Assignment 6

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

Problem Statement

Deploy model with CI/CD pipeline using GitHub Actions or Jenkins

---

Theory

Need of CI/CD in ML Deployment

Deploying ML models is not just about training and saving them, but also about ensuring smooth delivery, testing, and monitoring. Continuous Integration/Continuous Deployment (CI/CD) automates the workflow of building, testing, and deploying ML services whenever code changes are pushed.

Benefits of CI/CD in MLOps:

- **Automation**: Reduces manual steps in deployment.
- **Reliability**: Ensures code and models are tested before release.
- **Scalability**: Enables quick rollouts across environments.
- **Collaboration**: Multiple developers can contribute safely.

GitHub Actions / Jenkins:

- **GitHub Actions**: Cloud-native CI/CD tool integrated directly with GitHub. Triggers workflows based on events (push, PR, etc.).
- **Jenkins**: Self-hosted automation server for CI/CD, flexible with plugins.

In this assignment, we use **GitHub Actions** to automate deployment of a Dockerized FastAPI model service.

Workflow

1. Train ML model and save (iris_model.pkl).
2. Create FastAPI application (main.py) to serve predictions.
3. Write test cases (test_main.py).
4. Containerize application using Docker (Dockerfile).
5. Push code to GitHub repository.
6. Configure GitHub Actions workflow (.github/workflows/ci-cd.yml) to:
    - Install dependencies

- Run tests with pytest
- Build Docker image
- Push image to Docker Hub (optional)

Executions

Step 1 – Train and Save Model

File: **train_model.py**

```
import joblib
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
# Load dataset
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42
)
# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Save model
joblib.dump(model, "iris_model.pkl")
print("✅ Model trained and saved as iris_model.pkl")
```

Step 2 – FastAPI Application

File: **main.py**

```
from fastapi import FastAPI
import joblib
import numpy as np
from pydantic import BaseModel
# Load model
model = joblib.load("iris_model.pkl")
class IrisInput(BaseModel):
    sepal_length: float
    sepal_width: float
    petal_length: float
    petal_width: float
app = FastAPI(title="Iris Classifier API", version="1.0")
@app.get("/")
def home():
    return {"message": "Welcome to the Iris Classifier API"}
@app.post("/predict")
def predict(data: IrisInput):
    features = np.array([[data.sepal_length, data.sepal_width,
                          data.petal_length, data.petal_width]])
    prediction = model.predict(features)[0]
    return {"prediction": int(prediction)}
```

Step 3 – Test Cases

File: **test_main.py**

```
from fastapi.testclient import TestClient
from main import app
client = TestClient(app)
def test_home():
    response = client.get("/")
    assert response.status_code == 200
    assert response.json() == {"message": "Welcome to the Iris Classifier API"}
def test_predict():
    response = client.post("/predict", json={
        "sepal_length": 6.1,
        "sepal_width": 2.8,
        "petal_length": 4.7,
        "petal_width": 1.2
    })
    assert response.status_code == 200
    assert "prediction" in response.json()
```

Step 4 – Dockerize Application

File: **Dockerfile**

```
FROM python:3.9
```

```
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY.
EXPOSE 8000
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

File: **requirements.txt**

```
fastapi
uvicorn
scikit-learn
joblib
numpy
pytest
requests
```

Step 5 – GitHub Actions Workflow

File: **.github/workflows/ci-cd.yml**

```
name: CI/CD Pipeline
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v3
    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: "3.9"
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: Run Tests
      run: pytest -q
    - name: Build Docker image
      run: docker build -t iris-api .
```

Output

- ✅ Model trained and saved.
- ✅ FastAPI app served at http://127.0.0.1:8000.
- ✅ API tested successfully (/ and /predict).
- ✅ Pytest passed in CI pipeline.
- ✅ GitHub Actions workflow executed automatically on git push.

Conclusion

In this assignment, we deployed an ML model using **FastAPI**, containerized it with **Docker**, and automated deployment with a **CI/CD pipeline in GitHub Actions**.
This ensures reproducibility, reliability, and smooth integration into production environments. The pipeline automates testing and deployment, reducing manual effort and enabling faster releases.

1. **Link to GitHub Repository:** ***MLOPS_Assignments***
