

Lab 2 : Code Source management

Réalisée par :

El Hamzaoui Aya

Introduction

Dans un contexte MLOps, le développement d'un modèle de Machine Learning ne se limite pas à l'entraînement. Il implique également une gestion rigoureuse du code, des versions, des expérimentations, ainsi que des mécanismes de contrôle et de retour arrière. Ce lab a pour objectif de mettre en pratique l'utilisation de Git dans un projet MLOps réel, en simulant des situations concrètes rencontrées en équipe : modifications concurrentes, branches de fonctionnalités, conflits, annulation de changements et réécriture de l'historique.

Le projet utilisé est mlops-lab-01, qui représente un mini-système MLOps intégrant pipeline d'entraînement, registry de modèles, API de prédiction, monitoring et rollback.

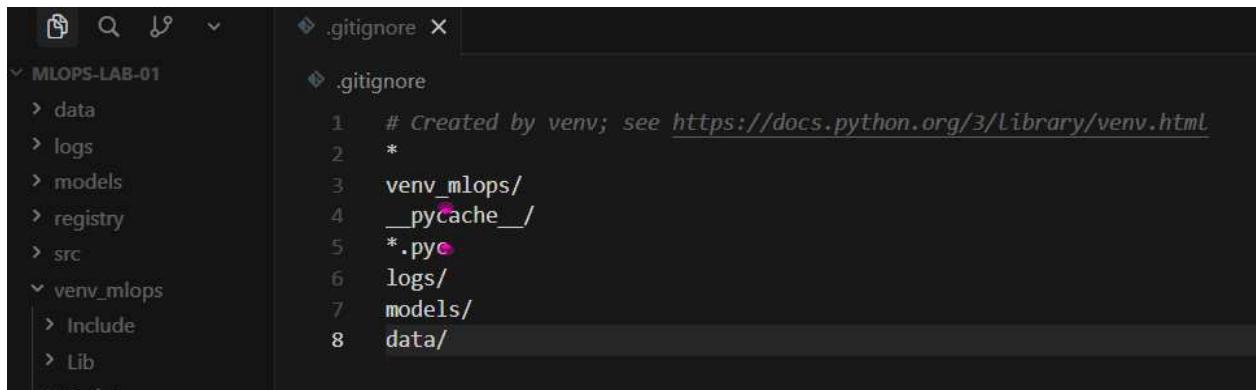
Objectifs du lab

Les objectifs principaux de ce lab sont :

- Comprendre le rôle de Git dans un projet MLOps
- Maîtriser les commandes Git essentielles dans un contexte réel
- Gérer des branches de fonctionnalités
- Comprendre et résoudre des conflits de merge
- Utiliser stash, reset, revert et rebase
- Faire la différence entre prototype et système prêt pour la production
- Adopter une logique de traçabilité, gouvernance et reproductibilité

Étape 1 : Initialiser Git dans mlops-lab-01

```
PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git init
Initialized empty Git repository in C:/Users/PC/Desktop/Master SDIA/S3/MLops/lab01/mlops-lab-01/.git/
```



```
● PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Étape 2 : Premier commit du projet MLOps

Cette étape marque la naissance officielle de l'historique Git du projet.

But : Créer une base stable et traçable du projet MLOps.

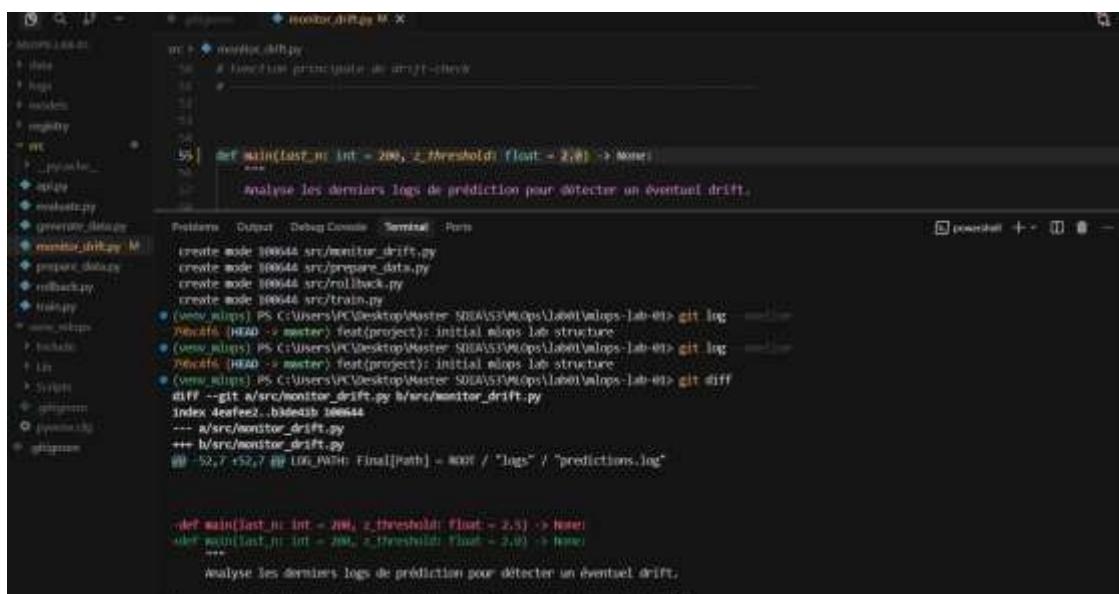
```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git add src data registry .gitignore
The following paths are ignored by one of your .gitignore files:
.gitignore
data
registry
src
hint: Use -f if you really want to add them.
hint: Disable this message with "git config set advice.addIgnoredFile false"

● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git commit -m "feat(project): initial mlops lab structure"
[master (root-commit) 79bc4f6] feat(project): initial mlops lab structure
 11 files changed, 2381 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 registry/current_model.txt
 create mode 100644 registry/metadata.json
 create mode 100644 registry/train_stats.json
 create mode 100644 src/api.py
 create mode 100644 src/evaluate.py
 create mode 100644 src/generate_data.py
 create mode 100644 src/monitor_drift.py
 create mode 100644 src/prepare_data.py
 create mode 100644 src/rollback.py
 create mode 100644 src/train.py

● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git log --oneline
79bc4f6 (HEAD -> master) feat(project): initial mlops lab structure
```

Étape 3 : Observer une modification avec git diff

Une modification est apportée au script monitor_drift.py.



```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mlops-lab-01> git diff
diff --git a/src/monitor_drift.py b/src/monitor_drift.py
index 4eefed2..b3deabb 100644
--- a/src/monitor_drift.py
+++ b/src/monitor_drift.py
@@ -52,7 +52,7 @@ def main(logPath = root / "logs" / "predictions.log"):
    analyze les derniers logs de prédition pour détecter un éventuel drift.

def main(last_n: int = 200, z_threshold: float = 2.5) -> None:
def main(last_n: int = 200, z_threshold: float = 2.0) -> None
    ...
    analyse les derniers logs de prédition pour détecter un éventuel drift.
```

```

● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git add src/monitor_drift.py
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git diff
diff --git a/src/monitor_drift.py b/src/monitor_drift.py
index 4eafee2..b3de41b 100644
--- a/src/monitor_drift.py
+++ b/src/monitor_drift.py
@@ -52,7 +52,7 @@ LOG_PATH: Final[Path] = ROOT / "logs" / "predictions.log"

 def main(last_n: int = 200, z_threshold: float = 2.5) -> None:
+def main(last_n: int = 200, z_threshold: float = 2.0) -> None:
    """
    Analyse les derniers logs de prédiction pour détecter un éventuel drift.

● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git commit -m "refactor(monitoring): adjust default drift threshold"
[master eaf3120] refactor(monitoring): adjust default drift threshold
 1 file changed, 1 insertion(+), 1 deletion(-)
○ (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01>

```

```

(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git checkout -b feature/api-request-id
>>
Switched to a new branch 'feature/api-request-id'

```

```

● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git add src/api.py
>> git commit -m "feat(api): improve request_id handling"
[feature/api-request-id a8b9c7f] feat(api): improve request_id handling
 1 file changed, 1 insertion(+), 1 deletion(-)
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git branch
 * feature/api-request-id
   master
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git switch master
Switched to branch 'master'
○ (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01>

```

```

● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git switch master
Already on 'master'
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git merge feature/api-request-id
Updating eaf3120..a8b9c7f
Fast-forward
  src/api.py | 2 ++
  1 file changed, 1 insertion(+), 1 deletion(-)
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git log --oneline
a8b9c7f (HEAD -> master, feature/api-request-id) feat(api): improve request_id handling
eaf3120 refactor(monitoring): adjust default drift threshold
79bc4fe feat(project): initial mllops lab structure
○ (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01>

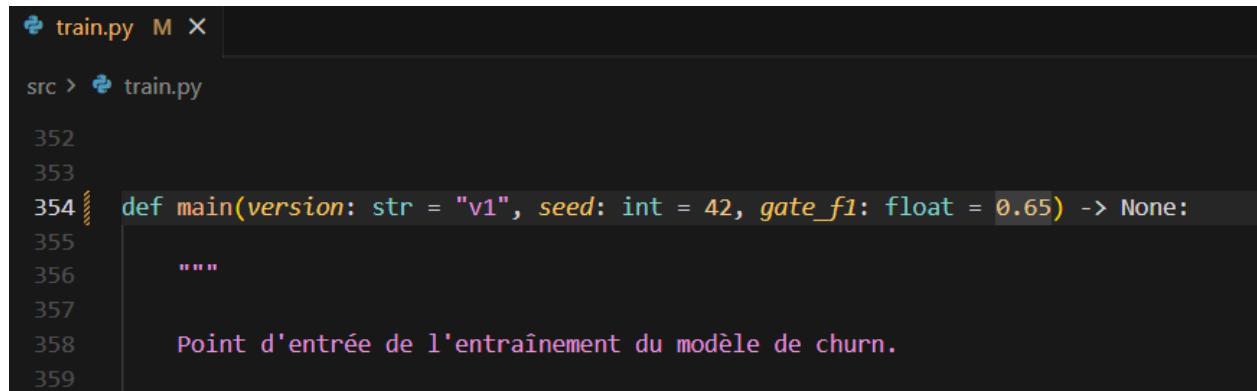
```

Étape 4 : Crée une branche de fonctionnalité liée au lab

Création d'une branche dédiée à une amélioration de l'API.

But : Isoler une fonctionnalité sans impacter la version stable du projet.

```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git checkout -b feature/change-gate
switched to a new branch 'feature/change-gate'
```



```
src > ⌂ train.py M X

src > ⌂ train.py

352
353
354 def main(version: str = "v1", seed: int = 42, gate_f1: float = 0.65) -> None:
355     """
356     Point d'entrée de l'entraînement du modèle de churn.
357
358
359
```

```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git add src/train.py
>> git commit -m "feat(train): set default gate_f1 to 0.60"
[feature/change-gate 5d75721] feat(train): set default gate_f1 to 0.60
1 file changed, 1 insertion(+), 1 deletion(-)
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git switch master
Switched to branch 'master'
```

Étape 5 : Fusionner la branche feature dans la branche principale

La fonctionnalité développée est intégrée dans la branche principale.

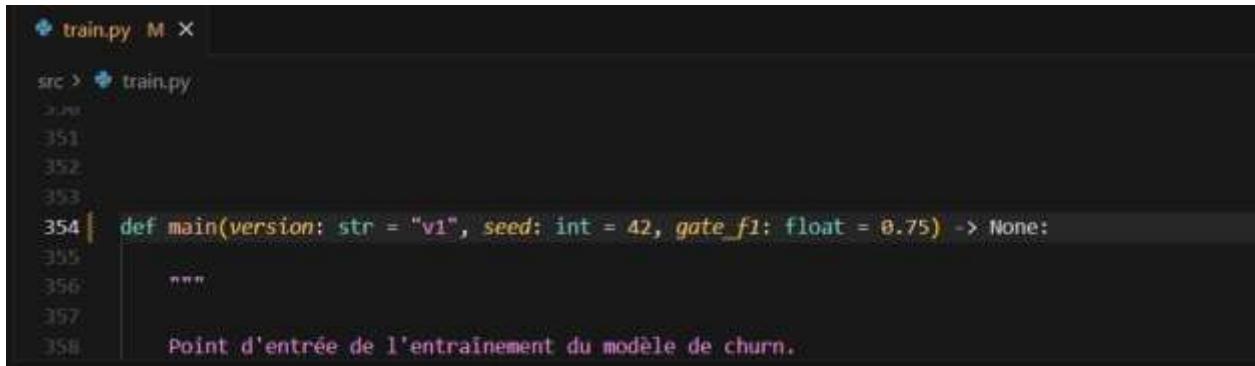
Comprendre le cycle feature → validation → intégration.

```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git switch master
Switched to branch 'master'
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git switch master
Already on 'master'
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git merge feature/api-request-id
Updating eaf3120..a8b9c7f
Fast-forward
 src/api.py | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git log --oneline
a8b9c7f (HEAD -> master, feature/api-request-id) feat(api): improve request_id handling
eaf3120 refactor(monitored): adjust default drift threshold
79bc4f6 feat(project): initial mlops lab structure
```

Étape 6 : Créer et résoudre un conflit de merge

Simulation d'un conflit réel sur le paramètre `gate_f1` du script `train.py`.

But : Apprendre à **lire, comprendre et résoudre un conflit Git**, compétence clé en équipe MLOps.



```
train.py M X
src > ⌂ train.py
351
352
353
354 | def main(version: str = "v1", seed: int = 42, gate_f1: float = 0.75) -> None:
355 |
356     """
357
358     Point d'entrée de l'entraînement du modèle de churn.
```

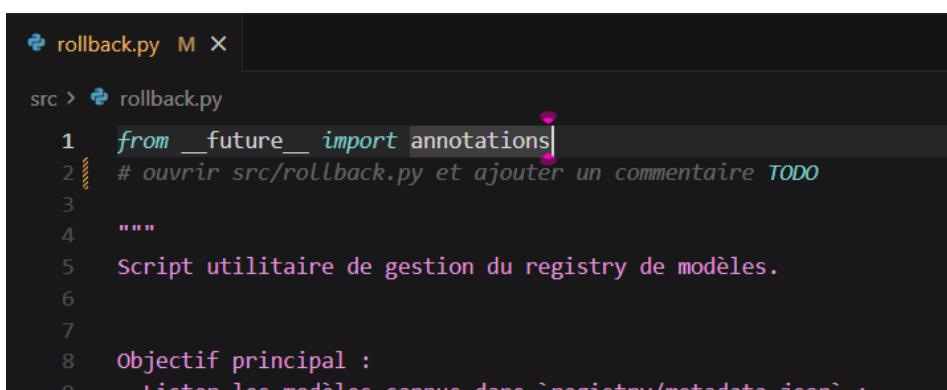
```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git add src/train.py
>> git commit -m "feat(train): set default gate_f1 to 0.75"
[master 49085ea] feat(train): set default gate_f1 to 0.75
 1 file changed, 1 insertion(+), 1 deletion(-)
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git merge feature/change-gate
Already up to date.
```

```
● (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01> git add src/train.py
>> git commit -m "merge(train): resolve gate_f1 conflict to 0.70"
On branch master
nothing to commit, working tree clean
○ (venv_mlops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mlops-lab-01>
```

Étape 7 : Utiliser git stash dans le contexte du lab

Gestion de modifications temporaires non prêtes à être committées.

But : Pouvoir **changer de contexte rapidement** sans perdre son travail.



```
rollback.py M X
src > ⌂ rollback.py
1  from __future__ import annotations
2  # ouvrir src/rollback.py et ajouter un commentaire TODO
3
4  """
5  Script utilitaire de gestion du registry de modèles.
6
7
8  Objectif principal :
9      · lister les modèles connus dans `registry/metadata.json` .
```

```
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01>git add src/rollback.py
>>
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   src/rollback.py
```

```
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git stash list
stash@{0}: WIP on master: 49085ea feat(train): set default gate_f1 to 0.75
stash@{1}: WIP on master: 49085ea feat(train): set default gate_f1 to 0.75
stash@{2}: WIP on master: 49085ea feat(train): set default gate_f1 to 0.75
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/rollback.py

no changes added to commit (use "git add" and/or "git commit -a")
● (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01> git stash pop
error: Your local changes to the following files would be overwritten by merge:
  src/rollback.py
Please commit your changes or stash them before you merge.
Aborting
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/rollback.py

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.
○ (venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\lab01\mllops-lab-01>
```

Étape 8 : Tester git reset sur un fichier d'expérimentation

Expérimentation des différents types de reset.

But : Comprendre l'impact de Git sur :

- l'historique
- le staging
- le working directory

```
Problems Output Debug Console Terminal Ports

Répertoire : C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01

Mode LastWriteTime Length Name
---- ----- ----
d---- 14/12/2025 20:03 experiments

(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01> echo "v1" > experiments/reset_test.txt
>> git add experiments/reset_test.txt
>> git commit -m "feat(experiments): add reset_test v1"
[master 37326ea] feat(experiments): add reset_test v1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 experiments/reset_test.txt
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01> echo "v2" > experiments/reset_test.txt
>> git add experiments/reset_test.txt
>> git commit -m "feat(experiments): update reset_test to v2"
>>
>> echo "v3" > experiments/reset_test.txt
>> git add experiments/reset_test.txt
>> git commit -m "feat(experiments): update reset_test to v3"
[master e6c2454] feat(experiments): update reset_test to v2
 1 file changed, 0 insertions(+), 0 deletions(-)
[master 7f4a16a] feat(experiments): update reset_test to v3
 1 file changed, 0 insertions(+), 0 deletions(-)
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01> git reset --soft HEAD~1
>> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   experiments/reset_test.txt
```

Étape 9 : Annuler un commit avec git revert

Simulation d'un changement incorrect en production.

But : Montrer que revert est la méthode sûre en production, car elle conserve l'historique.

```
Revert "feat(api): add bad change"

This reverts commit ee121e65be48771bdaf1087fa677c06759ac657.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       modified:   src/api.py
#
#
```

```
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01> type src\api.py
from __future__ import annotations

"""

API FastAPI de prédiction de churn pour le lab MLOps.

Ce service :
- charge dynamiquement le modèle courant indiqué dans "registry/current_model.txt" ;
- expose un endpoint "/health" pour vérifier l'état de l'API et du modèle ;
- expose un endpoint "/predict" pour faire une prédiction de churn à partir
de features simples (tenure, plaintes, durée de session, type d'abonnement,
région) ;
- journalise chaque requête de prédiction dans "logs/predictions.log" au
format JSON (une ligne par prédiction).

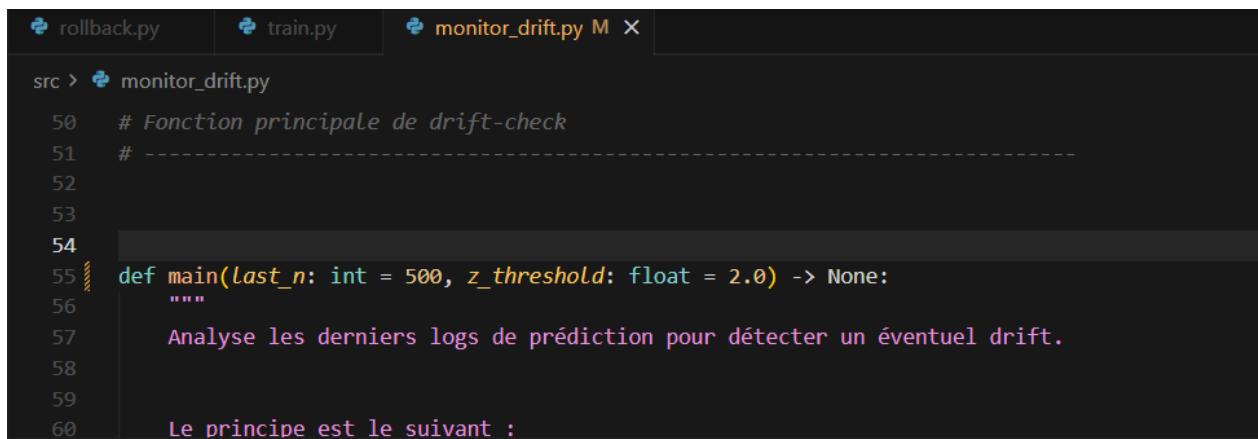
Cette API illustre une étape "Serve" dans un pipeline MLOps minimal :
un modèle versionné est promu dans registry, puis utilisé par un service
d'inference lancer.
```

Étape 10 : Rebase d'une branche feature sur la branche principale

Réécriture propre de l'historique d'une branche.

But : Maintenir un **historique Git linéaire, propre et lisible**, souvent exigé en MLOps.

```
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLops\lab01\mllops-lab-01> git checkout -b feature/drift-last-n
switched to a new branch 'feature/drift-last-n'
```



```
src > monitor_drift.py
  rollback.py  train.py  monitor_drift.py M X

src > monitor_drift.py
  50  # Fonction principale de drift-check
  51  # -----
  52
  53
  54
  55  def main(last_n: int = 500, z_threshold: float = 2.0) -> None:
      """
      Analyse les derniers logs de prédiction pour détecter un éventuel drift.
      Le principe est le suivant :
```

```

(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git add src/monitor_drift.py
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git commit -m "feat(monitored): increase last_n to 500"
[feature/drift-last-n 396a9d0] feat(monitored): increase last_n to 500
 2 files changed, 308 insertions(+), 309 deletions(-)
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git switch master # ou main
Switched to branch 'master'
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> echo "# comment v2" >> src/generate_data.py
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git add src/generate_data.py
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git commit -m "docs(data): document generation behavior"
[master f251806] docs(data): document generation behavior
 1 file changed, 0 insertions(+), 0 deletions(-)
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git switch feature/drift-last-n
Switched to branch 'feature/drift-last-n'
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git rebase master
Successfully rebased and updated refs/heads/feature/drift-last-n.
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> git log --oneline
a837bc4 (HEAD -> feature/drift-last-n) feat(monitored): increase last_n to 500
f251806 (master) docs(data): document generation behavior
ee121e6 feat(api): add bad change
a0b9c7f (feature/api-request-id) feat(api): improve request_id handling
ea1f120 refactor(monitored): adjust default drift threshold
79bc4f6 feat(project): initial mllops lab structure
(venv_mllops) PS C:\Users\PC\Desktop\Master SDIA\S3\MLOps\label1\mllops-lab-01> █

```

Conclusion

Ce lab montre clairement que le Machine Learning en production ne se résume pas à entraîner un modèle.

Un système MLOps fiable repose sur :

- une gestion rigoureuse du code
- un versionnement clair
- une traçabilité complète
- des mécanismes de contrôle (gate, rollback)
- une collaboration structurée via Git

la différence fondamentale entre :

- **un prototype de data scientist**
- **un système prêt pour la production**

Ce travail constitue une base solide pour aborder des pratiques avancées telles que **CI/CD**, **monitoring automatisé**, et **déploiement industriel des modèles**.