

TP2 – Intégration MLOps complète

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

Oussama Khouya

Radwane Khemisse

Achrafe Elalaoui

8 décembre 2025

1 Contexte et objectifs

Projet `ml-dvc-iris` complété avec un pipeline DVC (`prepare → train → evaluate`), génération automatique de rapport CML et artefacts stockés sous `metrics/` et `models/`. Objectifs : CI GitHub Actions, rapport CML, remote DVC Google Drive, containerisation Docker et promotion automatique du meilleur modèle.

2 Vérification du projet de départ

- Pipeline DVC existant : `dvc.yaml` (extrait ??).
- Données et artefacts présents : `data/iris.csv`, `data/iris_preprocessed.csv`, `models/random_forest`, `metrics/*.json`.
- Exécution locale du pipeline `dvc repro` réussie (??).

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 sauvégarde dans: models/random_forest.pkl
Métriques d'entraînement sauvégarde dans: metrics/train_metrics.json
Accuracy (test): 0.9333
Updating lock file 'dvc.lock'

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

```

FIGURE 1 – Commande `dvc repro` exécutée avec succès

3 Workflow GitHub Actions avec DVC et CML

3.1 Dépendances Python

- Fichier `requirements.txt` incluant bibliothèques ML, DVC et CML (??).

3.2 Script de rapport CML

- `scripts/generate_cml_report.py` lit les métriques JSON et produit `reports/cml_report.md`.
- Exécution manuelle : `python3 scripts/generate_cml_report.py` (??); aperçu du rapport généré (??).

Listing 2 – Extrait – génération du rapport CML

```

report_text = build_markdown(train_metrics, eval_metrics)
os.makedirs(REPORTS_DIR, exist_ok=True)
OUTPUT_PATH.write_text(report_text)

```

3.3 Workflow CI attendu

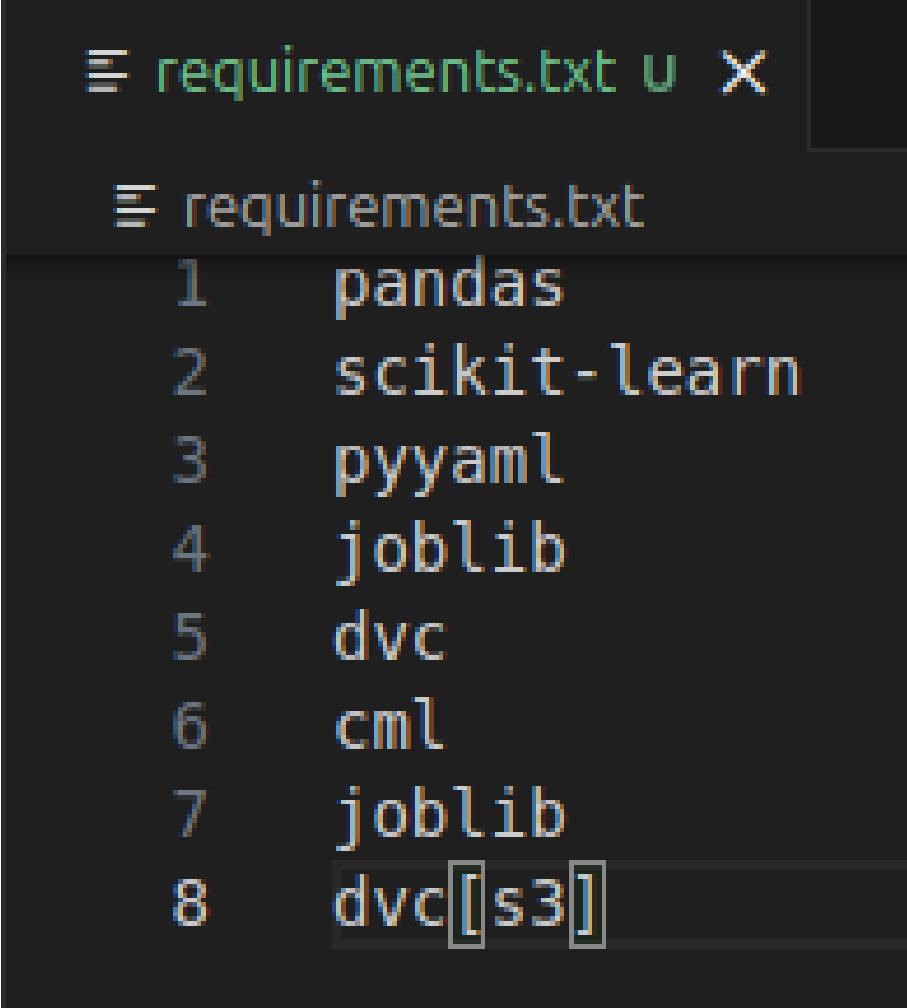
- Déclencheurs : `push` et `pull_request`.
- Étapes : `checkout`, installation Python & dépendances, `dvc pull`, `dvc repro`, `python3 scripts/generate_cml_report.py`, publication du commentaire CML sur la PR.
- Implémentation à ajouter dans `.github/workflows/mllops-pipeline.yml` (non encore présente dans ce dépôt).

4 Remote DVC Google Drive

- Remote attendu : `dvc remote add -d gdrive_remote gdrive://<ID_DOSSIER>`, suivi de `dvc push`.
- Statut actuel : aucune configuration remote dans `.dvc/config`; à compléter avec les identifiants de service Google et un secret CI.

5 Containerisation Docker

- Dockerfile attendu : image `python:3.11-slim`, installation `requirements.txt`, copie du projet, `CMD ["dvc", "repro"]`.



```
☰ requirements.txt ✘ ×  
☰ requirements.txt  
1   pandas  
2   scikit-learn  
3   pyyaml  
4   joblib  
5   dvc  
6   cml  
7   joblib  
8   dvc[s3]
```

FIGURE 2 – Dépendances Python (requirements.txt)

```
/home/o/p/e/_D/d/full-MLOps-integration master +4 12 > python3 scripts/generate_cml_report.py  
Report written to reports/cml_report.md
```

FIGURE 3 – Commande de génération du rapport CML

- Commandes de validation : `docker build -t ml-dvc-iris:latest .` puis `docker run -rm ml-dvc-iris:latest`.
- Statut actuel : Dockerfile à ajouter.

6 Test end-to-end sur Pull Request

- Branches de test : `feature/test-mlops` puis PR vers `main`.
- Attendus : pipeline CI vert, rapport CML commenté dans la PR.
- Statut : à activer une fois le workflow CI poussé.

7 Déploiement automatique du meilleur modèle

- Script `scripts/deploy.py` à implémenter : comparaison de l'accuracy courante (`metrics/eval_metrics` avec `metrics/best_metrics.json`) ; copie vers `models/production_model.pkl` en cas d'amélioration.



FIGURE 4 – Rapport CML produit (`reports/cml_report.md`)

- Stage DVC deploy à ajouter dans `dvc.yaml` dépendant de `models/random_forest.pkl` et produisant `models/production_model.pkl`.
- Statut actuel : non présent, à compléter.

8 Conclusion

Chaîne MLOps en place localement : pipeline DVC fonctionnel et génération de rapport CML automatisée. À finaliser : ajout du workflow GitHub Actions, configuration du remote Google Drive, Dockerfile et stage de déploiement pour promouvoir automatiquement le meilleur modèle. Les captures fournies documentent l'exécution locale du pipeline (??) et la génération du rapport CML (??, ??).