Computer Science & Information Systems

**DevOps for Cloud - Lab Sheet 8 - Module 11**

**(M11: MLOps)**

This lab sheet needs to be administered along with Module 11: MLOps

**Notation used in the document**

* ‘>’ represents the terminal, where we type the commands.
* The text mentioned within ‘[‘ and ‘]’ brackets provides additional documentation for the step.

1. Objectives:
2. To demonstrate a simple MLOps experiment using the “MLFlow” tool
3. Pre-requisite:
4. Visual Studio IDE
5. A dataset (can select from kaggle.com. Ex: “Pima Indian Diabetes” dataset or “Titanic” dataset, for binary classification)
6. Basic knowledge of Python, and writing a ML Classifier code in Python
7. Lab Exercise:

**Task 1: Install MLFlow and import libraries**

1. Open Visual Studio IDE
2. Open the terminal, and execute the following commands.

> pip install mlflow

> pip install psutil

1. Import necessary Python and MLFlow Libraries to run the experiment

# Import necessary libraries

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, precision\_score, recall\_score, f1\_score, confusion\_matrix

import mlflow

import mlflow.sklearn

import psutil

import time

**Task 2: Write Binary Classifier code and MLFlow monitoring code**

* Create a file called “main.py” and copy the following code.
* The code is for the “Covid” dataset. [If you are using any other dataset, modify the code accordingly].

# Set the MLflow tracking URI to 'http'

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

# Function for data preprocessing

def preprocess\_data(data):

    # Convert categorical variables to one-hot encoding

    data = pd.get\_dummies(data, columns=['Gender', 'CovidSeverity'])

    # Split data into X (features) and y (target)

    X = data.drop('DischargeType', axis=1)

    y = data['DischargeType']

    return X, y

# Function for training the model

def train\_model(X\_train, y\_train, max\_depth=3, n\_estimators=100):

    # Initialize the classifier

    clf = RandomForestClassifier(max\_depth=max\_depth, n\_estimators=n\_estimators, random\_state=42)

    # Train the model

    clf.fit(X\_train, y\_train)

    return clf

# Function to evaluate the model

def evaluate\_model(model, X\_test, y\_test):

    # Make predictions

    y\_pred = model.predict(X\_test)

    # Evaluate accuracy

    accuracy = accuracy\_score(y\_test, y\_pred)

    print(f"Accuracy: {accuracy:.2f}")

    # Display classification report

    print("Classification Report:")

    print(classification\_report(y\_test, y\_pred))

# Function to log model and system metrics to MLflow

def log\_to\_mlflow(model, X\_train, X\_test, y\_train, y\_test):

    with mlflow.start\_run():

        # Log parameters

        mlflow.log\_param("max\_depth", model.max\_depth)

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

        # Log model metrics

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

        accuracy = accuracy\_score(y\_test, y\_pred)

        precision = precision\_score(y\_test, y\_pred, average='micro')

        recall = recall\_score(y\_test, y\_pred, average='micro')

        f1 = f1\_score(y\_test, y\_pred, average='micro')

        confusion = confusion\_matrix(y\_test, y\_pred)

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

        mlflow.log\_metric("precision", precision)

        mlflow.log\_metric("recall", recall)

        mlflow.log\_metric("f1-score", f1)

        # Log confusion matrix

        confusion\_dict = {

            "true\_positive": confusion[1][1],

            "false\_positive": confusion[0][1],

            "true\_negative": confusion[0][0],

            "false\_negative": confusion[1][0]

        }

        mlflow.log\_metrics(confusion\_dict)

        # Log system metrics

        # Example: CPU and Memory Usage

        cpu\_usage = psutil.cpu\_percent(interval=1)

        memory\_usage = psutil.virtual\_memory().percent

        mlflow.log\_metric("cpu\_usage", cpu\_usage)

        mlflow.log\_metric("memory\_usage", memory\_usage)

        # Log execution time

        execution\_time = {}  # Dictionary to store execution times for different stages

        # Example: Execution time for training the model

        start\_time = time.time()

        model = train\_model(X\_train, y\_train)

        end\_time = time.time()

        execution\_time["model\_training"] = end\_time - start\_time

        # Log execution time for other stages similarly

        mlflow.log\_metrics(execution\_time)

        # Evaluate model and log metrics

        evaluate\_model(model, X\_test, y\_test)

        # Log model

        mlflow.sklearn.log\_model(model, "model")

# Main function

def main():

    # Load the dataset

    data = pd.read\_csv("Covid\_data.csv")  # Update with your actual file path

    # Preprocess the data

    X, y = preprocess\_data(data)

    # Split the data into training and testing sets

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

    # Train the model

    model = train\_model(X\_train, y\_train)

    # Evaluate and log to MLflow

    log\_to\_mlflow(model, X\_train, X\_test, y\_train, y\_test)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Task 3: Run the experiment and monitor the ML Operations**

1. In terminal, run command:

> mlflow ui --host 0.0.0.0 --port 5000

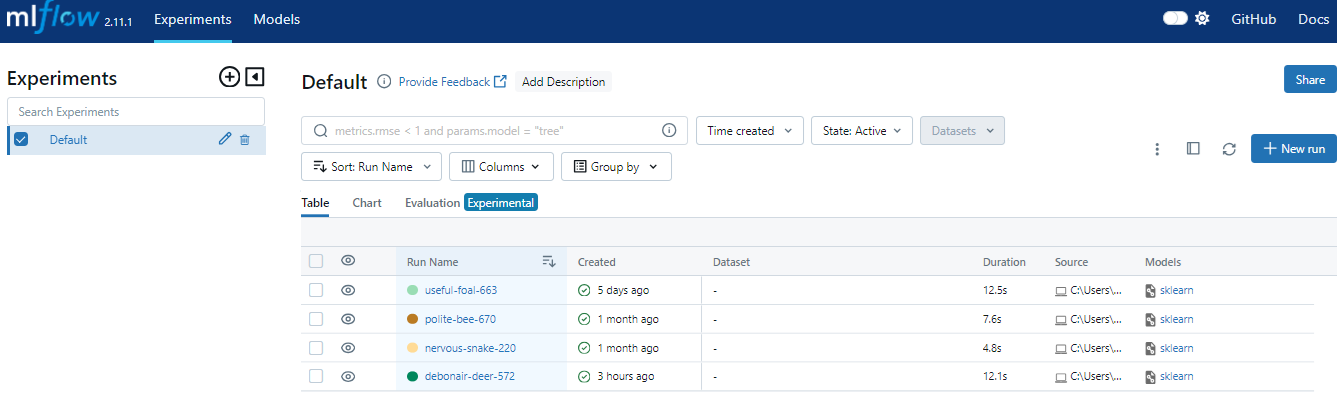
2. Right click on “main.py” and "run in interactive terminal"

3. Open localhost:5000 in browser and see the experimental results

4. In the “mlflow” dashboard, you can find the new experiment.

Open the experiment, and view the following tabs.

* Overview
* Model Metrics
* System Metrics
* Artifacts



4. Outputs/Results:

Students are expected to perform the tasks provided in the lab capsule, and thereby gain a practical understanding of the MLOps concept using the MLFlow tool.

References:

<https://mlflow.org/>