ชื่อ _____ รหัสนักศึกษา _____ Section _____

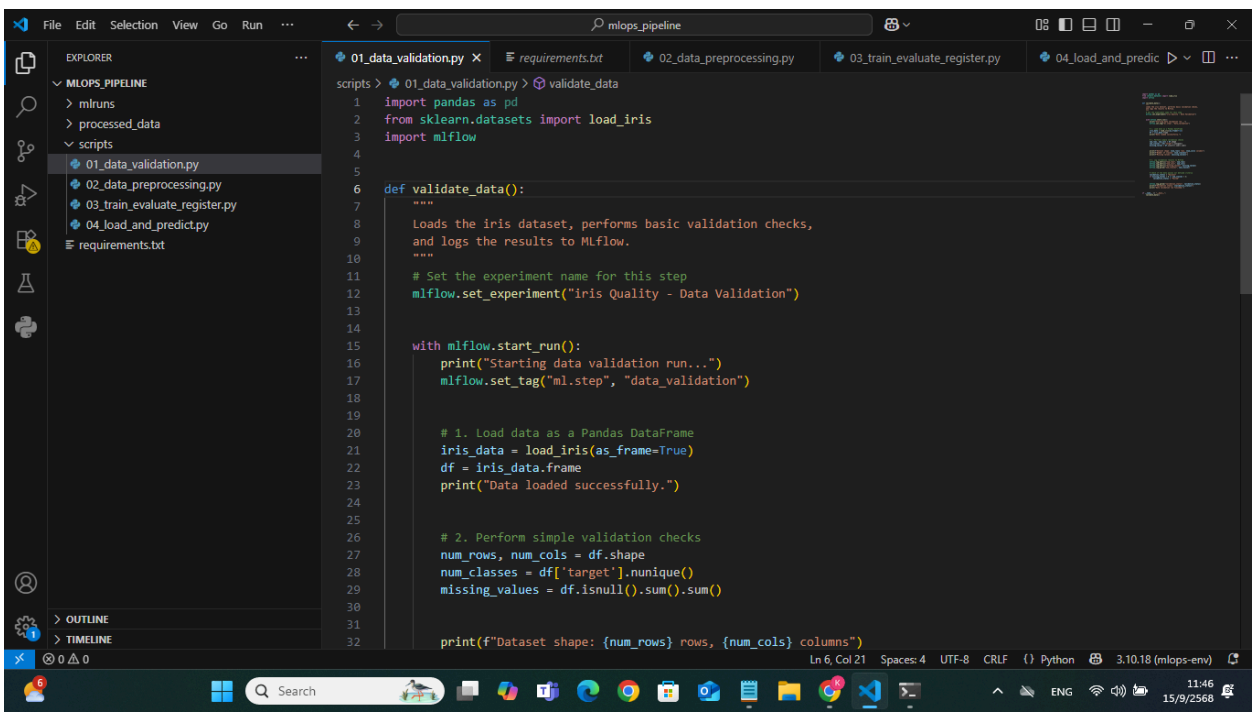
Lab 09: MLFLOW

1. ค้นหา public dataset ที่เป็นตารางข้อข้อมูล สำหรับ ทำ classification และ ใช้ MLFLOW เพื่อสร้างโมเดล และ API สำหรับ การทำ data classification

1.1 แหล่งที่มาของ Dataset (link)

https://scikit-learn.org/1.4/auto_examples/datasets/plot_iris_dataset.html

1.2 Captures โครงสร้างโฟลเดอร์ (แสดงชื่อไฟล์)



1.3 วางโค้ดทั้ง 4 ไฟล์

```
import pandas as pd
from sklearn.datasets import load_iris
import mlflow
```

```python
def validate_data():
    """
    Loads the iris dataset, performs basic validation checks,
    and logs the results to MLflow.
    """
    # Set the experiment name for this step
    mlflow.set_experiment("iris Quality - Data Validation")


    with mlflow.start_run():
        print("Starting data validation run...")
        mlflow.set_tag("ml.step", "data_validation")


        # 1. Load data as a Pandas DataFrame
        iris_data = load_iris(as_frame=True)
        df = iris_data.frame
        print("Data loaded successfully.")


        # 2. Perform simple validation checks
        num_rows, num_cols = df.shape
        num_classes = df['target'].nunique()
        missing_values = df.isnull().sum().sum()


        print(f"Dataset shape: {num_rows} rows, {num_cols} columns")
        print(f"Number of classes: {num_classes}")
        print(f"Missing values: {missing_values}")


        # 3. Log validation results to MLflow
        mlflow.log_metric("num_rows", num_rows)
        mlflow.log_metric("num_cols", num_cols)
        mlflow.log_metric("missing_values", missing_values)
        mlflow.log_param("num_classes", num_classes)


        # Check if the data passes our defined criteria
        validation_status = "Success"
```

```
        if missing_values > 0 or num_classes < 3:
            validation_status = "Failed"


        mlflow.log_param("validation_status", validation_status)
        print(f"Validation status: {validation_status}")
        print("Data validation run finished.")


if __name__ == "__main__":
    validate_data()
```

```
import os
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import mlflow


def preprocess_data(test_size=0.25, random_state=42):
    """
    Loads raw data, splits it into training and testing sets,
    and logs the resulting datasets as artifacts in MLflow.
    """
    # Set the experiment name
    mlflow.set_experiment("iris Quality - Data Preprocessing")


    with mlflow.start_run() as run:
        run_id = run.info.run_id
        print(f"Starting data preprocessing run with run_id:
{run_id}")
        mlflow.set_tag("ml.step", "data_preprocessing")


        # 1. Load data as a DataFrame
```

```
        iris_data = load_iris(as_frame=True)
        df = iris_data.frame


        # 2. Split the data into training and testing sets
        X = df.drop('target', axis=1)
        y = df['target']
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_size, random_state=random_state, stratify=y)


        # 3. Create a temporary directory to save processed data
        processed_data_dir = "processed_data"
        os.makedirs(processed_data_dir, exist_ok=True)

        # Recombine features and target for easy saving
        pd.concat([X_train, y_train],
axis=1).to_csv(os.path.join(processed_data_dir, "train.csv"),
index=False)
        pd.concat([X_test, y_test],
axis=1).to_csv(os.path.join(processed_data_dir, "test.csv"),
index=False)
        print(f"Saved processed data to '{processed_data_dir}'
directory.")


        # 4. Log parameters and metrics
        mlflow.log_param("test_size", test_size)
        mlflow.log_metric("training_set_rows", len(X_train))
        mlflow.log_metric("test_set_rows", len(X_test))


        # 5. Log the processed data directory as an artifact
        mlflow.log_artifacts(processed_data_dir,
artifact_path="processed_data")
        print("Logged processed data as artifacts in MLflow.")

        print("-" * 50)
        print(f"Data preprocessing run finished. Please use the
following Run ID for the next step:")
```

```
        print(f"Preprocessing Run ID: {run_id}")
        print("-" * 50)


if __name__ == "__main__":
    preprocess_data()
```

```
import sys
import os # เพิ่ม import นี้สำหรับจัดการ file path
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import mlflow
import mlflow.sklearn
from mlflow.artifacts import download_artifacts # เพิ่ม import นี้


def train_evaluate_register(preprocessing_run_id, C=1.0):
    """
    Loads preprocessed data, trains a model, evaluates it, and
    registers the model in the MLflow Model Registry if it meets
    the performance threshold.
    """
    ACCURACY_THRESHOLD = 0.94
    mlflow.set_experiment("iris Quality - Model Training")


    with mlflow.start_run(run_name=f"logistic_regression_C_{C}"):
        print(f"Starting training run with C={C}...")
        mlflow.set_tag("ml.step", "model_training_evaluation")
        mlflow.log_param("preprocessing_run_id",
preprocessing_run_id)
```

```
        # 1. โหลดข้อมูลจาก Artifacts ของ Preprocessing Run
        try:
            # --- START: โค้ดส่วนที่แก้ไขสำหรับ Windows ---
            # 1.1 ใช้ MLflow ดาวน์โหลด Artifacts ลงมาที่ local path
ชั่วคราว
            local_artifact_path = download_artifacts(
                run_id=preprocessing_run_id,
                artifact_path="processed_data"
            )
            print(f"Artifacts downloaded to: {local_artifact_path}")


            # 1.2 สร้างพาธไปยังไฟล์ CSV ที่ดาวน์โหลดมา
            train_path = os.path.join(local_artifact_path,
"train.csv")
            test_path = os.path.join(local_artifact_path,
"test.csv")

            # 1.3 อ่านไฟล์ CSV จาก local path ที่ถูกต้อง
            train_df = pd.read_csv(train_path)
            test_df = pd.read_csv(test_path)
            print("Successfully loaded data from downloaded
artifacts.")
            # --- END: โค้ดส่วนที่แก้ไข ---
        except Exception as e:
            print(f"Error loading artifacts: {e}")
            print("Please ensure the preprocessing_run_id is
correct.")
            sys.exit(1)


        X_train = train_df.drop('target', axis=1)
        y_train = train_df['target']
        X_test = test_df.drop('target', axis=1)
        y_test = test_df['target']

        # 2. สร้าง Scikit-learn Pipeline
        pipeline = Pipeline([
            ('scaler', StandardScaler()),
            ('model', LogisticRegression(C=C, random_state=42,
```

```
max_iter=10000))
        ])
        pipeline.fit(X_train, y_train)


        # 3. ประเมินผลโมเดล
        y_pred = pipeline.predict(X_test)
        acc = accuracy_score(y_test, y_pred)
        print(f"Accuracy: {acc:.4f}")


        # 4. Log Parameters, Metrics, และ Model (Pipeline)
        mlflow.log_param("C", C)
        mlflow.log_metric("accuracy", acc)
        mlflow.sklearn.log_model(pipeline,
"iris_classifier_pipeline")


        # 5. ตรวจสอบและลงทะเบียนโมเดล
        if acc >= ACCURACY_THRESHOLD:
            print(f"Model accuracy ({acc:.4f}) meets the threshold.
Registering model...")
            model_uri =
f"runs:/{mlflow.active_run().info.run_id}/iris_classifier_pipeline"
            registered_model = mlflow.register_model(model_uri,
"iris-classifier-prod")
            print(f"Model registered as '{registered_model.name}'
version {registered_model.version}")
        else:
            print(f"Model accuracy ({acc:.4f}) is below the
threshold. Not registering.")
        print("Training run finished.")


if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python scripts/03_train_evaluate_register.py
<preprocessing_run_id> [C_value]")
        sys.exit(1)
```

```
    run_id = sys.argv[1]
    c_value = float(sys.argv[2]) if len(sys.argv) > 2 else 1.0
    train_evaluate_register(preprocessing_run_id=run_id, C=c_value)
```

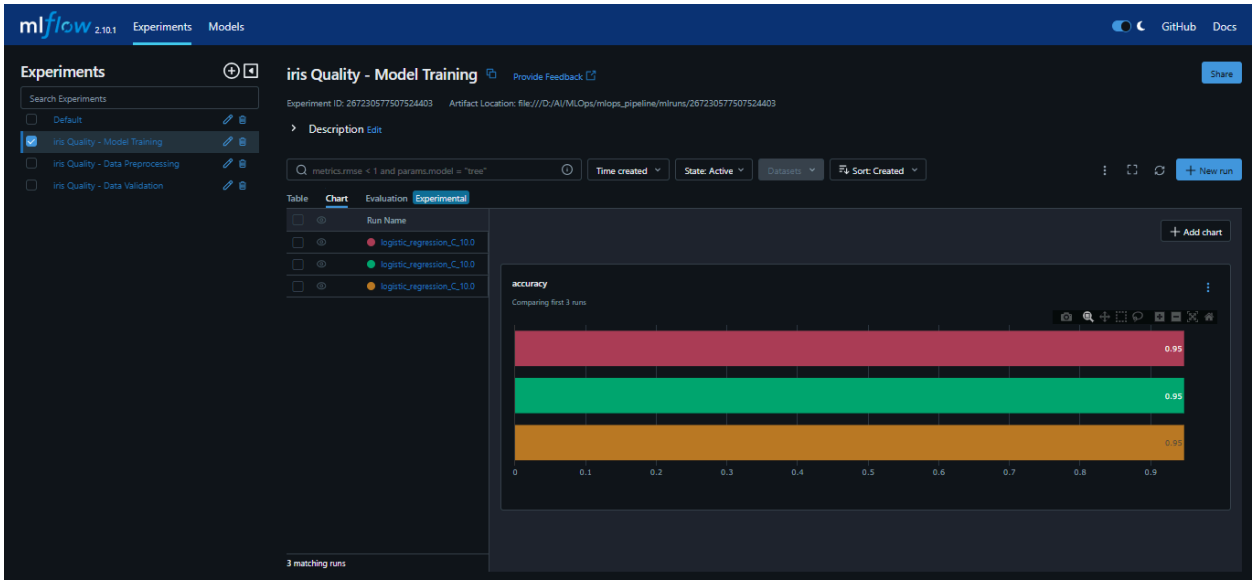
```
from sklearn.datasets import load_iris
import mlflow


def load_and_predict():
    """
    Simulates a production scenario by loading a model from a
specific
    stage in the MLflow Model Registry and using it for prediction.
    """
    MODEL_NAME = "iris-classifier-prod"
    MODEL_STAGE = "Staging" # Change to "Production" after
transitioning the model stage


    print(f"Loading model '{MODEL_NAME}' from stage
'{MODEL_STAGE}'...")

    # Load the model from the Model Registry
    try:
        model =
mlflow.pyfunc.load_model(model_uri=f"models:/{MODEL_NAME}/{MODEL_STA
GE}")
    except mlflow.exceptions.MlflowException as e:
        print(f"\nError loading model: {e}")
        print(f"Please make sure a model version is in the
'{MODEL_STAGE}' stage in the MLflow UI.")
        return


    # Prepare new sample data for prediction (using the first row of
the dataset)
```

```python
    X, y = load_iris(return_X_y=True, as_frame=False)
    sample_data = X[0:1] # Using the first row as a sample
    actual_label = y[0]


    # Use the loaded model to make a prediction
    # No manual preprocessing is needed because we logged the entire
pipeline
    prediction = model.predict(sample_data)


    print("-" * 30)
    print(f"Sample Data Features:\n{sample_data[0]}")
    print(f"Actual Label: {actual_label}")
    print(f"Predicted Label: {prediction[0]}")
    print("-" * 30)


if __name__ == "__main__":
    load_and_predict()
```

1.4 Capture ผลลัพธ์การทำนายจากคำสั่ง python scripts/04_load_and_predict.py)

```
------------------------------
Sample Data Features:
[5.1 3.5 1.4 0.2]
Actual Label: 0
Predicted Label: 0
------------------------------

(mlops-env) D:\AI\MLOps\mlops_pipeline>
```

1.5 Captures โมเดลใน MLFLOW ทุกเวอร์ชัน


mlflow 2.10.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
iris Quality - Model Training
iris Quality - Data Preprocessing
iris Quality - Data Validation
iris Quality - Model Training
Provide Feedback
Share
Experiment ID: 267230577507524403
Artifact Location: file:///D:/AI/MLOps/mlops_pipeline/mlruns/267230577507524403
Description Edit
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
Datasets
Sort: Created
New run
Table
Chart
Evaluation
Experimental
Run Name
logistic_regression_C_10.0
logistic_regression_C_10.0
logistic_regression_C_10.0
Add chart
accuracy
Comparing first 3 runs
0.95
0.95
0.95
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
3 matching runs