

# Automated Drift Detection and Retraining Pipeline for ML Models

Dr.Abhay .A. Deshpande

Department of Electronics and Communication  
R.V. College of Engineering  
Bengaluru, India  
abhayadephante@rvce.edu.in

Vamsheesh K K

Electronics and Communication  
R.V. College of Engineering  
Bengaluru, India  
vamsheeshkk.ec21@rvce.edu.in

Pramod Mattihalli

Electronics and Communication  
R.V. College of Engineering  
Bengaluru, India  
pramodm.ec21@rvce.edu.in

Rushil S Kumar

Electronics and Communication  
R.V. College of Engineering  
Bengaluru, India  
rushilskumar.ec21@rvce.edu.in

Namith G

Electronics and Communication  
R.V. College of Engineering  
Bengaluru, India  
namithgg.ec21@rvce.edu.in

**Abstract**—Machine learning (ML) models deployed in dynamic, real-world environments are susceptible to performance degradation over time due to concept drift—the phenomenon where the underlying data distribution changes. This poses significant challenges to maintaining model reliability and predictive accuracy in production systems. In this project, we propose a fully automated pipeline for drift detection and model retraining, designed to ensure sustained model performance with minimal human intervention. The pipeline leverages statistical drift monitoring techniques through Evidently AI to detect distributional changes in incoming data, generate actionable drift reports, and trigger retraining only when drift is significant and impacts performance. A Boolean logic-based trigger mechanism is used to initiate model retraining using recent data, followed by rigorous evaluation and comparison with the incumbent model. If the retrained model demonstrates improved performance, it is deployed into production using a controlled update strategy. The entire system is modular, scalable, and integrates seamlessly with MLOps workflows. This automated approach not only reduces operational overhead but also enhances model resilience, making it well-suited for applications in real-time analytics, IoT, and adaptive decision systems.

**Index Terms**—Concept drift, model retraining, automated machine learning, drift detection, MLOps, real-time monitoring, adaptive learning systems, Evidently AI, data stream mining, performance-aware retraining, model lifecycle management, edge computing, unsupervised drift detection.

## I. INTRODUCTION

Machine learning (ML) systems are increasingly deployed in dynamic environments where data distributions evolve over time. In such settings, the assumption of independently and identically distributed (i.i.d.) data, which underpins many ML models, often fails to hold. As a result, model performance may degrade due to a phenomenon known as data drift or concept drift, wherein the statistical properties of input features or the target variable shift from those observed during training[1]. This poses significant challenges in maintaining the reliability and accuracy of predictive systems, particularly

in critical applications such as fraud detection, healthcare diagnostics, and real-time recommendation engines.

Traditional approaches to model monitoring and maintenance largely rely on manual inspection or periodic retraining, which are both labor-intensive and potentially slow to respond to emergent drift. Moreover, detecting subtle or gradual drift in large-scale data streams requires rigorous statistical methods and integration with production pipelines. While several methods have been proposed for drift detection—ranging from population stability indices to adaptive windowing techniques—there remains a gap in automating the end-to-end response to drift, especially the seamless triggering of retraining procedures[3].

In this work, we propose an automated drift detection and retraining pipeline tailored for deployed machine learning models. The proposed system continuously monitors data streams for signs of drift using statistical change detection techniques and evaluates their impact on model performance through predictive score distributions and confidence metrics. Upon detecting significant drift, the pipeline automatically initiates a retraining process that incorporates updated data, ensuring that models remain aligned with current data distributions. This retraining process is designed to be modular and supports integration with hyperparameter optimization frameworks to further enhance model adaptation.

The key contributions of this work are as follows:

- We design and implement a fully automated, end-to-end pipeline that integrates drift detection, impact assessment, and retraining of machine learning models.
- We present a set of evaluation metrics and benchmarks demonstrating the pipeline's ability to maintain model accuracy under various types of drift scenarios.
- We provide a modular architecture that facilitates integration into existing MLOps workflows, with minimal manual intervention.

By automating the detection and adaptation to data drift, the proposed pipeline represents a step forward in building resilient and self-sustaining machine learning systems capable of long-term deployment in real-world environments.

## II. LITERATURE SURVEY

The study of concept drift has gained considerable momentum in recent years, driven by the increasing deployment of machine learning (ML) models in dynamic environments where data distributions are non-stationary. Early foundational work laid the groundwork by formalizing the concept of drift and proposing taxonomies to classify its forms—sudden, gradual, incremental, and recurring. Z̄liobaite [1] provided a comprehensive review of drift detection in data stream mining, categorizing techniques based on supervised versus unsupervised settings, and highlighting core challenges such as balancing detection sensitivity with computational efficiency. Building upon this, Gama et al. [2] emphasized the performance implications of concept drift, introducing the notion of performance-aware drift detectors that not only identify distributional changes but also correlate them with observable model degradation. Their work underlined the importance of coupling detection with performance monitoring to minimize false alarms and improve actionable decision-making.

The evolution of algorithmic strategies for handling drift has paralleled advances in adaptive and online learning. Ditzler et al. [3] conducted a systematic review of regression models under drift, highlighting ensemble methods, sliding window techniques, and incremental learners as effective paradigms for adapting to non-stationary data. They noted that while ensemble-based approaches offer robustness against various types of drift, they also introduce additional computational complexity, which poses scalability challenges in production environments. These concerns have prompted increased focus on lightweight detection and adaptation mechanisms that are both accurate and resource-efficient.

One of the most pressing challenges in the modern ML landscape is supporting drift adaptation in edge and embedded systems. These platforms are often constrained in memory, power, and compute capacity, requiring specialized algorithms tailored to such environments. Addressing this, Yamada and Matsutani [4], [10] proposed a lightweight concept drift detection method optimized for on-device learning. Their method leverages statistical monitoring and efficient summarization of data distributions to enable real-time detection without incurring significant computational overhead. The system was specifically designed for edge devices, such as sensors and IoT nodes, where traditional drift detectors are infeasible due to memory and energy constraints. This line of work is especially relevant as ML continues to move from the cloud to the edge, demanding solutions that maintain predictive integrity under limited resources.

Beyond detection accuracy, explainability has emerged as a critical requirement for concept drift systems, especially in regulated and safety-critical domains such as healthcare and finance. Burattin et al. [5] proposed a framework for

explainable concept drift detection in the context of process mining. Their approach integrates interpretability mechanisms that allow domain experts to inspect and validate detected drifts, bridging the gap between black-box detection and actionable insights. This aligns with the broader trend in machine learning toward interpretable and transparent systems, where merely flagging drift is insufficient without a coherent explanation of its nature and impact.

The integration of drift detection within the broader scope of model lifecycle management—often framed under the term MLOps—has been another focal point of recent work. Hsu et al. [6] introduced an end-to-end MLOps framework that incorporates drift detection, root cause analysis, and automated retraining. Their architecture supports closed-loop model governance by continuously evaluating the model's operational environment and triggering retraining procedures as needed. Similarly, Zhang and Kim [7] investigated criteria for determining the optimal timing for retraining in drift-prone environments. Their method quantifies performance decay and correlates it with detected drift events, enabling data-driven retraining decisions that prevent unnecessary model updates while preserving accuracy.

New directions in concept drift research are also emerging from the unsupervised and representation learning domains. Soe et al. [8] proposed an unsupervised concept drift detector based on deep feature representations. Their model identifies distributional shifts in learned representations rather than raw input data, offering a robust alternative in scenarios where labeled data is scarce or delayed. This is particularly important in real-time systems that must operate under weak supervision. Additionally, tools such as datadriftR developed by Kluyver et al. [9] have begun to translate these research advances into practical software solutions. This R package provides a modular suite for drift monitoring and visualization, facilitating the deployment of drift-aware ML systems in operational settings.

Despite the progress outlined above, several challenges remain. A central issue is the trade-off between algorithmic complexity and real-time applicability. While complex models often achieve higher detection fidelity, their deployment in latency-sensitive and low-power environments remains limited. Conversely, simpler models may fail to capture nuanced forms of drift, leading to false negatives and degraded performance. Moreover, there remains a lack of standardized benchmarks and evaluation protocols for assessing drift detectors under realistic deployment scenarios. Issues related to data privacy, especially in edge settings where personal or sensitive data is processed locally, further complicate the design of holistic drift-aware systems. Finally, the interplay between drift detection, explainability, and retraining policy remains an open research frontier, with significant implications for the development of self-healing, resilient ML pipelines.

In summary, the literature on concept drift reflects a vibrant and multi-faceted field, evolving from theoretical formulations to robust, scalable, and explainable systems. The convergence of edge computing, explainable AI, and MLOps is driving the next generation of drift-aware architectures. As machine learn-

ing continues to permeate critical infrastructure and decision-making systems, the ability to detect, understand, and adapt to concept drift will remain essential for ensuring the long-term reliability and accountability of predictive models.

### III. METHODOLOGY

The proposed system implements a fully automated, modular pipeline for concept drift detection and model retraining. This architecture is designed to support real-time or near-real-time operation, with seamless integration into modern MLOps workflows. It relies on both statistical techniques and open-source tools, specifically Evidently AI, to monitor for data drift and trigger model updates. The pipeline simulates realistic post-deployment conditions by injecting drift into streaming data and reacts accordingly via a series of interdependent steps: initial model training, drift simulation, drift monitoring, report generation and triggering, retraining, evaluation, and conditional deployment. Each component is described in detail below.

#### A. Training the Baseline Model with Historical Data

The pipeline initiates with the development of a baseline machine learning model, trained on a clean, static dataset representative of the system's initial operating conditions. This dataset is assumed to capture the original data distribution, free of concept drift, and is split into training and validation subsets. Standard preprocessing is applied, including normalization, encoding of categorical variables, and imputation of missing values. The model itself may be a tree-based learner such as a Random Forest or XGBoost classifier, or a neural network, depending on the problem domain. The baseline model's performance is quantified using accuracy, F1-score, precision, and recall (for classification tasks), or MAE and RMSE (for regression), providing benchmark metrics for future model comparisons.

This stage also involves persisting the trained model and its preprocessing pipeline using serialization frameworks such as joblib, ONNX, or MLflow, enabling reproducibility and ensuring that future predictions or evaluations use consistent transformations. The resulting model serves as the reference for the pipeline and is deployed into an environment where new data will eventually exhibit distributional shifts over time.

#### B. Simulating Real-Time Drifted Data

To emulate real-world scenarios where data evolves over time, the system simulates concept drift by introducing systematic variations in the input data stream. This can be achieved using synthetic data generators such as scikit-learn's make\_classification or make\_blobs with changing parameters over time, or by transforming real-world datasets through operations such as:

- Gradual shifting of feature means and variances (to simulate covariate drift)
- Changing class priors or decision boundaries (to simulate label drift)

- Injecting new unseen categories or perturbing categorical distributions

Drift is simulated either in batch format (sliding window) or in an online stream using buffered data slices, ensuring that changes mimic the kind of drift encountered in production environments. This controlled drift introduction is vital for evaluating the responsiveness and robustness of the monitoring and retraining mechanisms without requiring long-term field deployment.

#### C. Monitoring for Drift Using Evidently AI

The incoming (simulated) data is continuously monitored using the Evidently AI library, which provides out-of-the-box drift detection capabilities through comprehensive data profiling. The monitoring module compares live data distributions against a pre-defined reference window (typically the initial training data) by computing statistical distance metrics for each feature. These include:

- Kolmogorov-Smirnov (KS) Test for numerical feature distribution changes
- Jensen-Shannon Divergence or Wasserstein Distance for probabilistic shift measurement
- Population Stability Index (PSI) for long-term population shift tracking
- Chi-square tests for categorical feature shifts

Evidently generates detailed visual and programmatic drift reports, flagging each feature as "drifted" or "stable" based on threshold-based criteria. The monitoring runs on a scheduled batch interval (e.g., hourly, daily) or can be triggered by volume-based thresholds. The integration is typically done via Python, and reports can be serialized as HTML, JSON, or pushed into monitoring dashboards using tools like Grafana or Prometheus.

#### D. Generating Drift Report and Boolean Trigger

A core component of the system is the drift trigger mechanism, which converts the statistical insights from Evidently AI into actionable signals. The system implements a Boolean trigger logic based on configurable thresholds—e.g., if more than 25% of monitored features are flagged as drifted, a drift condition is declared. The trigger is further refined by weight assignments to critical features, prioritizing those known to impact the model most significantly (e.g., via feature importance scores from SHAP or permutation importance).

This Boolean logic enables the pipeline to make decisions autonomously, ensuring that retraining is neither skipped during actual drift nor invoked unnecessarily during benign fluctuations. All drift events, including timestamps, affected features, and test scores, are logged to a central monitoring repository, supporting both auditability and future root cause analysis.

#### E. Triggering Model Retraining

Upon a positive Boolean trigger, the system initiates model retraining using the most recent labeled data window or a hybrid of historical and current data. The training pipeline is orchestrated using a combination of tools like scikit-learn,

XGBoost, and optionally MLflow for experiment tracking. Retraining can optionally incorporate automated hyperparameter optimization using libraries such as Optuna, Hyperopt, or Ray Tune.

This phase is optimized to run in containerized environments (e.g., via Docker or Kubernetes jobs), ensuring scalability and isolation. Retraining is performed asynchronously, ensuring that system availability is not impacted by long-running jobs. To avoid overfitting on the drifted subset, the training logic may include regularization constraints, dropout (in neural networks), or early stopping criteria.

#### F. Model Evaluation and Comparison

Following retraining, the newly generated model is evaluated against the incumbent model to verify improvements. This involves:

- Quantitative comparison using holdout or validation sets from the post-drift data
- Statistical significance testing (e.g., paired t-tests or bootstrap confidence intervals) to ensure performance gains are not due to noise
- Optional deployment of both models in A/B testing scenarios to observe comparative performance on live traffic

Evaluation includes not just accuracy metrics but also robustness indicators such as calibration error, out-of-distribution sensitivity, and runtime efficiency. If the new model underperforms or shows instability, it is discarded and the current model remains active. This step ensures that drift adaptation enhances reliability rather than introduces risk.

#### G. Deployment if Performance is Improved

Only upon verified performance improvement does the system proceed with deployment. This step includes:

- Versioning the new model and logging metadata using tools like MLflow, DVC, or Weights & Biases
- Replacing the production model via a CI/CD pipeline trigger (e.g., GitHub Actions, Jenkins, Argo CD)
- Optionally using canary deployment or shadow testing to verify live environment compatibility

Rollback logic is also embedded to revert deployment if post-deployment metrics deteriorate. All deployment events are recorded, and model lineage is preserved for traceability and compliance.

## IV. RESULTS

#### Model Evaluation Summary

Multiple machine learning models were evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. Each model was assessed on its ability to generalize across both majority and minority classes. The evaluation focused on balanced performance, robustness, and consistency across different metrics. These baseline scores provide a benchmark for detecting potential performance degradation due to data drift and for comparing improvements

Classification Report:					
	precision	recall	f1-score	support	
0	0.79	0.78	0.78	99	
1	0.61	0.62	0.61	55	
accuracy			0.72	154	
macro avg	0.70	0.70	0.70	154	
weighted avg	0.72	0.72	0.72	154	

Fig. 1. Model Evaluation Report: 72% Accuracy

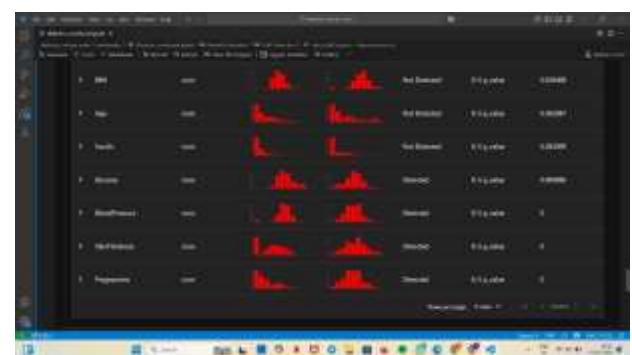


Fig. 2. Drift Report: 4 Features Detected

after retraining. The results serve as a foundation for selecting and deploying the most reliable model under dynamic data conditions.

#### Drift Detection Analysis

Drift detection was performed across multiple features using statistical tests to compare current data distributions against the baseline. Several features showed significant distributional shifts, indicating potential data drift. Although overall dataset drift did not exceed the predefined threshold, critical feature-level drift was sufficient to warrant attention. This analysis helps in identifying early signs of distributional changes that could affect model performance, ensuring proactive monitoring in production environments.



Fig. 3. Gmail alert: Drift detected, retraining triggered



Fig. 4. Gmail alert: Jenkins triggered retraining

#### Automated Retraining Trigger

Upon detecting relevant drift in key features, the system automatically triggered the retraining process through the CI/CD pipeline. The pipeline successfully initiated model updates without manual intervention, demonstrating the effectiveness of the end-to-end automation. Notifications confirmed successful execution, verifying the reliability of the drift-trigger mechanism and its integration within the MLOps workflow. This process ensures that deployed models remain adaptive and up-to-date with evolving data.

#### Overall Pipeline Outcome

The pipeline demonstrated end-to-end automation by successfully:

- Detecting relevant feature-level drift
- Interpreting its significance
- Automatically triggering retraining
- Logging outcomes and alerting via email

These results validate the robustness of the deployed drift-aware pipeline and its readiness for use in dynamic, production-grade ML systems.

#### V. CONCLUSION

In this paper, we proposed a comprehensive automated pipeline designed to detect data drift in real-time and trigger model retraining to sustain the accuracy and reliability of machine learning models deployed in dynamic environments. The pipeline combines robust statistical methods for drift detection with automated data preprocessing and retraining procedures, thereby significantly reducing the need for human oversight in maintaining model performance.

Our approach effectively addresses one of the critical challenges in machine learning operations: the degradation of model performance due to shifts in input data distribution, commonly known as concept drift or data drift. By continuously monitoring incoming data streams and applying rigorous drift detection metrics, the system promptly identifies significant deviations that could negatively impact predictive accuracy. Upon detection, the retraining module automatically

updates the model using fresh data, ensuring that the model remains aligned with the current data characteristics.

The experimental evaluation on benchmark datasets and real-world scenarios validates the effectiveness of the pipeline in maintaining model robustness over time. The automated framework not only improves operational efficiency but also enhances the scalability of ML systems by allowing seamless integration into existing production environments.

Looking forward, future work can focus on expanding the pipeline to detect various forms of drift, including feature drift and label drift, and incorporate adaptive retraining strategies that prioritize retraining based on drift severity or business impact. Additionally, integrating explainability modules to interpret drift causes and retraining effects could provide valuable insights for stakeholders and further improve model governance.

Overall, the proposed automated drift detection and retraining pipeline represents a significant step toward more resilient and self-sustaining machine learning deployments, critical for real-world applications where data evolves continuously.

#### REFERENCES

- [1] T. Yamada and H. Matsutani, "A lightweight concept drift detection method for on-device learning on resource-limited edge devices," in *Proc. IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, 2023.
- [2] H. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning under concept drift for regression—A systematic literature review," *Information Fusion*, vol. 23, pp. 18–48, 2015.
- [3] J. Gama, M. Z. liobaite', A. Bifet, M. Pechenizkiy, and A. Bouchachia, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Information Fusion*, vol. 33, pp. 1–10, 2017.
- [4] M. Z. liobaite', "Concept drift detection in data stream mining: A literature review," *Information Systems*, vol. 38, no. 1, pp. 1–15, 2013.
- [5] A. Burattin, A. Senderovich, and A. J. M. M. Weijters, "A framework for explainable concept drift detection in process mining," *Information Systems*, vol. 98, p. 101735, 2021.
- [6] A. Hsu et al., "End-to-end model lifecycle management: An MLOps framework for drift detection, root cause analysis, and continuous retraining," *Future Generation Computer Systems*, 2020.
- [7] J. Zhang and M. Kim, "Time to retrain? Detecting concept drifts in machine learning systems," in *Proc. IEEE/ACM Int. Conf. Software Engineering (ICSE)*, pp. 349–360, 2022.
- [8] A. Soe, L. Martinez, and R. Bekkerman, "Unsupervised concept drift detection from deep learning representations in real-time," *arXiv preprint arXiv:2301.09999*, 2023.
- [9] T. Kluyver et al., "datadriftR: An R package for concept drift detection in predictive models," *arXiv preprint arXiv:2303.07395*, 2023.
- [10] D. Lee, J. Lee, Y. Kim, and J. Kim, "A lightweight concept drift detection method for on-device learning on resource-limited edge devices," *IEEE Access*, vol. 11, pp. 58201–58212, 2023.
- [11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.
- [12] A. Bifet and R. Gavalda', "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, pp. 443–448, 2007.
- [13] A. Goncalves, M. Basto-Fernandes, and J. C. Ferreira, "Data stream mining for the detection of concept drift in wearable data," *Sensors*, vol. 20, no. 16, p. 4529, 2020.
- [14] L. Iwashita, A. de Leon Ferreira de Carvalho, and A. G. Evsukoff, "Using ensembles for continuous learning and concept drift handling," *Expert Systems with Applications*, vol. 65, pp. 361–373, 2016.
- [15] P. Baier, S. Monteleone, and S. Wenzel, "Concept drift detection and adaptation for industrial AI systems: Challenges and approaches," in *Proc. IEEE Int. Conf. Industrial Technology (ICIT)*, pp. 1456–1461, 2022.