MLOps Applied Machine Learning Engineer Take-Home Challenge

This is a take-home challenge to evaluate your knowledge in code writing ability and designing ML deployment patterns. The code challenge is entirely optional, but if you decide to proceed, the challenge will better equip us in deciding whether you're a good fit to our team. Feel free to use any open-source tools and/or resources.

IMPORTANT NOTE: We will not be discussing the results of this challenge during the technical panel, so our assessment of this challenge will purely be derived from your written deliverable.

You have **two options** to demonstrate your coding abilities:

Option 1) Write new code and push it to a publicly available git repo:

The Final Deliverable: A Git repo hosted on Github containing all component deliverables. We should be able to clone the repo and run your code locally. The repo should contain:

1) Code / scripts for **Task 1 and Task 2**
2) Diagrams / Written Explanation for **Task 2**
3) Any instructions necessary to ensure we can reproduce this result or to direct our attention to the right places

Instructions:

1. Download the flight_prizes.zip file, unzip it. You should find two datasets and a Jupyter Notebook:
    A. flight_prices_predict.csv
    B. flight_prices_training.csv
    C. modeling.ipynb
2. Here is the scenario: A data scientist has a built a model that can predict flight prices (the "price" column) using the rest of the features, and they have asked you, the machine learning engineer, to productionize said model. The *model.ipynb* file contains the model code that you'd need to productionize.
3. **Task 1 Model Deployment – You have two options to deploy your model:**
    A. Deploy the model as a batch process:
        - Your process should be two fold:
            - Training: This part of the process should be able to take in the **flight_prices_train.csv** and output the model artifact.
            - Prediction: This part of the process should be able to take in the model artifact and **flight_prices_test.csv** as input then output another CSV containing the predictions for each row in the test file.
        - Wrap the scripts above in a Docker container. I should be able to run this Docker container on my own local computer.

    AND/OR

    B. Deploy the model you built in Task 1 as an endpoint

- Training: Create a script that creates the model artifact using **flight_prices_train.csv** as the training data
- Prediction: Deploy this model artifact as a real-time inference API. The API should allow the user to submit a row containing the features and receive the flight price in return. The API should:
    - Take in a JSON of features as input (Ex: 'airline': 'AirAsia', 'stops': 'zero')
    - Return a JSON with a "prediction" field and the corresponding model output value
    - Be wrapped in a Docker container
    - Be able to run on your local machine for testing

4. **Task 2 Automation/Infrastructure Design – Provide a way to scale your deployment:**
    A. Write the automation for the deployment of this container with horizontally scalable technology. You can provide infrastructure/automation code, or you can provide high level diagrams/explanations.
    B. **NOTE:** This part doesn't need to be very complex or 100% perfect. Just submit something solid. We just want to see in a high-level how you would deploy your deployments above in a scalable way. Providing some infrastructure/deployment code would be a bonus, but not mandatory.

After completing the tasks above, send the link to your public repo to Timotius.Kartawijaya@grainger.com for us to review.

Option 2) Share existing code from a public git repo (one or more) that completes the tasks above.
Send us links to a public repository (or repositories) to Timotius.Kartawijaya@grainger.com that accomplishes the tasks above in a similar manner: model deployment, containerization of a service, infrastructure design. You can send us multiple repos that show each of these tasks individually. For example: Repo A that shows the batch process, Repo B that shows Containerization.

Additional Instructions & Advice:

- This challenge is intentionally open-ended, there is no one "correct" way of finishing the challenge.
- Keep in mind it's not necessary to make everything perfect. Think of each prompt as an opportunity. We understand that all of us don't have a lot of spare time to finish challenges. This challenge should take you 2-3 hours.
- If you have any trouble, please feel free to ask us whatever you'd want. We mean this challenge to allow us glimpses both into your code & what it might be like to work with you.
- If anything about this challenge seems strange, incorrect, unreasonable, or infeasible, please let us know.

Please reach out to us if you have any questions by sending an email to: Timotius.Kartawijaya@grainger.com.