Model Monitoring Pipeline and Drift Detection

In order to maintain and track performance and reliability of models in production we have designed a monitoring pipeline using Airflow which periodically validates key metrics against baselines. In instances that performance crosses baseline trigger notification via slack requiring human intervention. The document also indicates key metrics and indicators we want to monitor to identify model drift.

Proposed Model Monitoring Pipeline

A comprehensive model monitoring pipeline integrates several key components, orchestrated by **Apache Airflow** for scheduling and management:

1. **Automated Test Suite Execution:**
    - **Post-Deployment Tests:** An Airflow task triggers a comprehensive test suite immediately after each new model deployment. This suite includes integration tests, basic sanity checks, and performance comparisons against a baseline on a validation dataset. Test results are sorted by severity or failure type for quick triage and prioritized investigation.
    - **Daily Regression Tests:** A dedicated daily Airflow DAG executes a focused set of regression tests on recent production data (where ground truth is available). This proactively identifies immediate performance regressions or unexpected behaviors.
2. **Metric Calculation Engine:** Airflow orchestrates jobs to continuously compute vital performance and health indicators:
    - **Performance Metrics:** For classification, metrics like Accuracy, Precision, Recall, F1-score, and ROC AUC are calculated. For regression, RMSE, MAE, and R-squared are tracked. These are computed as ground truth becomes available.
    - **Operational Metrics:** Prediction latency, throughput, API error rates, and resource utilization (CPU, memory) are continuously tracked.
3. **Feedback Loop & Retraining Trigger:** Detected drift or performance degradation can automatically trigger further analysis notification via slack, model retraining pipelines (also orchestrated by Airflow), or human intervention for investigation and model updates.

Drift Detection Module

This dedicated module, scheduled by Airflow, continuously analyzes changes in data

distributions and model behavior. It compares current data against a defined baseline (e.g., training data or a recent stable period) to identify potential drift.

Alerting System

Integrated with Airflow, the alerting system triggers notifications when predefined thresholds are breached. Alerts are configured for Airflow task failures (e.g., data ingestion errors, test suite failures), significant deviations in performance metrics, or detection of data or concept drift. Notifications are sent via channels like email, Slack, or PagerDuty, providing context and links to relevant dashboards or logs.

Visualization with Grafana Dashboard

All calculated metrics, test results, and drift detection outputs are pushed to a time-series database (e.g., Prometheus, Elasticsearch) and visualized in a **Grafana dashboard**. This provides a real-time, comprehensive overview of model health, allowing engineers to quickly identify trends, anomalies, and potential root causes.

Tracking Model Drift

Model drift is categorized into data drift and concept drift, tracked by monitoring key indicators:

1. Data Drift (Covariate Shift): Changes in input feature distributions.
Feature Distribution Tests:
   - Numerical: Kolmogorov-Smirnov (KS) test.
   - Categorical: Chi-squared tests.
   - Feature Statistics: Monitor shifts in mean, median, standard deviation, and unique value counts.
2. Concept Drift: Changes in the relationship between input features and the target variable.
   - Performance Metric Degradation: Track sustained declines in accuracy, F1-score, or RMSE using control charts (e.g., CUSUM, EWMA).
   - Residual Analysis: Analyze changes in prediction error distributions for regression models.
   - Feature Importance Shifts: Monitor changes in feature influence using explainability methods (e.g., SHAP values).

Continuous monitoring of these indicators, orchestrated by Airflow, enables proactive management of model drift, ensuring long-term ML system effectiveness.