جامعة شعيب الدكالي
Université Chouaib Doukkali

Université Chouaib Doukkali
École Nationale des Sciences Appliquées El Jadida Master Science Des Données Et Intelligence Artificielle
Module : MlOps

ENSA EL JADIDA

# Lab 1 : Du notebook au mini-système production-ready

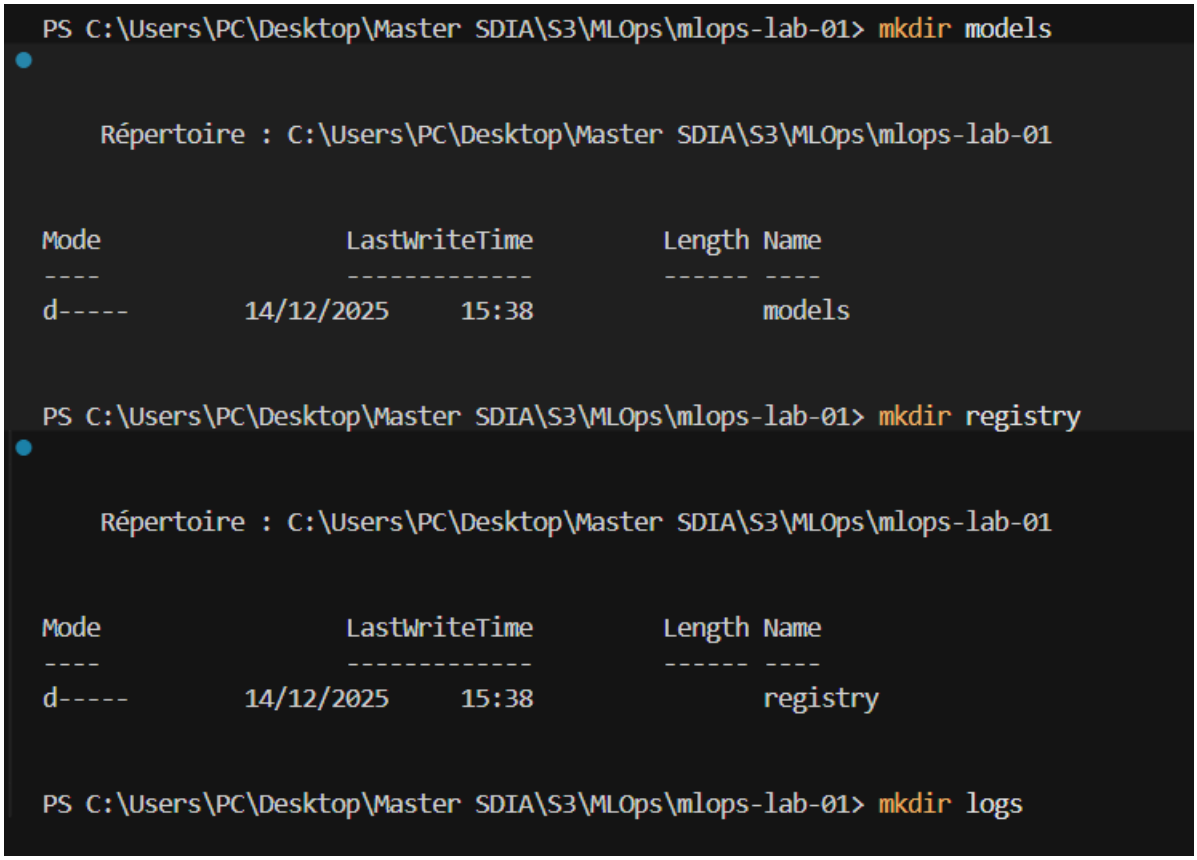
Réalisée par :

El Hamzaoui Aya

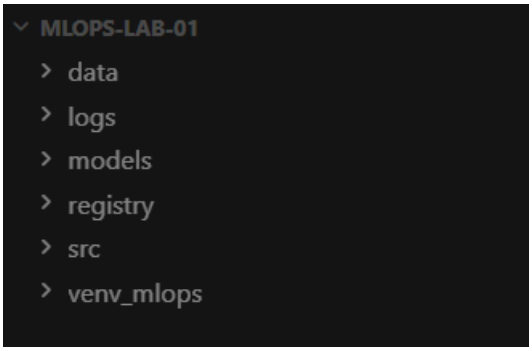
## 1. Introduction

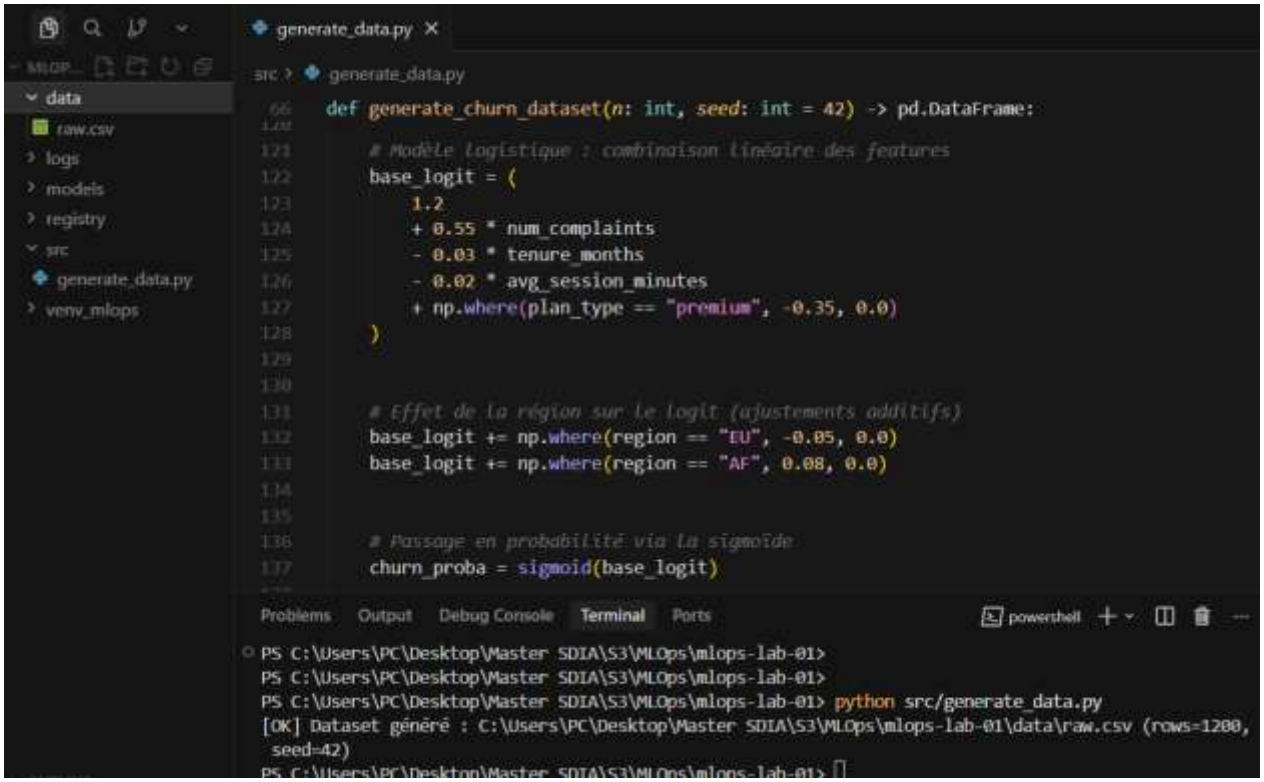
Le but de ce lab est de mettre en place un pipeline MLOps minimal pour un modèle de prédiction de churn. Les objectifs principaux sont :

1. Préparer et entraîner un modèle scikit-learn sur un dataset de churn.
2. Evaluer et versionner le modèle via un registry local (metadata.json et current_model.txt).
3. Déployer le modèle via une API FastAPI /predict.
4. Monitorer les requêtes pour détecter un **data drift**.
5. Gérer les versions et effectuer des rollback en cas de problème.



```
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> echo "" > registry\curre
nt_model.txt
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> python -m venv venv_mlop
s
ivate.ps1
(venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> pip install
 --upgrade pip
Requirement already satisfied: pip in c:\users\pc\desktop\master sdia\s3\mlops\mlo
ps-lab-01\venv_mlops\lib\site-packages (25.3)
(venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01>
(venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> pip install
 pandas numpy scikit-learn fastapi uvicorn joblib
Collecting pandas
  Using cached pandas-2.3.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
Collecting numpy
  Using cached numpy-2.3.5-cp313-cp313-win_amd64.whl.metadata (60 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.8.0-cp313-cp313-win_amd64.whl.metadata (11 kB)
Collecting fastapi
  Using cached fastapi-0.124.4-py3-none-any.whl.metadata (30 kB)
Collecting uvicorn
  Using cached uvicorn-0.38.0-py3-none-any.whl.metadata (6.8 kB)
Collecting joblib
  Using cached joblib-1.5.2-py3-none-any.whl.metadata (5.6 kB)
```

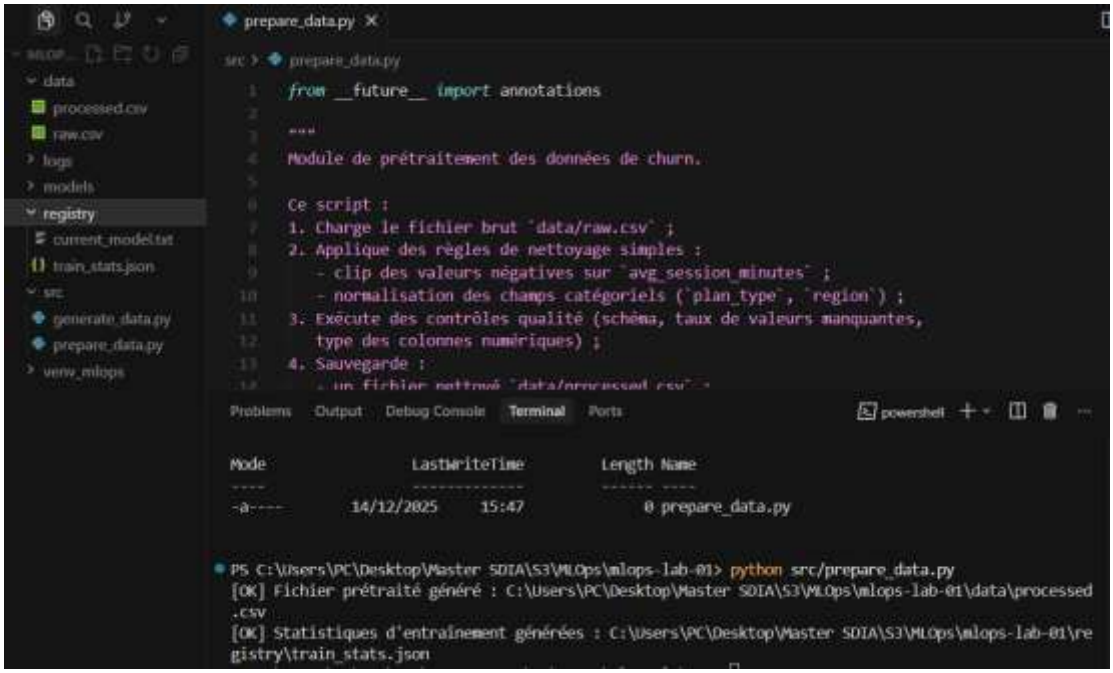


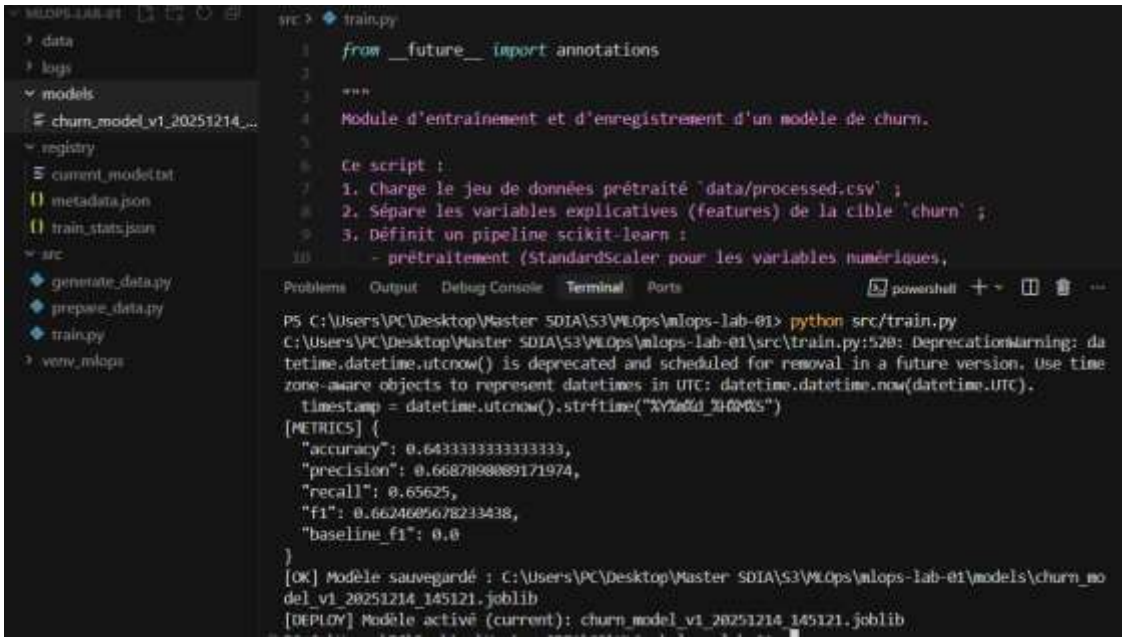



## 2. Préparation des données et statistiques

### 2.1 Prétraitement

- Les données brutes ont été nettoyées et transformées via prepare_data.py.
- Les features numériques ont été standardisées et les catégorielles encodées.
- Les statistiques d'entraînement (moyenne, écart-type) ont été sauvegardées dans :



## 3. Entraînement et évaluation du modèle (evaluate.py)

### 3.1 Pipeline scikit-learn

Le pipeline inclut :

- **StandardScaler** sur les features numériques : tenure_months, num_complaints, avg_session_minutes.
- **OneHotEncoder** sur les features catégorielles : plan_type, region.
- **LogisticRegression** comme classificateur.

### 3.2 Découpage train/test

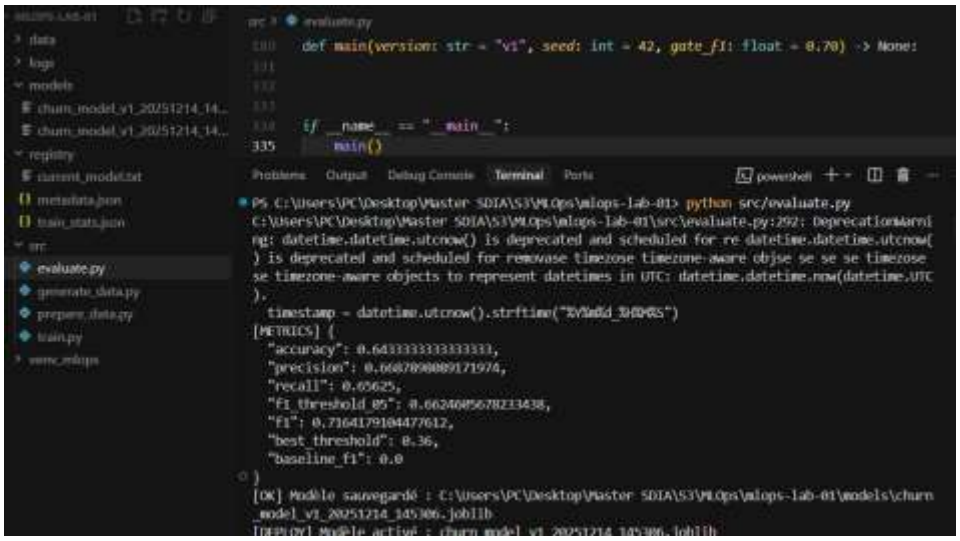
- Stratification sur la cible churn.
- Test set = 25% des données.

### 3.3 Tuning du seuil optimal

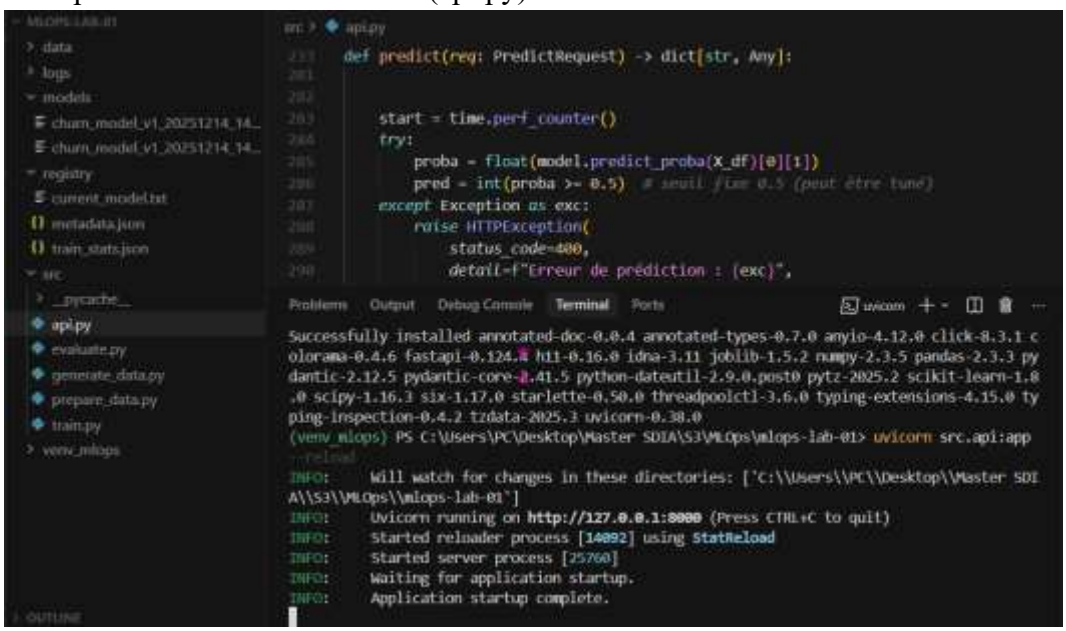
- La probabilité de churn est convertie en prédiction binaire selon un **seuil optimisé pour la F1**.
- Baseline triviale : prédire toujours 0 pour comparer.

### 3.4 Sauvegarde et registry

- Le modèle est sauvegardé dans models/ avec un timestamp et la version.
- Les métadonnées sont mises à jour dans registry/metadata.json.
- Si la F1 dépasse le **gate (0.70)** et bat la baseline, le modèle devient le current_model.txt.

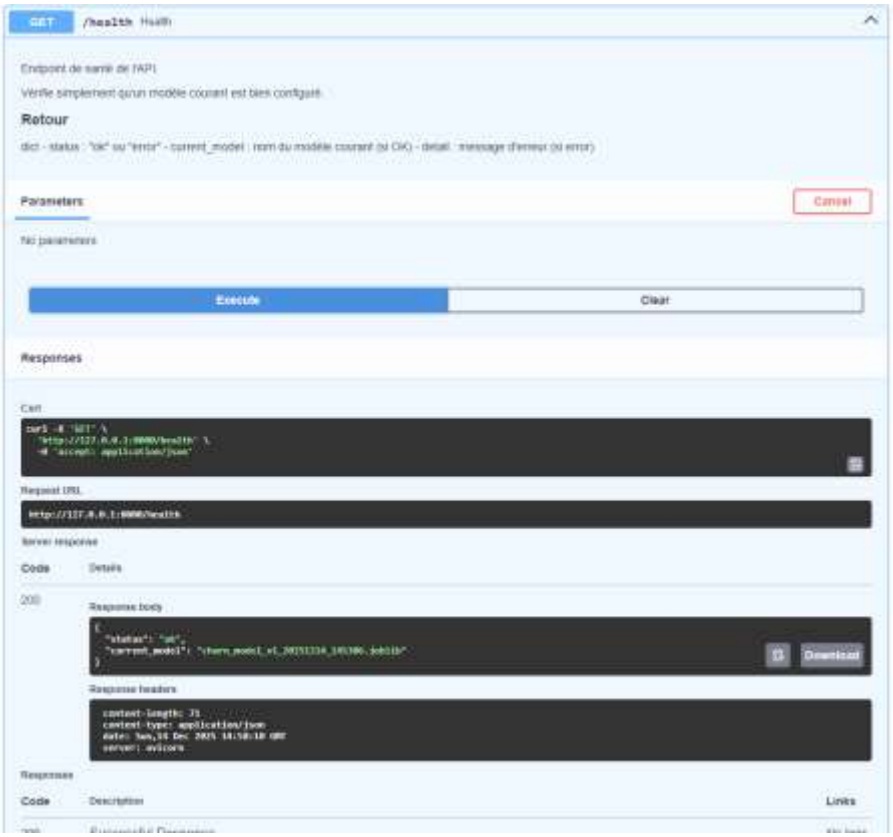


4. Déploiement via API FastAPI (api.py)



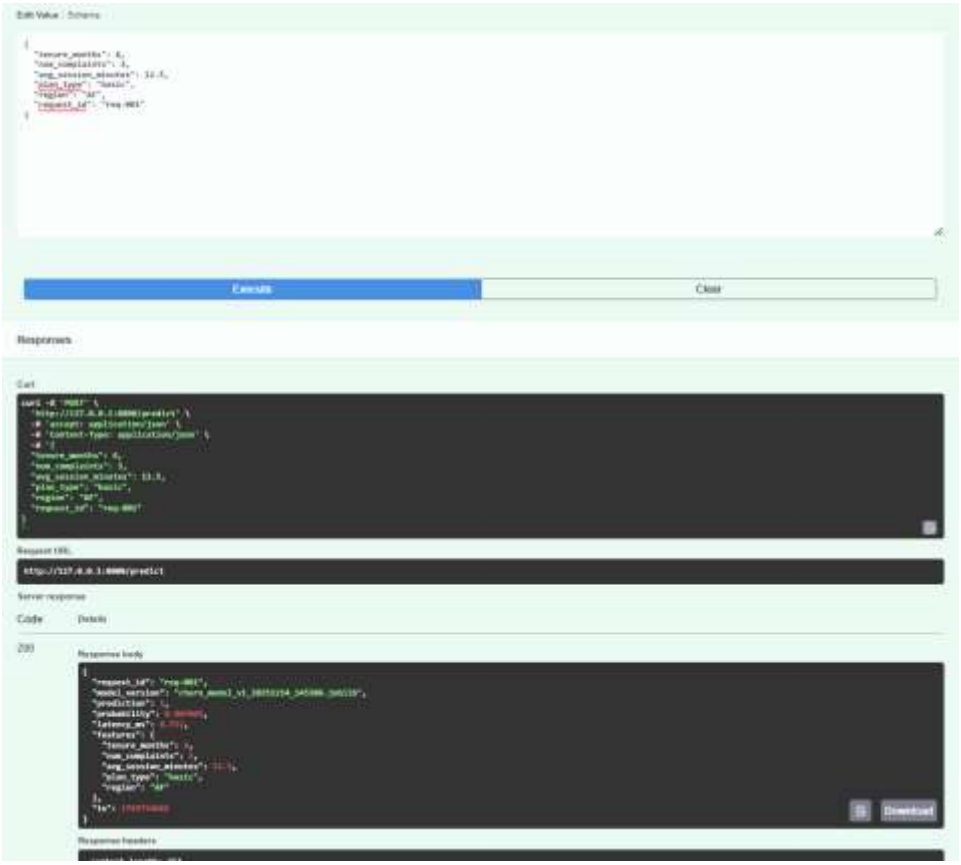

## 4.1 Endpoints

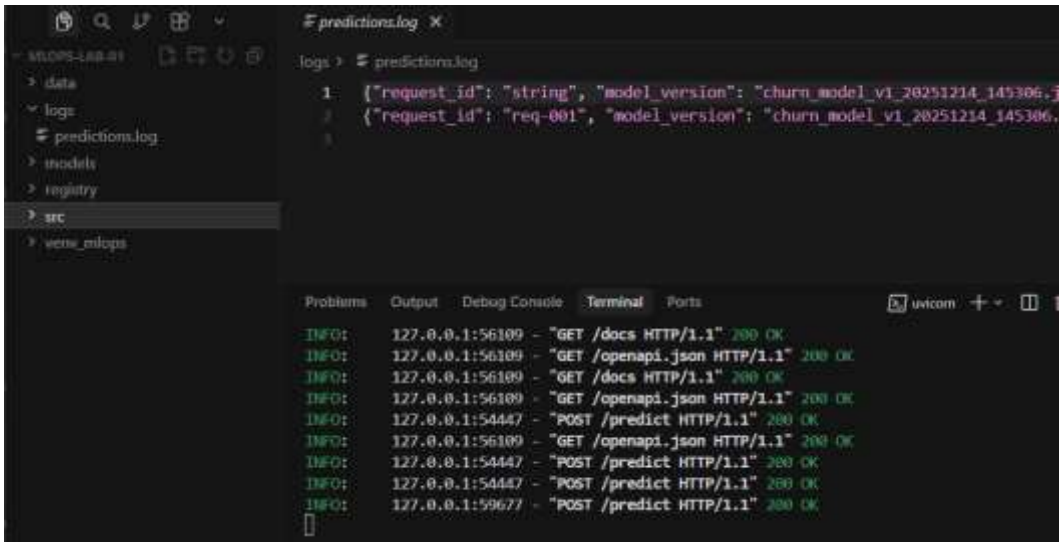
1. **GET /health**
    - Vérifie la présence du modèle courant et retourne son nom.



**2.POST /predict**

- Reçoit un JSON avec les features du client.
- Retourne :
    - prediction : 0 ou 1
    - probability : probabilité de churn
    - latency_ms : temps d'inférence
    - model_version : nom du modèle utilisé
    - request_id : si fourni
- Les requêtes sont journalisées dans logs/predictions.log.



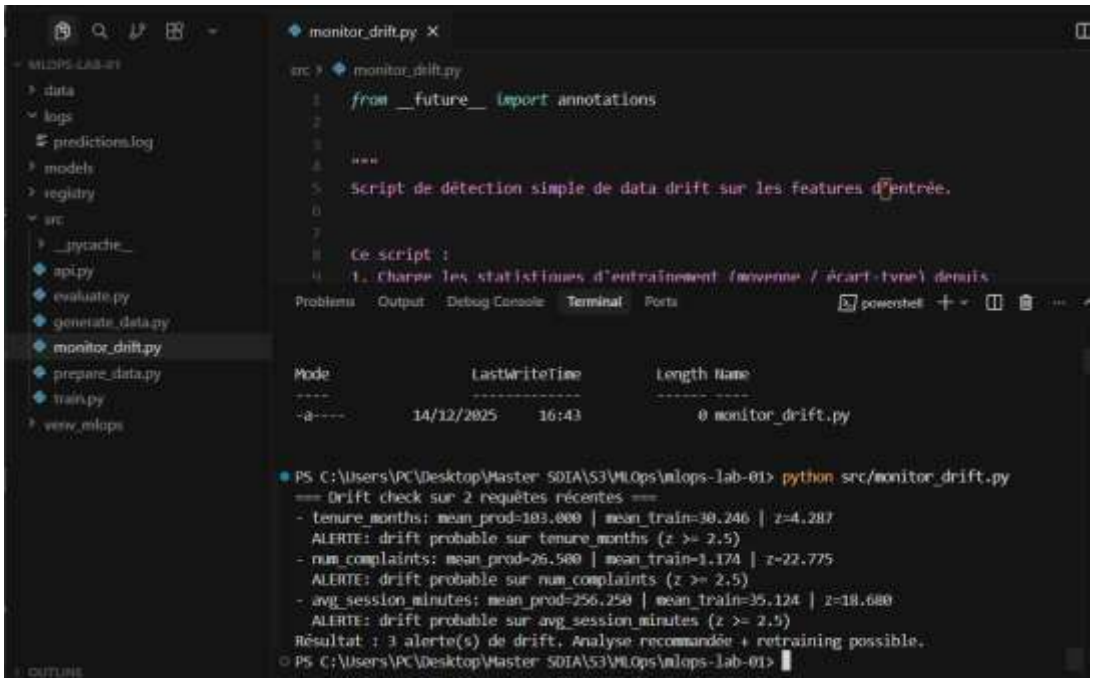



## 5. Monitoring du data drift (monitor_drift.py)

- Le script analyse les **N dernières requêtes** (par défaut 200).
- Calcule un score Z pour chaque feature numérique :

$$z = \frac{| \, mean_{prod} - mean_{train} \, |}{std_{train}}$$

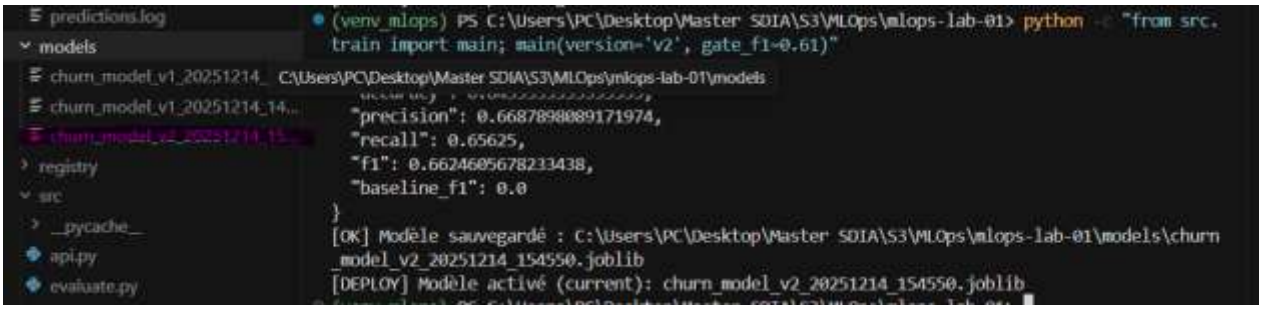
- Déclenche une alerte si $z \geq 2.5$.
- Peut envoyer un hook vers un outil externe si MONITORING_TOKEN est défini.



## 6. Gestion des versions et rollback (rollback.py)

### 6.1 Activation d'une nouvelle version

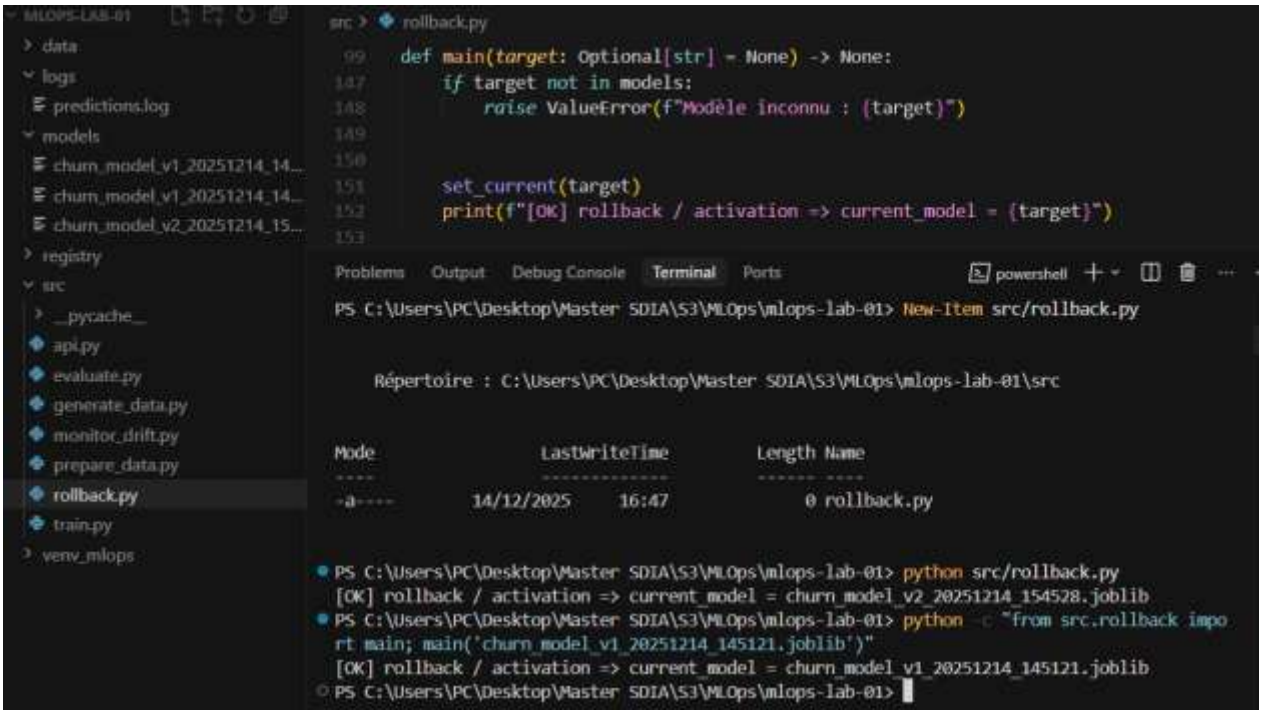
- Exemple pour entraîner et activer la version v2 :




### 6.2 Rollback automatique

- Active le **modèle précédent** en cas de problème :

Rollback vers un modèle précis



## 7. Conclusion

Ce lab illustre un **pipeline MLOps** :

1. Préparation des données et calcul des statistiques.
2. Entraînement, évaluation, tuning du seuil et enregistrement des métadonnées.
3. Déploiement via API avec logging des prédictions.
4. Monitoring simple de drift pour détecter les anomalies sur les features.
5. Gestion des versions et rollback pour maintenir la stabilité en production.

**Points clés :**

- L'importance du **gate F1** pour contrôler la qualité du modèle avant déploiement.
- La valeur des **logs de prédiction** pour monitorer la production et détecter un drift.
- La **gestion des versions** comme outil de sécurité et pédagogie MLOps.