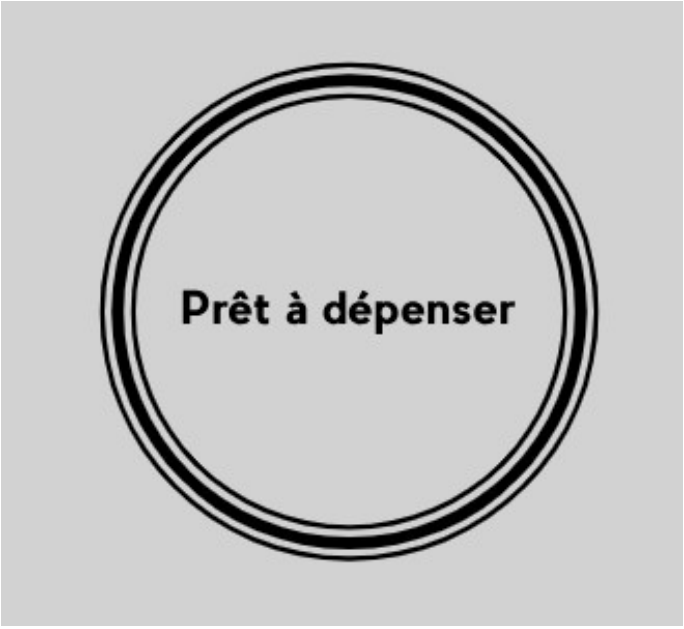
NOTE METHODOLOGIQUE

Implémenter un modèle de scoring



Projet 7 Data Scientist

Openclassrooms

Serge Davister 09/2023

I. Demande du client

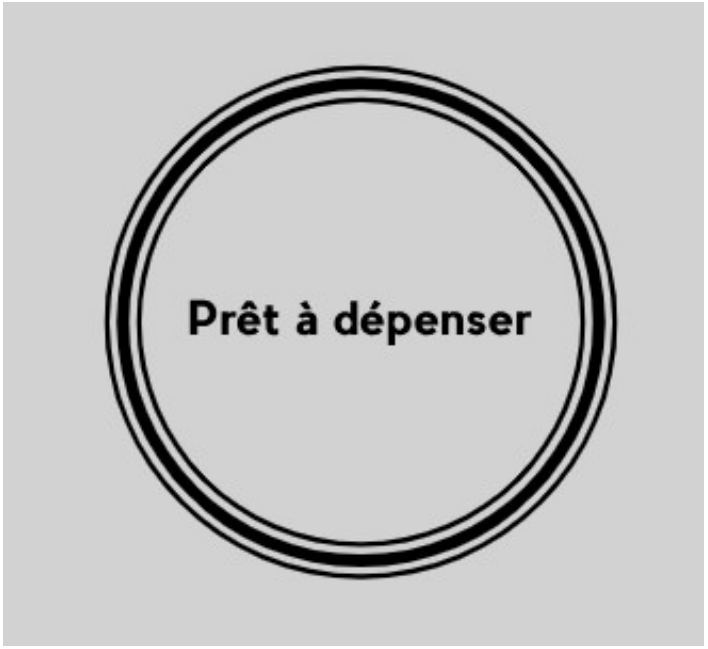
II. Traitement des données

III. Déséquilibre des classes et métriques

IV. Modélisation et optimisation des paramètres

V. Interprétation globale et locale du modèle

VI. Conclusion



I. Demande du client

La société financière Prêt à dépenser propose des crédits à la consommation pour des personnesayant peu ou pas du tout d'historique de prêt.

Modèle de scoring

L'entreprise souhaite mettre en œuvre un outil de scoring crédit qui calcule la probabilité qu'unclient rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. Elle souhaite donc développer un algorithme de classification en s'appuyant sur des sources de données variées(données comportementales, données provenant d'autres institutions financières, etc.)

Les données originales sont téléchargeables sur Kaggle à cette adresse.

Dashboard interactif

De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d'octroi de crédit. Cette demande de transparence des clients va tout à fait dans le sens des valeurs que l'entreprise veut incarner. **Prêt à dépenser** décide donc de développer un dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

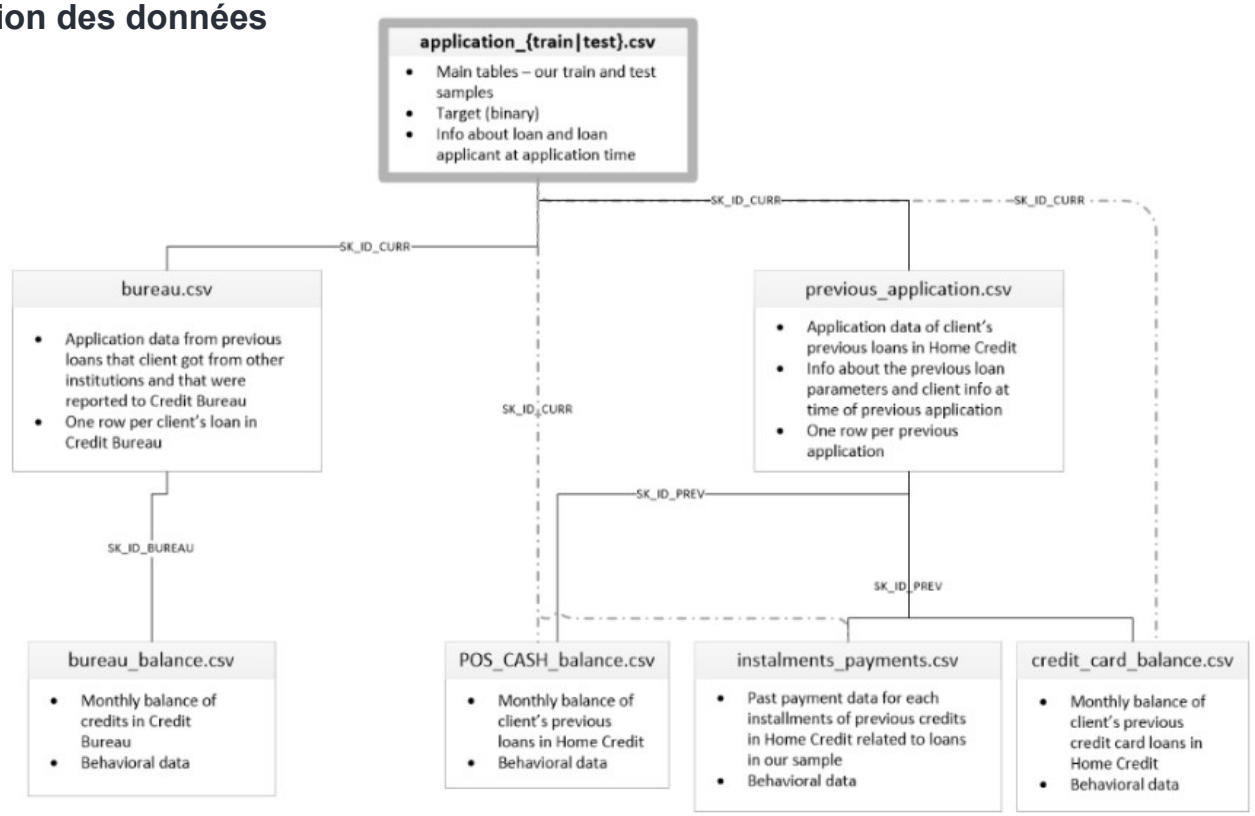
Le dashboard réalisé avec Streamlit est accessible en cliquant ici

Tous les programmes sont accessibles sur Github.

II. Traitement des données

Prêt à dépenser

Présentation des données



Les données sont réparties dans 7 fichiers .
La table « application » regroupe des informations personnelles ainsi que les données relatives au crédit demandé par les clients.Elle contient 2 jeux de données : l'application **train** regroupant 307 511 clients ayant demandé un crédit pour lesquels on connaît la finalité (accord ou refus de **Prêt à Depenser**)
L'application **test** qui contient les demandes en cours pour lesquelles la décision n'a pas encore été prise .
Les autres fichiers contiennent les données historiques de prêt de ces clients auprès de **Prêt à Depenser** (Previous_application) ou auprès d'autres organismes de prêts (bureau)

Prêt à dépenser

Les données ne peuvent pas être utilisées directement. Il y a lieu de vérifier si elles ne comportent pas de valeurs manquantes ou aberrantes. C'est l'étape du pré-traitement des données – **Data preprocessing -** qui commence par la phase de **nettoyage** des données. Ces erreurs peuvent survenir lors de l'encodage ou lors de l'acquisition de données lorsqu'il y a du bruit. Il peut s'agir de valeur incomplète manquante ou erronées.

J'ai écarté les variables comportant plus de 10 % de valeurs manquantes.
Les variables numériques avec moins de 10 % de données manquantes ont été complétées avec la valeur de leur plus proches voisins (KnnImputer)
Les variables catégorielles avec moins de 10 % de données manquantes ont été complétées avec la valeur de modalité la plus fréquente.

Une autre étape importante du pré-traitement est la réduction de données qui consiste à écarter les données qui ne seraient pas utiles pour notre modèle. La sélection des caractéristiques **- features selection –** permet de faire le tri et de ne conserver que les variables d'entrée les plus pertinentes .

Les entrées à variance nulle ou faible n'ont pas d'intêret pour notre modèle car elles n'interagissent pas ou peu avec notre variable cible. De même les variables qui sont fortement corrélées entre elles peuvent dégrader les performances du modèle.(utilisation de selecteur VarianceThreshold)

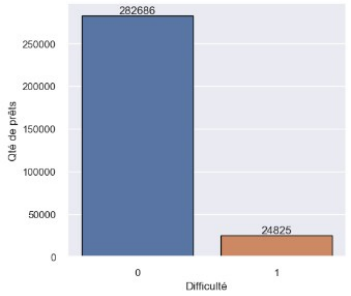
Enfin les étapes de **transformation** de données sont essentielles pour obtenir les meilleures performances du modèle.Lorsque les variables d'un jeu de données ont des ordres de grandeur différents, ces derniers peuvent fausser les estimations des algorithmes d'apprentissage automatique. J'ai effectué une **normalisation /standardisation** des données numériques afin qu'elles soient comparables sur une échelle commune avec le transformer StandardScaler qui centre et réduit les données(espérance nulle , variance =1 et écart-type =1)

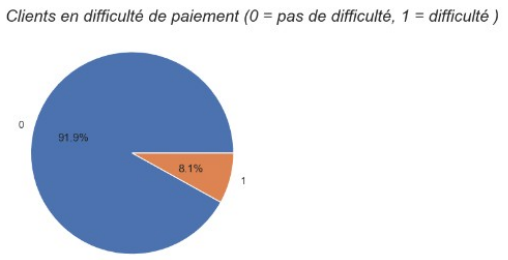
De même, les variables continues avec de nombreuses modalités peuvent être découpées et regroupées dans des intervalles;ce qui permet de réduire le nombre de modalités d'une variable et de supprimer les valeurs aberrantes présentes.On parle de discrétisation parce qu'en imposant le nombre de modalités, ces variables sont devenues des **variables discrètes**.

Nous obtenons un jeu de données d'entraînement de 307.511 clients décrits par 55 variables .

III. Déséquilibre des classes et métriques

Nombre de clients « sans difficulté » Vs Nombre de clients « avec difficultés de paiement »





Il y a 8,1 % de clients en difficulté de paiement contre 91.9% sans difficulté. En ML, cette différence importante de classes peut nuire à la fiabilité des prédictions .

1. **solution au niveau des données.**

On peut ré-équilibrer les classes en augmentant artificiellement le nombre de données dans la classe minoritaire (upsampling ou sur-échantillonnage) ou en réduisant dans la classe majoritaire (downsampling ou sous-échantillonnage)

- **RandomOverSampler** : on va créer de nouveaux points à partir de points existants de la classe minoritaire
- **SMOTE** : Synthetic Minority Oversampling Technic ou suréchantillonnage minoritaire synthétique et variantes **Borderline SMOTE** et **Borderline SVMSMOTE**
- A**DASYN** : ADAptive SYNthetic sampling .Ici le nombre d' échantillons synthétiques généré est proportionnel au nombre d'échantillons sélectionnés qui ne sont pas de la classe minoritaire.

- **RandomUnderSampler**: On peut retirer aléatoirement des points de la classe majoritaire pour réduire le déséquilibre.
- **NEARMISS** Son principe est de supprimer les observations de la classe majoritaire lorsque des observations associées à des classes différentes sont proches l'une de l'autre.

2. **solution de résolution au niveau des algorithmes.**

On affecte un poids plus important à la classe minoritaire.(apprentissage sensible aux coûts) On apprend à notre modèle que le fait de bien classer un point de la classe minoritaire est plus important que de bien classer un point de la classe majoritaire. Cela signifie qu'une erreur de classification d'un point de la classe minoritaire est considérée par le modèle comme étant plus grave qu'une erreur de classification sur la classe majoritaire. Et comme l'objectif du modèle est de toujours d'optimiser le paramètre prédéfini, l'algorithme va donner plus d'importance à la classe minoritaire et donc à améliorer les prédictions sur celle-ci. Ce qui dans notre cas est.le but recherché.

Avec des données déséquilibrées, il faut opter pour des métriques qui sont moins influencées par la classe majoritaire tels que le **recall** et le **F-score**.Si on ne le fait pas ,on peut penser que le modèle donne de bons résultats parce que le score est élevé alors que les prédictions ne sont correctes que pour la classe majoritaire et qu'une grande partie des points de la classe minoritaire sont mal classifiés. Le score est affecté par le déséquilibre des classes. De plus dans notre cas, une erreur de prédiction est plus dommageable que l'autre.

Les faux positifs sont de bons clients marqués comme mauvais.Le prêt ne leur est pas accordé.La banque perd le client (erreur de type 1)

Les faux négatifs sont de mauvais clients marqués comme bons . Le prêt leur est accordé. Cela représente une perte financière pour la société de crédit (erreur de type 2)

la mesure **Fbeta** calcule une moyenne harmonique pondérée de précision et de rappel tout en privilégiant les scores de rappel supérieurs aux scores de précision.Le modèle minimise les erreurs de classification pour la classe positive tout en privilégiant les modèles qui minimisent les faux négatifs plutôt que les faux positifs.

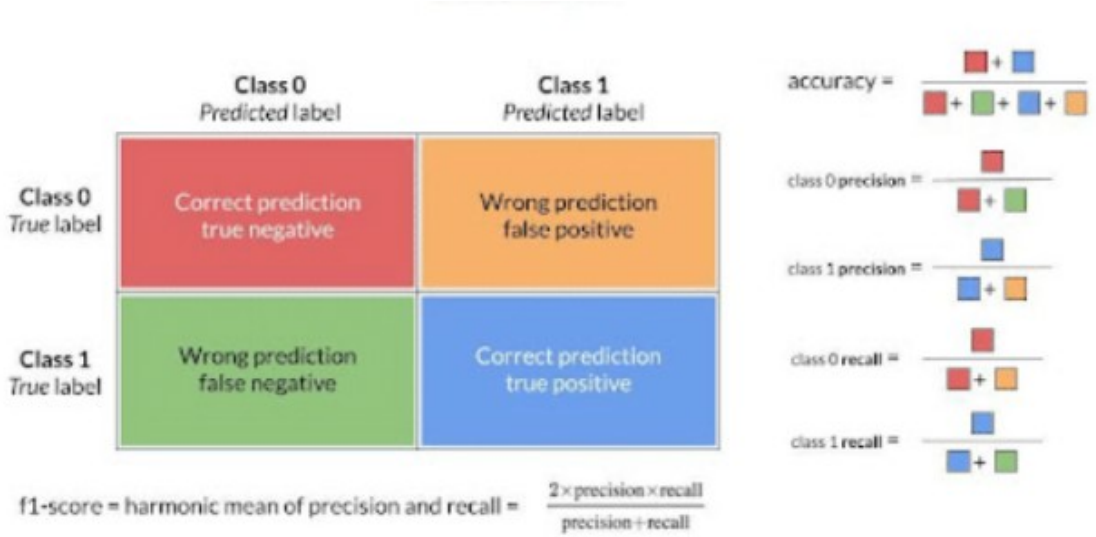
La **précision** est le nombre de résultats positifs corrects divisé par le nombre de tous les résultats positifs.

Le **rappel**(recall) est le nombre de résultats positifs corrects divisé par le nombre de résultats positifs qui auraient dû être renvoyés .

Le **score F1** est une mesure de la précision d'un test. Il prend en compte à la fois la précision et le rappel du test pour calculer le score. Le score F1 peut être interprété comme une moyenne pondérée de la précision et du rappel. **F1 score = 2 * ((precision * recall) / (precision + recall))**

L**'accuracy** décrit la performance du modèle sur les individus positifs et négatifs de façon symétrique. Elle mesure le taux de prédictions correctes sur l'ensemble des individus .

La **courbe ROC** (Receiver Operating Charasteristic) et l'**AUC** (Area Under Curve, soit l'aire sous la courbe) sont deux indicateurs permettant d'analyser un modèle et en particulier son seuil de détection. La courbe ROC (ou courbe des caractéristiques de fonctionnement du récepteur) est une mesure graphique pour évaluer les performances ou la précision d'un classificateur, qui correspond à la proportion totale d'observations correctement classées.



IV. Modélisation et optimisation des paramètres

Prêt à dépenser

J'ai utilisé une cross Validation cv=10 pour l'évaluation des modèles

modèle	Fit time	Durée	Train F1	Test F1	Train FBETA	Test FBETA	Train ROC_AUC	Test ROC_AUC	Train ACCURACY	Test ACCURACY
Dummy_Classifier	0.051093	0.021710	0.000000	0.000000	0.000000	0.000000	0.500000	0.500000	0.919227	0.919227
Logistic_Regression	0.699524	0.030148	0.011967	0.011565	0.007561	0.007307	0.725908	0.724768	0.919025	0.919013
Decision_Tree_Classifier	6.245317	0.040607	1.000000	0.142896	1.000000	0.151202	1.000000	0.532757	1.000000	0.847559
Random_forest Classifier	69.040850	1.387221	0.999776	0.002753	0.999642	0.001724	1.000000	0.707526	0.999964	0.919264
Bagging Classifier	56.898929	0.305051	0.900613	0.035326	0.850036	0.023186	0.999658	0.629017	0.985393	0.916839
Linear_SVC	98.407131	0.034709	0.000460	0.000459	0.000288	0.000287	0.726211	0.725059	0.919190	0.919180
XGB_Classifier	27.729241	0.085213	0.146445	0.051114	0.097672	0.033518	0.852534	0.731719	0.924751	0.918321
Gradient_Boosting_Classifier	113.782099	0.144258	0.015052	0.010805	0.009493	0.006811	0.747273	0.735978	0.919482	0.919213
LGBM-Classifier	3.112440	0.195128	0.040243	0.023355	0.025677	0.014863	0.794201	0.741952	0.920309	0.919190

En prenant en compte le déséquilibre des classes,les résultats sont meilleurs

modèle	Fit time	Durée	Train F1	Test F1	Train FBETA	Test FBETA	Train ROC_AUC	Test ROC_AUC	Train ACCURACY	Test ACCURACY
Dummy_Classifier_balanced	0.054079	0.026107	0.000000	0.000000	0.000000	0.000000	0.500000	0.500000	0.919227	0.919227
Logistic_Regression_balanced	0.676804	0.036756	0.247414	0.246938	0.394980	0.394282	0.726768	0.725487	0.677792	0.677432
Decision_Tree_Classifier_balanced	6.718395	0.047535	1.000000	0.138201	1.000000	0.139127	1.000000	0.531089	1.000000	0.859201
Random_forest Classifier_balanced	77.899738	1.673278	0.999719	0.001608	0.999550	0.001006	1.000000	0.710965	0.999955	0.919245
Bagging Classifier_balanced	56.090249	0.298299	0.900613	0.035326	0.850036	0.023186	0.999658	0.629017	0.985393	0.916839
Linear_SVC_balanced	118.143720	0.057929	0.253522	0.253796	0.389811	0.390055	0.723965	0.723021	0.710979	0.711322
XGB_Classifier_balanced	47.856769	0.117413	0.186907	0.168497	0.364926	0.329448	0.859862	0.697027	0.297412	0.276841
Gradient_Boosting_Classifier_balanced	107.048856	0.139634	0.015052	0.010805	0.009493	0.006811	0.747273	0.735978	0.919482	0.919213
LGBM_Classifier_balanced	3.215434	0.207417	0.289864	0.261739	0.453486	0.409283	0.792369	0.742679	0.712227	0.701199

Je conserve uniquement les 3 meilleurs modèles pour affiner les paramètres de class_weight. Il s'agit de :

LGBM_Classifier_balanced, Logistic_Regression_balanced, XGB_Classifier_Balanced

linear_SVC_balanced a été écarté pour des questions pratiques de temps de calcul trop long

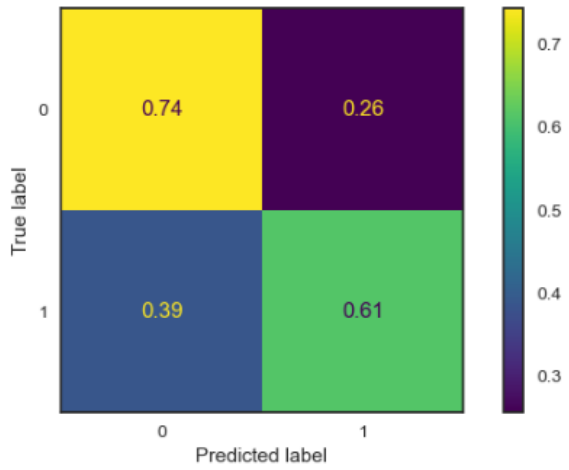
Tuning des hyper-paramètres du Cost Sensitive learning sur les 3 modèles retenus

```
# paramètres
seed=1234
lr_param_grid = [{'LR__class_weight': ['balanced',{0: 50, 1: 1},{0: 10, 1: 1}, {0: 1, 1: 10},{0: 1, 1: 50}]}]

#svm_param_grid = [{'SVM__class_weight': ['balanced',{0: 50, 1: 1},{0: 10, 1: 1}, {0: 1, 1: 10},{0: 1, 1: 50}]}]

lgbm_param_grid=[{'LGBM__scale_pos_weight':[99,50,10]}]

xgb_param_grid = [{'XGB__scale_pos_weight': [1, 10, 25, 50, 75, 99,100,1000]}]

#Pipelines
pipe_lgbm = Pipeline([('LGBM', LGBMClassifier(random_state=seed))])

pipe_lr = Pipeline([('LR',LogisticRegression(random_state=seed))])

#pipe_svm = Pipeline([('SVM', SVC(random_state=seed))])

pipe_xgb = Pipeline([('XGB', XGBClassifier(random_state=seed))])
```

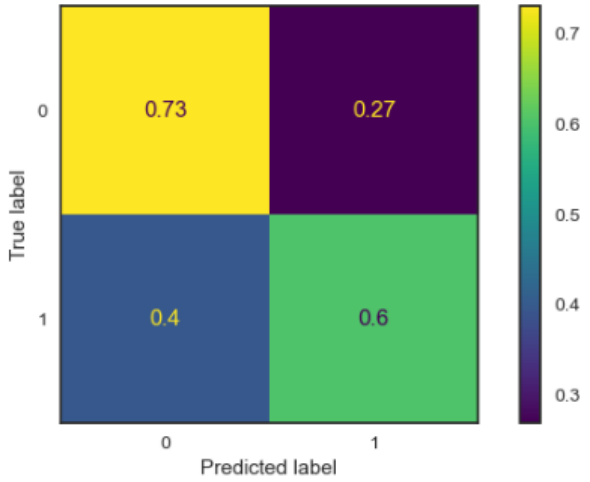
```
scoring = {'f1': 'f1','roc_auc':'roc_auc','accuracy':'accuracy',
           'fbeta' : make_scorer(custom_score)}

lgbm_grid_search = GridSearchCV(estimator=pipe_lgbm,param_grid=lgbm_param_grid,  scoring='f1',   cv=3)

lr_grid_search = GridSearchCV(estimator=pipe_lr, param_grid=lr_param_grid, scoring='f1', cv=3)

#svm_grid_search = GridSearchCV(estimator=pipe_svm, param_grid=svm_param_grid, scoring='f1', cv=3)

xgb_grid_search = GridSearchCV(estimator=pipe_xgb, param_grid=xgb_param_grid, scoring='f1', cv=3)
```

```
grids = [lgbm_grid_search, lr_grid_search, xgb_grid_search]  #, svm_grid_search
for pipe in grids:
    pipe.fit(X_train,y_train)
```

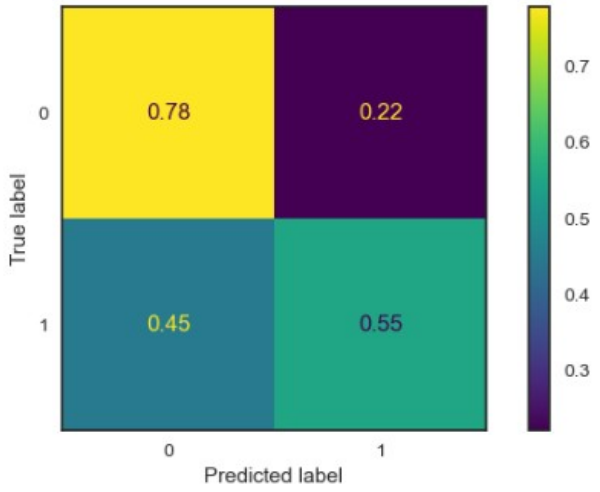
```
LightGBM Valid f1: 0.27
LightGBM Best Params: {'LGBM__scale_pos_weight': 10}
```




```
Logistic Regression Valid f1: 0.26
Logistic Regression Best Params: {'LR__class_weight': {0: 1, 1: 10}}
```




```
XGBoost Valid f1: 0.27
XGBoost Best Params: {'XGB__scale_pos_weight': 10}
```

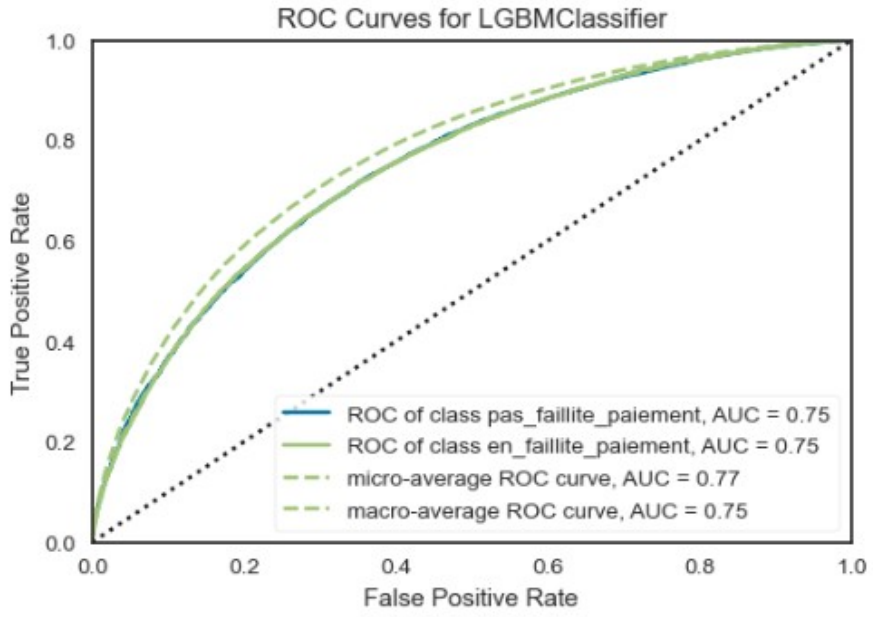


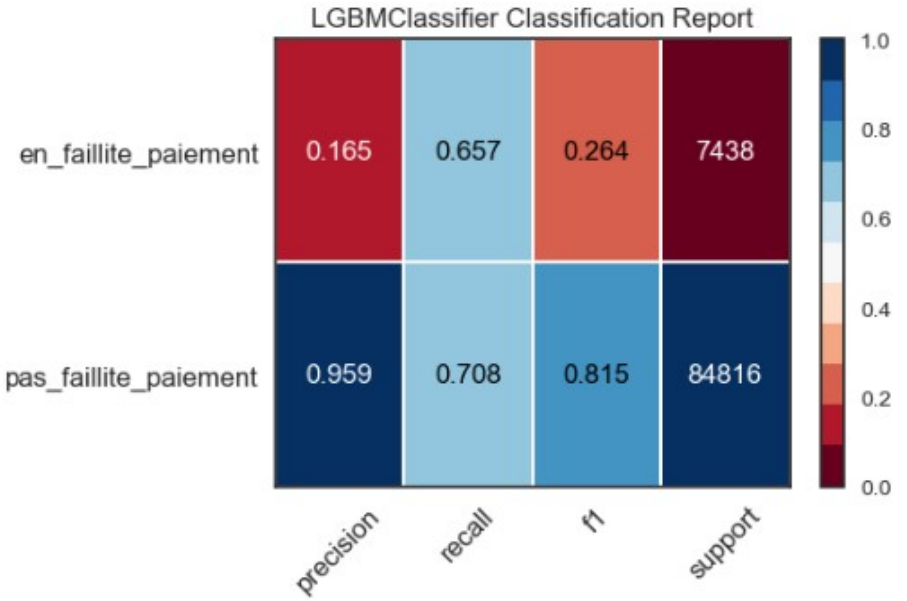
Le modèle LightGBM obtient le meilleur résultat avec un rapport de coût de 10 .

Tuning des hyper-paramètres de LightGBM

```
lgb_model = LGBMClassifier(boosting_type='gbdt',
                           learning_rate=0.15,
                           max_depth=5,
                           num_leaves=2**5,
                           n_estimators=100,
                           objective='binary',
                           class_weight="balanced",
                           random_state=1234,
                           subsample=0.1,
                           reg_alpha=10,
                           reg_lambda=10,

                           )
```

```
Train Accuracy : 71.46%
Test Accuracy : 70.41%

Train Precision : 18.12%
Test Precision : 16.48%

Train Recall : 72.01%
Test Recall : 65.68%

Train f1_score : 28.95%
Test f1_score : 26.35%

Train f2_score : 45.15%
Test f2_score : 41.13%

Train ROC-AUC : 71.71%
Test ROC-AUC : 68.25%
```






```
<Axes: title={'center': 'LGBMClassifier Classification Report'}>
```

Optimisation des hyper-paramètres avec OPTUNA

Optuna est un outil de recherche automatisé permettant l'optimisation des hyper-paramètres des modèles en Machine Learning.

Il faut au préalable définir une fonction coût.La fonction de perte ou de coût est la quantification de l'écart entre les prévisions du modèle et les observations réelles du jeu de donnée utilisé pendant l'entraînement. Une valeur Log Loss inférieure signifie de meilleures prédictions ; la valeur Log Loss d'un modèle parfait serait de 0.

```
Train Accuracy : 71.46%
Test Accuracy : 70.41%

Train Precision : 18.12%
Test Precision : 16.48%

Train Recall : 72.01%
Test Recall : 65.68%

Train f1_score : 28.95%
Test f1_score : 26.35%

Train f2_score : 45.15%
Test f2_score : 41.13%

Train ROC-AUC : 71.71%
Test ROC-AUC : 68.25%
```

Optuna

```
Train Accuracy : 75.75%
Val Accuracy : 72.65%

Train Precision : 22.17%
Val Precision : 16.73%

Train Recall : 79.77%
Val Recall : 60.11%

Train f1_score : 34.70%
Val f1_score : 26.17%

Train f2_score : 52.50%
Val f2_score : 39.58%

Train ROC-AUC : 77.58%
Val ROC-AUC : 66.93%
```

Le modèle est sauvegardé avec joblib

```
from joblib import dump, load
dump(model_LGBM, open('LGBM5_saved.joblib','wb'))
```

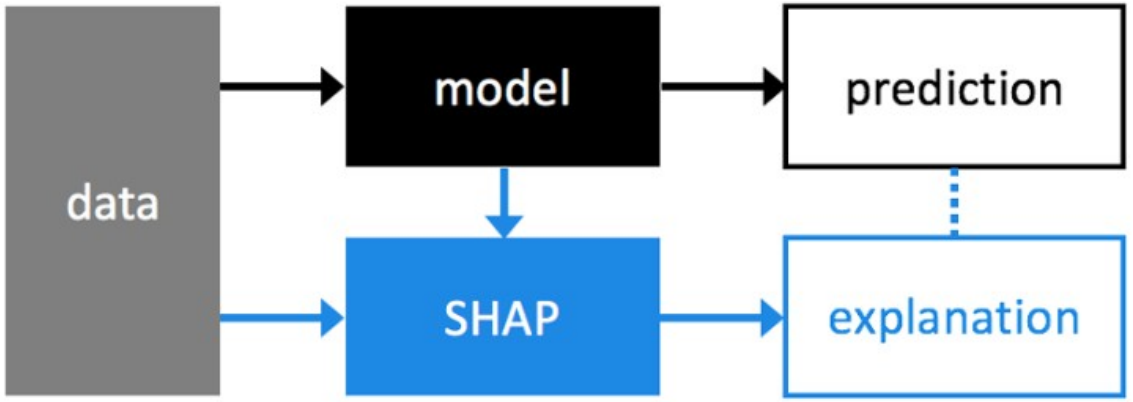
Le modèle est utilisé pour des prédictions sur le jeu de test

Prédictions sur l'ensemble de test

y_test_pred	COUNT	RATIO
0	34881	71.559577
1	13863	28.440423

V. Interprétation globale et locale du modèle

Cette technique permet d'expliquer les prédictions des modèles de Machine Learning de manière compréhensible aux humains. En attribuant une valeur à chaque caractéristique entrée, il montre comment et dans quelle mesure chaque caractéristique a contribué au résultat final de la prédiction. Il explique comment le modèle a pris sa décision et permet d'identifier les caractéristiques les plus importantes. Il fait cela en calculant la contribution moyenne de chaque caractéristique à la prédiction sur toutes les combinaisons possibles de caractéristiques.

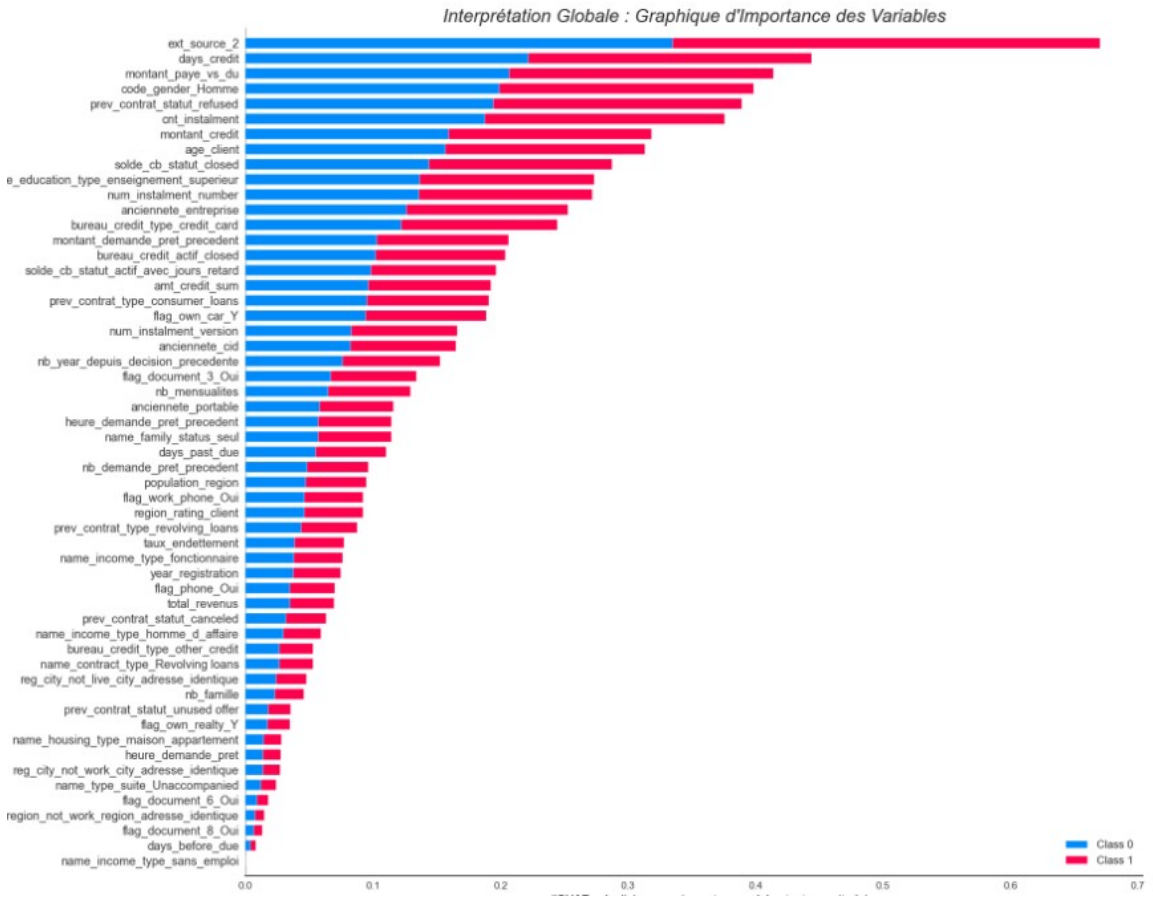


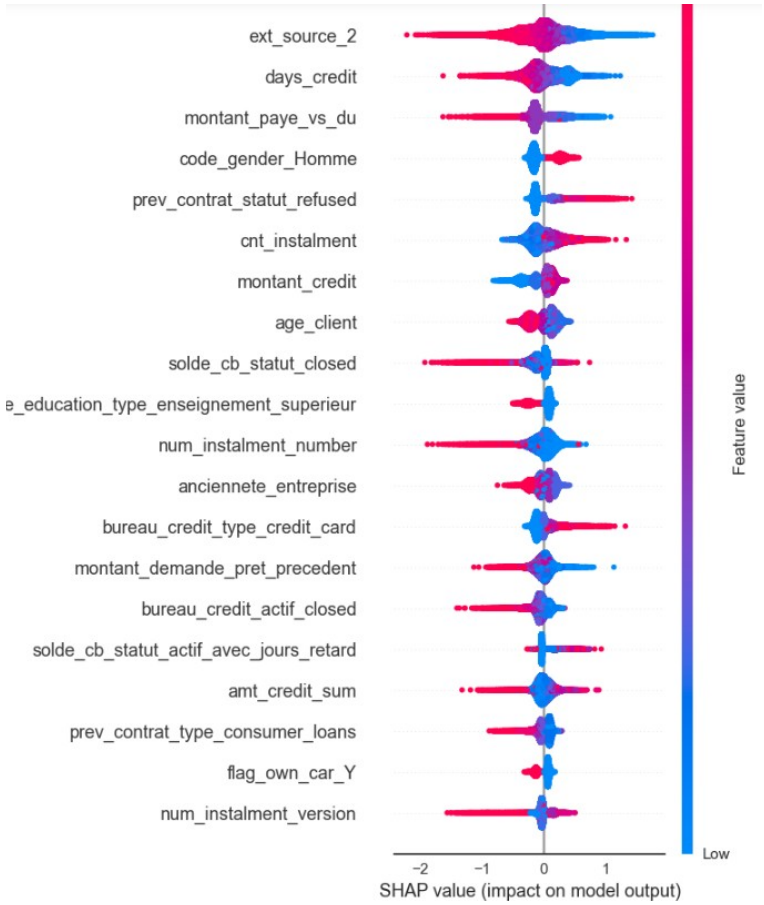
Les shap_values représentent la contribution de chaque caractéristique à la prédiction faite par le modèle pour chaque instance du dataset .

Les valeurs de Shapley calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable. Cependant, étant donné que l'ordre dans lequel un modèle voit les variables peut affecter ses prédictions,cela se fait dans tous les ordres possibles, afin que les fonctionnalités soient comparées équitablement. Cetteapproche est inspirée de la théorie des jeux.

Interprétabilité globale avec SHAP

Shap permet à la fois une interprétation globale et une interprétation locale



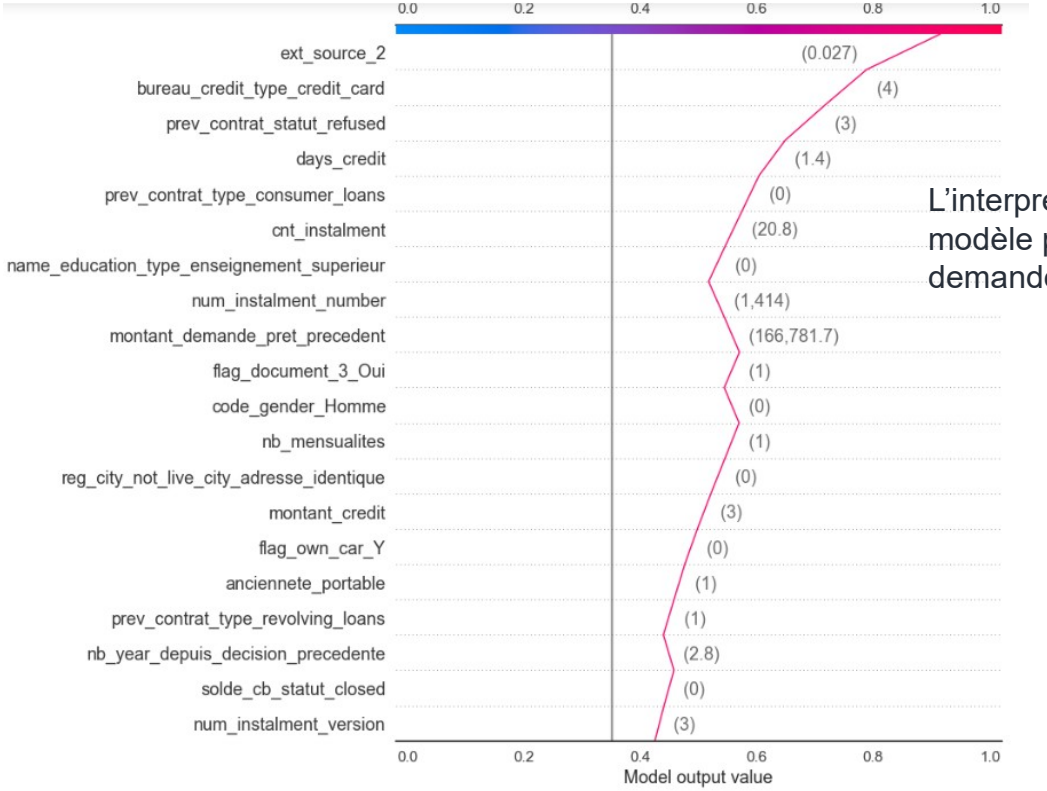


On peut regarder l'importance de chaque caractéristique pour la classe 1

L'interprétation globale cherche à expliquer le modèle dans sa globalité. C'est-à-dire quelles sont les variables les plus importantes pour le modèle. Dans notre cas, quelles caractéristiques affectent le plus le comportement général d'un modèle d'octroi de crédit ?

Interprétabilité locale avec SHAP

Le graphique decision_plot est une façon d'expliquer la prédiction. Il donne la valeur et l'impact sur le score de chaque variable par ordre d'importance Il indique la direction prise par la prédiction pour chacune des valeurs des variables affichées.



L'interprétation locale consiste à expliquer la prédiction f(x) du modèle pour un individu x donné. Dans notre cas, pourquoi la demande de prêt du client a-t-elle été approuvée ou rejetée ?

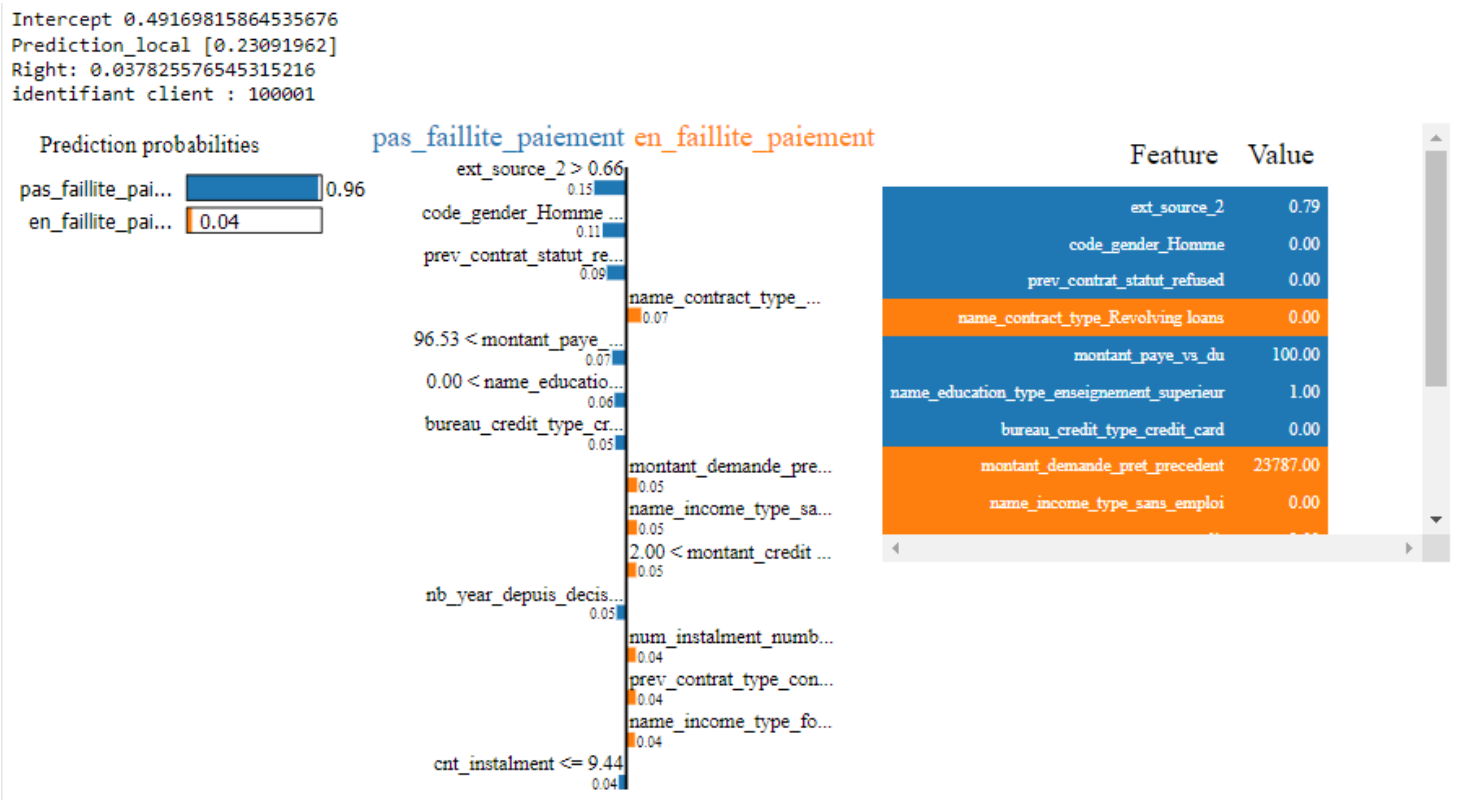
Un graphique force-plot situe la prédiction par rapport à la base value.

```
1 shap.plots.force(explainer.expected_value[1], shap_values[1][20, :], X_test.iloc[20, :],matplotlib = True)
```

Il indique les variables qui augmentent la probabilité du client d'être en défaut de paiement (en rouge) et celles qui la diminuent (en bleu), ainsi que la grandeur de cet impact.

Interprétabilité locale avec LIME

LIME est une technique qui permet de créer un modèle simple autour de la prédiction que nous voulons expliquer et d'utiliser ce modèle pour expliquer la prédiction. En pratique, pour une prédiction donnée , LIME va pertuber les entrées de la prédictions et regarder comment la pertubation a affecté la sortie du modèle. Les attributs qui affectent le plus la sortie lorsqu'ils sont pertubés sont considérés comme ayant une importance élevée pour cette prédiction .On est dans de l'**INTERPRETABILITE LOCALE**



Les 2 techniques donnent des résultats très similaires. LIME est indépendant du modèle et fournit des explications locales et interprétables, tandis que SHAP est spécifique au modèle et fournit des explications globales en utilisant une approche basée sur la théorie des jeux.

VI. Conclusion

Le data scientist dispose d'une panoplie d'outils pour l'aider dans la résolution de problèmes. Toutefois , l'expérience de la personne reste primordiale pour la partie nettoyage ,la sélection des variables ou pour choisir un modèle plutôt qu'un autre ou pour fixer les valeurs de départ pour le tuning des hyper-paramètres.

Ce qui pourrait être amélioré :

Le problème principal a été le manque de connaissances des métiers du secteur bancaire.Le vocabulaire employé n'est pas toujours pleinement compris et cela peut avoir des répercussions sur le choix et la façon de traiter certaines caractéristiques.
Il est clair aussi que le manque de connaissance du métier n'a pas permis de peaufiner la feature engineering.
Le choix des métriques s'est effectué avec le postulat qu'il était préférable de perdre un client plutôt que d'octroyer un prêt à un client susceptible de ne pas rembourser. Le niveau de risque acceptable n'a pas été défini en prenant en compte l'aspect métier.