

Implémentez un modèle de scoring

France LY



Repository Modélisation : https://github.com/flys-lf/scoring_model
Repository Déploiement API : https://github.com/flys-lf/deploiement_api

Contenu

- **Problématique & Jeu de données**
- **Preprocessing & Feature Engineering**
- **Modélisation**
- **Interprétation Locale & Globale**
- **Pipeline déploiement & Dashboard**
- **Data Drift**
- **Limites & Améliorations**





Problématique & Jeu de données

Problématique

La société « **Prêt à dépenser** » propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

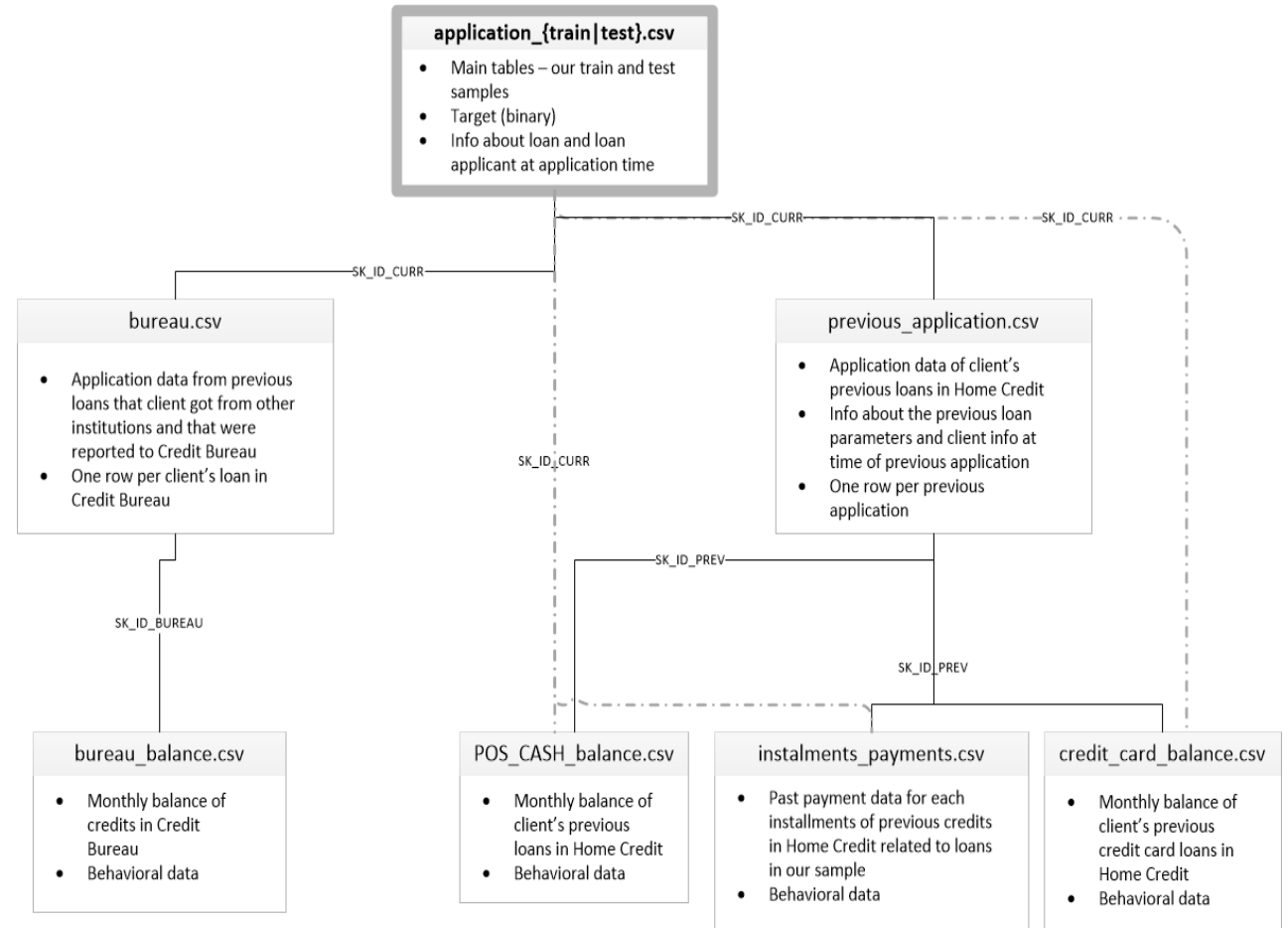
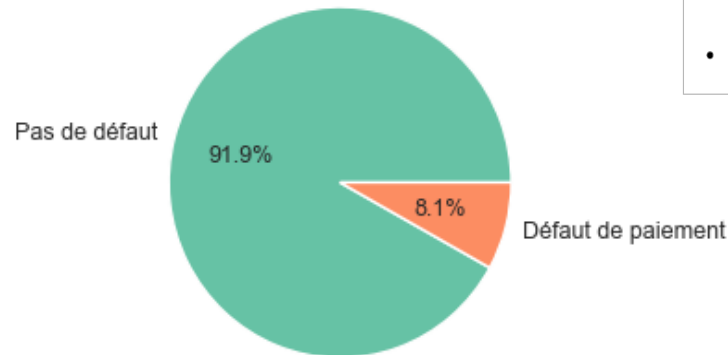
Objectifs :


- **Mettre en œuvre un outil de scoring crédit pour calculer la probabilité** qu'un client ne rembourse pas son crédit, puis classifie la demande en crédit accordé ou refusé.
- Développer un **algorithme de classification** (binaire) en s'appuyant sur des sources de données variées.
- **Transparence sur la décision d'octroi de crédit** pour permettre au chargé d'étude et au client de mieux comprendre le score attribué par le modèle.

Jeu de données

- **7 fichiers** avec en tout **219** variables.
- Données principales **Application** avec les informations personnelles des clients et relatifs au crédit demandé :
 - **Train** : 307 511 clients dont la décision d'octroi est connue (« Target »).
 - **Test** : 48 744 clients dont on ne connaît pas la décision d'octroi.
- Données historiques de prêt client dans d'autres institutions bancaires
- Données historiques prêt client dans la société « Prêt à dépenser ».
- Une **Variable Cible** « Target » décrivant si le client a des difficultés de paiement ou non.

Target Distribution By Value





Preprocessing & Feature Engineering

Preprocessing & Feature Engineering

- Utilisation des fonctions de preprocessing du kernel [LightGBM with Simple Features](#)
- Jointure des tables avec les tables principales application par l'id_client (SK_ID_CURR) à l'exception de bureau_balance
- **Agrégation** Min, Max, Mean, Sum et Var pour grouper les tables.
- **Création de features** : PAYMENT_RATE, INCOME_CREDIT_PERC, INCOME_PER_PERS ...
- **Discrétisation** des variables catégorielles, **One Hot Encoding** sans création de dummy NA
- Suppression des variables ayant plus de 60% de valeurs manquantes
- Pour les variables numériques, imputation des valeurs manquantes par la médiane

A l'issue de cette étape de preprocessing , nous avons un jeu de donnée final de **307507** clients et **563** variables descriptives .

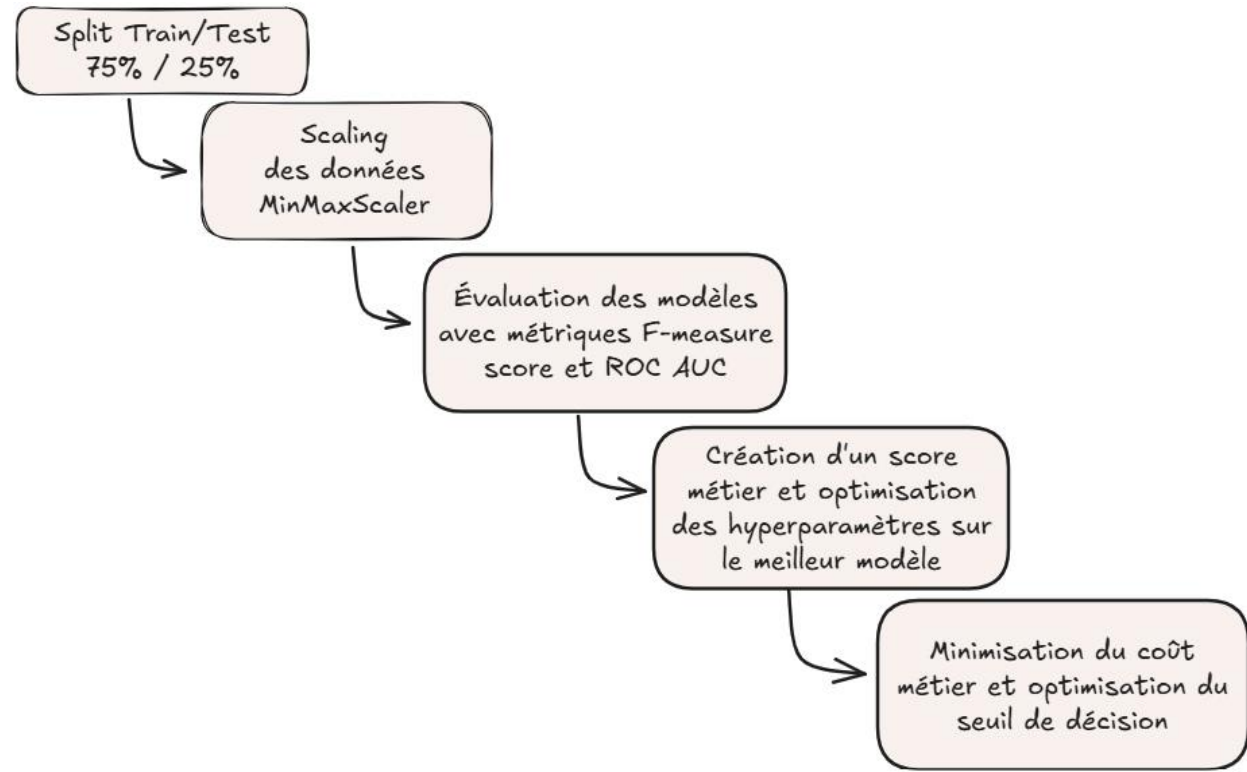


Modélisation

Modélisation

Deux problématiques spécifiques à prendre en compte dans **l'élaboration du modèle** :

- Le **déséquilibre** entre le nombre de **bons** et de **moins bons clients**
- Le **déséquilibre du coût métier** entre un faux négatif (FN - mauvais client prédit bon client : donc crédit accordé et perte en capital) et un faux positif (FP - bon client prédit mauvais : donc refus crédit et manque à gagner en marge)



Scaling des données

- Utilisation de la technique de normalisation **MinMaxScaler** : $y = (x - \min) / (\max - \min)$
- Mise à l'échelle des données dans la plage [0, 1]
- Technique utilisée lorsque les données ne sont pas distribuées normalement ou selon une distribution gaussienne.
- Technique sensible aux outliers mais qui permet de conserver les relations entre les variables.
- On s'assure de le faire après le split train test pour éviter le data leakage.
- Autre alternative possible **StandardScaler** permettant de recalculer les données afin qu'elles aient une moyenne de 0 et une variance de 1.

Choix métriques & implémentation d'un score métier

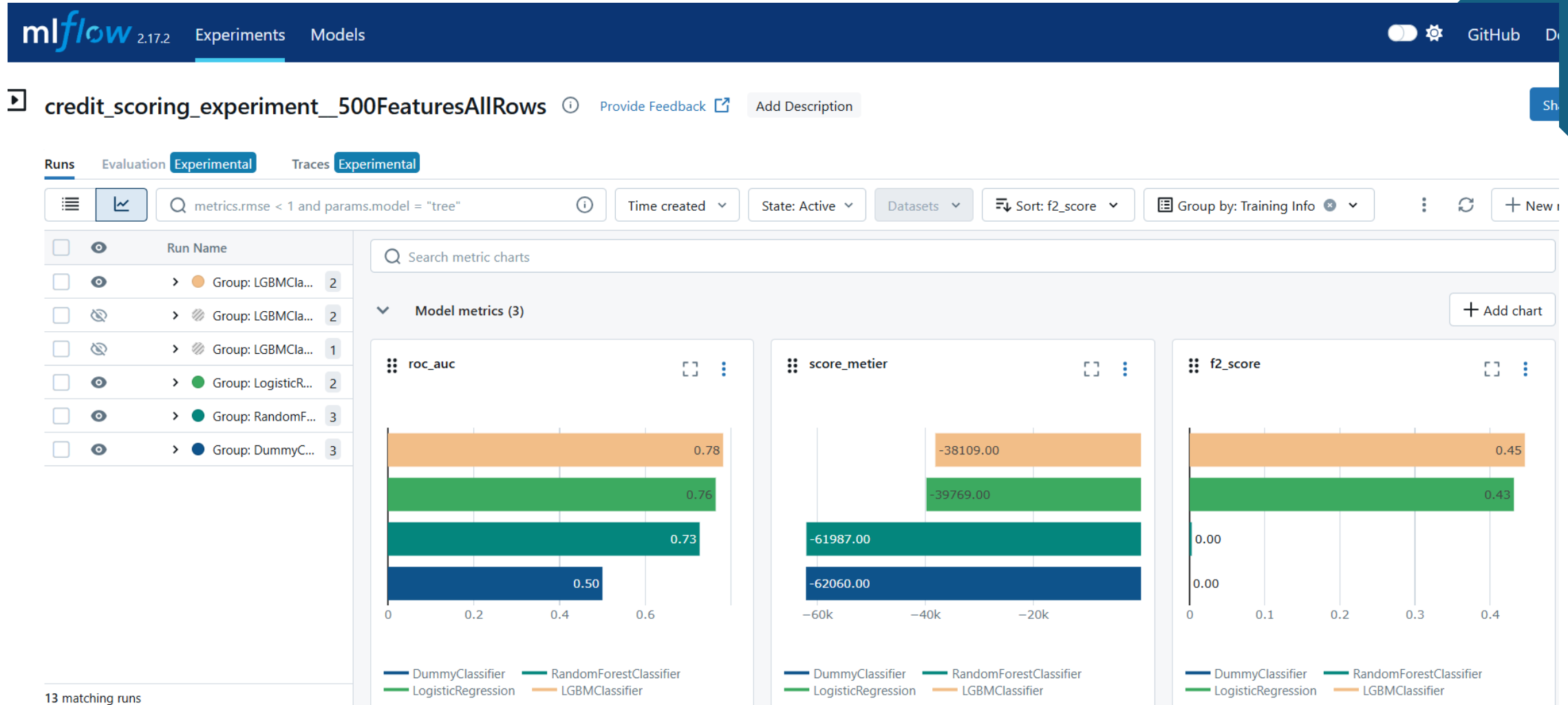
- Evaluation des modèles à l'aide des métriques **ROC AUC**, **F-Beta** et du **score métier**
- **Utilisation F-Beta** avec un paramètre $\beta = 2$, pour accorder un poids plus important sur la sensibilité (FN) plutôt que la précision.

Implémentation d'un **score métier** :

- Le coût d'un Faux Négatif est dix fois supérieur que le coût d'un Faux Positif
- Création d'un **score métier** pour évaluer le modèle : **Coût = 10FN + FP**
- Nous utiliserons ce score métier pour évaluer les modèles et optimiser les hyperparamètres du modèle choisi.

Tracking MLFlow

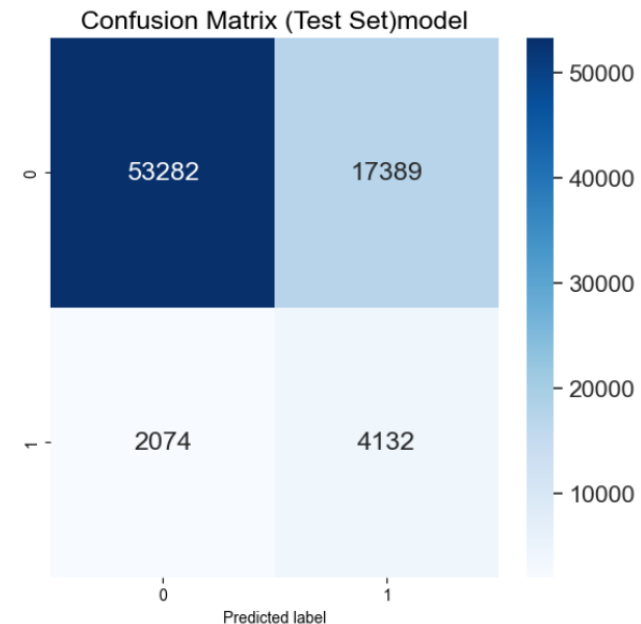
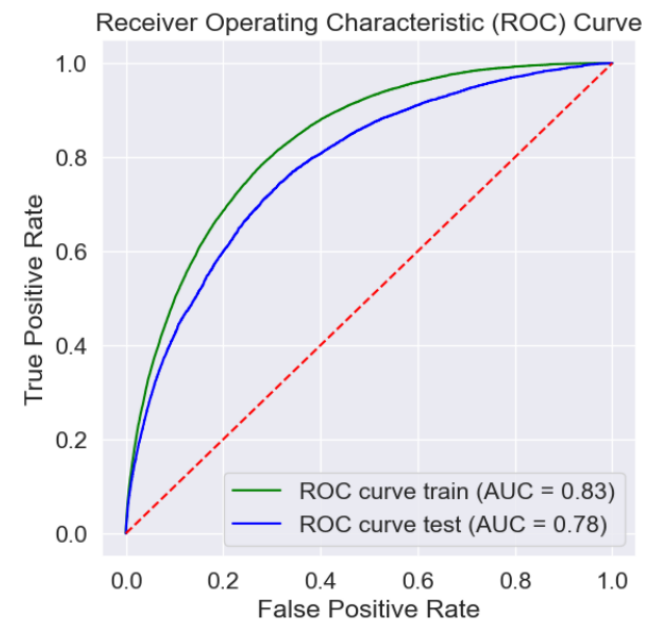
- Tracking des 4 modèles testés et suivi des métriques sur l'UI MLFlow.



Evaluation des Modèles

	model_name	f2_score	accuracy_score	recall	f1_score	roc_auc	score_métier
5	RandomForestClassifier_train	0.999699	0.999970	0.999624	0.999812	1.000000	-70
7	LGBMClassifier_train	0.493764	0.742206	0.762125	0.323106	0.831542	-99316
6	LGBMClassifier_test	0.446295	0.728241	0.691750	0.291268	0.781250	-38109
3	LogisticRegression_train	0.431366	0.699115	0.701380	0.273455	0.768851	-119433
2	LogisticRegression_test	0.431703	0.699741	0.701257	0.273823	0.763920	-39769
4	RandomForestClassifier_test	0.001610	0.919287	0.001289	0.002572	0.726610	-61987
0	DummyClassifier_test	0.000000	0.919274	0.000000	0.000000	0.500000	-62060
1	DummyClassifier_train	0.000000	0.919269	0.000000	0.000000	0.500000	-186190

- Baseline modèle **Dummy Classifier** : F2 score = 0% & ROC_AUC = 50%
- Utilisation de l'hyperparamètre `class_weight='balanced'` pour le rééquilibrage des classes
- Evaluation des métriques sur les échantillons de Train et Test
- **Overfitting** détecté avec l'utilisation du modèle **Random Forest**, un ajustement de la profondeur des arbres pourrait permettre de réduire l'overfitting.
- Le modèle **LightGBM classifieur** ressort comme étant le plus performant d'après les métriques score métier, F2 Score et ROC AUC.



Optimisation de l'algorithme LightGBM

- On tente d'optimiser les hyperparamètres du modèle LightGBM à l'aide de GridSearch pour minimiser le coût métier : **Coût = 10FN + FP**
- Les paramètres testés :
 - 'max_iter': [200, 300] # Nombre maximum d'itération pour que le solver converge
 - 'n_estimators': [500, 1000], # Nombre d'itération de boosting
 - 'learning_rate': [0.01, 0.02] # Rapidité de la descente de gradient
- Les meilleurs paramètres obtenus :

Best Parameters: {'m__learning_rate': 0.02, 'm__max_iter': 300, 'm__n_estimators': 500}
Best Score: -29042.0

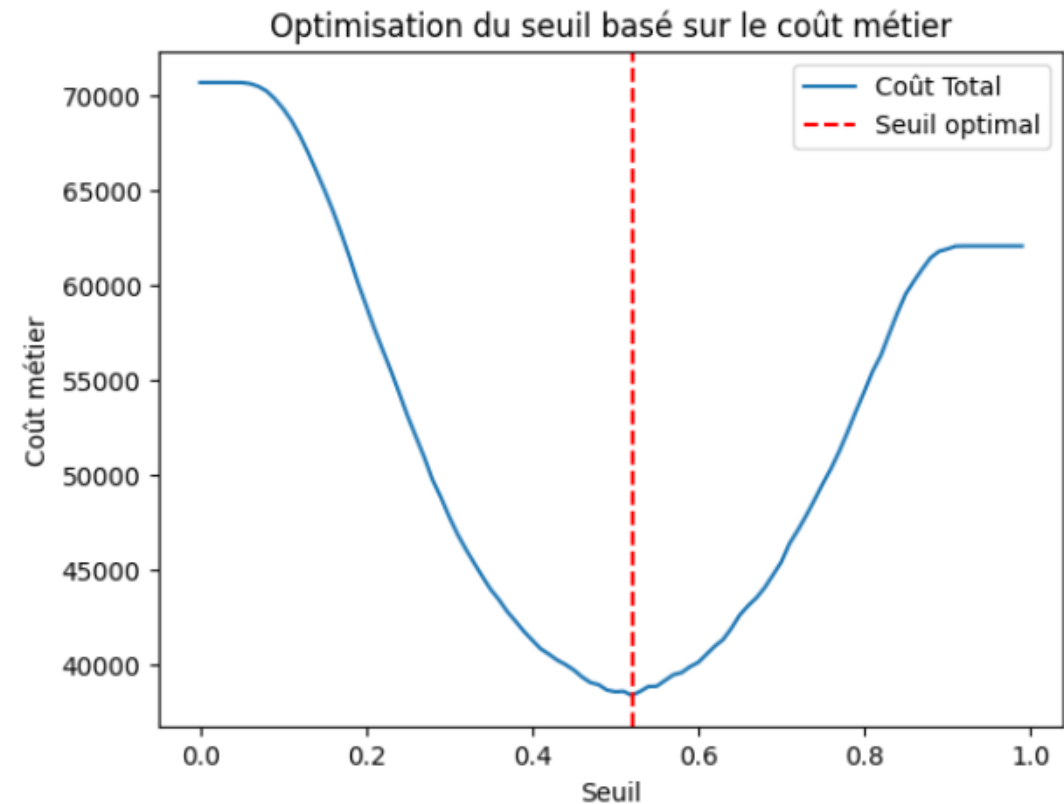
Les résultats obtenus sont sensiblement identiques à ceux sans optimisation des hyperparamètres :



Optimisation du Seuil de décision

- Optimisation du seuil pour minimiser la fonction de coût métier, seuil déterminant la class 0 ou 1 obtenu avec la probabilité résultant du modèle
- Recherche du seuil optimal en testant différents seuils avec un pas de 0,01 entre 0 et 1.

Seuil optimal
minimisant la fonction
de coût métier : **0,52**



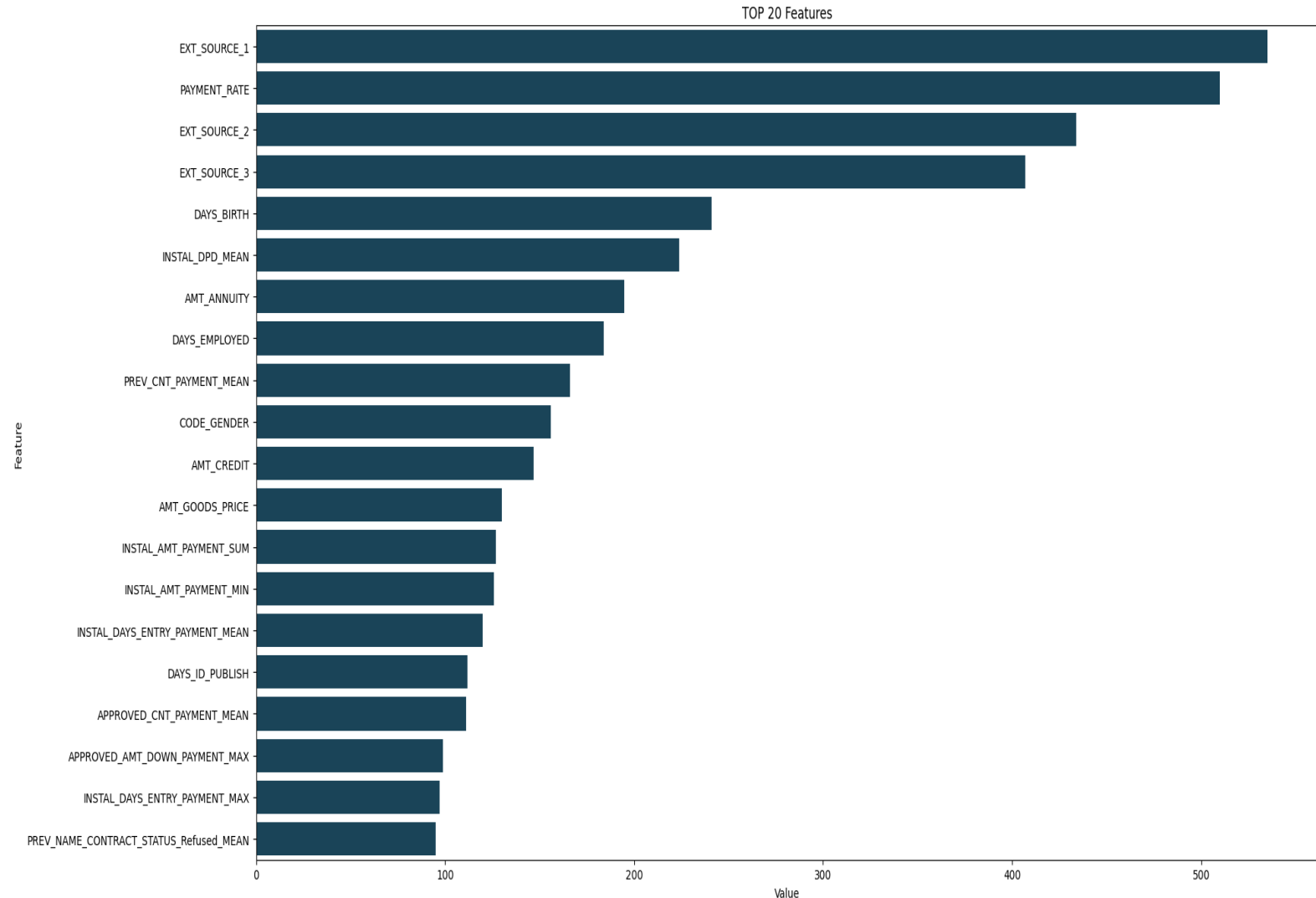


Interprétation Globale & Locale

Comprendre les modèles et les décisions est un enjeu fondamental.

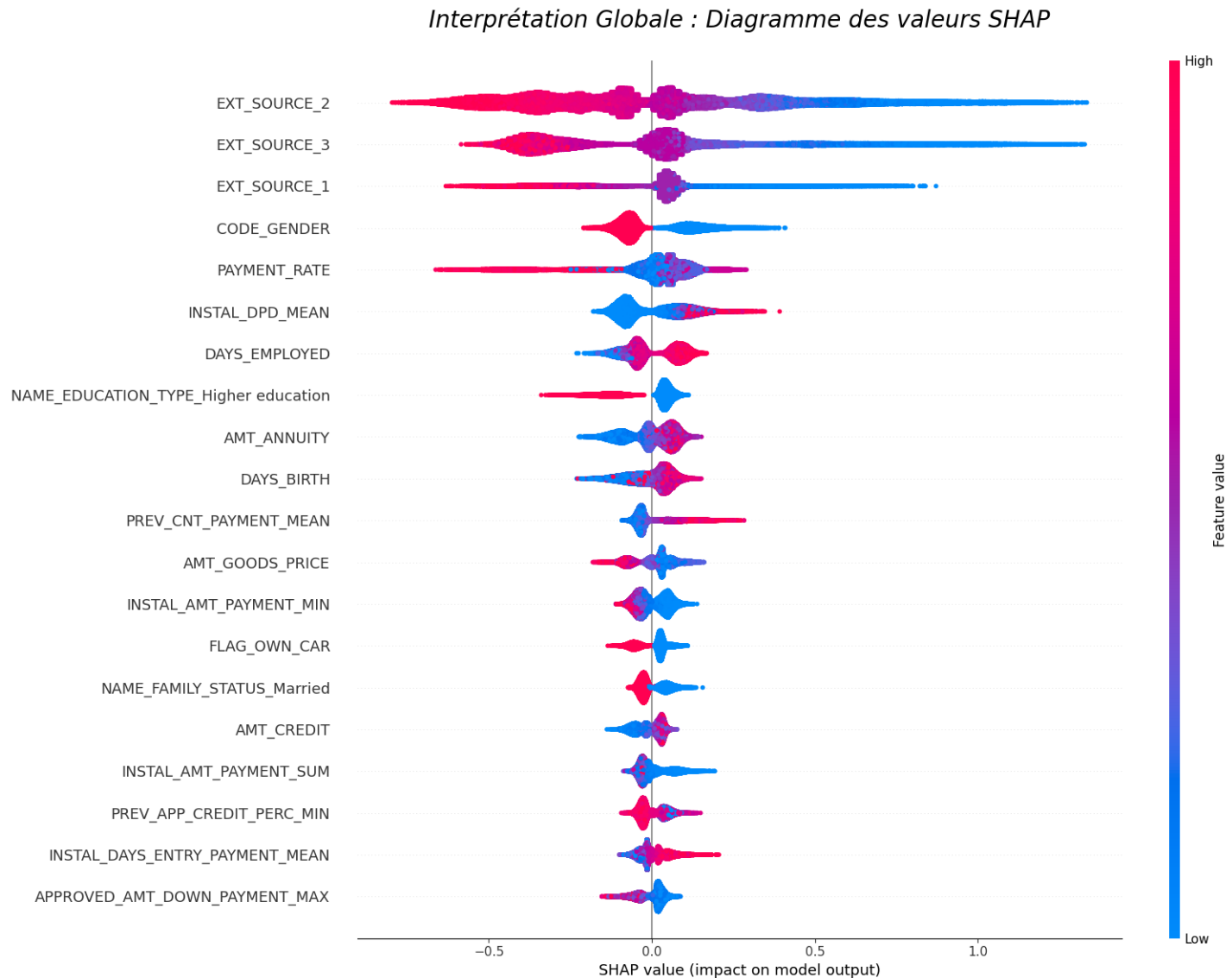
Feature Importance

- Le **top 20** des variables par ordre d'importance obtenu en sortie de l'entraînement du modèle LightGBM avec les meilleures performances
- La méthode de l'importance attribue un score aux données et les classe en fonction des résultats qu'elles ont obtenus
- Les variables EXT_SOURCE_1, PAYMENT_RATE, EXT_SOURCE_2, EXT_SOURCE_3 ressortent comme les plus importantes dans la prédiction du modèle sur les données d'entraînement



Interprétabilité Globale

- Les **valeurs de Shapley** calculent l'impact relatif de chaque variable en comparant ce qu'un modèle prédit avec et sans cette variable
- **Intelligibilité globale** du modèle cherche à expliquer quelles sont les variables les plus importantes en moyenne pour le modèle
- Les points rouges représentent des valeurs élevées de la variable et les points bleus des valeurs basses de la variable
 - **EXT_SOURCE 2** : Scores normalisé à partir d'une source de données externe, plus elle est faible et plus la valeur SHAP est élevé et la probabilité de défaut du client est élevée.
 - **CODE_GENDER** (0 = H, 1=F), plus code_gender est faible (homme) plus la probabilité de défaut prédite est élevée



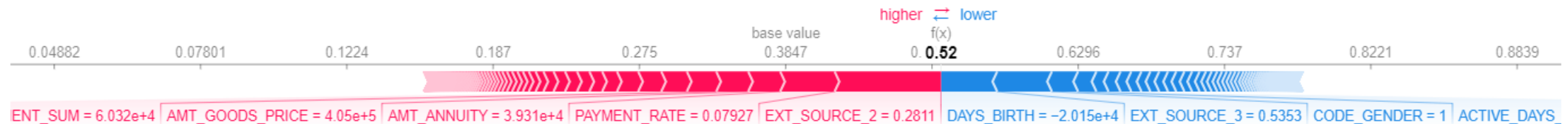
Interprétabilité Locale

- **Intelligibilité locale**, consiste à expliquer la prévision $f(x)$ d'un modèle pour un individu x donné
- En rouge, les variables qui ont un **impact positif** (contribuent à ce que la prédiction soit plus élevée que la valeur de base) et, en bleu, celles ayant un **impact négatif** (contribuent à ce que la prédiction soit plus basse que la valeur de base)

ID Client numero : 176699.0

Prediction : Classe 1.0

Il y a 51.5% de risques que le client ait des difficultés de paiement

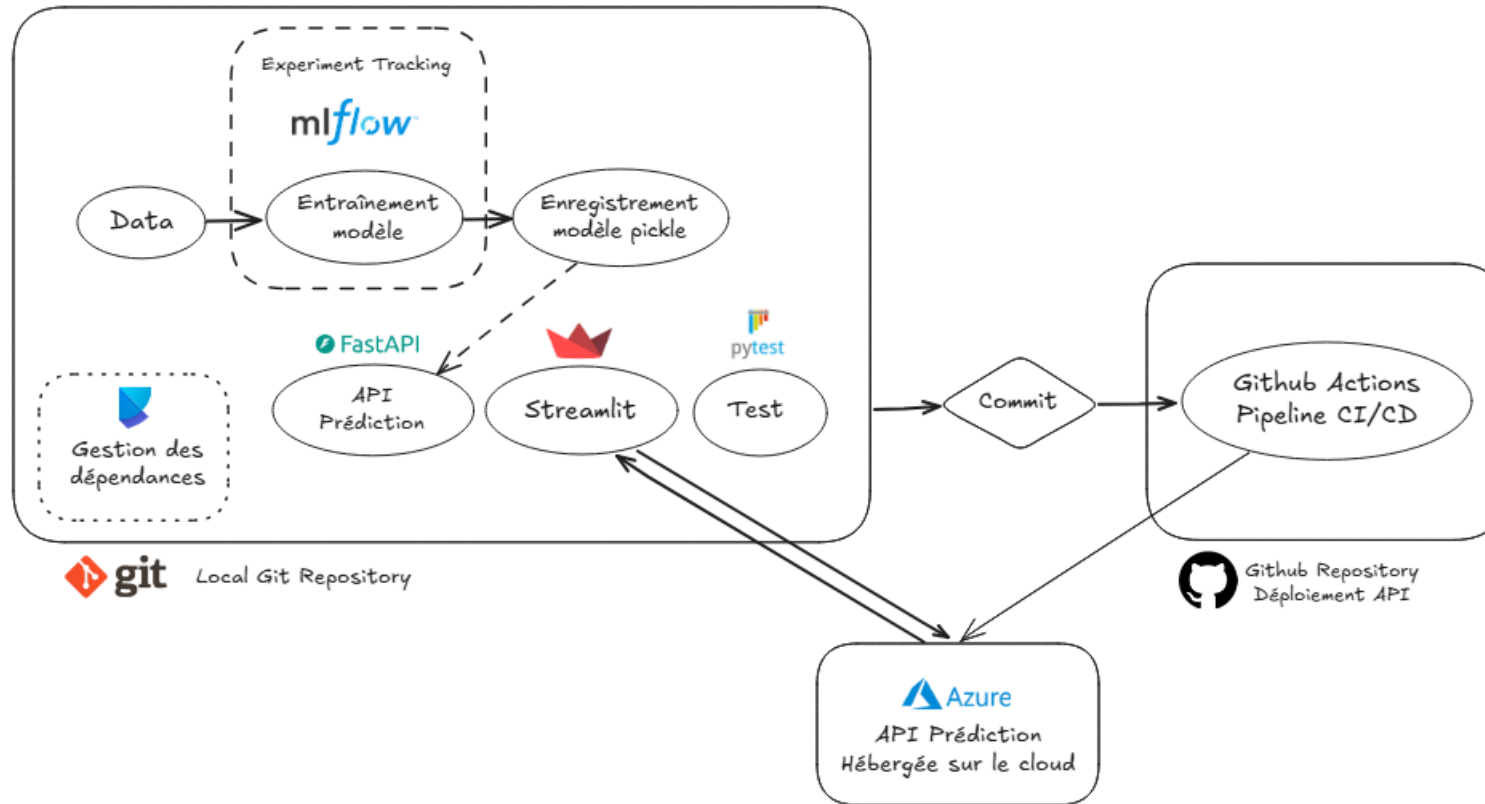


- Le client est prédit **classe 1** avec une **probabilité de défaut de paiement de 51,5%**
- Les variables ayant contribué à augmenter le score sont EXT_SOURCE_2, PAYMENT_RATE, AMT_ANNUITY et AMT_GOODS_PRICES
- L'âge du client, à l'inverse, contribue à diminuer la probabilité de défaut de paiement.
- Ce client se situe au dessus de la moyenne des valeurs de prédiction des clients (base value) qui est à 0,3847.



Pipeline de déploiement

Pipeline de déploiement



Pipeline de déploiement

- Commits dans un repository GitHub
https://github.com/flys-lf/deploiement_api
- Pipeline CI/CD avec GitHub Actions :
 - Build de l'environnement avec Poetry
 - Exécution des tests unitaires
 - Exécution du déploiement de l'API sur Azure

✓ add permissions #6

Summary

Jobs

- ✓ build
- ✓ deploy

Run details

- Usage
- Workflow file

Triggered via push last week
👤 flys-lf pushed → 25a452e **main**

Status
Success

Total duration
10m 42s

Artifacts
1

main_scoringapi.yml
on: push

✓ build

32s

✓ deploy

9m 52s

Commits

main

All users All time

Commits on Dec 19, 2024

change API URL	France LY committed last week - ✓ 2 / 2	a438708	<>
add permissions	France LY committed last week - ✓ 2 / 2	25a452e	<>
fix deployment	France LY committed last week - ✗ 1 / 2	7a4604e	<>
test deploy	France LY committed last week - ✗ 1 / 2	390cb0f	<>
add test_data_sample	France LY committed last week - ✗ 1 / 2	bd08982	<>
add yml	France LY committed last week - ✗ 0 / 2	8de6a4e	<>
Add or update the Azure App Service build and deployment workflow config	flys-lf committed last week - ✗ 0 / 2	f94627b	<>

n Dec 18, 2024

build

succeeded last week in 32s

- > ✓ Set up job
- > ✓ Run actions/checkout@v4
- > ✓ Set up Python version
- > ✓ Install Poetry
- > ✓ Install dependencies with Poetry
- > ✓ Run pytest
- > ✓ Zip artifact for deployment
- > ✓ Upload artifact for deployment jobs
- > ✓ Post Set up Python version
- > ✓ Post Run actions/checkout@v4
- > ✓ Complete job

deploy

succeeded last week in 9m 52s

- > ✓ Set up job
- > ✓ Download artifact from build job
- > ✓ Unzip artifact for deployment
- > ✓ Login to Azure
- > ✓ Deploy to Azure Web App
- > ✓ Post Login to Azure
- > ✓ Complete job

Exemple scoring client API

- API de prédiction déployé sur le Cloud avec Microsoft Azure AppService
- URL API : <https://scoringapi-ewckf3cxfrdbadhw.northeurope-01.azurewebsites.net/>
- Appel de l'API sur le cloud pour la prédiction du scoring client via un dashboard Streamlit

scoringapi-ewckf3cxfrdbadhw.northeurope-01.azurewebsites.net

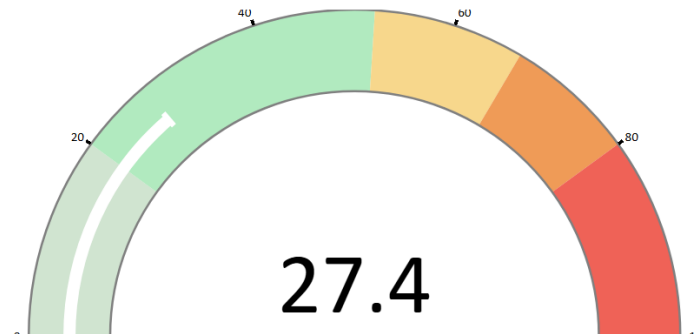
Impression élégante ☒

```
{  
  "message": "Credit Scoring API"  
}
```

- Exemple de scoring client via appel d'API :

Le client N° 100001 a une probabilité de défaut de paiement estimé à : 27.39%

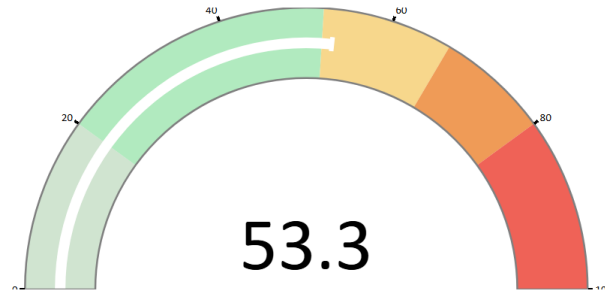
Crédit Accordé



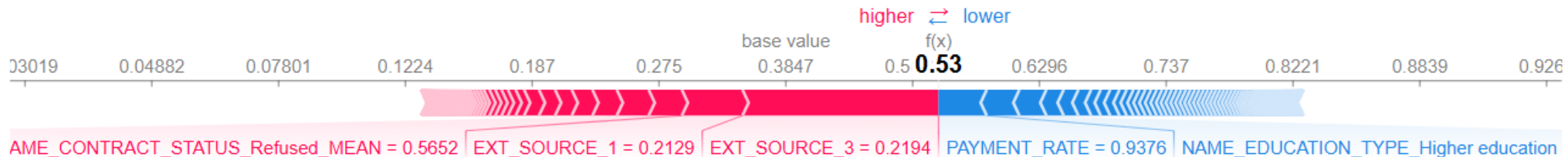
Analyse d'un cas de refus de crédit

Le client N° 100067 a une probabilité de défaut de paiement estimé à : 53.31%

Crédit Refusé



Explication Locale Client N°100067



- Le client est prédit **classe 1** avec une **probabilité de défaut de paiement de 53,3%**
- Les variables ayant contribuées à augmenter le score sont EXT_SOURCE_3, EXT_SOURCE_1, PREV_NAME_CONTRACT_STATUS_Refused_MEAN
- Le Taux de paiement et le fait que le client ait fait des études supérieures, à l'inverse, contribuent à diminuer la probabilité de défaut de paiement.
- Ce client se situe au dessus de la moyenne des valeurs de prédiction des clients (base value) qui est à 0,3847.



Data Drift

Data Drift

- Phénomène du « **Data Drift** » : une fois le modèle de prédiction mis en production, il est crucial de surveiller que les données ne dérivent pas par rapport aux données d'entraînement
- Ce problème doit être détecté et anticipé, car il dégrade les performances de prédiction au fur et à mesure du temps
- Calcul du data drift sur un nouveau jeu de données et sur les **100 features les plus importantes** avec evidently
- **14%** de colonnes avec un drift détecté selon la **distance de Wasserstein** qui mesure la distance entre deux distributions de probabilités sur un espace mathématique donné.

Dataset Drift

Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5

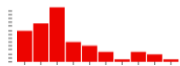
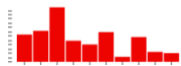


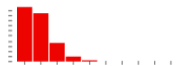

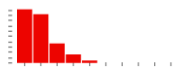
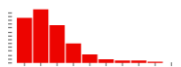




100
Columns

14
Drifted Columns

0.14
Share of Drifted Columns

Drift is detected for 14.0% of columns (14 out of 100).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
> PAYMENT_RATE	num			Detected	Wasserstein distance (normed)	0.574751
> AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (normed)	0.339409
> AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.210686
> AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207339
> INCOME_CREDIT_PERC	num			Detected	Wasserstein distance (normed)	0.179299
> AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.160954



Limites & Améliorations

Limites & Améliorations

- Une **sélection de features plus poussée et fine** en collaboration avec les métiers qui ont une connaissance opérationnelle serait nécessaire pour bien comprendre l'impact de chaque variable et affiner la pertinence du modèle
- Cela permettrait également de réduire le nombre de variables, car en pratique l'utilisation de 563 variables est complexe pour un conseiller qui utiliserait l'outil de scoring
- Axe d'amélioration du **dashboard** : L'utilisateur devrait pouvoir ajouter un client à la volée pour accéder à un scoring client, mais cela impliquerait que toutes les données utilisées pendant l'entraînement soient disponibles pour le nouveau client

Merci

