Mlflow – Managing the ML Lifecycle

Introduction

Machine Learning models don’t stop at training.

Need to manage:

- Experiments
- Model versions
- Reproducibility
- Deployment

Mlflow = Open-source platform for ML lifecycle

Why Mlflow?

Common ML problems:

- Which hyperparameters gave me 95% accuracy?
- How do I reproduce my colleague’s experiment?
- Which version of the model is in production?

Mlflow solves these with tracking, packaging and model registry.

Mlflow Components (Overview)

Mlflow Tracking: Log and compare experiments

Mlflow Projects: Reproducible ML code

Mlflow Models: Save and serve models easily

Mlflow Model Registry: Centralized model store

Mlflow Tracking

Logs:

- Parameters (e.g. learning rate)
- Metrics (accuracy, loss, F1-score)
- Artifacts (plots, datasets, models)
- Code versions

Example use:

```python
import mlflow
mlflow.log_param("learning_rate", 0.01)
mlflow.log_metric("accuracy", 0.95)
```

Mlflow Projects

Standardizes ML code for sharing/reuse.

Uses conda.yaml or Dockerfile to capture dependencies.

Example: run colleague's experiment with:

mlflow run https://github.com/user/repo

Mlflow Models

Supports multiple ML frameworks (scikit-learn, PyTourch, TensorFlow, XGBoost, etc.)

Models are stored in a standard format -> Easy to deploy.

Deploy to: REST API, Docker, Azure, AWS, etc.

Mlflow Model Registry

Central hub for managing models.

Stages: Staging -> Production -> Archived.

Helps in CI/CD pipelines.

Example: move a model from staging to production after testing.

Example Workflow (with Random Forest)

Train Random Forest with different hyperparameters.

Use Tracking to log metrics.

Save best model with Mlflow Models.

Register it in Model Registry.

Deploy -> Monitor -> Retrain.