# MLFlow

## Getting system ready with MLflow

1. Create Virtual environment for ml flow

* Conda create -n mlflow-venv python=3.10

1. Install MLflow

* Pip install mlflow

1. Launch Mlflow user interface

* Activate the MLflow venv and run ‘*mlflow ui*’. It will by default it hosts at ‘http://127.0.0.1:5000’. Paste this to browser and one can see the MLflow UI.

## MLflow Tracking

1. mlflow.set\_tracking\_uri().

a Python function used in MLflow to specify the location where your ML experiments and models will be tracked and stored. It's essentially setting the address of the MLflow tracking server

e.g.

import mlflow

# Set the tracking URI to a local server

mlflow.set\_tracking\_uri("http://localhost:5000")

# Set the tracking URI to a remote server

mlflow.set\_tracking\_uri("https://your-mlflow-server.com")

1. mlflow.get\_tracking\_usri()

returns the current tracking uri

1. mlflow.creat\_experiment()

creates new experiment and returns it’s unique ID. We can pass this id with mlflow.start\_run() command track the progress under it.

import mlflow

# Create a new experiment named "My First Experiment"

experiment\_id = mlflow.create\_experiment("My First Experiment")

1. mlflow.set\_experiment()

activate the exiting experiment or create the new one and activate it.

1. mlflow.start\_run()

Python function used in MLflow to initiate a new ML run within a *specific experiment*. A run represents a single execution of your ML code, including the training process, model evaluation, and any other relevant steps.

import mlflow

# Start a run under a specific experiment

with mlflow.start\_run(experiment\_id=123):

# Log parameters and metrics

mlflow.log\_param("batch\_size", 32)

mlflow.log\_metric("loss", 0.25)

1. mlflow.end\_run() to end experiment run.
2. Mlflow.log\_params()

a function in MLflow used to log multiple parameters in a single line of code. Parameters are essentially the hyperparameters or configuration settings that you use to train your machine learning model. By logging these parameters, you can track how different configurations affect the model's performance.

import mlflow

with mlflow.start\_run():

params = {

"learning\_rate": 0.01,

"batch\_size": 32,

"num\_epochs": 10

}

mlflow.log\_params(params)

1. mlflow.log\_metric()

a function in MLflow used to log a metric value during an ML run. Metrics are quantitative measurements that evaluate the performance of your model, such as accuracy, loss, precision, recall, etc. By logging metrics, you can track how your model's performance evolves over time and compare different model configurations.

import mlflow

with mlflow.start\_run():

# Log a metric

mlflow.log\_metric("accuracy", 0.92)

1. mlflow.log\_artifact()

a function in MLflow used to log an artifact to the current ML run. Artifacts are files or directories that you want to associate with your ML experiment, such as model files, data files, or configuration files. By logging artifacts, you can easily track and manage the files that are essential to your ML workflow

import mlflow

with mlflow.start\_run():

# Log a model file

mlflow.log\_artifact("model.pkl")

### MLflow Workflow

***Refer the file ‘mlflow\_demo.py’ in the repo***

### Loan prediction model using MLFlow tracking

***Refer the file ‘loan\_prediction.py’ in the repo***

## MLflow Project

Creates and package the ML model along with dependencies which can be directly run on any environments

1. Create Mlproject.txt file and add bewlow content

name: Loan-Prediction

conda\_env: environment.yaml # Or specify a requirements.txt file

entry\_point:

main:

command: ’python ‘loan\_prediction.py’

1. Define environment.yaml file with all dependencies

channels:

- conda-forge

dependencies:

- python=3.10.15

- pip<=24.2

- pip:

- mlflow==2.17.2

- cloudpickle==3.1.0

- numpy==2.1.3

- pandas==2.2.3

- scikit-learn==1.5.2

- scipy==1.14.1

name: mlflow-env

1. Run ‘mlflow run . experiment-name Loan-Prediction’

## MLflow Models

To deploy the packaged model

### Fetches the Model and predict locally

1. Create .py file. In this case it is MLflow\_model.py
2. When we log the model parameters and artifacts in mlflow it by default creates the model executable folder under artifact tab in mlflow ui
3. Fetch the model and predict as described in ‘MLflow\_model.py’ file.

### Deploy model to local REST server

1. Setting tracking server with ‘set export MLFLOW\_TRACKING\_URI=http://localhost:5000’
2. Run below command to deploy selected model

mlflow models serve -m runs:/2b42b068fecc4e178a4007f90137f561/LogisticRegression --port 9000 --no-conda

1. Run below command to get response from REST API

*Invoke-WebRequest -Uri http://127.0.0.1:9000/invocations -Method Post -ContentType 'application/json' -Body '{"dataframe\_split": {"columns": ["Gender","Married","Dependents","Education","Self\_Employed","Loan\_Amount","Loan\_Amount\_term","Credit\_History","Property\_Area","TotalIncome"], "data": [[1.0, 0.0, 0.0, 0.0, 0.0, 4.98, 360.0, 1.0, 2.0, 8.60]]}}'*

## Model Registry

1. Select the model from mlflow ui
2. Register the model under artifact tab.
3. Now you can transition this model to ‘staging’ to ‘production’ to ‘archive’. These changes can be made in mlflow ui
4. Check model-serve.py file in repo to get production out of staging/production models.