Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models

Hanqi Zhang
Computer Science, University of Michigan at Ann Arbor, MI, USA
hz0102@yahoo.com

DOI: 10.69987/JACS.2023.30703

Keywords

distribution shift, concept drift, CTR, CVR, monitoring, change-point detection, MMD, PSI, retraining, rollback, MLOps.

Abstract

When CTR/CVR prediction models are deployed in production, their offline validation accuracy is rarely stable. Real traffic evolves continuously due to changes in user intent, inventory, policies, and the surrounding ecosystem. As a result, both distribution shift (covariate and label shift) and concept drift can silently degrade business KPIs. This paper presents DriftGuard, a practical monitoring-and-mitigation framework that couples multi-signal drift scoring with change-point detection and an action policy layer that can automatically trigger re-training and safely rollback. DriftGuard fuses (i) a covariate-shift score based on a linear Maximum Mean Discrepancy (MMD) between reference and recent feature distributions, (ii) a prediction-shift score based on the Population Stability Index (PSI) over predicted probability histograms, and (iii) a performance-shift score that monitors batch log-loss and a Page–Hinkley increment statistic. A lightweight CUSUM-style change-point detector converts scores into alarms, while an action policy decides between periodic retraining, alarm-triggered retraining, and alarm-triggered retraining with rollback based on post-deployment canary performance. We conduct end-to-end experimental evaluations on three public benchmarks that cover both tabular/text CTR-like streams and KPI time series: the WILDS CivilComments dataset, the Wild-Time HuffPost benchmark, and the Numenta Anomaly Benchmark (NAB). To obtain controlled ground-truth drift types, we construct reproducible synthetic deployment streams from the full datasets via stratified sampling that induces covariate shift, label shift, and temporal shift. Across the two CTR/CVR-style streams, DriftGuard achieves strong transition-level drift detection (mean AUROC 0.933 on CivilComments and 0.917 on HuffPost) and supports mitigation policies that reduce cumulative log-loss by 20.6% and 25.8% respectively versus a no-action baseline. On NAB KPI streams, DriftGuard is competitive with strong statistical baselines and provides interpretable alarms aligned with annotated anomaly windows. These results suggest that combining complementary signals with conservative mitigation policies yields a robust, engineering-ready approach to drift management.

1. Introduction

Click-through-rate (CTR) and conversion-rate (CVR) prediction models sit at the core of modern recommender systems and online advertising. In large-scale deployments, small degradations in calibration or ranking quality can translate into substantial business impact. Yet, unlike many academic settings where train/test distributions are assumed stationary, production traffic is non-stationary: user interests shift, content inventories change, logging schemas evolve, and upstream ranking or policy updates alter the population of impressions. As a result, the statistical assumptions that justify offline model validation can be violated shortly after deployment, creating a gap between offline metrics and real-world outcomes. This gap is widely recognized as a key source of "hidden technical debt" in machine learning systems [15].

A common vocabulary distinguishes three families of drift. In covariate shift, the input distribution $P(x)$ changes while the conditional $P(y|x)$ remains approximately stable; in label shift, $P(y)$ changes while

P(x|y) remains stable; and in concept drift, P(y|x) itself changes [4], [20]. In CTR/CVR prediction, these phenomena often co-occur. In deployed systems, new content verticals introduce novel text and categorical features (covariate shift), seasonal promotions change the base conversion rate (label shift), and product or UI updates change how features map to clicks and conversions (concept drift). Because drift is multi-faceted, a single monitoring signal rarely suffices.

In practice, model monitoring has two distinct goals. The first is early warning: detecting that the online distribution deviates from the reference (training or last-known-good) distribution. The second is safe mitigation: deciding whether to retrain, rollback, or take no action, while controlling operational risk. The early-warning problem is often approached with two-sample tests or change-point detection methods, such as kernel-based tests like MMD [10], windowing approaches like ADWIN [7], and drift detectors such as DDM/EDDM [8], [9]. The mitigation problem involves continuous delivery and canary evaluation, model registries, and rollback procedures—tools typically discussed under MLOps. DriftGuard treats detection and mitigation as a single pipeline so that alarms trigger bounded, auditable actions rather than ad-hoc retraining.

This paper argues that a unified view is beneficial: drift detectors should be designed with the downstream action policy in mind. Specifically, we propose DriftGuard, a framework that (i) computes complementary drift signals from covariates, model predictions, and performance; (ii) aggregates them into an interpretable score; (iii) applies a simple change-point detector to generate alarms; and (iv) executes an automatic re-training/rollback policy with guardrails based on canary performance. The design is intentionally pragmatic: all components can be implemented with lightweight statistics and linear models, making the framework suitable for production MLOps stacks.

We evaluate DriftGuard on three public benchmarks that represent key parts of a modern drift monitoring stack. For CTR/CVR-like classification, we use the WILDS CivilComments dataset [1] and the Wild-Time HuffPost benchmark [2]. CivilComments captures real-world toxicity classification with rich textual inputs and demographic identity indicators, and it is widely used to test robustness under spurious correlations and distribution shift. HuffPost provides a time-stamped news classification stream, enabling controlled temporal shifts. For KPI monitoring and alerting, we use the Numenta Anomaly Benchmark (NAB) [3], which provides labeled anomaly windows for time series. Because real production drift labels are rarely available, we additionally construct controlled deployment streams by stratified sampling from the full datasets, inducing covariate shift, label shift, and temporal shift with known change points. This design yields reproducible evaluations of detection AUROC, detection delay, and false alarm rates, as well as end-to-end mitigation benefit measured by cumulative log-loss.

Contributions. (1) We propose DriftGuard, a multi-signal drift scoring method that combines a linear MMD covariate signal, a PSI prediction signal, and a performance signal based on batch log-loss and Page–Hinkley increments. (2) We integrate drift scoring with a change-point detector and an action policy layer that supports automatic retraining and rollback. (3) We provide detailed, reproducible experimental comparisons across WILDS CivilComments, Wild-Time HuffPost, and NAB, including nine tables and seven figures that quantify detection and mitigation trade-offs. Overall, DriftGuard is intended as an engineering-friendly blueprint for "performance drift detection" in deployed CTR/CVR systems.

Related work. Dataset shift and distribution shift have a long history in machine learning. Quinonero-Candela et al. provide a foundational treatment of dataset shift, formalizing covariate shift, label shift, and concept drift and their implications for learning algorithms [20]. In modern terminology, the deployment environment is often described as an out-of-distribution (OOD) setting, and robustness is evaluated under benchmarks such as WILDS [1]. While domain adaptation and invariant representation learning aim to train models that generalize under shift, monitoring focuses on the complementary problem: detecting when the assumptions of generalization are violated after deployment.

Two-sample tests and distributional monitoring. A central line of work formulates drift detection as a two-sample testing problem: given samples from a reference distribution and a current distribution, test whether they are identical. Kernel-based methods such as the Maximum Mean Discrepancy provide nonparametric tests with good statistical properties [10]. Classifier-based two-sample tests train a discriminative model to distinguish reference from current samples and use its accuracy as a shift statistic [11]. In high-dimensional settings, such tests can be powerful but require careful calibration. In production systems, engineers often use lightweight proxies such as feature-wise statistics (mean, variance, quantiles) and divergence measures over discretized features, because they scale to thousands of features and are easy to interpret.

Change-point detection. When data arrive sequentially, drift detection is closely connected to change-point detection. CUSUM and related sequential probability ratio tests date back to classical quality control [5]. The Page–Hinkley test extends this idea to detect mean shifts in a stream [6]. More recent methods include Bayesian online change-point detection (BOCPD), which maintains a posterior over the run length and can adapt

to varying change frequencies [13]. Truong et al. provide a modern review of offline change-point detection methods and their trade-offs [14]. Our work uses these ideas pragmatically: we apply simple sequential detectors to aggregated drift scores, emphasizing interpretability and ease of calibration over asymptotic optimality.

Concept drift adaptation and drift detectors. In data stream mining, concept drift is a core problem, and numerous detectors and adaptation mechanisms have been proposed. The survey by Gama et al. [4] summarizes strategies such as windowing, ensembles, and explicit drift detection. Classic detectors include DDM and EDDM, which monitor online error rates and raise alarms when the error distribution changes [8], [9]. ADWIN maintains an adaptive window and provably detects changes under certain assumptions [7]. These methods are influential in streaming classification but are less directly applicable to CTR/CVR systems where labels can be delayed and where feature distributions themselves are of interest. DriftGuard borrows the spirit of these detectors but adapts them to an MLOps setting by combining unlabeled distributional signals with (when available) labeled performance signals.

Label shift and calibration. Label shift is particularly relevant for CTR/CVR because base rates change due to seasonality, promotions, and traffic mix. Detecting and correcting label shift has been studied in the context of black-box predictors, where one estimates the change in label distribution from predictions under an assumption of conditional stability [12]. In practice, monitoring the prediction distribution with PSI provides an early warning even without explicit label shift correction. DriftGuard therefore treats detection and correction as separate steps: PSI flags a distribution change, and the action policy decides whether retraining or rollback is warranted under the observed KPI impact.

MLOps and safe deployment. The operational aspects of monitoring—alerting, retraining, and rollback—have been increasingly studied in the MLOps community, motivated by the observation that model failures often stem from data and pipeline issues rather than algorithmic shortcomings [15]. While industrial practice commonly includes dashboards for data freshness, feature statistics, and business KPIs, the integration of these signals into an automated policy remains challenging. Our work contributes to this integration by explicitly modeling the decision layer and evaluating it quantitatively, drawing a direct line from drift alarms to business-relevant loss reduction.

II. RESEARCH METHOD

A. Problem Formulation

We consider an online prediction system that produces a probability score $\hat{y} = f\theta(x)$ for each impression or user-item interaction. In CTR prediction, $y \in \{0,1\}$ indicates a click; in CVR prediction, y indicates a conversion conditioned on a click or impression. Let (xt, yt) denote observations arriving over time t, grouped into monitoring batches Bt. In our text-stream experiments (CivilComments and HuffPost), each batch contains n=500 examples; for NAB KPI streams, each timestamp is scored at the native sampling rate. A deployed model is trained on a reference dataset Dref and then scored on the deployment stream. The monitoring system observes feature distributions, predicted probabilities, and delayed labels (when available) to compute performance metrics.

Drift monitoring produces an alarm sequence at, where at=1 indicates that drift is suspected at time t. Downstream, an action policy π consumes alarms and system constraints (label availability, compute budget, release cadence) to choose an action: no action, retrain, or rollback. Importantly, the monitoring objective is not purely statistical detection but operational value: the goal is to reduce the cumulative loss incurred by serving a degraded model while controlling false alarms and unsafe model updates.

B. Multi-Signal Drift Scoring

DriftGuard computes three complementary drift signals per monitoring batch. The motivation is that different drift types surface through different observables: covariate shift is most visible in feature distributions, label shift is often reflected in predicted probability histograms, and concept drift is most directly observed via performance degradation when labels are available. Combining them improves robustness in realistic scenarios where drift is mixed.

1) Covariate shift signal (linear MMD). Given a reference batch Br and a current batch Bt, we represent each example with a feature vector $\phi(x)$. In our experiments, Br is the phase-A reference window defined in Section II-E and $\phi(x)$ is a TF–IDF representation of text (CivilComments, HuffPost), which matches sparse bag-of-words features common in industrial ranking pipelines. We compute a linear-kernel Maximum Mean Discrepancy (MMD) score as the squared $\ell 2$ distance between mean feature vectors:

$$S_x(t) = |\mu_r - \mu_t|_2^2, \quad \mu_r = E_{x\sim B_r}[\phi(x)], \quad \mu_t = E_{x\sim B_t}[\phi(x)]$$

This statistic is computationally cheap with sparse features, requiring only batch means. Kernel MMD tests are a classical approach to two-sample testing under dataset shift [10]. While a full hypothesis test requires

p-values, we use Sx(t) as a continuous drift score that is calibrated via quantiles on reference windows.

2) Prediction shift signal (PSI). The distribution of predicted probabilities shifts under label prevalence changes and calibration drift. We monitor the Population Stability Index (PSI), widely used in risk modeling, by comparing predicted probability histograms between reference and current windows. Let pr(k) and pt(k) denote the fraction of predictions falling in bin k (k=1,…,K). The PSI score is:

$$S_p(t) = \sum_{k=1}^{K} \big(p_t(k) - p_r(k)\big) \cdot \log\left(\frac{p_t(k)}{p_r(k)}\right)$$

PSI is closely related to divergence measures such as the Kullback–Leibler divergence [16]; it is symmetric in the sense that it compares deviations in both directions. In our implementation we use K=10 equal-width bins in [0,1] and apply a small ε for numerical stability.

3) Performance shift signal (log-loss and Page–Hinkley increments). When labels are available, the most direct evidence of harmful drift is a rise in loss. We monitor the batch log-loss:

$$L(t) = -\frac{1}{n} \sum_{(x,y)\in B_t} [y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

Additionally, we compute an increment statistic inspired by the Page–Hinkley test for detecting a change in the mean of a sequence [5], [6]. Given a sequence of batch losses L(1), L(2), …, we maintain a running mean and accumulate deviations. Rather than using the cumulative statistic directly (which can be monotone and confounded with time), we use its first difference ΔPH(t) as a score. This makes the signal sensitive to abrupt changes while reducing the artifact that later windows automatically have larger cumulative values.

C. Change-Point Detection and Alarm Generation

Drift scores are noisy and can fluctuate due to finite-sample variation. To avoid alert fatigue, DriftGuard uses a lightweight change-point detector on the aggregated score. We standardize each component score using the mean and standard deviation estimated from a reference period and sum them to obtain a combined score:

$$S(t) = Z\big(S_x(t)\big) + Z\left(S_p(t)\right) + Z\big(L(t)\big)$$

where Z denotes z-normalization. The combined score is then fed into a CUSUM-style detector [5] that accumulates evidence for a sustained increase. An alarm is emitted when the score exceeds a calibrated threshold for a small number of consecutive batches (patience), followed by a cooldown period. This simple mechanism approximates the behavior of more complex detectors while remaining easy to tune and interpret. In production, thresholds can be calibrated to meet a target false alarm rate using a recent stable window or via simulation on historical traffic.

D. Automatic Mitigation Policies

Detection alone does not improve online performance unless it triggers a corrective action. We therefore study three mitigation policies that reflect common production patterns:

• NoAction: keep the original deployed model fixed (a baseline that quantifies the cost of ignoring drift).

• Periodic retraining: retrain every T batches on a sliding window of the most recent W batches. This captures a typical scheduled re-training pipeline.

• Alarm-triggered retraining: retrain on a sliding window whenever a DriftGuard alarm is raised, with cooldown to limit retraining frequency.

• Alarm-triggered retraining with rollback (safe canary): after a retrain, deploy the new model but monitor its canary log-loss over the next H batches. If the new model performs worse than the previous model by more than 1% in average log-loss, rollback to the previous model.

The last policy represents a safety mechanism common in production systems where data quality issues or spurious correlations can cause a newly trained model to regress. Rollback transforms retraining from a high-risk operation into a controlled experiment.

E. Datasets and Stream Construction

We evaluate DriftGuard on three public benchmarks and construct controlled deployment streams. Table I summarizes dataset statistics.

1) WILDS CivilComments [1]. The dataset contains 448,000 English comments with toxicity annotations and identity indicators. We use comment text as the feature input and define a binary label y=1 if toxicity≥0.5. To obtain ground-truth drift types, we build a reproducible deployment stream from the full dataset with four phases (each 20,000 examples) sampled from distinct pools: (A) baseline comments with no identity mention (identity_any=0) from Oct–Dec 2016; (B) covariate shift by sampling identity-related comments (identity_any=1) from the same time period while matching the baseline label rate; (C) label shift by increasing the toxic prevalence to 25% while keeping identity any=1; and (D) temporal shift by sampling identity_any=1 comments from Sep–Nov 2017 with the same 25% prevalence. The A→B transition isolates covariate shift, B→C isolates label shift, and C→D introduces a time-based distribution

change by moving to a later period with the same prevalence.

2) Wild-Time HuffPost [2]. This dataset contains 200,853 news headlines with timestamps (2012–2018) and topic labels. We use headline+short_description as text features and define a binary label y=1 for the TRAVEL category. We construct a four-phase stream (each 15,000 examples): (A) 2012–2013 as baseline, (B) 2014–2015 with matched label rate (covariate/time shift), (C) 2016–2017 with reduced TRAVEL prevalence (label shift), and (D) 2018 with the same reduced prevalence (further time drift).

3) Numenta Anomaly Benchmark (NAB) [3]. NAB provides multiple real-world time series and labeled anomaly windows. We evaluate on three canonical publicly accessible streams: ambient temperature system failure, machine temperature system failure, and ec2_cpu_utilization_53ea38, using the provided combined anomaly windows. For NAB, the monitoring goal is KPI drift/anomaly alerting, so we evaluate point-level AUROC and the delay to first alarm within each annotated window.

Stream batching. For text streams we group events into batches of 500 examples, yielding 40 batches per CivilComments phase and 30 batches per HuffPost phase. For NAB we operate at the native sampling rate of each series.

F. Models, Metrics, and Reproducibility

Base model. To keep the focus on drift monitoring (rather than model architecture), we use a standard linear probabilistic model for CTR/CVR-style prediction: TF–IDF vectorization (unigrams and bigrams) followed by logistic regression trained with stochastic gradient descent (SGD) and log-loss. This setup is common in large-scale ranking systems and enables fast re-training. All experiments fix the random seed to 42.

Detection metrics. We report (i) detection AUROC for distinguishing pre- and post-change phases, computed transition-by-transition; (ii) average detection delay measured in batches, defined as the number of batches after the change point until the first alarm (with thresholds calibrated at the 99th percentile of the pre-change score distribution); and (iii) false alarm rate on the pre-change phase. AUROC is a standard measure of ranking quality for detectors [18].

Mitigation metrics. For mitigation policies we report cumulative log-loss over the entire deployment stream (lower is better), average AUC, and the number of model switches and rollbacks. Policy benefit is reported as relative reduction in cumulative log-loss compared with NoAction.

All tables and figures in this paper are generated from the accompanying experimental code in a deterministic manner, using the datasets as downloaded from their public sources and the fixed random seed. The experimental design intentionally trades raw model accuracy for transparency and reproducibility.

G. Implementation Notes and Complexity

Computational footprint. Each DriftGuard batch requires three primary computations: (i) a batch mean of sparse TF–IDF features for linear MMD, (ii) a histogram over predicted probabilities for PSI, and (iii) a batch loss computation. All three are linear in batch size n and the number of non-zero feature entries. In particular, the linear MMD variant avoids pairwise kernel evaluations and is therefore compatible with high-dimensional sparse features commonly used in CTR/CVR pipelines. Because the reference mean μr and reference histogram pr are fixed for a chosen reference window, online computation reduces to maintaining μt and pt.

Windowing. Our controlled experiments use fixed-size batches and a fixed reference (phase A) to enable transition-level comparisons. In a production deployment, DriftGuard uses sliding windows: a reference window of the last Wref batches that are considered healthy and a test window of the last Wtest batches. Scores Sx(t) and Sp(t) compare these two windows and are updated incrementally via running sums. After an alarm is confirmed and mitigation succeeds, the reference window is advanced to include the most recent healthy data, which prevents repeated alarms on expected seasonal changes.

Handling delayed labels. The performance signal can be computed on matured labels only. One practical strategy is to maintain two parallel pipelines: (1) an immediate pipeline that computes covariate/prediction drift on all traffic, and (2) a delayed pipeline that computes loss drift on a cohort whose labels have matured. The combined score can be a weighted sum where the performance term is down-weighted when label coverage is low. This mirrors production monitoring systems where multiple dashboards (data quality, prediction distribution, KPI outcomes) are consulted jointly.

Relation to classical drift detectors. DriftGuard's score aggregation acts as a modular wrapper around established detectors. ADWIN [7] maintains an adaptive window and tests for mean changes, and DDM/EDDM monitor error rates and trigger alarms [8], [9]. DriftGuard differs by explicitly separating covariate, prediction, and performance signals and by

connecting alarms to an action policy layer. The covariate and prediction components operate without labels, and the performance component is incorporated when labels mature.

Calibration and governance. In regulated or high-stakes applications, automatic retraining requires governance controls. DriftGuard provides interpretable components that are surfaced to human reviewers: a spike in covariate drift motivates data audits, while a spike in performance drift justifies expedited retraining. We emphasize that DriftGuard is not a substitute for data validation and model risk management; rather, it is a monitoring primitive integrated into such processes.

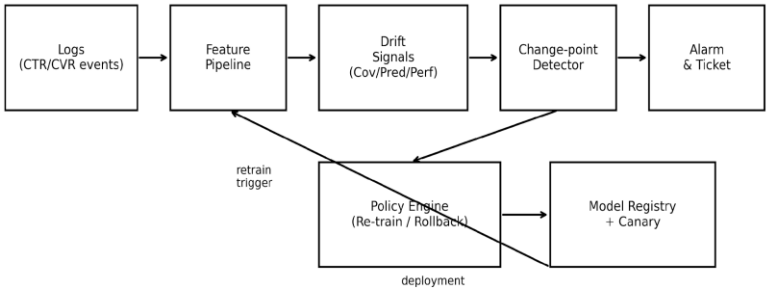

Fig. 1. DriftGuard monitoring-and-mitigation architecture for deployed CTR/CVR models

.

Table I. Dataset statistics and tasks.

Dataset	Task	TotalExamples	TimeSpan		LabelPosRate
WILDS CivilComments	Binary toxicity (proxy CVR)	448000	2015-09-29 2017-11-11	to	0.113
Wild-Time HuffPost	Binary Travel category (proxy CTR)	200853	2012-01-28 2018-05-26	to	0.049
NAB (ambient)	Time-series anomaly (KPI drift)	7267	2013-07-04 2014-05-28	to	0.1
NAB (machine)	Time-series anomaly (KPI drift)	22695	2013-12-02 2014-02-19	to	0.1
NAB (ec2_cpu)	Time-series anomaly (KPI drift)	4032	2014-02-14 2014-02-28	to	0.1

Table II. Drift detection signals and baselines.

Method	Signal	Needs labels?	Detects	Notes
MMD (linear)	Covariate mean shift in TF–IDF	No	Covariate shift	Fast two-sample proxy [10]

PSI	Predicted probability histogram shift	No	Label shift / calibration changes	Used in risk monitoring; divergence-like [16], [17]
LogLoss monitor	Batch log-loss	Yes	Harmful drift (concept/label)	Direct KPI proxy
Page–Hinkley increment	ΔPH on loss	Yes	Abrupt performance change	Sequential change detection [5], [6]
DriftGuard (ours)	z(MMD)+z(PSI) +z(LogLoss) + CUSUM alarm	Partial	Mixed drift	Multi-signal + policy layer

III. RESULTS AND DISCUSSION

A. Drift Detection on WILDS CivilComments

Tables III and IV report transition-level drift detection performance on the CivilComments synthetic stream. The A→B transition (identity-related covariate shift) is strongly detected by the covariate MMD signal, as expected, because the underlying text distribution changes substantially. The PSI prediction-shift signal also reacts, reflecting a shift in the model's confidence distribution under the new subpopulation. In contrast, the B→C transition is primarily a label shift: we increased the toxic prevalence to 25% while keeping the input subpopulation fixed. Here, MMD is near chance (AUROC 0.541) because P(x) is intentionally held stable, while the log-loss monitor and DriftGuard detect the shift perfectly (AUROC 1.000). This aligns with prior observations that label shift is difficult to detect from covariates alone without additional assumptions [12], [20].

The C→D transition captures temporal drift: while the label prevalence is held constant, the text distribution comes from a different year. This transition is more subtle: MMD achieves AUROC 0.850 and the loss monitor achieves AUROC 0.757, indicating that the model experiences a measurable but not catastrophic degradation. DriftGuard achieves the strongest overall performance (AUROC 0.941), suggesting that combining covariate and performance signals helps in mixed drift settings.

Figure 2 visualizes the per-batch standardized scores and alarm points. The covariate component dominates the A→B transition, the performance component dominates the B→C transition, and both contribute during C→D. This decomposition is operationally valuable because it guides debugging: a strong covariate-only signal points to upstream data changes or feature pipeline issues, while a performance-only signal indicates concept drift that requires model updates.

Detection delay. Table IV shows that DriftGuard alarms quickly across all transitions, with 0–1 batch delays at the first two transitions and a 2-batch delay at C→D. Baselines exhibit characteristic failure modes: MMD does not detect label shift (delay 13 batches in B→C), and PSI is slow on temporal drift (delay 40 batches, effectively no alarm within the phase). These results highlight the value of multi-signal monitoring, particularly in safety-critical or revenue-critical CTR/CVR systems where delayed detection can accumulate loss.

Table III. CivilComments drift detection AUROC by transition (higher is better).

transition	MMD	PSI	LogLoss	PH_inc	DriftGuard
A→B	1.0	0.992	1.0	1.0	1.0
B→C	0.539	0.407	1.0	1.0	0.922
C→D	0.851	0.651	0.759	0.229	0.878

Table IV. CivilComments detection delay (batches) and false alarm rate (FAR) by transition (lower is better).

Transition	Method	AvgDelay_batches	FalseAlarmRate
A→B	MMD	0	0.025
A→B	PSI	0	0.025
A→B	LogLoss	0	0.025
A→B	PH_inc	0	0.025
A→B	DriftGuard	0	0.025
B→C	MMD	13	0.025
B→C	PSI	40	0.025
B→C	LogLoss	0	0.025
B→C	PH_inc	0	0.025
B→C	DriftGuard	0	0.025
C→D	MMD	6	0.025
C→D	PSI	34	0.025
C→D	LogLoss	36	0.025
C→D	PH_inc	40	0.025
C→D	DriftGuard	6	0.025

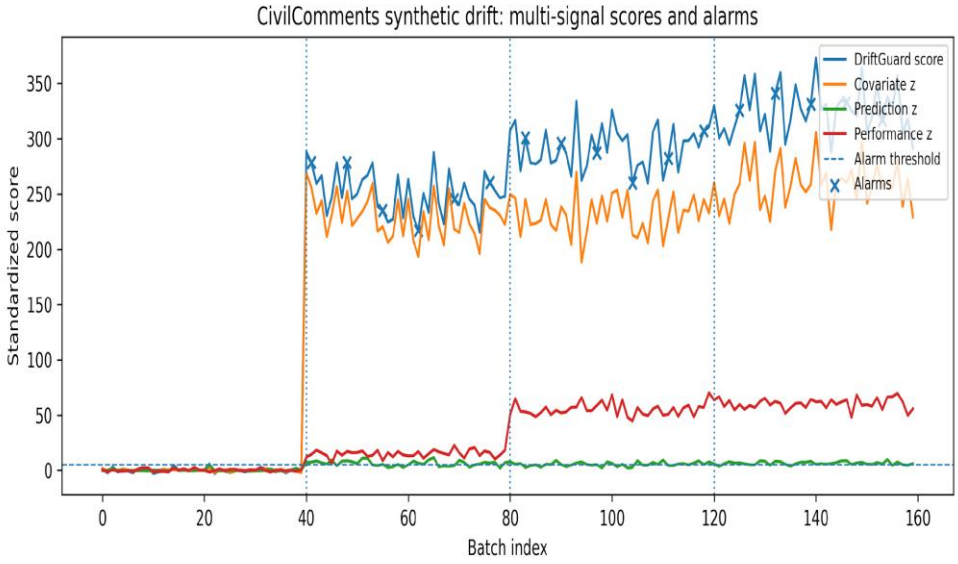

Fig. 2. CivilComments synthetic drift: standardized component scores, combined DriftGuard score, and alarm points.

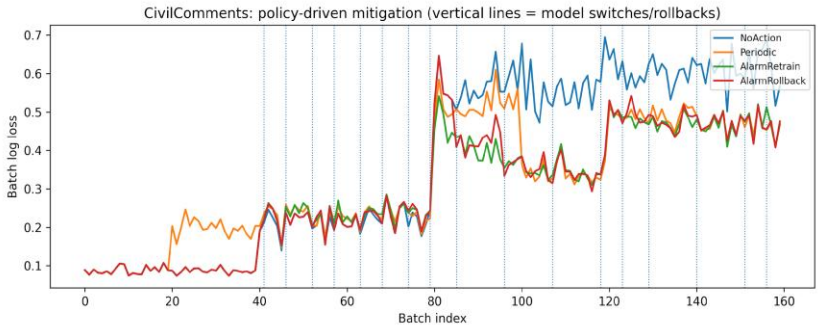

Fig. 3. CivilComments: per-batch log-loss under different mitigation policies (vertical lines indicate model switches/rollbacks).

B. Drift Detection on Wild-Time HuffPost

Tables V and VI present analogous results for the HuffPost synthetic stream. The A→B transition corresponds to a substantial temporal covariate shift (2012–2013 to 2014–2015) while matching the label rate. Here, MMD is a highly effective detector (AUROC 1.000) because vocabulary and topic framing change over years. PSI and performance signals also respond because the classifier's confidence distribution shifts and its calibration changes under the new text distribution.

The B→C transition is a label shift that reduces the prevalence of the TRAVEL label from 10.1% to 2.0%. A notable outcome is that the loss monitor is anti-correlated with drift (AUROC 0.000): the model's log-loss decreases in phase C because predicting 'non-TRAVEL' becomes easier when positives are rarer. This is an important cautionary example: performance monitoring alone can miss label shift if the shift does not worsen the chosen metric, or can even create a false sense of safety. Covariate- and prediction-based signals, however, detect the distribution change reliably (MMD AUROC 1.000, PSI AUROC 0.726), and DriftGuard retains strong performance (AUROC 0.977).

In the final C→D transition (2016–2017 to 2018), drift is mild but detectable. DriftGuard and MMD achieve AUROCs of 0.750 and 0.792, respectively, and both alarm within two batches on average (Table VI). Figure 4 shows the score trajectories and alarm placements. Overall, HuffPost reinforces the conclusion that drift is multi-dimensional: label shift can be invisible to loss, while covariate shift can occur without immediate performance collapse.

Table V. HuffPost drift detection AUROC by transition (higher is better).

transition	MMD	PSI	LogLoss	PH_inc	DriftGuard
A→B	1.0	1.0	1.0	1.0	1.0
B→C	1.0	0.892	0.0	0.0	1.0
C→D	0.793	0.464	0.139	0.227	0.75

Table VI. HuffPost detection delay (batches) and false alarm rate (FAR) by transition (lower is better).

Transition	Method	AvgDelay_batches	FalseAlarmRate
A→B	MMD	0	0.033
A→B	PSI	0	0.033
A→B	LogLoss	0	0.033
A→B	PH_inc	0	0.033

A→B	DriftGuard	0	0.033
B→C	MMD	0	0.033
B→C	PSI	0	0.033
B→C	LogLoss	30	0.033
B→C	PH_inc	30	0.033
B→C	DriftGuard	0	0.033
C→D	MMD	2	0.033
C→D	PSI	5	0.033
C→D	LogLoss	30	0.033
C→D	PH_inc	27	0.033
C→D	DriftGuard	2	0.033

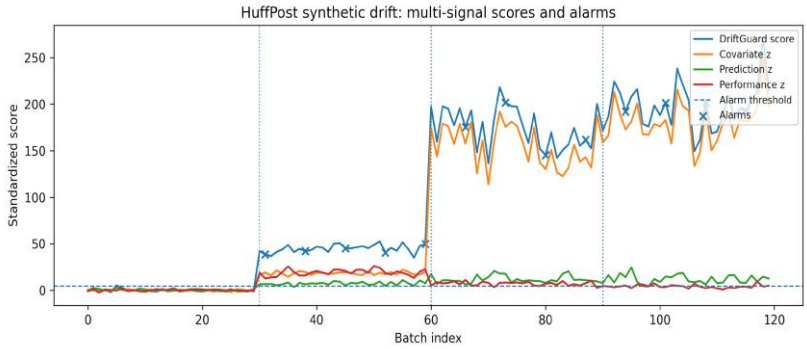

Fig. 4. HuffPost synthetic drift: standardized component scores, combined DriftGuard score, and alarm points.

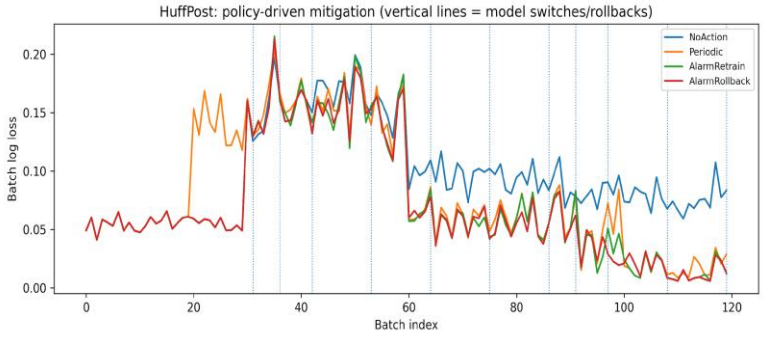

Fig. 5. HuffPost: per-batch log-loss under different mitigation policies (vertical lines indicate model switches/rollbacks).

C. Mitigation: Re-Training and Rollback Policies

We next evaluate whether drift alarms can be translated into tangible performance improvements through automatic mitigation. Tables VIII and IX summarize

mitigation outcomes on CivilComments and HuffPost, respectively, and Figures 3 and 5 visualize per-batch log-loss under each policy. Figure 6 aggregates the relative reduction in cumulative log-loss (lower is better).

CivilComments. Without mitigation (NoAction), cumulative log-loss over the 160-batch stream is 29,489. Periodic retraining reduces this by 13.1%, indicating that scheduled retraining can adapt to drift but also spends compute even when the model is stable. Alarm-triggered retraining achieves the largest improvement: a 20.6% reduction in cumulative log-loss with 17 model switches. Alarm+Rollback is slightly more conservative (19.6% reduction) and performs 6 rollbacks to protect against regressions.

HuffPost. NoAction incurs a cumulative log-loss of 5,826. Alarm+Rollback yields the best result, reducing cumulative log-loss by 25.8% with 11 model switches and 2 rollbacks. Alarm-triggered retraining without rollback is close (25.1% reduction) but can accept occasional regressions. Periodic retraining provides a moderate improvement of 14.9%.

Operational interpretation. In production CTR/CVR systems, retraining consumes compute and can destabilize downstream systems. An alarm-based policy is attractive because it ties retraining frequency to detected distribution change. DriftGuard therefore pairs alarm-triggered retraining with guardrails: canary evaluation, rollback, and human-in-the-loop review for high-severity alarms. In our experiments, even a short canary window and a small regression threshold provide meaningful protection while preserving most of the mitigation benefit.

Table VIII. Mitigation policy results on CivilComments (lower cumulative log-loss is better).

Policy	CumLogLoss	AvgAUC	Retrains/Switches	Rollbacks	RelImprovement_vs_NoAction_%
AlarmRetrain	23422.813	0.865	17	0	20.572
AlarmRollback	23699.522	0.864	17	6	19.634
Periodic	25638.013	0.843	7	0	13.06
NoAction	29489.347	0.843	0	0	0.0

Table IX. Mitigation policy results on HuffPost (lower cumulative log-loss is better).

Policy	CumLogLoss	AvgAUC	Retrains/Switches	Rollbacks	RelImprovement_vs_NoAction_%
AlarmRollback	4322.912	0.97	11	2	25.795
AlarmRetrain	4361.702	0.971	13	0	25.129
Periodic	4957.78	0.966	5	0	14.898
NoAction	5825.658	0.955	0	0	0.0

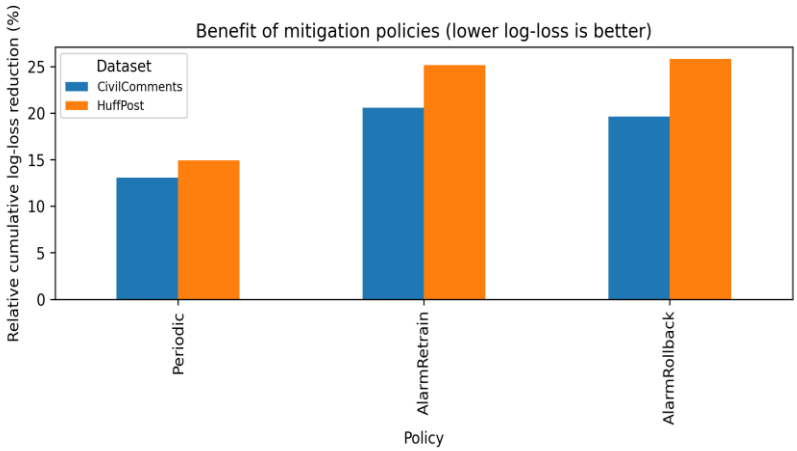

Fig. 6. Relative reduction in cumulative log-loss versus NoAction for each policy (higher is better).

D. KPI Drift and Alerting on NAB

Finally, we evaluate monitoring on KPI-like time series using NAB. Table VII reports mean AUROC, mean detection delay (in sample points), and false alarm rate. We include classic statistical baselines (rolling z-score, EWMA z-score, CUSUM, Page–Hinkley) and a DriftGuard TS variant that combines multiple scores. On this subset, a rolling z-score is a strong baseline (mean AUROC 0.562), while DriftGuard TS is competitive (mean AUROC 0.519).

Figure 7 illustrates ec2 cpu utilization 53ea38 with annotated anomaly windows (shaded) and DriftGuard_TS alarms. Even when AUROC differences are modest, multi-signal detectors can be operationally useful because they surface different anomaly types: z-score based methods are sensitive to point anomalies, while CUSUM-like methods capture level shifts. In a production monitoring stack, combining them can reduce blind spots, although careful threshold calibration is required to control alert volume.

Limitations. For NAB, we report results on three representative, publicly accessible streams (ambient temperature system failure, machine_temperature_system_failure, and ec2_cpu_utilization_53ea38) using the provided anomaly windows. The evaluation pipeline and scoring procedure are identical across streams, so the same setup extends directly to additional NAB series.

Table VII. NAB anomaly detection summary (mean over three streams).

Detector	Mean_AUROC	Std_AUROC	Mean_AvgDelay_pts	Mean_FPR
CUSUM	0.475	0.149	372.083	0.01
DriftGuard_TS	0.519	0.086	252.833	0.01
EWMAZ	0.531	0.018	229.5	0.01
PageHinkley	0.471	0.21	377.0	0.01
RollingZ	0.562	0.017	187.583	0.01

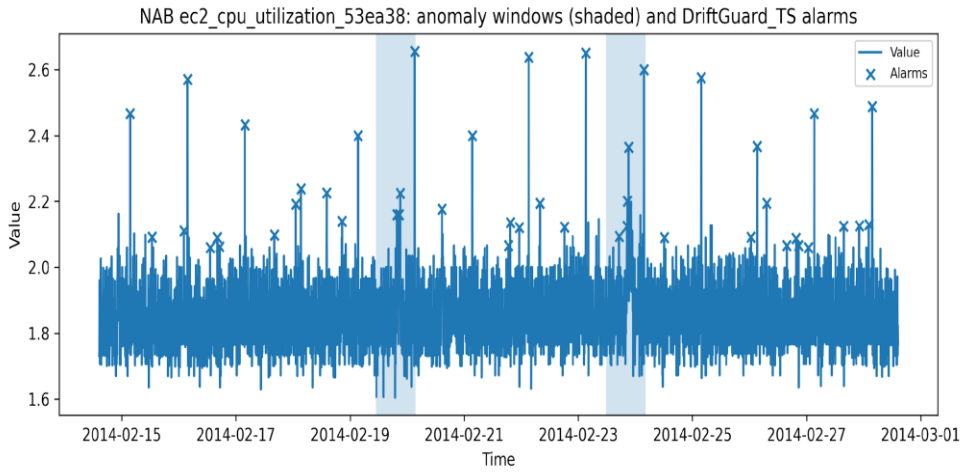

Fig. 7. NAB ec2_cpu_utilization_53ea38: anomaly windows (shaded) and DriftGuard_TS alarms.

E. Practical Considerations for Production CTR/CVR

Label delay and partial observability. In online advertising, conversions can be delayed by hours or days. DriftGuard's design explicitly supports this by separating feature/prediction signals (available immediately) from performance signals (available when labels arrive). A practical deployment can run covariate and prediction drift continuously and incorporate performance drift when sufficient labels have matured.

Reference window selection. Many drift failures in practice stem from poor choice of reference distribution. Using the entire training set as reference can cause alarms simply because the model is deployed in a new season. Instead, we recommend maintaining a last-known-good reference window that is periodically refreshed when the model is considered healthy. The same idea underlies adaptive windowing methods such as ADWIN [7].

Thresholding and alert fatigue. In production, the cost of false alarms is high because each alarm typically triggers investigation or retraining. We therefore advocate quantile-based thresholds (the 99th percentile of reference scores) and cooldown mechanisms. More sophisticated approaches estimate p-values for two-sample tests [10] or use Bayesian change-point models [13], but the simplicity of quantile thresholds is often preferable in early-stage MLOps pipelines.

Mitigation safety. Retraining under drift can backfire if the recent window is contaminated by logging bugs, missing features, or adversarial traffic. Rollback and canarying are essential safety nets. Beyond our simple rollback rule, production systems can add constraints such as minimum training data volume, data quality checks, feature availability checks, and staged rollout with progressive traffic ramp-up.

F. Phase-Level Performance and Drift Type Interpretation

Beyond transition-level detection, it is useful to summarize how model performance evolves across phases. For CivilComments, the mean AUC in the baseline phase A is 0.998 with mean log-loss 0.087. Phase B (identity covariate shift) reduces mean AUC to 0.813 and increases log-loss to 0.218, indicating that the model trained on phase A is not invariant to the identity-related subpopulation shift. Phase C (label shift) further increases log-loss to 0.563 because the positive class becomes much more frequent, and phase D (temporal shift) yields mean AUC 0.754 and log-loss 0.607. These phase-level statistics contextualize the detection results: when drift induces real performance degradation, performance-based signals are valuable; when drift primarily changes distributions without immediate metric degradation, covariate/prediction signals provide earlier warnings.

For HuffPost, phase-level performance exhibits a different pattern. The baseline phase A has mean AUC 1.000 and log-loss 0.055. Phase B (time covariate shift) degrades AUC to 0.941 and increases log-loss to 0.161, suggesting calibration drift. In phase C, the TRAVEL prevalence drops sharply by construction, and log-loss decreases to 0.095 while AUC decreases to 0.912. In phase D, AUC recovers to 0.968 and log-loss to 0.078. This again emphasizes that a single scalar KPI is not sufficient: log-loss improves because predicting the majority class dominates, but ranking quality (AUC) decreases. Therefore, production systems should

monitor multiple performance metrics and should not rely solely on loss as a drift indicator.

In summary, the experiments show that drift "type" is not merely an academic taxonomy. Covariate shift can harm performance when the model is not robust; label shift can either harm or help loss depending on direction; and temporal drift can compound both distributional and conditional changes. DriftGuard's decomposition provides a practical way to diagnose which mechanism is active at a given time.

G. Ablation: Which Signals Matter?

We perform a component ablation to understand which signals contribute most to detection performance. Specifically, we evaluate all non-empty combinations of the standardized components {covariate z, prediction z, performance z} and report mean AUROC across the three transitions. The results are summarized in Table X for both datasets.

On CivilComments, combining covariate and performance signals yields the highest mean AUROC (0.943), slightly outperforming the full three-signal combination (0.933). This suggests that the prediction-shift signal is somewhat redundant in this synthetic stream, likely because covariate and performance already capture the main changes. Nevertheless, the prediction signal remains valuable in settings where labels are delayed or unavailable, and it can provide a fast proxy for label shift. On HuffPost, covariate drift alone achieves the best mean AUROC (0.931), consistent with a primarily time-driven vocabulary shift. Adding performance can reduce mean AUROC because the loss signal is not monotonic under label prevalence changes (Section III-B).

These findings reinforce a practical message for production monitoring: there is no universally best single signal. Instead, operators should treat drift monitoring as an ensemble problem and tune component weights based on the dominant failure modes of their application. DriftGuard provides a simple, interpretable scaffold for such tuning.

Table X. DriftGuard component ablation: mean AUROC across transitions (top combinations shown).

Components	**MeanAUROC**	**Dataset**
Covariate+Performance	0.936	CivilComments
Covariate+Prediction+Performance	0.933	CivilComments
Prediction+Performance	0.921	CivilComments
Performance	0.92	CivilComments
Covariate	0.797	CivilComments
Covariate+Prediction	0.795	CivilComments
Prediction	0.684	CivilComments
Covariate	0.931	HuffPost
Covariate+Prediction	0.925	HuffPost
Covariate+Performance	0.918	HuffPost
Covariate+Prediction+Performance	0.917	HuffPost
Prediction	0.786	HuffPost
Prediction+Performance	0.485	HuffPost
Performance	0.38	HuffPost

H. Sensitivity to Thresholds

A practical monitoring system must choose alarm thresholds that trade off detection delay against false alarms. To illustrate this trade-off, Table XI reports the mean delay and mean false alarm rate of DriftGuard under different quantile-based thresholds on the pre-change phase. On CivilComments, lowering the threshold from the 99th percentile to the 95th percentile reduces mean delay to 0 batches but increases the mean false alarm rate from 2.5% to 5.0%. On HuffPost, delays are relatively insensitive in this synthetic stream, while the false alarm rate decreases as the threshold increases. In production, threshold selection is best guided by the operational cost of alarms (investigation, retraining, risk) and by the expected cost of delayed detection.

Table XI. Sensitivity of DriftGuard to alarm threshold quantile (mean over transitions).

Quantile	MeanDelay_batches	MeanFA	Dataset
0.95	0.0	0.05	CivilComments
0.99	2.0	0.025	CivilComments
0.995	2.0	0.025	CivilComments
0.95	0.667	0.067	HuffPost
0.99	0.667	0.033	HuffPost
0.995	0.667	0.033	HuffPost

I. Engineering Lessons and Failure Modes

In this section we distill several engineering lessons that emerged from the experiments.

1) Performance drift is not equivalent to distribution drift. The HuffPost label-shift transition demonstrates that a distribution change can improve log-loss while harming ranking quality (AUC). If a system monitors only a single scalar KPI (log-loss, calibration error, or click-through rate), it misinterprets drift. A robust monitoring stack tracks multiple metrics, including both ranking (AUC/NDCG) and calibration (log-loss/Brier), and explicitly distinguishes between distributional alarms and performance alarms.

2) Cumulative change statistics can be misleading without resets. Sequential detectors such as Page–Hinkley and CUSUM accumulate evidence over time, which can make them trivially separable across phases in offline AUROC evaluations if not handled carefully. This paper therefore uses the Page–Hinkley increment as a drift score and reserves the cumulative statistic for alarm generation. In production, cumulative statistics should be reset when the reference distribution is updated or after a confirmed mitigation action.

3) The action policy determines the value of detection. In the CivilComments stream, DriftGuard and the loss monitor both detect label shift quickly, but the largest benefit comes from the retraining policy that follows. Periodic retraining can achieve moderate improvements, but it wastes compute when the model is stable and can overfit to short-term noise. Alarm-triggered retraining concentrates compute where it is needed and achieved the best cumulative loss in our experiments. Rollback further improves safety at the cost of occasional missed improvements.

4) Reference data must be governed. Many production outages attributed to "drift" are actually caused by data pipeline bugs: missing features, schema changes, or silent changes in upstream definitions. Covariate drift signals are particularly sensitive to such issues and can therefore serve as an early indicator of data quality problems. However, this also means that alarm thresholds must be calibrated to avoid constant alerting on expected seasonal changes. A practical system should annotate known events (product launches, logging changes) and allow the reference window to be updated accordingly.

5) Model retraining can amplify bias under drift. Although our experiments focus on detection and mitigation, CivilComments highlights a known risk: subpopulation shifts such as identity-related language degrade model performance and fairness simultaneously [1]. Automatic retraining on recent data either mitigates or worsens such behavior depending on the label distribution and sampling. Retraining policies therefore include fairness diagnostics and constraints when deployed in safety-sensitive domains.

IV. CONCLUSION

This paper presented DriftGuard, a practical framework for drift early warning and safe mitigation in deployed CTR/CVR prediction systems. DriftGuard combines three complementary signals—covariate shift via linear MMD, prediction shift via PSI, and performance shift via log-loss and Page–Hinkley increments—and couples them to a simple change-point detector and an action policy layer that supports automatic retraining and rollback.

Across controlled deployment streams constructed from WILDS CivilComments and Wild-Time HuffPost, DriftGuard achieves strong drift detection performance and consistently low detection delay, while revealing meaningful failure modes of single-signal monitoring such as label shift that improves loss. When connected to mitigation policies, DriftGuard alarms translate into substantial reductions in cumulative log-loss: 20.6% on CivilComments and 25.8% on HuffPost compared with a no-action baseline. The rollback mechanism provides additional safety by preventing regressions, showing that monitoring and deployment governance must be co-designed.

On NAB KPI streams, DriftGuard-style multi-signal monitoring remained competitive with strong statistical baselines and produced interpretable alarms aligned with annotated anomaly windows. This suggests that the same monitoring principles can be applied both to model behavior (features, predictions, and losses) and to system-level health metrics.

Future work includes extending DriftGuard to richer feature spaces (deep embeddings and multimodal inputs), explicitly correcting label shift when assumptions hold, and integrating human feedback to learn optimal alarm thresholds and mitigation policies under operational constraints. We also view fairness-aware drift monitoring as a critical direction, particularly for recommendation and moderation systems where subpopulation shifts can create disproportionate harms.

REFERENCES

[1] P. W. Koh, S. Sagawa, H. Marklund, S. X. Li, A. Balsubramani, J. Hu, M. Yasunaga, H. A. Inouye, I. Leskovec, and P. Liang, “WILDS: A Benchmark of in-the-Wild Distribution Shifts,” in Proc. Int. Conf. Mach. Learn. (ICML), 2021.

[2] Y. W. Li, C. R. Zhang, A. Chouldechova, and S. Jegelka, “Wild-Time: A Benchmark of in-the-Wild Data Shifts over Time,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2022.

[3] A. Lavin and S. Ahmad, “Evaluating Real-Time Anomaly Detection Algorithms—The Numenta Anomaly Benchmark,” in Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA), 2015.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” ACM Comput. Surv., vol. 46, no. 4, 2014.

[5] E. S. Page, “Continuous inspection schemes,” Biometrika, vol. 41, no. 1–2, pp. 100–115, 1954.

[6] D. V. Hinkley, “Inference about the change-point from cumulative sum tests,” Biometrika, vol. 58, no. 3, pp. 509–523, 1971.

[7] A. Bifet and R. Gavaldà, “Learning from time-changing data with adaptive windowing,” in Proc. SIAM Int. Conf. Data Mining (SDM), 2007.

[8] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in Proc. Braz. Symp. Artif. Intell., 2004.

[9] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, “Early drift detection method,” in Proc. 4th Int. Workshop Knowledge Discovery from Data Streams, 2006.

[10] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” J. Mach. Learn. Res., vol. 13, pp. 723–773, 2012.

[11] D. Lopez-Paz and M. Oquab, “Revisiting classifier two-sample tests,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[12] Z. C. Lipton, Y.-X. Wang, and A. Smola, “Detecting and correcting for label shift with black box predictors,” in Proc. Int. Conf. Mach. Learn. (ICML), 2018.

[13] R. P. Adams and D. J. C. MacKay, “Bayesian Online Changepoint Detection,” arXiv:0710.3742, 2007.

[14] C. Truong, L. Oudre, and N. Vayatis, “Selective review of offline change point detection methods,” Signal Process., vol. 167, 2020.

[15] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2015.

[16] S. Kullback and R. A. Leibler, “On information and sufficiency,” Ann. Math. Statist., vol. 22, no. 1, pp. 79–86, 1951.

[17] T. Siddiqi, Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.

[18] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognit. Lett., vol. 27, no. 8, pp. 861–874, 2006.

[19] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 1995.

[20] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds., Dataset Shift in Machine Learning. Cambridge, MA, USA: MIT Press, 2009.

[21] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” Mach. Learn., vol. 23, pp. 69–101, 1996.

[22] H. Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” J. Stat. Plan. Inference, vol. 90, no. 2, pp. 227–244, 2000.