**Assignment 3**

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

**Problem Statement**

Register and manage trained models using **MLflow Model Registry** or **AWS SageMaker**.

**Theory**

**Need of Model Registry**

In a machine learning project, after training multiple models, teams need a way to:

* Track which model performed best.
* Version control models like datasets and code.
* Promote models through stages such as *Staging*, *Production*, or *Archived*.
* Rollback to older versions if the deployed model fails.

This is achieved using a **Model Registry**, which acts as a central hub to manage ML models.

**MLflow Model Registry**

MLflow Model Registry is a centralized store for models that supports:

1. **Versioning**: Each time a model is registered, a new version is created.
2. **Staging**: Models can be moved through stages (*None → Staging → Production → Archived*).
3. **Annotation**: Add descriptions or tags to models.
4. **Collaboration**: Teams can share, compare, and reuse models.

**AWS SageMaker Model Registry (Alternative)**

AWS SageMaker also provides a model registry for managing versions, approvals, and deployments. However, in this assignment, we focus on MLflow Model Registry for local implementation.

**Workflow with MLflow**

1. Train and log model with MLflow.
2. Register model into MLflow Model Registry.
3. Promote model version to *Production*.
4. Load model from registry for inference.
5. List registered models.

**Executions**

**Step 1 – Train and Register Model**

**File: train\_and\_register.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

**# Setup experiment**

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

**# Load dataset**

iris = load\_iris(as\_frame=True)

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

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

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)

**# Log parameters, metrics, and model**

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

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

mlflow.sklearn.log\_model(model, artifact\_path="iris\_rf\_model")

# Register model

registered\_model = mlflow.register\_model(

"runs:/{}/iris\_rf\_model".format(mlflow.active\_run().info.run\_id),

"IrisClassifier"

)

print(f"Model registered as IrisClassifier with accuracy: {acc:.4f}")

**Step 2 – Promote Model to Production**

**File: promote\_model.py**

from mlflow.tracking import MlflowClient

client = MlflowClient()

model\_name = "IrisClassifier"

**# Fetch latest version**

versions = client.get\_latest\_versions(model\_name, stages=None)

latest\_version = versions[0].version

print(f"Promoting version {latest\_version} to Production...")

client.transition\_model\_version\_stage(

name=model\_name,

version=latest\_version,

stage="Production"

)

print("Model promoted to Production.")

**Step 3 – Load Production Model**

**File: load\_model.py**

import mlflow

import pandas as pd

from sklearn.datasets import load\_iris

print("Trying to load Production model...")

model = mlflow.pyfunc.load\_model("models:/IrisClassifier/Production")

print("Loaded Production model.")

**# Test predictions**

iris = load\_iris(as\_frame=True)

X\_sample = iris.data.sample(5, random\_state=42)

y\_pred = model.predict(X\_sample)

print("Sample Predictions:")

print(pd.DataFrame({"Features": X\_sample.to\_dict(orient="records"), "Pred": y\_pred}))

**Step 4 – List Registered Models**

**File: list\_models.py**

from mlflow.tracking import MlflowClient

client = MlflowClient()

print("Registered Models:")

for model in client.search\_registered\_models():

print(f"Model: {model.name}")

for v in model.latest\_versions:

print(f" - version: {v.version}, stage: {v.current\_stage}")

**Execution Commands**

**# Start MLflow server**

mlflow server --backend-store-uri sqlite:///mlflow.db --default-artifact-root ./mlruns --host 127.0.0.1 --port 5000

**# Run scripts**

python train\_and\_register.py

python promote\_model.py

python load\_model.py

python list\_models.py

**Output**

* Model IrisClassifier registered in MLflow registry.
* Promoted to *Production*.
* Successfully loaded and made predictions.
* Listed model versions with their stages.

**Conclusion**

Model registries provide a reliable mechanism to manage the ML lifecycle after training. Using **MLflow Model Registry**, we successfully:

* Tracked trained models.
* Registered them with versioning.
* Promoted one model to *Production*.
* Loaded it back for predictions.

This ensures collaboration, reproducibility, and smooth CI/CD pipelines for ML projects.

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