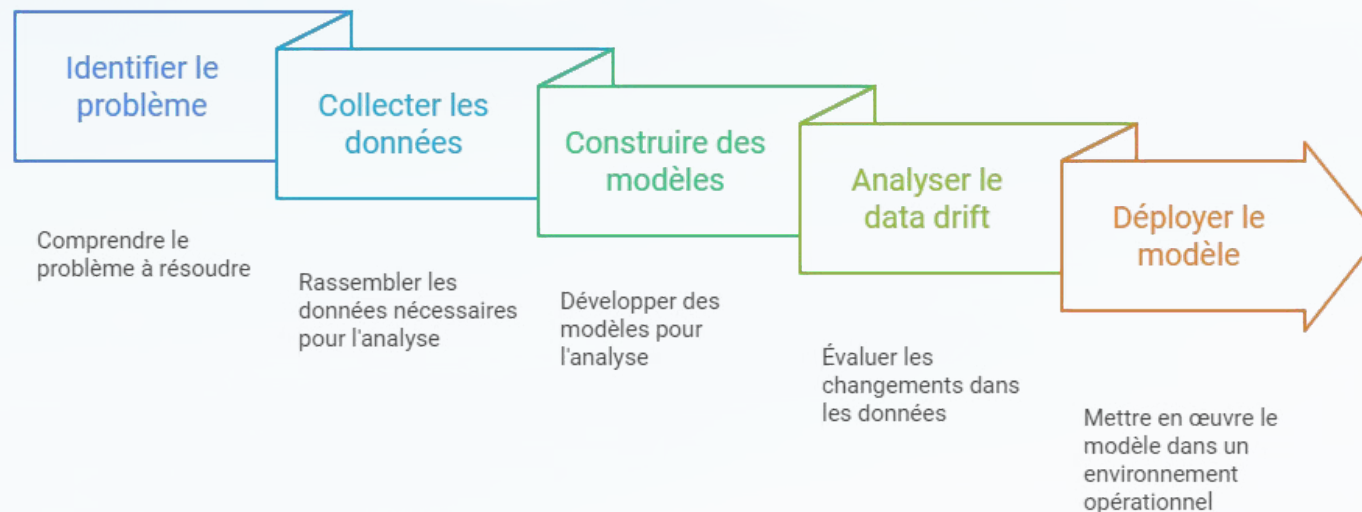


# P7: Implémentez un Modèle de Scoring



Nous explorons l'implémentation d'un modèle de scoring pour l'institution financière "Prêt à Dépenser".

## Processus d'analyse de données et de déploiement



# Contexte et Problématique

## Contexte

L'entreprise propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

## Problématique

Faux négatifs (FN) :

- Correspondent à des clients risqués qui sont classés à tort comme solvables (crédit accordé alors qu'ils ne rembourseront pas).
- Coût élevé pour l'entreprise : pertes financières directes dues aux défauts de paiement.

Faux positifs (FP) :

- Correspondent à des clients solvables qui sont classés à tort comme risqués (crédit refusé à un client fiable).
- Coût indirect : perte d'opportunités commerciales et insatisfaction des clients.

L'importance du rappel (éviter les faux négatifs) est donc critique, car accorder des crédits à des clients risqués peut avoir des conséquences graves sur la rentabilité.

**Missions :** Développer un modèle de scoring pour prédire la probabilité de défaut de paiement des clients de Prêt à Dépenser.

# Les données

**07 fichiers.csv** : historique des opérations bancaires des clients de l'entreprise

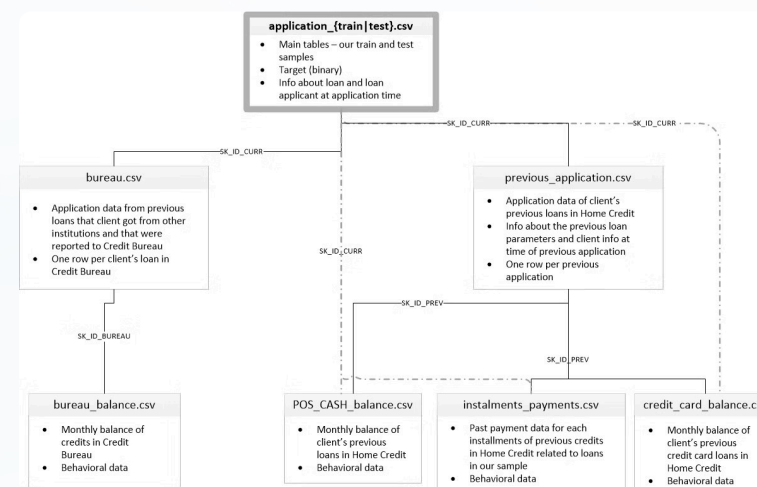
**01 fichier.csv** : description des caractéristiques

**application\_train** : fichier principal contenant contenant les demandes prêts en lignes, avec une cible TARGET :

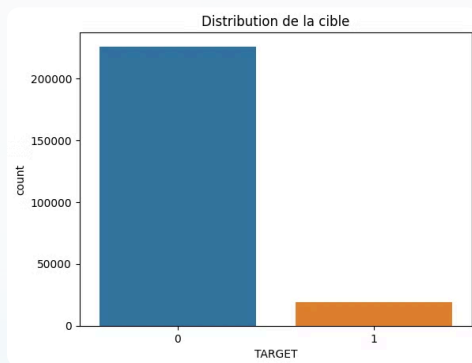
- **1** : Le client a fait défaut sur son prêt.
- **0** : Le client a remboursé son prêt.

**bureau et bureau\_balance** : Transactions financières dans d'autres banques.

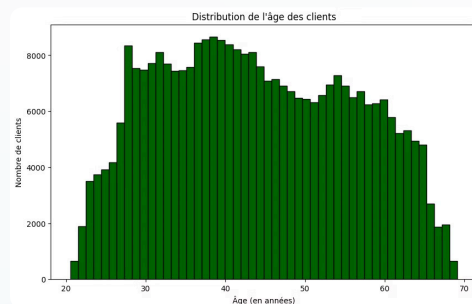
A travers ces tables, on a toutes informations nécessaires (**Montant du crédit, durée, statut de remboursement, revenu du client, etc.**)



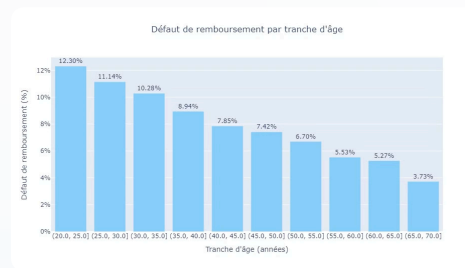
# Analyse exploratoire des données



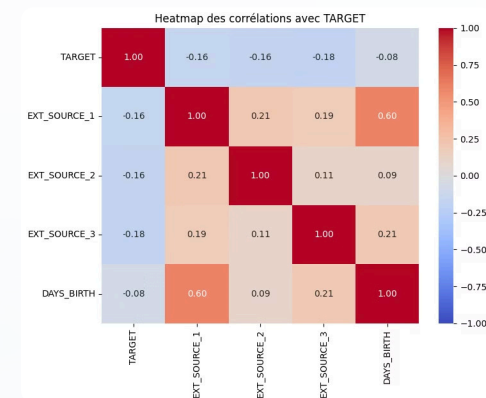
Nous avons, ici, un problème de déséquilibre de classes. Il y a beaucoup plus de prêts remboursés à temps que de prêts non remboursés.



On observe que les clients sont principalement âgés entre 25 et 65 ans.



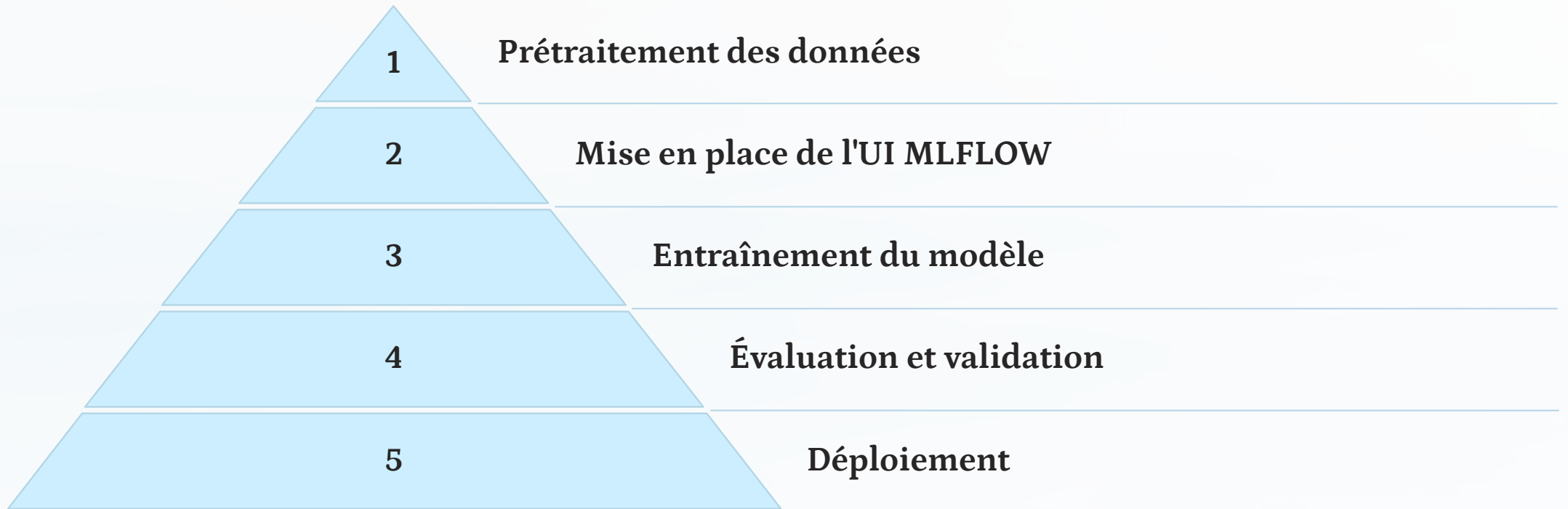
La proportion moyenne de défaut de paiement (Target= 1) dans différentes tranches d'âge révèle que les jeunes sont plus susceptibles de faire défaut par rapport aux plus âgées



Variables plus corrélées au Target.

Les clients plus âgés pourraient recevoir un score EXT\_SOURCE\_1 plus élevé.

# Démarche de Modélisation



# Pré-traitement et Features engineering

Pour enrichir les données de la table **application\_train**, nous avons utiliser les fichiers :

- **bureau** : informations sur les anciens prêts des clients dans d'autres institutions financières. Chaque ligne correspond à un prêt.
- **bureau\_balance** : qui fournit des informations mensuelles sur les anciens prêts. Chaque ligne correspond à un mois pour un prêt spécifique.

1. Supprimer les valeurs aberrantes (1000 ans pour certains clients)
2. Agréger les observations avec les fonctions **min, max, mean, sum**
3. Créer de nouvelles de variables
4. Imputer les NANs
5. Merger les tables
6. Encoder avec **one-hot-encoding** et **Label encoder**
7. **StandardScaler** pour éviter la dominance des variables

Pour avoir une table complète de taille **(307511, 333)**

# Modèles de classification

- Rechercher les variables significatives (**corr**, **chi2**)
- **GridsearchCV** pour les combinaisons de paramètres optimales

## Regression Logistique

```
class_weight={0: 1, 1: 10},  
  
max_depth=10,  
  
min_samples_leaf=1,  
  
min_samples_split=5,  
  
n_estimators=300,  
  
random_state=42
```

## RandomForestClassifier

```
class_weight={0: 1, 1: 10},  
  
max_depth=10,  
  
min_samples_leaf=1,  
  
min_samples_split=5,  
  
n_estimators=300,  
  
random_state=42
```

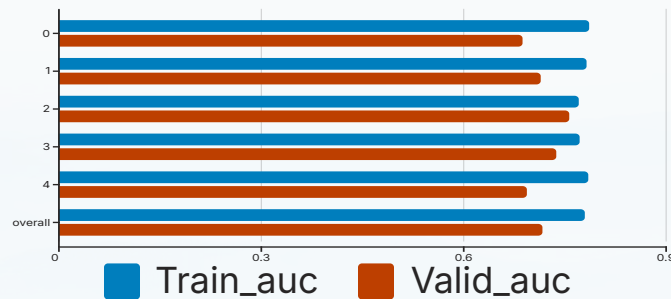
## LightGBMClassifier

```
class_weight={0: 1, 1: 10}  
  
random_state=42  
  
learning_rate= 0.1  
  
max_depth= 5  
  
n_estimators= 100  
  
num_leaves= 50
```

# Cross-validations

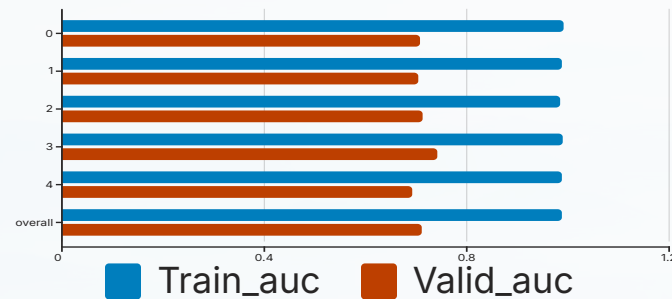
- Evaluer la robustesse et la performance des modèles en les testant sur différentes folds des données.
- Résultats obtenus pour chaque modèle par fold:

Régression Logistique



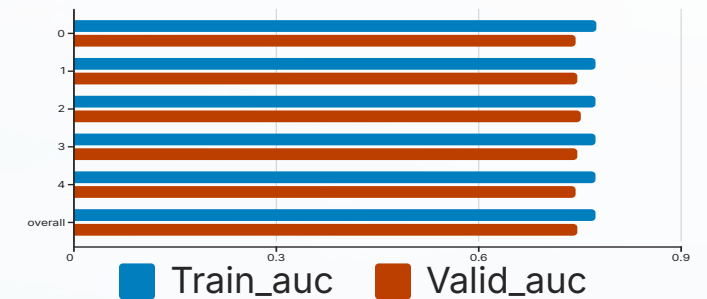
- **Overfitting modéré** :  $\text{train\_auc} > \text{valid\_auc}$
- **Performance variable** : variabilité des scores sur les deux ensembles

RandomForestClassifier



- **Overfitting** :  $\text{Train\_auc}$  nettement supérieur  $\text{valid\_auc}$
- **Performance variable** : certaine variabilité à travers les folds

LightGBMClassifier



- **Compromis entre biais et variance** (Scores proches)
- Résultats sur les différentes itérations **stables**



# Choix des Mesures

Une fonction **Business cost** =  $10 \cdot \text{FN} + \text{FP}$  à minimiser

## Précision (Accuracy)

Proportion de prédictions correctes de défaut de paiement.

## Score F2

Mesure qui favorise la détection des clients risqués. Il donne 2 fois plus de poids au rappel qu'à la précision

## AUC (Area Under Curve)

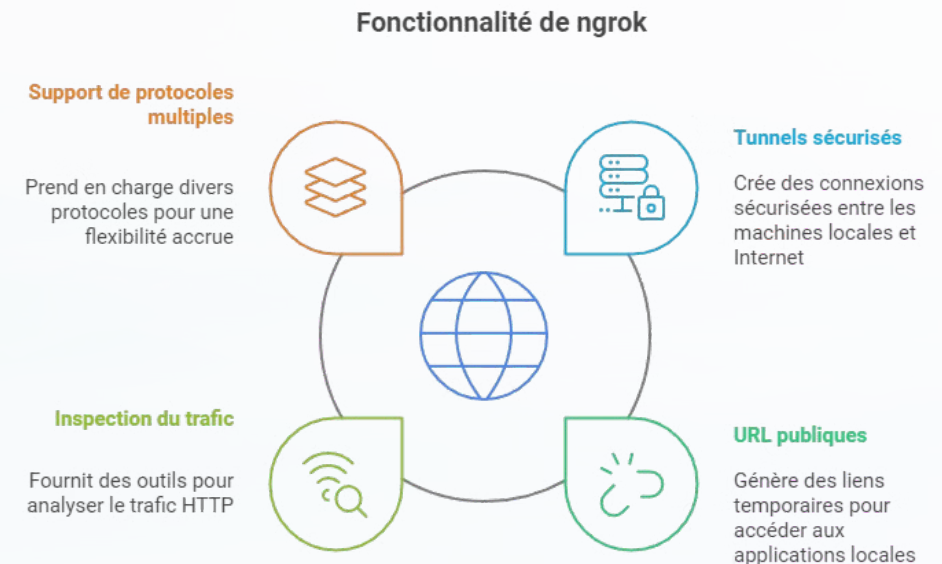
Capacité du modèle à distinguer les clients à risque des clients non à risque.

## Seuil optimal

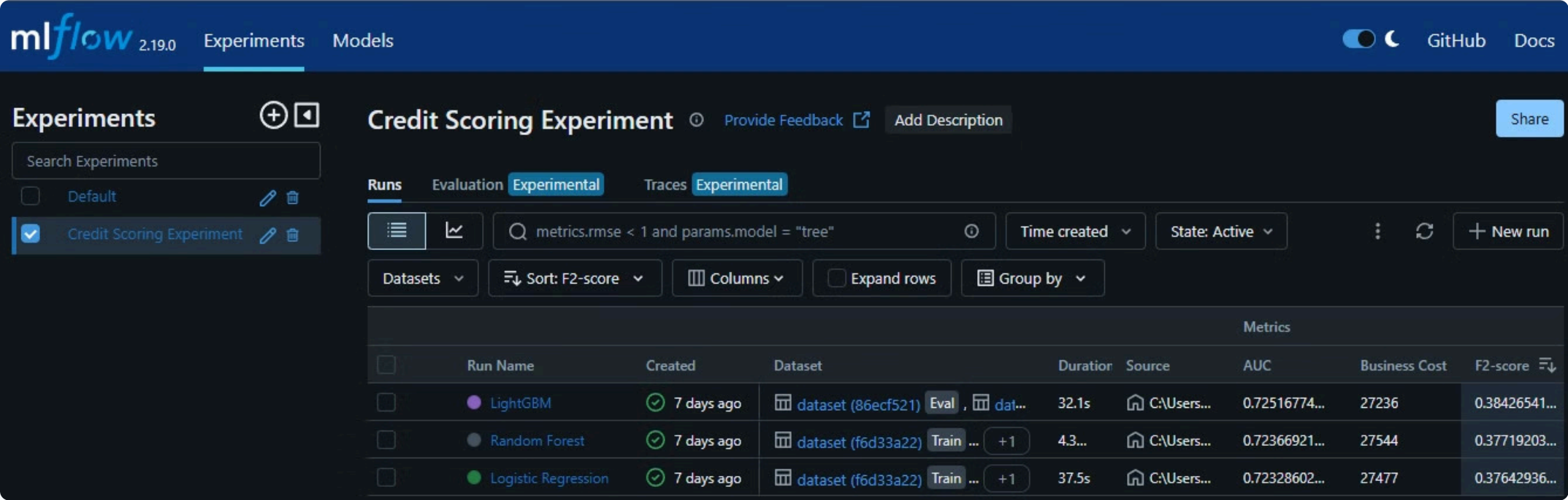
Probabilité permettant d'optimiser le Business cost. Seuil au-delà duquel le prêt est refusé.

# Environnement ui mlflow

- Importer les packages nécessaire
- Créer mon exécution avec un experiment à défaut
- Faire mon premier run avec **LogisticRegression optimisé**
- Faire un run avec **RandomForestClassifier optimisé**
- Faire un 2nd run avec **LGBMClassifier optimisé**
- Exposer le serveur web avec **ngrok**
- Lancer le serveur MLflow : **!mlflow ui**



# Tracking Mlflow



Métriques

Visualisation des performances du modèle.



Paramètres

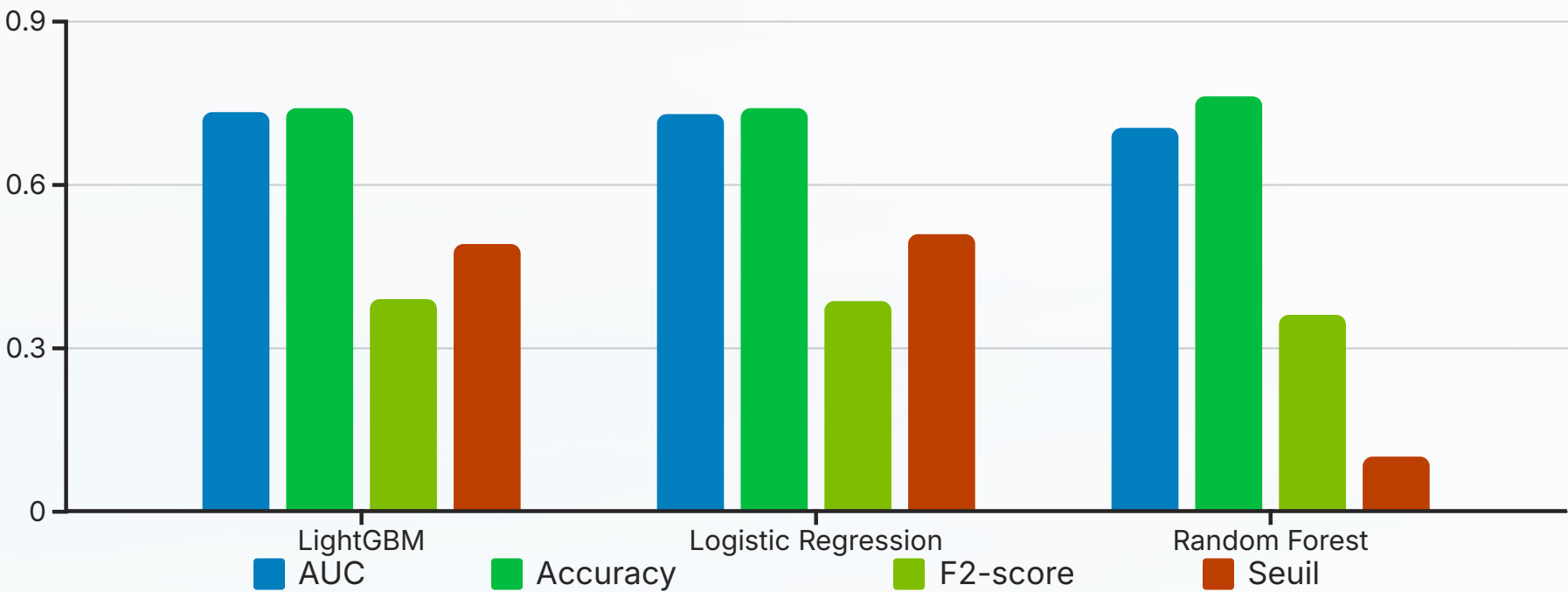
Gestion des hyperparamètres du modèle.



Artefacts

Stockage des modèles et des données d'entraînement.

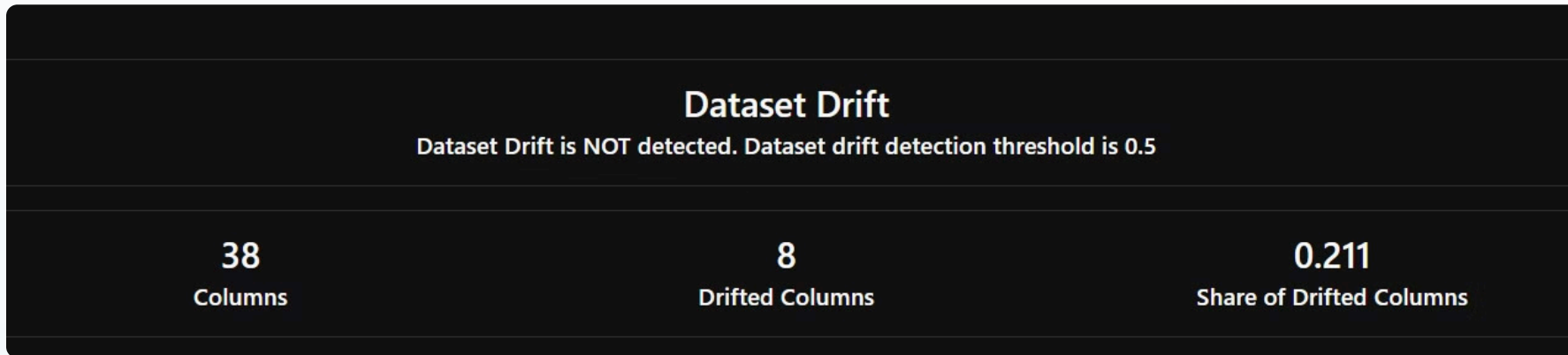
# Évaluation et validation



Le LightGBM semble être le meilleur modèle pour résoudre notre problème de classification.

# Analyse du Data Drift

- Surveiller les changements dans la distribution des données au fil du temps avec **EVIDENTLY**



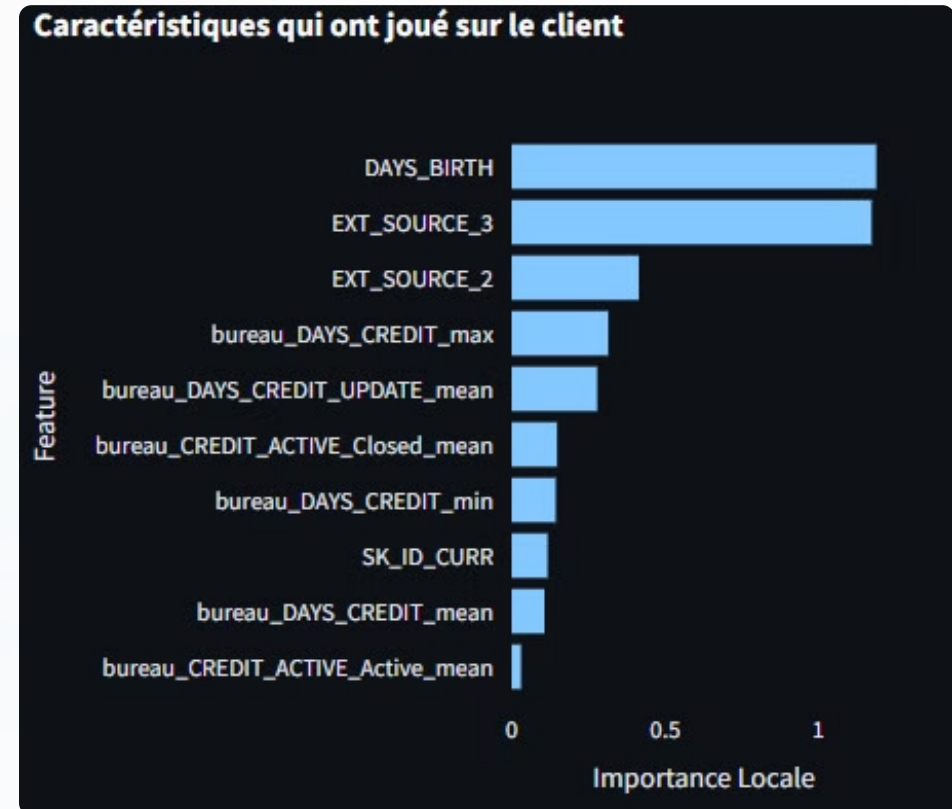
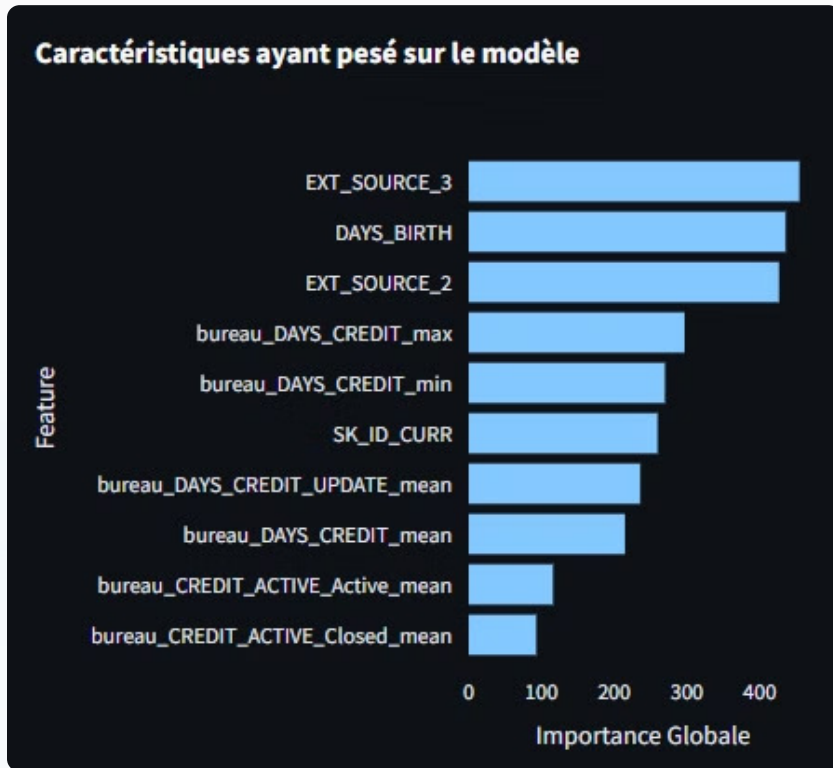
- Numeric columns : Test statistique de Wasserstein distance (similarité entre les distributions)
- Categorical columns : Test de Jensen-Shannon distance (mesure de la similarité en deux distributions)

**20% de Data Drift** : pas besoin de ré-entraîner le modèle en cas de nouvelles données

# Zoom sur les Drifted columns

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
client_bureau_balance_MONTHS_BALANCE_count_max	num			Detected	Wasserstein distance (normed)	0.543471
client_bureau_balance_MONTHS_BALANCE_count_mean	num			Detected	Wasserstein distance (normed)	0.524569
client_bureau_balance_MONTHS_BALANCE_sum_min	num			Detected	Wasserstein distance (normed)	0.329017
client_bureau_balance_MONTHS_BALANCE_mean_mean	num			Detected	Wasserstein distance (normed)	0.284277
client_bureau_balance_MONTHS_BALANCE_sum_mean	num			Detected	Wasserstein distance (normed)	0.265408
client_bureau_balance_STATUS_C_sum_mean	num			Detected	Wasserstein distance (normed)	0.245253
client_bureau_balance_MONTHS_BALANCE_count_min	num			Detected	Wasserstein distance (normed)	0.1758
NAME_CONTRACT_TYPE	cat			Detected	Jensen-Shannon distance	0.14755

# Les features importances



Les drifted columns n'ont pas pesé sur le modèle.

# Déploiement du Modèle

1

Le modèle est déployé en production via une API, accessible aux applications de prêt.

2

Un pipeline de déploiement automatisé gère la compilation, les tests et le déploiement du modèle.

```
$ pytest test_unitaire.py
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.3.4, pluggy-1.5.0
rootdir: C:\Users\hp\Desktop\DA_LeroyM\Leroy Merlin\OpCR_LM\Projet_OCR\P7_Implémentez_modele_Scoring\Mod-le-de-Scori
ng
plugins: anyio-4.7.0
collected 1 item

test_unitaire.py . [100%]

===== warnings summary =====
```

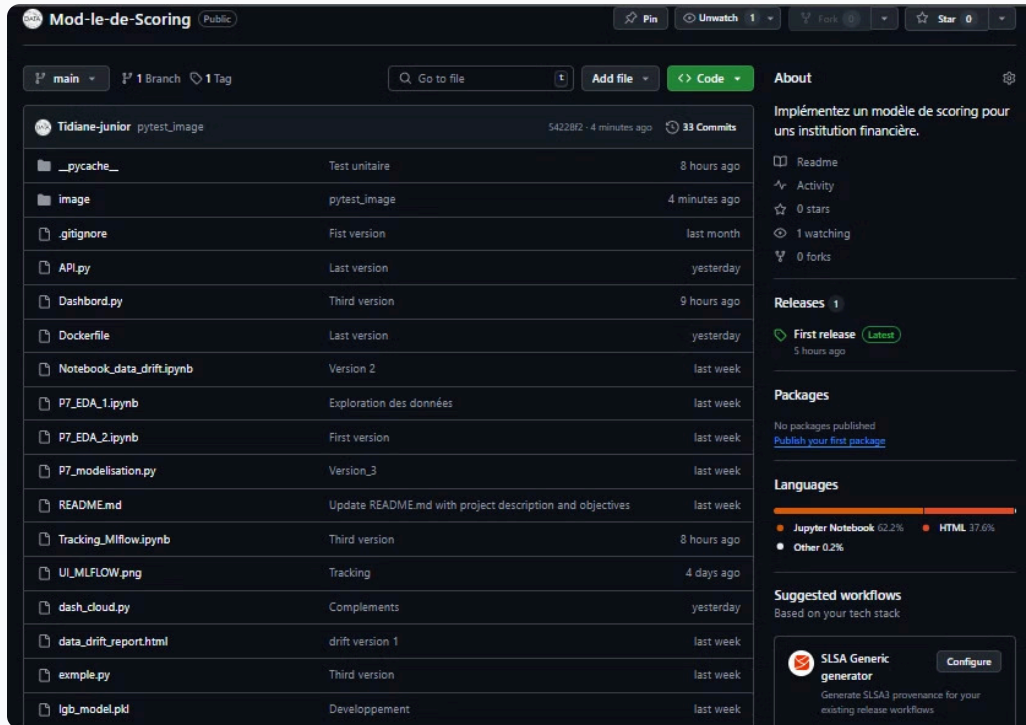
3

Le code source du modèle est versionné avec Git et hébergé sur Github, assurant une gestion collaborative et un suivi des modifications.



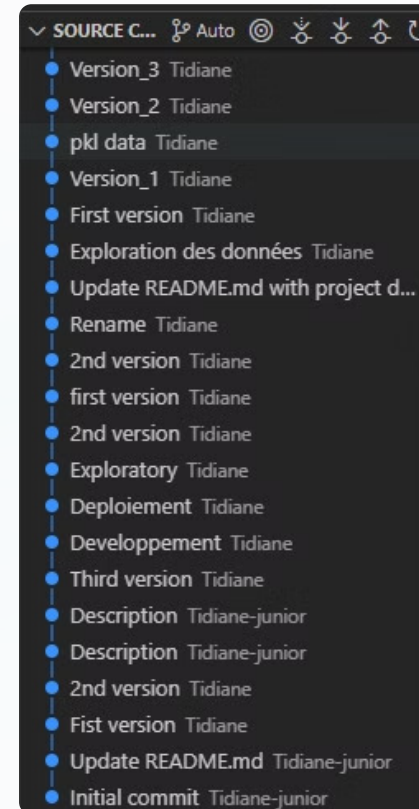
# Versionnage du Modèle

## Dossier GitHub

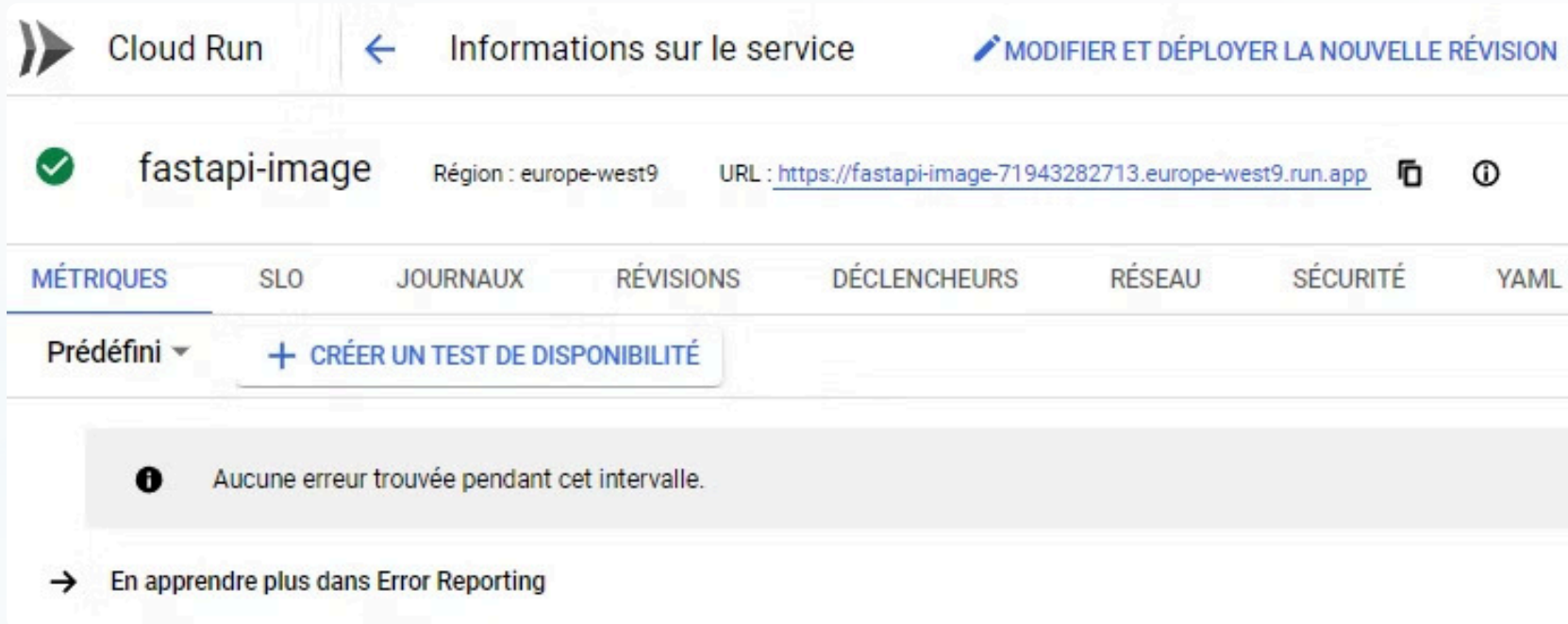


Lien GitHub : <https://github.com/Tidiane-junior/Mod-le-de-Scoring>

## Les commits



# Dashboard avec l'API déployé sur le cloud



The screenshot shows the Google Cloud Run dashboard for a service named 'fastapi-image'. At the top, there's a navigation bar with 'Cloud Run' and 'Informations sur le service', along with a link to 'MODIFIER ET DÉPLOYER LA NOUVELLE RÉVISION'. Below this, the service name 'fastapi-image' is displayed with a green checkmark, indicating it's running. The region is 'europe-west9' and the URL is 'https://fastapi-image-71943282713.europe-west9.run.app'. A tab bar below shows various metrics: MÉTRIQUES (selected), SLO, JOURNAUX, RÉVISIONS, DÉCLENCHERS, RÉSEAU, SÉCURITÉ, and YAML. Under the 'MÉTRIQUES' tab, there's a 'Prédéfini' dropdown and a button to '+ CRÉER UN TEST DE DISPONIBILITÉ'. A message box states 'Aucune erreur trouvée pendant cet intervalle.' with an information icon. At the bottom, there's a link to 'En apprendre plus dans Error Reporting'.

Cloud Run

← Informations sur le service

MODIFIER ET DÉPLOYER LA NOUVELLE RÉVISION

✓ fastapi-image Région : europe-west9 URL : <https://fastapi-image-71943282713.europe-west9.run.app>

MÉTRIQUES SLO JOURNAUX RÉVISIONS DÉCLENCHERS RÉSEAU SÉCURITÉ YAML

Prédéfini + CRÉER UN TEST DE DISPONIBILITÉ

Aucune erreur trouvée pendant cet intervalle.

→ En apprendre plus dans Error Reporting

Lien API cloud : <https://fastapi-image-71943282713.europe-west9.run.app>

# Conclusion

- Définir et mettre en œuvre un pipeline d'entraînement des modèles
- Définir la stratégie d'élaboration d'un modèle d'apprentissage supervisé
- Évaluer les performances des modèles
- Mettre en œuvre un logiciel de version de code (Git & GitHub)
- Concevoir un déploiement d'un moteur d'inférence sur une plateforme Cloud

**Merci de votre attention**