

In-Class Exercise: MLflow Experiment Tracking and Model Management

1 Objective

By the end of this exercise, you should be comfortable navigating the MLflow UI, understanding the relationship between experiments and runs, and explaining why MLflow is a core component of modern MLOps workflows.

2 Setup and Starting Point

You are provided with a Jupyter notebook that already contains the following components:

- A dataset loading step
- A simple model training pipeline (scikit-learn)
- Computation of training and validation metrics

You should **not** modify the model architecture, data preprocessing, or evaluation logic unless explicitly instructed. Your task is to integrate MLflow into the existing workflow in a clean and meaningful way.

Before proceeding, ensure that:

- MLflow is installed in your environment
- You can launch the MLflow UI using `mlflow ui`
- You can access the UI in your browser

3 Part 1: Creating and Using an MLflow Experiment

Run the code once as it is provided and observe what appears in the MLflow UI. At this stage, runs may be logged under the default experiment or may not be tracked at all.

Next, create a new MLflow experiment with a descriptive name. Update the code so that all future runs are logged under this experiment.

Re-run the code and verify in the MLflow UI that the run appears under the correct experiment. Confirm that the experiment name is visible and that the run is no longer associated with the default experiment.

4 Part 2: Logging Runs, Parameters, and Metrics

Wrap the model training code inside an explicit MLflow run using `mlflow.start_run()`.

Identify at least three meaningful parameters that already exist in the code or from the sklearn documentation (google sklearn lbfgs). Log these values using MLflow parameter logging in the UI.

Next, log at least two metrics other han accuracy. Ensure that these metrics appear correctly in the MLflow UI and are associated with the current run.

Re-run the code and confirm that parameters and metrics are visible and correctly recorded.

5 Part 3: Comparing Multiple Runs

Modify only the parameter values and re-run the code at least three times. Each run should represent a distinct training attempt.

After completing these runs, use the MLflow UI to compare them. Identify which run achieved the best performance and observe how changes in parameters affected outcomes.

You should be able to answer the following questions by inspecting the UI: Which run performed best? Which parameter values seem most promising? Are the observed differences significant or marginal?

6 Part 4: Using MLflow Tags for Organization

Add MLflow tags to each run to help distinguish them. Examples include a tag indicating the run type (e.g., baseline or modified), or a tag containing your name.

Tags should be used to describe context and intent rather than numerical values.

Reflect briefly on how tags differ from parameters and why both are needed.

7 Part 6: Registering a Model

Select the best-performing run based on validation metrics. From that run, register the trained model in the MLflow Model Registry using a clear model name .

Verify that the model appears in the registry and that it is linked to the correct run. You are not required to deploy the model or assign stages.