MLOps and LLM Observability - Day 2

Karthikeyan Vaiyapuri

01

Overview

Today's Learning Objectives

01

Set up complete MLOps environment with industry- standard tools

02

Master MLflow for experiment tracking and model management

03

Implement model versioning and dataset management

04

Hands-on practice with real MLflow workflows

Objectives

Production-Ready Environment

01 **Consistency:** Same environment across development, testing, production

02 **Reproducibility:** Anyone can recreate your results

03 **Scalability:** Easy to deploy and scale applications

04 **Collaboration:** Team members work with identical setups

What is Anaconda?

- **Package manager** for Python and R
- **Environment manager** for isolated development
- **Pre-installed libraries** for data science
- **Cross-platform** support (Windows, Mac, Linux)

02

Introduction to MLflow

What is MLflow?

Open-source platform for ML lifecycle management

Experiment tracking and comparison

Model packaging and deployment

Model registry for version control

MLflow Components

- **MLflow Tracking:** Record and query experiments
- **MLflow Projects:** Package ML code for reuse
- **MLflow Models:** Deploy models to various platforms
- **MLflow Registry:** Centralized model store

03

MLflow Architecture

Core Components Overview

MLFlow Tracking	MLFlow Projects	MLFlow Models	MLFlow Registry
• Experiments • Runs • Parameters • Metrics	• Code Package • Dependencies • Entry Points	• Model Format • Deployment • Serving	• Model Store • Versioning • Stage Mgmt

04

MLflow Installation and Setup

Installation Methods

pip install mlflow

conda install -c conda-forge mlflow

pip install mlflow[extras]

Verify Installation

mlflow --version

mlflow ui

Configuration Options

Options

Tracking Server: Remote or local

Backend Store: SQLite, MySQL, PostgreSQL

Artifact Store: Local, S3, Azure, GCS

05

Flask Integration with MLflow

Why Flask + MLflow?

Benefits

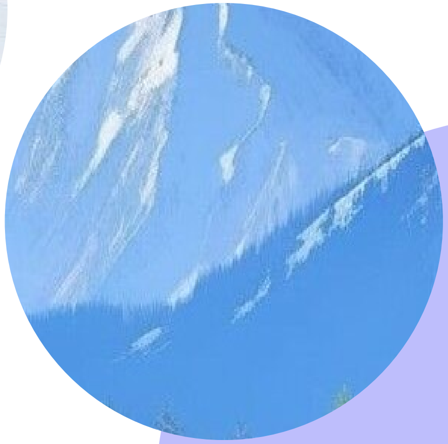
- **Model serving** through REST APIs
- **Integration** with MLflow model registry
- **Custom deployment** logic
- **Lightweight** web framework

Key Integration Points

- **Load models** from MLflow registry
- **Track predictions** back to MLflow
- **Monitor model performance**
- **A/B test** different model versions

Example Architecture

```
Client Request → Flask API → MLflow Model → Response
      ↓
MLflow Tracking ← Prediction Logs ← Model Metrics
```



06

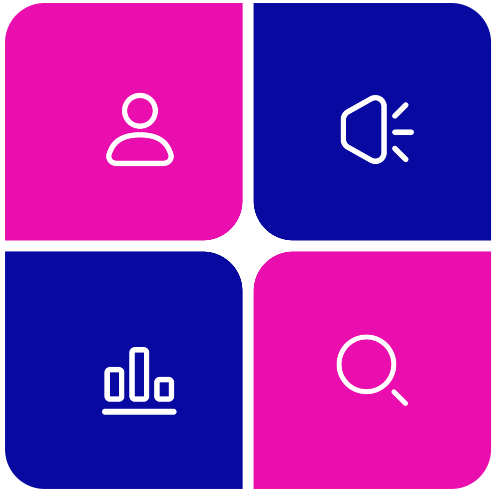
MLflow Tracking Fundamentals

Key Concepts

Experiment: Collection of related runs

Parameters: Input values (hyperparameters)

Artifacts: Files (models, images, data)



Run: Single execution of ML code

Metrics: Numeric values that change over time

Concepts Breakdown

Tracking Workflow

Workflow Steps

1. **Start experiment** or use default

2. **Log parameters** before training

3. **Log metrics** during/after training

4. **Save artifacts** (model, plots, data)

5. **End run** and review results

07

Logging Parameters and Metrics

Parameters vs Metrics

Comparison

Parameters	Metrics
Static values	Dynamic values
Set once per run	Can change over time
Hyperparameters	Performance measures
String/Number	Numeric only

Code Examples

```python
import mlflow

# Start run
with mlflow.start_run():
    # Log parameters
    mlflow.log_param("learning_rate", 0.01)
    mlflow.log_param("n_estimators", 100)

    # Log metrics
    mlflow.log_metric("accuracy", 0.95)
    mlflow.log_metric("loss", 0.05)

    # Log multiple metrics
    mlflow.log_metrics({
        "precision": 0.92,
        "recall": 0.88
    })
```

08

Artifact Management

What are Artifacts?

01

Model files: Trained models, weights

02

Data files: Datasets, preprocessed data

03

Plots/Images: Visualizations, reports

04

Code files: Scripts, notebooks

Types

Logging Artifacts

```
# Log single file
mlflow.log_artifact("model.pkl")

# Log directory
mlflow.log_artifacts("plots/", artifact_path="visualizations")

# Log model with metadata
mlflow.sklearn.log_model(
    model,
    "model",
    registered_model_name="my_model"
)
```

Artifact Storage

Storage Solutions

Local filesystem (default)

Cloud storage (S3, Azure, GCS)

Network filesystems

09

Model Versioning Strategy

Why Version Models?

01 Track model evolution

02 Enable rollbacks

03 Compare performance

04 Reproducibility

05 Compliance and auditing

Versioning Levels

1. Code versioning: Git commits

2. Data versioning: Dataset snapshots

3. Model versioning: MLflow registry

4.Environment versioning: Docker images

Best Practices

- Semantic versioning (v1.2.3)
- Document changes

- Tag important versions
- Automate versioning

10

MLflow Model Registry

Registry Features

Centralized model store

Version management

Stage transitions
(Staging → Production)

Model lineage
tracking

Approval workflows

Model Stages

None → Staging → Production → Archived

Registry Operations

```
# Register model
mlflow.register_model(
    "runs:/12345/model",
    "MyModel"
)

# Transition stage
client = mlflow.MlflowClient()
client.transition_model_version_stage(
    name="MyModel",
    version=1,
    stage="Production"
)
```

11

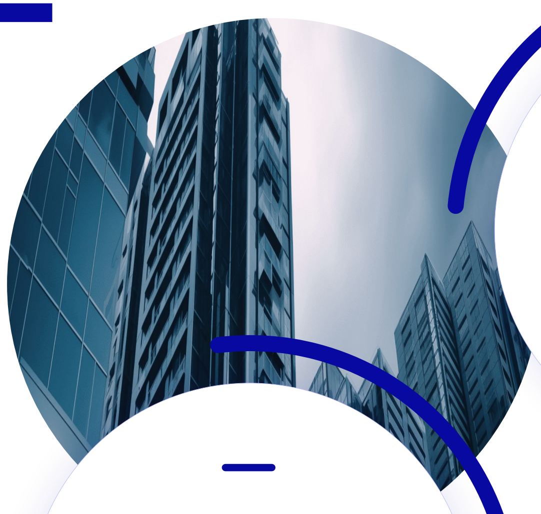
Dataset Versioning with MLflow

Dataset Tracking Challenges

Challenges

Large file sizes

Data drift detection



Lineage tracking

Storage efficiency

MLflow Dataset Features

```python
import mlflow.data
from mlflow.data.pandas_dataset import PandasDataset

# Create dataset
dataset = mlflow.data.from_pandas(
    df,
    source="data/train.csv",
    name="training_data"
)

# Log dataset
with mlflow.start_run():
    mlflow.log_input(dataset, context="training")
```

Dataset Versioning Benefits

01 Automatic checksums

02 Schema validation

03 Source tracking

04 Experiment reproducibility

Benefits

12

Experiment Organization

Experiment Hierarchy

```
Project/
├── experiment_1/
│   ├── run_1/
│   ├── run_2/
│   └── run_3/
├── experiment_2/
│   ├── run_1/
│   └── run_2/
└── experiment_3/
    └── run_1/
```

Naming Conventions

Descriptive names: "customer_churn_rf_v1"

Team prefixes: "ds_team_sentiment_analysis"

Date stamps: "model_training_20240727"

Version tags: "baseline_v1.0"

Organization Tips

01

One experiment per major approach

02

Use tags for filtering

03

Document objectives

04

Archive old experiments

13

MLflow UI Navigation

Main Sections

Experiments: View all experiments

01

Models: Access model registry

02

Runs: Detailed run information

03

Compare: Side-by-side comparisons

04

Key Features

Features

Search and filter runs

Compare parameters/metrics

Sort by metrics

View run details

Download artifacts

Collaboration Benefits

Shared experiment tracking

Team visibility

Result sharing

Knowledge transfer

Thanks

Karthikeyan Vaiyapuri