HTX xData Technical Test Question – Engineer

[Ignatius Tang Hong-Quan]

Essay Question: propose a model monitoring pipeline and describe how you would track model drift in 500 words.

Model Monitoring Pipeline and Model Drift Tracking

Introduction

Machine learning models deployed in production environments are susceptible to performance degradation over time due to shifts in input data distributions or evolving patterns in the relationship between features and labels. This issue, known as **model drift**, can compromise the reliability and fairness of predictive systems. To address this, I propose a scalable, end-to-end model monitoring pipeline(see Diagram 1 below) that systematically captures, analyses, and responds to drift, using a combination of logging infrastructure, statistical monitoring, and model lifecycle automation.

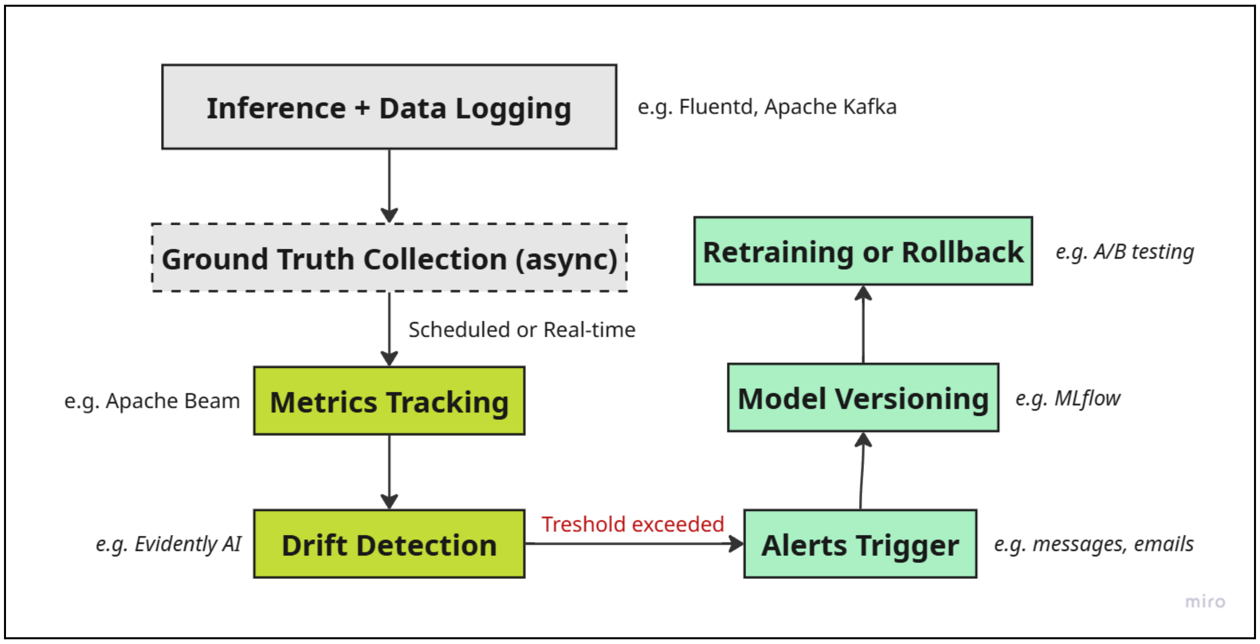

Diagram 1: Flowchart of Model Monitoring Pipeline

Methodology

The proposed monitoring pipeline is structured as a linear, modular sequence of components:

(1) Inference and Data Logging

At runtime, input data and model predictions are passed through an inference API. I integrate logging at this stage to capture inputs, outputs, timestamps, and model versioning tags.

Technologies such as *Fluentd* or *Apache Kafka* can be employed for real-time log streaming and persistence into cloud-based storage systems like *AWS* or *Google Cloud Storage*.

(2) Ground Truth Collection (Asynchronous)
Collected inference data is paired with delayed labels or real-world feedback. For applications like fraud detection or product recommendations, label collection is often delayed by hours or days. A metadata table indexed by UUID or session ID will be maintained to align predictions with true outcomes once labels are available.

(3) Metric Tracking
Performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC will be computed periodically using batched evaluation scripts or real-time stream processing tools like *Apache Beam*. These metrics will be visualised using *Prometheus* to observe longitudinal trends and trigger alarms upon degradation.

(d) Drift Detection
To track data drift, I can integrate *Evidently AI*, which computes statistical distances (e.g., Jensen-Shannon divergence or Population Stability Index) between the live input feature distribution and the original training data. *Evidently AI*'s statistical methods provide quantifiable indicators of drift and generate human-readable reports, making it ideal for early detection of subtle but impactful data shifts.

(4) Alerting
Alert thresholds will be pre-configured for both metric drops and statistical drift scores. Upon breach, alerts can be sent via *Slack integrations* or email, ensuring timely human review and response.

(5) Model Versioning
I can utilise tools like *MLflow*, *DVC* to track each deployed model's artifacts, metadata, and performance signatures. This versioning enables traceability and simplifies rollback if needed.

(6) Retraining and Rollback Pipeline
If the model drift is confirmed, automated retraining jobs can be triggered using *Kubeflow* **Pipelines**, using updated logs and verified ground truth data. Models are first evaluated via *shadow deployment* or *A/B testing* before full release. Alternatively, the system can safely revert to the last validated model version.

Conclusion
This proposal outlines a robust and modular approach to tracking and mitigating model drift in production machine learning systems. By integrating modern data engineering tools with **statistical monitoring** and **lifecycle automation**, the pipeline ensures sustained model accuracy, transparency, and resilience in dynamic data environments.