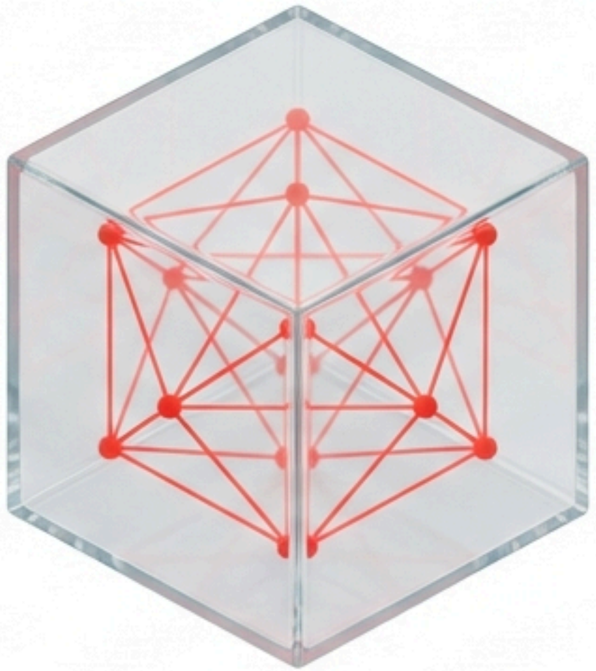


Day 12: MLflow Basics

From Experiment Tracking to Model Registry

14 DAYS AI CHALLENGE

Tracking the Invisible

The Mission Objectives

In this session, we will execute a complete MLflow lifecycle run within a Databricks environment. Success is defined by four key milestones.

1. Train

Build a simple regression model using Scikit-Learn.

2. Log

Capture parameters, metrics, and model artifacts.

3. View

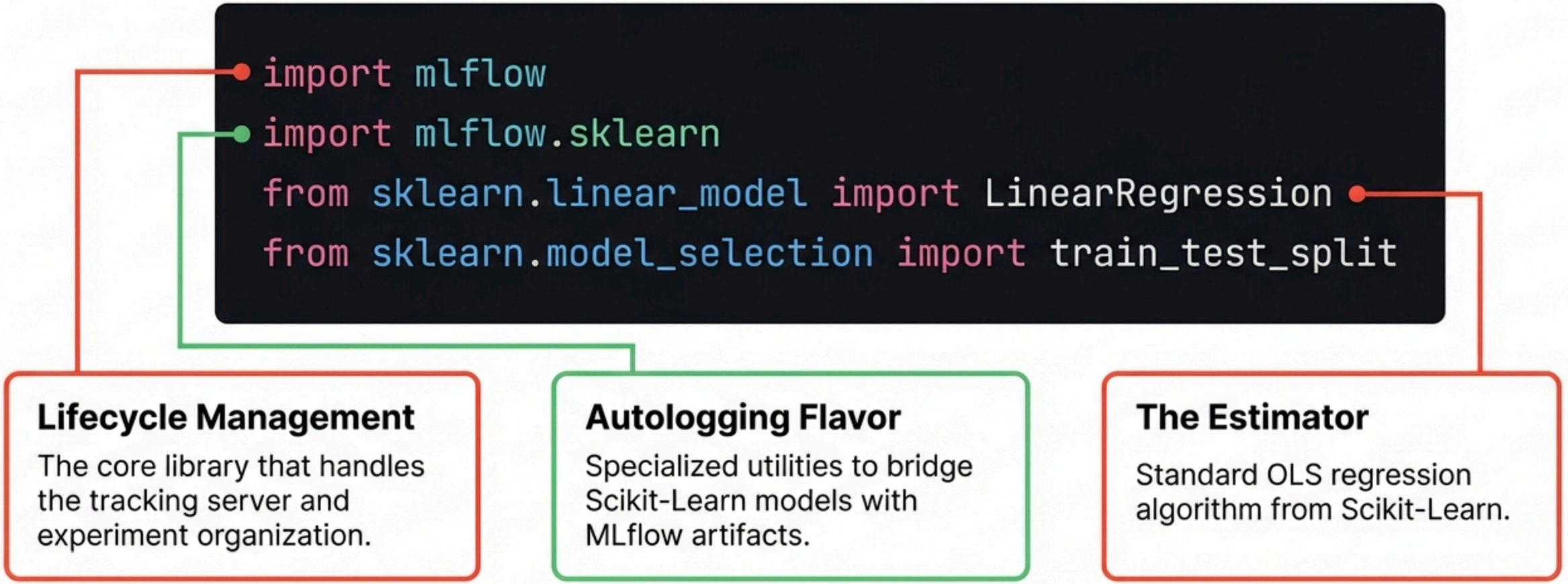
Verify the experiment results in the MLflow UI.

4. Compare

Analyze run performance and run history.

The Toolkit

Initializing the necessary libraries for tracking and modeling.



Data Preparation: The Micro-Batch

```
sample_data = pd.DataFrame({
    'views': [1, 2],
    'cart_adds': [0, 1],
    'purchases': [0, 1]
})
df = spark.createDataFrame(sample_data)
df = df.toPandas()
```

Note: We are intentionally creating a tiny 2-row dataset for demonstration mechanics.

pandas DataFrame

views	cart_adds	purchases
1	0	0
2	1	1

Total Size: 2 Rows

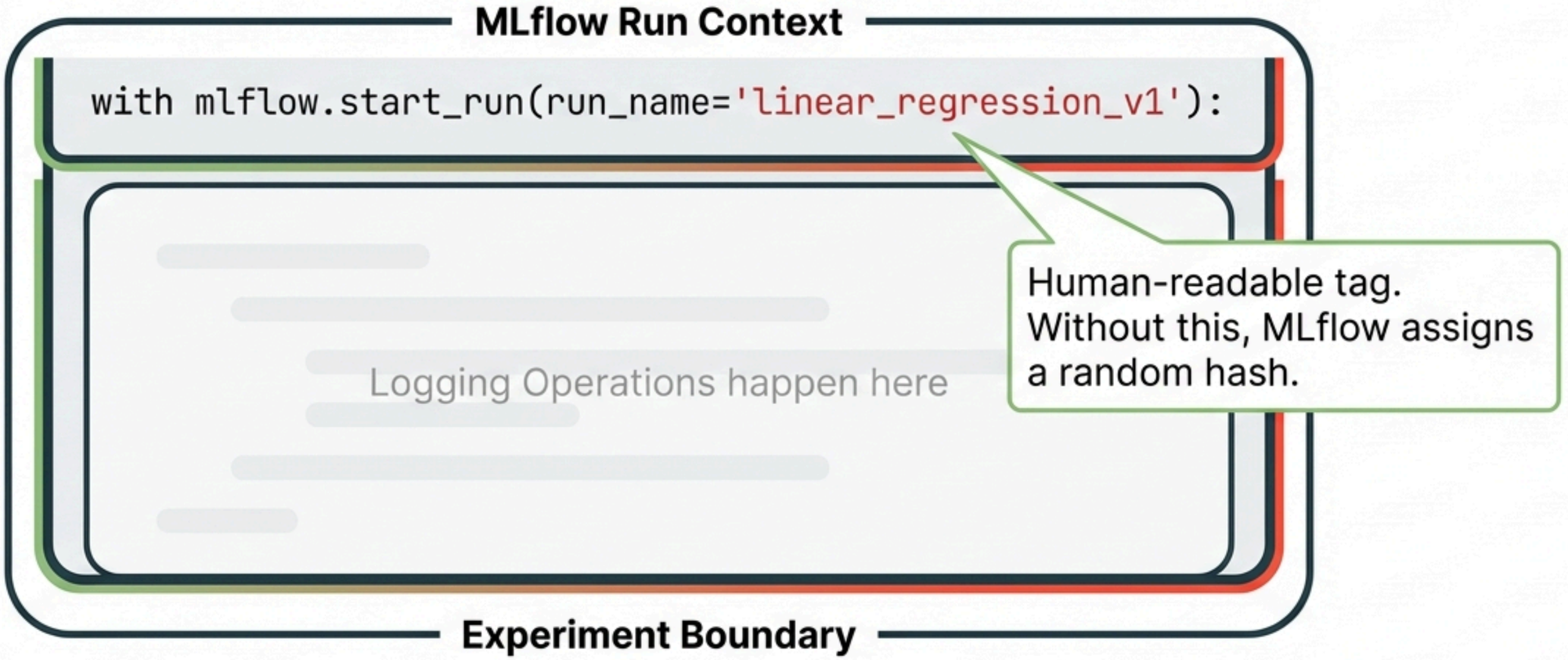
Defining the Training Ground

```
x = df[['views', 'cart_adds']]
y = df['purchases']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Reference the train_test_split code line.

Initiating the Run

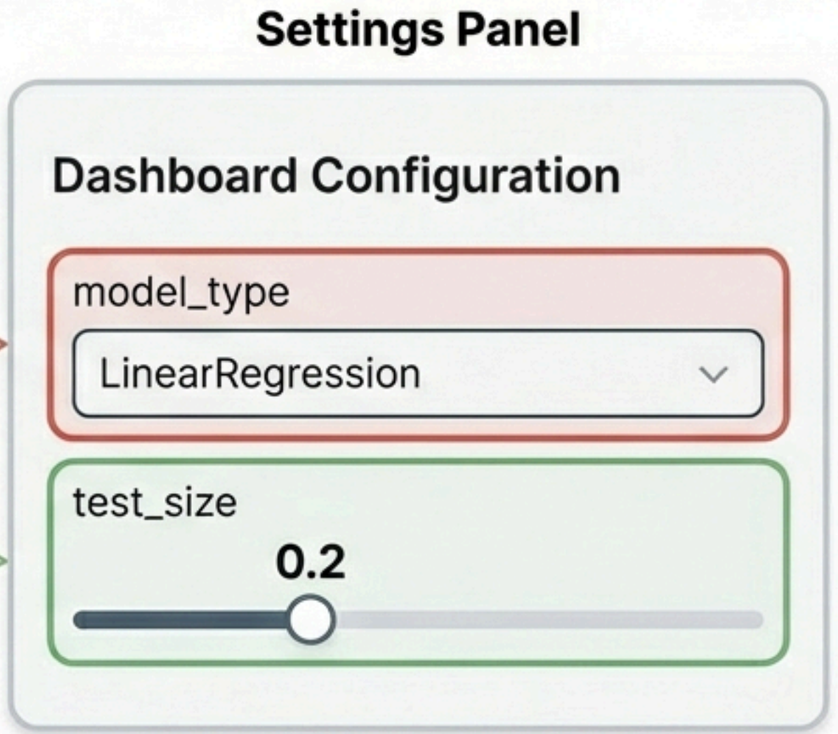


Reference the **start_run** code block.

Step 1: Logging Parameters

Tracking the inputs configuration.

```
# Log parameters
mlflow.log_param('model_type', 'LinearRegression')
mlflow.log_param('test_size', 0.2)
```

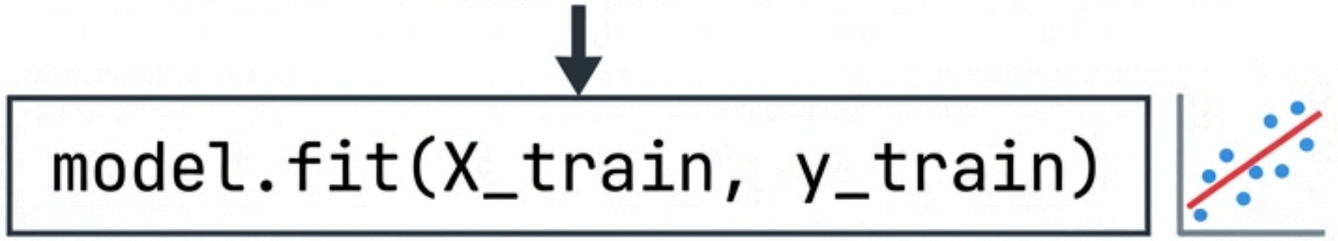


The Training Loop

Standard Scikit-Learn execution flow

```
model = LinearRegression()
```

Initialization



```
model.fit(X_train, y_train)
```


Fitting / Learning

```
score = model.score(X_test, y_test)
```

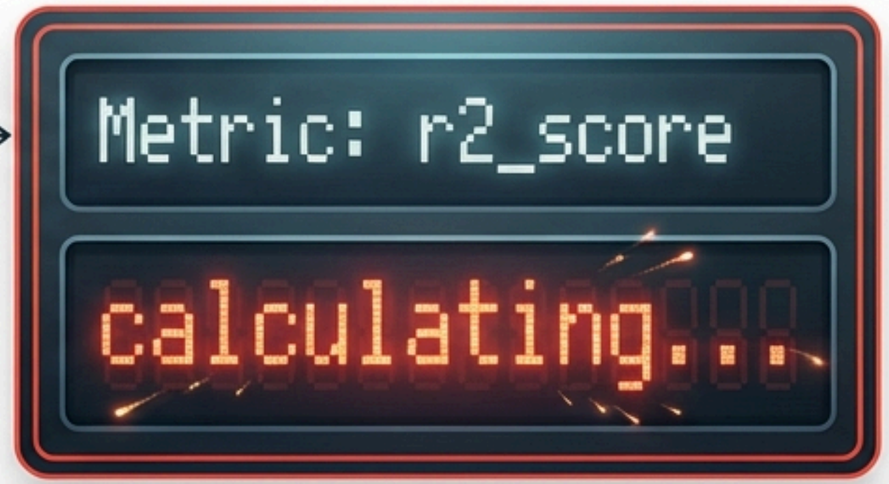
Evaluation

Step 2: Logging Metrics

Tracking the performance outputs.

```
mlflow.log_metric('r2_score', score)
```

Outputs



Metrics are **quantitative** measures (Accuracy, MSE, R2) produced **AFTER** training.

Reality Check: The Data Warning

Incident

WARNING

```
UndefinedMetricWarning: R^2 score is not well-defined with less than two samples.
```

```
R^2 Score: nan
```

The Explanation

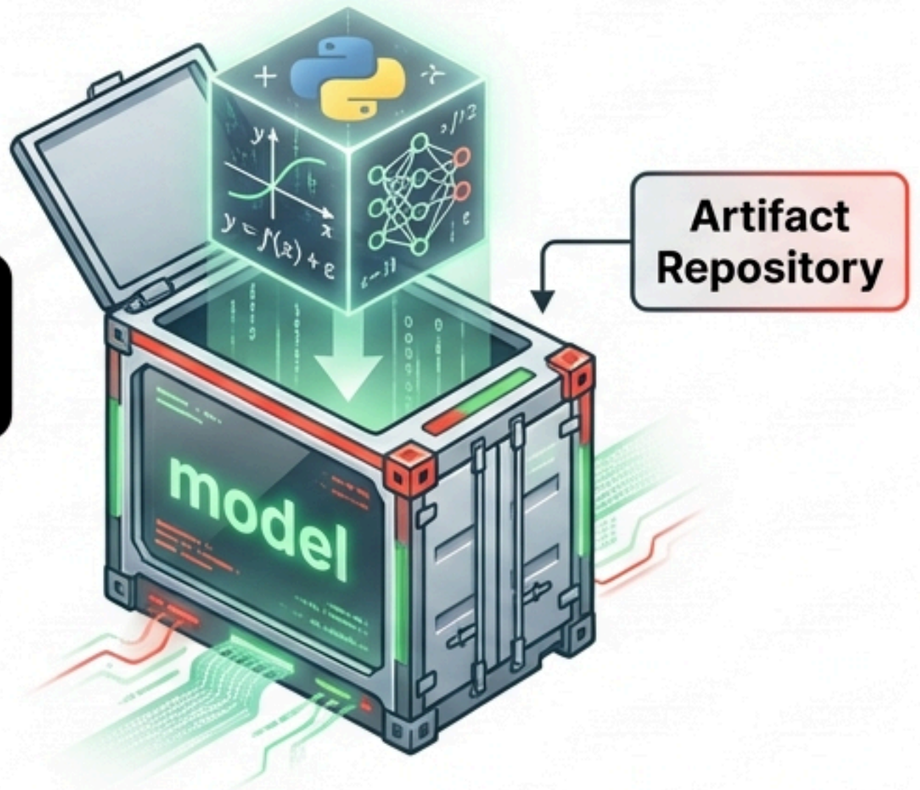
1. Dataset Size: 2 Rows → **2. Test Split (20%):** < 1 Row → **3. Variance Calculation:** Requires >= 2 samples. Result = Undefined.

takeaway The code logic is correct, but the data volume is **insufficient**. This results ' (Not a Number) metric.

Step 3: Immortalizing the Model

Saving the artifact for future inference.

```
mlflow.sklearn.log_model(model, 'model')
```



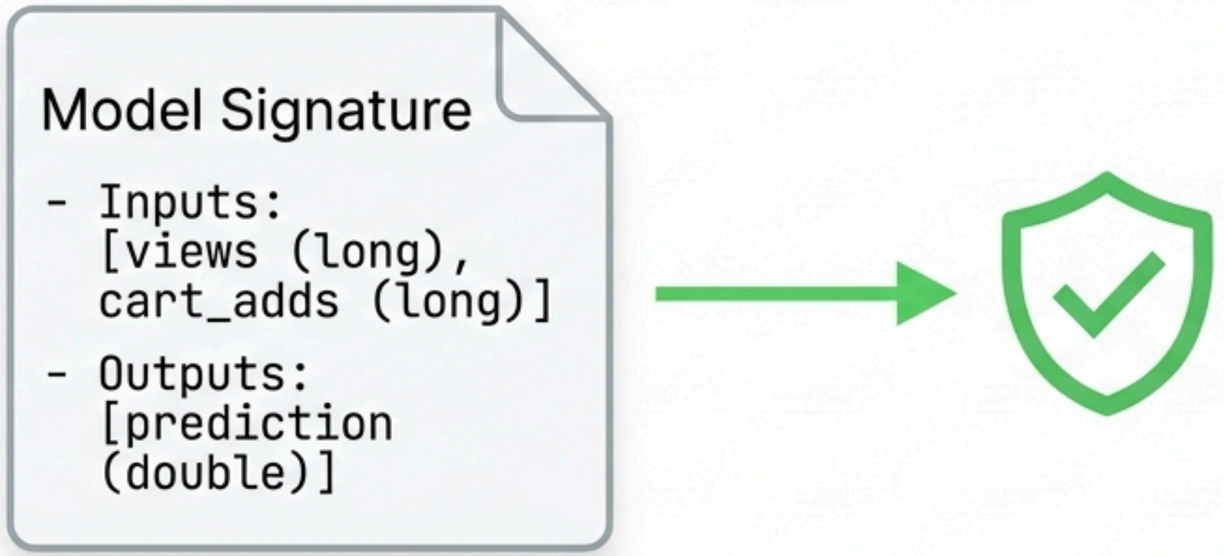
This step **serializes the model object**. It can now be loaded from the MLflow UI or API for prediction in any other environment.

Pro-Tip: Model Signatures

The Warning

```
WARNING: Model logged without a signature and input example. Please set
'input_example' parameter when logging the model to auto infer the model signature.
```

The Solution



Signatures define the expected input schema. This prevents downstream errors when serving **the model to production API endpoints.**

The MLflow UI View

Visualizing the Experiment Run

Run Name | Date	Date	Source	Parameters	Metrics	Artifacts
linear_regression_v1	Just now	Notebook	**model_type**: LinearRegression **test_size**: 0.2	**r2_score**: nan	

Result of "Logged 1 run" in notebook

Mission Debrief

Train simple regression model [COMPLETED]

Log parameters & metrics [COMPLETED]

Log model artifacts [COMPLETED]

Compare runs [PARTIAL - DATA LIMIT]

The pipeline is functional. The mechanics of MLflow are verified.

Next Steps

Scale Up

Swap the dummy data for a real dataset (e.g., Boston Housing) to fix the variance issue.

Verify

Re-run the notebook and observe a valid 'r2_score' in the logs.

Explore

Add the `'input_example'` parameter to your `log_model` call to fix the signature warning.

Day 12 Complete