Senior MLOps Engineer Take-Home Technical Assessment

Overview

Welcome to the Senior MLOps Engineer take-home technical assessment. In this challenge, you'll focus on structuring and operationalizing an existing XGBoost churn model within a scalable MLOps framework. This test is meant to evaluate your ability to design robust, reproducible, and automatable ML systems. We are looking for thoughtfulness in architecture, a clean approach to pipeline design, and comfort working across tooling like MLflow and Dagster.

Background

At Better Collective (BC), one of the ways we generate revenue is by referring our customers to sportsbook operators. This marketing model is the main focus of our Affiliate Analytics department. To accomplish this, we display ads on our various owned and operated websites (such as Action Network and VegasInsider) that direct users to sportsbooks. These operators pay us when BC referred customers create accounts and deposit money.

For this assessment, you are joining the team responsible for productionizing churn models at scale. We've already built a first version of a model (an XGBoost classifier) to predict player churn. You are now responsible for designing and partially implementing the infrastructure around it. You will receive the following artifacts:

- A native JSON export or bin
- A sample of the feature set (X_test_sample.json) and a list of column names

The model predicts whether a player will churn based on a rich set of features including betting behavior, state, source of acquisition, and first played product.

Challenge Scope

You are free to assume missing pieces and document them. We want to see how you think and how you design.

Challenge Sections

Section 1: MLflow & Experiment Tracking (~1 hour)

- Set up an **MLflow tracking environment** using a **PostgreSQL backend** for the metadata store and **S3 (or local filesystem)** for the artifact store.
- Demonstrate the logic (code/config, but doesn't have to be runnable) to:
    - Log the model and its metadata (input schema, parameters, metrics)
    - Register the model under a versioned name
    - Use model signatures or schema validation for reproducibility

💡 *Bonus*: Show how you would structure the MLflow project to support multiple model variants and datasets (e.g., different geographies).

Section 2: Orchestration with Dagster (~2 hours)

- Propose a **Dagster job (graph/pipeline)** that orchestrates the following:
    - Fetch raw input data (simulate it or use the sample provided)
    - Apply feature transformations (if any)
    - Load a trained model artifact from MLflow
    - Generate predictions and store them in a target system (e.g., database or flat file)
- Bonus: Outline how you would implement:
    - A schedule to run the pipeline weekly
    - A sensor to detect new data availability
    - Retry/failure logic for any critical steps

You can implement the core of the pipeline in a Dagster job, or explain via a README.md and placeholder code.

Section 3: Deployment, Versioning & Monitoring (~2–3 hours)

Please outline your approach for the following:

Model Serving

- Would you serve this model in real-time or batch? Justify your decision.
- If real-time: how would you containerize and expose the model (e.g., FastAPI + Docker + ECS)?
- If batch: how would you manage versioned batch scoring jobs (e.g., with a feature store or metadata tagging)?

CI/CD + IaC

- How would you structure CI/CD pipelines for this system?
- What kind of GitHub Actions / GitLab CI steps would you include?
- Provide a terraform/ folder structure or snippet showing how you would provision:
    - MLflow backend (PostgreSQL + S3)
    - Compute for orchestration (e.g., ECS/Fargate or EKS)

Observability & Retraining

- How would you monitor for data drift or model degradation?
- Where would you log metrics, alerts, and errors (e.g., Prometheus + Grafana)?
- What criteria would you use to trigger a retraining pipeline?

Submission

Please submit a Git repository that includes:

- A proposed folder structure
- Dagster pipeline code or sketches
- MLflow configuration + model logging code
- Infrastructure provisioning files (or mock Terraform files)
- README.md documenting:
    - Architecture diagram (draw.io or markdown)
    - Decisions and trade-offs
    - Assumptions made
    - Suggested improvements if more time was available

We look forward to reviewing your submission. Good luck!