MLOps Engineering
Machine Learning Operations V2.0.0
Final Group Assignment

MsC in Business Analytics and Data Science

Madrid, Jun 2025

ie UNIVERSITY

```
(mlops_project) (base) idiazl@IvanDiaz:~/2025_MLOps/mlops_project$ mlflow run . -P steps=data_load
2025/06/04 05:18:36 INFO mlflow.utils.conda: Conda environment mlflow-224b471268c78153b1775630c75c64aaea673cb7 already exists.
2025/06/04 05:18:36 INFO mlflow.projects.utils: === Created directory /tmp/tmpgli8lqeb for downloading remote URIs passed to arguments of
type 'path' ===
2025/06/04 05:18:36 INFO mlflow.projects.backend.local: === Running command 'source /home/idiazl/miniconda3/bin/../etc/profile.d/conda.sh
&& conda activate mlflow-224b471268c78153b1775630c75c64aaea673cb7 1>&2 && python main.py main.steps=data_load
' in run with ID '5faff5a758da4522af4aca381a0ecac9' ===
Running step: data_load
2025/06/04 05:18:39 INFO mlflow.utils.conda: Conda environment mlflow-729753b51ca33fb90ad94457ca1cbb8c7cf56cce already exists.
2025/06/04 05:18:39 INFO mlflow.projects.utils: === Created directory /tmp/tmp71gxx2f_ for downloading remote URIs passed to arguments of
type 'path' ===
2025/06/04 05:18:39 INFO mlflow.projects.backend.local: === Running command 'source /home/idiazl/miniconda3/bin/../etc/profile.d/conda.sh
&& conda activate mlflow-729753b51ca33fb90ad94457ca1cbb8c7cf56cce 1>&2 && python run.py $(echo '')
' in run with ID 'b9fb3e2919664a46b86aa384d7b1e150' ===
wandb: Currently logged in as: idiazl to https://api.wandb.ai. Use `wandb login --relogin` to force relogin
wandb: Tracking run with wandb version 0.20.0
wandb: Run data is saved locally in /home/idiazl/2025_MLOps/mlops_project/src/data_load/wandb/run-20250604_051841-y0buhzx4
wandb: Run `wandb offline` to turn off syncing.
wandb: Syncing run data_load_20250604_051841_opiod_raw_data.csv
wandb: ⭐ View project at https://wandb.ai/idiazl/opioid_mlops_project
wandb: 🚀 View run at https://wandb.ai/idiazl/opioid_mlops_project/runs/y0buhzx4
[2025-06-04 05:18:43,272][data_load][INFO] - Started WandB run: data_load_20250604_051841_opiod_raw_data.csv
[2025-06-04 05:18:43,272][data_loader][INFO] - Loaded environment from: .env
[2025-06-04 05:18:43,272][data_loader][INFO] - Loading config from: /home/idiazl/2025_MLOps/mlops_project/config.yaml
[2025-06-04 05:18:43,291][data_loader][INFO] - Loaded data from /home/idiazl/2025_MLOps/mlops_project/data/raw/opiod_raw_data.csv (csv), s
hape=(1000, 22)
[2025-06-04 05:18:44,708][data_load][INFO] - Logged raw data artifact to WandB
wandb:
wandb:
wandb: Run summary:
wandb:       n_cols 22
wandb: n_duplicates 0
wandb:       n_rows 1000
wandb:
wandb: 🚀 View run data_load_20250604_051841_opiod_raw_data.csv at: https://wandb.ai/idiazl/opioid_mlops_project/runs/y0buhzx4
wandb: ⭐ View project at: https://wandb.ai/idiazl/opioid_mlops_project
wandb: Synced 5 W&B file(s), 2 media file(s), 5 artifact file(s) and 0 other file(s)
wandb: Find logs at: ./wandb/run-20250604_051841-y0buhzx4/logs
[2025-06-04 05:18:46,318][data_load][INFO] - WandB run finished
2025/06/04 05:18:47 INFO mlflow.projects: === Run (ID 'b9fb3e2919664a46b86aa384d7b1e150') succeeded ===
2025/06/04 05:18:47 INFO mlflow.projects: === Run (ID '5faff5a758da4522af4aca381a0ecac9') succeeded ===
(mlops_project) (base) idiazl@IvanDiaz:~/2025_MLOps/mlops_project$
```

https://github.com/zedr/clean-code-python.git

Next steps on MLOps journey – Final submission (Tech section)

Step	Rationale	Technical details
1. Freeze baseline	Locks a known-good state so you can always revert and compare	• Tag repo e.g. `v0.1_notebook_to_mlops` • Add checkpoint note in `README.md`
2. Add Hydra config	Centralizes parameters and paths, teaching clean separation of code and config	• Create `conf/config.yaml` • Decorate `src/main.py` with `@hydra.main` • Replace hard-coded paths with `hydra.utils.to_absolute_path`
3. Wrap with MLflow Projects	Gives single-command, reproducible runs that are easy to share	• Add `MLproject` with `conda_env` and `main` entry point • Run pipeline via `mlflow run .` • Will sue MLflow to launch steps only (wand for logging)
4. Integrate Weights & Biases	Provides best-practice experiment tracking, lineage, and dashboards	• `pip install wandb` and set `WANDB_API_KEY (.env)` • Call `wandb.init` inside training script • Log metrics, configs, and model artefacts
5. Config-driven step pipeline	Shows how to build modular, extensible workflows students can tweak safely	• Define `_steps` list in `src/main.py` • Read `main.steps` from Hydra to choose which MLflow runs execute • Keep each step in its own folder under `src/`
6. CI/CD with GitHub Actions	Automates tests and pipeline runs, enforcing quality and reproducibility	• Create `.github/workflows/ci.yml` • Use `setup-miniconda` to load env from `conda.yml` • Run `pytest` then `mlflow run . --no-conda` • Store `WANDB_API_KEY` as GitHub secret • Add CI badge to `README.md`
8. Dockerize the API	Makes the app portable and easy to deploy anywhere, teaching real-world reproducibility	• Write `Dockerfile` for FastAPI app • Copy source, model, and config files into image • Test locally: `docker build` and `docker run` • Document usage in `README.md`
9. Serve with FastAPI & deploy (Dockerized project)	Completes end-to-end lifecycle, exposing the model as a live service	• Build `src/api.py` with FastAPI `/predict` endpoint loading latest model • Test locally via uvicorn src.api:app • Add Procfile and deploy to Railway/ Render free tier • Document curl example in `README.md`

On top of 1st submission expectations

Expectations for both best practices to be showcased, and the modules to be delivered for this Final project (Tech section)

General best practices

1. **Pipeline orchestration**: Step-based control using MLflow Projects
2. **Config-driven workflows**: Use Hydra to manage parameters, paths, and active steps
3. **Experiment tracking**: Use Weights & Biases (W&B) to track metrics, configs, models
4. **CI/CD automation**: Use GitHub Actions to run tests and pipeline on every push
5. **Deployment preparation**: Dockerize for reliable, environment-agnostic serving
6. **Reproducibility**: Conda environment fully specified and consistent across runs
7. **Modular evolution**: Each step encapsulated in its own module, callable independently
8. **Clear documentation**: Update README to reflect new run and deployment methods

Pipeline & code specifics

1. **MLproject**: Defines environment and main entrypoint
2. **conda.yaml**: Full environment reproducibility
3. **conf/config**.yaml: Main Hydra config, with parameters and active steps
4. **main.py**: Pipeline controller executing steps using mlflow.run
5. **src/step_name/:** Folder per step with independent main.py
6. **wandb.init():** Captures experiment context, logs metrics and artifacts
7. **.github/workflows/ci.yml**: Runs pytest and mlflow run . with --no-conda
8. **Dockerfile**: Containerizes FastAPI app with preloaded model for deployment
9. **src/api.py**: REST endpoint for real-time inference

On top of 1st submission expectations

Final group assignment expectations and guidelines (Tech section)

Category	Details
MLflow Integration	• The MLproject file defines a single main entry point. **Steps are modular and called** using *mlflow.run* **with parameters from Hydra config**
Hydra Config Management	• **All pipeline settings (paths, seeds, steps, hyperparameters)** are managed via config.yaml. **Supports CLI overrides** and uses hydra.utils paths
W&B Tracking	• **All key** metrics, configurations, and model artifacts are **logged with wandb**. Experiments are grouped and **versioned via W&B dashboard** (one can see the lineage of the project)
CI/CD Pipeline	• A complete GitHub Actions **workflow** (ci.yml) **runs tests and the full pipeline** using the defined Conda environment. API keys managed via secrets
Docker & Serving	• A Dockerfile builds the FastAPI app. Model is loaded and served from /predict endpoint. **Ready for local or cloud deployment** (e.g. Railway)
Pipeline Structure	• Each step (e.g. preprocess, train, evaluate) exists in its own folder with an independent main.py. **Controlled through the steps list logic**
Documentation & Usability	• The README.md is **updated to reflect Phase 2 capabilities**: how to run via MLflow, config overrides, W&B usage, CI/CD, and API testing instructions
Code Quality	• Modular, clean code with logging, testing, and reproducibility. Each step behaves as an independent unit that can be debugged or reused

On top of 1st submission expectations

Final project expectations and guidelines (Tech Section)

Criterion	Fundamentals missing (0–4)	Basic attempt (5–6)	Good implementation (7–8)	Excellent – Industry grade (9–10)
MLflow Project	MLproject file missing or misconfigured	Basic file; poor param integration	Valid project file; reproducible	Single main entry point; parameters passed via CLI; MLflow runs configured per step
Hydra Config	No config or hard-coded values	Incomplete or confusing config	All major values centralized hydra.utils.to_absolute_path()	Modular config + CLI overrides work well
W&B Integration	No W&B or only local logging	Some metrics logged	Metrics + artifacts logged	Tracked with aliases or version tags (e.g., prod), artifacts logged as part of W&B run
GitHub Actions	No CI/CD or broken workflow	Basic syntax; doesn't run full flow	Installs, runs pipeline	Tests, pipeline, W&B logging, badge shown
Docker & Serving	No Docker, no API or broken	Docker builds but not linked to app	Docker + local FastAPI works Endpoint returns valid responses	Clean container, exposed endpoint, documented usage
Code Modularity	Monolithic script, no steps logic	Main.py partial orchestration	Steps isolated, config-driven	Fully modular, reproducible across steps
Documentation	No README updates	Minimal additions	Covers pipeline, configs, W&B Focus on "Why"	Clear run/deploy guide, diagrams, CI badge

On top of 1st submission expectations

Deliverable check-list review (Final group assignment)

1. Does the MLproject file define a valid main entry point with a steps parameter?
2. Is the entire pipeline controlled via config.yaml using Hydra (paths, params, seeds)?
3. Are all pipeline steps isolated in their own folders with a main.py each?
4. Are W&B runs logging all relevant metrics, config values, and artifacts?
5. Does .github/workflows/ci.yml exist and successfully run all required jobs?
6. Is conda.yaml fully defined and used in both local runs and GitHub Actions workflows?
7. Does the Dockerfile build cleanly and serve a working FastAPI app?
8. Does the API endpoint (e.g., /predict) correctly load the model and return valid responses?
9. Does the README clearly explain how to run the pipeline, override configs, track experiments, and deploy the model?
10. Is the code modular, logged, documented, and fully passing all tests via pytest in CI?

On top of 1st submission expectations

Business Case Guidelines to be showcased in 10' presentation (Business prezo section)

1. **Hypothetical client & Business Context**
   - Industry, business unit, and **AI maturity** (Data, Tools, Talent, Governance)
   - Clearly define a **real-life problem** you're solving from your previous notebook work
2. **Problem & Goals**
   - Define **baseline** (BAU/Jupyter workflow) vs. **desired impact** with MLOps
   - State a clear, measurable **business goal** (KPI)
3. **Solution Overview**
   - Describe the MLOps pipeline and its functionalities in **executive (non-technical) terms**
   - **Highlight improvements** in automation, monitoring, reproducibility, and model lifecycle
4. **Organizational Readiness**
   - Skills, roles required, infra needs (on-prem/cloud), and **change required to adopt**
5. **Client Benefits**
   - Tangible/intangible **value** vs. notebooks-only workflow: time-to-insight, collaboration, governance, trust
   - Short-term **ROI** + long-term scalability and competitive advantage
6. **Scalability & Risk**
   - How can this pipeline **scale** across other models/functions?
   - Anticipated **risks**: e.g. model drift, shadow IT, poor adoption, over-engineering
   - **Mitigations**: CI/CD coverage, training, testing, stakeholder involvement
7. **Responsible AI & Governance**
   - Data quality, audit trails, model versioning, human oversight checkpoints
   - Ethical impact or regulatory relevance (e.g., fairness, explainability, GDPR readiness)
8. **Effort & Cost Overview**
   - Estimated talent, infra, and time needed to deliver and sustain the MLOps pipeline
9. **Change mgmt. considerations**
   - Prepare a clear adoption roadmap with executive sponsorship, stakeholder buy-in, user training, and feedback loops to drive sustained usage

On top of 1st submission expectations

Final project expectations and guidelines (Business prezo section)

Criterion	Fundamentals missing (0–4)	Basic attempt (5–6)	Good implementation (7–8)	Excellent – Industry grade (9–10)
Problem Definition	No business need identified	Vague or generic pain point	Clear problem + impact metric	Strong pain point with baseline KPI and urgency
Client Value & ROI	No differentiation from notebooks	Basic value noted	Tangible benefit + comparison to BAU	ROI articulated short & long-term; competitive insight shown
Solution Framing	No pipeline overview or benefits	Partial or technical-only framing	Business-level summary + impact	Executive-ready view with clear business functionalities
Scalability & Risk Awareness	Not considered	Mentioned but shallow	Risks & scale options described	Tradeoffs, risks, mitigations, and scale paths clearly outlined
Organizational Feasibility	No roles/skills/environment context	General effort listed	Roles, infra, time estimated	Clear roadmap to deploy, org impact understood
Governance & RAI	Not addressed	Basic mention of ethics/fairness	Governance partially considered	Auditability, explainability, compliance well framed
Presentation Clarity	Fragmented or off-topic	Some flow but unclear points	Structured and timed presentation (10')	Clear storyline, confident delivery, supported by visual clarity