INDIAN DATA CLUB

CODE BASICS

databricks

14 DAYS

AI CHALLENGE

DAY 14

Topic:

AI-Powered Analytics: Genie & Mosaic AI

Challenge:

1. Train simple regression model
2. Log parameters, metrics, model
3. View in MLflow UI
4. Compare runs

#DatabricksWithIDC

Databricks 14-Day AI Challenge: The Final Day

This final challenge from the 14-day AI event focuses on **"AI-Powered Analytics: Genie & Mosaic AI,"** involving a streamlined four-step machine learning workflow.

1. Train a Simple Regression Model

Build and train a basic regression model as the first step.

2. Log Parameters, Metrics & Model

Record the key components of your experiment for tracking.

3. View in MLflow UI

Use the MLflow user interface to visualize the logged results.

4. Compare Runs

Analyze and compare the performance of different experiment runs.

INDIAN DATA CLUB

CODE BASICS

databricks

DAY 14

AI CHALLENGE

AI-Powered Analytics

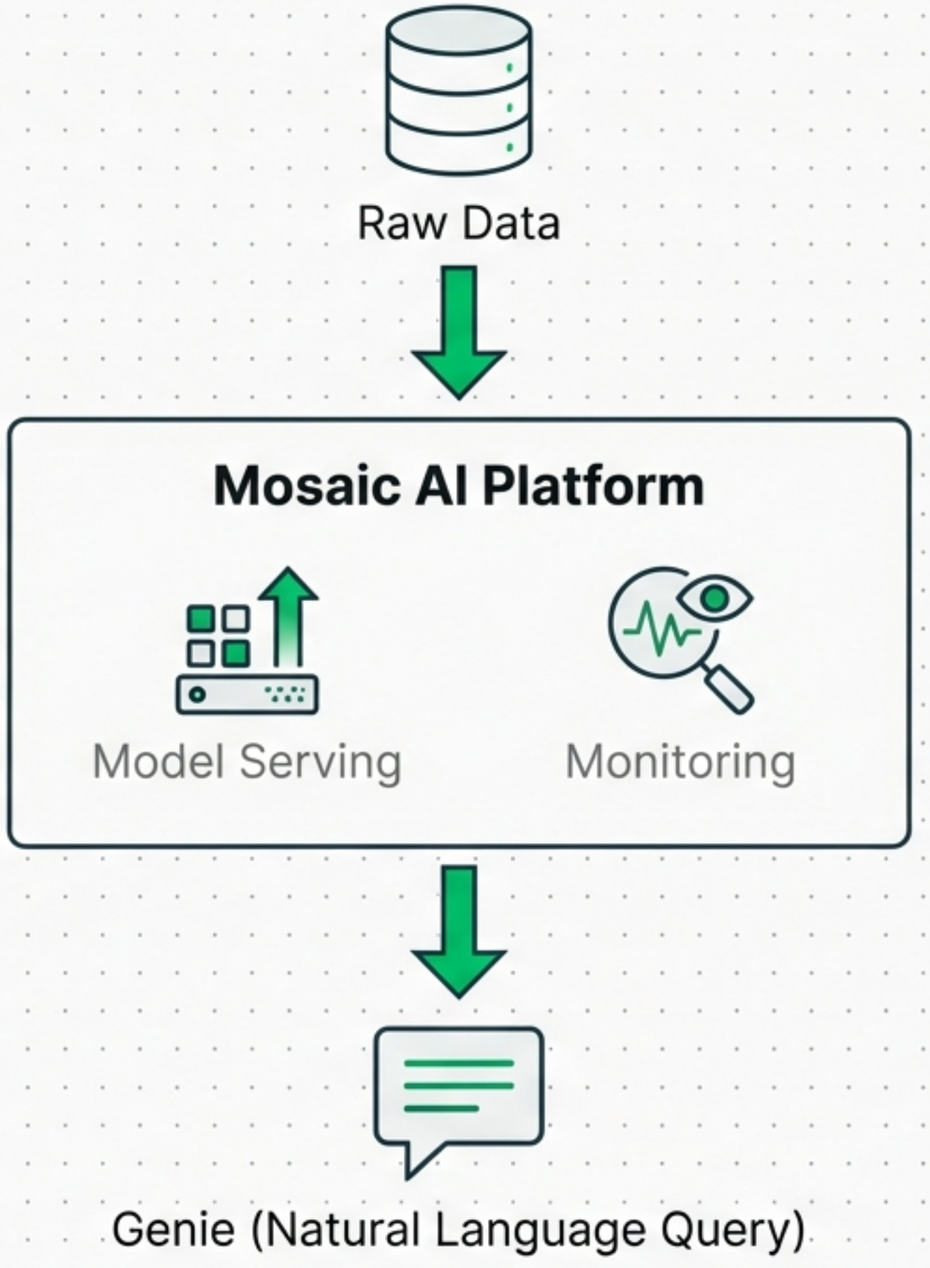
Mastering Genie, Mosaic AI, and the MLflow Lifecycle

#DatabricksWithIDC

The Goal is Intelligent Analytics

Today's topic focuses on two pillars of the Databricks intelligence platform:

- **Mosaic AI**: The unified tooling infrastructure that allows you to build, deploy, and monitor AI solutions securely.
- **Genie**: An AI-powered natural language query engine that allows business users to ‘talk’ to their data.



The Engineering Foundation: MLflow

We cannot build reliable AI without rigorous tracking. Today's challenge uses MLflow to ensure our models are reproducible and audit-ready.

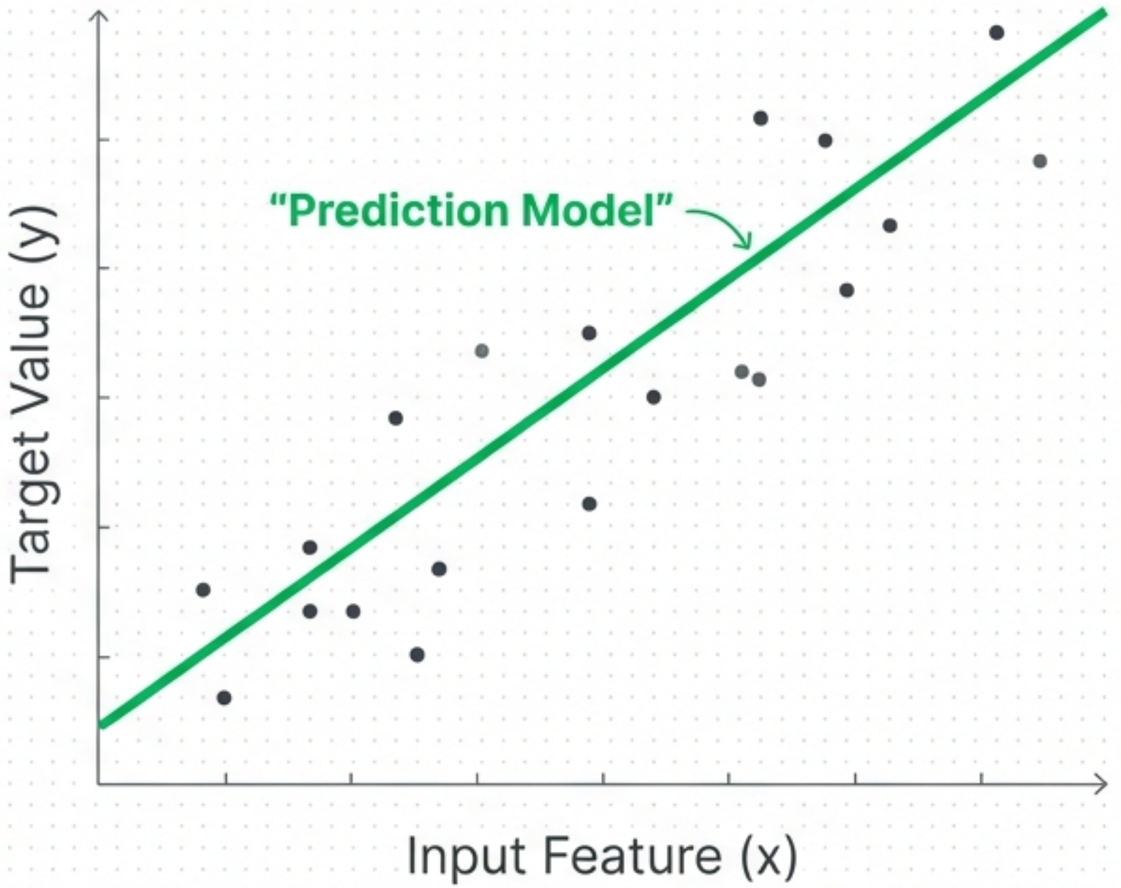
1. [ ] Train simple regression model
2. [ ] Log parameters, metrics, model
3. [ ] View in MLflow UI
4. [ ] Compare runs

THE MISSION

Step 1: The Concept (Simple Regression)

The Task: Train a simple regression model.

The Logic: We are predicting a continuous output value (y) based on input data (x). Unlike classification (cat vs. dog), regression estimates quantity.



Step 1: Execution

```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Prepare the data
# X represents our features, y is our target
X_train, X_test, y_train, y_test = train_test_split(X, y)

# Initialize the model
model = LinearRegression()

```

Instantiating the Scikit-Learn model object.

Step 2: The Audit Trail (Logging Parameters)

Science requires reproducibility. We don't just run the model; we record *how* we ran it by logging configuration parameters.

```
import mlflow

# Start the MLflow run context
with mlflow.start_run():

    # Log the configuration "settings"
    mlflow.log_param("fit_intercept", True)

    # Fit the model to the training data
    model.fit(X_train, y_train)
```

Step 2: Defining Success (Metrics & Artifacts)

We record performance metrics (outputs) and the model file itself (artifacts). This creates a permanent versioned record.

```
# Calculate error metric
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)

# Log the performance score
mlflow.log_metric("mse", mse)
```

→ MSE: 0.24

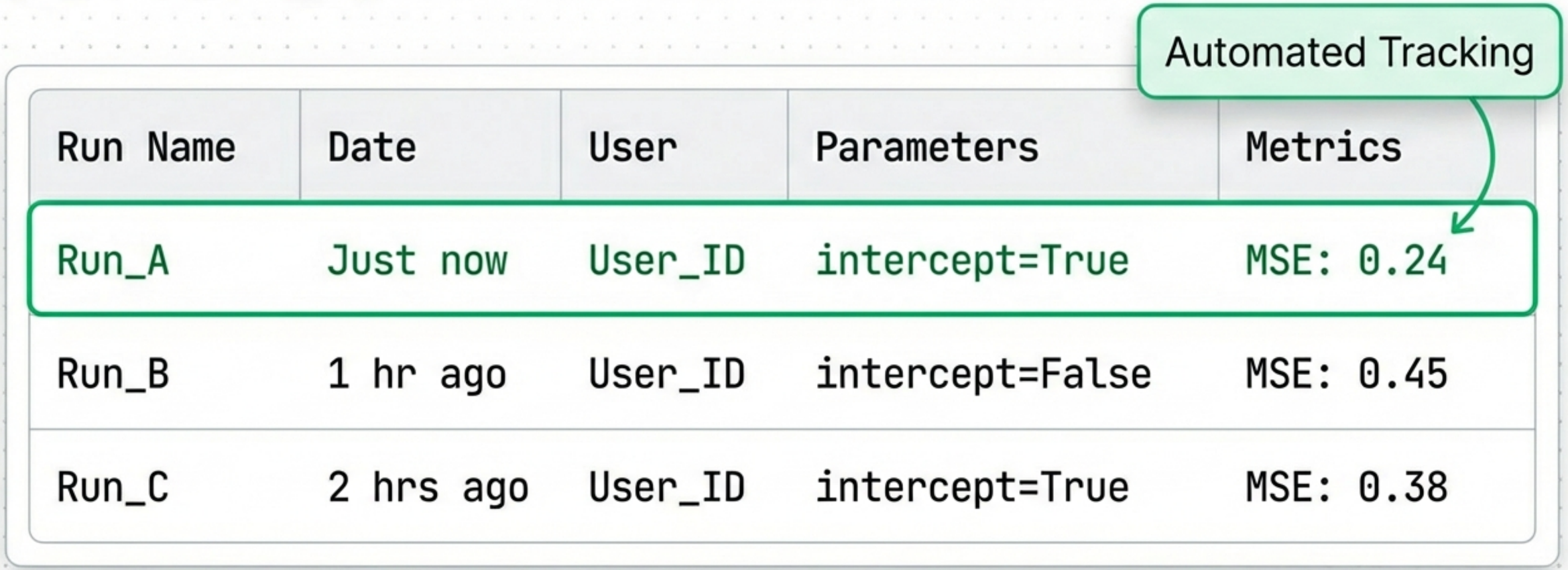
```
# Save the model file for future deployment
mlflow.sklearn.log_model(model, "regression-model")
```

→ 

model.pkl

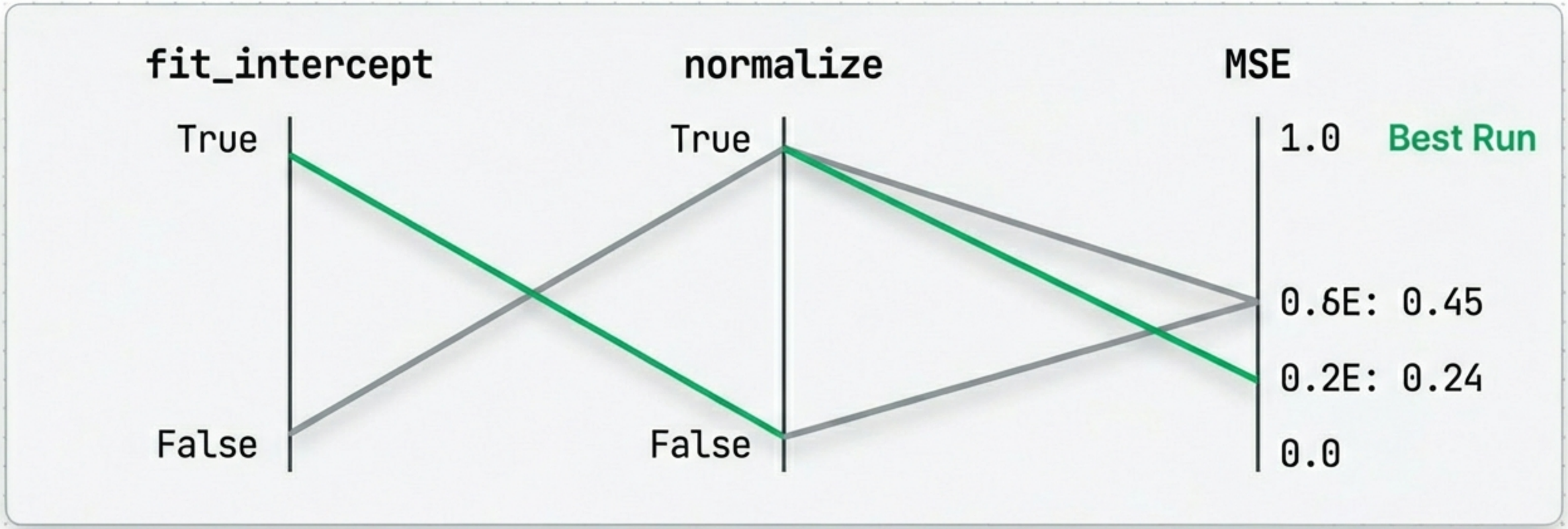
Step 3: Navigating the MLflow UI

Where does the code go? Into the Experiment UI. This is your command center.



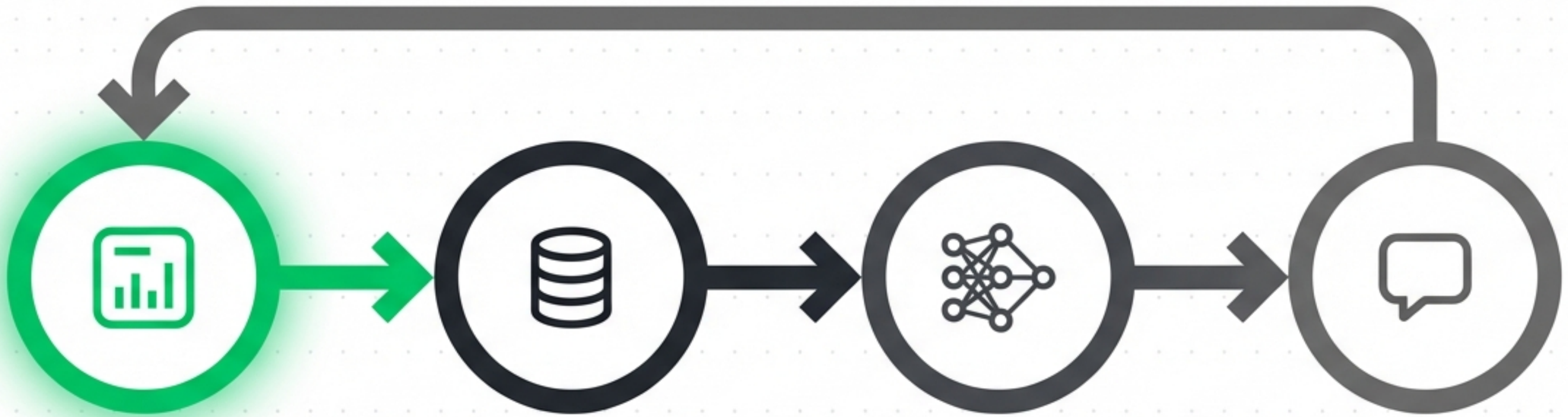
Step 4: Data-Driven Decisions

The final step is comparison. By running the training code with different parameters, we can visually identify the best model.



The Bigger Picture

Today’s regression model is the **atomic unit** of the **entire** AI ecosystem.



Mission Accomplished

- [x] Train simple regression model
- [x] Log parameters, metrics, model
- [x] View in MLflow UI
- [x] Compare runs

Share Your Progress

You have successfully bridged the gap between analytics theory and MLOps practice.

#DatabricksWithIDC

Indian Data Club Code Basics databricks