MLOps Course: Data Pipeline Submission Guidelines

1 Overview

For the next phase of your project, you are required to submit a well-structured data pipeline in DAG format. The pipeline should be built using Airflow (or a similar orchestration tool) and must cover all essential stages from data acquisition to preprocessing, testing, versioning, and workflow management. Follow the detailed guidelines below:

2 Key Components to Include in Your Data Pipeline

1. **Data Acquisition**: Write code to download or fetch data from the necessary sources (APIs, databases, etc.). Ensure this step is reproducible, specifying any external dependencies in the `requirements.txt` or `environment.yml` file.

2. **Data Preprocessing**: Include clear steps for data cleaning, transformation, and feature engineering. Your preprocessing code should be modular and reusable, enabling easy adjustment.

3. **Test Modules**: Write unit tests for each component of your pipeline, especially for data preprocessing and transformation. Use testing frameworks such as `pytest` or `unittest` to ensure robustness. Test for edge cases, missing values, or possible anomalies.

4. **Pipeline Orchestration (Airflow DAGs)**: Structure your pipeline using Airflow DAGs, ensuring logical connections between tasks. Airflow should handle the entire workflow, from downloading data to generating final outputs.

5. **Data Versioning with DVC**: Use **Data Version Control (DVC)** to track and version control your data. Include the relevant `.dvc` files for datasets, and ensure that Git tracks your code and configurations for reproducibility.

6. **Tracking and Logging**: Incorporate logging into your pipeline to track progress. Use Python's `logging` library or Airflow's built-in logging. Set up monitoring for anomalies and alerts when errors are detected.

7. **Data Schema & Statistics Generation**: Automate the generation of data schema and statistics using tools such as *MLMD* or *TensorFlow Data Validation (TFDV)* or **DAG**. Ensure that schema and data quality are validated over time.

8. **Anomaly Detection & Alerts**: Set up mechanisms to detect data anomalies such as missing values, outliers, or invalid formats. Ensure your pipeline triggers an alert (e.g., email, Slack) in case of anomalies.

9. **Pipeline Flow Optimization**: Use Airflow's Gantt chart to identify bottlenecks in your pipeline. Optimize slow tasks by parallelizing or improving performance.

3 Data Bias Detection Using Data Slicing

1. **Detecting Bias in Your Data**: To ensure that your data is not biased, you need to perform data slicing and analyze performance across different subgroups. Identify demographic or categorical features in your data (such as age, gender, location, etc.) and evaluate how your model behaves for each of these subgroups.

2. **Data Slicing for Bias Analysis**: Use tools such as `SliceFinder`, `TensorFlow Model Analysis (TFMA)`, or `Fairlearn` to implement data slicing techniques. These tools can help you split your data into meaningful slices and track model performance on each slice. This helps you determine whether the model is biased toward or against certain groups.

3. **Mitigation of Bias**: If you detect any significant performance differences across slices, you must take steps to mitigate bias. Techniques such as re-sampling underrepresented groups, applying fairness constraints, or adjusting decision thresholds can be used to address this.

4. **Document Bias Mitigation Process**: Clearly document the steps you took to detect and mitigate bias. Include explanations of the types of bias found, how you addressed them, and whether any trade-offs were made in terms of overall performance.

4 Additional Guidelines

1. **Folder Structure**: Ensure your project is well-organized. Below is an example structure:

```
/Project Repo
|- Data-Pipeline/
    |- dags/
    |- data/
    |- scripts/
    |- tests/
    |- logs/
    |- dvc.yaml
    \-- README.md
```

2. **README Documentation**: Your GitHub repository must include a detailed `README.md` file, which should contain:
    - Instructions for setting up the environment.
    - Steps to run the pipeline.
    - Code structure explanation.
    - Reproducibility details and data versioning with DVC.
3. **Reproducibility**: Provide clear steps in the `README.md` file to replicate your pipeline on other machines. Anyone should be able to clone the repository, install dependencies, and run the pipeline without errors.
4. **Code Style**: Follow modular programming practices and ensure your code adheres to Python's `PEP 8` guidelines. Ensure clarity, modularity, and maintainability in your code.
5. **Error Handling & Logging**: Implement error handling for potential failure points in the pipeline (e.g., data unavailability, file corruption). Ensure logs provide sufficient information for troubleshooting.

5 Evaluation Criteria

You will be evaluated based on the following criteria:

1. **Proper Documentation**: Your GitHub repository should have a clear `README`, well-commented code, and a structured folder organization.
2. **Modular Syntax and Code**: Each component of your pipeline should be written in a modular and reusable format, allowing for easy updates and testing.
3. **Pipeline Orchestration (Airflow DAGs)**: Your data pipeline must be implemented using Airflow DAGs (or a similar orchestration tool). The flow of tasks should be logical and error handling should be properly implemented.

4. **Tracking and Logging**: Ensure proper tracking of tasks and incorporate logging throughout the pipeline. Set up error alerts and notifications for anomalies or failures.

5. **Data Version Control (DVC)**: Properly manage your data and models using DVC. Ensure that data is versioned and that the history of changes is maintained alongside your code in Git.

6. **Pipeline Flow Optimization**: Use tools such as Airflow Gantt charts to identify and resolve bottlenecks in the pipeline, optimizing task performance and overall flow.

7. **Schema and Statistics Generation**: Automatically generate data schema and statistics using tools like `Great Expectations` or `TFDV`. You will be evaluated on how well you validate the quality of your data.

8. **Anomalies Detection and Alert Generation**: Implement mechanisms to detect data anomalies and generate alerts. You should be able to handle issues such as missing values, outliers, and schema violations.

9. **Bias Detection and Mitigation**: You will be assessed on your ability to detect and mitigate bias in the dataset through data slicing. Ensure that your model performs equitably across different subgroups and that bias is addressed in your report.

10. **Test Modules**: Unit tests should be included for each key component of your pipeline, particularly for data preprocessing and transformation steps. This ensures robustness and allows testing for edge cases.

11. **Reproducibility**: Your entire pipeline should be easily reproducible on another machine. This includes instructions on setting up the environment and running the pipeline, without any errors.

12. **Error Handling and Logging**: Ensure that your pipeline has robust error handling for potential failure points. Logs should provide sufficient information to troubleshoot issues.

You will be evaluated based on how thoroughly you meet each of these criteria. Every step of the pipeline must be carefully designed, documented, and tested to ensure functionality, robustness, and reproducibility.