**Assignment 2**

**Name:** Snehal Laxmikant Yelwande  
**Class:** TY A CSE(AI)  
**Roll No:** 63  
**Batch:** P-2  
**PRN:** 22311760

**Problem Statement**

Track experiments using **MLflow** and **Weights & Biases (W&B)**.

**Theory**

**Need of Experiment Tracking**

In machine learning, multiple experiments are run with different hyperparameters, algorithms, datasets, and preprocessing pipelines. Without proper tracking, it becomes difficult to:

* Compare results across experiments.
* Reproduce successful experiments.
* Monitor model performance and metadata.
* Share results with collaborators.

Experiment tracking tools like **MLflow** and **Weights & Biases (W&B)** solve this problem by automatically logging metrics, parameters, and artifacts.

**MLflow (Assignment 2a)**

MLflow is an open-source platform for managing the ML lifecycle.  
**Key Features:**

1. **Tracking:** Log experiments (parameters, metrics, models).
2. **Projects:** Package ML code to reproduce runs.
3. **Models:** Manage and serve models.
4. **Registry:** Version and manage models.

**Why MLflow?**

* Provides a local or server UI for visualizing experiments.
* Integrates easily with scikit-learn, TensorFlow, PyTorch.
* Supports artifact logging (datasets, models).

**Weights & Biases (Assignment 2b)**

W&B is a SaaS platform for experiment tracking, model monitoring, and collaboration.

**Key Features:**

1. **Experiment Tracking:** Logs hyperparameters, metrics, system stats.
2. **Visualization:** Interactive dashboards and plots.
3. **Collaboration:** Shareable dashboards for teams.
4. **Integration:** Supports TensorFlow, PyTorch, scikit-learn.

**Why W&B?**

* Cloud-based, accessible anywhere.
* Provides automatic GPU/CPU usage logging.
* Rich visualization features.

**Workflow**

1. Define ML experiment (train/test split, train model).
2. Integrate logging using MLflow / W&B.
3. Run experiments with different hyperparameters.
4. Visualize metrics and compare runs in respective UIs.

**Executions**

**Assignment 2a – MLflow Tracking**

**Script: assignment2a.py**

import mlflow

import mlflow.sklearn

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load dataset

iris = load\_iris()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

iris.data, iris.target, test\_size=0.2, random\_state=42

)

# Enable MLflow autologging

mlflow.set\_experiment("iris-mlflow-demo")

with mlflow.start\_run():

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

acc = accuracy\_score(y\_test, preds)

mlflow.log\_param("n\_estimators", 100)

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

mlflow.sklearn.log\_model(model, "random-forest-model")

print(f"Accuracy: {acc}")

**Execution Commands:**

python assignment2a.py

mlflow ui --port 5000

**Output:**

* Experiment logged in MLflow UI at http://127.0.0.1:5000.
* Parameters (n\_estimators) and metrics (accuracy) stored.
* Model artifact (random-forest-model) saved.

**Assignment 2b – Weights & Biases Tracking**

**Script: train\_wandb.py**

import wandb

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Initialize W&B run

wandb.init(project="iris-demo", config={"n\_estimators": 100, "test\_size": 0.2})

iris = load\_iris()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

iris.data, iris.target, test\_size=wandb.config.test\_size, random\_state=42

)

model = RandomForestClassifier(n\_estimators=wandb.config.n\_estimators, random\_state=42)

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

acc = accuracy\_score(y\_test, preds)

# Log results

wandb.log({"accuracy": acc})

print(f"Accuracy: {acc}")

**Execution Commands:**

python train\_wandb.py

**Output:**

* W&B run created and synced to dashboard: <https://wandb.ai>.
* Accuracy, hyperparameters, and system stats logged.
* Interactive charts available in W&B UI.

**Conclusion**

Experiment tracking is essential for reproducibility and performance comparison in ML workflows.

* Using **MLflow**, experiments were logged locally with parameters, metrics, and models stored in the MLflow UI.
* Using **W&B**, experiments were logged in the cloud with rich visualizations and collaborative dashboards.

Both tools complement each other: MLflow is great for local development and registry, while W&B excels in visualization and collaboration.

**Link to the GitHub Repository:** [**MLOPS\_Assignments**](https://github.com/SnehalLY/MLOPS_Assignments)