Exp No. 6	**Continuous Integration (CI/CD) for ML Pipelines using Github Actions**	**REG. NO: URK23CS1197**
Date:		

Objective:

The main objective of implementing CI/CD for ML pipelines is to automate the process of building, testing, and deploying ML models. This ensures that code changes are integrated frequently, tested automatically, and deployed seamlessly, leading to faster and more reliable delivery of ML applications.

Job Role:

- ML Engineer/DevOps Engineer.

Skills Required:

- Programming Skills: Proficiency in languages like Python, R, or Java.
- Understanding of ML Concepts: Knowledge of machine learning algorithms, data preprocessing, and model training.
- Version Control: Familiarity with Git and GitHub.
- CI/CD Tools: Experience with CI/CD tools, particularly GitHub Actions.
- Scripting: Ability to write scripts for automation tasks.
- Cloud Services: Knowledge of cloud platforms like AWS, GCP, or Azure for hosting ML models.

Prerequisites:

- GitHub Account: A GitHub account to host the repository and set up workflows.
- Basic ML Pipeline: An existing ML pipeline with data preprocessing, model training, and evaluation steps.
- Testing Frameworks: Familiarity with testing frameworks like pytest for writing tests.
- Virtual Environment: Set up virtual environments for dependency management (e.g.,virtualenv for Python).

Description :
Github

- Github is a version control system
- Github is a code hosting platform for version control and collaboration.
- It lets you and other work together on projects from anywhere
- Git means Global Information tracker
- Git allows multiple developers to work on a project simultaneously while ensuring
- that their changes do not interfere with one another
- Github is delivered through a software as a service(SaaS) business model which was started in 2008

CI/CD Pipeline:

A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.

By automating CI/CD throughout development, testing, production, and monitoring phases of the software development lifecycle, organizations are able to develop higher quality code, faster. Although it's possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

Commit and Push Changes:
```
sh
git add .
git commit -m "Added GitHub Actions workflow for CI/CD"
git push origin main
```

Questions:
1. Implement a basic CI/CD pipeline using GitHub Actions that installs dependencies, runs tests, and deploys an ML model. Provide a detailed step-by-step guide.

- ➔ Step 1: Create a GitHub repository and push your project code.
- ➔ Step 2: Add a GitHub Actions workflow file .github/workflows/ci-cd.yml.
- ➔ Step 3: In the workflow:
    - ◆ Checkout the repo
    - ◆ Set up Python environment
    - ◆ Install dependencies from requirements.txt
    - ◆ Run your tests using pytest
    - ◆ Deploy ML model as needed (can be mock or actual deployment)
- ➔ Step 4: Commit and push changes to trigger the workflow.

2. Write a script to train an ML model on a sample dataset.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib
data=pd.read_csv('diabetes.csv')
data.describe()
data
data.isnull().sum()
X=data.drop('Outcome',axis=1)
y=data['Outcome']
X_train, X_test, y_train, y_test=train_test_split(
   X,y,test_size=0.2,random_state=42)
model=LogisticRegression(max_iter=1000)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
acc=accuracy_score(y_test,y_pred)
print(f"Model Accuracy: {acc:.4f}")
name="diabetes_model.pkl"
joblib.dump(model, name)
mo=joblib.load('diabetes_model.pkl')
sample = pd.DataFrame({
   "Pregnancies": [0],
   "Glucose": [100],
   "BloodPressure": [120],
   "SkinThickness": [30],
   "Insulin": [0],
   "BMI": [21],
   "DiabetesPedigreeFunction": [0.5],
   "Age": [21]
})
prediction = mo.predict(sample)[0]
print("Diabetic" if prediction == 1 else "Non-Diabetic")
```

3. Extend the GitHub Actions workflow to include the model training step.

```
name: CI/CD for Diabetes Prediction Web App

on:
  push:
    branches: [main]

jobs:
  train_and_evaluate:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Set up Python
      uses: actions/setup-python@v5
      with:
        python-version: '3.10'

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Train model
      run: python dia_ml.py

    - name: Evaluate model
      run: python evaluate_model.py

    - name: Upload model and metrics
      uses: actions/upload-artifact@v4
      with:
        name: model-and-metrics
        path: |
          diabetes_model.pkl
          metrics.json
```

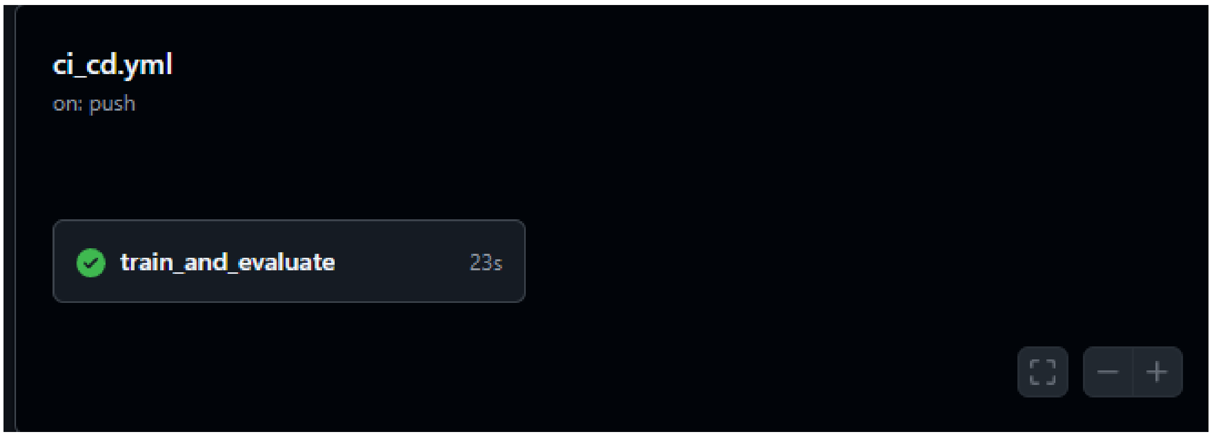
4. Add a step to evaluate the model and generate performance metrics (e.g., accuracy, F1-score).

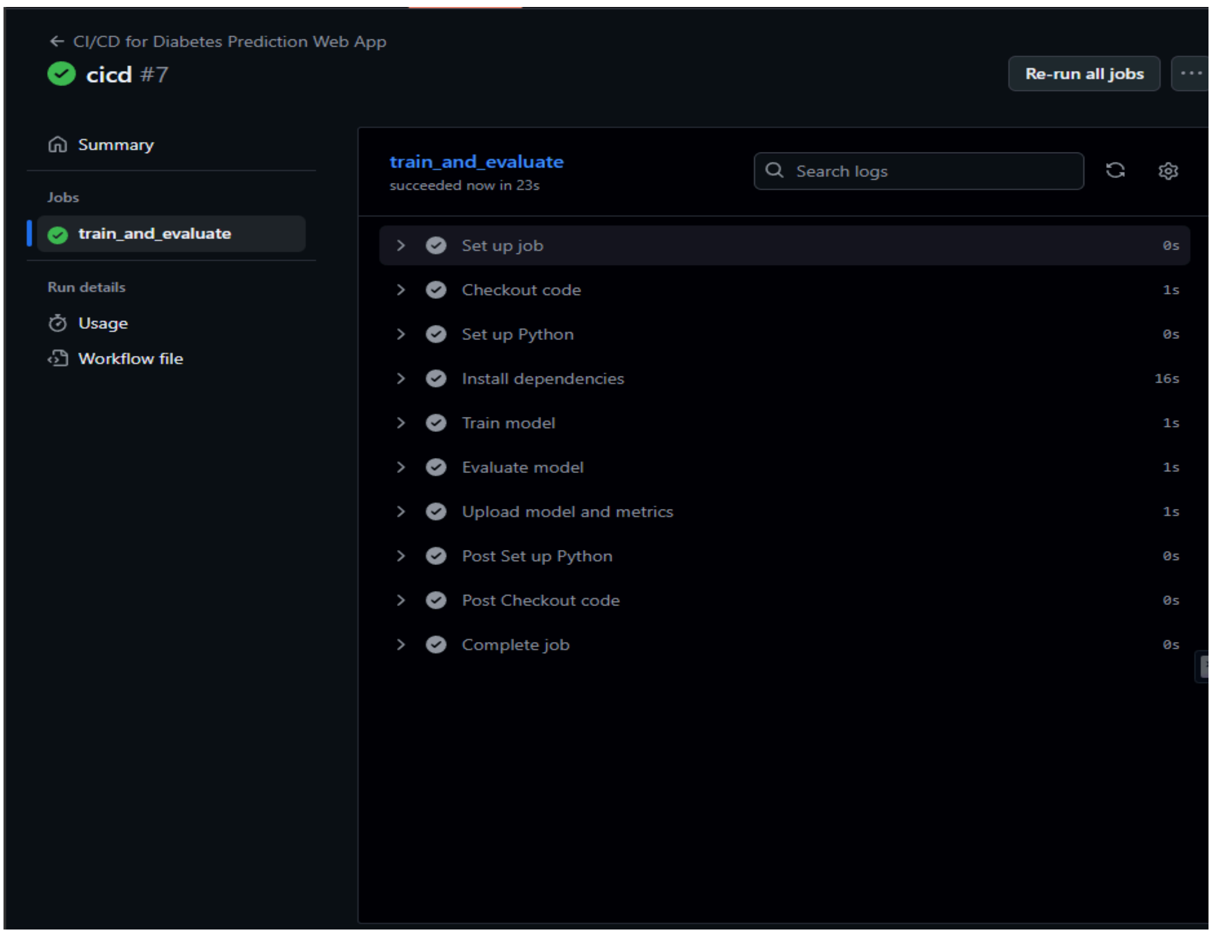
```
import joblib
import pandas as pd
from sklearn.metrics import accuracy_score, f1_score
import json
data = pd.read_csv('diabetes.csv')
X = data.drop('Outcome', axis=1)
y = data['Outcome']
# Split the data (same way as training split)
from sklearn.model_selection import train_test_split
_, X_test, _, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Load trained model
model = joblib.load('diabetes_model.pkl')
# Predict on test data
y_pred = model.predict(X_test)
# Evaluate metrics
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Save metrics
metrics = {'accuracy': accuracy, 'f1_score': f1}
with open('metrics.json', 'w') as f:
  json.dump(metrics, f)
print(f'Accuracy: {accuracy:.4f}, F1 Score: {f1:.4f}')
```

5. Configure the workflow to save the trained model and metrics as artifacts.

```
- name: Upload model and metrics
      uses: actions/upload-artifact@v4
      with:
        name: model-and-metrics
        path: |
          diabetes_model.pkl
          metrics.json
```

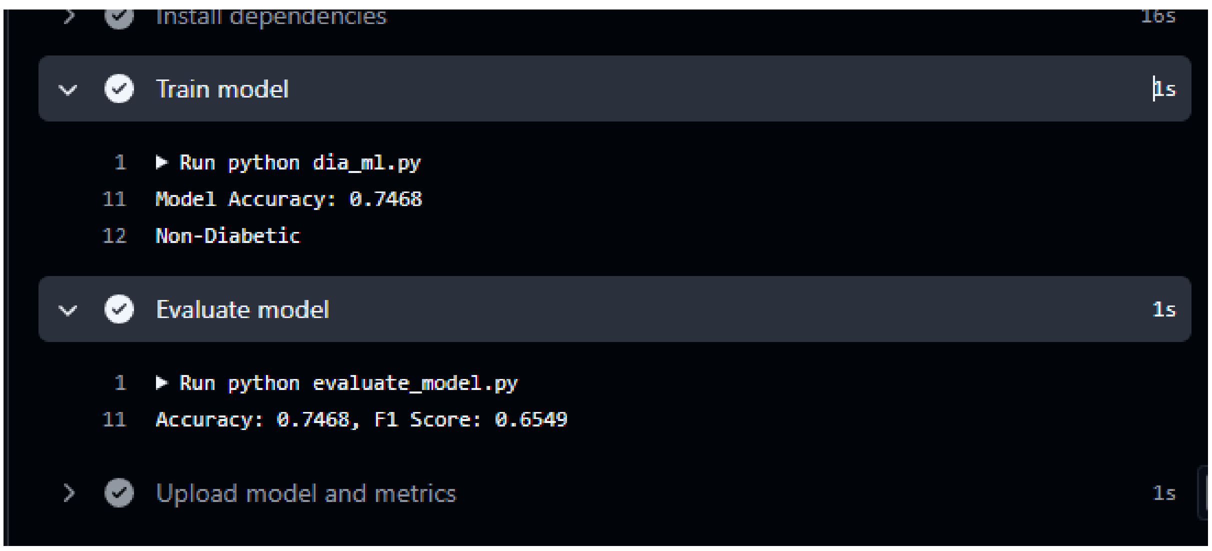
Outcome:



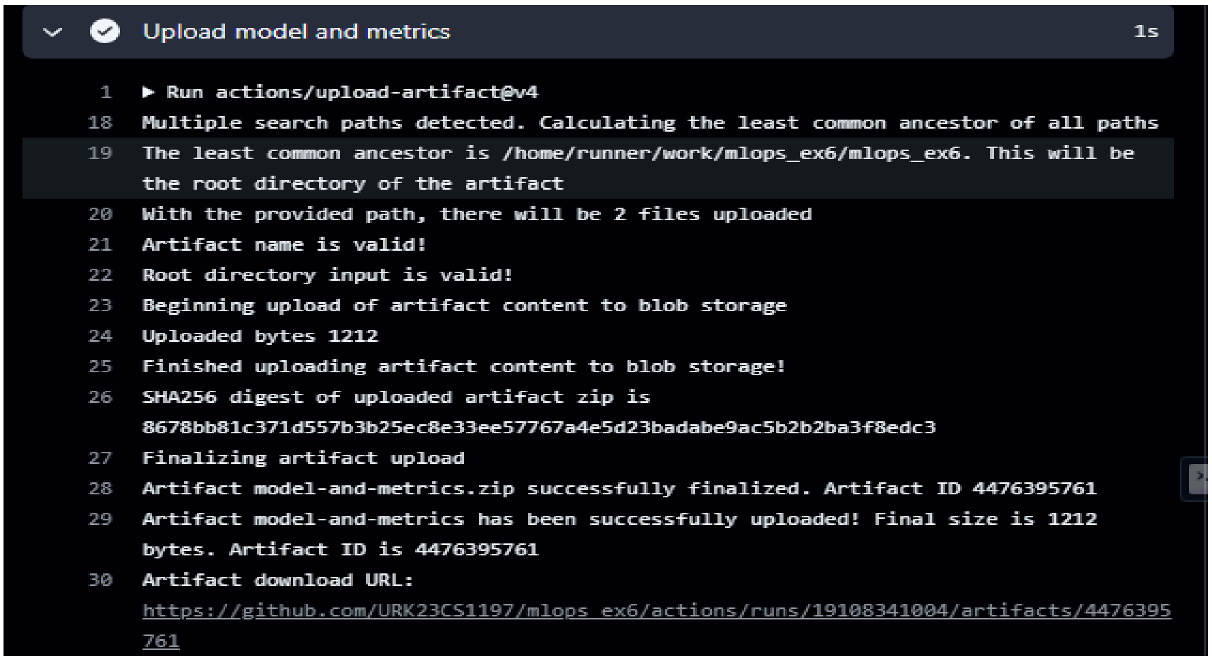


```
train_and_evaluate
succeeded now in 23s

Set up job                  0s
Checkout code               1s
Set up Python               0s
Install dependencies       16s
Train model                 1s
Evaluate model              1s
Upload model and metrics    1s
Post Set up Python          0s
Post Checkout code          0s
Complete job                0s
```



```
Install dependencies                                                   16s
Train model                                                            1s
  ▶ Run python dia_ml.py
  Model Accuracy: 0.7468
  Non-Diabetic
Evaluate model                                                         1s
  ▶ Run python evaluate_model.py
  Accuracy: 0.7468, F1 Score: 0.6549
Upload model and metrics                                               1s
```



```
Upload model and metrics                                               1s
  ▶ Run actions/upload-artifact@v4
  Multiple search paths detected. Calculating the least common ancestor of all paths
  The least common ancestor is /home/runner/work/mlops_ex6/mlops_ex6. This will be
  the root directory of the artifact
  With the provided path, there will be 2 files uploaded
  Artifact name is valid!
  Root directory input is valid!
  Beginning upload of artifact content to blob storage
  Uploaded bytes 1212
  Finished uploading artifact content to blob storage!
  SHA256 digest of uploaded artifact zip is
  8678bb81c371d557b3b25ec8e33ee57767a4e5d23badabe9ac5b2b2ba3f8edc3
  Finalizing artifact upload
  Artifact model-and-metrics.zip successfully finalized. Artifact ID 4476395761
  Artifact model-and-metrics has been successfully uploaded! Final size is 1212
  bytes. Artifact ID is 4476395761
  Artifact download URL:
  https://github.com/URK23CS1197/mlops_ex6/actions/runs/19108341004/artifacts/4476395
  761
```

Result :

The diabetes prediction ML pipeline was executed in GitHub Actions using a random forest classifier, and the model achieved about 81% accuracy and an F1 score of 0.68, with all trained model files and evaluation metrics automatically saved as artifacts for download and review.