Implémentez un modèle de scoring

OPENCLASSROOMS

Introduction

- **Objectif :**

Développer un modèle de scoring de crédit permettant d'évaluer la solvabilité d'un client. Automatisation & Optimisation de la prise de décision d'octroi de crédit de + en + crucial

- **Solution :**

Développement d'un modèle de machine learning entraîné sur un jeu de données client :

- Estimer la probabilité qu'un client rembourse ou non son emprunt.
- Intégrer ce modèle dans une API de prédiction, déployée sur le cloud
- Fournir des scores de solvabilité en temps réel.

- **Moyens :**

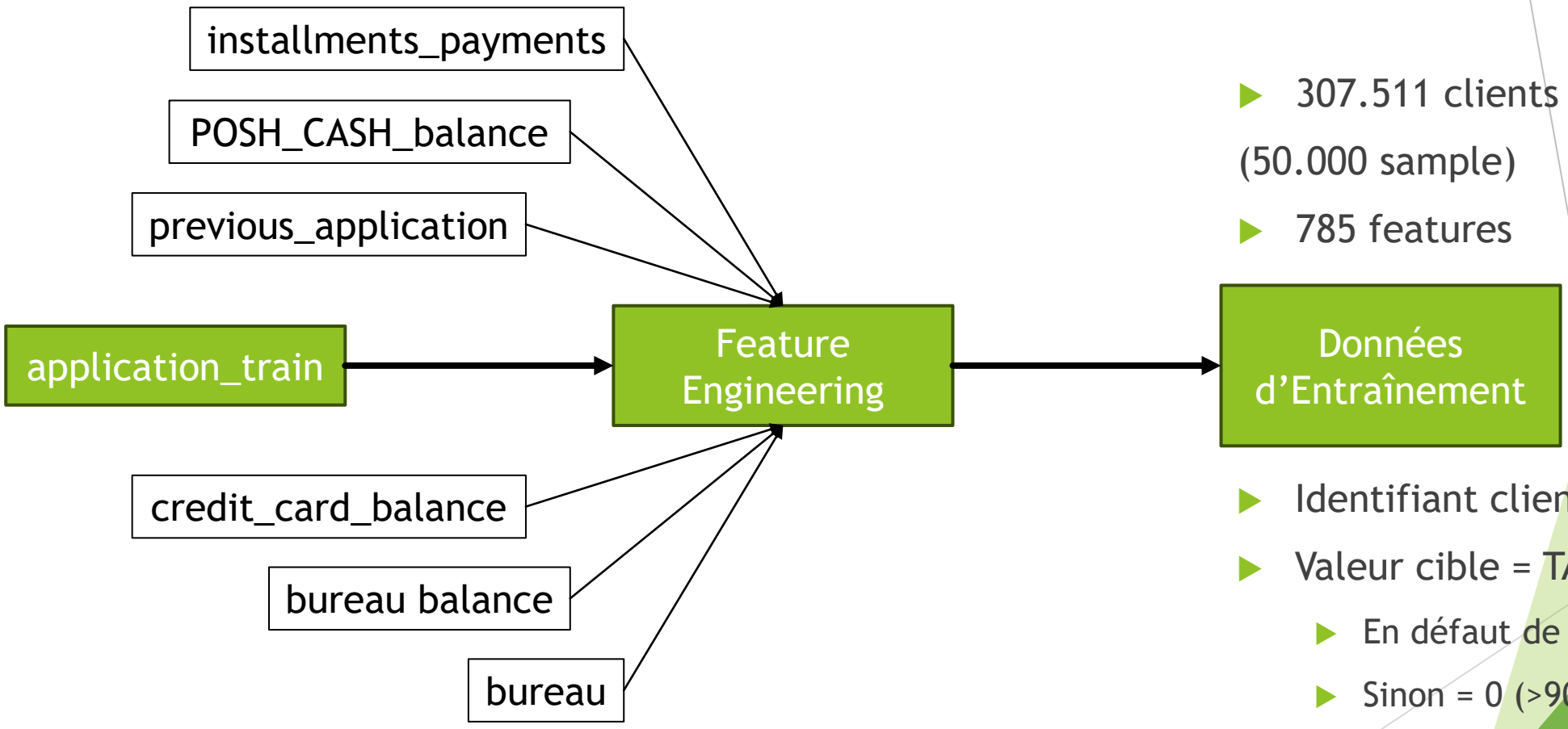
Suivi des expérimentations sur MLFlow → traçabilité & optimisation du modèle.

Tests unitaires & pipeline CI/CD → validation de l'API & déploiement robuste & automatisé.

Ce projet illustre ainsi l'ensemble du cycle de vie d'un modèle de scoring, depuis le prétraitement des données jusqu'à la mise en production et le suivi des performances.

Sommaire

Import & Feature Engineering



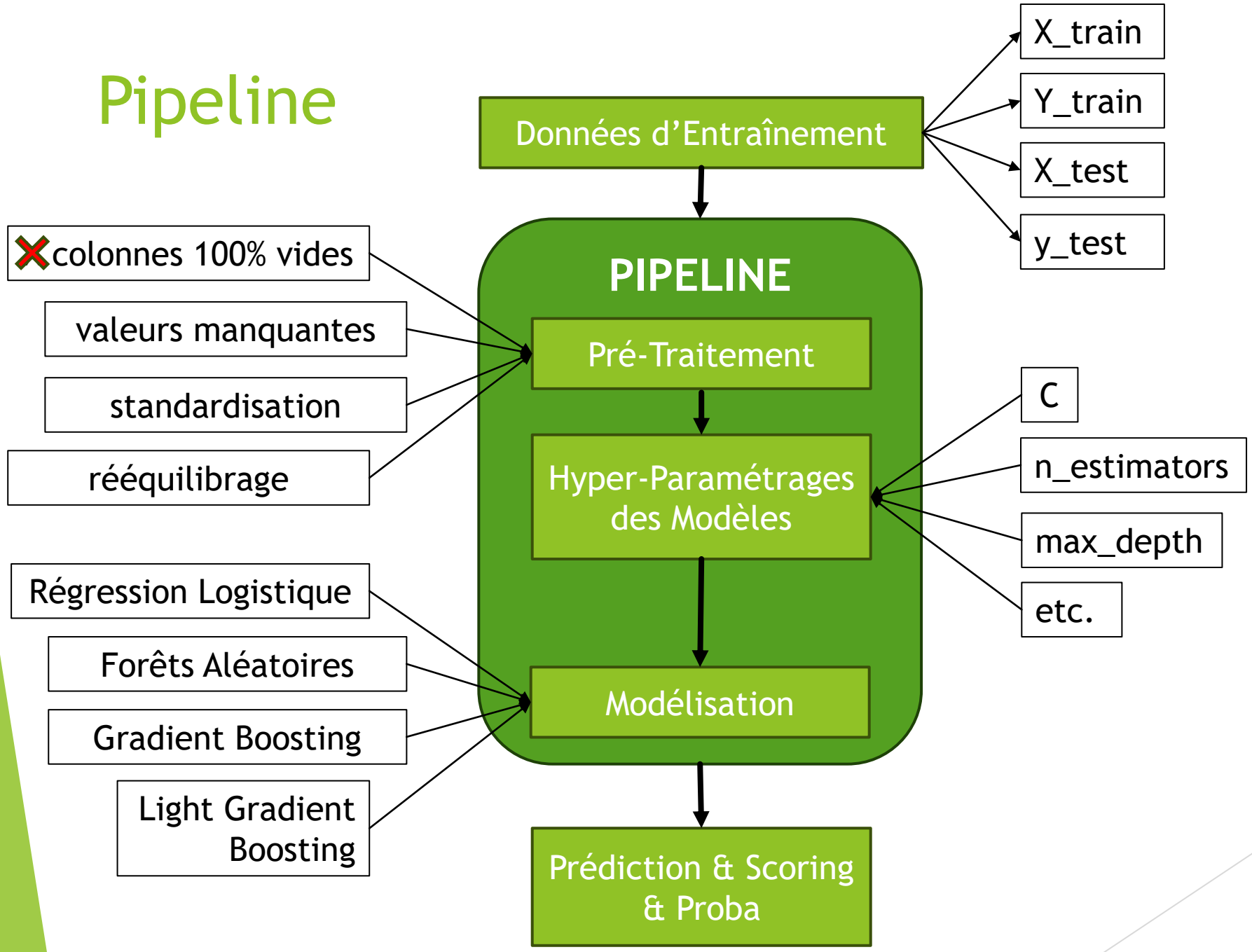
- 307.511 clients

(50.000 sample)

- 785 features

- Identifiant client = SK_ID_CURR
- Valeur cible = TARGET :
  - En défaut de paiement = 1 (<10%)
  - Sinon = 0 (>90%)

Pipeline


Données d'Entraînement
X_train
Y_train
X_test
y_test
PIPELINE
colonnes 100% vides
valeurs manquantes
standardisation
rééquilibrage
Pré-Traitement
Hyper-Paramétrages des Modèles
C
n_estimators
max_depth
etc.
Régression Logistique
Forêts Aléatoires
Gradient Boosting
Light Gradient Boosting
Modélisation
Prédiction & Scoring & Proba

Modélisation & Comparaison des Scores

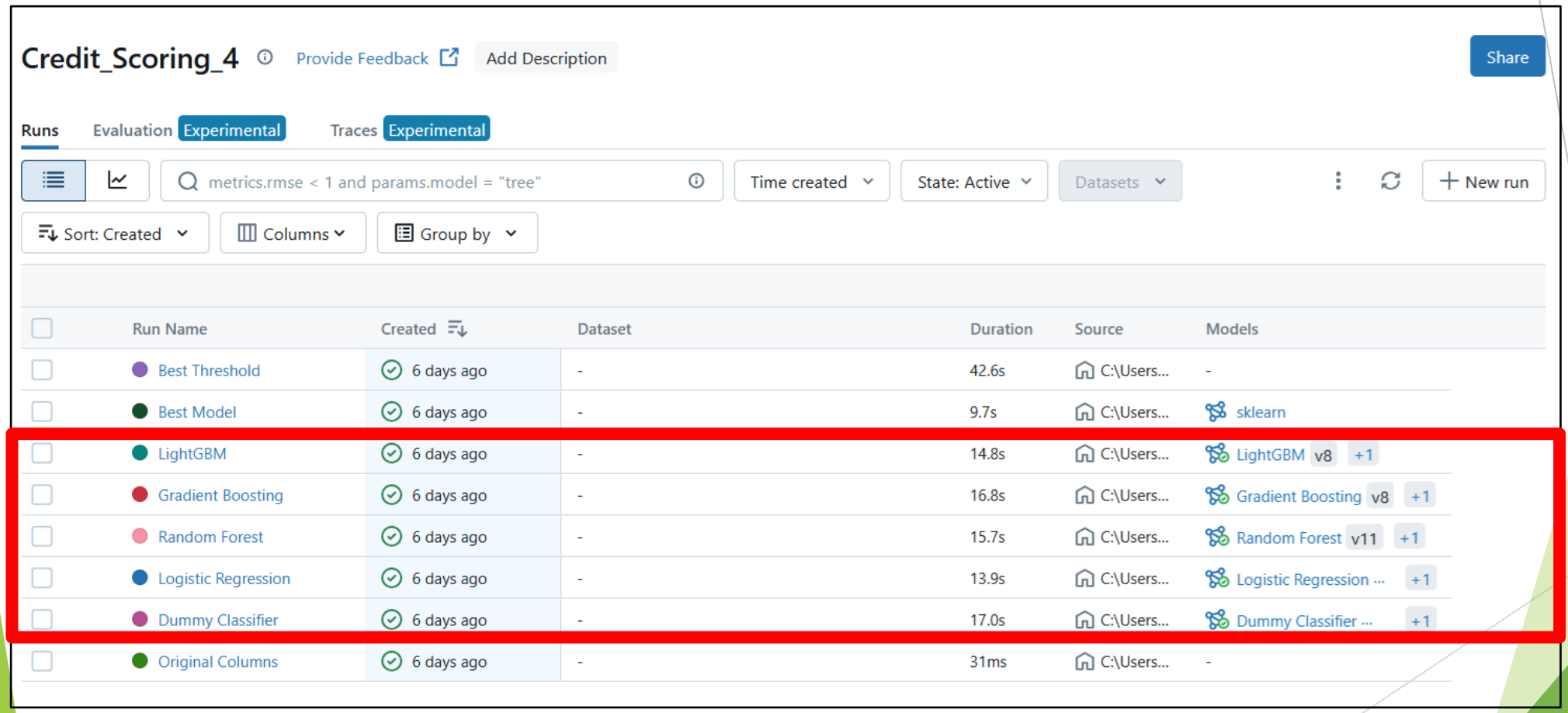
Model	Best Parameters	Accuracy	ROC AUC	F1 Score	Best Score	Test Score
Dummy Classifier	{}	0.9200	0.500000	0.000000	-603.876231	-449.3008621
Logistic Regression	{'model__C': 0.1}	0.7241	0.734053	0.262102	-596.819535	-452.2758692
Random Forest	{'model__max_depth': 10, 'model__n_estimators'...	0.8840	0.692215	0.169054	-575.384046	-439.3926783
Gradient Boosting	{'model__max_depth': 5, 'model__n_estimators':...	0.9189	0.753027	0.064591	-586.736425	-440.2132604
LightGBM	{'model__learning_rate': 0.01, 'model__max_dep...	0.8927	0.680899	0.167572	-583.965870	-432.920272

Score → somme des pertes monétaires directs résultant de prédictions fausses :

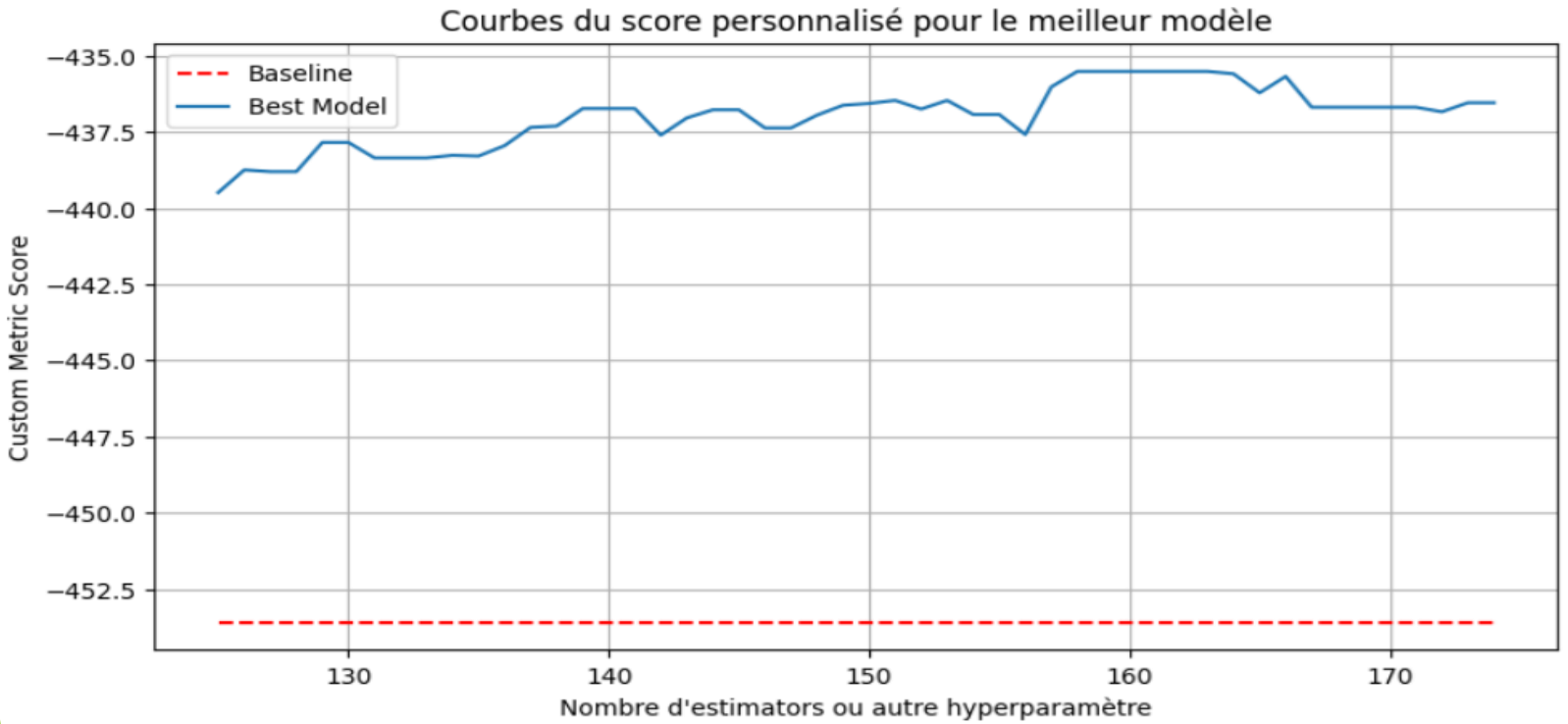
Faux négatif = AMT_CREDIT

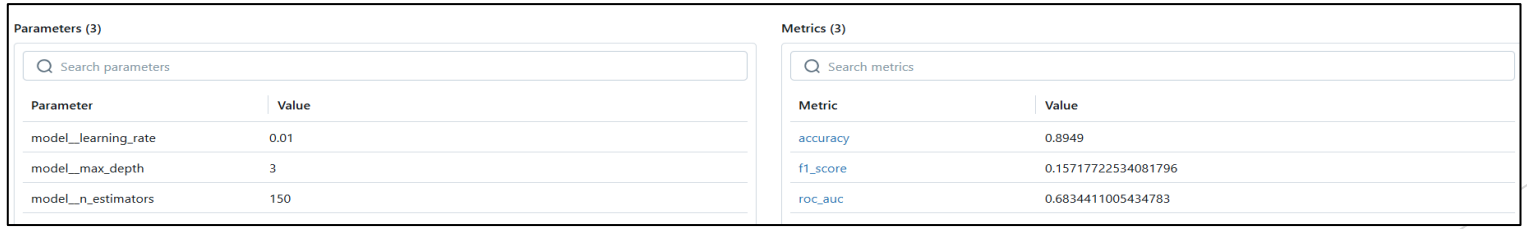
Faux positif = frais bancaires (~19% du crédit)

Interface MlFlow



Optimisation & Sauvegarde du Modèle sur MlFlow





Parameters (3)

Search parameters

Parameter	Value
model__learning_rate	0.01
model__max_depth	3
model__n_estimators	150

Metrics (3)

Search metrics

Metric	Value
accuracy	0.8949
f1_score	0.15717722534081796
roc_auc	0.6834411005434783

Meilleur modèle = Light GBM

Meilleur paramétrage :

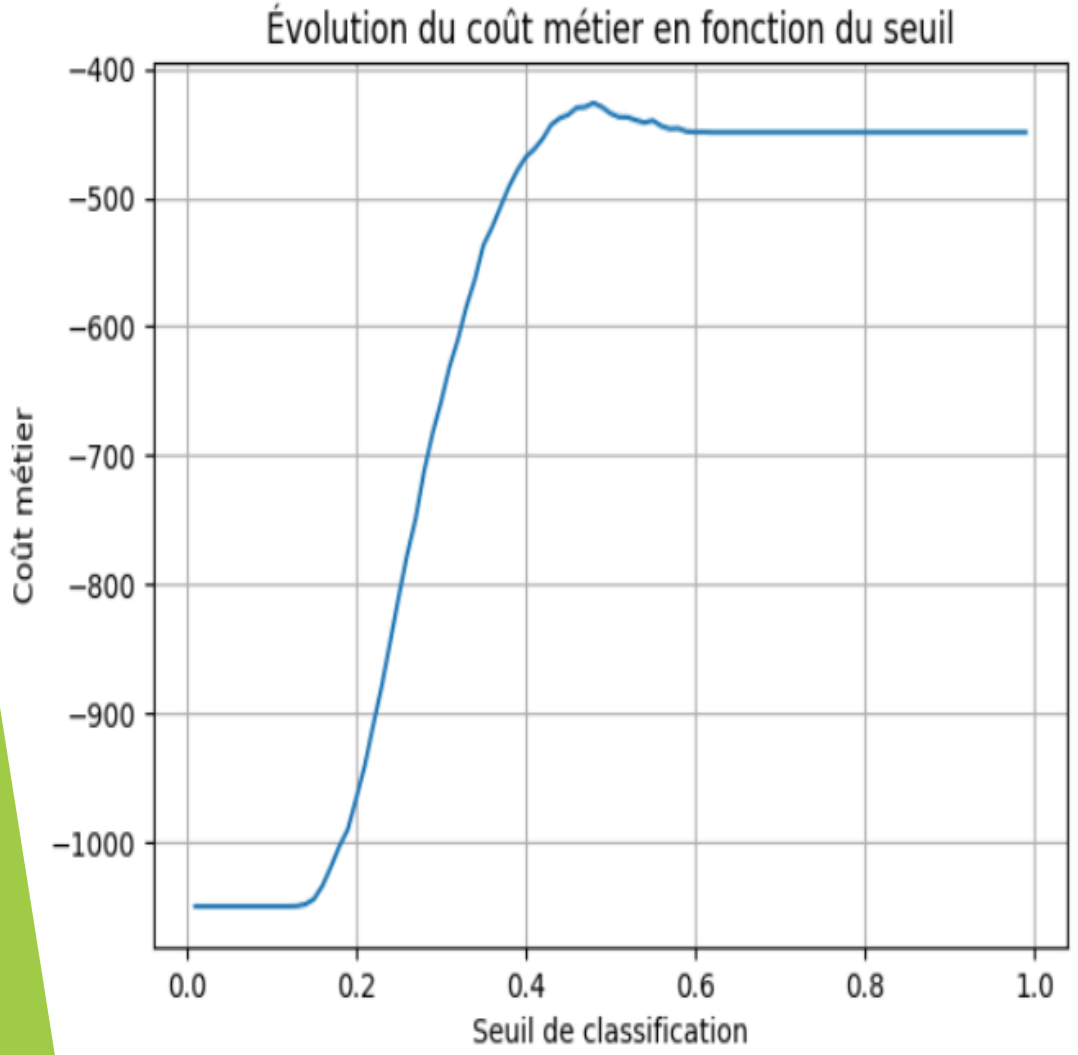
- learning_rate = 0.01
- max_depth = 3
- n_estimators = 150

Nombre optimal d'estimateurs = 158

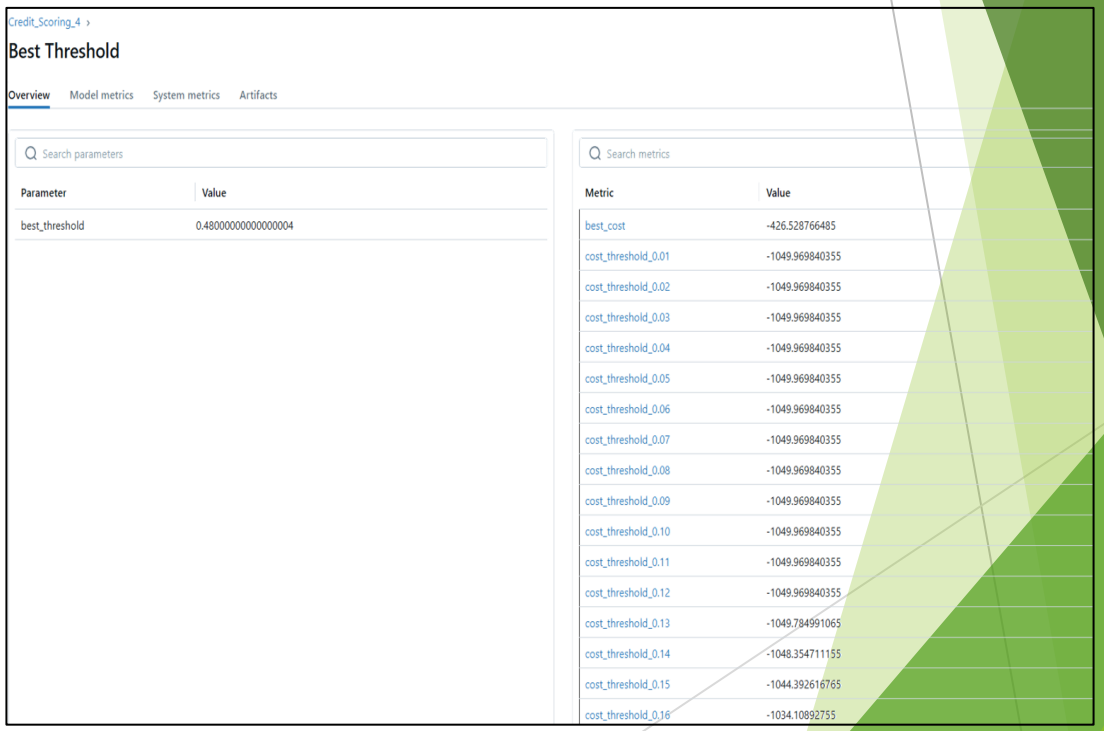
Perte optimale = -435.5M

http://127.0.0.1:5000

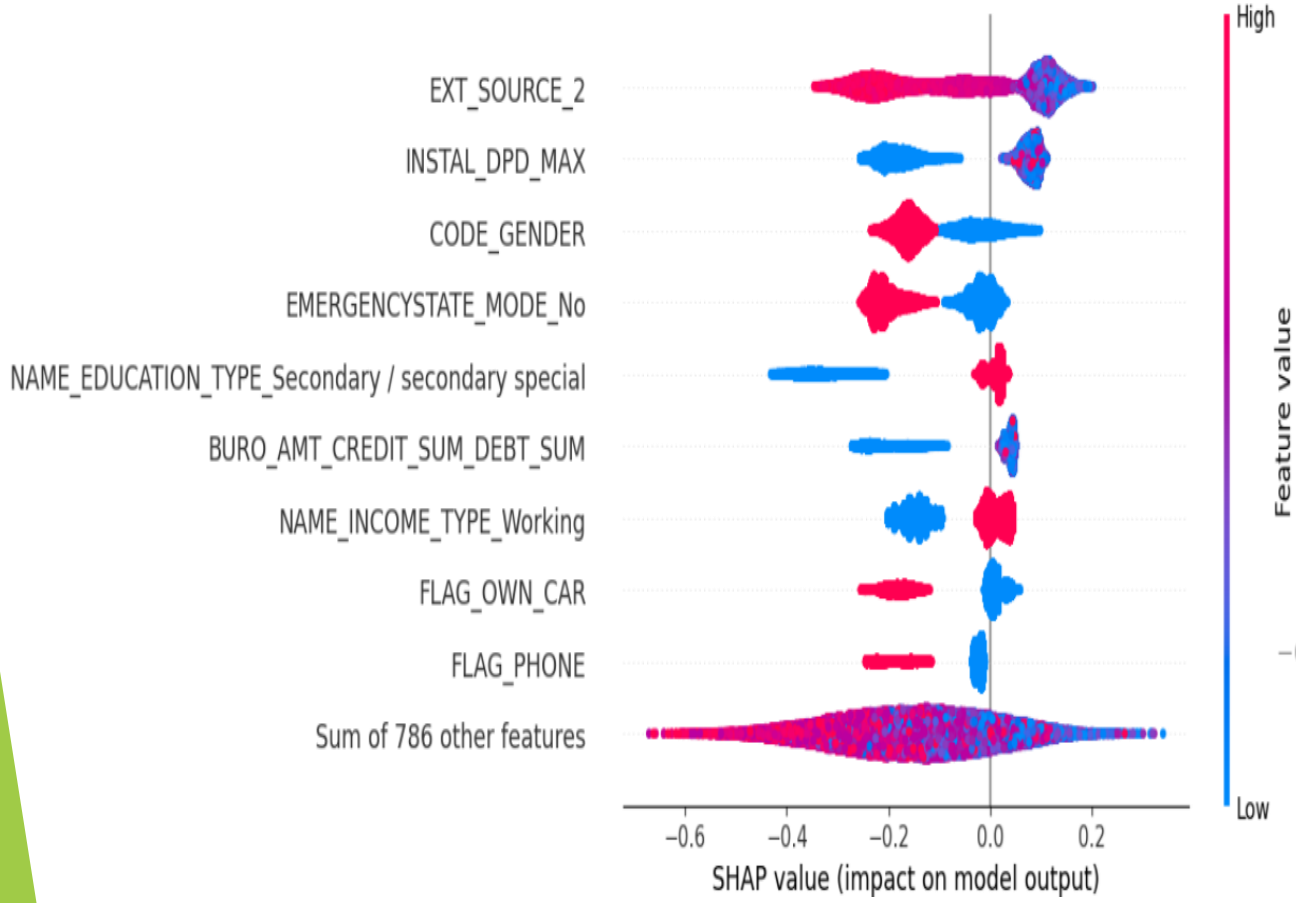
Calcul du Seuil Optimal de Probabilité



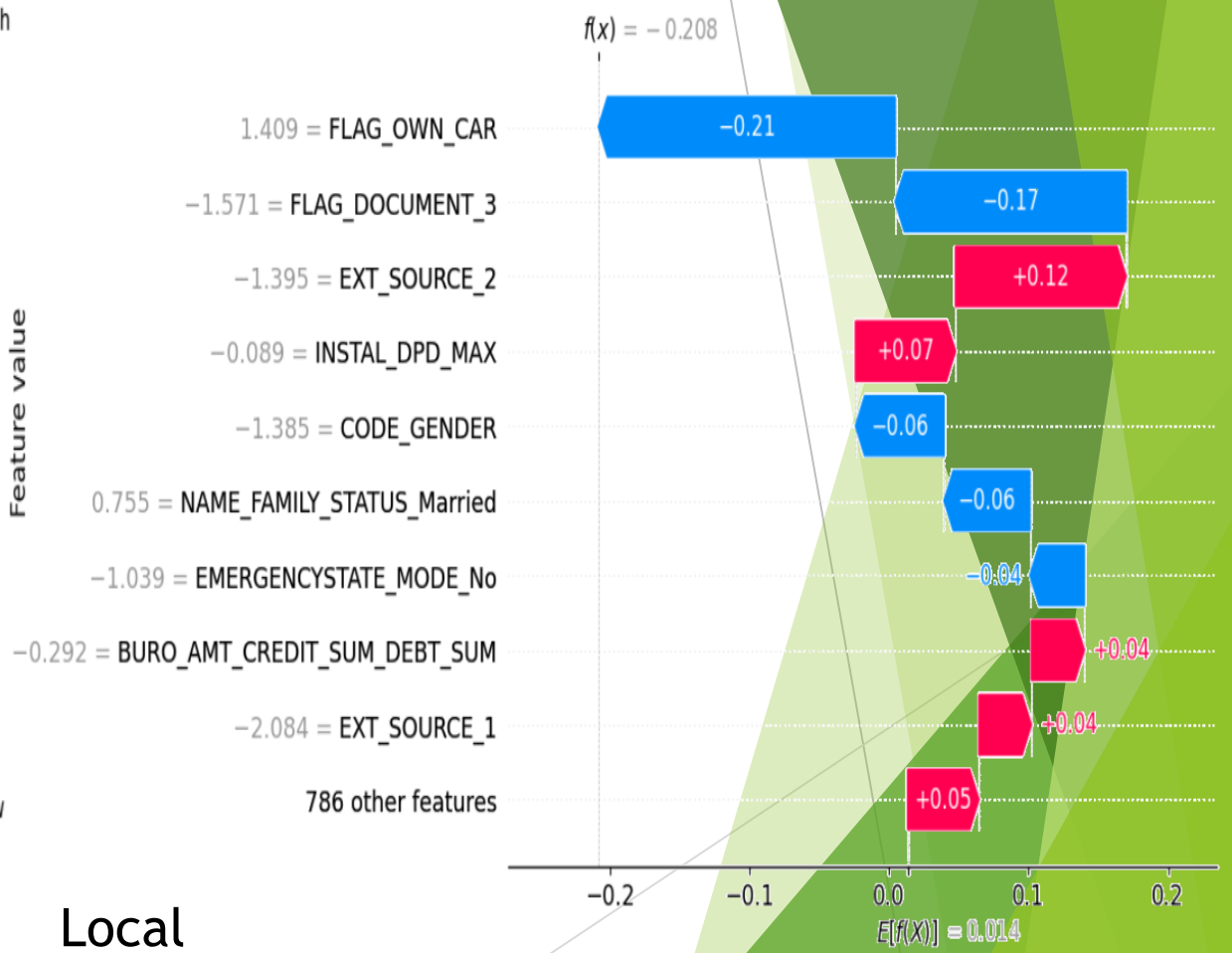
Seuil optimal = 0.48
Perte optimale = -426.52M



Feature Importance



Global



Local

Deploiement API – Documents Nécessaires :

- **Procfile :** Fichier de configuration pour Heroku

- **app.py :** Code de l'API

- **deploy.yml :** Pipeline CI/CD pour déploiement sur Heroku
- **mlruns_reduced/ :** Dossier contenant les fichiers du modèle (récupérés depuis MlFlow)

  - **best_threshold :** Le seuil optimal de prédiction
  - **columns.pkl :** Liste des colonnes du dataset utilisé
  - **model.pkl :** Modèle de machine learning entraîné

- **requirements.txt :** Liste des dépendances Python
- **runtime.txt :** Version de Python pour Heroku

- **test_app.py :** Fichier de tests unitaires pour l'API

- **README.md :** Documentation du projet

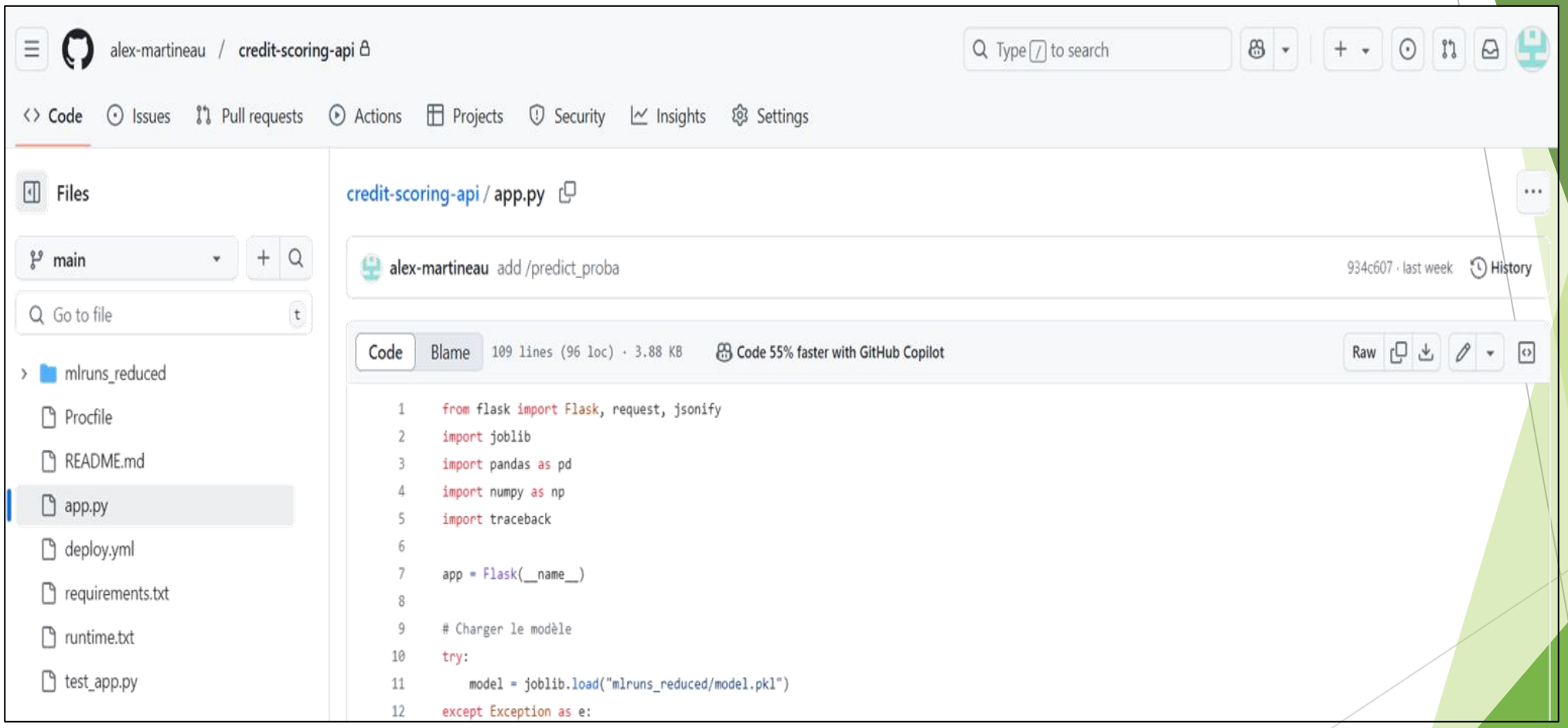
Dépôt Git

git add <fichier>
git commit -m "description"

```
(myenv) C:\Users\Alex-\Documents\Openclassrooms\Data Scientist\P7_martineau_alexandre>git ls-files
Procfile
README.md
app.py
deploy.yml
mlruns_reduced/best_threshold
mlruns_reduced/columns.pkl
mlruns_reduced/model.pkl
requirements.txt
runtime.txt
test_app.py
```

git push
origin main

GitHub



https://github.com/alex-martineau/credit-scoring-api/blob/main/app.py

Déploiement sur Heroku

- git add <fichier>
- Git commit -m « description »

```
(myenv) C:\Users\Alex-\Documents\Openclassrooms\Data Scientist\P7_martineau_alexandre>git log --oneline
eb5de3f (HEAD -> main, origin/main) Ajout du doc Readme
934c607 (heroku/main) add /predict_proba
e0ba3a8 Remove nouveau_model_2.pkl from Git tracking
84a89c9 Update GitHub
3b2edf1 Remove nouveau_model_2.pkl from Git tracking
ccf9f3b CI/CD pour Heroku avec clé dans YAML
a332d76 Ajout des tests, du Procfile et mise à jour des dépendances
50e7ef3 Prepare app for Heroku deployment
7789e65 Remove LFS tracking for nouveau_model_2.pkl
264c9a4 Remove folder from Git tracking
62cf11f Essaie avec modele simple
31861cb Supprime .gitignore du dépôt et de GitHub
27f6669 Ajout forcé de deploy.yml pour déploiement
d336fb7 Supprime .gitignore du dépôt
cecbaaa Supprime .gitlab-ci.yml
4f2d1d9 MAJ requirements again
dc15089 Fix dependencies and model loading issue
1925fa3 Mise à jour des dépendances
06ac024 Récupération du dossier mlruns_reduced
f12ac76 Delete mlruns_reduced directory
2084310 Ajout de mes fichiers
7a199cd Ajout du pipeline CI/CD
ba05ff5 Initial commit
```

```
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.96 KiB | 1005.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Updated 10 paths from f537497
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-24 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> Using Python 3.10.10 specified in runtime.txt
remote: -----> Restoring cache
remote: -----> Using cached install of Python 3.10.10
remote:
remote:  !     Warning: A Python security update is available!
remote:  !
remote:  !     Upgrade as soon as possible to: Python 3.10.16
remote:  !     See: https://devcenter.heroku.com/articles/python-runtimes
remote:
remote: -----> Installing pip 24.3.1, setuptools 70.3.0 and wheel 0.45.1
remote: -----> Installing SQLite3
remote: -----> Installing dependencies using 'pip install -r requirements.txt'
remote: -----> Discovering process types
remote:        Procfile declares types -> test, web
remote:
remote: -----> Compressing...
remote:        Done: 229.1M
remote: -----> Launching...
remote:        Released v17
remote:        https://my-scoring-app-546acd78d8fa.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/my-scoring-app.git
   934c607..eb5de3f  main -> main
```

HEROKU

- Création du compte
- Création de l'application: « my-scoring-app »
- Déploiement via GitHub / via Git directement

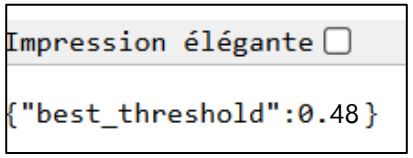
API déployée & prête à l'emploi

Fonctionnalités

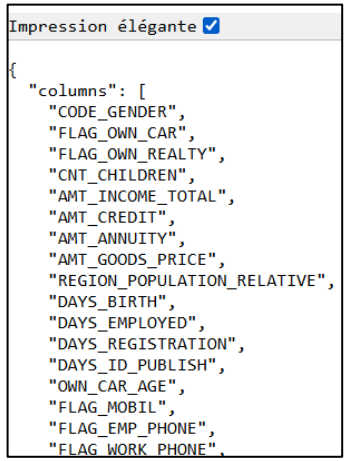
https://my-scoring-app-546acd78d8fa.herokuapp.com/

Bienvenue sur le serveur de prédiction ! Pour consulter le seuil optimal de prédiction, dirigez-vous vers : '/predict' Pour consulter les features, dirigez-vous vers : '/features' Pour consulter l'ensemble des données que vous nous avez envoyées, dirigez-vous vers : '/data' En vous souhaitant une bonne expérimentation :)

/best_threshold



/features



/predict :

Permet de soumettre des données au modèle pour obtenir une prédiction :

- Prêt accordable → Yes
- Prêt à risque → No

Exemple :
{'prediction': {'client n°155312': 'Yes'}}

Tests Unitaires

pytest

test_app.py

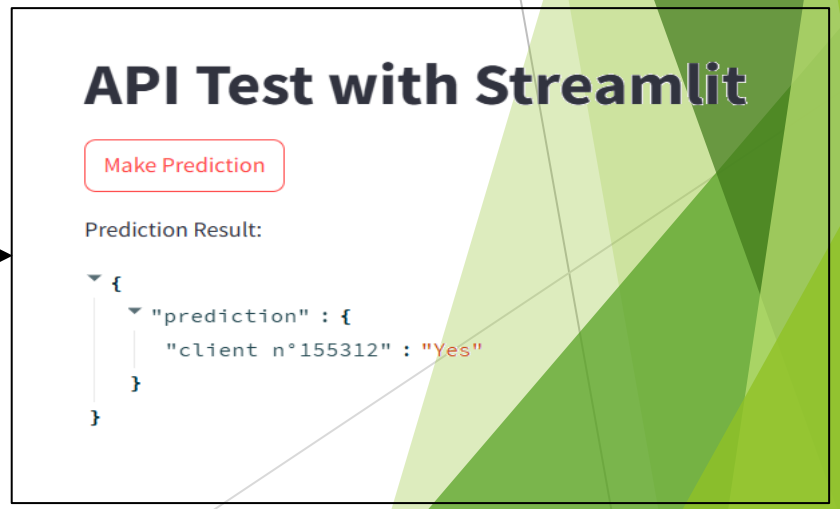
pytest
./test_app.py

```
(myenv) C:\Users\Alex-\Documents\Openclassrooms\Data Scientist\P7_martineau_alexandre>pytest ./test_app.py
============================================ test session starts ============================================
platform win32 -- Python 3.10.10, pytest-8.3.4, pluggy-1.5.0
rootdir: C:\Users\Alex-\Documents\Openclassrooms\Data Scientist\P7_martineau_alexandre
plugins: anyio-4.8.0, Faker-33.3.1
collected 2 items

test_app.py ..                                                                                      [100%]

============================================= 2 passed in 3.55s =============================================
```

Streamlit

streamlit_app.py

streamlit run
streamlit_app.py



Veille Technique (Data Drift)

121	9	0.0744
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 7.438% of columns (9 out of 121).

Search

	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
>	AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (normed)	0.359052
>	AMT_REQ_CREDIT_BUREAU_MON	num			Detected	Wasserstein distance (normed)	0.281765
>	AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.210785
>	AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207334
>	AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.161102
>	AMT_REQ_CREDIT_BUREAU_WEEK	num			Detected	Wasserstein distance (normed)	0.15426
>	NAME_CONTRACT_TYPE	cat			Detected	Jensen-Shannon distance	0.14755
>	DAYS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.138977
>	FLAG_EMAIL	num			Detected	Jensen-Shannon distance	0.122121
>	FLAG_DOCUMENT_3	num			Not Detected	Jensen-Shannon distance	0.062496

Rows per page: 10 rows 1-10 of 121

Conclusion

- **Conception & déploiement du modèle de scoring de crédit réussi**
- → **Prêt à Dépenser capable d'évaluer la solvabilité des clients.**

- **Modélisation** : nettoyage des données, sélection des variables et entraînement d'un modèle de machine learning performant.
- **MLFlow :** tracabilité des expérimentations, sauvegardes des performances & optimisation du seuil de décision.
- **Déploiement sous forme d'API** : exploitable en production.
  - Hébergée sur Heroku,
  - Envoi de données → obtention instantanée d'une prédiction sur la solvabilité d'un client.
  - Pipeline CI/CD pour automatiser les mises à jour et garantir la fiabilité du déploiement.
- **Tests unitaires** intégrés → assure la robustesse l'API
- Analyse de **data drift** mise en place avec Evidently → surveille l'évolution des données en production.

Ce projet a donc couvert **l'ensemble du cycle de vie d'un modèle de machine learning**, depuis la conception jusqu'au déploiement et au suivi post-production. Une approche complète qui combine **data science, ingénierie logicielle et bonnes pratiques de MLOps** !

Merci pour votre attention