**Dokumentasi Pipeline Machine Learning**

Nama : Angelina Yessyca R.H

## 1. Gambaran Proyek

Nama Proyek: Pipeline Prediksi Harga Rumah

Deskripsi: Proyek ini mengimplementasikan pipeline machine learning untuk memprediksi harga rumah menggunakan data terstruktur. Pipeline mencakup langkah-langkah seperti memuat data, preprocessing, pelatihan model, evaluasi, dan pencatatan menggunakan MLflow. Aplikasi ini juga mencakup API untuk melayani model menggunakan FastAPI serta deployment melalui Docker.

## 2. Langkah-Langkah Pipeline

Pipeline diimplementasikan dalam run_pipeline.py dan terdiri dari langkah-langkah berikut:

Langkah 1: Memuat Data

- Deskripsi: Memuat dataset dari file CSV.
- Implementasi: Dataset dimuat menggunakan kelas DataLoader, yang memastikan file tersedia dan memvalidasi kolom yang diperlukan.
- Kode Utama:

```
# 1. Load Data
logger.info("Step 1: Loading data")
data_loader = DataLoader()
df = data_loader.load_data()

if not data_loader.validate_data(df):
    logger.error("Data validation failed. Check required columns.")
    raise ValueError("Data validation failed")
```

Langkah 2: Preprocessing Data

- Deskripsi: Melakukan preprocessing pada dataset dengan menangani nilai yang hilang, encoding variabel kategori, dan scaling fitur numerik.
- Implementasi: Kelas DataProcessor mengelola preprocessing menggunakan pipeline ColumnTransformer dengan StandardScaler dan OneHotEncoder.
- Kode Utama:

```
# 2. Preprocessing
logger.info("Step 2: Preprocessing data")
target_col = config.get("data").get("target_column", "SalePrice")
preprocessor = DataProcessor()
X, y = preprocessor.fit_transform(df, target_col=target_col)
```

Langkah 3: Membagi Data Train-Test

- Deskripsi: Membagi data menjadi set pelatihan dan pengujian.
- Implementasi: Menggunakan train_test_split dari scikit-learn dengan pembagian 80-20.
- Kode Utama:

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Langkah 4: Pelatihan dan Evaluasi Model

- Deskripsi: Melatih beberapa model regresi dan mengevaluasi performanya.
- Implementasi: Kelas ModelTrainer menangani pelatihan, evaluasi, dan pencatatan metrik.
- Kode Utama:

```
# 3. Model Training
logger.info("Step 3: Training and evaluating models")
trainer = ModelTrainer(experiment_name)
trainer.train_all_models(X_train, y_train, X_test, y_test)

# Log best model info
best_model = trainer.get_best_model()
```

Langkah 5: Pencatatan MLflow

- Deskripsi: Mencatat eksperimen, metrik, parameter, dan artefak menggunakan MLflow.
- Implementasi: Setiap run pelatihan dicatat sebagai run bersarang di MLflow, dan model terbaik dipilih berdasarkan RMSE.
- Kode Utama:

```
mlflow.log_param("train_size", X_train.shape[0])
mlflow.log_param("test_size", X_test.shape[0])
```

```
mlflow.log_metrics(
    {f"best_model_{k}": v for k, v in best_model["metrics"].items()}
)
```

Langkah 6: Menyimpan Preprocessor dan Model

- Deskripsi: Menyimpan pipeline preprocessing dan model untuk digunakan di kemudian hari.
- Implementasi: Artefak disimpan ke direktori lokal atau dicatat ke MLflow.
- Kode Utama:

```
preprocessor.save_preprocessors()
```

3. Deployment

**Konfigurasi Docker**

Deployment dilakukan melalui Docker dengan dua layanan utama: FastAPI dan MLflow UI.

**Dockerfile**

Digunakan untuk mengkontainerisasi aplikasi FastAPI.

```
# Gunakan base image Python
FROM python:3.9-slim

# Tetapkan working directory di dalam container
WORKDIR /app

# Salin file requirements.txt untuk menginstal dependencies
COPY requirements.txt .

# Instal dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Salin semua kode aplikasi ke dalam container
COPY . .

# Expose port 8000 (port default FastAPI)
EXPOSE 8000

# Jalankan aplikasi FastAPI menggunakan Uvicorn
CMD ["uvicorn", "src.api.main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]
```

**docker-compose.yml**

Mendefinisikan pengaturan multi-container untuk FastAPI dan MLflow.

```
version: "3.8"

Run All Services
services:
  Run Service
  fastapi:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8000:8000" # Akses FastAPI di host melalui port 8000
    volumes:
      - .:/app # Sinkronisasi kode lokal ke dalam container untuk pengembangan
    environment:
      - MLFLOW_TRACKING_URI=sqlite:///mlflow.db
    depends_on:
      - mlflow # Pastikan MLflow berjalan sebelum FastAPI
    restart: always

  Run Service
  mlflow:
    image: mlflow:2.0.1 # Gunakan image resmi MLflow
    ports:
      - "5000:5000" # Akses MLflow UI di host melalui port 5000
    volumes:
      - ./mlruns:/mlflow/artifacts # Simpan artifacts MLflow ke folder lokal
      - ./mlflow.db:/mlflow/mlflow.db # Database MLflow SQLite
    environment:
      - MLFLOW_BACKEND_STORE_URI=sqlite:///mlflow/mlflow.db
      - MLFLOW_ARTIFACT_ROOT=/mlflow/artifacts
    restart: always
```

4. Error dan Catatan

Masalah pada FastAPI

Masalah: API gagal dimulai karena model atau preprocessor tidak ditemukan.

Masalah pada Docker

Masalah: Docker Engine awalnya tidak berjalan.

Upaya Solusi: Merestart Docker Desktop dan memastikan Linux Containers diaktifkan.

5. Kesimpulan

Pipeline machine learning berhasil diimplementasikan, termasuk langkah-langkah memuat data, preprocessing, pelatihan, dan pencatatan dengan MLflow. Namun, deployment menggunakan FastAPI dan Docker menghadapi beberapa masalah yang memerlukan debugging lebih lanjut. Meski demikian, pipeline ini memberikan fondasi yang kuat untuk pengembangan di masa depan.