ML Ops

Assignment 2

Shubham Bagwari: p22cs201@iitj.ac.in **Divyaansh Mertia:** m23cse013@iitj.ac.in

Instructions for Submission and Report

- Follow the general instructions as discussed in the class: creation of conda env → requirements.txt → setting up github repo using command line. **[Using web upload will result in a penalty of 30 Marks!]**
- Submit a PDF file only [YourRollNo_A2.pdf].
- Place the GitHub link at the top of the report and make sure the link is public after the deadline.
- Comprehensive documentation is required, including step-by-step screenshots covering both command-line interactions and GitHub workflows. A detailed analysis should also be included.
- **Performance comparison report (Accuracy/F1-Score):**
  - Print loss and accuracy/F1 score.

Marks Distribution: Total 50

- **Code + Git:** 10 Marks
- **Documentation:** 10 Marks
- **Viva:** 30 Marks

Objective

The objective of this assignment is to build a practical MLOps pipeline using GitHub Actions. You will implement a digit classification pipeline using Logistic Regression and automate its testing, training, and inference via continuous integration workflows. This assignment emphasises code correctness, modularity, reproducibility, and CI/CD best practices using GitHub Actions.

Outcomes

By the end of this assignment, you will be doing the following:

- Parameterise model training with JSON config files.
- Write unit and integration tests for ML training pipelines.
- Design CI/CD pipelines using GitHub Actions.
- Pass artifacts between jobs using job dependencies.

Dataset and Model

- Dataset: `sklearn.datasets.load_digits` (built-in, no download required)
- Task: Multiclass classification (digits 0–9)
- Features: 64 grayscale pixel values (flattened 8x8 images)
- Model: `LogisticRegression` from `sklearn.linear_model`

Repository Setup

- Create a public GitHub repository named `mlops-artifact-pipeline`.
- Initialize it with a `README.md` only.
- Clone the repository to your local machine.

Project Structure

```
.
|-- src/
|   |-- train.py
|   |-- inference.py
|   |-- utils.py
|-- config/
|   |-- config.json
|-- tests/
|   |-- test_train.py
|-- .github/
|   |-- workflows/
|       |-- train.yml
|       |-- test.yml
|       |-- inference.yml
|-- requirements.txt
|-- README.md
```

Branching Instructions

To maintain a clean and traceable development history, you must follow the prescribed branching order. This ensures that the codebase from previous phases is always available when beginning the next one.

Branching Guideline

You should **not** merge your branches back into `main` after each phase. Instead, follow this linear branching approach:

1. Start with the `main` branch (contains only the README).
2. Create a new branch `classification_branch` from `main`.
3. Complete Phase 1 in `classification_branch`.
4. Checkout a new branch `test_branch` from `classification_branch`.
5. Complete Phase 2 in `test_branch`.
6. Checkout a new branch `inference_branch` from `test_branch`.
7. Complete Phase 3 in `inference_branch`.

At no point should you merge back into `main`. Each branch builds upon the previous one in a linear order.

Summary: `main` → `classification_branch` → `test_branch` → `inference_branch`

This ensures all code and workflows from earlier phases are retained without conflicts, and CI workflows remain functional across branches.

Phase 1: Training Pipeline

Branch name: `classification_branch`

Task 1.1 Implement `src/train.py`:

- Load the digits dataset.
- Read hyperparameters from `config/config.json`.
- Train a `LogisticRegression` model with config parameters.
- Save the model as `model_train.pkl` using `pickle/joblib.dump()`.

Task 1.2 Test the script locally. Ensure the model file is saved correctly.

Task 1.3 Create `.github/workflows/train.yml` to:

- Checkout the code repository.
- Set up Python.
- Install all required dependencies using `requirements.txt`.
- Run `train.py`
- Upload `model_train.pkl` as an artifact using `actions/upload-artifact`

Phase 2: Testing using Pytest

Branch name: `test_branch`

Task 2.1 Write Unit Tests for Training Pipeline. Create a test script named `test_train.py` inside a new `tests/` directory. Your tests must validate the following components of your code:

(a) **Configuration File Loading**

- Test that the configuration file (`config/config.json`) loads successfully.
- Check that all required hyperparameters exist in the configuration:
  - `C` (float): inverse regularization strength
  - `solver` (string): optimization algorithm (e.g., `"lbfgs"`)
  - `max_iter` (int): number of training iterations
- Check that the values have the correct data types.

(b) **Model Creation**

- Call your training function (e.g., `train_model(X, y, config)`).
- Verify that the function returns a `LogisticRegression` object.
- Optionally, confirm the object has been fitted (e.g., by checking attributes like `.coef_` or `.classes_`).

(c) **Model Accuracy**

- Train the model on the digits dataset and evaluate it on the same data.
- Check that the accuracy is above some threshold to verify correctness of training logic.

Task 2.2 Test the script locally.

Task 2.3 Create `.github/workflows/test.yml` to:

- Checkout the code repository.
- Set up Python.
- Install all required dependencies using `requirements.txt`.
- Run the full test suite using `pytest`.

Phase 3: Inference and Multi-Job Workflow

Branch name: `inference_branch`

Task 3.1 Implement `src/inference.py`:

- Create a new Python script named `inference.py` and place it inside the `src/` directory.
- The script should load the trained model saved as `model_train.pkl`, which was produced during the training phase.
- It must then use this model to generate predictions on the digit classification dataset used earlier (i.e., the same dataset from `sklearn.datasets.load_digits`).

Task 3.2 Test the script locally.

Task 3.3 Create a new workflow file named `inference.yml` inside the `.github/workflows/` directory. This workflow must consist of **three separate jobs**, each dependent on the previous:

(a) **Test Cases Job:**

- Runs all test cases implemented in Phase 2.
- This job must run first and the rest of the pipeline must only proceed if the test job is successful.

(b) **Train Job:**

- Executes the training script that generates the trained model.
- This job should only run after the test cases job has passed.
- The resulting model must be saved and uploaded as an artifact so that it can be reused in the next step.

(c) **Inference Job:**

- Downloads the model artifact created in the train job.
- Executes the inference script that generates predictions.
- This job must only run after the train job has successfully completed.

* For the above Phase make sure to use 'needs' parameter for the jobs requiring previous execution.

Submission Checklist

Repository is public and named `mlops-artifact-pipeline`

All tasks completed.

`model_train.pkl` is uploaded as a GitHub artifact

Final inference workflow chains test $\rightarrow$ train $\rightarrow$ inference correctly

Report as highlighted above.

Important Notes

- All hyperparameters must be read from a JSON file; no hardcoded values.
- workflows must pass in GitHub Actions. Points for that will be deducted otherwise.
- Plagiarism in any form among students will lead to zero marks.
- In the report, write your analysis [not just screenshots and code].
- Do not delete any branch.
- To prevent others from copying your work, make your Git repository public after the deadline only.