Assignment #4: MLOps Pipeline with Kubeflow, DVC, and Jenkins/GitHub Workflows

Course: Cloud MLOps (BS AI)
Total Marks: 100
Must be done **individually**

IMPORTANT WARNING: ACADEMIC INTEGRITY

This is an individual assignment. Plagiarism, which includes copying code from another, using solutions from online repositories without significant original modification, or outsourcing the work, is strictly prohibited. Any detected case of plagiarism will result in a score of ZERO for the entire assignment for both members and may lead to a final grade of 'F' in the course, in accordance with the university's academic integrity policy.

Submission Instructions:

1. Create a folder containing a PDF document as well as the project source files, do not upload cache or library binaries, only the code files in the corresponding project repository structure.
2. The document must start with the full name and student ID.
3. In the document, for each of the 5 tasks, include the specified deliverables (URL of your repository and screenshots) in order. Ensure screenshots are clear and legible.
4. Rename the folder as ROLL-NUM_SECTION (e.g., 22i-0001_A) and compress the folder as a zip file. (e.g. 22i-0001_A.zip). Do not submit a .rar file.
5. All the submissions will be done on Google Classroom within the deadline.
6. Submissions other than Google Classroom (e.g., email, etc.) will not be accepted.
7. The student is solely responsible for checking the final zip files for issues like corrupt
8. Files, viruses in the file, and mistakenly executed sent. If we cannot download the file from Google Classroom for any reason, it will lead to zero marks in the assignment.
9. Displayed output should be well-mannered and well-presented. Use appropriate
10. Comments and indentation in your source code.
11. Be prepared for a viva or anything else after the submission of the assignment.
12. Understanding the assignment is also part of the assignment.

Objective

To design, implement, and manage a complete Machine Learning Operations (MLOps) pipeline for a simple machine learning project. This assignment involves data versioning, pipeline orchestration, model training, and continuous integration using industry-standard tools like DVC, Kubeflow Pipelines, and Jenkins.

Tools to be Used:
Git, GitHub, Docker, Python, DVC, Kubeflow Pipelines (on Minikube), Kubernetes (minikube), kubectl, Scikit-learn, Jenkins/GitHub Workflows.

Tasks and Deliverables

Task 1: Project Initialization and Data Versioning (20 Marks)

Description: Set up the project foundation and implement data versioning with DVC.

Actions:

1. Create a new public GitHub repository named mlops-kubeflow-assignment.
2. Clone the repository locally and create the following structure:
    - data/ (directory for raw and processed data)
    - src/ (directory for Python scripts)
        - pipeline_components.py (Contains Kubeflow component definitions)
        - model_training.py (Training script)
    - components/ (directory for compiled Kubeflow components)
    - pipeline.py (The main Kubeflow pipeline definition)
    - requirements.txt (Project dependencies)
    - Dockerfile (For building a custom pipeline component image, if needed)
    - Jenkinsfile/GitHub Workflows (A declarative pipeline script for Jenkins/GitHub Actions)
    - .dvc/.gitignore (To ignore DVC's cache)

3. **Data and DVC Setup:**
    - Choose a simple dataset (Boston housing).
    - Initialize DVC in your repository (dvc init).
    - Set up remote storage for DVC (e.g., a separate folder, Google Drive, or AWS S3 bucket).

- Add your dataset to the data/ directory and start tracking it with DVC (dvc add data/raw_data.csv).
- Commit the resulting .dvc file and DVC meta-files to Git.

Deliverable 1:

- Screenshot of your GitHub repository file structure showing all created files and folders.
- Screenshot of the commands dvc status and dvc push executing successfully, proving the data is versioned and stored remotely.
- Content of your requirements.txt file showing essential libraries (kfp, dvc, scikit-learn, pandas, etc.).

Task 2: Building Kubeflow Pipeline Components (25 Marks)

Description: Create the core building blocks of your ML pipeline as reusable Kubeflow components.

Actions:

1. Write Python functions in src/pipeline_components.py for the following steps:
   - **Data Extraction:** A function that uses dvc get or dvc import to fetch the versioned dataset from the remote storage.
   - **Data Preprocessing:** A function that handles cleaning, scaling, and splitting the data into train/test sets.
   - **Model Training:** A function that trains a simple classifier (e.g., Random Forest) on the training data and saves the model artifact.
   - **Model Evaluation:** A function that loads the trained model, evaluates it on the test set, and saves metrics (e.g., accuracy, F1-score) to a file.
2. Use the kfp.dsl.component decorator to compile these Python functions into Kubeflow components. Specify their inputs (e.g., parameters, data path) and outputs (e.g., model artifact, metrics file).
3. Compile the components to YAML files in the components/ directory using the kfp.components.create_component_from_func method.

Deliverable 2:

- Screenshot of your src/pipeline_components.py file, showing at least two of the component functions.
- Screenshot of the components/ directory containing the generated YAML files for your components.
- Explanation of the inputs and outputs you defined for your training component.

Task 3: Orchestrating the Pipeline on Minikube (30 Marks)

Description: Assemble the components into a full pipeline and run it on Kubeflow Pipelines deployed in Minikube.

Actions:

1. **Setup:**
    - Install and start Minikube.
    - Deploy Kubeflow Pipelines (KFP) using the standalone installation or as part of the full Kubeflow deployment.
    - Ensure you can access the KFP dashboard.
2. **Pipeline Definition:**
    - In pipeline.py, use the kfp.dsl.pipeline decorator to define your pipeline.
    - Construct the pipeline by calling your components (from the YAML files or directly) and passing outputs from one component as inputs to the next.
    - Compile the pipeline into a YAML file (pipeline.yaml).
3. **Execution:**
    - Upload and run the compiled pipeline.yaml through the KFP UI.
    - Verify that the pipeline runs successfully from start to finish.

Deliverable 3:

- Screenshot of the minikube status command showing a running cluster.
- Screenshot of the Kubeflow Pipelines UI showing the graph of your successfully completed run, with all steps (data extraction, preprocessing, training, evaluation) connected.
- Screenshot of the pipeline run details, showing the outputs (e.g., the accuracy of the trained model).

Task 4: Continuous Integration with Jenkins/GitHub Workflows (15 Marks)

Description: Automate the process of testing and re-running the pipeline using Jenkins/GitHub Workflows.

Actions:

1. Populate the Jenkinsfile/GitHub Workflows with a declarative pipeline that has at least three stages:
    - **Stage 1:** Environment Setup. Checkout code, install Python dependencies from requirements.txt.

    - **Stage 2:** Pipeline Compilation. Run a script to compile your Kubeflow pipeline (pipeline.py) to ensure it is syntactically correct and generates the pipeline.yaml without errors.
2. Set up a Pipeline job in Jenkins that is linked to your GitHub repository and uses the Jenkinsfile from the main branch.
3. Trigger the Jenkins job (manually or via a webhook) and verify it passes all stages.

Deliverable 4:

- Screenshot of the Jenkins pipeline run console output showing all stages were executed successfully.
- Content of your Jenkinsfile.

Task 5: Final Integration and Documentation (10 Marks)

Description: Document the entire MLOps process to ensure reproducibility and clarity.

Actions:

1. Create a comprehensive README.md file in the root of your repository with the following sections:
    - **Project Overview:** A brief description of the project and the ML problem.
    - **Setup Instructions:** How to set up Minikube, Kubeflow Pipelines, and DVC remote storage.
    - **Pipeline Walkthrough:** Instructions on how to compile and run the Kubeflow pipeline.
2. Ensure all code is pushed to your GitHub repository.

Deliverable 5:

- URL to your final GitHub repository.
- Screenshot of the repository's main page, clearly showing the README.md file and the project structure.