In-Class Exercise: MLflow Deep dive

Learning goals

By the end of this short in-class exercise, you should be able to:

- Use MLflow to log multiple models (with autologging) and view their runs and metrics.
- Search for the best logged model based on a performance metric.
- Register a selected model into the MLflow Model Registry and manage *champion/challenger* roles via aliases and tags.
- Enable MLflow *dependency locking* and understand how it changes the model's environment specification.

Dataset and models

For this exercise you will use the **Adult Income** classification dataset ("Wine Quality Dataset") from the UCI repository via OpenML. In `scikit-learn`, you can load it as follows (you do not need to paste this into your submission; it is just a hint):

```
from sklearn.datasets import fetch_openml
wine = fetch_openml("wine-quality-red", version=1, as_frame=True)
X = wine.data
y = (wine.target.astype(int) >= 6).astype(int)
```

The prediction is like this: Is the wine good quality or not, 1 is good, 0 is bad. More than You will train several standard classifiers on this dataset:

- Logistic Regression
- Decision Tree
- Random Forest
- AdaBoost

You may perform only minimal preprocessing (e.g., simple handling of categorical features or using a subset of columns) so that training runs quickly.

Baseline model logging with MLflow autolog (no locking)

In this part, you will simply get MLflow to track your experiments.

1. Create a new Python file or notebook. Set up MLflow to track experiments
2. Enable **MLflow autologging** for scikit-learn and set an experiment name

3. Train at least **four models** on the Adult dataset: one Logistic Regression, one Decision Tree, one Random Forest, and one AdaBoost classifier. For all of these, also vary two hyperparameters (for example, `max_depth`, `n_estimators` for Random Forest) so that you create multiple runs with different metrics. Check skleaqrn documentation for these classifiers accordingly

4. For each run, record evaluation metrics such as `accuracy, precision, recall, f1-score` on a held-out test set

5. Open the MLflow UI and verify that:

    - Multiple runs were created under your experiment.
    - Models were logged as artifacts within those runs.

Enable dependency locking and compare environments

Now you will repeat the logging with MLflow *dependency locking* enabled and observe the difference.

1. Ensure you have the `uv` package installed in your environment

2. At the beginning of your script or notebook, before importing MLflow, set the environment variable accordingly to enable dependency locking.

3. Repeat training for the top two performing models in terms of accuracy **two** of your earlier models, again using MLflow autologging and the same experiment or a new experiment.

4. After the runs finish, verify and note the difference in:

    - How many packages are listed.
    - Whether versions are pinned exactly and whether transitive dependencies appear.

Part C: Search for the best model and register a champion

Next, you will use the MLflow API to find the best logged model and register it as a *champion* in the Model Registry.

1. Using Python and MLflow, call `mlflow.search_logged_models` to retrieve logged models for your experiment. Order them by *class_1_recall* and select the **top** model.

2. Register this top model into the MLflow Model Registry, giving it a model name accordingly

3. Use `MlflowClient` to:

    - Add a short description to this model version
    - Set a model version tag that marks this version as `role=champion`.
    - Assign the alias `champion` to this version

Load the champion model and inspect metadata

Finally, you will pretend you are a consumer of the model and retrieve the champion by alias.

1. Write a short Python snippet that loads the champion model
2. Use the model to make predictions on a small batch of test examples and print the predictions.
3. Retrieve the metadata for the champion model (for example, run ID, version, source, and tags) and print them and double check ifn they match with the original run metadata.

What to show the instructor

At the end of the exercise, be prepared to briefly show the instructor:

- The MLflow UI with multiple runs and logged models.
- The environment files for a run without locking and a run with locking
- The registered model in the Model Registry with the `champion` alias.
- A short snippet (or notebook cell) that loads the champion by alias and prints both predictions and basic metadata