Delivery 5 – Experiment Tracking & Model Versioning with MLflow

Updated Instructions — Students must create a new `train_model.py` with MLflow logging

Goal

In this delivery, students will **create a new `train_model.py` script** that integrates **MLflow experiment tracking**, logs multiple models, compares performance, stores artifacts, and automatically selects & registers the best model.
This simulates real MLOps workflows used in production-grade machine learning systems.

Before You Start — Install MLflow

Students must install MLflow locally:

```
pip install mlflow
```

The new `train_model.py` **will not run unless MLflow is installed.**

🎯 Tasks

1. Create a new `train_model.py` with full MLflow logging

Students must **write a new file** named **`train_model.py`** that extends the training pipeline by logging everything to MLflow.

Your new training script must log:

Model Information

- model name
- parameters
- tags (e.g., version `"v1.0.0"`)

Metrics

- accuracy
- F1-macro
- precision
- recall
- ROC-AUC (when available)

Artifacts

- confusion matrix image
- classification report text
- serialized model file (`.joblib`)

MLflow should automatically group all runs under one experiment.

---

2. Train multiple models (Model Zoo)

Once `train_model.py` is created, run:

```
python train_model.py
```

The script must:

✔ Train **four models** (RandomForest, LogisticRegression, SVM, KNN)
✔ Log each run to MLflow
✔ Save metrics, artifacts, and models
✔ Identify the **best model using F1-macro score**

---

3. Save the best model locally

Your script must automatically write two files after training:

```
app/model.joblib
app/model_meta.json
```

Example `model_meta.json`:

```
{
  "best_model": "SVM",
  "metrics": {
    "accuracy": 0.967,
    "f1_macro": 0.960
  },
  "mlflow_run_id": "abc123...",
  "version": "v1.0.0"
}
```

These files will be used later by the Streamlit interface.

---

4. Register the best model in the MLflow Model Registry

Your new `train_model.py` must also:

- create (or reuse) a registered model called **`IrisModel`**
- create a **new version** each time training runs
- store the model artifact under the MLflow run that produced it

Expected result inside MLflow UI:

```
Models → IrisModel → Version 1
```

…and additional versions for subsequent training runs.

---

5. Explore your experiment in MLflow UI

Launch MLflow UI:

```
mlflow ui --backend-store-uri ./mlruns --port 5000
```

Open the dashboard:

```
http://localhost:5000
```

You should see:

✔ an experiment: **iris-model-zoo**
✔ 4 runs (one per model)
✔ logged metrics
✔ confusion matrix images
✔ classification reports
✔ registered model: **IrisModel**

---

6. Verify Streamlit displays metadata from MLflow

Run the app:

```
streamlit run app/app.py
```

The footer of the app must display:

- version
- best model name
- MLflow run ID
- accuracy
- a clickable link to MLflow UI

Example footer:

```
Version: v1.0.0 • Best model: SVM • MLflow run: 626071… • Accuracy: 
0.967
```

✅ You Are Done When…

MLflow Shows:

- experiment **iris-model-zoo**
- 4 logged runs
- all metrics + artifacts
- a registered model: **IrisModel v1**

Local Files Exist:

- `model.joblib`
- `model_meta.json`

Streamlit App Displays:

- model version
- best model
- MLflow run ID
- accuracy