

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> mkdir models
```

Répertoire : C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01

Mode	LastWriteTime	Length	Name
d----	14/12/2025 15:38		models

```
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> mkdir registry
```

Répertoire : C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01

Mode	LastWriteTime	Length	Name
d----	14/12/2025 15:38		registry

```
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> mkdir logs
```

```

PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> echo "" > registry\current_model.txt
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> python -m venv venv_mlops
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\mlops-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)

```

```

v MLOPS-LAB-01
> data
> logs
> models
> registry
> src
> venv_mlops

```

The screenshot shows a VS Code editor with a file named `generate_data.py` open. The file explorer on the left shows a project structure with folders `data`, `logs`, `models`, `registry`, `src`, and `venv_mlops`. The `src` folder is expanded, showing `generate_data.py`. The code in the editor is as follows:

```

66  def generate_churn_dataset(n: int, seed: int = 42) -> pd.DataFrame:
67
68      # Modèle logistique : combinaison linéaire des features
69      base_logit = (
70          1.2
71          + 0.55 * num_complaints
72          - 0.03 * tenure_months
73          - 0.02 * avg_session_minutes
74          + np.where(plan_type == "premium", -0.35, 0.0)
75      )
76
77      # Effet de la région sur le logit (ajustements additifs)
78      base_logit += np.where(region == "EU", -0.05, 0.0)
79      base_logit += np.where(region == "AF", 0.08, 0.0)
80
81      # Passage en probabilité via la sigmoïde
82      churn_proba = sigmoid(base_logit)

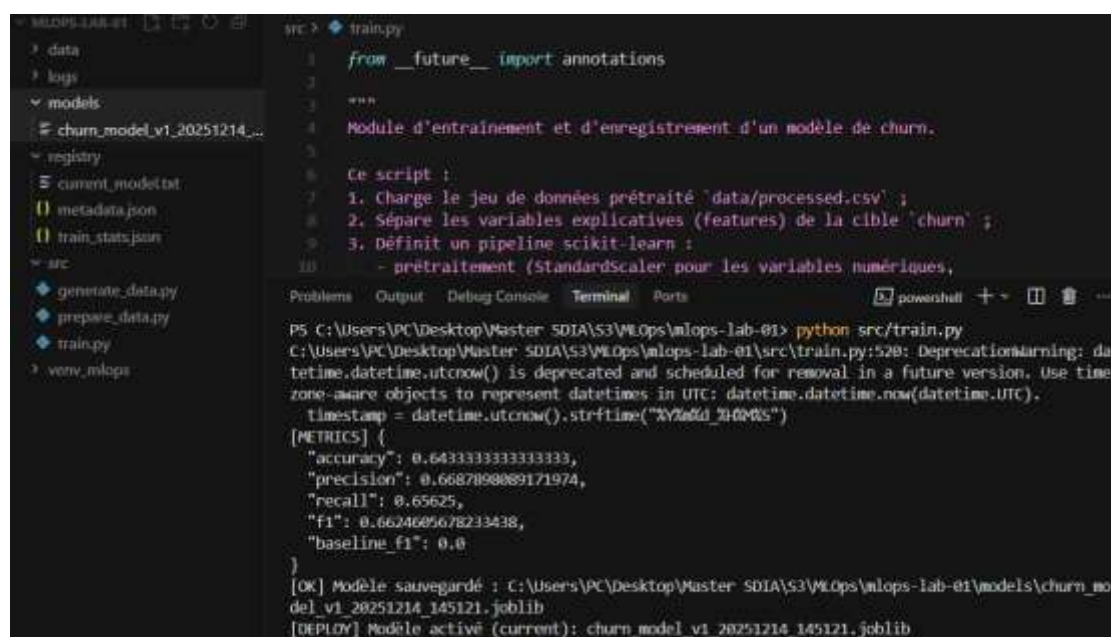
```

Below the editor, the terminal shows the execution of the script:

```

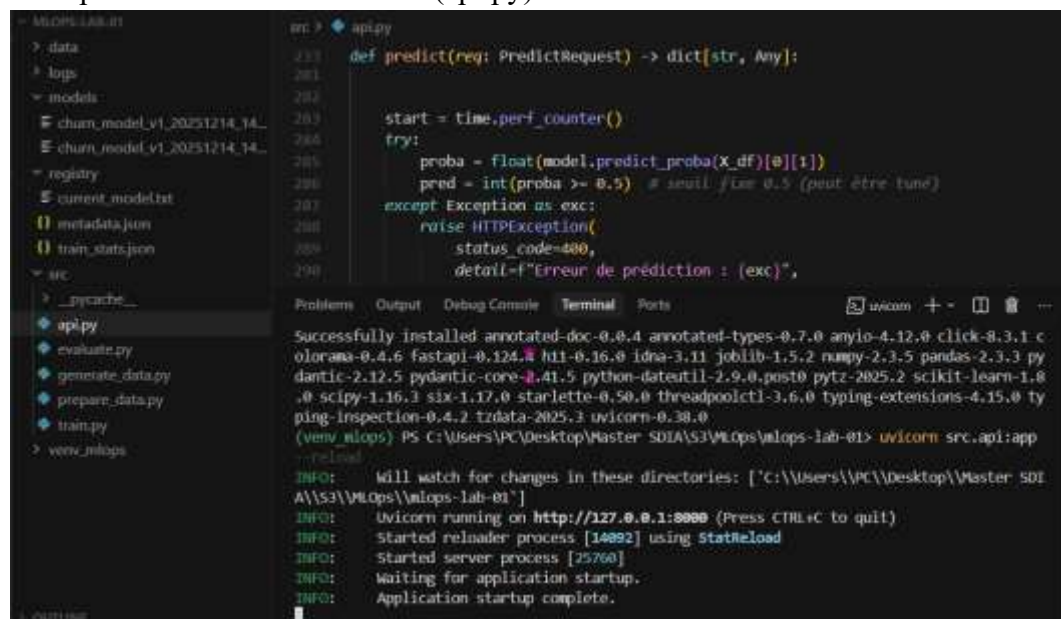
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01>
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01>
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01> python src/generate_data.py
[OK] Dataset généré : C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01\data\raw.csv (rows=1200, seed=42)
PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\mlops-lab-01>

```



```
src > evaluate.py
> data
180 def main(version: str = "v1", seed: int = 42, gate_fit: float = 0.70) -> None:
> logs
191
> models
192
193
194 # churn_model_v1_20251214_14...
195 # churn_model_v1_20251214_14...
> registry
196 # current_model.txt
197
198 # metadata.json
199 # train_stats.json
> src
200
201 # evaluate.py
202 # generate_data.py
203 # prepare_data.py
204 # train.py
205 # utils.py
```

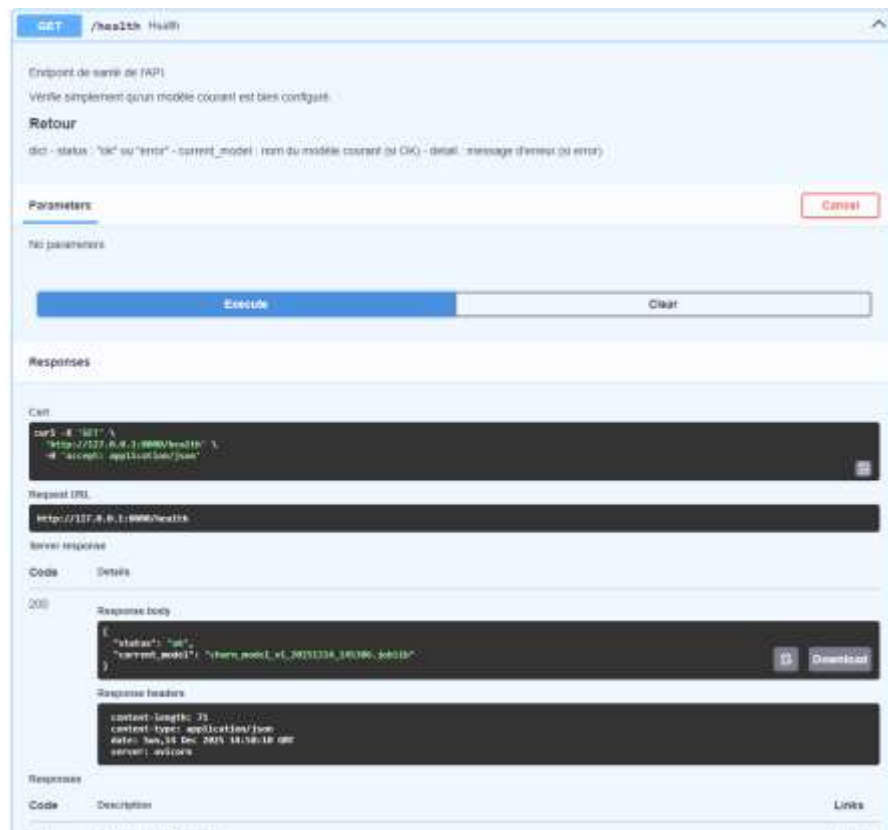
4. Déploiement via API FastAPI (api.py)



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.

The screenshot displays a REST client interface with the following sections:

- Body Value / Schema:** A JSON object representing client features:

```
{  "tenure_months": 6,  "tenure_months2": 3,  "avg_session_minutes": 32.5,  "plan_type": "basic",  "region": "us",  "request_id": "req-001"}
```
- Buttons:** "Execute" and "Clear".
- Request:** A cURL command:

```
curl -X POST -s \  -H "Host: api.example.com" \  -H "Content-Type: application/json" \  -d '{  "tenure_months": 6,  "tenure_months2": 3,  "avg_session_minutes": 32.5,  "plan_type": "basic",  "region": "us",  "request_id": "req-001"  }'
```
- Request URL:** `http://127.0.0.1:8080/predict`
- Server response:** A tabbed interface with "Code" and "Details". The "Code" tab shows the response body:

```
{  "request_id": "req-001",  "model_version": "model_v1_202304_04000_04010",  "prediction": 1,  "probability": 0.9999,  "latency_ms": 0.01,  "features": {    "tenure_months": 6,    "tenure_months2": 3,    "avg_session_minutes": 32.5,    "plan_type": "basic",    "region": "us"  }  }
```



```

# predictions.log
logs > # predictions.log
1  {"request_id": "string", "model_version": "churn_model_v1_20251214_145306.1"}
2  {"request_id": "req-001", "model_version": "churn_model_v1_20251214_145306.1"}
3

Problems Output Debug Console Terminal Ports
INFO: 127.0.0.1:56109 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:56109 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:56109 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:56109 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:54447 - "POST /predict HTTP/1.1" 200 OK
INFO: 127.0.0.1:56109 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:54447 - "POST /predict HTTP/1.1" 200 OK
INFO: 127.0.0.1:54447 - "POST /predict HTTP/1.1" 200 OK
INFO: 127.0.0.1:59677 - "POST /predict HTTP/1.1" 200 OK

```

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.

```

src > monitor_drift.py
1  from __future__ import annotations
2
3
4  """
5  Script de détection simple de data drift sur les features d'entrée.
6
7
8  Ce script :
9  1. Charge les statistiques d'entraînement (moyenne / écart-type) depuis

```

```

Mode                               LastWriteTime           Length Name
----                               -
-a----          14/12/2025   16:43                0 monitor_drift.py

```

```

PS C:\Users\PC\Desktop\Master_S01A\53\MLops\mlops-lab-01> python src/monitor_drift.py
==> Drift check sur 2 requêtes récentes ==
- tenure_months: mean_prod=103.000 | mean_train=30.246 | z=4.287
ALERTE: drift probable sur tenure_months (z >= 2.5)
- num_complaints: mean_prod=26.580 | mean_train=1.174 | z=22.775
ALERTE: drift probable sur num_complaints (z >= 2.5)
- avg_session_minutes: mean_prod=256.250 | mean_train=35.124 | z=18.680
ALERTE: drift probable sur avg_session_minutes (z >= 2.5)
Résultat : 3 alerte(s) de drift, Analyse recommandée + retraining possible.
PS C:\Users\PC\Desktop\Master_S01A\53\MLops\mlops-lab-01>

```


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

6.1 Activation d'une nouvelle version

- Exemple pour entraîner et activer la version v2 :

```
(venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01> python -c "from src.train import main; main(version='v2', gate_f1=0.61)"

models
├── churn_model_v1_20251214_14...
├── churn_model_v1_20251214_14...
└── churn_model_v2_20251214_15...

registry
└── src
    ├── __pycache__
    ├── api.py
    └── evaluate.py

[OK] Modèle sauvegardé : C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01\models\churn_model_v2_20251214_154550.joblib
[DEPLOY] Modèle activé (current): churn_model_v2_20251214_154550.joblib
```

6.2 Rollback automatique

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

Rollback vers un modèle précis

```
src > rollback.py

def main(target: Optional[str] = None) -> None:
    if target not in models:
        raise ValueError(f"Modèle inconnu : {target}")

    set_current(target)
    print(f"[OK] rollback / activation => current_model = {target}")

Problems Output Debug Console Terminal Ports
PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01> New-Item src/rollback.py

Répertoire : C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01\src

Mode                LastWriteTime         Length Name
----                -
-a-----          14/12/2025      16:47             0 rollback.py

PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01> python src/rollback.py
[OK] rollback / activation => current_model = churn_model_v2_20251214_154528.joblib
PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01> python -c "from src.rollback import main; main('churn_model_v1_20251214_145121.joblib')"
[OK] rollback / activation => current_model = churn_model_v1_20251214_145121.joblib
PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\mlops-lab-01>
```

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.