MLOps Group No: 92

Group Members Names:

1. PEYALA SAMARASIMHA REDDY - 2023AA05072
2. PEGALLAPATI SAI MAHARSHI - 2023AA05924
3. ANIRUDDHA DILIP MADURWAR - 2023AA05982
4. K VAMSIKRISHNA - 2023AA05209

M1: MLOps Foundations

CI/CD Pipeline for Titanic Survival Prediction Model: Report

Introduction

In this project, We worked on building a **Titanic Survival Prediction Model** using a **Random Forest Classifier**. The goal was to predict whether a passenger survived the Titanic disaster based on various features such as class, age, fare, and more.

The model was trained using the Titanic dataset and the results were saved as a **Pickle (.pkl)** file, representing the trained model. To ensure smooth integration and delivery of updates, I implemented a **CI/CD pipeline** using **GitHub Actions**. This pipeline automates several stages of the machine learning lifecycle, including:

- **Linting** the code to ensure best practices and code quality.
- **Testing** to ensure the model training process runs without errors.
- **Deployment** to save the trained model and upload it for future use.

By integrating this workflow, the entire process from code push to model deployment is automated, providing a seamless way to ensure continuous integration and delivery for future changes to the project.

CI/ CD Pipeline

Objective: The CI/CD pipeline aims to automate the processes of linting, testing, and deploying a Titanic survival prediction model using the Random Forest algorithm. The pipeline ensures

that code quality is maintained, the model training works correctly, and that the model is deployed successfully.

1. Overview of the CI/CD Pipeline

The Continuous Integration and Continuous Deployment (CI/CD) pipeline is implemented using **GitHub Actions**. This pipeline automates the following key stages for the Titanic survival prediction model:

1. **Linting** - Ensures code quality and style.
2. **Testing** - Runs the model training script to verify that the model training process functions correctly.
3. **Deployment** - Saves the trained model and optionally uploads it as an artifact for future use.

2. Workflow Triggering Events

The pipeline is triggered on the following events:

- **Push to the main branch**: The pipeline automatically starts when there is a push to the `main` branch.
- **Pull Request to the main branch**: The pipeline also runs when a pull request is created targeting the `main` branch.

The configuration for this is specified under the `on` field of the workflow file:

```
on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
```

3. Pipeline Jobs

The pipeline is divided into three main jobs:

1. **Linting**
2. **Testing**
3. **Deployment**

Each job runs in sequence, ensuring that linting is completed before testing, and testing is completed before deployment.

4. Linting Job

Objective: To ensure that the code follows the defined style guide and maintains high code quality.

- **Action**: This job runs on an **Ubuntu** virtual environment.
- **Steps**:
    1. **Checkout Repository**: The first step is to fetch the latest code from the repository.
    2. **Set up Python**: The Python version 3.8 is set up in the environment.
    3. **Install dependencies**: The job installs all dependencies specified in `requirements.txt`.
    4. **Lint Python code**: The job installs **pylint** and runs it on `train.py`. The specific warning for missing module docstrings is disabled for this task: `pylint --disable=C0114 train.py`
- **Outcome**: The linting job ensures that the Python code follows best practices and adheres to the PEP8 style guide.

---

5. Testing Job

Objective: To verify that the model training works correctly and that any future tests (e.g., unit tests) are executed.

- **Action**: This job runs after the successful completion of the linting job and on the **Ubuntu** virtual environment.
- **Steps**:
    - **Checkout Repository**: Fetch the latest code from the repository.
    - **Set up Python**: Set up Python 3.8 in the environment.
    - **Install dependencies**: Install all required dependencies using `requirements.txt`.
    - **Run Model Training and Tests**: The model training script `train.py` is executed to ensure that the machine learning model is being trained without errors. This ensures that the Random Forest model is working correctly.
    - Future tests (e.g., using `pytest`) can be added here as well.
- **Outcome**: The testing job ensures that the model training script runs successfully without errors, and the model is capable of making predictions based on the Titanic dataset.

6. Deployment Job

Objective: To save the trained model and optionally upload it for later use.

- **Action**: This job runs after the successful completion of the testing job and also on an **Ubuntu** virtual environment.
- **Steps**:
    1. **Checkout Repository**: Fetch the latest code from the repository.
    2. **Set up Python**: Set up Python 3.8 in the environment.
    3. **Install dependencies**: Install all required dependencies.
    4. **Train Model and Save to Models Folder**: The `train.py` script is run again, which trains the model and saves the trained model as a `.pkl` file in the `models` folder.
    5. **Upload the Trained Model as an Artifact**: The trained model (`titanic_model.pkl`) is uploaded as an artifact using the `actions/upload-artifact@v3` GitHub Action. This allows the model to be easily retrieved and used in future steps or deployments.
- **Outcome**: The deployment job ensures that the model is saved correctly and can be retrieved later for deployment in production environments. Uploading the model as an artifact provides additional flexibility.

7. Complete CI/CD YAML Configuration

The complete configuration for the CI/CD pipeline is present in the uploaded code files.

8. Conclusion

The CI/CD pipeline automates the process of linting, testing, and deploying the Titanic survival prediction model. By using GitHub Actions, the workflow ensures that each code change is automatically tested and validated, and that the model is trained and deployed efficiently. The pipeline is flexible, allowing for future extensions such as adding more tests or integrating additional deployment stages.

A Git repository link with branches and merge history

https://github.com/samarasimhapeyala/Titanic-MLOps-Project

Please find the Branches in branches, branch history in commits, Merge history in Pull requests, CI/CD Workflows in Actions.