International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ENGINEERING ROBUST AI PRODUCTS THROUGH CONTINUOUS QUALITY ASSURANCE: A FRAMEWORK FOR TESTING, MONITORING, AND VALIDATION OF ADAPTIVE LIVE LEARNING AI/ML SYSTEMS IN DYNAMIC PRODUCTION ENVIRONMENTS

Savi Grover[1], Shilpi Yadav[2], Sujeet Kumar Tiwari[3], Sooraj Ramachandran[4]

[1]Independent Researcher, Software Quality Engineer, Rahway, New Jersey, USA,
savig447@gmail.com

https://orcid.org/0009-0001-6928-1512

[2]Independent Researcher, Technical Solution Architect , Durham, North Carolina, USA,
shilpi.yadav16@gmail.com

https://orcid.org/0009-0006-7400-4356

[3]IEEE member, Software Developer Engineer in Test, Durham, North Carolina, USA,
sujeet0414@gmail.com
https://orcid.org/0009-0004-0809-2752

[4]Independent Researcher, Test Automation Architect, New York, USA,
sooraj171@hotmail.com
https://orcid.org/0009-0007-3724-6521

Abstract

The rise of online learning, continuous data ingestion, and live machine learning systems in production environments introduce new challenges for evaluating testing methodologies. Unlike traditional ML workflows, these adaptive systems continuously absorb new data and evolve over time, rendering static testing methods and offline validation datasets inadequate for ensuring reliability, fairness, and robustness. Existing quality assurance practices largely emphasize pre-deployment validation and lack systematic, automated approaches for real-time quality assurance within continuously evolving ML pipelines.

This paper proposes a framework that is implemented as a modular prototype and evaluated using data streaming scenarios, such as real-time recommendation systems and fraud detection models. The research aims to demonstrate the effectiveness of continuous verification in reducing undetected model failures, improving system reliability, and shortening feedback-to-repair cycles. By embedding QA automation into MLOPs pipelines, this study addresses a critical gap in current testing methodologies for adaptive ML systems, contributing towards scalable, audit-ready and resilient AI deployments.

Keywords: Continuous Machine Learning, Live ML models, AI-ML testing, Model Validation, ML-Quality Assurance

Introduction:

1.1 Historical Studies and Background:

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

As of early 2020, Artificial Intelligence systems came into highlight by advanced LLM automation, NLP enhancements, chatbot and recommendation systems used with every regular software as result of massive data analytics and data decisive operations. Challenges in Quality Assurance assistance with integration in ML workflow poses a great problem at every SDLC stage in ML lifecycle. In 2021, researchers from University of Calgary [1], came up with total 31 of these challenges by making a taxonomy of QA issues in MLSA- Machine Learning Software Application and dividing the ML development into six classes of SDLC phases - (1) Requirement analysis, (2) design and planning, (3) implementation, (4) testing, (5) deployment, and (6) maintenance. They classified the issues into 3 categories- Data, Practice and Standard. These challenges cover all QA related problems like vague requirements to privacy concerns along with biased and unfair data, non-deterministic output, ML deployment issues, black-boxiness of algorithms and no time to employ QA measures between continuous model training and rapid experimentation.

Another study in 2021[2], a study conducted on ML applications reported lack of a process model for ML applications, many project organizations rely on alternative models that are closely related to ML, such as, the Cross-Industry Standard Process model for Data Mining (CRISP-DM), but CRISP has mainly 2 shortcomings -first it focused more on Data mining projects and less on ML projects, second CRISP-DM lacks guidance on Quality Assurance (QA) methodology. The study addressed by deriving an end-to-end process model for the development of practical ML applications that cover all relevant phases in the life cycle of a ML application, using CRISP-DM as a basis and by anchoring QA methodology.

In 2022, a study based on industrial and electrical ML engineering showcased QA measures in MLOps by fostering data quality metrics for data integrity, trustworthiness by selecting individual data dimensions such as timeliness, uniqueness, relevancy, and location intelligence and approaches which can be taken when dealing with QA in industrial MLOps [3].

1.2 Latest advancements:

In 2024, researchers from Singapore Management University and University of Victoria, Canada [4] jointly focused on establishing quality standards for AI-ML software's. Also, there is an importance of gaining a thorough understanding of Quality Assurance for AI systems (QA4AI) from the perspective of industry practitioners. This study helped bridge the gap of lack on standards for ML practitioners, also illustrating many common challenges and their resolutions. It was a mixed method study through a combination of interviews with 15 AI practitioners, and surveys with 50 additional practitioners and defined eight QA4AI properties, namely correctness, model relevance, robustness, security, efficiency, fairness, interpretability, privacy. Another paper by Springer, in Journal of Big Data [5]- proposed a new taxonomy for AI/ML QA measure– Model Quality. The generalization measure refers to how well the model performs on new data from the target distribution, which it has not encountered during training. The main problem is overfitting–if the model overfits the training data, it leads to poor performance on new data. Balanced model fitting during training is achieved by finding the right values of hyperparameters that control model complexity, such as regularization [5].

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Many model testing strategies were combined and detailed along with selected research literature overview on data quality practices as well.

With advancements in these applications with Gen-AI, other automation related challenges are summarized by [6]. With techniques like AI-Driven Automation Frameworks, Quantum-Resistant Encryption in Testing, Predictive Analytics for Test Optimization and Voice Quality Testing for AI Assistants- there are a few rapid changes in automation tactics and e2e scripting resolution. The Future of Software Test Automation [7] expands the uses of AI in Software Testing and explains how manual test efforts are challenged by these aspects.

1.3 Problem Statement and Focus of this paper: As machine learning systems transition towards online and continual learning paradigms, where models adapt in real-time, live streaming and live data based on incoming data streams, traditional QA practices—focused on static datasets and offline model evaluations—are becoming inadequate. Static test cases fail to capture the evolving nature of the data distributions and model behaviors in production environments. Furthermore, the absence of explicit ground truth labels for real-time predictions creates a significant bottleneck in validating model outputs, exposing these systems to data drift, concept drift, accumulate bias and performance degradation over time. Currently, there is a lack of systematic QA frameworks and automation strategies that can ensure continuous validation, monitoring, and remediation of evolving ML models. Existing MLOps tools primarily emphasize deployment automation and monitoring basic metrics like latency and accuracy, without addressing the nuanced QA needs of adaptive learning systems.

This research proposes a Continuous QA Framework tailored for Online Learning systems, designed to automate validation tasks across the model lifecycle. The framework integrates real-time data validation, drift detection, dynamic test oracles, and QA-driven feedback loops to detect model degradations, ensure data quality, and trigger automated retraining interventions. Unlike conventional QA workflows, this approach addresses the lack of explicit ground truth labels in live environments by leveraging proxy metrics, statistical drift detection methods, and weak supervision techniques.

Research Objectives: This research paper emphasizes some objectives to showcase an advance QA framework for testing the live learning ML models. With **O1**: Develop Continuous QA architecture tailored for Online Learning pipelines. **O2**: Design and implement real-time QA monitoring metrics (accuracy, drift, fairness, stability). **O3**: Propose methods for dynamic test oracles that validate model outputs in absence of static labels. **O4**: Automate early warning systems and QA-driven feedback loops for live retraining interventions.

Quality Assurance Workflow Architecture

2.1 What is Continuous learning: Continuous learning, also known as continuous machine learning (CML), is a process in which a model learns from new data streams without being re-trained. Contrary to traditional approaches, where models are trained on a static dataset, deployed, and periodically re-trained, continuous learning models iteratively update their parameters to reflect new distributions in the data. In the latter process, the model improves itself by learning from the latest iteration and updating its knowledge as new data becomes

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

available. The continuous learning model life cycle enables models to remain relevant over time due to their inherently dynamic quality.

2.2 Types of continuous machine learning- There are multiple continuous machine learning approaches to modeling. Popular strategies include incremental learning, transfer learning, and lifelong learning. Other examples are experience replay methods and regularization techniques. Like with all things data, the choice of approach is not black and white. Instead, it depends on the data, model architecture, desired performance, task complexity, and computational resources available. Oftentimes, a combination of approaches is implemented to enhance learning capabilities. [8]

2.3 QA Architecture- Following is a comprehensive architecture design for a Continuous QA Framework for Online Learning and Live Input Systems, followed by detailed explanations of each architectural component and how QA strategies are implemented across the system.

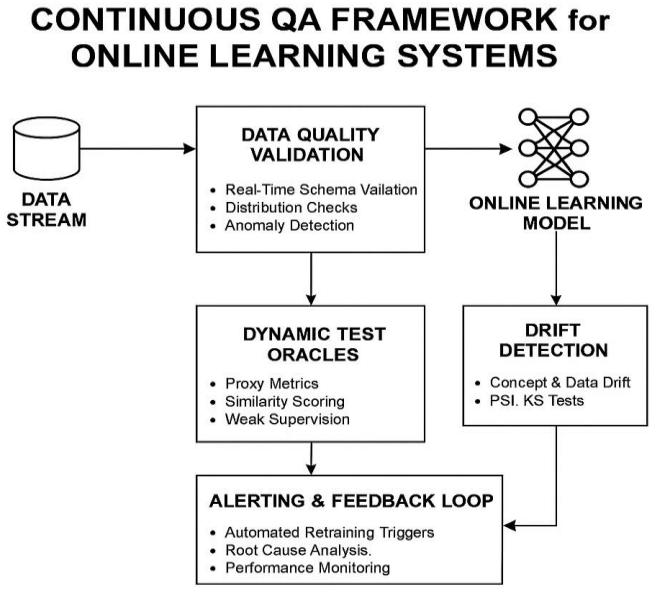

Figure 1 – shows a proposed QA workflow architecture diagram of different stages of Continuously Learning live ML System.

2.3.1. Live Data Stream Input- This is the real-time feed of inputs from multiple sources coming from sensors, users, signals, APIs, etc. Examples: Clickstreams, transaction logs, user sessions, or telemetry data.

QA Strategies: Monitor data volume, arrival rates, missing values, metadata, timestamp, source ID, data completeness. Flag unexpected bursts, schema mismatches, or out-of-bound values and API errors.

Tools: Apache Kafka / Kinesis / MQTT + Schema Registry (e.g., Confluent)

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2.3.2. Data Validation & QA Layer- Validates data integrity, schema, type, and compliance before feeding into the model.

QA Strategies: Data drift early detection, Validate schema: types, nulls, regex checks. Value distribution checks (e.g., categorical coverage, min/max ranges), QA rules stored as Data Contracts.

Tools: Great Expectations, Deequ, Pandera, Pydantic (with Kafka Hooks)

2.3.3. Feature Store & Drift Detection Engine- Transforms raw input into features. Tracks historical distributions and computers drift metrics.

QA Strategies: Compute PSI, KS-test, entropy to detect drift in features, Version feature sets, Alert if training distribution deviates significantly from production.

Tools: Feast (Feature Store), Evidently AI, River ML (streaming drift detection)

2.3.4. Online Learning Model API- Serves predictions in real-time and updates the model incrementally with new data.

QA Strategies: Validate prediction confidence, Track model convergence and stability over batches., Model reproducibility QA via logging seeds and configs.

Tools: River ML, Vowpal Wabbit, TensorFlow DataService + SavedModel APIs

2.3.5. Prediction Logger & Monitor- Logs all prediction requests/responses, confidence scores, and context data for future audit or delayed validation.

QA Strategies: Record inference latency, Capture input/output pairs, Tag prediction sessions for shadow testing or RCA.

Tools: Fluentd + ElasticSearch or Prometheus + Loki

2.3.6. QA Monitoring & Alerting- Analyzes logs and metrics in real time to detect violations of quality thresholds.

QA Strategies: Rolling window accuracy, latency, fairness metrics. Alert when drift exceeds thresholds (e.g., PSI > 0.25). Thresholds for stability and bias issues.

Tools: Grafana Dashboards, Prometheus Alerts, Evidently AI (HTTP APIs)

2.3.7. Dynamic Test Oracles Layer- Validates predictions without relying on static ground truth.

QA Strategies: Proxy labels via weak supervision, Prediction consistency for repeat inputs, Model disagreement detection (production vs shadow models), Check domain constraints (e.g., score $\in [0,1]$).

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Tools: Snorkel, SHAP, custom ensemble voting mechanisms

2.3.8. Anomaly Scoring Engine- Aggregates drift, bias, stability, and performance anomalies into a composite score.

QA Strategies: Composite Anomaly Risk Index (CARI):

$$w_1 PSI_{+\ \omega_2} ErroRate + \omega_3\ FairnessGap + w_4 PVI$$

Use thresholds to trigger retraining or human validation.

Here PSI= Population Stability Index

PVI= Pandemic or Place-Based vulnerability Index

Tools: Custom Python Service, Scikit-learn for scoring, or Evidently thresholds

2.3.9. Feedback Loop & Automation Layer- Automates retraining, shadow model rollout, and feedback collection based on anomaly triggers.

QA Strategies: Auto-trigger retraining with the latest validated data, roll out models via canary deployments, Incorporate human feedback into online learning stream.

Tools: MLflow, Kubeflow Pipelines, SageMaker Pipelines, Airflow, Argo Workflows

2.3.10. QA Dashboards & Audit Logs- Central UI for tracking QA health, incident history, retraining events, and metric visualizations.

QA Strategies: SLA dashboards (e.g., % predictions within latency targets). Visualize model health over time (accuracy, PSI, fairness). Log all QA actions for regulatory audits.

Tools: Grafana, Superset, Tableau, Streamlit (for custom dashboards)

Example End-to-End QA Flow

- New user transaction enters the stream → validated by schema and domain checks.
- Feature drift detected in calculated, example (PSI = 0.29).
- Prediction confidence drops below threshold in segment/epoch A.
- Anomaly engine flags risk → triggers retraining pipelines.
- Shadow model deployed and A/B tested.
- New model passes drift, fairness, and accurate checks → fully deployed.
- All events logged in QA Dashboard.

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Quality Methodologies on Data Layer

A. Live Data Quality Validation-

Live data for continuous model training presents several risks that necessitate careful consideration and mitigation strategies. These risks can be broadly categorized as follows:

1. **Data Drift and Concept Drift-** The statistical properties of the incoming live data can change over time (data drift), or the relationship between input features and target variables can evolve (concept drift). If the model is not continuously updated to reflect these changes, its performance can degrade significantly.
2. **Data Poisoning-** Malicious actors can inject corrupted or misleading data into the live stream, intentionally manipulating the training process and causing the model to learn biased, incorrect, or even dangerous behaviors.
3. **Data Source Testing-** specific params indicating actual sources of data are labeled so that a malicious, simulated data source does not play and authentic and real data can be used for model training and analysis predictions.
4. **Data Skew and Imbalance-** Live data streams can exhibit skewed or imbalanced distributions, where certain classes or features are over-represented or under-represented or closely associated. This can lead to biased models that perform poorly in minority classes.
5. **Data Loss or Corruption-** Network issues, hardware failures, or software bugs in the streaming pipeline can lead to incomplete or inaccurate data being processed, impacting model training.
6. **Data Schema Validation-** Metadata testing incoming data, schema validation and restricting improper data, missing data. Prebuilding data pipelines for injecting proper formats of data along with all field values.
7. **Data dimensions-** Accuracy, Completeness, Consistency, Timeliness, Validity, Uniqueness, Currency, precision, availability, relevancy, non-duplication, and usability of data is defined.
8. **Quality & quantity-** It is useful to distinguish imprecise data (random noise) from inaccurate data (systematic problem). Imprecise data from a noisy measurement process may lead to less confident models and occasionally spurious predictions if the model overfits to random measurement noise. However, noise in training data, especially random noise, is something that most machine-learning techniques can handle quite well. Here, having a lot of data helps to identify true patterns as noise gets averaged out.[9]
9. **Exploratory Data Analysis-** Exploratory data analysis is the process of exploring the data, typically with the goal of understanding certain aspects of it. This process often starts with understanding the shape of the data. In addition, it is often a good idea to explore trends and outliers. This sometimes provides a sense of precision and of typical kinds of mistakes. Sometimes, it is possible to go back to the original data or domain experts to check correctness.[9]
10. **Enforcing schemas in schema-less data-** In machine-learning contexts, data is often exchanged in schema-less formats, such as text entries in log files, tabular data in CSV files, tree-structured documents (json, XML), and key-value pairs. Such data is regularly

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

communicated through plain text files, through REST APIs, through message brokers, or stored in various (non-relational) databases. [9]

11. **Repair ill formatted data-** A schema can detect when data does not have the expected format. How to handle or repair ill-formatted data depends on the role of the data in the application. End users can be informed about the formatting problem and asked to manually repair the inputs, like the validation mechanisms in a web form. Ill-formatted data can also simply be dropped or collected in a file for later manual inspection. Finally, automated repair is possible, such as replacing missing or ill-formatted values with defaults or average values. These can also be called in-line pipeline validations- especially after transformations or aggregations, to ensure data integrity is maintained. Tools like Great Expectations and Cerberus can be used to integrate within pipeline code (Python, Java, etc.) to programmatically enforce rules and schemas.

12. **Define and Monitor Data Quality Metrics:** Establish clear definitions for data quality dimensions like accuracy, completeness, consistency, timeliness, and uniqueness, relevant to your use case. Implement automated monitoring systems to track these metrics in real-time, detecting anomalies and deviations as they occur.

13. **Implement Data Validation and Cleansing Pipelines:** Integrate automated checks within data pipelines to validate against predefined rules and schemas. Utilize techniques like data imputation (e.g., mean, median, regression) to fill missing values and data transformation to standardize and normalize data for improved analysis and model compatibility.

14. **Perform Regular Audits and Reviews:** Conduct regular audits of the data and the quality assurance procedures themselves to verify compliance with standards and identify areas for improvement.

15. **Focus on Relevant Features:** Select the most relevant and stable features to train the model, reducing the impact of changes in data distribution.

B. Model Stability and Performance Validation:

1. Catastrophic Forgetting- In continuous learning, new data can sometimes cause the model to "forget" previously learned information, particularly if the new data distribution significantly differs from past training data. This can also be the result of data drift and concept drift.

2. Code Testing- Testers knowing a little about code conditions, assertions, initializations, parameters, fixtures, constants, algorithms applied, loops used and changes around latest pr's may enhance capabilities of intuition around what areas of models are modified and targeted.

3. Shadow Testing- Deploying a new model parallel with the existing model, routing requests to but serving predictions only from the existing model, allowing for real-time performance comparison without impacting users.

4. Model Instability- Rapid and significant changes in the live data stream can lead to model instability, where the model's predictions become erratic or unreliable. QA testing should be involve around examining these input – by tuning the model, track test and train data values between iterative training rounds, notice data differences between train and test data – by recording overfitting, underfitting scenarios and identifying a range of good fit model data.

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

5. Behavioral Testing of NLP Models [10]

- **Invariance Testing -** To check if model's prediction remains constant despite specific changes in data input
- **Directional Expectation Test-** You can run directional expectation tests to define input distribution changes and expected effects on the output. A typical example is testing assumptions about the number of bathrooms or property size when predicting house prices. A higher number of bathrooms should mean a higher price prediction. Seeing different results might reveal wrong assumptions about the relationship between our input and output or the distribution of our dataset (e.g., small studio apartments are overrepresented in expensive neighborhoods) [11]
- **Minimum Functionality Test**: The minimum functionality test helps you decide whether individual model components behave as you expect. The reason behind these tests is that overall, output-based performance can conceal critical upcoming issues in your model. Here are ways to test individual components:

- Create samples that are "very easy" for the model to predict to see if they consistently deliver these types of predictions.
- Test data segments and subsets that meet specific criteria (e.g., run your language model only on short sentences of your data to see its ability to predict short sentences).
- Test for failure modes you have identified during manual error analysis. [11]

6. Latency and Performance Management- Achieving low latency in real-time processing and training requires careful performance management and optimization, as any delays can undermine the benefits of continuous learning. Test the model processing in case of low network, possible errors validation toast messages or service unavailable pages if there is a halt in continuous learning prediction.

7. Accuracy Testing Definition- This involves measuring how accurate the model is in predicting outcomes or making classifications. The dynamic nature of live data streams means that data distributions and the relationship between inputs and outputs can change over time, leading to **data drift** (changes in input data) and **concept drift** (changes in the relationship between inputs and outputs). These drifts can degrade model performance and lead to inaccurate predictions.

8. Batch Testing – on one epoch, check for degrading loss of data in epoch, check shape of output, produce overfit on a single batch, test early stopping conditions – in case good fit outcomes comes timely during model processing, saving time, check same model on different processing devices. (CPU, GPU)

9. Temperature Testing- In machine learning, "temperature" is a hyperparameter that influences the randomness or creativity of a model's output, particularly in tasks like text generation and classification. It acts as a "dial" that adjusts the probability distribution of the model's predictions. A higher temperature leads to more random and diverse outputs, while a lower temperature results in more predictable and focused outputs. It is particularly used in text generation LLM and NLP models.

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- **Low Temperature (e.g., 0.1-0.5):** Makes the model more deterministic, selecting the most probable next words. This is suitable for tasks requiring factual accuracy, conciseness, or adherence to strict formats (e.g., summarization, code generation, factual question answering).
- **High Temperature (e.g., 0.8-1.5):** Increases randomness, allowing the model to explore less probable but potentially more creative or diverse options. This is beneficial for tasks like creative writing, brainstorming, or generating varied conversational responses.
- **Default Temperature (1.0):** Offers a balance between creativity and coherence.

10. Repeatability Testing- This ensures that when the model is asked the same question multiple times, it provides the same response consistently.

11. LLM Models testing techniques-

- **Output Accuracy-** checks for accurate results being predicted or text generated by LLM regarding the asked set of input statements. ML model operates according to its intended purpose and delivers meaningful outputs.
- **Style Transfer Testing**- This checks the model's ability to adjust the tone or style of output, depending on user input or context. For example- A text generation model could be asked to write a formal email, and then a casual message on the same topic. The responses should differ in tone, formality, and word choice while remaining coherent.[12]
- **Relevancy Testing and Fantasy Claims Testing**- This testing makes sure model remains relevant to the questions asked/predictions asked and does not fake answers/ does not make irrelevant claims beyond known facts. Example: Asking a health-related AI, "Can drinking tea cure cancer?" and making sure it doesn't respond with an unscientific or fantastic claim like "Yes, tea is a miracle cure for all diseases." [12]
- **Intent Recognition by Model**- This tests whether the model can correctly interpret the intent behind the user's input, including recognizing sarcasm or humor.
- **Content Management Testing**- This evaluates the model's ability to retain context over multiple interactions, maintaining coherence. A series of conversations with a Agentic AI or chat bot should be able to retain and answer with the same chain of thought and should not deviate from topic.
- **User Action Prompt with choices-** This checks if the model can present actionable options in response to user inputs.
- **Zero Shot Testing –** testing with unseen prompt or not seen training or test data. In regular ML models – also called cross validation testing.
- **Repeatability and invariance testing** also applies in LLM model testing.

Drift Detection or Anomaly Detection

A. **What is Data Drift-** Drift detection in ML models refers to the process of identifying when the statistical properties of the data used for training a machine learning model differ significantly from the data the model encounters in a production environment. This discrepancy, known as "drift," can lead to a degradation in the model's performance and accuracy over time**.**

B.

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

B. Types of Drift-

- Data Drift (Covariate Drift):Changes in the distribution of input features. For example, if a model trained on customer demographics suddenly encounters a new demographic group not present in the training data.
- Concept Drift: Changes in the relationship between input features and the target variable. This means the underlying concept the model is trying to predict has evolved. For instance, in a spam detection model, new types of spam might emerge that the model wasn't trained to identify.
- Prediction Drift: Changes in the distribution of the model's predictions. This can be a symptom of either data or concept drift. It can also occur if the model adjusts well to the new environment [13]

C. Data Drift Detection Measures-

- **Visible Stats Summary-** Noticeable changes in mean, median, mode, variance, quartiles of input data can indicate data drift. Although alone these changes do not guarantee drift but a viable option in case of limited features.
- **Feature Range Compliance**- track feature range compliance, such as whether the values stay within a min-max range. This helps detect data quality issues, such as negative sales or sudden shifts in feature scale. However, this approach might not catch data drift when values remain within the expected range but show a different distribution pattern [13]
- **Statistical Tests-** Kolmogorov-Smirnov (KS) test: For numerical features, comparing the cumulative distribution functions of two samples. The results of the K-S test show that two data sets appear to come from different populations or same population, in case of different sources the data drift has likely occurred, making the K-S test a reliable drift detector. Chi-squared test: For categorical features, comparing observed and expected frequencies. Page-Hinkley method: Detects changes in the meaning of a data series over time.
- **Distance Metrics-** Jensen-Shannon Divergence: Measures the similarity between two probability distributions. Euclidean distance, Cosine distance: Used to compare numerical data or embeddings. Wasserstein distance compares training data to new input data being fed into a machine learning model. It excels in identifying complex relationships between features and can navigate outliers for consistent results
- **Model-based approaches-** Training a separate "drift detector" model to identify when the current data deviates from the expected patterns.
- **Rule-based checks-** To perform some drift checks using straightforward rules. For example, the share of the predicted “fraud” category is over 10%. A new categorical value appears in a feature “location” or “product type.” More than 10% of the feature “salary” values are out of the defined min-max range. [13]
- **Continuous Evaluation-** In this method, the model performance is validated after every n number of days using newly labeled test data drawn from a recent time. The model drift can be understood by comparing the performance of the model in the newly drawn test data with older ones.

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- **Population Stability Index (PSI)-** It is a metric to quantify how much a variable has changed in distribution between two samples drawn at different times. PSI for features are calculated to identify drift. If this value is less than 0.1 it indicates very slight change, a value between 0.1 to 0.2 indicates some minor change and a value greater than 0.2 indicates significant change in the distribution.
- **Z-Score-** The feature distribution of training data and live data can be compared using z-score. A z-score of +/- 3 indicates that the data might have changed. [14]

D. **Best practices to avoid model drift-**

• Automate Drift Detection - This program for detecting model drift should also track which transactions caused the drift, enabling them to be relabeled and used to retrain the model, restoring its predictive power during runtime.

• Automate Data Summary Validation - Automatic comparison of set of input data to test data – if specific in a range and flag the ML system to early stop/terminate.

• Automate Model -

– Validation of models in preproduction with tests to detect bias and drift and then generating test reports.

– Transferring the successful predeployment test configurations for a model to the deployed version of the model and continued automated testing.

– Synchronizing model, data and test result information with systems of record.

– Automation that can provide consistent and reliable notifications and provide more time for teams to focus on model development instead of model monitoring. [15]

• Drift Monitor - Trigger alerts for performance drops

• Retraining - Update model with new data patterns

• Root Cause Analysis - Analyze the root cause/or impact of any external factor which forces data drift/model drift. If in control, force stops the causal event.

• Real time retraining - Consider modifying the decision process on top of the model output. For example, when running a fraud detection model, we might lower the decision threshold that flags transactions as suspicious to send more of those for manual review. Consider adding other business rules to filter out potentially unreliable model predictions, such as overriding some extreme predictions.

Consider switching to other decision-making processes, from alternative fallback models to the human in the loop. For example, you can bring in domain experts to replace the automated decisions with good old human decision-making.

• Programmatically - Evidently is an open-source Python library that helps implement testing and monitoring for production machine learning models. Evidently helps run various checks for your datasets and get interactive visual reports to analyze the results, including data drift. [10]

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Other tools- River ML, Alibi Detect

• Adversarial Data Testing - Adversarial testing involves proactively challenging Machine Learning (ML) models with intentionally crafted, often malicious or difficult-to-handle input data. This process is crucial for understanding how the model behaves under stress, identifying vulnerabilities, and ultimately building more robust and reliable AI systems. Intentionally hitting ML model with data faults, unknown data can help proactively prepare model for drift in advance.

Dynamic Test Oracle methodologies suited for non-static ML environments

A test oracle in software testing is a mechanism used to determine whether the actual output of a software system matches the expected output for a given test case. It essentially acts as a reference point to validate the correctness of the software's behavior. Without a test oracle, it's difficult to objectively assess whether a test has passed or failed.

Following are few considerable test oracles to justify the performance of ML systems.

A. Consistency-based Test Oracles

Principle: If the test input applied to the model is similar, the output should be in a similar range and close -by.

Implementation:

- Maintain a **cache of recent inputs and corresponding outputs**.
- Measure **Prediction Consistency Score (PCS)** by comparing outputs for repeated or similar inputs.
- Flag inconsistencies (e.g., >10% variation) as potential QA violations.

Use Case:

- Recommendation Systems, Fraud Detection Models.

B. Weak Supervision-based Oracles

Principle: Use similarity rules to group inputs, **heuristic rules, programmatic labeling functions, or human-in-the-loop feedback** to generate “weak labels” that act as pseudo ground truths.

Implementation:

- Employ tools like **Snorkel** or custom rule-based scripts to auto-label data points.
- Compute a **Confidence Score** for each weak label to weigh oracle trust.
- Validate model predictions against these pseudo-labels.

Use Case:

- Sentiment Analysis, Entity Recognition, Anomaly Detection.

C. Proxy Metrics-based Oracles

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Principle: Indirectly infer prediction correctness using **proxy behavioral or business metrics**.

Examples:

- Click-Through Rate (CTR) improvements in recommendation systems.
- Fraud alerts validated by downstream transaction cancellations.

Implementation:

- Define domain-specific KPIs as real-time QA proxy indicators.
- Correlate significant deviations in proxy metrics with prediction anomalies.

Use Case:

- Online Ads, eCommerce personalization, financial fraud detection.

D. Cross-Model Agreement (Ensemble Oracles)

Principle: Run multiple models (ensemble, shadow models, prior versions) and check for discrepancies in outputs.

Implementation:

- Deploy a lightweight **shadow model** (or ensemble) alongside production.
- Trigger QA alerts when the **production model significantly disagrees with the ensemble average**.

Use Case:

- Model version upgrades, Continuous deployment of ML.

E. Domain-Invariant Constraints (Specification-based Oracles)

Principle: Define domain-specific **logical or physical invariants** that predictions must respect.

Examples:

- Credit Risk Scores must lie between 0 and 1.
- Anomaly detection scores must increase for known outliers.

Implementation:

- Encode these **specification rules** into automated validation scripts.
- Violation of these invariants signals QA failures.

Use Case:

- Fintech models, Sensor data processing, Healthcare diagnostics.

F. Drift-Based Test Oracle Approximation

Principle: Significant input data drift correlates with potential output inaccuracies.

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Implementation:

- Use **drift detection frameworks (Evidently AI, River ML)** to monitor feature distribution shifts.
- Outputs during high-drift windows are flagged as "QA-critical" requiring deeper inspection.

Use Case:

- Evolving data environments, Concept Drift-prone applications.

G. Model Confidence & Uncertainty-Based Oracles

Principle: Use the model's own confidence scores or uncertainty estimates to flag dubious predictions.

Techniques:

- Monte Carlo Dropout for Bayesian Uncertainty Estimation.
- SoftMax entropy measures.

Implementation:

- Set thresholds for acceptable confidence intervals.
- Predictions with low confidence are flagged for human review or QA checks.

Use Case:

- Image Classification, NLP tasks, Safety-critical systems.

H. Human-in-the-Loop (HITL) Feedback Loops

Principle: When automatic validation is insufficient, it involves humans selectively in reviewing predictions.

Implementation:

- Active Learning techniques to select **"high-uncertainty" or "QA-critical" predictions** for manual review.
- Continuously refine the model based on feedback.

Use Case:

- Content Moderation, Medical Diagnostics, Data Labeling augmentation.

I. Anomaly Detection on Output Space

Principle: Treat the model's output distribution as a **time-series anomaly detection problem**.

Implementation:

- Use algorithms like Isolation Forest, One-Class SVM, or Autoencoders to flag abnormal prediction patterns.

Received: August 07, 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Use Case:

- Fraud detection, Outlier monitoring in operational systems.

J. Recommendation: Hybrid Oracle System

For robust QA, implement a **hybrid oracle architecture** that:

1. Uses **automated dynamic oracles (consistency, drift, constraints)** for real-time QA.
2. Selectively escalates to **HITL reviews** based on confidence thresholds or anomaly flags.
3. Continuously refines oracle heuristics through feedback loops.

Feedback Loops and Quality Alerts for continuous model Retraining in MLOPs

Feedback loops and quality alerts are crucial for continuous model retraining in MLOps, enabling models to adapt and improve over time. These mechanisms ensure models remain relevant and effective by continuously learning from real-world performance data. Feedback loops involve using model outputs to refine the model, while quality alerts signal when the model's performance degrades, triggering retraining.

Model retraining and feedback loops are essential components of robust and adaptive machine learning systems. By automating retraining workflows and leveraging tools for incremental learning, organizations can ensure their models remain relevant in dynamic environments. The key lies in integrating data monitoring, retraining triggers, and effective feedback loops into a cohesive pipeline. With the right frameworks and strategies, retraining becomes a seamless and impactful process that drives sustained model performance.

A. Continuous QA Monitoring- Real-time monitoring of QA metrics:

Compare model predictions against delayed ground-truth labels (where available).

In many real-world machine learning scenarios, the "ground truth" (the actual outcome or label) for a model's predictions isn't immediately available. Instead, there's a delay, sometimes weeks or even months, before the true outcome can be observed. This delayed feedback poses a unique challenge when it comes to monitoring and evaluating a model's performance in production.

- **Model Accuracy (Rolling Window)- Time-windowed evaluation:** Group predictions and their corresponding delayed ground truth within specific time windows (e.g., daily, weekly, or monthly) to track performance trends over time.
- **Data & Concept Drift- Monitor data drift:** Changes in the input data distribution over time (data drift) can impact model performance even with accurate ground truth. Tracking data drift can help explain why a model's performance might be degrading, even before the delayed ground truth arrives.
- **Back testing with historical data**: Use historical data with known ground truth to simulate how the model would perform with different delays in ground truth availability.

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

This can help you anticipate how your model might behave in scenarios with delayed feedback.

- **Establish a baseline:** Even with delayed feedback, it's possible to assess performance over time. You can compare predictions made at a particular point against the ground truth that becomes available later, using standard metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) for regression tasks, or accuracy, precision, and recall for classification tasks.

B. Anomaly Detection & Scoring

- Implement **Anomaly Scoring Functions**:

1Drift Score (PSI, KS Test results).
2Accuracy Deviation Score.
3Fairness Disparity Score- Demographic Parity Difference / Equal Opportunity Difference. Real-Time Fairness Metrics by Sensitive Attributes (e.g., Gender, Region).
4Stability Score.

- Compute a **Composite Anomaly Risk Index (CARI)**.

1Formula Example:
CARI = w1*(Drift Score) + w2*(Accuracy Drop %) + w3*(Bias Disparity) + w4*(Stability Variance)

2Thresholds:

- **CARI > 0.7:** High Risk (Immediate Intervention).
- **CARI 0.5–0.7:** Moderate Risk (Human QA Review).
- **CARI < 0.5:** Normal (No Action).

C. Early Warning Trigger Mechanism

- When thresholds are breached:
  - Generate alerts via Slack, Email, or PagerDuty.
  - Push events into a **message queue (Kafka/SQS)**.
  - Log incident in QA dashboards.
  - Datadog queries and trigger alerts.

D. Automated Feedback Loop Actions

Feedback loops are a process where model outputs are evaluated and used to refine the model iteratively. They involve feeding the model's predictions back into the system to learn from both correct and incorrect predictions.

- **Purpose**: Feedback loops help identify weaknesses in the model, adapt to changing data distributions, and improve performance over time.

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- **Examples:**
- Model Outputs as a Source of Learning: Using the model's predictions as new training data to refine its understanding of the input space.
- User Feedback Integration: Incorporating user interactions and feedback to identify areas where the model needs improvement.
- Automated Retraining Pipelines: Automatically retraining models on new or updated data to adapt to changes in the environment.

E. Live Retraining Workflow

1. Pull latest data batch from the data lake.
2. Validate data quality (schema, null checks, integrity).
3. Retrain model using existing pipeline (with adjusted sampling or fairness constraints if flagged).
4. Evaluate retrained model on recent streaming data.
5. Deploy retrained model (canary deployment first).
6. Re-enable monitoring loop.

F. MLOps Tools for Automation

- Monitoring and Alerts - Use tools like Prometheus, Grafana, Evidently AI, Arize AI
- Anomaly Scoring Engine - Use tools like Custom Python service (pandas, scikit-learn), River ML
- Messaging and Orchestration - Use tools like Kafka, Apache Airflow, AWS Step Functions
- CI/CD Pipeline Automation - Use tools like Kubeflo Pipelines, MLflow, Argo Workflows
- Retraining Automation - Use tools like TensorFlow Ex-tended (TFX), Sagemaker Pipelines

G. Feedback Loop Decision Logic

Events **Action Path**

- **High Risk (CARI > 0.7) =** Auto-trigger retraining → deploy → re-monitor.
- **Moderate Risk (CARI 0.5–0.7) =** Escalate to QA Analysts for Human Validation → Action.
- **Low Risk (CARI < 0.5) =** Log event for reporting, no immediate action.

H. Visualization Dashboard Panels

- Live QA Health Score (CARI Gauge).
- Drift Trend Lines (Features PSI over time).

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- Retraining Trigger Logs & Action History.
- Real-time Alert Feed (Slack-integrated).

I. Self-Healing QA Bots

- Implement "QA Bots" that:
  - Analyze root causes of QA failures.
  - Suggest retraining parameters (e.g., changing sampling strategy).
  - Recommend manual interventions where automation is risky.

Other Quality Measures for Fair responsive ML Testing

A. **Responsible /Ethical AI Testing**-It refers to the process of evaluating and ensuring that artificial intelligence (AI) systems are developed and deployed in a manner that aligns with ethical, fair, transparent, and reliable standards. The goal is to address the risks that AI can pose, including bias, lack of transparency, security vulnerabilities, and unintended societal impacts.

B. **Fairness Testing**-Fairness testing ensures that the model treats all demographic groups equitably, without unfair bias based on race, gender, age, socioeconomic status, etc. Example: In a hiring model, fairness testing might involve examining whether the algorithm favors one gender or racial group over another in making decisions on medicine suggestions.

C. **Bias Detection and Mitigation**-Identifying and mitigating inherent biases in the model. Example Testing image recognition model across diverse dataset for equal accuracy. Mitigation Methods Rebalancing training data and using fairness-aware algorithms.

D. **Transparency Testing** (Explainability)- Ensures model decisions can be easily understood and explained. Loan approval model explaining reasons for approval or denial.

E. **Ethical Testing**-Tests ethical implications of AI decisions, ensuring adherence to moral standards. The health diagnosis model does not recommend life-changing treatments with high confidence. Prevents overstepping boundaries in sensitive contexts like healthcare or criminal justice.

F. **Data Privacy and Security Testing**- Ensures model adheres to privacy regulations and protects sensitive information. Example- Facial recognition system not leaking personal images during testing or production. Differential privacy and federated learning for data protection.

G. **Model Generalization Testing**-Ensures model generalizes well across diverse environments and datasets. Example- Sentiment analysis model trained on U.S. data working well with other countries' data. Prevents performance drop-offs when applied to new or diverse datasets.

H. **Societal Impact Testing**-These tests the broader societal implications of deploying the AI model, ensuring that its use will have a net positive impact and align with social norms and values. Example: A predictive policing model could be tested to see if its use disproportionately

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

affects certain communities. Societal impact testing would check if such an algorithm reinforces existing inequalities or creates new problems. [12]

I. **Latency Testing**-Ensures predictions are made within an acceptable time frame. Measure prediction time and ensure it meets defined SLAs. Response time under normal and peak load, scalability testing.

J. **Integration and A/B Testing**- Compares new model performance against existing models with real users. Split user traffic between current (A) and new (B) models. Compare metrics, evaluate user interactions, ensure statistical significance.

K. **Privacy, PII Protection, and Regulatory Compliance**-Protecting user privacy and personally identifiable information (PII) is paramount in any industry, including AI. Adherence to regulatory compliance is not a 'can have' but a 'must have', as failure to do so can result in reputation damage and financial and legal consequences. [11]

L. **Accountability and Governance**-To monitor and responsibly manage the ML project life cycle, you need to create a data governance unit within your organization to build accountability and governance frameworks. This unit will be responsible for creating the rules and regulations for cross-functional teams working on your machine learning project at different phases, aiding adherence to regulatory compliance. [11]

Conclusion

Testing is an iterative process and can be difficult when working on live ML projects that may require huge amounts of data along with long model training cycles. For small and large teams, time is an important resource that they do not have a monopoly on. This means that teams must pick the best test frameworks that work for their unique situation. This might mean testing and validating random data samples in small batches or unit tests for testing specific behaviors of the model, among others that can be chosen. Different teams, no matter what their size, should integrate these practices into their overall ML project lifecycle to improve the overall quality of their product. The framework's architecture is designed to automate validation tasks throughout the model life cycle, integrating several key components: Real-time Data Validation and Drift Detection: Proactively identifying shifts in input data and feature distributions using methods like Population Stability Index (PSI) and Kolmogorov-Smirnov (KS) tests. Real-time QA Monitoring and Anomaly Scoring: Implementing metrics for accuracy, drift, fairness, and stability (Objective 2), aggregated into a Composite Anomaly Risk Index (CARI) to provide a holistic view of model health. Dynamic Test Oracles: Validating model outputs in the absence of static ground truth labels through techniques such as consistency-based checks, weak supervision, proxy metrics, and cross-model agreement (Objective 3). Automated Early Warning Systems and Feedback Loops: Triggering automated retraining and remediation interventions when quality thresholds are breached, effectively shortening feedback-to-repair cycles (Objective 4).

Future Scope and improvement Areas

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

While the framework's design offers a robust solution, its full impact is contingent upon thorough empirical validation in diverse real-world scenarios. The future advancement in different industrial domains will focus on solidifying these empirical claims, exploring advanced explainability techniques, proactive adversarial robustness, sophisticated Human-in-the-Loop integrations, and optimizing scalability for even broader application. Deployment teams need to measure the average performance of models and how much they vary with different amounts of prompts. We can determine how this model behaves under various types of stress, network outages by using performance test metrics. Otherwise, a sizable memory footprint could prevent a model from running on mobile devices or simple servers. By analysing output results, structured caching, prompts scalability and input filtration processes, this framework helps to balance decisions by ensuring models are compatible with the infrastructure environment. This continuous evolution will further enhance the resilience and trustworthiness of AI systems in an increasingly dynamic technological landscape.

REFRENCES

1. Alamin, M. a. A., & Uddin, G. (2021, May 3). *Quality assurance challenges for machine learning software applications during software development life cycle phases*. arXiv.org. https://arxiv.org/abs/2105.01195
2. Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*, *3*(2), 392–413. https://doi.org/10.3390/make3020020
3. Chatterjee, A., Ahmed, B. S., Hallin, E., & Engman, A. (2022, November 23). *Quality Assurance in MLOPs setting: An Industrial Perspective*. arXiv.org. https://arxiv.org/abs/2211.12706
4. Wang, C., Yang, Z., Li, Z. S., Damian, D., & Lo, D. (2024, February 26). *Quality assurance for Artificial intelligence: A study of industrial concerns, challenges and best practices*. arXiv.org. https://arxiv.org/abs/2402.16391
5. Ogrizović, M., Drašković, D., & Bojić, D. (2024). Quality assurance strategies for machine learning applications in big data analytics: an overview. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-01028-y
6. Pinna, S. (2024, December 2). How to Write a High-Impact Technical Research Paper. *International Journal of Innovative Research in Technology*. https://ijirt.org/article?manuscript=172216
7. Pandhare, H. V. (2025). Future of software test automation using AI/ML. International Journal of Engineering and Computer Science, 14(05), 27159–27182. https://doi.org/10.18535/ijecs.v14i05.5139
8. Y. Ferreiro, "What is Continuous Learning? Revolutionizing Machine Learning & Adaptability," *Datacamp.com*, Sep. 26, 2023. https://www.datacamp.com/blog/what-is-continuous-

Received: August 07, 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

learning?irclickid=3BzSbRyjTxycR641liRkfXVcUkpzynwFRW0PxE0&irgwc=1 (accessed Aug. 25, 2025).

9. C. Kästner, "Data Quality for Building Production ML Systems," *Medium*, Oct. 17, 2023. https://ckaestne.medium.com/data-quality-for-building-production-ml-systems-2e0cc7e6113f
10. Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.
11. Shir Chorev, "How to Test Machine Learning Models | Deepchecks," *Deepchecks*, Jun. 07, 2024. https://www.deepchecks.com/how-to-test-machine-learning-models (accessed Aug. 25, 2025)
12. Rahul Shetty, "Introduction to Machine Learning Models (AI) Testing," *Udemy*, 2025. https://www.udemy.com/course/machine-learning-models-ai-testing/ (accessed Aug. 25, 2025).
13. "What is data drift in ML, and how to detect and handle it," *www.evidentlyai.com*. https://www.evidentlyai.com/ml-in-production/data-drift
14. Domino, "What is Model Drift in Machine Learning? | Domino Data Lab," *domino.ai*. https://domino.ai/data-science-dictionary/model-drift
15. IBM, "Model drift," *Ibm.com*, Jul. 16, 2024. https://www.ibm.com/think/topics/model-drift

Received: August 07, 2025