**Arize ML Observability Project – Coverage & Gaps Report**

**1. Overview**

This report provides an assessment of the current **Arize AI integration demo** implemented for ML observability.

The existing setup demonstrates an end-to-end workflow for monitoring machine learning models using synthetic data (fraud detection use case).

However, based on the latest feedback and requirements from Davin, the deliverable must now focus on traditional ML model data and cover two specific use cases related to observability, alerting, and feature store integration.

**2. Current Coverage**

The present implementation consists of three main Python modules:

* **generate\_fake\_data.py** – Generates synthetic data for a binary classification model.
* **examine\_data.py** – Produces local analytics (accuracy, fraud rate, feature statistics).
* **main.py** – Logs the data to Arize using defined schema and credentials for observability.

**Key Features Implemented**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Arize Integration** | API-based connection using Client with API key and Space ID. |
| **Schema Definition** | Defines mapping of features, labels, and prediction fields for Arize logging. |
| **Data Pipeline** | End-to-end flow: *Data Generation → Analytics → Arize Logging → Dashboard Monitoring*. |
| **Observability Components** | Demonstrates feature drift detection, performance metrics, and prediction-label alignment. |
| **Visualization Readiness** | Data successfully visualized in Arize dashboard with metric distributions and trends. |

The current implementation effectively demonstrates:

* Real-time logging of prediction data.
* Computation of local metrics before upload.
* Visualization of model accuracy and feature trends through Arize.

**3. Missing Components (As per Updated Requirements)**

According to the new instructions and feedback from Davin, the following areas need to be added or revised:

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Current Status** | **Required Update** |
| **Use Case #1 – Alerting & Dashboarding** | Not included | Demonstrate or describe how Arize can be used for real-time alerting (e.g., model drift, data quality alerts, accuracy threshold breaches). Include example screenshots or alert configurations. |
| **Use Case #2 – Full Feature Store & Predictions Upload** | Partially covered | The current pipeline already uploads features and predictions, but documentation must highlight that Arize automatically performs calculations (e.g., drift detection, performance metrics, feature correlation) when full data is sent. |
| **Traditional ML Model Data** | Focused on synthetic fraud data | Replace “fraud detection” context with generic ML dataset (e.g., customer churn, sales forecasting) to represent typical business use cases. |
| **Lessons Learned Section** | Not included | Add a closing section summarizing key lessons, best practices, challenges, and next steps for production deployment. |

**4. Recommended Document Structure (for Final Submission)**

1. **Introduction to ML Observability**
   * Explain the role of observability in ML lifecycle.
2. **Overview of the Arize Platform**
   * Key capabilities and integration benefits.
3. **Use Case #1 – Alerting and Dashboarding**
   * Data flow, alert setup, monitoring strategy, and screenshots.
4. **Use Case #2 – Full Feature Store and Predictions Upload**
   * Explanation of feature store upload, Arize calculations, and drift metrics.
5. **Lessons Learned & Recommendations**
   * Insights gained, limitations observed, and improvement suggestions.
6. **Screenshots and Visual Insights**
   * Arize dashboard visuals with annotations.

**5. Summary**

The existing Arize demo successfully demonstrates **ML observability fundamentals** such as data logging, schema mapping, and dashboard visualization.  
To align with the **new deliverable expectations**, the report and slide deck must extend beyond the fraud detection demo and explicitly address:

* Alerting workflows,
* Full feature store integrations,
* Traditional ML datasets, and

After these updates, the deliverable will comprehensively meet the **“ML Observability Features for Arize”** requirement requested by Davin.