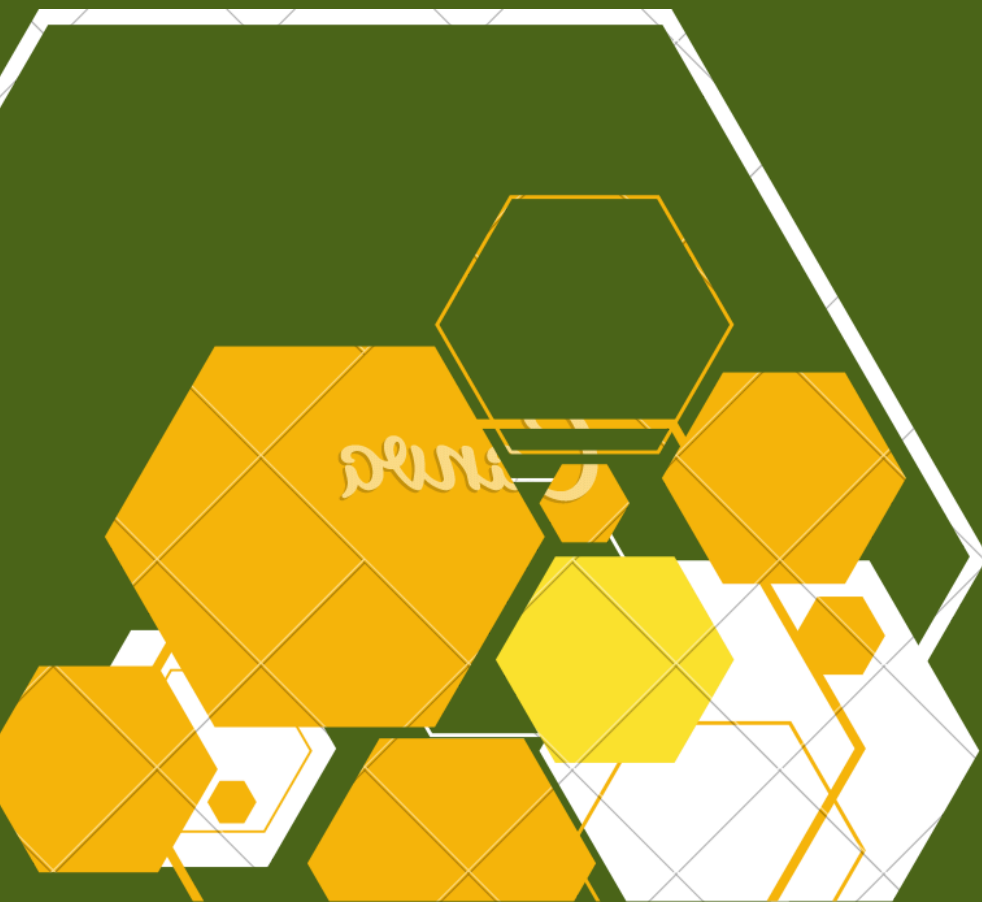


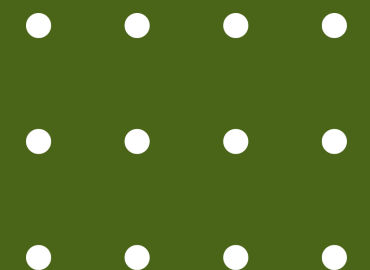
Prêt à dépenser

Implémenter un modèle de
scoring

<https://github.com/tkappe/scoringp7.git>



Présenté par Thierry KAPPE



Plan



01

Contexte, objectifs et données

02

Modélisation

03

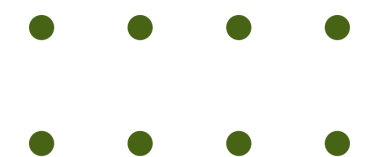
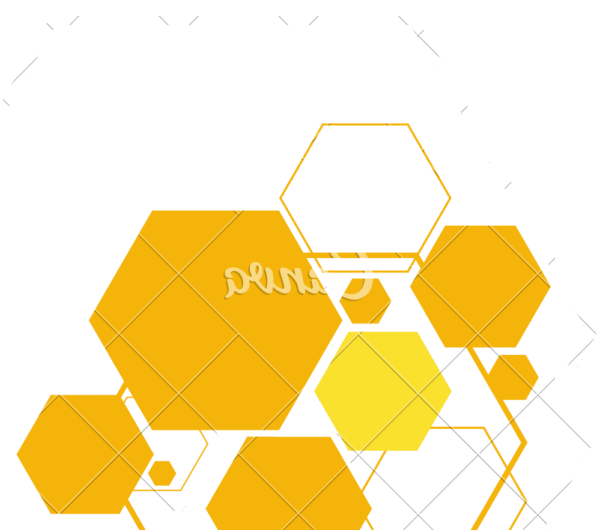
Pipeline de déploiement

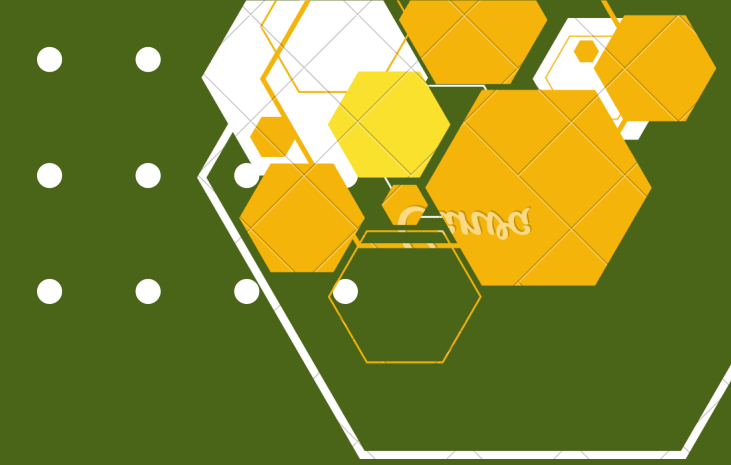
04

API de prédiction

05

Analyse data drift





Contexte et objectifs

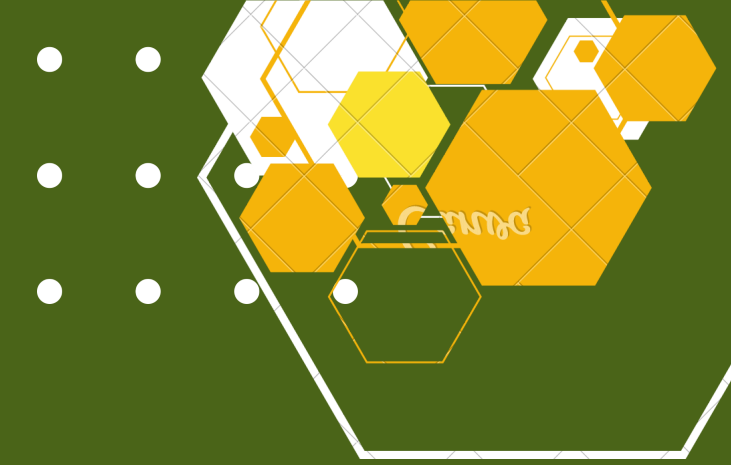


La société financière Prêt à dépenser propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

L'entreprise souhaite mettre en œuvre un outil de scoring crédit qui calcule la probabilité qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. ceci passera par:

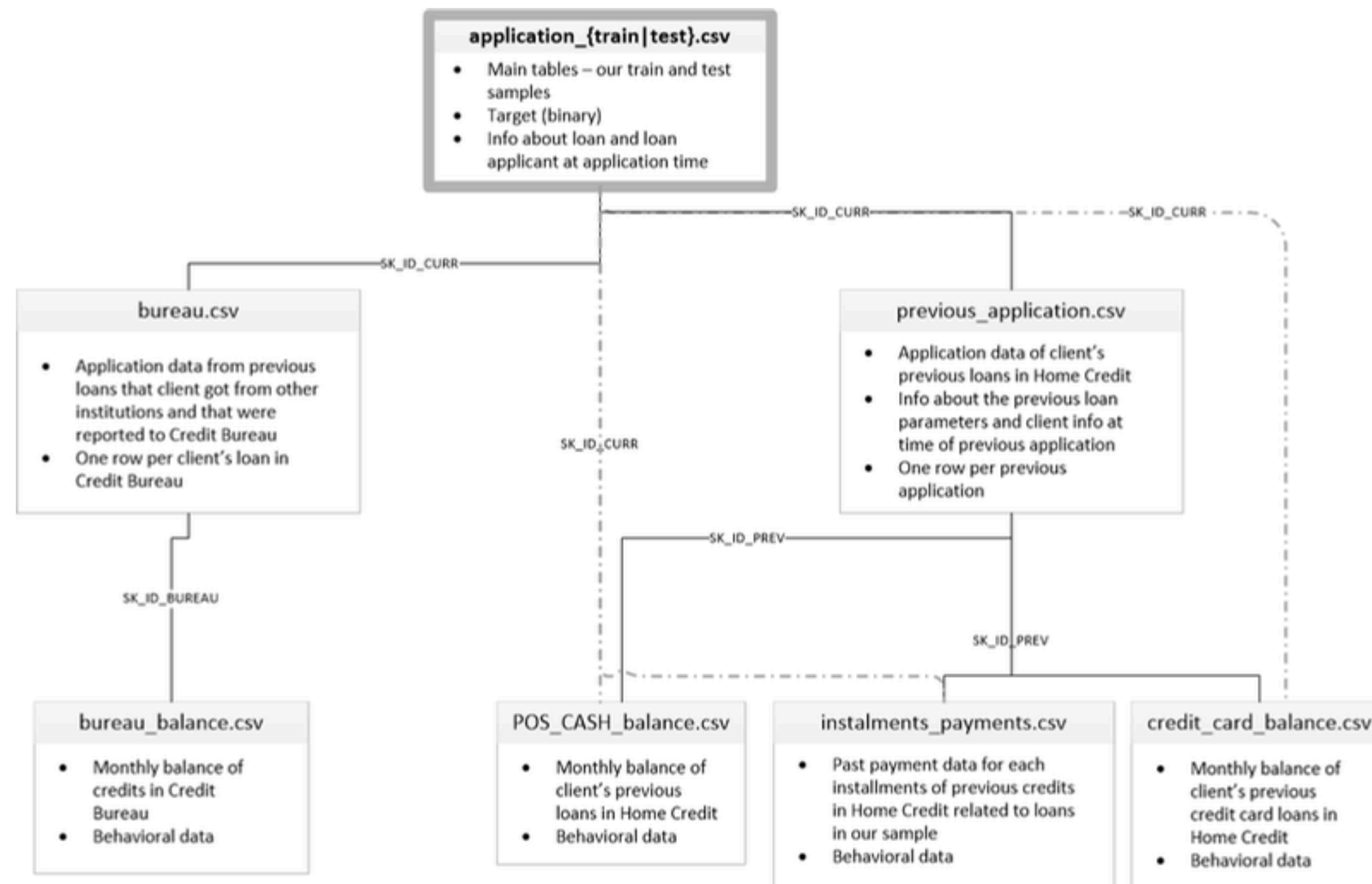
- La construction d'un modèle de prédiction du score client;
- L'analyse des features (contribution globale et locale);
- Mise en production du modèle via une API;
- Mise en œuvre d'une approche globale MLOps (tracking, analyse data drift).

Les données originales sont téléchargeables sur Kaggle [ici](#)



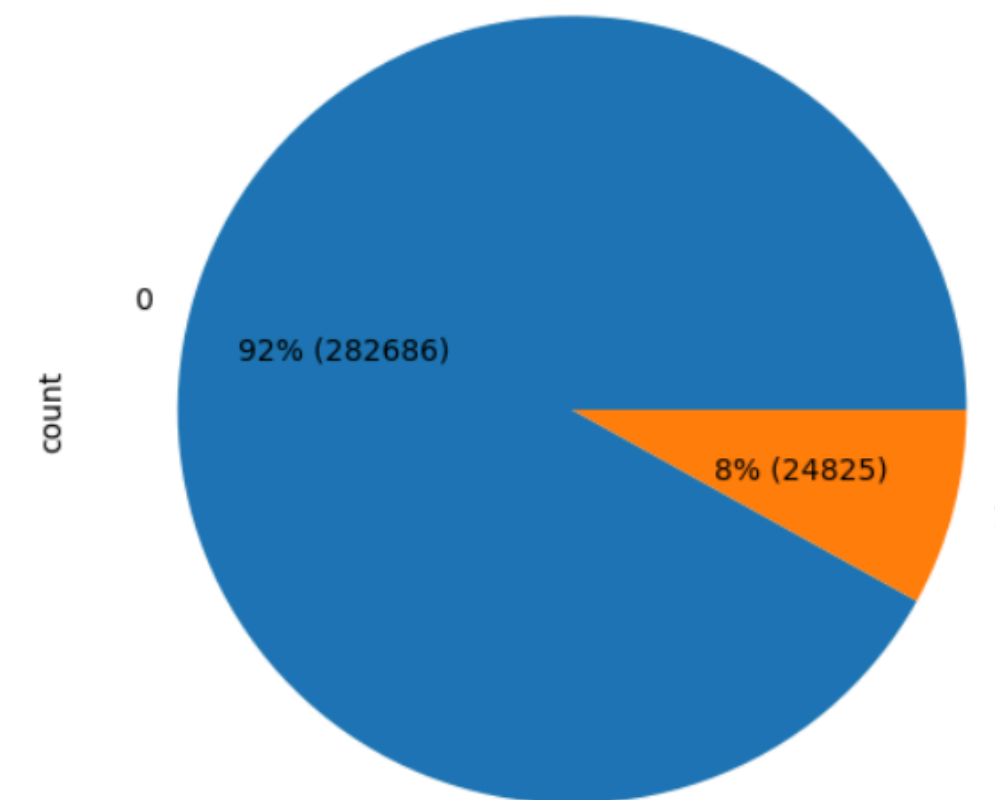
Les données

- 7 fichiers pour environ 200 variables

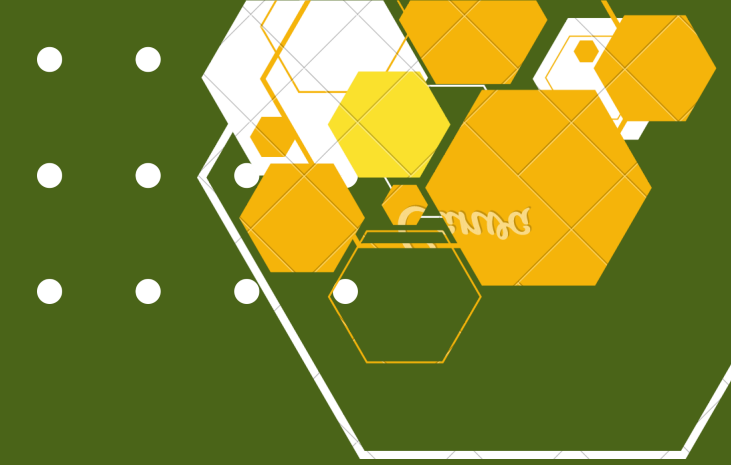


- Application "train" regroupe 307 511 clients dont on connaît la décision de « Prêt à Dépenser » sur l'octroi du crédit (variable "Target")

Répartition des étiquettes de la variable cible



- Application "test" regroupe 48 744 clients dont on ne connaît pas cette décision.



Features engineering

Nous nous appuyons sur le notebook d'exemple fourni [ici](#)

Jointure des fichiers de données

Suppression des variables ayant plus de 60% de valeurs manquantes

Création de nouvelles variables (ratio)

Encodage de variables catégorielles

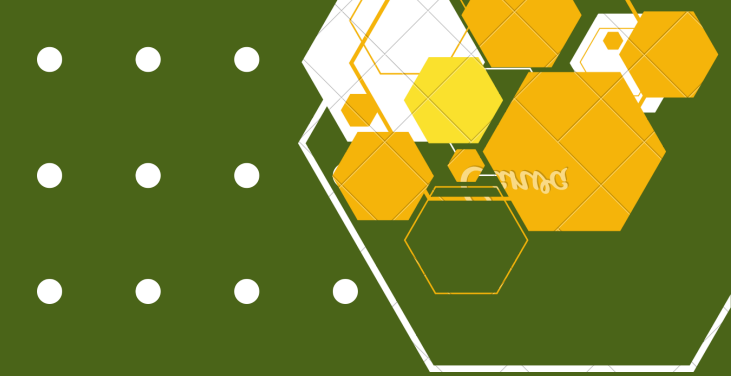
Agrégation (somme, moyenne, max,...)

Imputation des valeurs manquantes
SimpleImputer

Standardisation des données
MinMaxScaler

training data
(307507, 564)

testing data
(48744, 563)



Choix métrique d'évaluation et modèle

Train / Test Split : 80% / 20%

Chaque sous-échantillon contient le même mélange d'exemples par classe, c'est-à-dire environ 92% de classe 0 et 8% de classe 1

Évaluation des modèles candidats à l'aide d'une validation croisée stratifiée répétée k-fold.

Équilibrage des classes

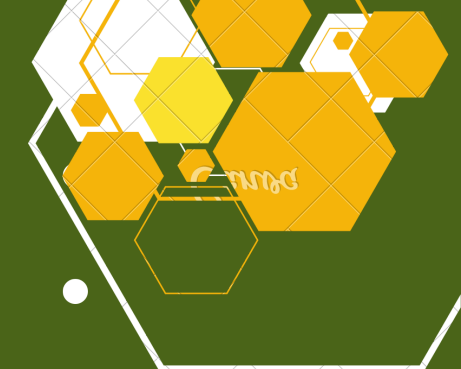
SMOTE Imblearn.over_sampling

Evaluation des modèles de base et choix du modèle à optimiser
minimisation du score métier
maximisation de l'AUC

Problématique : Classification sur des données déséquilibrées

La difficulté ici réside dans le fait que les faux négatifs sont plus dommageables que les faux positifs. Les faux négatifs sur cet ensemble de données sont des cas où un **mauvais client est marqué comme un bon client et se voit accorder un prêt** alors que les faux positifs sont des cas où un bon client est marqué comme un mauvais client et la société de crédit ne lui accorde pas de prêt.

	Model	Business score	Accuracy	Precision	Recall	F-1 score	AUC score
0	Dummy	0.857208	0.502488	0.500933	0.503148	0.395134	0.503148
1	LogisticRegression	0.540552	0.706465	0.563048	0.684964	0.541158	0.753695
2	RandomForest	0.729505	0.899304	0.592370	0.549633	0.560817	0.718714
3	LightGBM	0.720871	0.912767	0.652074	0.553916	0.571435	0.766116



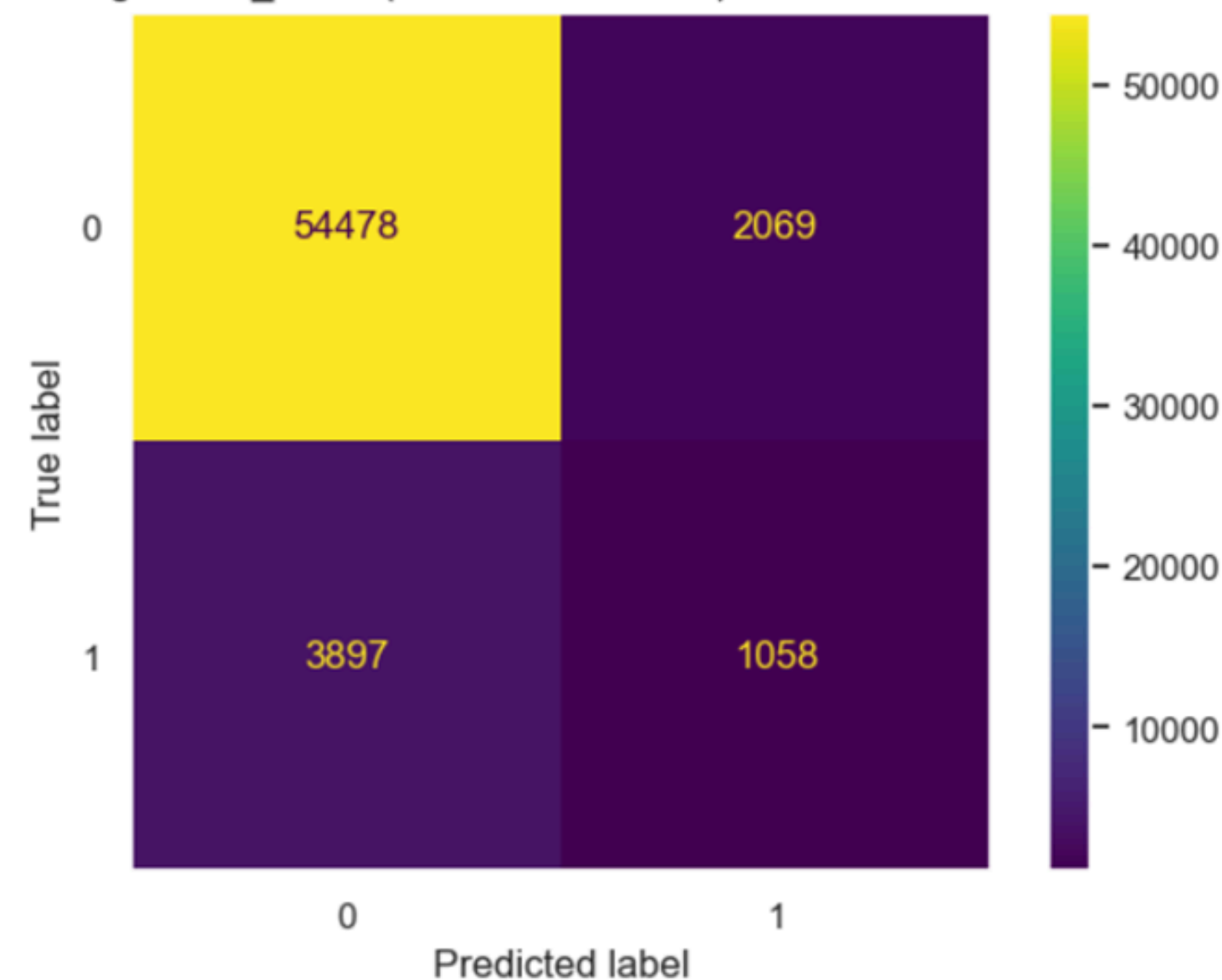
Optimisation du modèle *LightGBM*

```
lgbmt_space = {'max_depth': scope.int(hp.quniform("max_depth", 3, 11, 1)),
               'subsample': hp.uniform('subsample', 0.5, 1.),
               'colsample_bytree': hp.quniform('colsample_bytree', 0.3, 1, 0.1),
               'learning_rate': hp.quniform('learning_rate', 0.1, 1, 0.1),
               'min_child_samples' : scope.int(hp.quniform('min_child_weight', 0, 200, 50)),
               'n_estimators': scope.int(hp.quniform('n_estimators', 50, 500, 50)),
               'num_leaves': scope.int(hp.quniform("num_leaves", 5, 13, 2)),
               'min_data_in_leaf': scope.int(hp.quniform('min_data_in_leaf', 100, 300, 50)),
               'reg_alpha': hp.quniform('reg_alpha', 0.1, 1, 0.1),
               'reg_lambda': hp.quniform('reg_lambda', 0, 1, 0.2)}
```

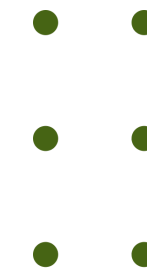
```
lgbmt_best_model.get_params
```

```
<bound method LGBMModel.get_params of LGBMClassifier(colsample_bytree=0.7000000000000001,
               learning_rate=0.6000000000000001, max_depth=9,
               min_child_samples=200, min_data_in_leaf=150, n_estimators=450,
               num_leaves=6, random_state=123, reg_alpha=0.4, reg_lambda=0.8,
               subsample=0.7120936420086237)>
```

LightGBM_tuned (standard threshold): Confusion matrix

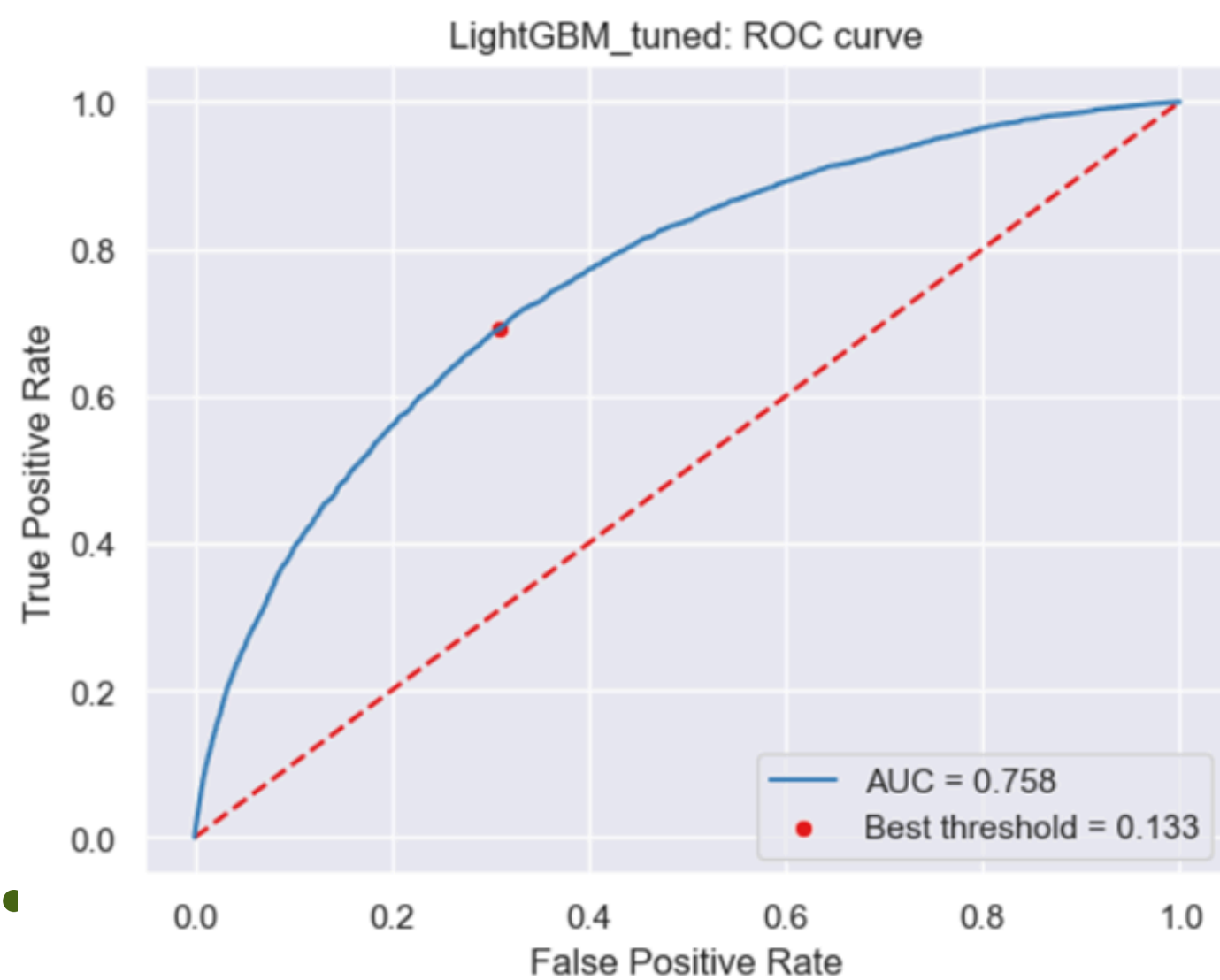


● ● *Obtenu par validation croisée en minimisant le score métier*

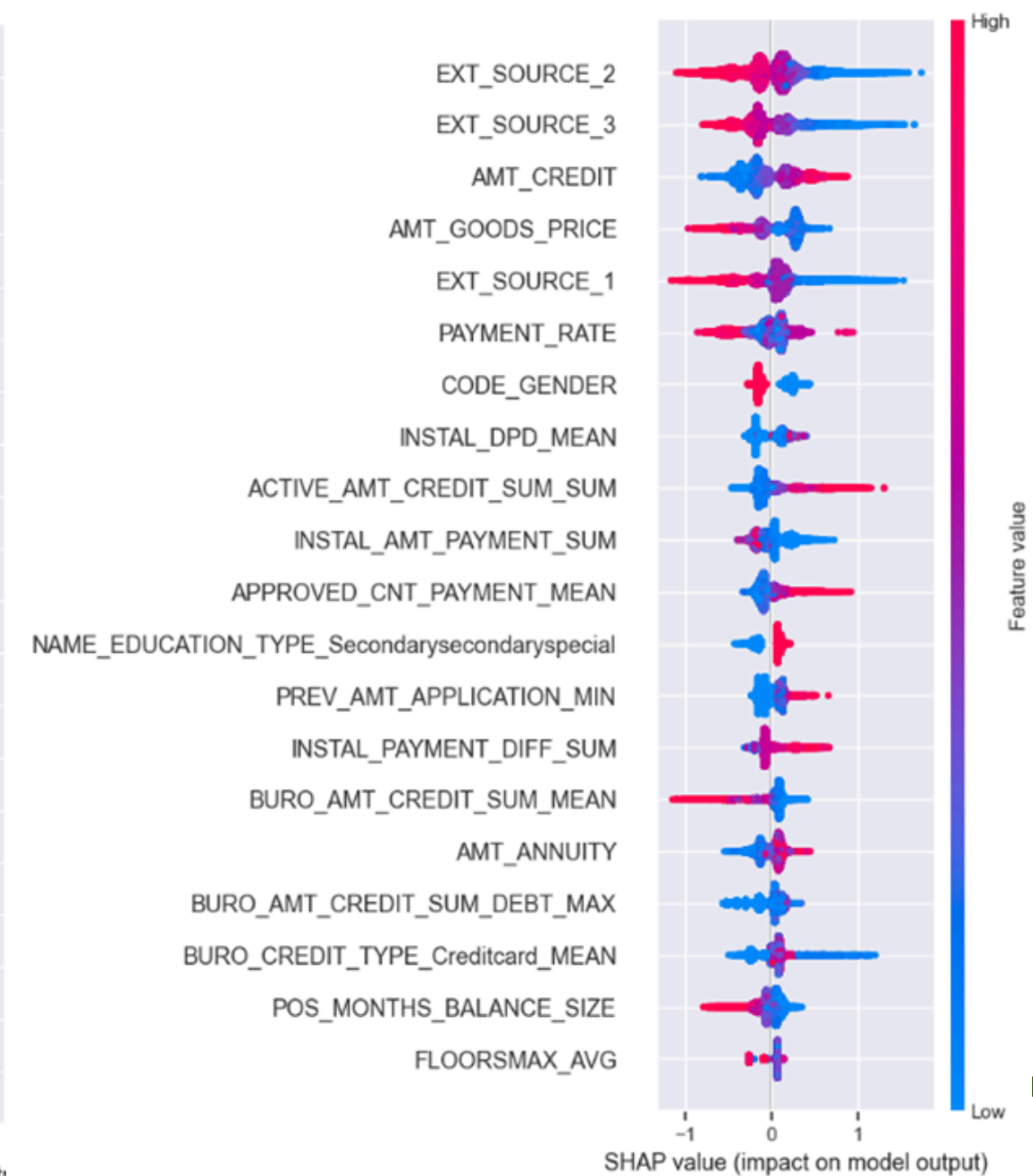
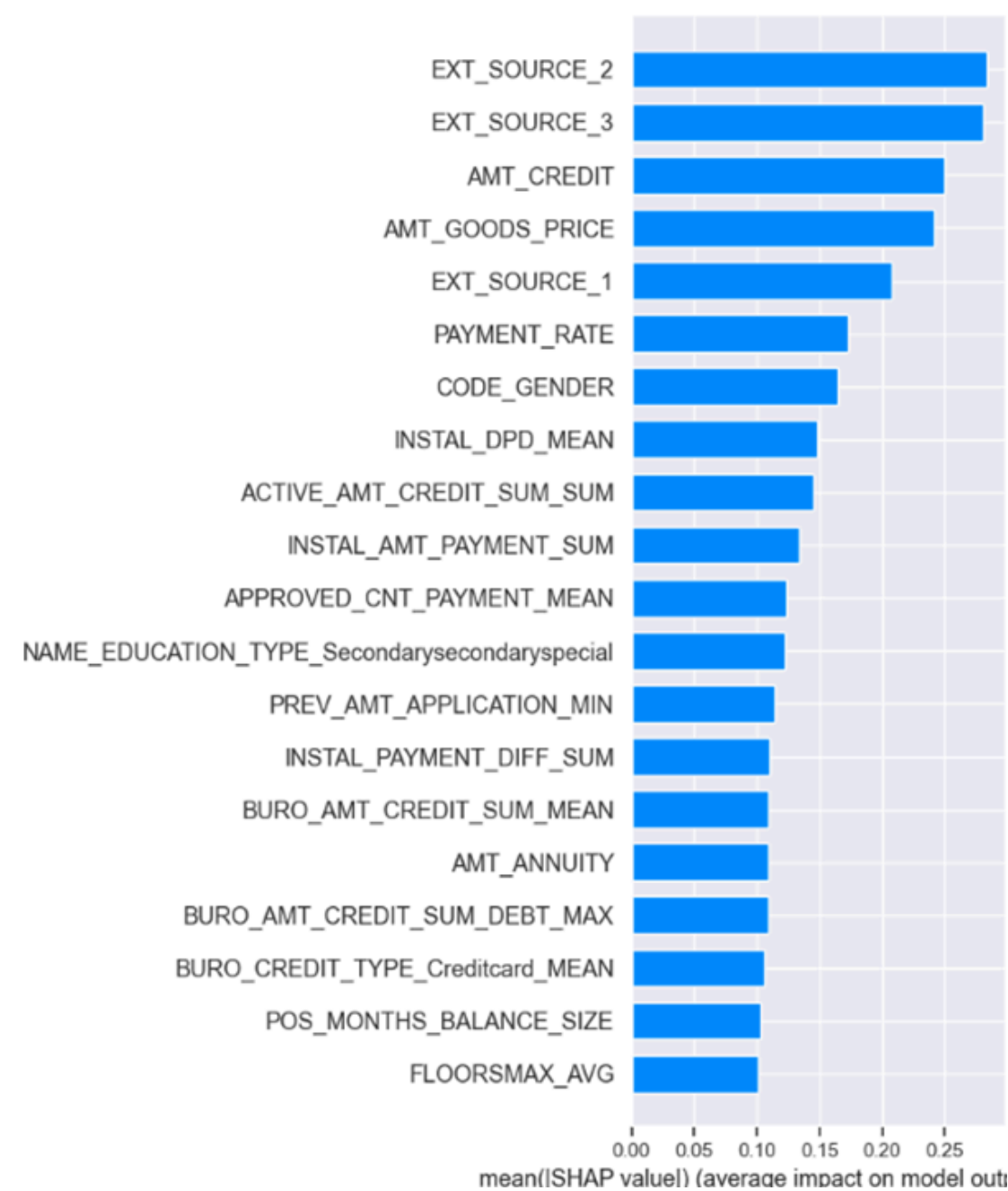




Courbe ROC et Seuil optimal de classification

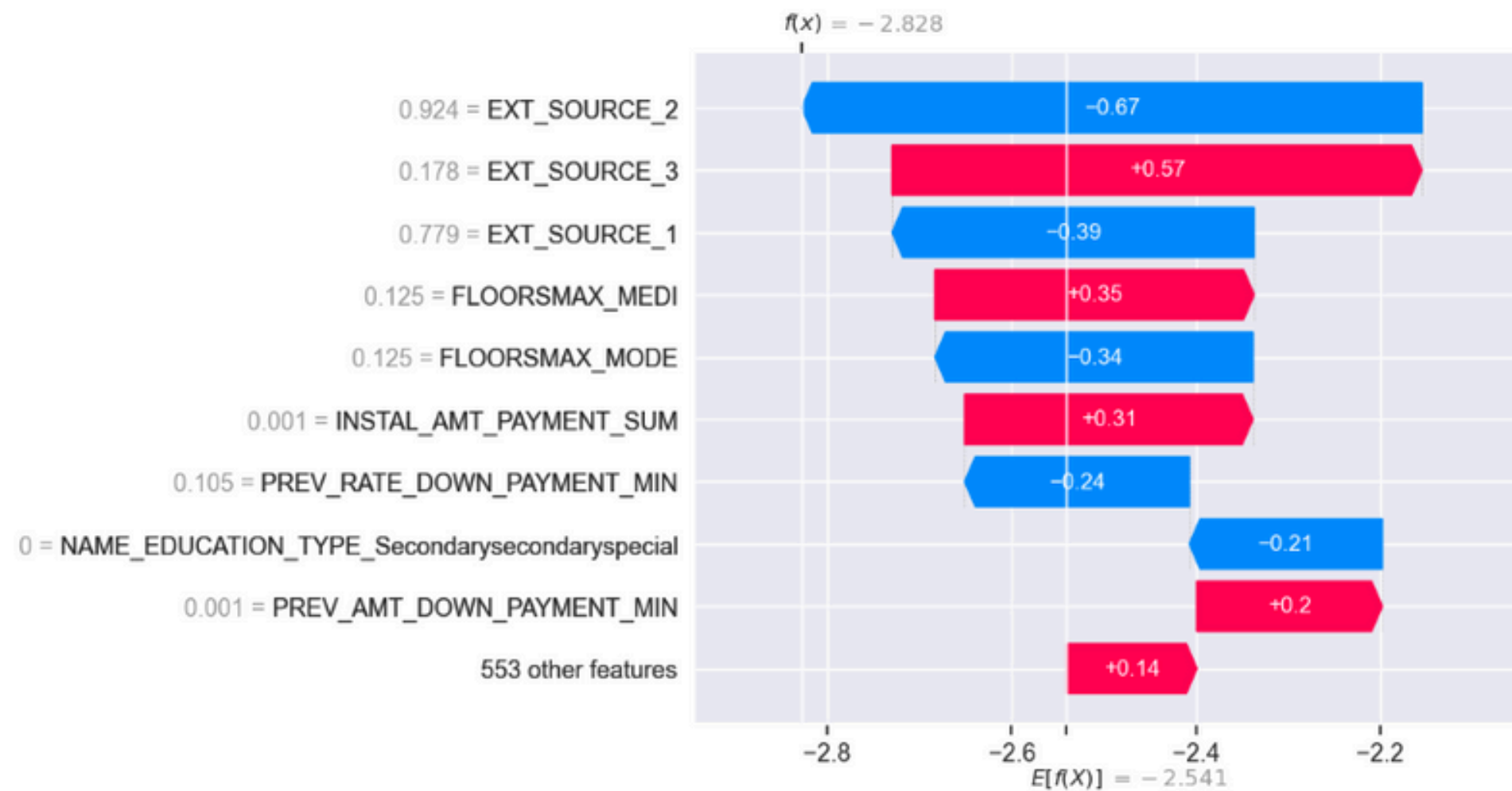


Interprétation globale

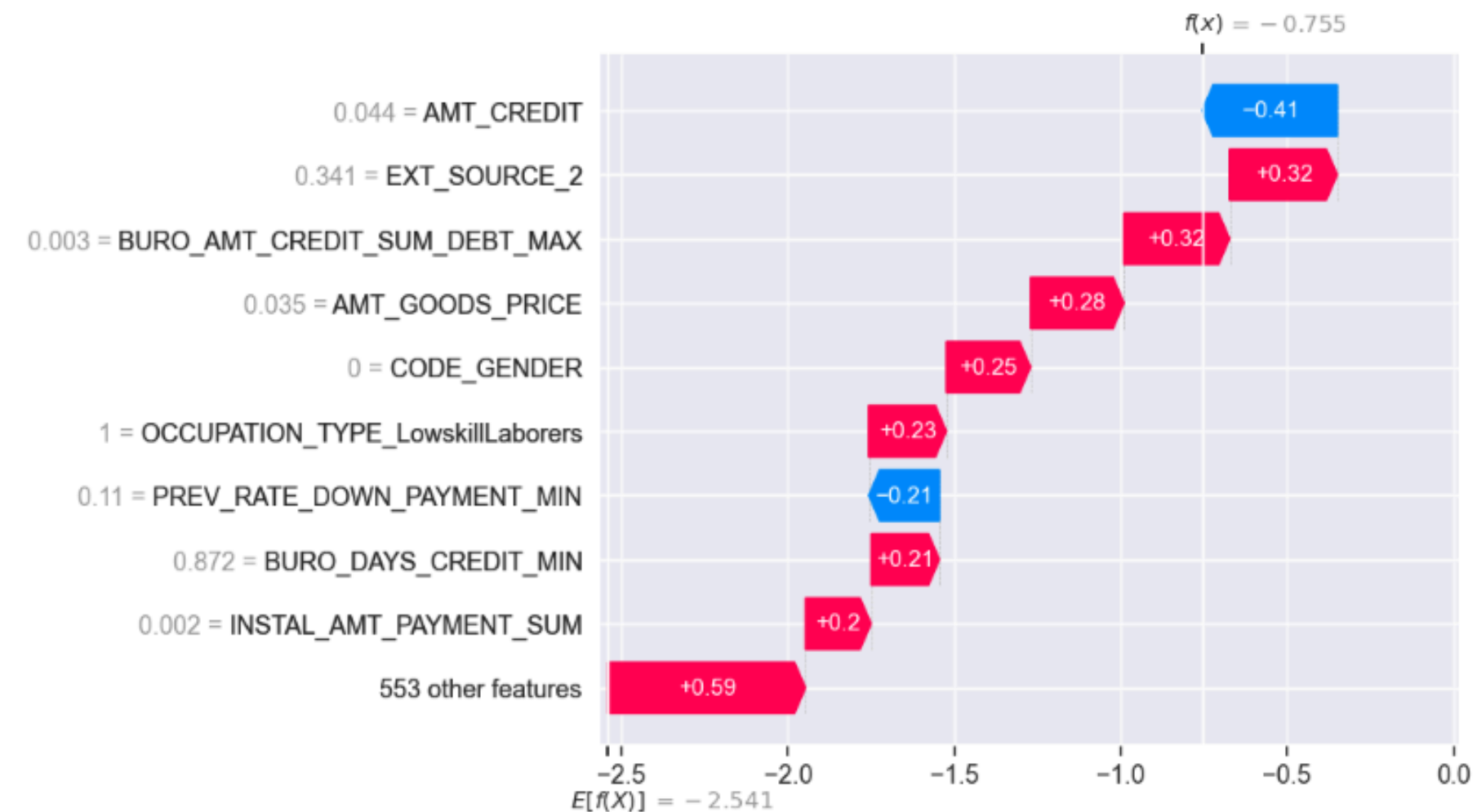


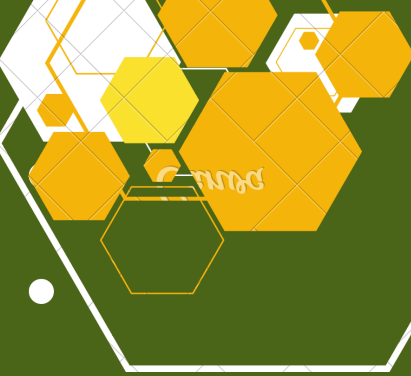
Interprétation locale

- Cas du client d'indice 0, client numéro 100001

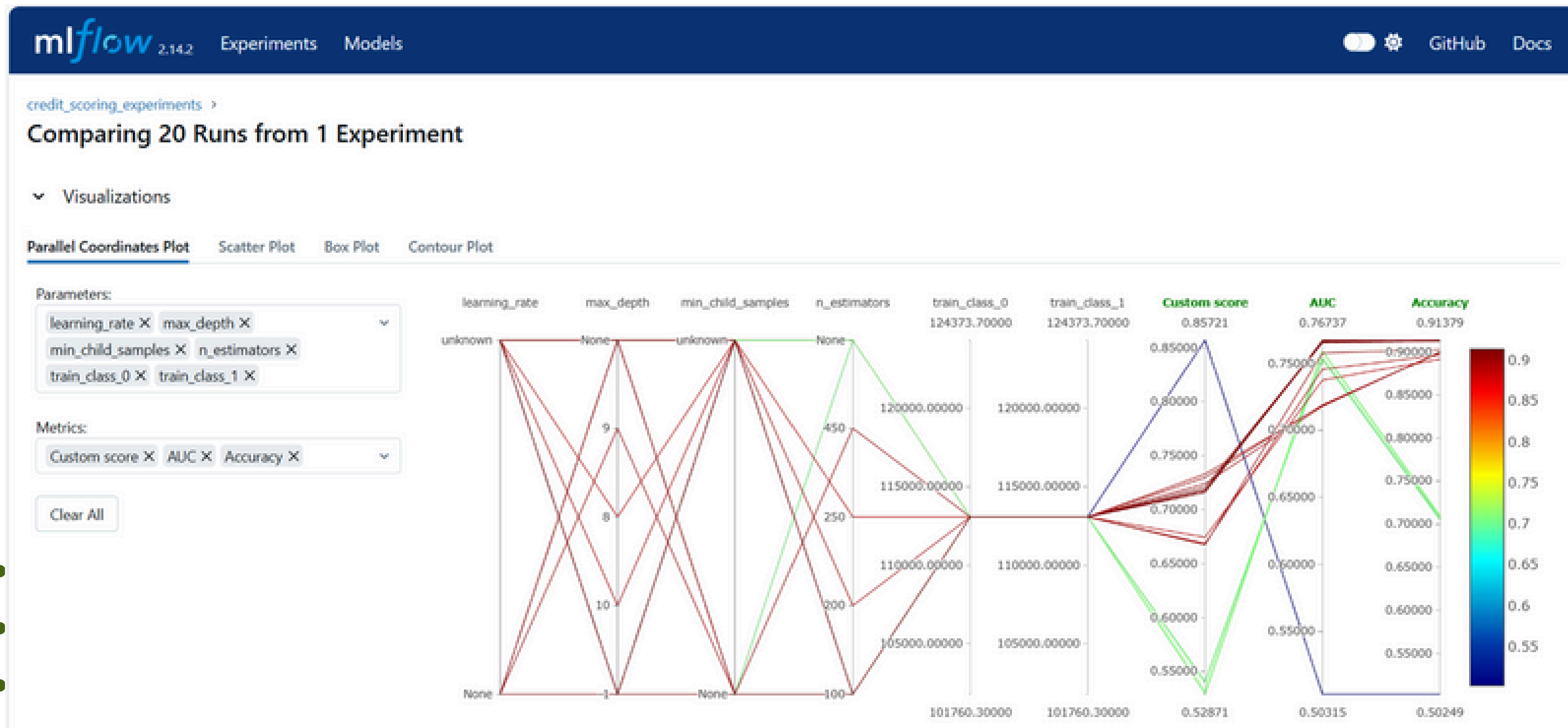


- Cas du client d'indice 1, client numéro 100005





Tracking des expériences : présentation de l'interface ML Flow UI



03

Pipeline de déploiement

Présentation de Git, GitHub et test unitaires : [Repo GitHub](#)

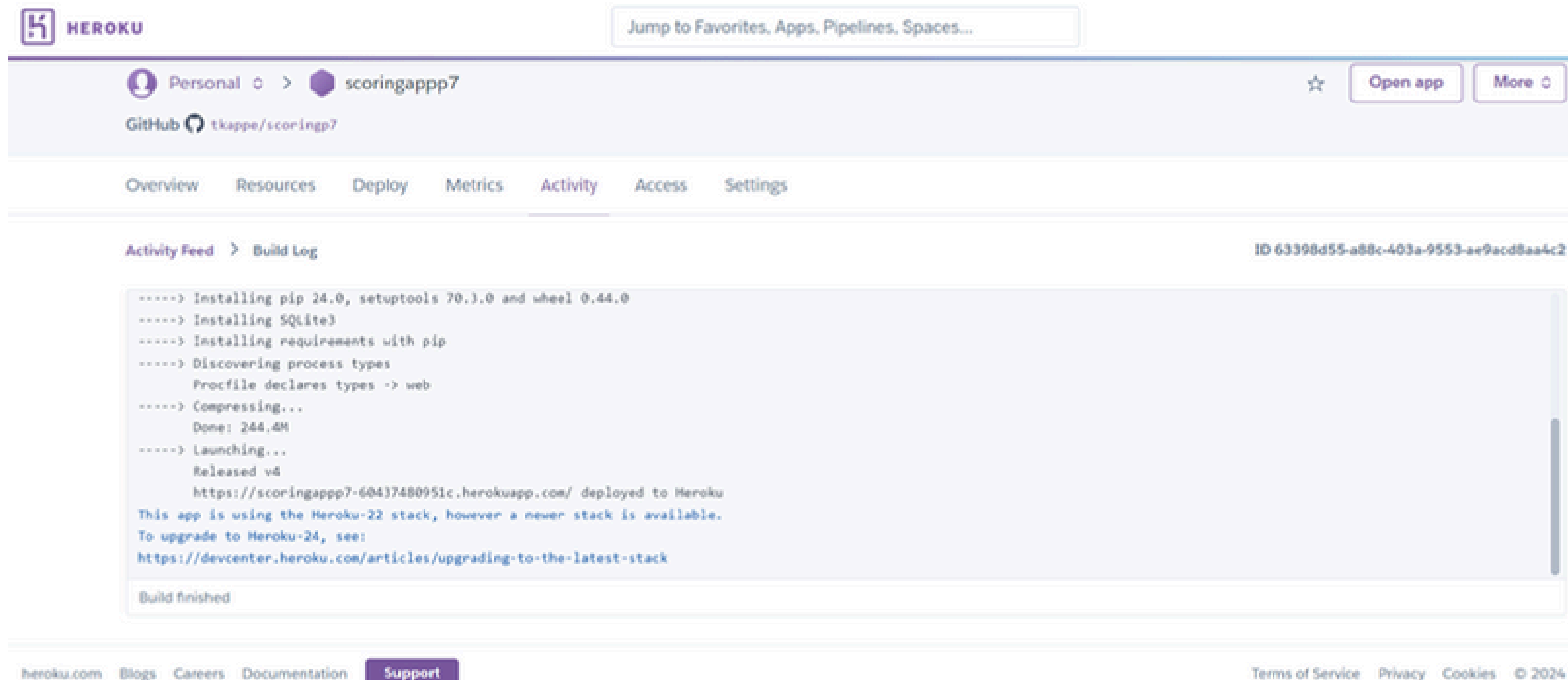
The screenshot shows the GitHub interface for the repository 'tkappe / scoringp7'. The repository is public and has 14 commits. The file list includes:

File	Commit Message	Time
tkappe	completed README.md	3 days ago
.github/workflows	specifying python version to run	3 weeks ago
data	adding tests gitignore app.py templates	3 weeks ago
myenv	adding requirements.txt	3 weeks ago
templates	adding tests gitignore app.py templates	3 weeks ago
test	Dockerfile, Procfile, github actions settings	3 weeks ago
.gitignore	adding tests gitignore app.py templates	3 weeks ago
LICENSE	Initial commit	3 weeks ago
Procfile	Dockerfile, Procfile, github actions settings	3 weeks ago
README.md	completed README.md	3 days ago
app.py	last changes app.py	3 days ago
lgbm1_best_model.pkl	adding requirements.txt	3 weeks ago
requirements.txt	adding requirements.txt	3 weeks ago
runtime.txt	specifying python version to run	3 weeks ago

The right sidebar shows the repository's metadata:

- About:** Scoring model implementation v2. Includes Readme, MIT license, Activity, 0 stars, 1 watching, and 0 forks.
- Releases:** No releases published. [Create a new release](#)
- Packages:** No packages published. [Publish your first package](#)
- Languages:** Python 95.8%, Cython 3.1%, C 0.9%, C++ 0.1%, Fortran 0.1%, PowerShell 0.0%.

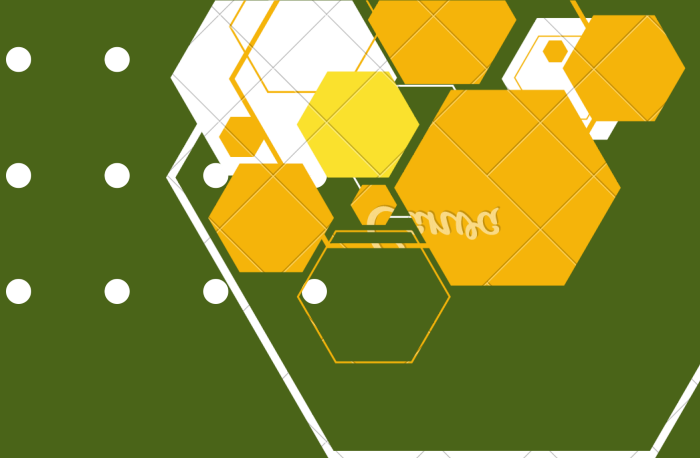
Lien de l'API déployé sur le cloud via Heroku : [ICI](#)



The screenshot shows the Heroku dashboard for the application 'scoringapp7'. The top navigation bar includes the Heroku logo, a search bar, and links to 'Personal', 'scoringapp7', 'Open app', and 'More'. Below the navigation bar, there are tabs for 'Overview', 'Resources', 'Deploy', 'Metrics', 'Activity', 'Access', and 'Settings'. The 'Activity' tab is selected, showing the 'Build Log' for the build ID '63398d55-a88c-403a-9553-ae9acd8aa4c2'. The build log contains the following text:

```
-----> Installing pip 24.0, setuptools 70.3.0 and wheel 0.44.0
-----> Installing SQLite3
-----> Installing requirements with pip
-----> Discovering process types
       Procfile declares types -> web
-----> Compressing...
       Done: 244.4M
-----> Launching...
       Released v4
       https://scoringapp7-60437480951c.herokuapp.com/ deployed to Heroku
This app is using the Heroku-22 stack, however a newer stack is available.
To upgrade to Heroku-24, see:
https://devcenter.heroku.com/articles/upgrading-to-the-latest-stack
Build finished
```

The footer of the Heroku dashboard includes links to 'heroku.com', 'Blogs', 'Careers', 'Documentation', 'Support', 'Terms of Service', 'Privacy', 'Cookies', and '© 2024'.



Lien tableau HTML analyse data drift : data drift

Dataset Drift

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

120
Columns

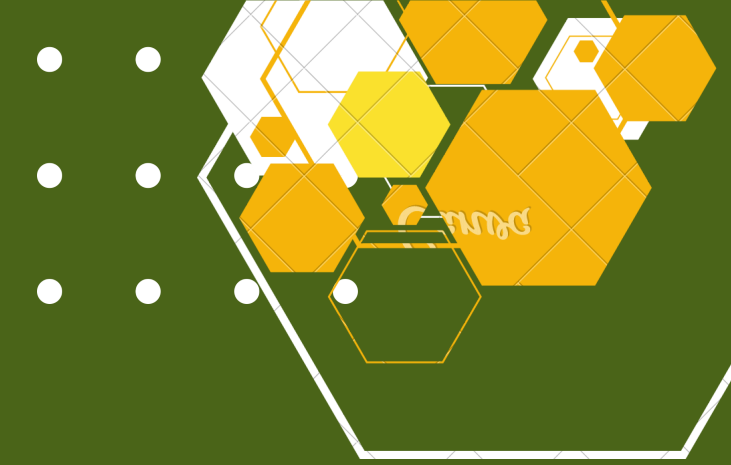
9
Drifted Columns

0.075
Share of Drifted Columns

Data Drift Summary

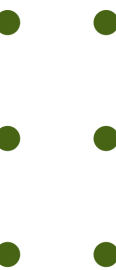
Drift is detected for 7.5% of columns (9 out of 120).

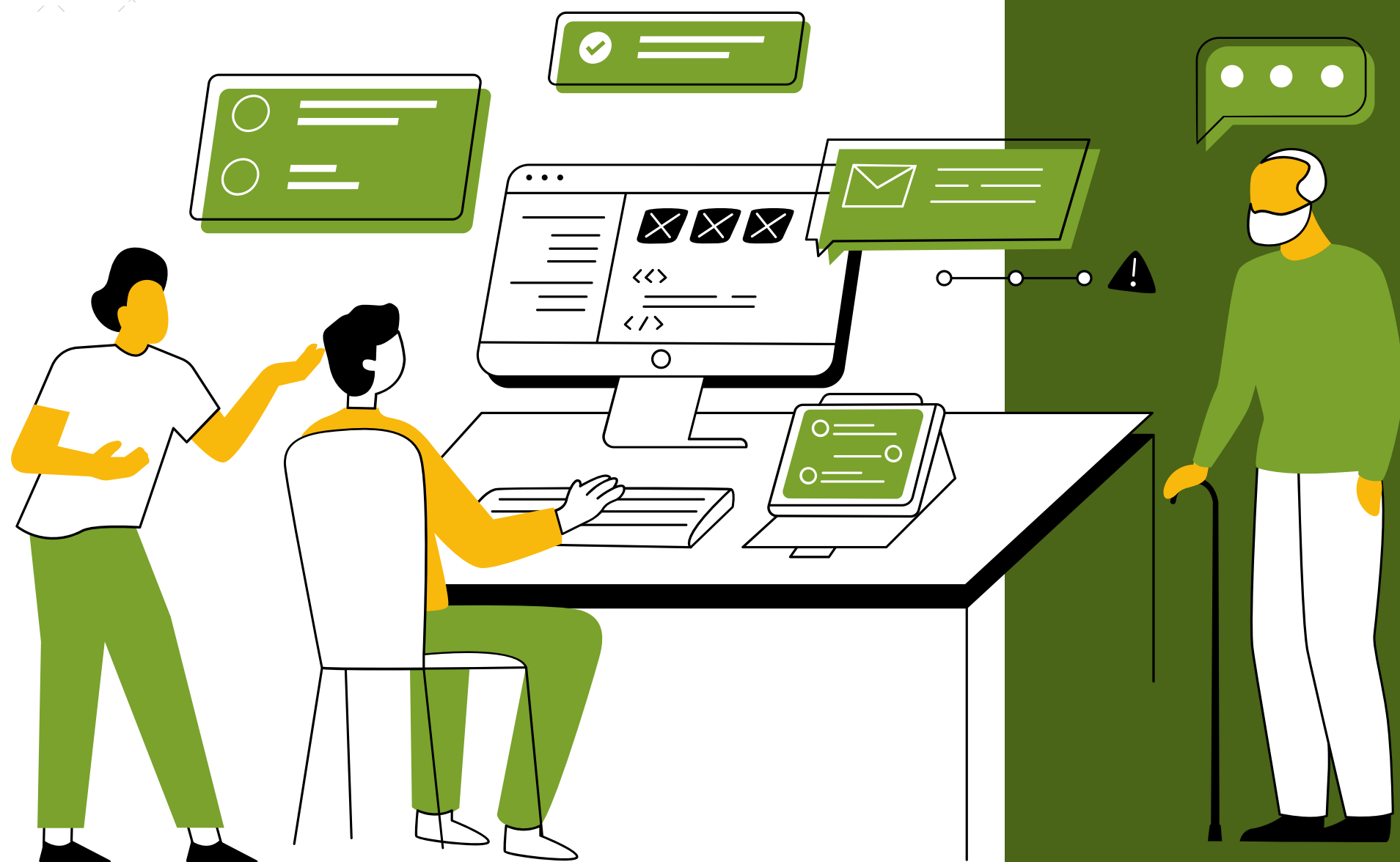
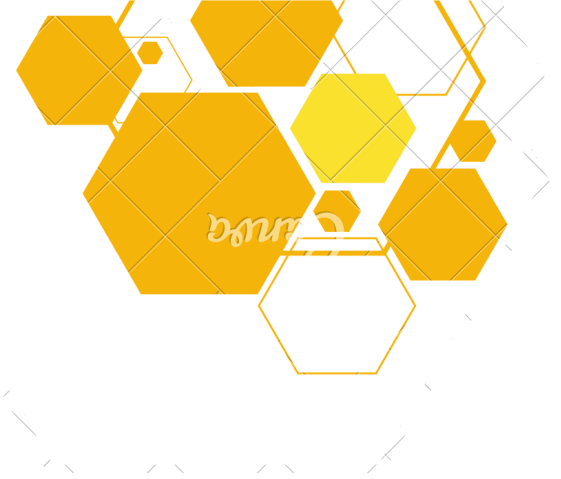
Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (sorted)	0.379052
AMT_REQ_CREDIT_BUREAU_MON	num			Detected	Wasserstein distance (sorted)	0.280765
AMT_GOODS_PRICE	num			Detected	Wasserstein distance (sorted)	0.210785
AMT_CREDIT	num			Detected	Wasserstein distance (sorted)	0.207334
AMT_ANNUITY	num			Detected	Wasserstein distance (sorted)	0.140302
AMT_REQ_CREDIT_BUREAU_WEEK	num			Detected	Wasserstein distance (sorted)	0.11426
AMT_REQ_CREDIT_BUREAU_YEAR	num			Detected	Wasserstein distance (sorted)	0.11426



Limites et améliorations possibles

- *Méconnaissance du milieu bancaire ==> vérification de la cohérence du pré-processing*
- *Amélioration des performances de la modélisation*
- *Il serait intéressant de développer un dashboard avec une page « banque » et une page « client ». Cela permettrait au chargé clientèle d'avoir accès à certaines données permettant d'expliquer la réponse, sans nécessairement pouvoir les montrer au client.*
- *Par ailleurs, il serait intéressant de rajouter une partie interactive qui permettrait au client de voir quelle valeur sur quelle variable aurait pu lui permettre d'obtenir son crédit.*
- *En ce sens, on pourrait envisager une page « scenario » où l'on pourrait changer une ou plusieurs valeurs du profil du client et voir l'impact sur la réponse de la banque.*





Merci

