MLOps Course Assignment

Group Assignment

INSTRUCTOR: MANARANJAN PRADHAN

This deliverable has 50% weightage in the Consolidated Score Sheet.

General Instructions:

1. This is a **group assignment**. Please adhere to the group details posted on LMS.
2. **Do NOT submit .zip files**.
3. Please note that both **the report (pdf file) and the code files you have used (.ipynb/collab) are mandatory** for evaluation.
4. **Please ensure that before submitting the assignment(.ipynb files) all the cells are run and output is retained**.
5. Code files rendered/exported as pdfs will **strictly not be considered for evaluation**.
6. The code files should be named and submitted as mentioned in the instructions below(**Please look into submission requirement section**).
7. Make sure that **only one person** from the group makes the submission.
8. Marks shall not be awarded to students whose names are not mentioned in the Assignment Submission Form.
9. Any late submission will attract a penalty.
10. **Please adhere to the given instructions, otherwise, your submission will not be accepted, or a severe penalty will be applied**.

Assignment Objective

This assignment is designed to help students understand the full lifecycle of an ML model deployment, including version control, tracking experiments, and building an interactive UI. Each student should carefully follow the steps outlined below to develop a small but complete

MLOps project. All development should be completed in a Jupyter Notebook environment. The final product should be well-documented and follow the best coding standards as described in the guidelines section.

Assignment Requirements

1. Dataset Selection

- **Task**: Datasets to be shared with participants.

2. Dataset Schema and Storage

- **Task**: Define the dataset schema and store it in Parquet format.
- **Implementation**:
    - Define the feature types (e.g., numerical, categorical) and constraints (e.g., allowed values, nullability).

3. Profiling the Dataset

- **Task**: Generate a profile report of the dataset.
- **Tools**: Use the *pandas_profiling* or *ydata-profiling* library to create a data profile report.

4. Train-Test Split

- **Task**: Split the dataset into three parts: Training, Test and Production
    - Save the datasets as a Parquet file.

- **Requirements**:
    - Example 60-20-20 split for training, test and production.
    - Ensure reproducibility by setting a random seed.

5. Data Version Control

- **Task**:
    - Create a github repo with a datasets directory and store all datasets (train, test and prod in parquet format) and the original dataset (csv or json format)
    - Version control the dataset and related files using GitHub to read and write from the GitHub.
- **Requirements**:
    - Push the Parquet file, profiling report, and the original dataset to GitHub.
    - Use meaningful commit messages and organize the repository well.

6. ML Pipeline with Scikit-Learn

- **Task**: Read the dataset from GitHub and create an ML pipeline using scikit-learn.
- The pipeline should contain the transformation for fields like scaling, encoding, imputation etc.
- **Requirements**:

- Use GitHub raw file links to load the dataset into your notebook.
- Build a data pre-processing and model pipeline.

7. ML Experimentation and Tracking with MLflow/Weights and Biases

- Read the train and test sets from the github
- Run at least 5 ML experiments and track them using MLflow/WB.
  - These experiments can include trying different ***algorithms*** (e.g., Linear Regression, Decision Trees), modifying ***hyperparameters*** (e.g., max depth, criteria), or experimenting with different data ***transformations*** (e.g., scaling, encoding). Each experiment should be logged with MLflow/WB for tracking purposes.
- Evaluate the models using k-fold cross-validation to ensure that the model generalizes well to unseen data.
- Evaluate the model against the test set
- **Requirements**:
  - Track all information about the model in the experiment tracking tool including k-fold evaluation metrics, and test evaluation metrics, hyperparameters, if any, and data transformations for each experiment.
  - Log the models to MLflow for versioning and future reference.

8. Model Deployment Using FastAPI

- **Task**: Deploy the model as an endpoint using FastAPI.
- **Requirements**:
  - Load the best model from MLflow.
  - Create a RESTful API using FastAPI to serve predictions which takes the input as a json and returns the predictions also as a json, if possible.

9. User Interface Development with Streamlit or Gradio

- **Task**: Create a UI for the ML application using Streamlit or Gradio.
- **Requirements**:
  - The UI should allow users to input data for all features and get predictions from the FastAPI endpoint.
  - Use the appropriate UI fields for creating the UI
  - It should display appropriate messages and results based on the inputs.

10. Model Monitoring

- **Task**: Conduct a data drift analysis between train and prod data.
- **Requirements**:
  - Use alibi-detect library
  - Print the test details in tabular format

Code Requirement Guidelines

Coding Standards

- Follow PEP 8 guidelines for Python code.
- Use clear, descriptive variable and function names.
- Maintain consistent indentation and spacing.

Documentation

- Include inline comments for explaining important code blocks.
- Each function should have a docstring explaining the parameters, return values, and purpose.
- The notebook should contain markdown cells that explain the process, purpose, and key steps.

Libraries to Use

- **Data Handling**: pandas, numpy
- **Profiling**: ydata-profiling
- **Modeling**: scikit-learn
- **Experiment Tracking**: mlflow/weights and biases
- **Deployment**: fastapi, uvicorn
- **UI Development**: streamlit or gradio
- **Version Control**: Use Git and GitHub for versioning.
- **Data Drift**: alibi-detect

Submission Requirements

- **GitHub Repository**: Submit the URL to your GitHub repository.
- **Notebook File**: Submit three notebook files
    - Notebook 1: Which covers step 1 to step 7
    - Notebook 2: Which contains fastapi deployment module
    - Notebook 3: Which contains the UI implementation
    - Notebook 4: ML Monitoring

  NOTE : Please name your notebooks as Group number_Notebook-XYZ
  Example : for group -1 Notebook -1 : Gr-01_Notebook-1.ipynb

- **Deployment**: Ensure the FastAPI endpoint and Streamlit UI are running before the final demonstration.
- **Final Report**: This should contain the very few descriptions of the approach, snapshots of github, model pipeline flow, experiment tracking tools with experiment details and one or two examples of the UI showing predictions and the data drift results in tabular format.
- **The report should not be more than 4 pages and should a conclusion section with lessons learnt**.

Evaluation Criteria

- **Correctness**: The extent to which all the assignment tasks are correctly implemented.
- **Coding Standards and Documentation**: Clarity of code and documentation and inline comments including coding guidelines. There will be 25% weightage on code clarity and documentation.
- **Experimentation**: Completeness of the experiment tracking and insightful metrics logging.
- **Deployment and UI**: Functionality and usability of the FastAPI endpoint and the user interface.

Note: The assignment is meant to demonstrate your understanding of the MLOps lifecycle, from dataset preparation and versioning to model experimentation, deployment, and user interaction. Make sure each step is properly executed, tracked, and well-documented.