

MLOPS Lab 2

Name : Ashiq Firoz
Roll : 2022BCD0013

Github Repository Link : <https://github.com/2022bcd0013-ashiq-firoz/lab2>

Job summary with metrics for all experiments

The screenshot shows the GitHub Actions interface for the repository 2022bcd0013-ashiq-firoz / lab2. The left sidebar contains sections for Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area displays 'All workflows' and '4 workflow runs'. Each run is listed with its name, commit hash, branch, event time, status, and duration. The runs are:

- Model-RandomForest, 100 trees, depth 15, testsplit-0.2, pre-processing – stand... (Commit a5bc63a, main, 16 minutes ago, 37s)
- Model-RandomForest, 50 trees, depth 10, testsplit-0.2, pre-processing – stand... (Commit e947969, main, 18 minutes ago, 30s)
- Model-Linear-lasso, regularize, testsplit-0.2, pre-processing – stand... (Commit 0ecd320, main, 22 minutes ago, 30s)
- Model-LinearReg, Default, testsplit-0.2, pre-processing – none (Commit ce35c65, main, 32 minutes ago, 32s)

The screenshot shows the job summary for the Model-LinearReg experiment. The left sidebar has sections for Summary, All jobs, Run details, Usage, and Workflow file. The main area includes a 'Model Evaluation Summary (Python 3.12)' section with details for Name: Ashiq Firoz, Roll Number: 2022BCD0013, Model: Linear Regression, Mean Squared Error (MSE): 0.5467, and R² Score: 0.2598. Below this is an 'Artifacts' section listing two files: training-artifacts-py3.11 and training-artifacts-py3.12, both with a size of 1.29 KB and a digest starting with sha256.

1. How did GitHub Actions improve experiment reproducibility?

GitHub Actions enforces a fixed execution environment, dependency versions, and scripted training steps. Every run is automatically triggered and executed identically on clean machines. This eliminates environment drift common in local or notebook-based experiments.

2. How easy was it to compare results across runs?

Comparison was straightforward since metrics were logged in a consistent JSON format and summarized per run. Each workflow execution stored artifacts and evaluation summaries. Python version matrices further enabled systematic cross-environment comparison.

3. What role does Git commit history play in experiment tracking?

Each experiment is tied to a specific commit hash, ensuring code–result traceability. Changes in performance can be directly mapped to code modifications. This creates an implicit, version-controlled experiment log.

4. What were the benefits of this approach compared to Lab 1?

Unlike Lab 1’s manual notebook runs, this approach automated training, evaluation, and logging. It supported multiple models and environments without manual intervention. Results were reproducible, auditable, and centrally stored.

5. What limitations does this approach have?

GitHub Actions has limited compute resources and execution time. It is less interactive than notebooks for exploratory analysis. Advanced experiment management (e.g., hyperparameter sweeps) requires additional tooling.