**Lesson 02 Demo 02**

**Setting up MLflow**

**Objective:** To perform the operation of setting up MLflow

**Tools Required:** Python FSD lab

**Prerequisites:** None

Steps to be followed:

1. Set up an environment
2. Access the MLflow UI

**Step 1: Set up an environment**

1. Go to the project directory using the below command:

**cd /home/nidhissimplilea/Desktop/MLOps/Demo2/**



1. Create a virtual environment using the below command:

**python -m venv mlflow-env**



**Step 2: Access the MLflow UI**

2.1 Run the below command to install seaborn:

**pip install seaborn**



2.2 Create irismodel\_demo.py using the below command:

**nano irismodel\_demo.py**



* Define model – Define steps like loading data, preprocessing, and model building.

Add the below codes to the **irismodel\_demo.py** file

1. Loading data:

- This section involves loading the dataset and extracting the feature matrix (`X`) and target vector (`y`) using the `load\_iris` function from scikit-learn.

# Load the Iris dataset

**data = load\_iris(as\_frame=True)**

**X = data.data**

**y = data.target**

**```**

2. Exploratory Data Analysis (EDA):

- In this section, summary statistics of the dataset are computed, and a pairplot is created to visualize relationships between features. EDA is essential for understanding the dataset.

# EDA: Summary statistics and visualization

**eda\_summary = X.describe()**

# Pairplot for visualizing relationships between features

**sns.set(style="ticks")**

**eda\_pairplot = sns.pairplot(data=eda\_summary, diag\_kind="kde")**

**```**

3. Data Splitting:

- The dataset is split into training and testing sets using `train\_test\_split` to prepare it for model training and evaluation.

# 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)**

**```**

4. Data Preprocessing:

- This section involves standardizing (scaling) the feature data using `StandardScaler` to ensure that each feature has a mean of 0 and a standard deviation of 1.

# Data preprocessing: Standardize features

**scaler = StandardScaler()**

**X\_train = scaler.fit\_transform(X\_train)**

**X\_test = scaler.transform(X\_test)**

**```**

5. Model Creation and Training:

- Here, a Random Forest classifier is created and trained using the standardized

training data.

# Create and train the model

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

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

**```**

6. Model Evaluation:

- Predictions are made on the test set (`X\_test`), and a classification report is generated to evaluate the model's performance.

# Make predictions on the test set

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

# Generate a classification report

**classification\_rep = classification\_report(y\_test, y\_pred, target\_names=data.target\_names, output\_dict=True)**

*```*

7. MLflow Integration:

- An MLflow run is started to log various aspects of the experiment, including EDA summary, pairplot, parameters, metrics (precision, recall, f1-score), the trained model, and the classification report.

# Start an MLflow run

**with mlflow.start\_run():**

# Log EDA summary as a text artifact

**eda\_summary\_file = "eda\_summary.txt"**

**with open(eda\_summary\_file, "w") as eda\_file:**

**eda\_file.write(eda\_summary.to\_string())**

**mlflow.log\_artifact(eda\_summary\_file)**

# Log the EDA pairplot as an image artifact

**pairplot\_file = "eda\_pairplot.png"**

**eda\_pairplot.savefig(pairplot\_file)**

**mlflow.log\_artifact(pairplot\_file)**

# Log parameters

**mlflow.log\_params({**

**"n\_estimators": 100,**

**"random\_state": 42**

**})**

# Log metrics (precision, recall, f1-score)

**for class\_name in data.target\_names:**

**metrics = classification\_rep[class\_name]**

**lflow.log\_metrics({**

**f"precision\_{class\_name}": metrics['precision'],**

**f"recall\_{class\_name}": metrics['recall'],**

**f"f1-score\_{class\_name}": metrics['f1-score']**

**})**

# Log the model

**mlflow.sklearn.log\_model(model, "random\_forest\_model")**

# Log the classification report as a text artifact

**classification\_report\_file = "classification\_report.txt"**

**with open(classification\_report\_file, "w") as text\_file:**

**text\_file.write(classification\_report(y\_test, y\_pred, target\_names=data.target\_names))**

**mlflow.log\_artifact(classification\_report\_file)**

2.3 Run the above Python file using the below command:

**python irismodel\_demo.py**



* 1. Use the following command to activate the environment:

**source mlflow-env/bin/activate**



* 1. Install mlflow using the below command:

**pip install mlflow**



* 1. View MLflow UI using the below command:

**mlflow ui**

**Note**: If the mlflow command throws port error as shown below,

run **fuser -k 5000/tcp:**

**A screenshot of a computer error

Description automatically generated**



2.7 Run the following command again:

**mlflow ui**

A screenshot of a computer

Description automatically generated

2.8 Access MLflow at 127.0.0.1:5000

A screenshot of a computer

Description automatically generated

2.9 Click on **Run Name** to access details about the experiment

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

2.10 Check **Artifacts** for EDA, evaluation metrics, and so on

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

By following these steps, you have successfully performed the operation of setting up MLflow.