Implémenter un modèle de scoring

29/03/2024

Introduction

- “Prêt à dépenser”: propose des prêts à la consommation à des clients n’ayant pas ou peu d’historique de prêt

- Mission => outil de scoring
  - Calcul probabilité de défaut
  - Classification

- Démarche globale:
  - Élaborer un modèle de scoring
  - Déployer le modèle sous forme d’une API Flask
  - Intégrer et optimiser le système MLOps

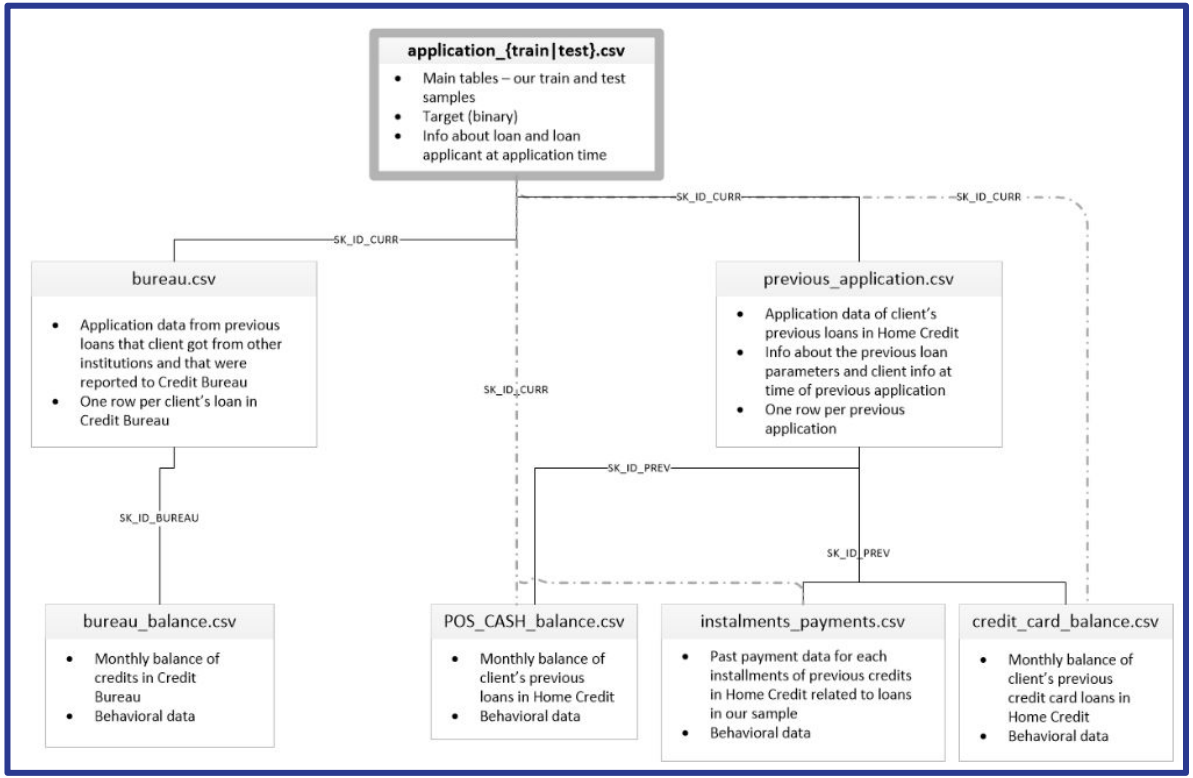
Description du jeu de données

- 7 fichiers
  - données comportementales
  - données historiques autres établissements …
- Application_test:
  - pour vérifier absence de data leakage
  - pour analyse du data drift

- Nettoyage et features engineering

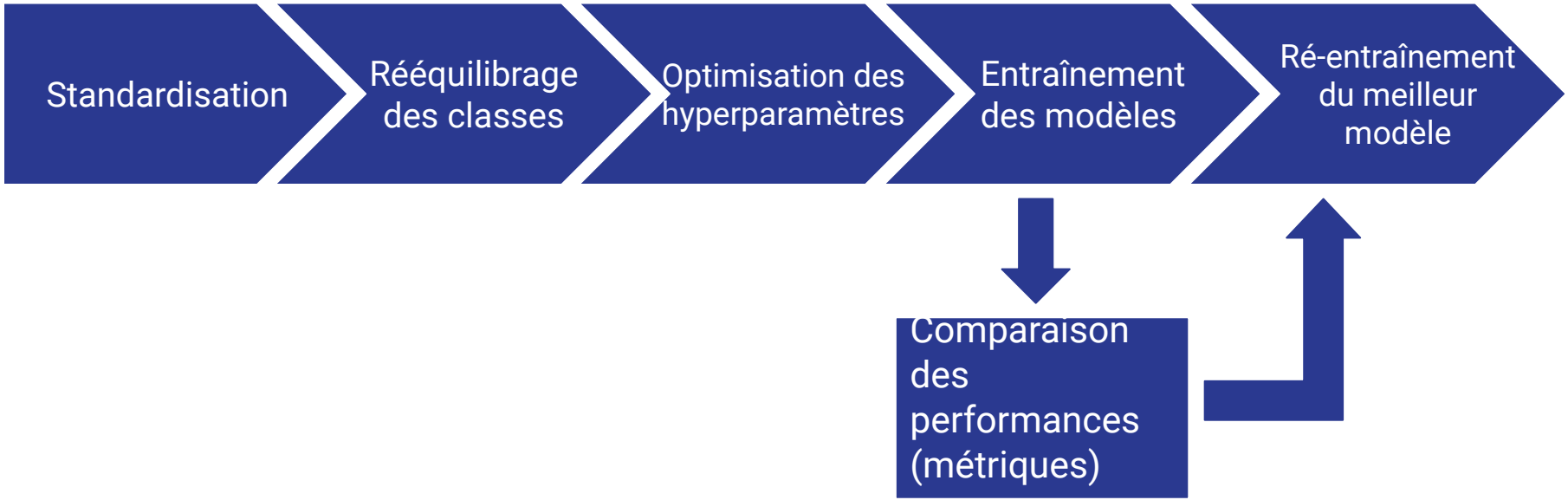
  => Kernels Kaggle adaptés

- Dataframe obtenu (307497, 815)



Partie 1: Modélisation

Démarche de modélisation



Rééquilibrage des classes avec Smote

- Déséquilibre de classes: inégalement représentées, risques de biaiser le modèle

- Smote: Suréchantillonnage des Minorités Synthétiques (SMOTE génère de nouvelles observations synthétiques pour la classe minoritaire en se basant sur les voisins les plus proches)

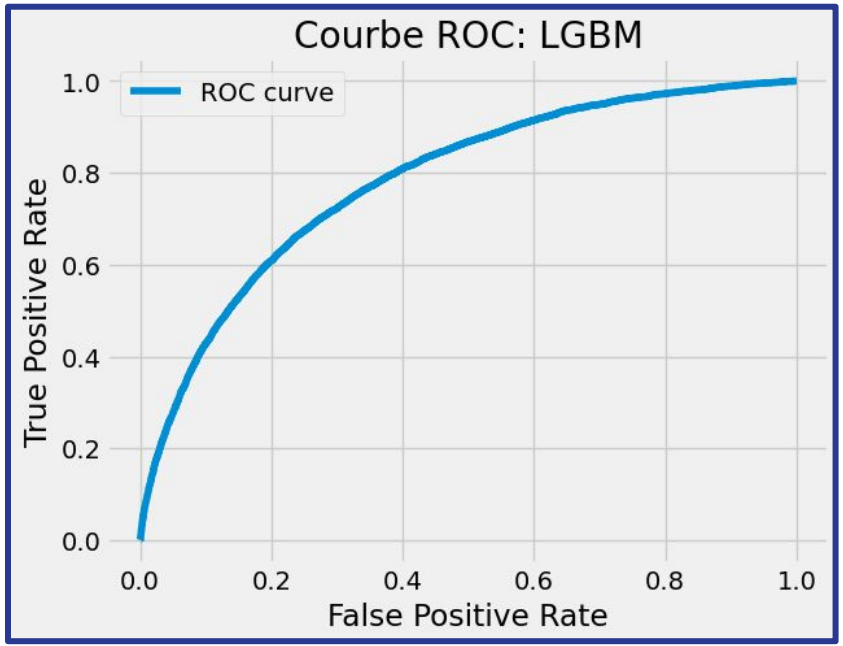
Choix des modèles

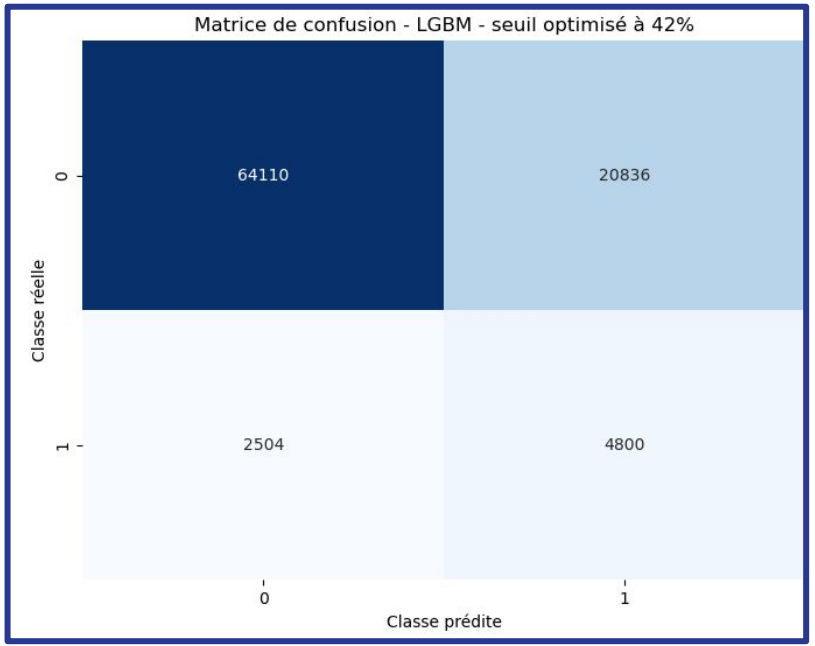
- **Baseline**: Régresseur naïf

- **Modèle linéaire:** Régression logistique
  - hyperparamètres:
    - C: régularisation (pénalise certaines features pour empêcher overfitting)
    - Penalty (l1, l2 ou ElasticNet)

- **Modèle non linéaire:** LGBM
  - algo arbres de décision
  - construction arbres feuille par feuille
  - hyperparamètres:
    - n_estimators
    - learning_rate
    - num_leaves

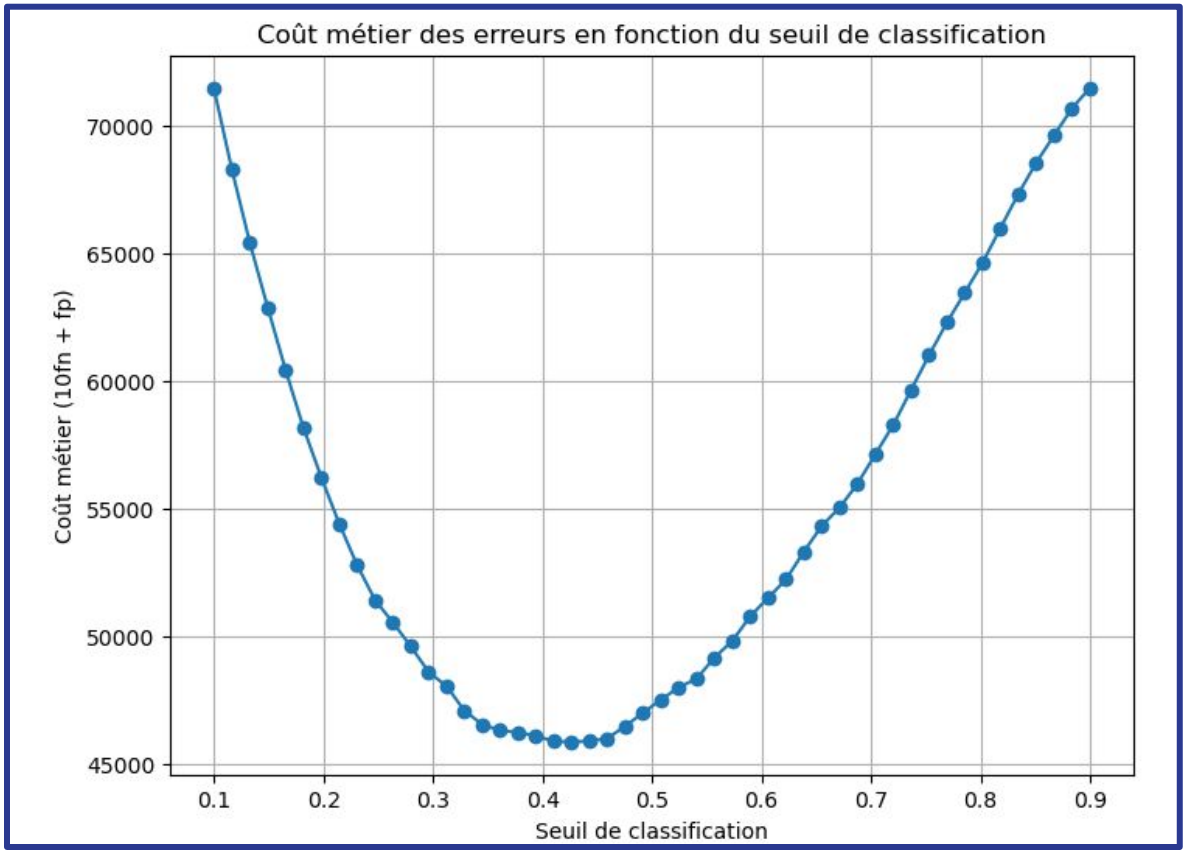
Choix des mesures - AUC





- **TPR= Rappel** = Sensibilité = tp/(tp+fn) => capacité à détecter difficulté de remboursement (Proportion d'échantillons positifs que l'on a classés positifs)
- FPR= 1 - Spécificité = 1 - (TN/(FP + TN)) => Spécificité: taux de vrais négatifs

Choix des mesures - Coût métier des erreurs



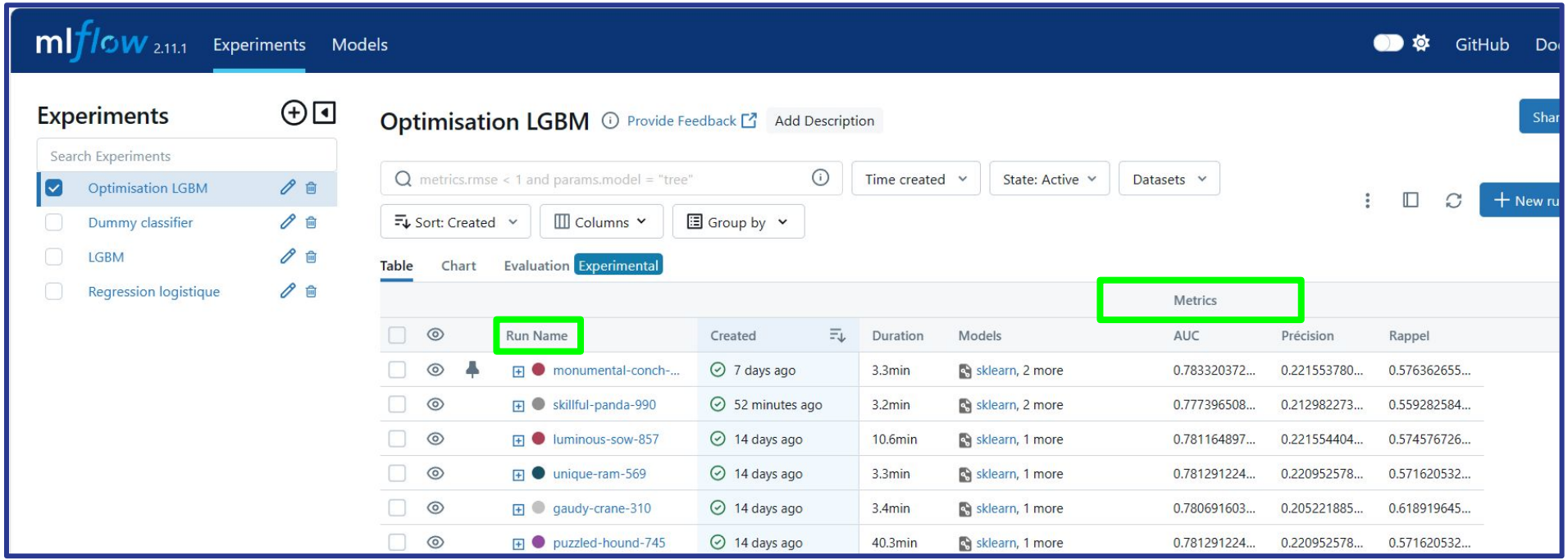
- **Les FN coûtent 10 fois plus que les FP**

- Création d'une fonction:

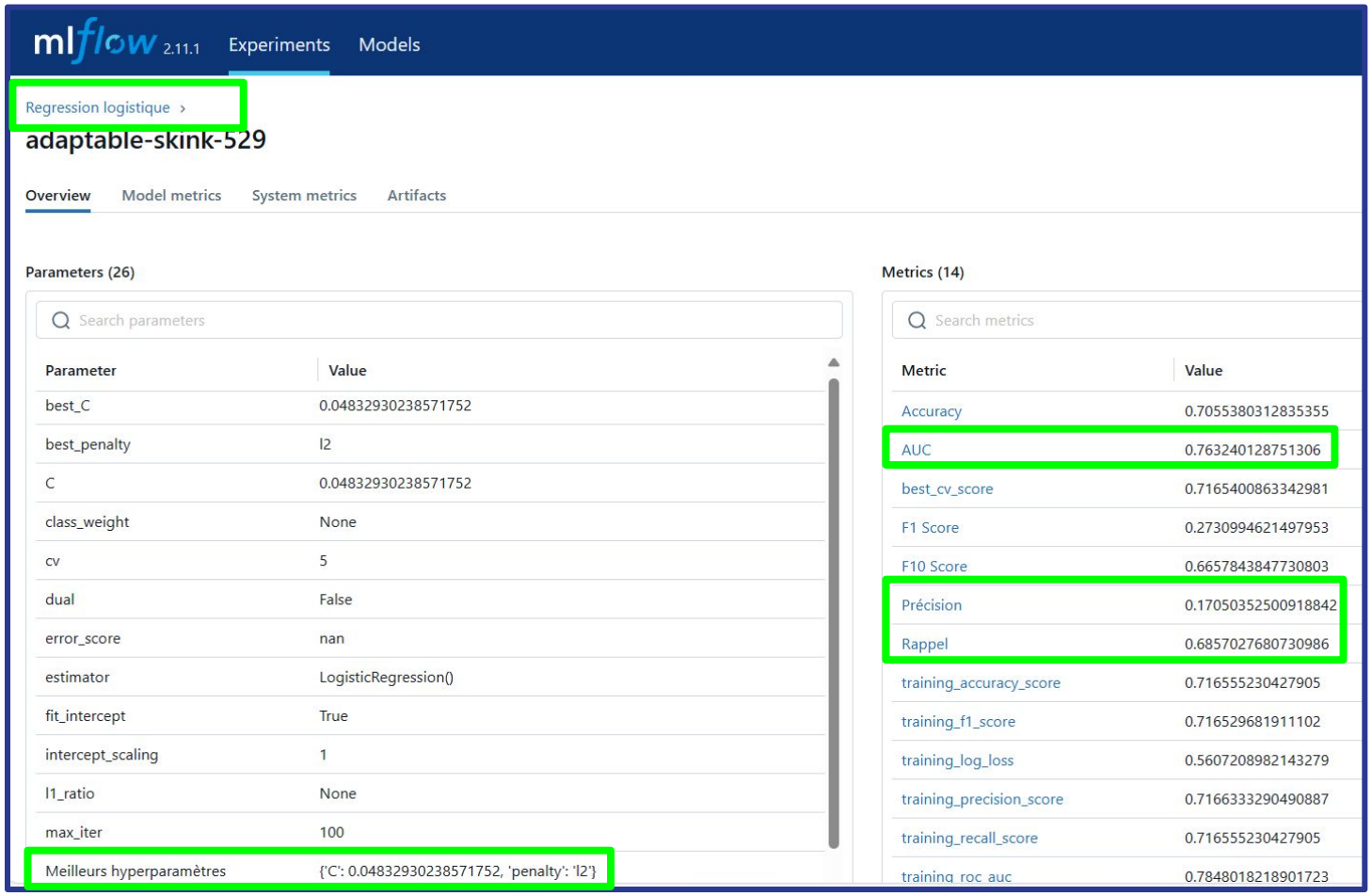
  coût=10 FN+ FP

- Optimiser le seuil de classification pour minimiser la fonction de coût

Tracking ML Flow



Tracking ML Flow

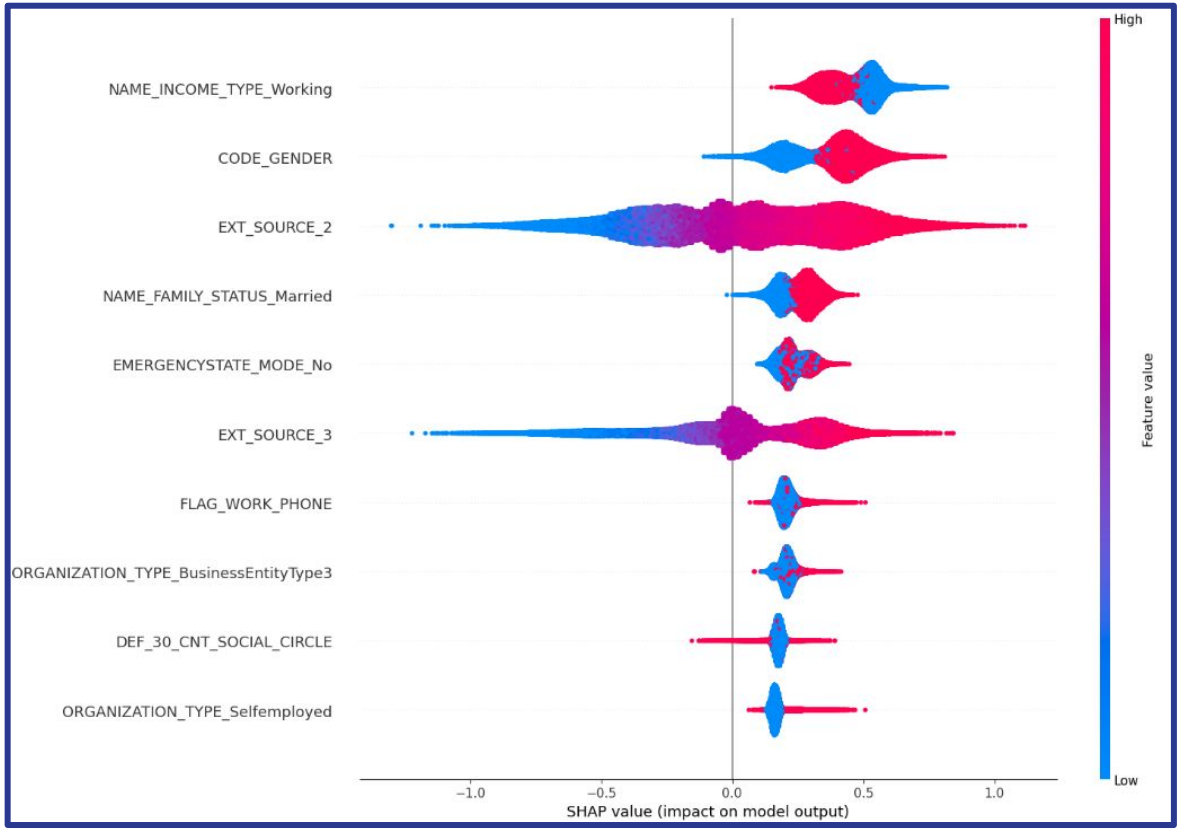


Comparaison des performances

Modèle	Hyperparamètres optimisés	AUC	Rappel	Précision	Coût Métier (à minimiser)	Temps de train
Dummy classifier	strategy=stratified	0.497	0.498	0.079	78295	9 sec
Régression logistique	C=0.048 penalty=l2	0.763	0.685	0.17	48392	8 min
LGBM	learning_rate=0.01, n_estimators=1850, num_leaves=85 max_depth	**0.781**	0.608	0.21	**45876**	3 min

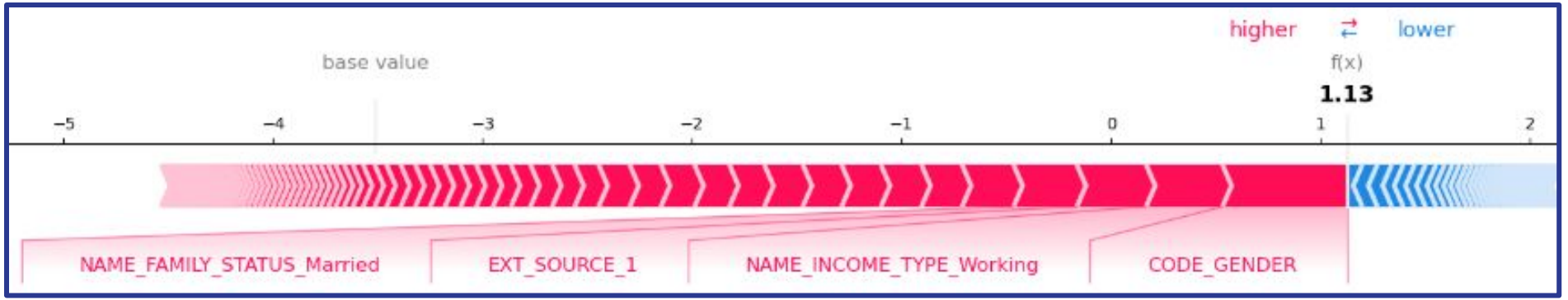
Analyse de l’importance des variables globale avec SHAP

- **SHAP**: Contribution de variables à la différence entre la valeur prédite par le modèle et la moyenne des prédictions
- Variables les plus importantes, qui ont le plus d’impact sur les prédictions



Analyse de l’importance des variables locale avec SHAP

- **SHAP** en local: Pour un prêt spécifique, affichage des features qui ont le plus impacté la prédiction

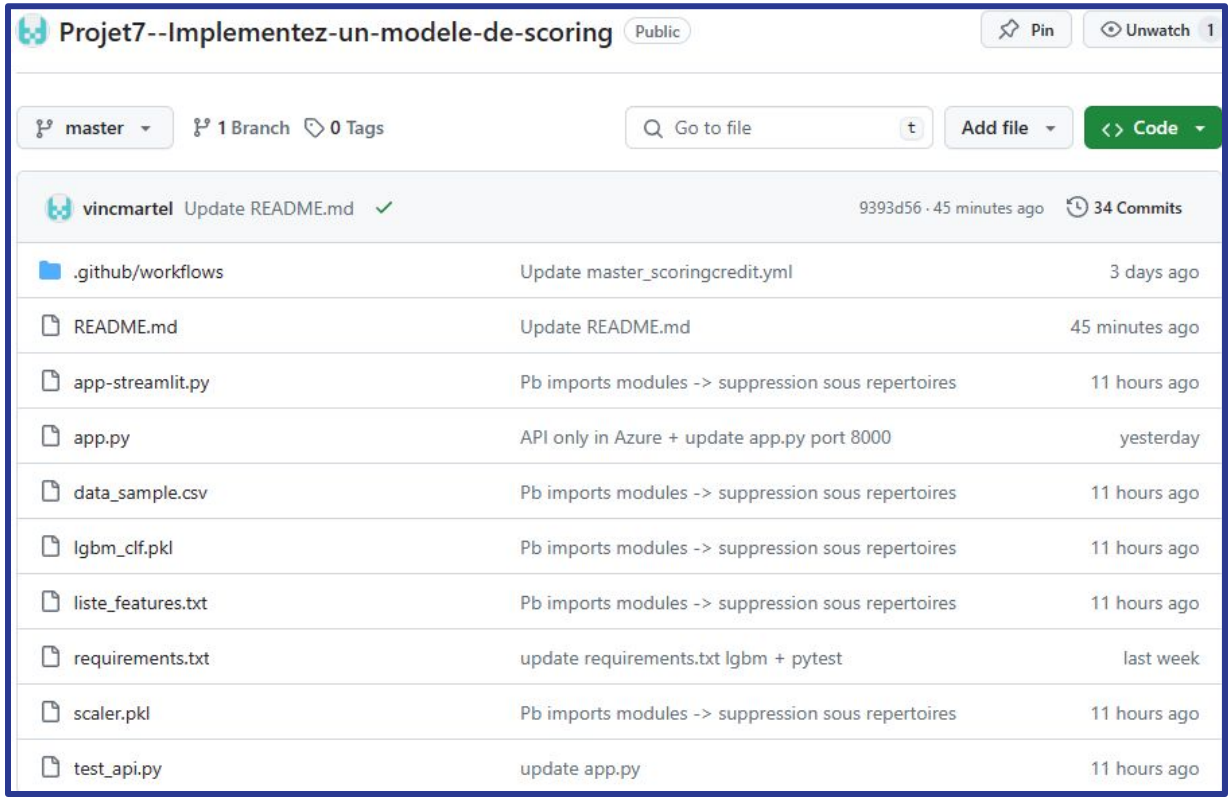


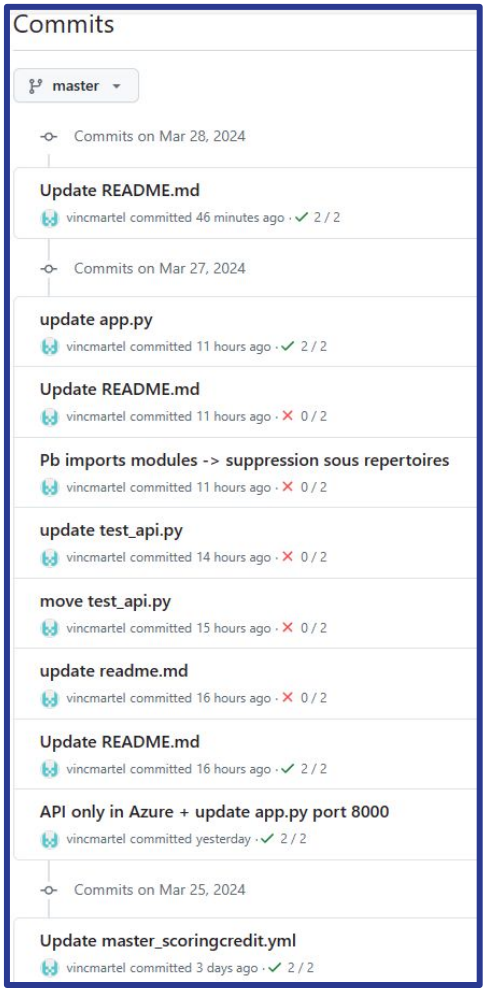
Partie 2: Pipeline de déploiement

Outils plateforme MLOps

Dépot Github

[vincmartel/Projet7--Implementez-un-modele-de-scoring (github.com)](https://github.com/vincmartel/Projet7--Implementez-un-modele-de-scoring)



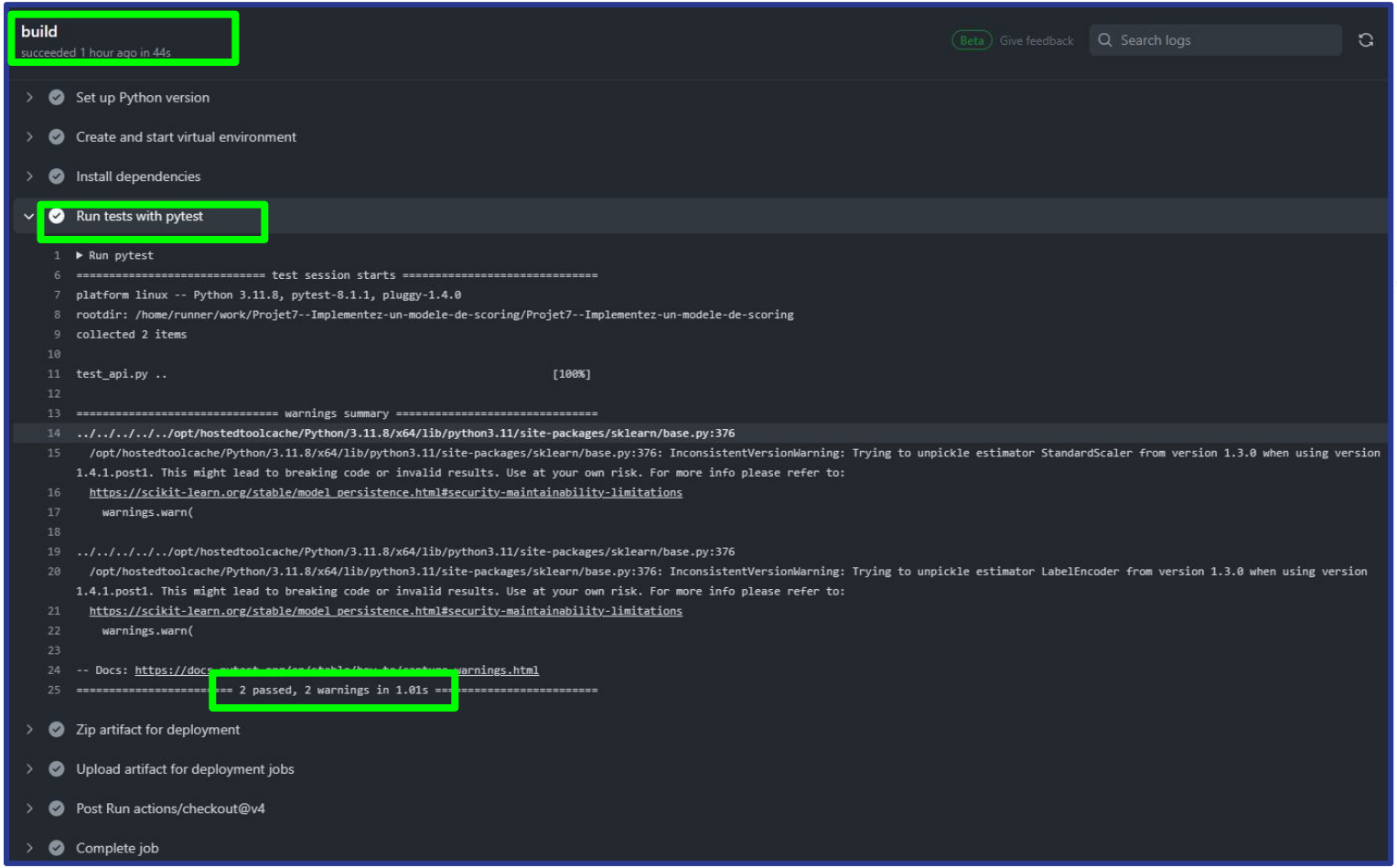


Github Actions: build et deploy

- Workflow -> définition des étapes de build et deploy dans fichier.yml

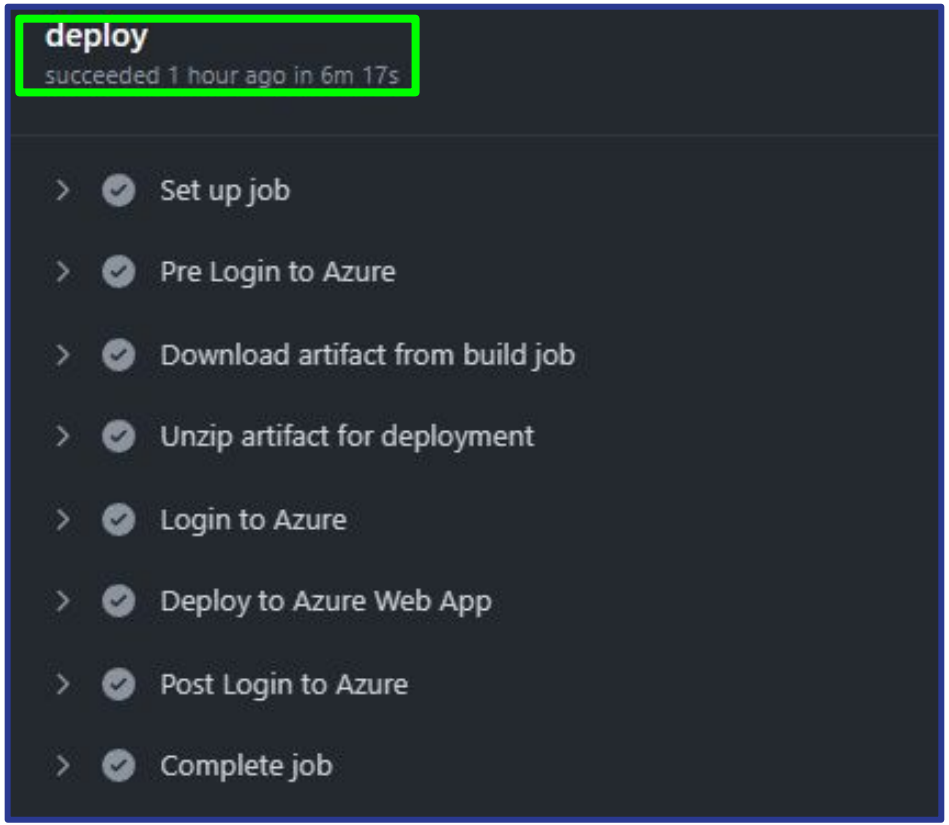

```yaml
on:
  push:
    branches:
      - master
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python version
        uses: actions/setup-python@v1
        with:
          python-version: '3.11'

      - name: Create and start virtual environment
        run: |
          python -m venv venv
          source venv/bin/activate

      - name: Install dependencies
        run: pip install -r requirements.txt

      - name: Run tests with pytest
        run: pytest

      - name: zip artifact for deployment
        run: zip release.zip ./* -r

      - name: Upload artifact for deployment jobs
        uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: |
            release.zip
            !venv/

  deploy:
    runs-on: ubuntu-latest
    needs: build
    environment:
      name: 'Production'
```

Github Actions: build



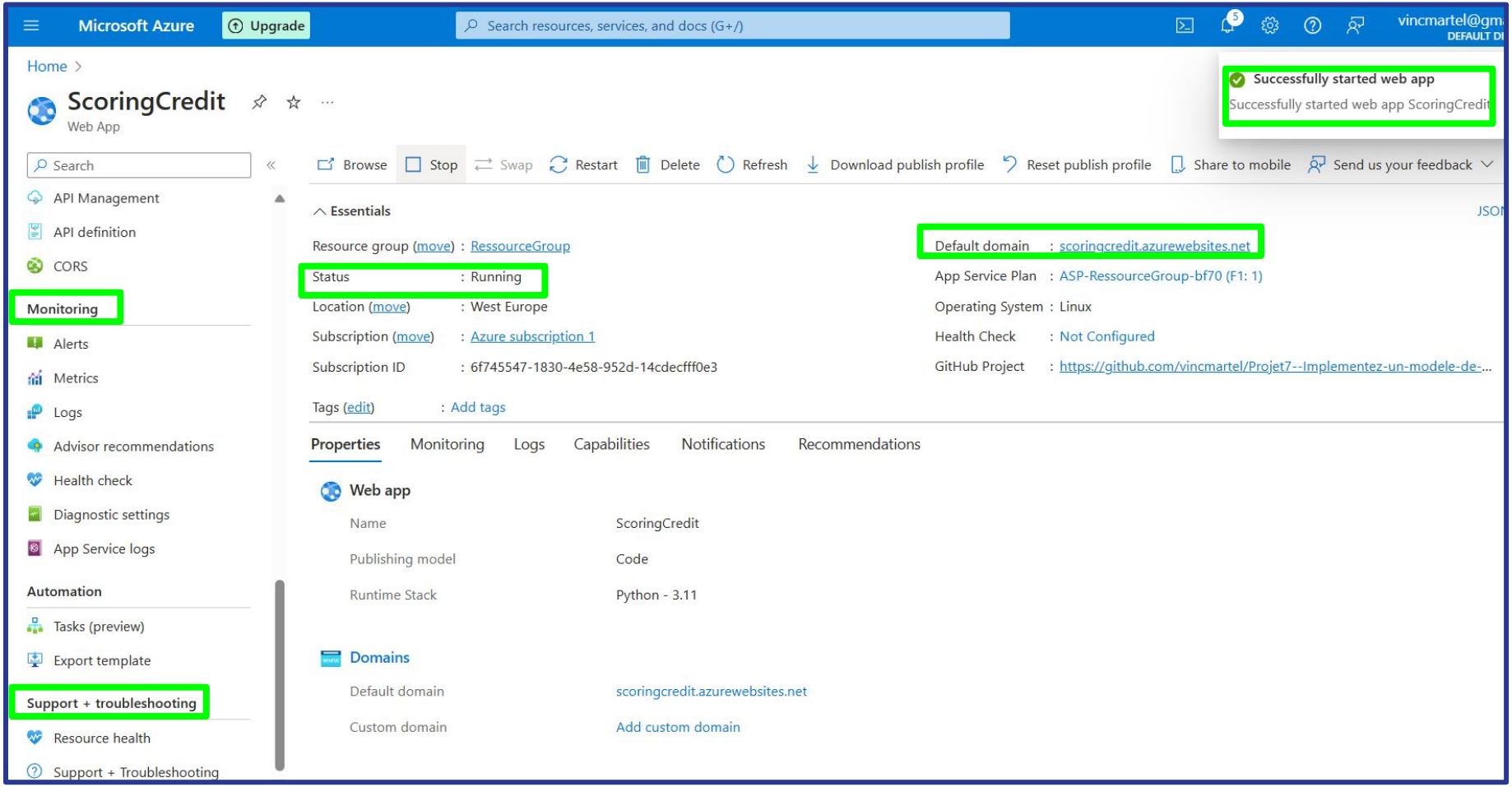
pytest

Github Actions: deploy



Microsoft Azure

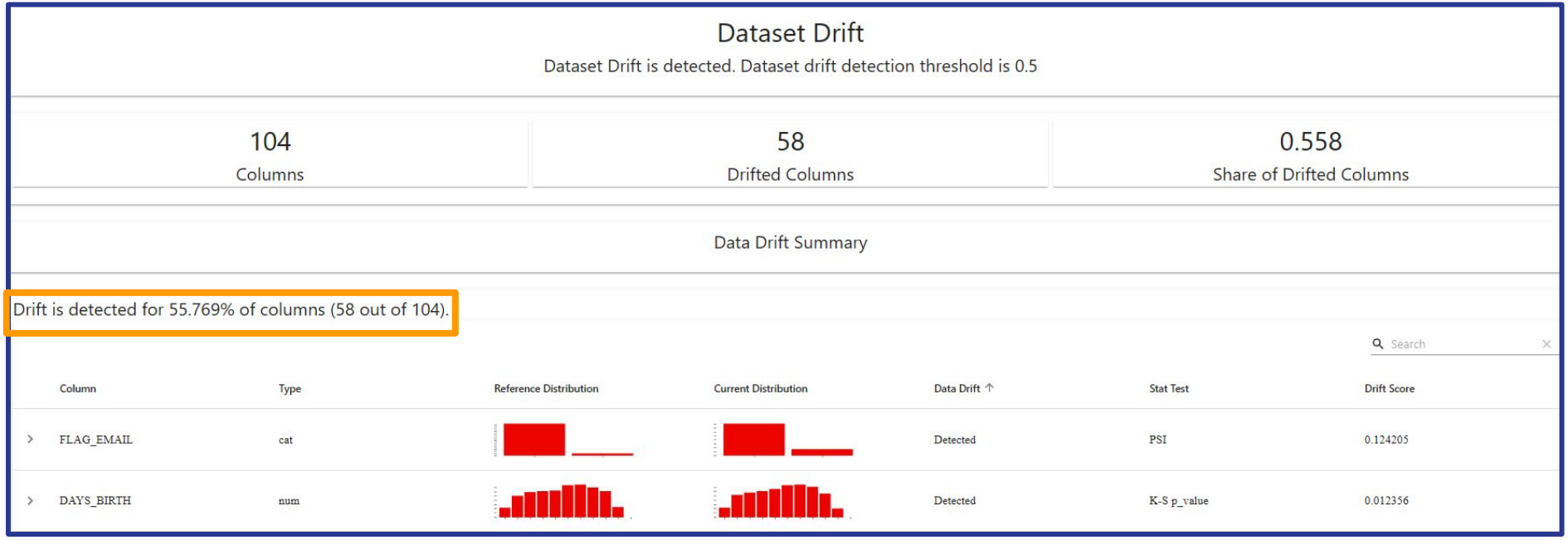
Azure: management de l'API



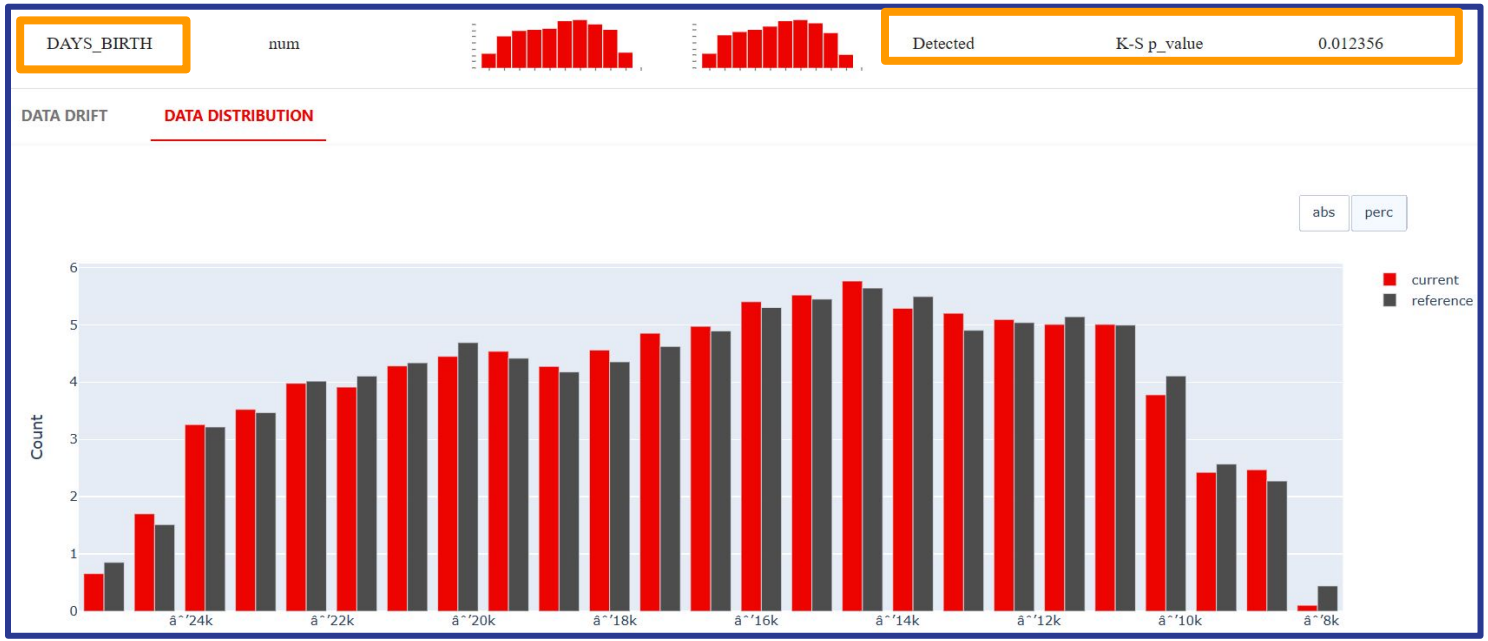
Partie 3: Data drift

Analyse du data drift

- Définition: variation des données en production VS données d'entraînement (risque diminution performances)
- But: suivre l'efficacité d'un modèle en production
- Data drift entre application_train et application_test



Analyse du data drift



- Actions d'amélioration possibles:
  - Nouvelles données d'entraînement
  - Sélection de features
  - Optimisation hyperparamètres, ré-entraînement du modèle

Partie 4: Démo scoring crédit

localhost:8501

Décision d'attribution de crédit

Identifiant du prêt

100002

Résultat

Probabilité de défaut : 80% (La limite acceptable est 42%)

Décision : prêt refusé

Décision d'attribution de crédit

Identifiant du prêt

100003

Résultat

Probabilité de défaut : 11 % (La limite acceptable est 42%)

Décision : prêt accordé