TP2 – Intégration MLOps complète

(GitHub Actions, DVC, CML, Docker, Google Drive)

Oussama Khouya Radwane Khemisse Achrafe Elalaoui

8 décembre 2025

1 Introduction

Projet `ml-dvc-iris` dérivé du TP2 (pipeline DVC : prepare → train → evaluate). Objectifs : automatiser la chaîne MLOps (DVC + GitHub Actions + CML), stocker les artefacts sur Google Drive, assurer la reproductibilité (`dvc repro`), préparer la containerisation et la promotion du meilleur modèle.

2 Architecture MLOps mise en place

- Pipeline DVC (`dvc.yaml`) avec trois stages (??).
- Données/artefacts suivis : `data/iris.csv`, `data/iris_preprocessed.csv`, `models/random_forest.pkl` `metrics/*.json`.
- Remote DVC Google Drive : `new_gdrive` configuré via secrets GitHub.
- CI/CD : workflow GitHub Actions `mlops-pipeline.yaml` (checkout, install deps, config remote, `dvc pull`, fallback dataset, `dvc repro`, rapport CML, commentaire PR).
- Reporting : `scripts/generate_cml_report.py` et commentaire CML automatique.
- Containerisation : Dockerfile à ajouter (build/run non réalisés).
- Promotion modèle : stage `deploy` et `models/production_model.pkl` à compléter.

Listing 1 – Pipeline DVC

```
stages:
  prepare:
    cmd: python scripts/preprocess.py
    deps:
      - scripts/preprocess.py
      - data/iris.csv
    outs:
      - data/iris_preprocessed.csv

  train:
    cmd: python src/train.py
    deps:
      - src/train.py
      - data/iris_preprocessed.csv
      - params.yaml
    outs:
      - models/random_forest.pkl
      - metrics/train_metrics.json

  evaluate:
    cmd: python src/evaluate.py
    deps:
      - src/evaluate.py
```

```
      - models/random_forest.pkl
      - data/iris_preprocessed.csv
    outs:
      - metrics/eval_metrics.json
```

```
/home/o/p/e/_/D/d/full-MLOps-integration master +4 ?9 ❯ dvc repro
Stage 'prepare' didn't change, skipping
Running stage 'train':
> python src/train.py
Modèle entraîné sauvegardé dans: models/random_forest.pkl
Métriques d'entraînement sauvegardées dans: metrics/train_metrics.json
Accuracy (test): 0.9333
Updating lock file 'dvc.lock'

Running stage 'evaluate':
> python src/evaluate.py
Métriques d'évaluation sauvegardées dans: metrics/eval_metrics.json
Accuracy (données complètes): 0.9867
Updating lock file 'dvc.lock'
```

FIGURE 1 – Exécution locale du pipeline `dvc repro`

3 CI GitHub Actions et CML

3.1 Dépendances et rapport

— Dépendances ML/DVC/CML dans `requirements.txt` (??).
— Rapport Markdown généré par `scripts/generate_cml_report.py` (??, ??).

3.2 Workflow GitHub Actions

— Triggers : `push`, `pull_request`.
— Étapes : checkout, Python 3.11, install deps (`dvc[gdrive]`, `cml`), config remote GDrive, `dvc pull`, fallback `scripts/download_iris.py` si besoin, `dvc repro`, génération du rapport, `cml comment create`.
— Succès du workflow (??) et commentaire CML en PR (??).

Listing 2 – Extrait du workflow mlops-pipeline.yaml

```
on:
  push:
  pull_request:
jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.11" }
      - uses: iterative/setup-cml@v2
      - run: |
          python -m pip install --upgrade pip setuptools wheel
          pip install -r requirements.txt
          pip install "dvc[gdrive]==3.63.0"
      - name: Configure DVC remote (OAuth)
        run: |
```

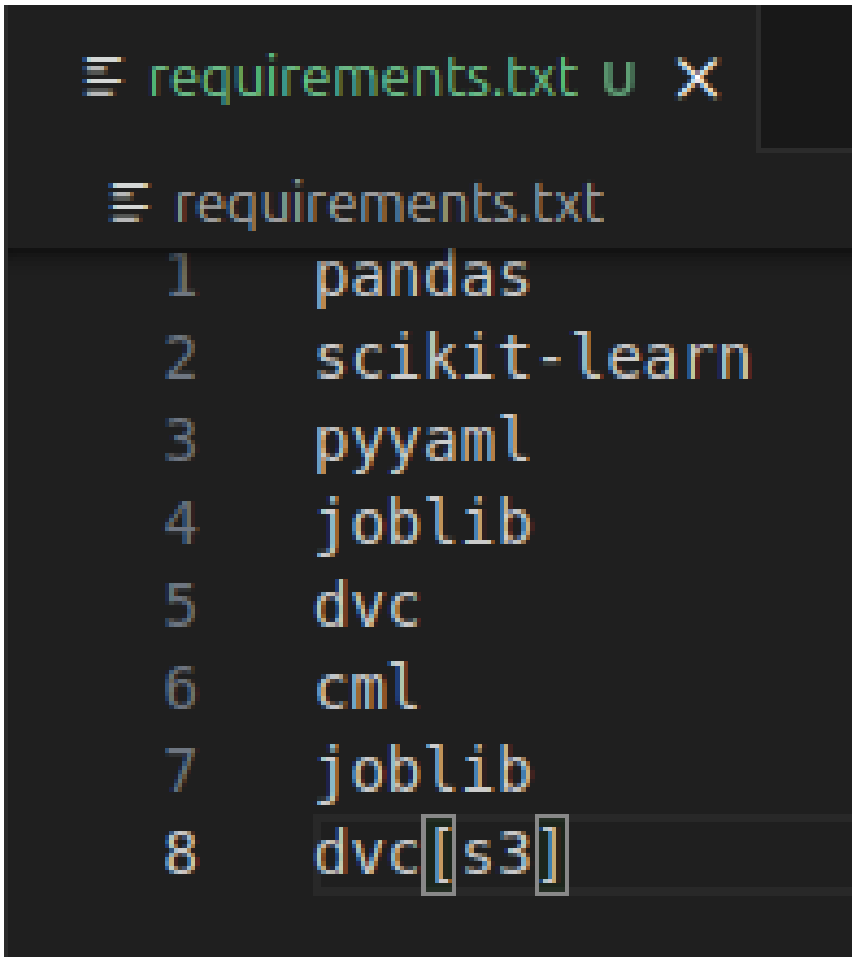


FIGURE 2 – Dépendances Python

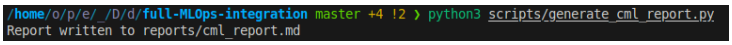
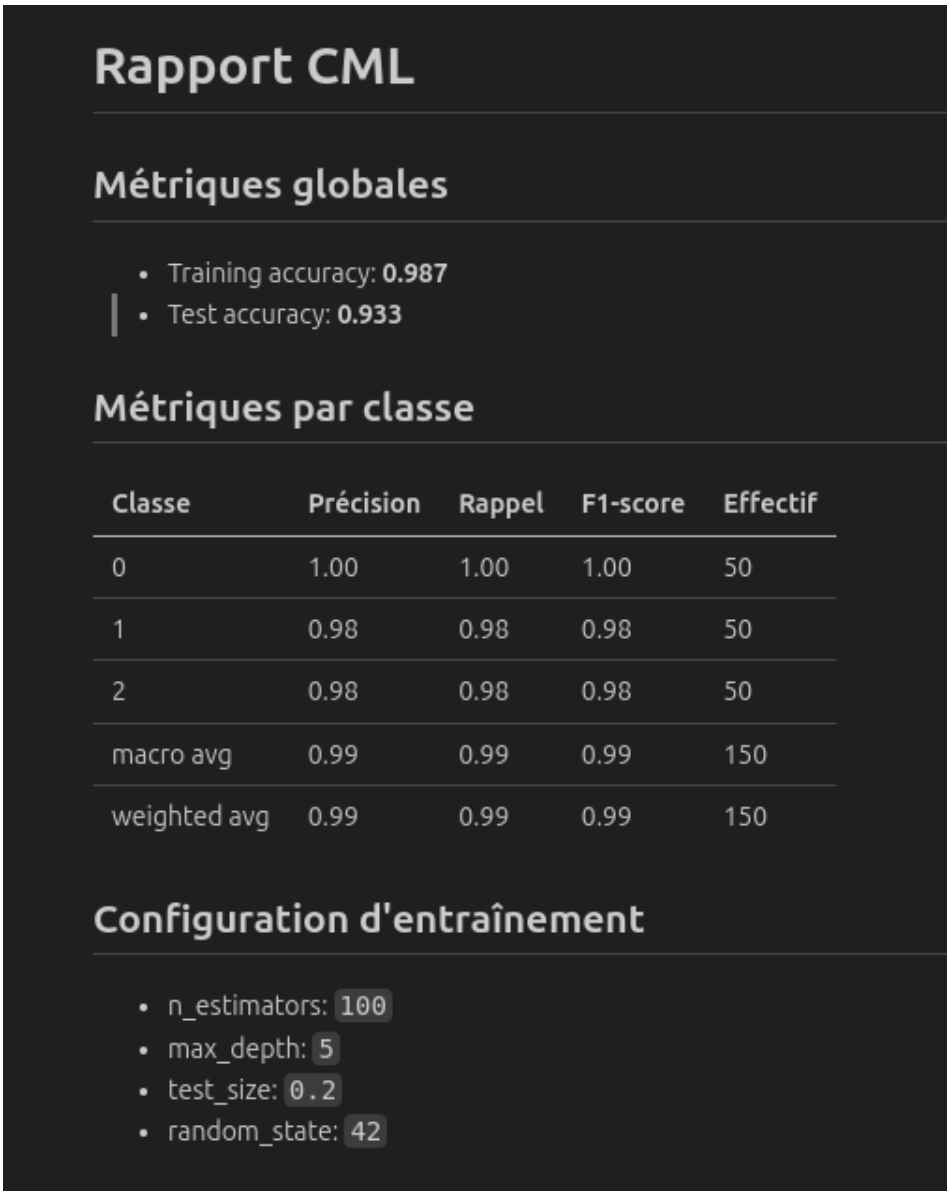


FIGURE 3 – Génération du rapport CML

```
      echo '${{ secrets.GDRIVE_CREDENTIALS_JSON }}' >
         gdrive_user_credentials.json
      dvc remote modify --local new_gdrive
         gdrive_user_credentials_file gdrive_user_credentials.json
      dvc remote modify --local new_gdrive gdrive_client_id "${{
         secrets.GDRIVE_CLIENT_ID }}"
      dvc remote modify --local new_gdrive gdrive_client_secret "${{
          secrets.GDRIVE_CLIENT_SECRET }}"
  - run: dvc pull -v
  - run: if [ ! -f data/iris.csv ]; then python scripts/
     download_iris.py; fi
  - run: dvc repro
  - run: |
      python3 scripts/generate_cml_report.py
      cml comment create reports/cml_report.md
```

4 Remote DVC Google Drive

— Remote `new_gdrive` pointant vers l'ID Drive, authentifié via secrets OAuth.



Rapport CML

Métriques globales

- Training accuracy: **0.987**
- Test accuracy: **0.933**

Métriques par classe

Classe	Précision	Rappel	F1-score	Effectif
0	1.00	1.00	1.00	50
1	0.98	0.98	0.98	50
2	0.98	0.98	0.98	50
macro avg	0.99	0.99	0.99	150
weighted avg	0.99	0.99	0.99	150

Configuration d'entraînement

- n_estimators: `100`
- max_depth: `5`
- test_size: `0.2`
- random_state: `42`

FIGURE 4 – Aperçu de `reports/cml_report.md`

— Synchronisation vérifiée : `dvc pull` (**??**) et `dvc push` (**??**).

Listing 3 – Extrait .dvc/config

```
[core]
    remote = new_gdrive

[remote "new_gdrive"]
    url = gdrive://1qnTG-xYstcnUbljTv94pp2izEqLpBP-o
    gdrive_use_service_account = true
    gdrive_service_account_json_file_path = /home/oldhome/pc/enset/_S3/
       DevOps/dvc/gdrive_user_credentials.json
```

5 Containerisation Docker

— Attendu : Dockerfile (base `python:3.11-slim`, install deps, copie projet, `CMD ["dvc","repro"]`).
— Statut : non réalisé, aucune capture de build/run dans `./snapshots`.

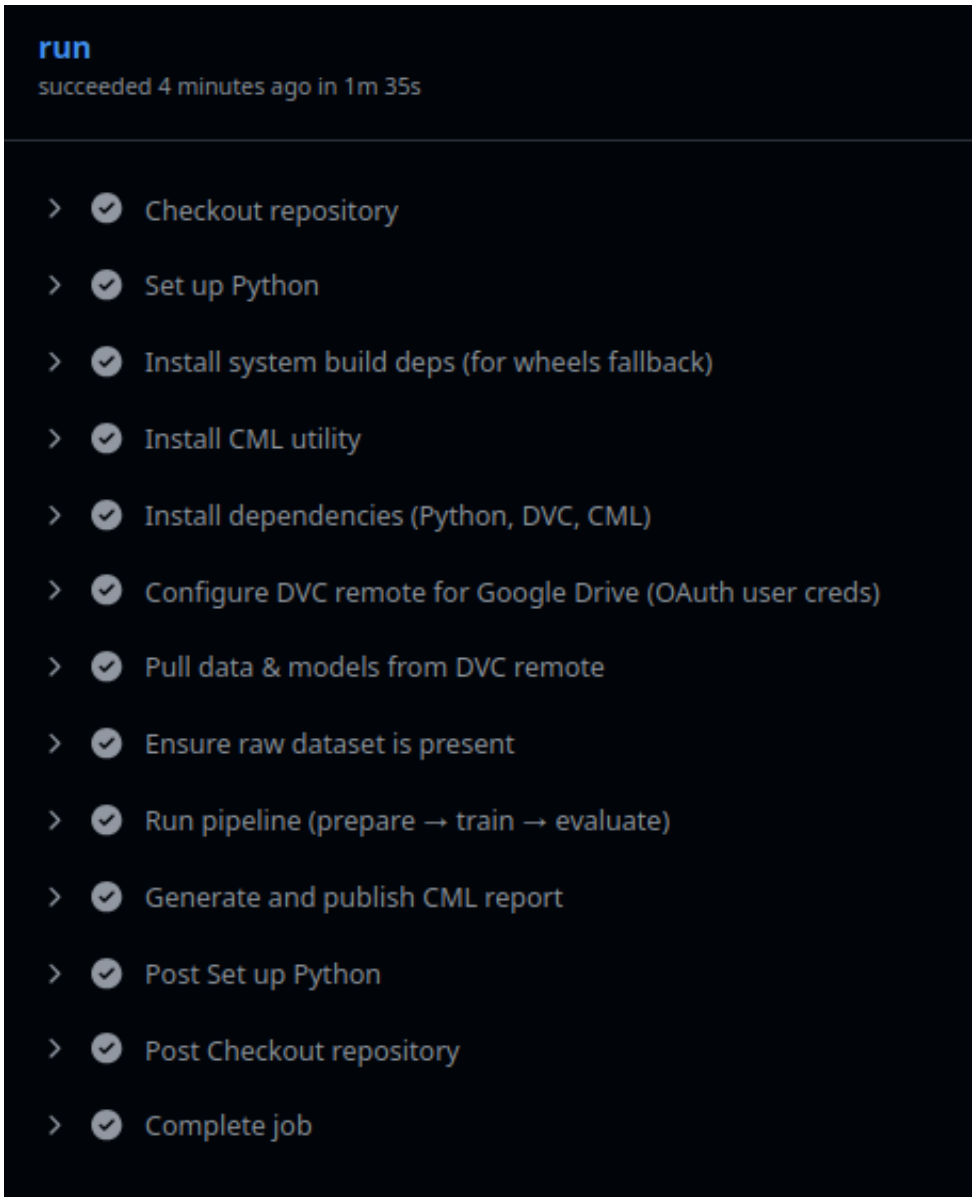


FIGURE 5 – Exécution réussie du workflow GitHub Actions

— À faire : ajouter Dockerfile, exécuter `docker build/run`, produire la capture.

6 Promotion du meilleur modèle

— Attendu : script `scripts/deploy.py` et stage `deploy` produisant `models/production_model.pkl`.
— Statut : non implémenté, fichier `models/production_model.pkl` non généré.
— À faire : comparer `metrics/eval_metrics.json` à `metrics/best_metrics.json`, copier `models/random_forest.pkl` en cas de meilleure accuracy.

7 Difficultés rencontrées et solutions

— Authentification GDrive : tokens corrompus/type invalide. Solution : reconfigurer `new_gdrive` avec OAuth via secrets et valider `dvc pull/push`.
— Build PyYAML en CI : échec de wheel. Solution : pinner `pyyaml==6.0.2` et installer `libyaml-dev+build-essential`.

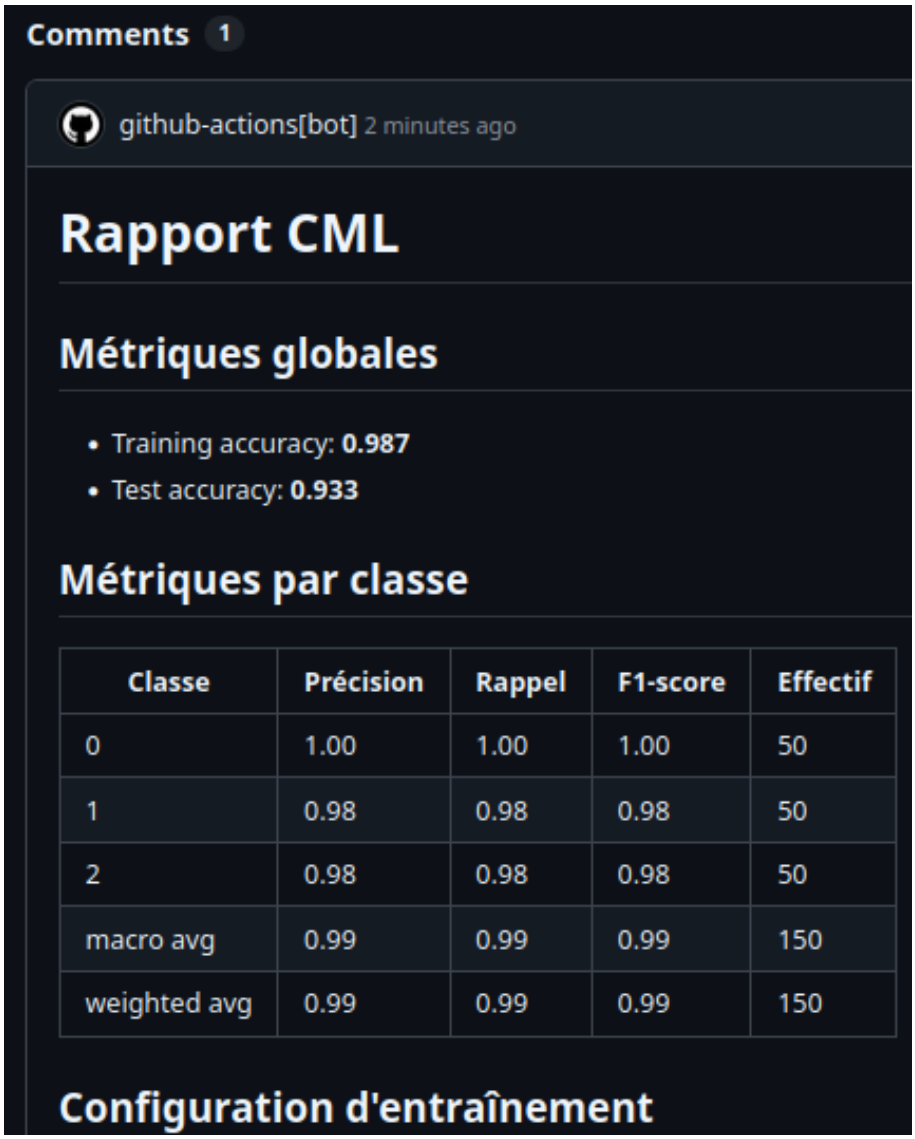


FIGURE 6 – Commentaire CML généré dans la Pull Request

— Données brutes absentes : `iris.csv` manquante au début. Solution : `dvc add data/iris.csv` puis `dvc push` ; fallback `scripts/download_iris.py` dans la CI.
— Plugin GDrive manquant en CI : erreur "dvc-gdrive". Solution : installer explicitement `dvc[gdrive]==3.63.0` dans le workflow.
— Reste à faire : Dockerfile/build-run, stage `deploy` et génération de `models/production_model.pkl`.

8 Conclusion

Chaîne MLOps quasi complète : pipeline DVC reproductible, remote Google Drive fonctionnel, workflow GitHub Actions + CML opérationnel avec rapport et commentaire PR. Travaux restants : containerisation Docker (build/run), ajout du stage `deploy` pour produire `models/production_model` et finaliser la promotion du meilleur modèle.

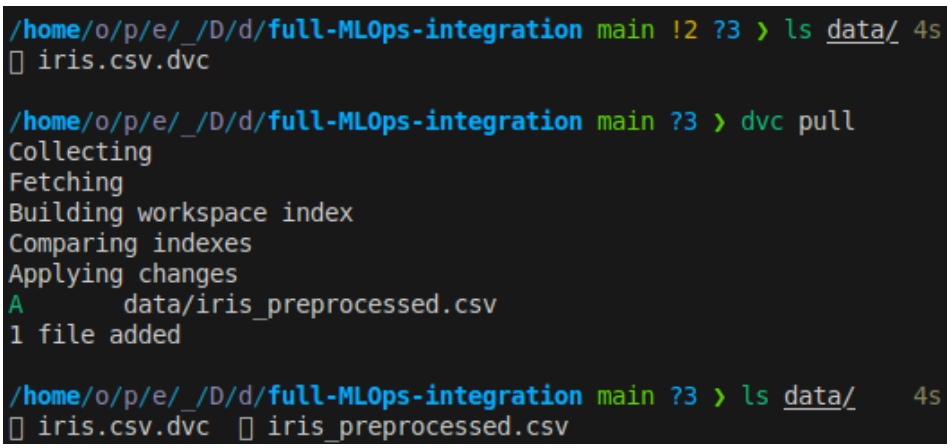


FIGURE 7 – Récupération des artefacts par `dvc pull`

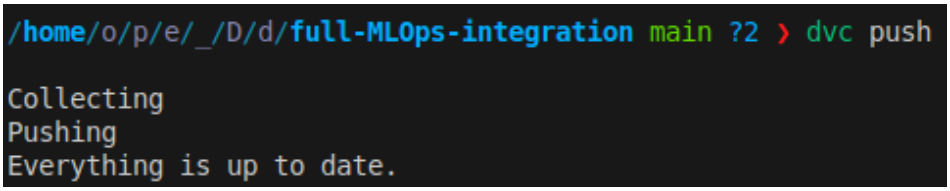


FIGURE 8 – `dvc push` indiquant un cache distant à jour