Machine Learning Application Development Lifecycle

Four-step process : platform solution to standardize the daily work in terms of a set of tools and languages and algorithms

1. **Data collection**
2. **Data clean**
3. **Fit and train model**: hyperparameter tuning
4. **Deploy model**: model governance, track record of hyperparameter to train the model
   Source code, performance metric

MLflow

An open source platform solution for machine learning life cycle to work with any machine learning library

1. **Integrate any ML framework**
2. **Reproducibility – same training or production code to execute with the same result regardless of whether in the cloud, local machine.**
3. **Scalability**

MLflow three components:

1. Tracking – centralized repository for metadata
2. Projects – self-contained packaging format for modeled code, training code
3. Models – a standard model format enabling any model produced by Mlflow to be deployed in any environment

MLflow Tracking

Centralized training metadata repository, MLflow to capture important metadata regardless model is trained in cloud or in-prem

1. **Hyper parameters or configuration**
2. **Log performance metrics**
3. **Log source code to produce a model**
4. **Log arbitrary files including training, test data, and models**

A working example:

- Initialize training session
- Log hyper parameters
- Log performance metrics
- Log visualization artifacts
- Persist model

MLflow Projects

Reproducible packaging format for model training sessions regardless execution context

1. **Self contained training code project specification that bundles ML training code along with version library dependencies, its configuration and test data**
2. **Simply a directory contains configuration file, code and library dependency specification, data**

A working example:

- A directory
- Run with parameters
- Automatically log during run
- Link with tracking UI

MLflow Models

A general purpose model format supporting a diverse variety of production environments: SageMaker, Kubernetes and Databricks

1. **A unified model abstraction layer to avoid one-to-one mapping problem: models developed using different ML tools to deployed to a variety ML environments**
2. **Simply a directory contains serialized model artifacts**

A working example:

- A directory
- A model bundled with two flavours: Tensorflow and Python function flavour
- A Tensorflow flavour enable model to be loaded as a native Tensorflow object
- Python function flavour introduces an addition layer of representing as a vanilla Python objects such as Pandas data frame