Assignment 1

Completion requirements

**Opened:** Monday, 21 July 2025, 12:00 AM

**Due:** Monday, 11 August 2025, 11:55 PM

**“Build, Track, Package, Deploy and Monitor an ML Model using MLOps Best Practices”**

*(Choose either Iris (classification) or California Housing (regression) dataset)*

**Scenario**

You’ve been tasked with building a minimal but complete MLOps pipeline for an ML model using a well-known open dataset. Your model should be trained, tracked, versioned, deployed as an API, and monitored for prediction usage.

**Learning Outcomes**

* Use Git, DVC, and MLflow for versioning and tracking.
* Package your ML code into a REST API (Flask/FastAPI).
* Containerize and deploy it using Docker.
* Set up a GitHub Actions pipeline for CI/CD.
* Implement basic logging and optionally expose monitoring metrics.

**Technologies**

* Git + GitHub
* DVC (optional for Iris, useful for housing)
* MLflow
* Docker
* Flask or FastAPI
* GitHub Actions
* Logging module (basic); Optional: Prometheus/Grafana

**Assignment Tasks**

**Part 1: Repository and Data Versioning (4 marks)**

* Set up a GitHub repo.
* Load and preprocess the dataset.
* Track the dataset (optionally with DVC if using California Housing).
* Maintain clean directory structure.

**Part 2: Model Development & Experiment Tracking (6 marks)**

* Train at least two models (e.g., Logistic Regression, RandomForest for Iris; Linear Regression, Decision Tree for Housing).
* Use MLflow to track experiments (params, metrics, models).
* Select best model and register in MLflow.

**Part 3: API & Docker Packaging (4 marks)**

* Create an API for prediction using Flask or FastAPI.
* Containerize the service using Docker.
* Accept input via JSON and return model prediction.

**Part 4: CI/CD with GitHub Actions (6 marks)**

* Lint/test code on push.
* Build Docker image and push to Docker Hub.
* Deploy locally or to EC2/LocalStack using shell script or docker run.

**Part 5: Logging and Monitoring (4 marks)**

* Log incoming prediction requests and model outputs.
* Store logs to file or simple in-memory DB (SQLite).
* Optionally, expose /metrics endpoint for monitoring.

**Part 6: Summary + Demo (2 mark)**

* Submit a 1-page summary describing your architecture.
* Record a 5-min video walkthrough of your solution.

**Bonus (4 marks)**

* Add input validation using pydantic or schema.
* Integrate with Prometheus and create a sample dashboard.
* Add model re-training trigger on new data.

**Deliverables**

* GitHub repo link (code, data, model, pipeline)
* Docker Hub link (image)
* Summary document
* 5-min screen recording