PRJ-MLE-005: ML Model Performance Monitoring Dashboard

Certification: AWS Certified Machine Learning Engineer – Associate
Domain: Model Observability

---

1. Project Overview

This project focuses on creating a comprehensive, near real-time monitoring dashboard for a deployed machine learning model. While SageMaker Model Monitor provides baseline drift detection, this project extends that capability by building a full-fledged business intelligence (BI) dashboard in Amazon QuickSight. This dashboard will visualize not only statistical drift but also model quality metrics (like accuracy and precision) and business KPIs (like prediction latency).

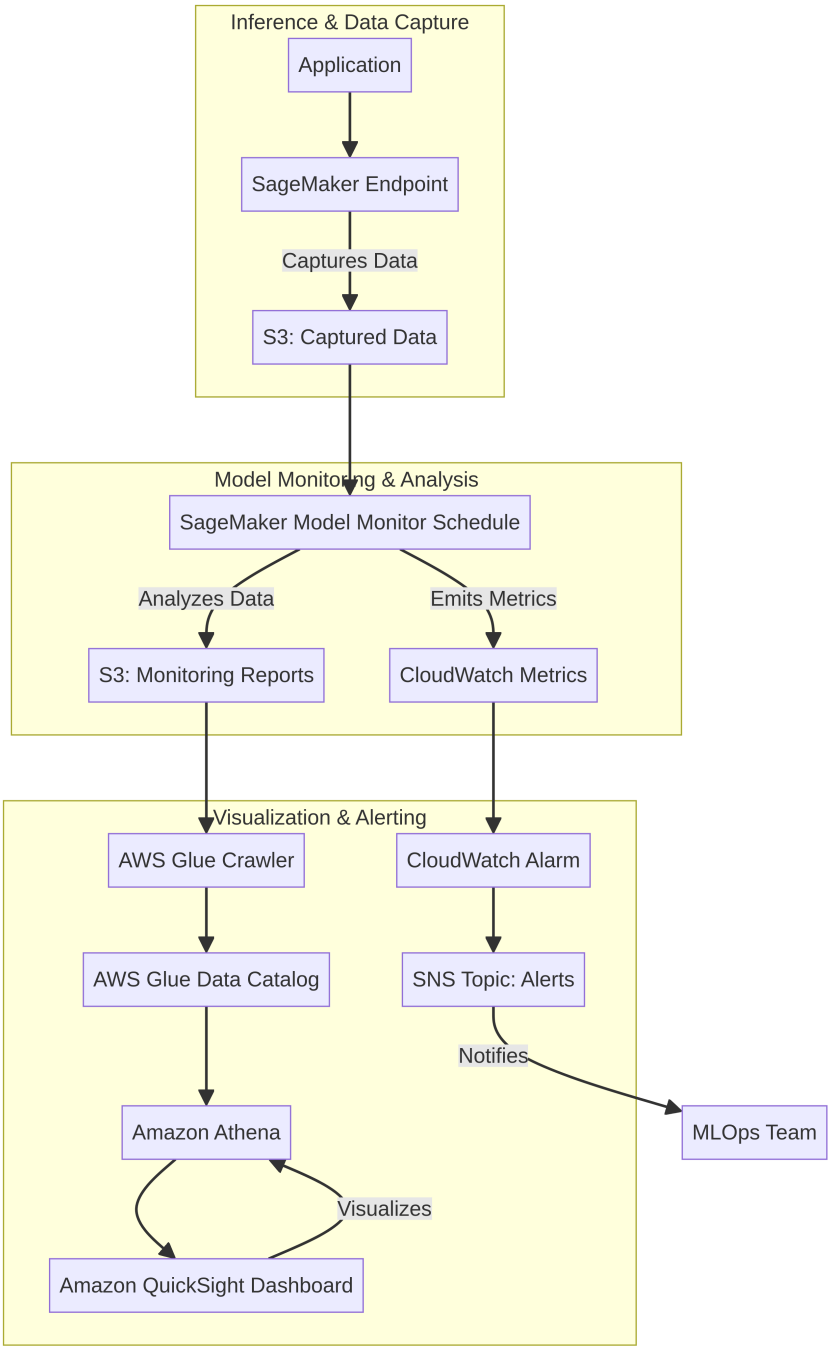
Key Objectives

- Deploy a SageMaker model with data capture enabled.
- Set up a SageMaker Model Monitoring schedule to continuously analyze production data.
- Use AWS Glue to crawl the monitoring reports and create a data catalog.
- Query the monitoring data using Amazon Athena.
- Build an interactive dashboard in Amazon QuickSight to visualize model performance, data drift, and business metrics.
- Configure a CloudWatch Alarm and SNS topic for automated alerting on severe drift.

---

2. Architecture

The architecture creates a feedback loop from production inference back to the MLOps team through a powerful visualization and alerting system.



Workflow Steps:

1. **Inference & Data Capture:** An application sends inference requests to a SageMaker endpoint. The endpoint is configured to capture both request and response data and store it in a designated S3 bucket.

2. **Model Monitoring:** A scheduled SageMaker Model Monitor job runs periodically (e.g., hourly). It analyzes the captured data against a pre-calculated baseline, checking for data drift, and also calculates quality metrics if ground truth is provided.
3. **Generate Reports:** The monitoring job outputs its findings as JSON files in S3.
4. **Data Cataloging:** An AWS Glue Crawler runs on a schedule, scanning the monitoring reports in S3 and creating/updating a table schema in the AWS Glue Data Catalog.
5. **Interactive Querying:** Amazon Athena uses the Glue Data Catalog to run standard SQL queries against the monitoring reports stored in S3.
6. **Visualization:** An Amazon QuickSight dashboard connects to Athena as a data source. It provides interactive charts and graphs to visualize feature drift, model accuracy, prediction distribution, and other key metrics over time.
7. **Alerting:** The monitoring job also emits metrics to CloudWatch. A CloudWatch Alarm is configured to watch for significant drift. If the threshold is breached, it triggers an SNS topic to send an email alert to the MLOps team.

---

3. Prerequisites

- An AWS account with an **Enterprise edition** subscription for Amazon QuickSight. You can start a free trial. Standard edition does not support Athena integration.
- Permissions for SageMaker, S3, IAM, Glue, Athena, QuickSight, CloudWatch, and SNS.
- A SageMaker Studio domain.
- An email address for SNS notifications.

---

4. Step-by-Step Deployment Guide

This guide is designed to be run within a **SageMaker Studio Notebook**.

Step 4.1: Setup and Data Preparation

We will use the California Housing dataset, a standard regression problem.

```
# Cell 1: Setup
import sagemaker
import boto3
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = sess.boto_region_name

prefix = "housing-model-monitoring"

# Cell 2: Load and Prepare Data
housing = fetch_california_housing()
data = pd.DataFrame(housing.data, columns=housing.feature_names)
data["TARGET"] = housing.target

# Split data
train, test = train_test_split(data, test_size=0.2, random_state=42)

# Save and upload to S3
train.to_csv("train.csv", index=False, header=True)
test.to_csv("test.csv", index=False, header=True)

s3_train_path = sess.upload_data("train.csv", bucket=bucket, key_prefix=f"
{prefix}/data")
s3_test_path = sess.upload_data("test.csv", bucket=bucket, key_prefix=f"
{prefix}/data")

print(f"Data uploaded to {s3_train_path}")
```

Step 4.2: Train and Deploy a Model with Data Capture

```
# Cell 3: Train XGBoost Model
from sagemaker.image_uris import retrieve

container = retrieve("xgboost", region, "1.2-1")

xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    instance_count=1,
                                    instance_type="ml.m5.large",

output_path=f"s3://{bucket}/{prefix}/output",
                                    sagemaker_session=sess)

xgb.set_hyperparameters(objective="reg:squarederror", num_round=100)

# We need to format the data for XGBoost (label in first column)
train_xgb = pd.concat([train["TARGET"], train.drop(["TARGET"], axis=1)],
axis=1)
train_xgb.to_csv("train_xgb.csv", index=False, header=False)
s3_train_xgb_path = sess.upload_data("train_xgb.csv", bucket=bucket,
key_prefix=f"{prefix}/data")

xgb.fit({"train": s3_train_xgb_path})

# Cell 4: Deploy with Data Capture
from sagemaker.model_monitor import DataCaptureConfig

endpoint_name = f"{prefix}-endpoint"
data_capture_path = f"s3://{bucket}/{prefix}/data-capture"

data_capture_config = DataCaptureConfig(
    enable_capture=True,
    sampling_percentage=100,
    destination_s3_uri=data_capture_path,
    capture_options=["REQUEST", "RESPONSE"]
)

predictor = xgb.deploy(initial_instance_count=1,
                       instance_type="ml.t2.medium",
                       endpoint_name=endpoint_name,
                       data_capture_config=data_capture_config)
```

```
print(f"Endpoint ‘{endpoint_name}’ deployed with data capture enabled.")
```

Step 4.3: Create Monitoring Schedule

```
# Cell 5: Create Baseline and Monitoring Schedule
from sagemaker.model_monitor import DefaultModelMonitor,
CronExpressionGenerator

my_monitor = DefaultModelMonitor(
    role=role,
    instance_count=1,
    instance_type="ml.m5.xlarge",
    volume_size_in_gb=20,
    max_runtime_in_seconds=3600,
)

my_monitor.suggest_baseline(
    baseline_dataset=s3_train_xgb_path,

dataset_format=sagemaker.model_monitor.dataset_format.DatasetFormat.csv(header
    output_s3_uri=f"s3://{bucket}/{prefix}/baseline",
    wait=True,
    logs=True
)

my_monitor.create_monitoring_schedule(
    monitor_schedule_name=f"{prefix}-schedule",
    endpoint_input=endpoint_name,
    output_s3_uri=f"s3://{bucket}/{prefix}/monitoring-output",
    statistics=my_monitor.latest_baselining_job.baseline_statistics(),
    constraints=my_monitor.latest_baselining_job.suggested_constraints(),
    schedule_cron_expression=CronExpressionGenerator.hourly(),
    enable_cloudwatch_metrics=True
)

print("Monitoring schedule created.")
```

Step 4.4: Generate Inference Traffic

Run this cell to generate some traffic. The monitoring job needs data to analyze.

```
# Cell 6: Generate Traffic
from sagemaker.serializers import CSVSerializer
import time

predictor.serializer = CSVSerializer()

# Get test features (without the label)
test_features = test.drop(["TARGET"], axis=1)

print("Generating inference traffic for 5 minutes...")
for i in range(300):
    sample = test_features.iloc[i].values
    predictor.predict(sample)
    time.sleep(1)

print("Traffic generation complete.")
# Note: It can take 5-10 minutes for the captured data to appear in S3.
```

Step 4.5: Setup Glue, Athena, and QuickSight

Wait for the first monitoring job to run. This happens hourly, so you may need to wait up to an hour. Once it runs, you will see output in the `monitoring-output` S3 folder.

1. **Create a Glue Crawler:**
    - Go to the **AWS Glue Console** -> **Crawlers** -> **Create crawler**.
    - **Name:** `model-monitoring-crawler`
    - **Data source:** S3. Include the path to your monitoring output folder: `s3://<your-bucket>/<prefix>/monitoring-output/`.
    - **IAM Role:** Create a new role that gives Glue permission to access S3.
    - **Database:** Create a new database named `sagemaker_monitoring_db`.
    - **Schedule:** Run on demand.
    - Finish creating the crawler, then **run it**. It will find the JSON files and create a table.
2. **Query with Athena:**

- Go to the **Amazon Athena Console**.
- **Settings:** Set a query result location in your S3 bucket.
- On the left, select `sagemaker_monitoring_db` as the database.
- You should see a table named `monitoring_output` (or similar). Run a query:

```
SELECT * FROM "monitoring_output" LIMIT 10;
```

3. **Setup QuickSight:**
   - Go to the **Amazon QuickSight Console**.
   - **Manage QuickSight -> Security & permissions:** Ensure QuickSight has access to Athena and the S3 bucket where your data is stored.
   - **Datasets -> New dataset -> Athena.**
   - **Data source name:** `sagemaker_monitoring_data`
   - Select the `sagemaker_monitoring_db` database and the `monitoring_output` table.
   - Click **Edit/Preview data** and then **Save & visualize**.
4. **Build the Dashboard:** Now you are in the QuickSight analysis view. You can create visualizations by dragging fields onto the canvas.
   - **Feature Drift:** Create a line chart. Use `report_timestamp` for the X-axis and `feature_name` and `value` for the Y-axis. Filter by a specific feature to see its drift over time.
   - **Violations Over Time:** Create a bar chart. Use `report_timestamp` for the X-axis and `num_violations` for the value.
   - **Prediction Distribution:** Create a histogram of the `prediction_value` field.
   - **Latency:** Create a line chart of `invocation_latency` over time.
   - Arrange these visuals into a dashboard and give it a name like `Model Performance Dashboard`.

Step 4.6: Configure Alerting

1. **Create SNS Topic:** Create a topic named `ModelDriftAlerts` and subscribe your email.
2. **Create CloudWatch Alarm:**
    - Go to **CloudWatch -> Alarms -> Create alarm**.
    - **Select metric:** Browse to **SageMaker -> Processing Jobs**. Find the metric `feature_baseline_drift_...` for your monitoring schedule.
    - **Metric:** Select the `value` statistic for a specific feature you want to monitor closely.
    - **Conditions:** Set a threshold (e.g., `Greater > 0.8`).
    - **Notification:** Select the `ModelDriftAlerts` SNS topic.
    - Name and create the alarm.

---

5. How to Use the Dashboard

- **Refresh Data:** In QuickSight, you can set up a schedule to refresh the dataset from Athena (e.g., hourly) to keep your dashboard up to date.
- **Analyze Drift:** Use the dashboard to visually inspect which features are drifting, how model predictions are changing, and whether performance is degrading.
- **Receive Alerts:** If a significant drift occurs that breaches your CloudWatch Alarm threshold, you will receive an email notification, prompting you to investigate.

---

6. Cleanup

1. **Delete QuickSight Dashboard and Dataset.**
2. **Delete Glue Crawler and Database.**
3. **Delete CloudWatch Alarm and SNS Topic.**
4. **Delete SageMaker Endpoint and Monitoring Schedule:**

```
# In your SageMaker notebook
my_monitor.delete_monitoring_schedule()
predictor.delete_endpoint()
```

5. **Empty and delete the S3 bucket.**

```
aws s3 rb s3://<your-bucket-name> --force
```

Business Context

The Problem

Organizations struggle to deploy ML models securely and at scale. ML workflows lack security controls, model training data is not protected, and model endpoints are vulnerable to attacks. Manual ML operations are slow and error-prone, preventing rapid iteration and deployment.

The Solution

Secure MLOps pipeline with automated model training, validation, and deployment. Implements data encryption, model versioning, endpoint security, and monitoring. Provides reproducible ML workflows with security and compliance built-in from the start.

Business Value

- **Accelerated ML Deployment:** Reduces model deployment time from weeks to days
- **Data Protection:** Encrypts training data and model artifacts at rest and in transit
- **Model Governance:** Complete audit trail of model versions, training data, and performance
- **Scalable Infrastructure:** Auto-scaling endpoints handle production traffic efficiently

Risk Mitigation

Protects sensitive training data, prevents model theft, ensures model integrity, and maintains compliance with data privacy regulations.

GRC Mapping

Compliance Frameworks

- **NIST AI RMF:** Govern, Map, Measure, Manage functions
- **ISO 27001:** A.9.4 (System access control), A.14.2 (Security in development)
- **NIST CSF:** PR.DS-1 (Data protection), ID.RA-1 (Asset vulnerabilities)
- **AWS Well-Architected ML Lens:** Security and operational excellence

Security Controls Implemented

- Data encryption at rest and in transit
- Model versioning and artifact management
- Endpoint authentication and authorization
- Model monitoring and drift detection
- Training data access controls

Audit Evidence

- Model training logs and hyperparameters
- Data lineage and provenance records
- Model performance metrics and validation results
- Endpoint access logs and API calls

Regulatory Alignment

- **GDPR:** Article 22 (Automated decision-making), Article 35 (Data protection impact assessment)
- **HIPAA:** § 164.308(a)(3) (Workforce access), § 164.312(a)(1) (Access control)

- **AI Act (EU):** High-risk AI system requirements
- **SOC 2:** CC6.1 (Logical access), CC7.2 (System monitoring)