.
- θ_0 est le biais

En conclustion,cette pipeline en 3 étapes représente notre modèle. Les performances sont plutôt correctes. Nous avons un recall et un AUC proche de 0.8. Le temps d'entrainement du modèle est raisonnable car il est inférieur à 1h.

Optimisation et évaluation du modèle

Un client qui ne va pas rembourser son crédit revient environ dix fois plus cher qu'un client catégorisé à risque alors qu'il ne l'est pas. De ce fait, le choix d'optimiser notre modèle avec l'AUC et le recall à la place de l'accuracy semble plus stratégique.Ainsi,nous avons choisi d'optimiser nos hyperparamètres en fonction d'un score F-Beta avec β égale 10 et un seuil de décision de probabilité égale à 0.4. Ci dessous, notes stratégie d'optimisation implémentée en python.

```
# Tuning avec fbeta logreg
ftwo_scorer = make_scorer(fbeta_score, beta=10)

start = time.time()
model = Pipeline([
        ('scaler', StandardScaler()),
        ('sampling', SMOTE()),
        ('classification', LogisticRegression())
    ])

params = { 'classification__C':[1,5,10,16], 'classification__max_iter':[100,300,400]}


clf = GridSearchCV(model,params, cv=3, scoring =ftwo_scorer)

clf.fit(X_train, y_train)
end = time.time()
temps_logreg_ajuste= end - start
```

Figure 4: Algorithme d'optimisation et fonction coût

Une fonction de coût est une fonction qui représente la différence entre ce que notre modèle prédit et la réalité. On souhaite minimiser cette fonction pour obtenir les meilleures performances. Le F1-score est une métrique pour évaluer la performance des modèles de classification à 2 classes ou plus car il permet de résumer les valeurs de la precision et du recall en une seule métrique. On peut le repondérer avec un paramétre β qui nous permet de favoriser le recall. Ci- dessous les formules équivalentes:

$$\text{F1-score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$\text{F}_\beta\text{-score} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Donc :

- Si $\beta \geq 1$, on accorde plus d'importance au recall (autrement dit aux faux négatifs).
- Si $\beta \leq 1$, on accorde plus d'importance à la precision (autrement dit aux faux positifs).

Notre β est égale à 10 pour favoriser le recall.

Ensuite, nous avons utiliser cette fonction de coût personnalisé pour tuner nos hyperparamètres. Nous avons choisi d'optimiser deux paramètre qui sont C et max_iter. C est un hyperparamètre qui jouent sur la force de la régularisation. Plus C est proche de 0 plus la régularisation est forte. max_iter correspond aux nombre d'itérations maximum pour la convergence du modèle.

Ce modèle a été entraîné avec une validation croisée à 3 blocs. Notre stratégie d'optimisation est plutôt correcte car le recall et l'AUC de notre dataset de validation sont situés autour de 0.8.

Interprétabilité globale et locale

Notre modèle est interprétable pour éviter un rejet de la part des métiers sans expertise en data. La compréhension du modèle est permise par SHAP. SHapley Additive exPlanations est un algorithme issue de la théorie des jeux pour expliquer vie des valeurs de shapley un modèle de machine learning.

L'interprétabilité globale cherche à identifier les variables les plus importantes du modèle en analysant les contributions de chaque variable en output. Elle permet de rendre le processus de la prise de décision transparent pour toutes les données. Dans notre modèles ce sont par exemples les sources de revenus externes qui contribuent le plus à l'accord d'un crédit.

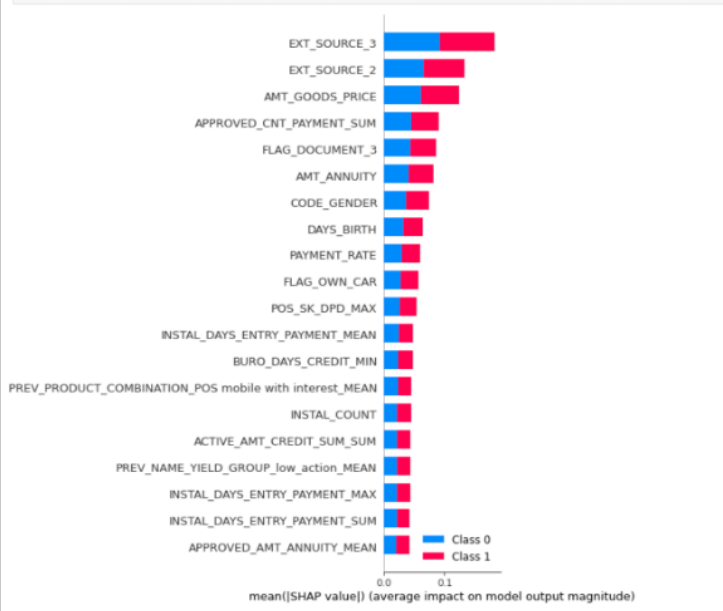

Figure 5: Interprétabilité globale

L'interprétabilité locale cherche à décrypter le comportement du modèle à l'échelle d'un individu. Elle permet d'identifier la contribution de chaque variable sur la probabilité de défaut de crédit d'un client spécifique. C'est un moyen d'améliorer la communication et la justification des résultats de l'algorithme au client. Ci-dessous, un résultat de la construction de la probabilité de défaut d'un client lambda.

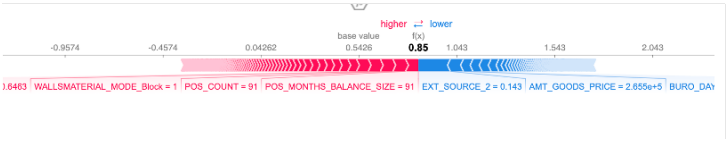

Figure 6: Interprétabilité locale

Limites et améliorations possibles

En conclusion, notre modèle permet d'obtenir des métriques d'évaluation satisfaisante. Nous avons un AUC de et un recall autour de 0.8. De plus le temps d'exécution est inférieur à 1h. Le point négatif est que l'accuracy de 0.5 est équivalente à un classifieur aléatoire. Cela nous permet d'avoir des clients pour le crédit mais cela montre que beaucoup sont éjectés de l'offre de crédit. Notre stratégie d'optimiser le recall et l'AUC dessert l'accuracy. Pour autant, ce modèles est le meilleur

que nous ayons testés. Ci-dessous des résultats d'autres modèles.

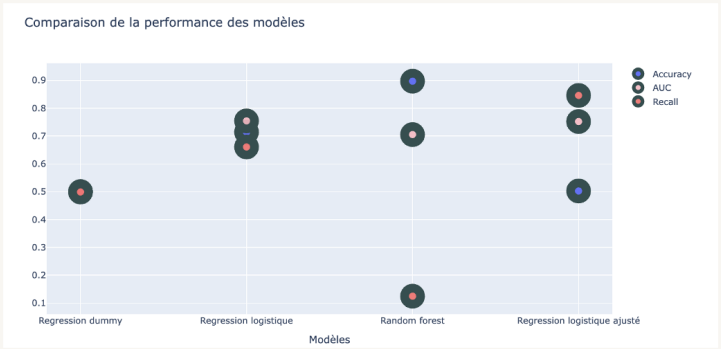

Figure 7: Résultats comparatifs avec d'autres pistes de modélisation

On peut soulever des limites technique et éthiques à notre modèle de scoring.

Le modèle que nous utilisons n'est peut-être pas le meilleur techniquement. Pour améliorer la performance deux approches sont envisageables. On peut soit améliorer le tuning des hyperparamètres du modèle ou sinon changer de modèle en s'orientant vers de nouvelle approches ensemblistes (boosting ou randomforest tuné différemment).

On peut soulever aussi un problème éthique. L'idée de ne pas accorder un crédit à un individu car il est une femme ne semble pas éthique. Peut-être pourrait-il intéressant de discuter de l'éthique de chaque variable inclues dans le modèle ? Peut-être que les femmes ont une caractéristique statistique plus présente que les hommes et qu'elle manque à notre modélisation... Pénaliser des individus sur des caractéristiques qui ne dépendent pas d'eux-même peut représenter un risque juridique et un réjet éthique.

References

[1] Marc-Au (2019) ,*La régression logistique expliquée à ma grand-mère*

[2] Morand STUDER (2017) ,*interpretabilite-des-modeles-de-machine-learnings*

[3] Openclassrooms (2022) ,*Réalisez des modélisations de données performantes*

[4] sklearn (2022) ,*sklearn LogisticRegression doc*