
Prêt à dépenser

Prêt à dépenser

Mission

1. Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.

2. Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.

Prêt à dépenser

Mission 1

Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.

Prêt à dépenser

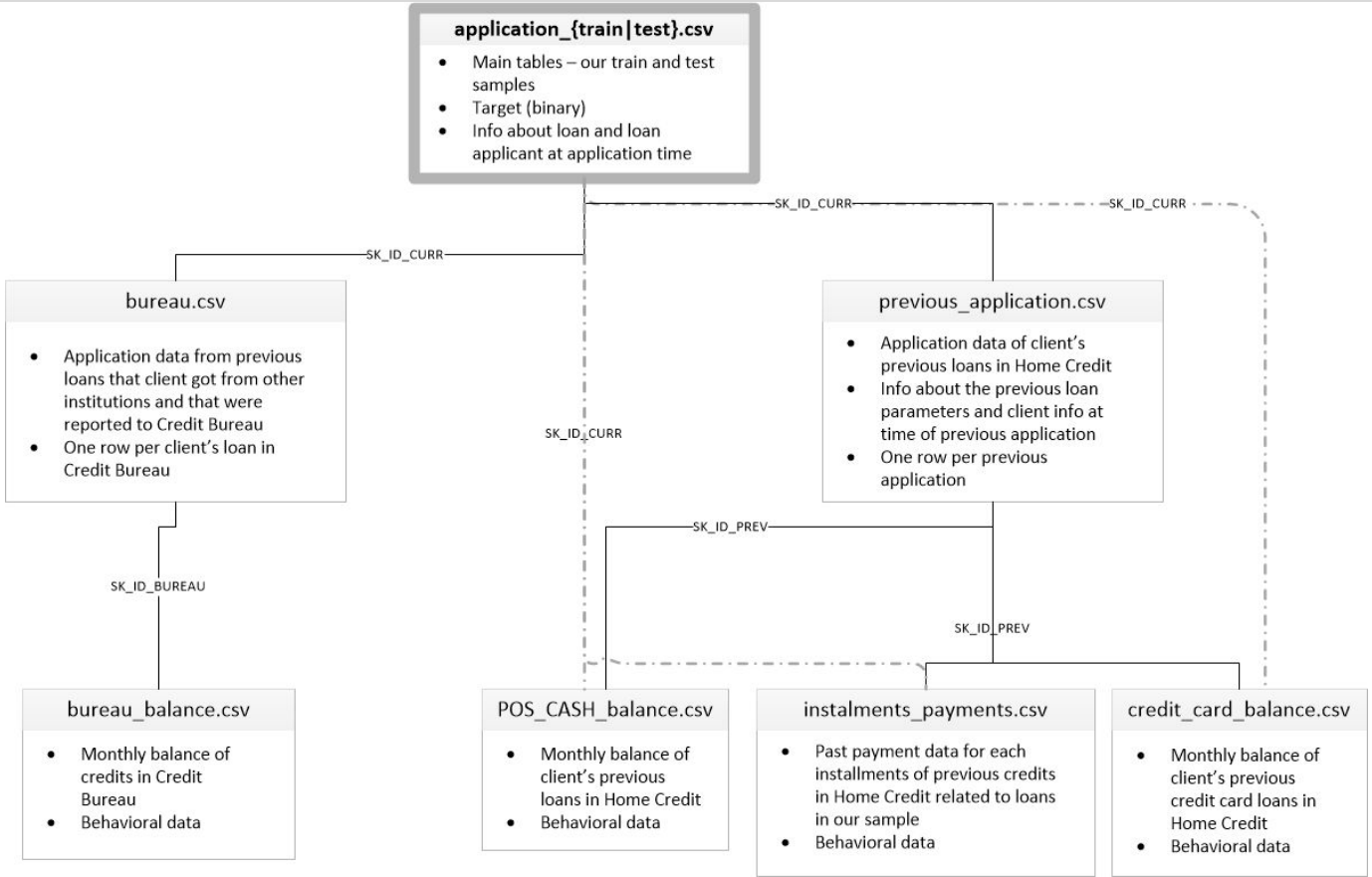
Les données

Avant engineering :

8 fichiers csv

Après merge et engineering :

Dataframe avec 114 features et 307507 individus



Prêt à dépenser

Data engineering

Features catégorielles :

NaN ad hoc : la valeur XNA, qui correspond à NaN pour les features catégorielles, est transformé en Nan

Features binaires : les features à 2 valeurs sont transformées en 0 et 1

Features quantifiables : les features quantifiables, par exemple le niveau d’ études qui peut être réinterprété comme années d’études, sont transformé en numériques

Features à valeur dominante : les features qui ont un nombre négligeable de valeurs minoritaires, sont supprimées

Hyponymes : Réduction du nombre de valeur en les substituant par un hyperonyme commun (par exemple, ‘loan’ à la place de 'Car loan' et 'Microloan')

One hot encoding

Prêt à dépenser

Data engineering

Les techniques de feature engineering appliquées aux données :

Outliers : Transformation des valeurs aberrantes en NaN.

Composition et dérivation de features et drop : par exemple
df['INCOME_CREDIT_RATIO'] = df['AMT_INCOME_TOTAL'] / df['AMT_CREDIT']

Agrégations par ID : plusieurs lignes avec le même ID son agrégées en calculant la moyenne ou la somme pour les features, ou, moins souvent, les valeurs uniques ou le nombre

Ré-Agrégations par valeurs Accepted et Rejected et puis ID

Drop des features similaires et avec une corrélation forte : en s'appuyant sur un heatmap

Permutation feature importance : drop des features moins importants

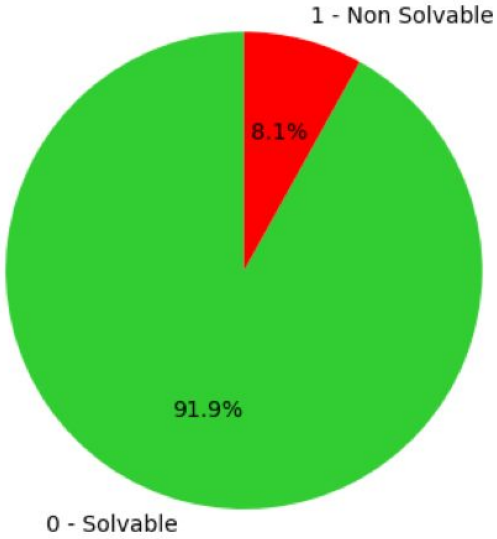
Drop des colonnes avec plus de 75% de nans

Prêt à dépenser

Resampling

La distribution des TARGET 0 (pour « solvable ») et 1 (pour « à risque de défaut ») est très déséquilibrée :

Plusieurs tests de resampling ont été fait tout au long du en utilisant la librairie Imblearn et la technique SMOTE pour rééquilibrer les deux classes en créant des nouveaux individus de classe 1, mais on n'a pas obtenu des améliorations significatives.

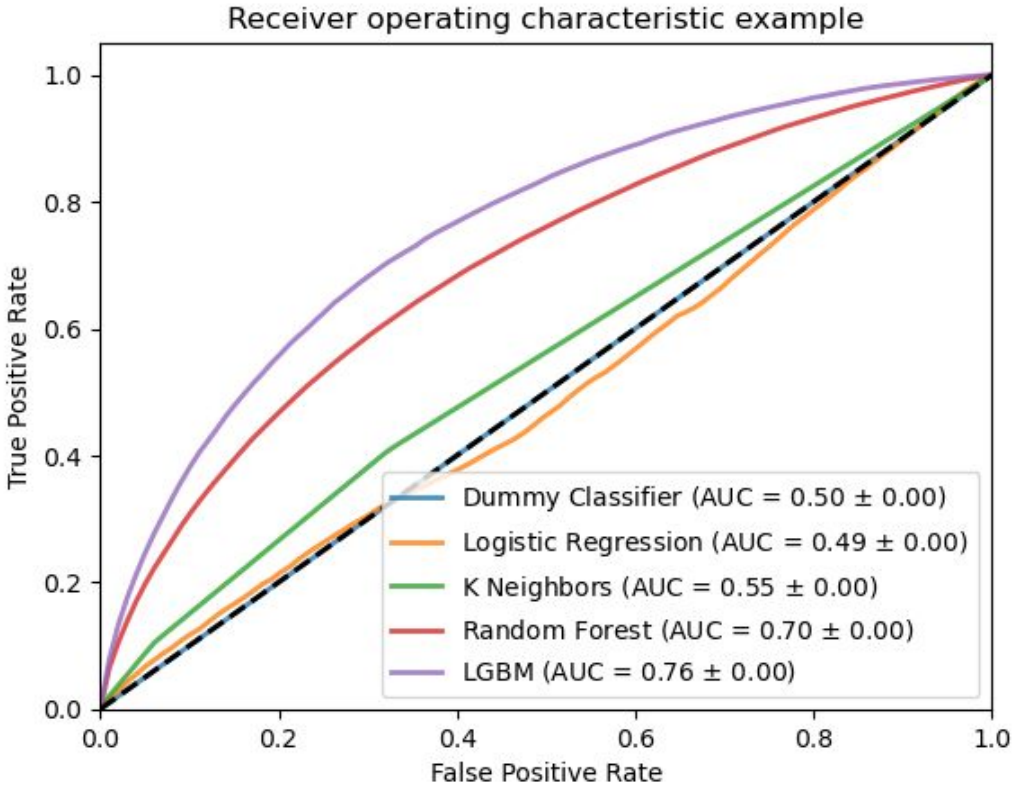


L'utilisation de SMOTE demande un dataframe sans NaNs, donc à chaque fois, la pipeline est :

```
smote_pipeline=imbPipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('sampling', SMOTE()),
        ('classification', <classifier> )])
```

Prêt à dépenser

Meilleur classifier, métrique ROC AUC



DummyClassifier()
LogisticRegression()
KNeighborsClassifier()
RandomForestClassifier()

LGBMClassifier()

```
preprocess  = Pipeline([
    ('imputer',
SimpleImputer(strategy='me
dian')),
    ('scaler', RobustScaler()),
])
```

Recherche des hyperparamètres, métrique ROC AUC

HalvingGridSearch

Stratified Kfolds : 3

9 hyperparamètres, 1152 combinations, 3436 fits

Sans SMOTE AUC score 0.7416738155817025

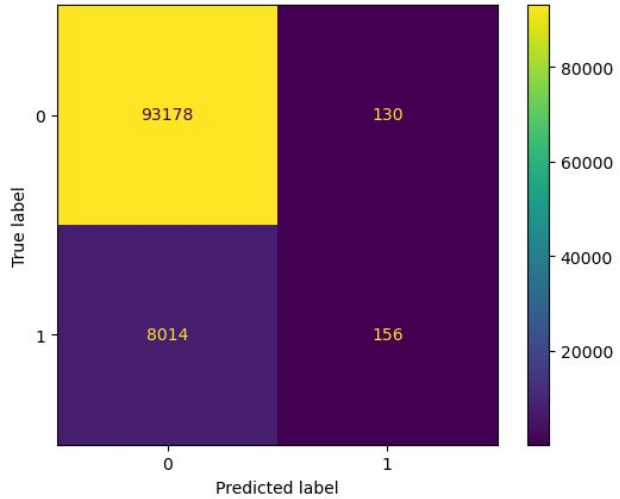
Avec SMOTE AUC score 0.72340286325661

Prêt à dépenser

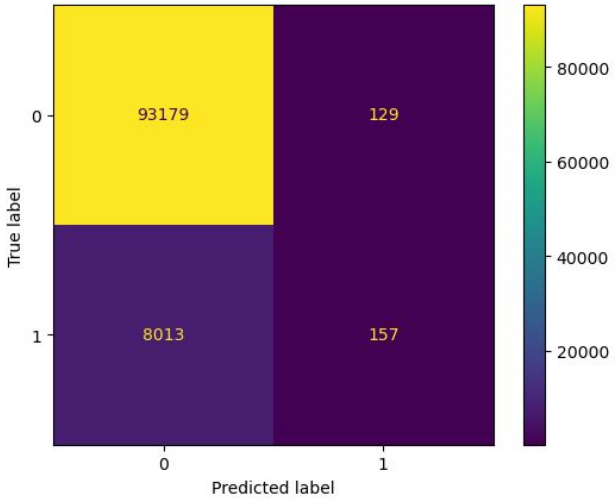
Confusion matrix

Le classifieur optimisé a une meilleure performance. Mais reste modeste.

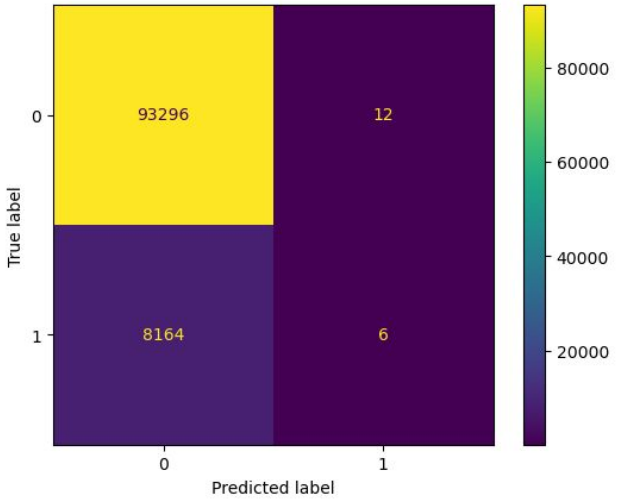
Baseline



Optimisé

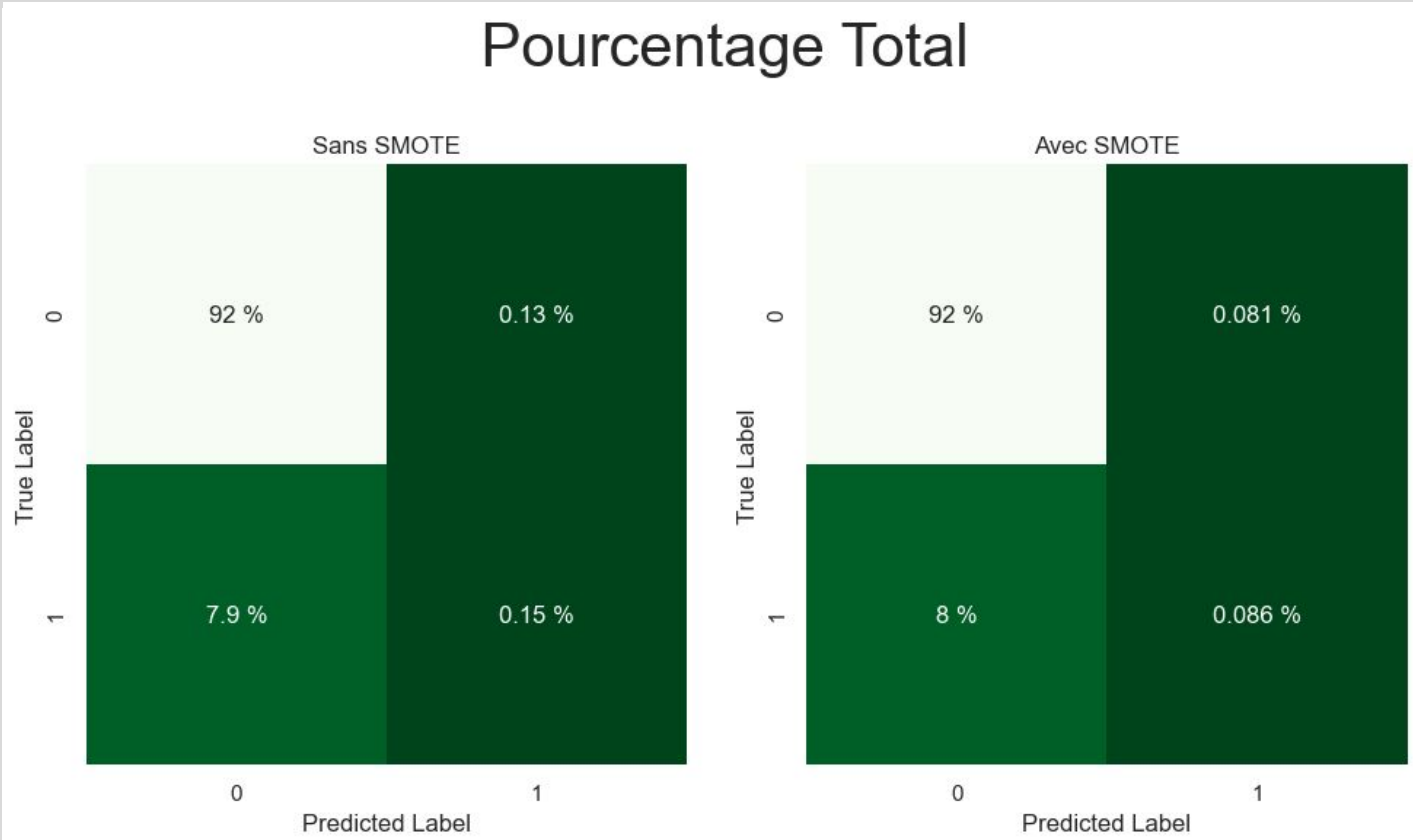


Optimisé et SMOTE



Prêt à dépenser

Confusion matrix



Prêt à dépenser

Recherche des hyperparamètres, métrique métier

Chaque résultat dans une matrice de confusion a un coût ou un bénéfice pour notre banque :

- TP: Clients non solvables identifiés comme non solvables. True one. Coût 0, bénéfice 0. La banque ni gagne ni risque rien.
- TN: Clients solvable identifiés comme solvables. True zero. Bénéfice 1. La banque gagne sur les intérêts payés par les clients.
- FP: Clients solvable identifiés comme NON solvables. False one. Coût 1. La banque perd un bon client.
- FN: Clients NON solvable identifiés comme solvables. False zero. Coût 10. La banque risque une perte d'argent. La perte d'argent est supérieure parce qu'il s'agit de toute la somme prêtée.

		Predicted	
		0	1
True	0	TN: 1	FP: -1
	1	FN: -10	TP: 0

Prêt à dépenser

Recherche des hyperparamètres, métrique métier

HalvingGridSearch

Stratified Kfolds : 3

8 hyperparamètres, 1944 combinations, 5832 fits

Sans SMOTE	B score 0.7056487350282747
Avec SMOTE	B score 0.7026483878421282

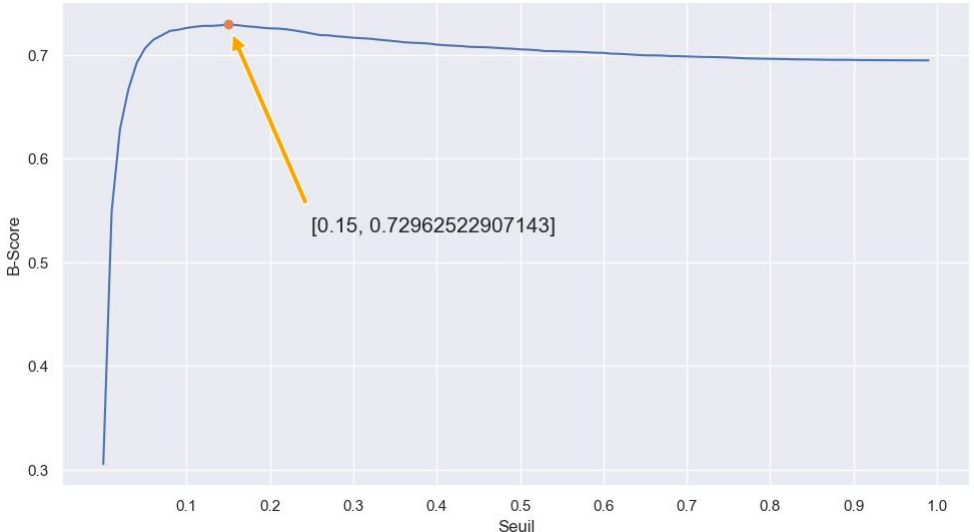
Prêt à dépenser

Seuil, métrique métier

Dans un classifier, l'appartenance à une classe binaire est calculée sur une probabilité entre 0 et 1, avec un seuil à 0,5. Pour repérer le seuil optimal selon la métrique métier, on calcule le coût global pour chaque seuil entre 0 et 1, avec des intervalles de 0,01

Seuil optimal : 0.15

B-Score=0.72962

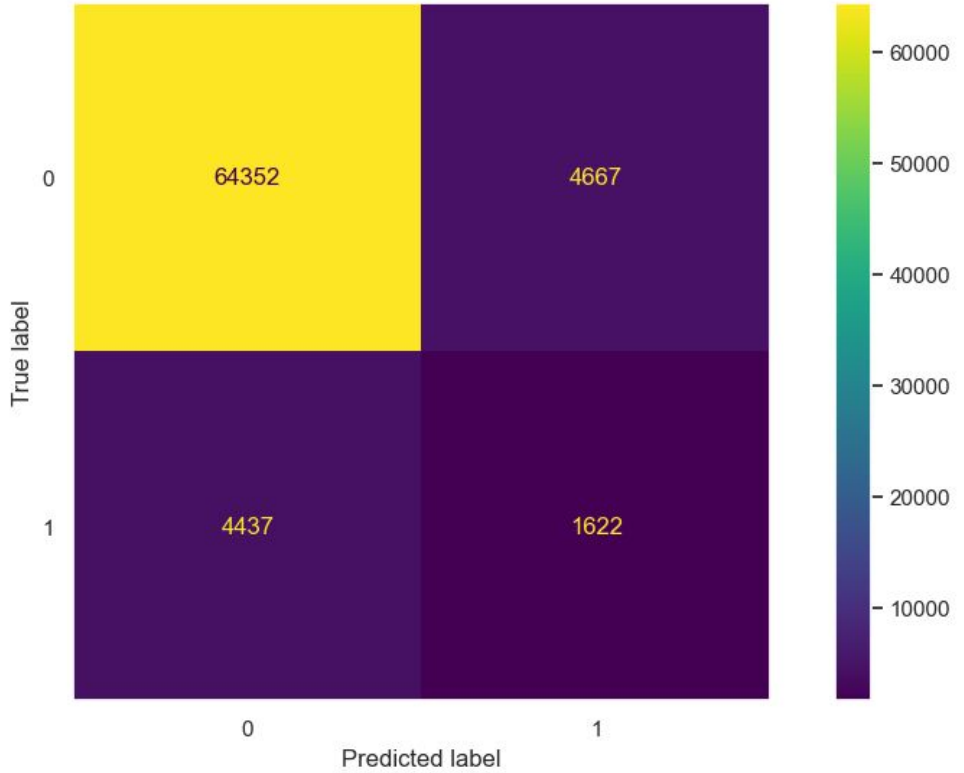


Prêt à dépenser

Matrice de confusion

La matrice de confusion avec optimisation métier et seuil.

Considérablement meilleure que sans métrique métier.



Prêt à dépenser

Mission 2

2

Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.

Prêt à dépenser

Dashboard

Critères :

- Permettre de visualiser le score et l'interprétation de ce score pour chaque client de façon intelligible pour une personne non experte en data science : **Tab 3**, Cards et Gauge
- Permettre de visualiser des informations descriptives relatives à un client : **Tab 3**, Table et Shap
- Permettre de comparer les informations descriptives relatives à un client à l'ensemble des clients ou à un groupe de clients similaires : **Tab 4**, Table et Plot
- **Tab 1** et **Tab 2** montrent des informations générales sur l'ensemble des clients, entre autre la feature importance.

Prêt à dépenser

API

api.py

2 endpoints

1. SHAP
2. PREDICT_PROBA

1 fichier pickled

https://oc7.herokuapp.com/

Procfile

requirements.txt

Prêt à dépenser

Dashboard

mydashboard.py

https://oc7.herokuapp.com/

Procfile

Requirements.txt

2 Fichier csv : dataframe, dataframe imputed

imputer.py

Prêt à dépenser

Dashboard

- DEMO https://oc7.herokuapp.com/

Prêt à dépenser

Limites et améliorations

Un certain nombre de pistes sont parcourables pour améliorer la performance du modèle :

1. Un features engineering plus poussé. Feedback d'un expert de banque.

2. Plus de ressources pour le tuning des hyperparamètres.

3. Tester l'optimisation bayesienne, comme dans le kernel d'exemple.

4. Autres techniques de resampling ou d'autres hyperparamètres pour les données déséquilibrées.

5. Métrique métier validée par un expert. Elle pourrait être plus fine et quantifier le risque en fonction du prêt.