KFUPM
Journal of Undergraduate Research International

Diamond Open Access 45

Article Computer Science

Benchmarking Drift-Resilient Anomaly Detection in Streaming Industrial Data: A Case Study on Turbofan Engine Failures

Rakan M. AlKhulaif

Cite https://doi.org/10.64589/juri/214614

Submitted: June 04, 2025 **Revised:** October 26, 2025 **Accepted:** November 23, 2025

ABSTRACT

Concept drift undermines the reliability of unsupervised anomaly detection in safety-critical systems, in which sensor distributions evolve over time. Using a modified version of NASA's C-MAPSS FD001 turbofan dataset, this study benchmarks the drift robustness of two widely used detectors (Isolation Forest and One-Class SVM) alongside a streaming change detector (ADWIN). To emulate gradual system degradation, we injected 5% synthetic anomalies across sensors 1–5 and applied a multiplicative drift (1.0→1.5) from cycles 80 to 120. All models were trained on pre-drift data (<cycle 80) using min-max normalization and evaluated across the pre-, mid-, and post-drift phases. The isolation forest achieved an F1 score of 0.857 pre-drift but degraded to 0.140 mid-drift and 0.080 post-drift; retraining at cycle 120 failed to recover the performance. One-Class SVM exhibited weak detection throughout (pre-drift F1 = 0.364; post-drift = 0.080). Meanwhile ADWIN, used naïvely as an anomaly signal on aggregate sensor values, returned F1 = 0 throughout. These findings illustrate that static detectors collapse under moderate drift, simple retraining is ineffective when drift co-occurs with anomalies, and drift signals alone do not reliably flag anomalies. We released our simulation protocol and codebase to support future work on drift-resilient anomaly detection for streaming safety-critical environments.

Keywords: concept drift, anomaly detection, streaming data, turbofan engines, predictive maintenance

1. INTRODUCTION

Industrial cyber-physical systems (CPS), including aerospace turbofans, power-generation turbines, and manufacturing lines, increasingly rely on continuous sensor monitoring for predictive maintenance [1]. Unsupervised anomaly detectors are widely adopted because they require no labelled failure data and can theoretically identify novel fault signatures. However, these methods typically assume a stationary data distribution [2,3]. In practical deployments, sensor readings evolve over time because of component ageing, environmental conditions, or maintenance actions: a phenomenon known as concept drift [2,4]. When static detection boundaries are applied to drifting data, reliability plummets, resulting in missed fault signals or a proliferation of false alarms. Thus ensuring robust anomaly detection during drift is critical for maintaining system integrity and safety [1,5,6].

Although concept drift has been studied extensively in the context of classification tasks, there exists a marked paucity of systematic benchmarks for evaluating unsupervised anomaly detectors under realistic drift scenarios. Prior work often relies on synthetic benchmarks or focuses exclusively on classification accuracy. In the domain of anomaly detection for CPS, few studies examine the interplay between drift and rare fault occurrences, particularly in continuous, streaming settings. As a result, there is limited guidance on whether static detectors suffice, how quickly adaptive schemes respond, or how retraining strategies should be orchestrated [2,4,7–10].

This study presents a reproducible pipeline to simulate gradual drift and evaluates three approaches: (i) use of static unsupervised detectors (Isolation Forest [11] and One-Class SVM [12]) trained solely on pre-drift data, (ii) the utilisation of streaming drift detector ADWIN [13] as a naive anomaly proxy, and (iii) scheduled retraining of static detectors post-drift. By employing NASA's C-MAPSS FD001 turbofan dataset [14], the aim is to quantify the robustness of each method under drifting sensor distributions in a safety-critical streaming context and provide a full codebase and drift protocol to facilitate replication and extension.

The remainder of this paper is organized as follows:

Section 2 reviews related work on concept drift and unsupervised anomaly detection.

Section 3 describes on the dataset, drift simulation, anomaly injection, and model configurations.

Section 4 presents experimental results, including F1 scores and recovery analyses.

Section 5 discusses implications, limitations, and potential future directions.

Section 6 concludes the paper.

2. RELATED WORK

2.1. Concept Drift in Streaming Environments. Concept drift refers to changes in the statistical properties of data streams over time, which can degrade model performance if unaddressed. Prior work has catagorized drift as abrupt, gradual, incremental, or recurring, each requiring distinct detection and adaptation strategies[7]. Early detectors such as DDM focused on classification error rates[7], while EDDM targeted slower, gradual changes[14]. ADWIN monitors distributional change via an adaptive window and adjusts its size using a statistical test to avoid manual window parameters[15]. Stream-based classification with drift detectors has been well-studied; however, analogous evaluations for unsupervised anomaly detection remain scarce[8,15–19].

2.2. Unsupervised Anomaly Detection Techniques. Isolation Forest identified anomalies by recursively partitioning data; outliers require fewer partitions. It scales well to high-dimensional data and is commonly used in industrial CPS. One-Class SVM constructs a boundary around normal data by maximising the margin from the origin in feature space; its performance is sensitive to kernel choice and parameters (e.g., ν, γ). Other techniques such as local outlier factor and clustering-based methods have been applied in static scenarios, but their drift resilience is under-explored[1,5,6,20–22].

2.3. Drift-Aware Anomaly Detection. Recent efforts have embedded drift awareness into anomaly detection using three recurring ideas. First, windowed retraining or accuracy-updated ensembles aim to recover quickly by refreshing models on short post-drift windows, at the cost of instability if the window is contaminated[7,8] Second, change-detector-gated policies use signals such as ADWIN to trigger adaptation; these signals indicate when to adapt rather than what is anomalous, so naïve use as labels risks systematic errors[15,17]. Third, hybrid or physics-informed scoring uses residuals from simple physical expectations as inputs to data-driven detectors, thereby improving the identifiability of genuine faults under shift[1]. Our contribution is a minimally controlled benchmark that isolates drift and anomaly factors and systematically compares (i) static detectors, (ii) the drift-signaled use of ADWIN, and (iii) scheduled retraining under the same streaming protocol. A summary of these adaptive strategies is provided in Table 1.

2.4. Turbofan Engine Prognostics and Anomaly Detection. NASA's C-MAPSS dataset[13] is a canonical benchmark for prognostic health management (PHM) and anomaly detection in aerospace engines. Several studies have focused on remaining useful life (RUL) estimation using supervised or semi-supervised deep learning. Anomaly detection work on C-MAPSS often uses autoencoders or probabilistic approaches but typically assumes a stationary distribution. To the best of our knowledge, no prior work has robustly simulated concept drift within C-MAPSS for unsupervised anomaly detector benchmarking. Therefore, we adopted a minimal, reproducible protocol that isolates drift and anomaly factors for a fair comparison in Sections 4–5.

3. METHODOLOGY

3.1. Dataset Description. We used NASA's C-MAPSS FD001 subset[13] containing multivariate time-series sensor data from 100 simulated turbofan engines. Each engine unit produces a sequence of observation cycles until failure. Per cycle, there are 21 sensor measurements and three operational settings. For simplicity, we focused the main analysis of Unit 1 and replicated the protocol in Units 2–3 (Section 4.5). We excluded three operational settings and used only the 21 sensor features.

3.2. Anomaly Injection. To emulate rare fault events, we injected synthetic anomalies into 5% of the data points. Let N be the total number of cycles in unit 1. We sampled indices without replacement to select anomalies of approximately N × 0.05 anomalies. We distributed the anomalies evenly: one-third of the anomalies in cycles <80 (pre-drift), one-third in cycles 80–120 (drift window), and one-third in cycles >120 (post-drift). At each selected index, we added Gaussian noise with a mean of 0.5 and standard deviation of 0.1 to sensors 1–5. These rows are markedwith an is_anomaly flag (1) leaving the remainder at zero. Figures 1–3 illustrate the sensor distributions in the pre-, mid-, and post-drift phases.

3.3. Concept Drift Simulation. We simulated gradual drift in sensors 1–5 over cycles 80–120 by applying a multiplicative factor.

$$m(c) = 1.0 + 0.0125 \cdot (c - 80), \quad 80 \leq c \leq 120 \tag{1}$$

Table 1. *Summary of adaptive strategies for non-stationary streams (from Related Work)*

Approach	Adaptation trigger	Labels needed	Typical pros	Typical cons/risks	Example references
Windowed	Time/window schedule or accuracy decay	No (unsupervised)	Fast recovery after drift; simple to implement	Unstable if the window is contaminated; window size sensitive	[7,8–10]
Drift-gated policies (e.g., ADWIN)	Change detector fires	No (uses change flag)	Adapts only when needed; avoids constant retraining	Flags indicate when to adapt, not anomalies; misalignment risk	[14,15,17]
Hybrid / physics-informed residuals	Residual vs simple physical model	No (unsupervised)	Better fault identifiability under shift	Requires a domain model; residual drift is still possible	[1,5,6,22]

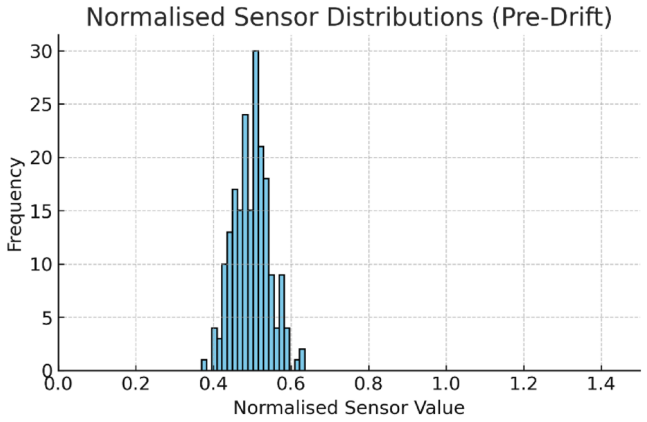

Figure 1. Normalized sensor distributions during the pre-drift phase (cycles <80); x-axis fixed to 0–1.5

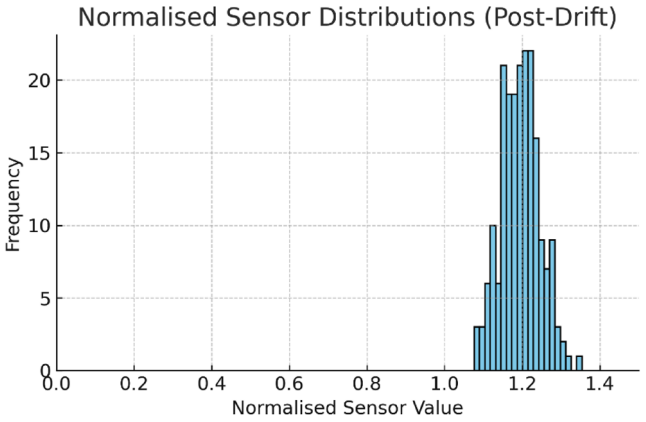

Figure 3. Normalized sensor distributions during the post-drift phase (cycles >120); x-axis fixed to 0–1.5

Here, $m(80) = 1.0$, $m(120) = 1.5$, and $m(c) = 1.0$ for c <80 and $m(c) = 1.5$ for c >120; the multiplier is applied only to sensors 1–5. Each value $s_i(c)$ for $i = 1\ldots5$ is replaced by $s_i(c)\cdot m(c)$; sensors 6–21 remain unchanged.

3.4. Data Normalization. In real-world streaming pipelines, the normalization parameters are often fixed during deployment to avoid leakage of future information. We fitted a MinMaxScaler to all 21 sensor features using only the pre-drift data (cycles <80). The scaler maps each feature to $[0, 1]$ based on its minimum and maximum values in the pre-drift window. We then applied the same scaler to transform every cycle (including drift and anomalies), ensuring that no future statistics influence the model decisions. After normalization, data at cycles >120 (drifted sensors 1–5) often exceed the $[0, 1]$ range in raw terms, but are represented relative to the original pre-drift bounds. Figures 1–3 show the histograms of the normalized sensor distributions across the three segments.

3.5. Model Configurations.

3.5.1. Isolation Forest (IF). We used the PyOD implementation of Isolation Forest [11] with 100 estimators and contamination = 0.05 (matching the evaluated anomaly rate). The model was trained solely on the pre-drift normalized data (cycles <80).

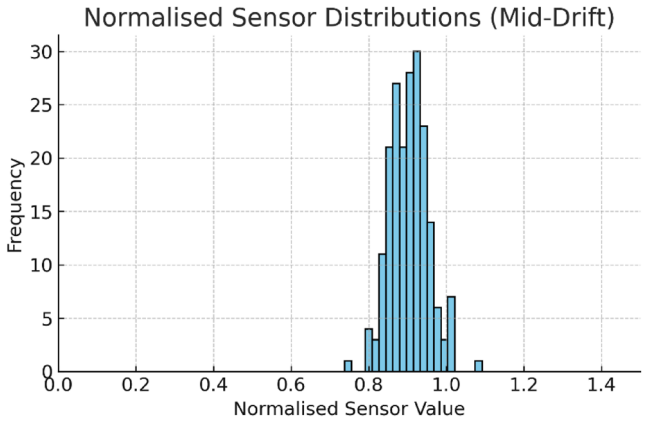

Figure 2. Normalized sensor distributions during the mid-drift phase (cycles 80–120); x-axis fixed to 0–1.5

3.5.2. One-Class SVM (OCSVM). We employ PyOD's OCSVM [12] with an RBF kernel, $\nu = 0.05$ (matching contamination), and $\gamma =$ 'scale'. Similar to the IF, we trained the OCSVM on the pre-drift normalized data. This creates a hypersphere (or hyper-surface) enclosing normal points, where the points outside are flagged as anomalies.

3.5.3. ADWIN. ADWIN is a streaming drift detector that adaptively maintains a sliding window whose size adjusts to detect distribution changes with confidence δ. We instantiated ADWIN with $\delta = 0.002$. As we streamed each new cycle, we computed the sum of sensors 1–5 and input it into the ADWIN update() method. If ADWIN reported a change in cycle c, we flagged that single cycle as an anomaly proxy (binary 1); otherwise, we flagged it at 0. This is a deliberately naive usage: ADWIN is not designed to detect individual point anomalies but rather to drift in the aggregated statistic. We included this to evaluate whether raw change flags correlated with the anomaly events.

3.6. Streaming Experiment Protocol. We simulated the streaming by iterating through the cycles in order. The pipeline is as follows:

1. **Training (cycles <80):** IF and OCSVM were fit to pre-drift normalized data (cycles <80).
2. **Drift simulation (cycles 80–120):** $m(c)$ was gradually appllied to sensors 1–5 and then normalized using the pre-drift scaler.
3. **Anomaly injection (all cycles):** At pre-selected cycles, Gaussian noise was added to sensors 1–5.
4. **Inference:** At each cycle c, the 21-dimensional normalized feature vector $x(c)$ was extracted, static detector predictions $y_a(c)$ (IF or OCSVM, both trained on cycles <80). Were obtained, and $\sum(i=1)^5 xi(c)$ was fed into ADWIN to obtain drift flag $d(c)$
5. **Evaluation segments:**
   - **Pre-drift:** c <80
   - **Mid-drift:** $80 \leq c \leq 120$
   - **Post-drift:** c >120
6. **Metrics:** For each model and segment, Precision, Recall, and F1 were computed between the predicted labels and ground truth as is_anomaly.

Table 2. *Isolation Forest performance by drift phase (retraining at cycle 120)*

Phase	Precision	Recall	F1
Pre-drift	0.857	0.857	0.857
Mid-drift	0.117	0.208	0.140
Post-drift	0.080	0.080	0.080
Post-retrain	0.080	0.080	0.080

Note:. The model trained on cycles of <80. Retraining was performed at cycle 120 using the mixed post-drift data.

7. **Retraining (cycle 120):** At c = 120, all data up to c = 120 (drifted normal + anomalies)were collected, IF was retrained on this mixture, and the post-drift (c >120) was evaluated to measure recovery.
8. **Recovery latency:** For ADWIN, the earliest post-drift cycle where a 10-cycle rolling F1 $\geq$ 0.9 $\times$ pre-drift F1 was computed; for IF with retraining, recovery is immediate at cycle 120 if F1 post-retrain $\geq$ 0.9 $\times$ pre-drift F1.

3.7. Reproducibility and Availability. Implementation uses scikit-learn[23], Matplotlib[24], and PyOD[25]. The codes and ready-to-run Google Colab are available (repository branch: feat/multi- unit-juri-revision). The artifacts for this revision are as sollows: results/juri_revision/ (cross_unit_summary .csv; unit_{1,2,3}_metrics.json; fig5_unit _{1,2,3}_adwin_v._sano malies.png). In Google Colab, the data path /content/cmapss _data/CMaps/.

4. EXPERIMENTAL RESULTS

Across Units 1–3, the same pattern holds (Table 4): the Isolation Forest degrades sharply after drift, the One-Class SVM remains weak throughout, and ADWIN's change flags do not coincide with the anomaly timestamps.

4.1. Isolation Forest Performance. Table 2 shows that Isolation Forest performs well under stationary conditions (F1 = 0.857), but fails under drift, with performance collapsing to F1 = 0.140 mid-drift and further degrading post-drift. Retraining at cycle 120 does not help, due to overlap between drifted normal and anomalous patterns.

4.2. One-Class SVM Performance. As shown in Table 3, the OC-SVM performed worse than theIF across all phases, starting at a low (F1 = 0.364) and collapsing entirely under drift. Its sensitivity to boundary noise and overfitting from the RBF kernel prevents recovery even with retraining.

4.3. ADWIN as Anomaly Proxy. ADWIN failed to detect anomalies in any drift phase (Table 4, F1 = 0). although it flags some distributional changes mid-drift, these do not align

Table 3. *One-Class SVM performance by drift phase*

Phase	Precision	Recall	F1
Pre-drift	0.444	0.308	0.364
Mid-drift	0.143	0.130	0.136
Post-drift	0.080	0.080	0.080

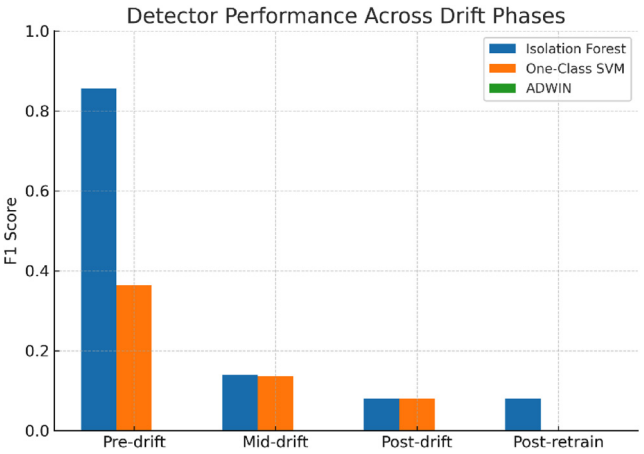

Figure 4. Detector performance across drift phases, reported as F1 score. Isolation Forest collapses after drift, One-Class SVM performs poorly throughout, and ADWIN fails to align change flags with anomalies

with anomaly timestamps. This mismatch indicates that change detection does not equate to anomaly detection.

4.4. Recovery Latency Analysis. The phase-wise detector performance is summarized in Figure 4 (introduced in Section 4.3); we used it to analyze the recovery latency. We define recovery latency as the earliest cycle t $\geq$ 120 where a 10-cycle rolling F1 $\geq$ 0.9 $\times$ pre-drift F1. For ADWIN, pre-drift F1 = 0, so any non-zero F1 could be counted; however, ADWIN never scored above zero, so no recovery was observed. For IF with scheduled retraining at cycle 120, post-retraining F1 = 0.080, whereas 0.9 $\times$ pre-drift F1 = 0.771. IF did not meet this threshold, indicating no meaningful recovery. In effect, neither streaming drift detection nor fixed retraining restores acceptable anomaly detection performance.

To provide a consolidated view of detector performance across all phases, Figure 4 reports the F1 scores for the Isolation Forest, One-Class SVM, and ADWIN. The comparison confirms three key findings: (i) static detectors collapse once drift sets in, (ii) simple retraining fails to restore performance, and (iii) drift detectors such as ADWIN, which aresensitive to distributional changes, do not align with anomaly incidence.

4.5. Cross-Unit Robustness. To gauge whether the observed failure was idiosyncratic to an individual engine or a universal phenomenon, we applied our protocol to other FD001 units. Table 4 presents the results for Units 2 and 3. The outcome closely resembles Unit 1: The Isolation Forest attains fair pre-drift accuracy but fails once drift begins, the One-Class SVM consistently performs below par across periods, and ADWIN generates change flags irrelevant to anomalies, resulting in zero F1 across the board. These reproducible tendencies across multiple engines verify that the failure modes we observed are not isolated artifacts but inherent weaknesses of stationary detectors and naive drift signals under non-stationary conditions.

5. DISCUSSION

5.1. Limitations of Static Detectors. Static detectors trained only on pre-drift data degrade sharply once a moderate

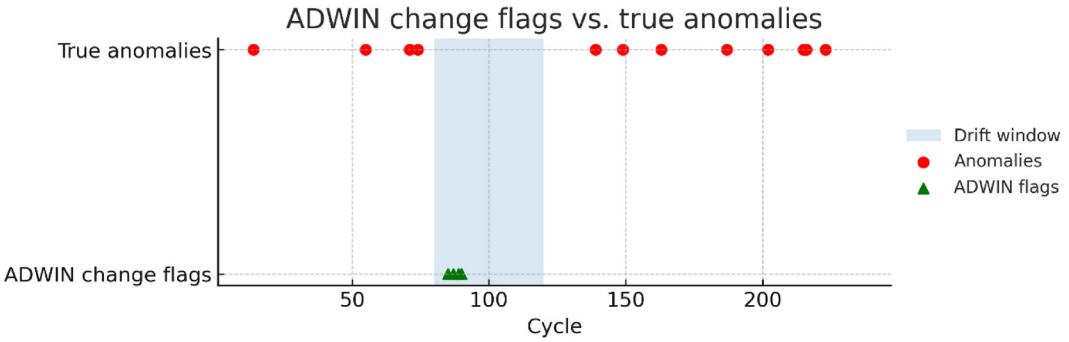

Figure 5. ADWIN change flags versus injected anomalies on FD001 (sensors 1–5); drift window shaded for cycles 80–120.Sub-subsection 1

Table 4. *Detector performance across multiple FD001 units*

Unit	Detector	Pre-drift F1	Mid-drift F1	Post-drift F1
1	Isolation Forest	0.857	0.140	0.080
1	One-Class SVM	0.364	0.136	0.080
1	ADWIN	0.000	0.000	0.000
2	Isolation Forest	0.750	0.182	0.069
2	One-Class SVM	0.167	0.178	0.069
2	ADWIN	0.000	0.000	0.000
3	Isolation Forest	0.667	0.095	0.127
3	One-Class SVM	0.400	0.093	0.127
3	ADWIN	0.000	0.000	0.000

Note: Units 1–3 of the FD001 dataset. Anomalies were injected at 5% across sensors 1–5. Drift window = 80–120 cycles. The results are consistent with those of Unit 1: The Isolation Forest collapses under drift, One-Class SVM performs poorly throughout, and ADWIN fails to align drift flags with the anomalies.

drift hits the key sensor channels. Isolation Forest performs well in static settings, but its F1 drops to 0.140 mid-drift; the One-Class SVM performs weakly before drift and collapses thereafter. Scheduled retraining at cycle 120 is ineffective because the drifted normal data and injected anomalies are confounded; therefore, retraining does not separate the evolving normal range from the actual faults.

Mechanistically, the failure modes are different. Isolation Forest's training-time partitions approximate pre-drift quantiles, and when the means shift for sensors 1–5, large regions of now-normal readings fall into sparse partitions and are flagged as outliers. One-class SVM, a margin-based method, shifts its decision function toward the new mean, and with increased variance after drift, nominal and anomalous scores overlap, driving both false positives and false negatives [2,3,11,12].

Sensor-wise effects mirror this story: mid-drift right-shifts are concentrated on sensors 1–5 by design (Figures 1–3), whereas sensors 6–21 remain comparatively stable and contribute less to errors. In other words, IF falters because its partitions no longer fit the data, while OCSVM falters because its boundary adapts poorly when the variance itself is non-stationary.

5.2. Misapplication of Drift Detectors. As configured, ADWIN is designed to signal significant shifts in aggregated statistics and not isolate single-cycle anomalies. Its failure (F1 = 0) confirms that the raw change flags should not be used directly as anomalous flags. As Figure 5 shows, ADWIN produces change flags around the onset of drift, yet none coincide with the anomaly timestamps. This mismatch underlines the fact that distributional change signals, which are useful as adaptation triggers, do not substitute anomaly labels. ADWIN detects changes in the mean of a data stream by maintaining adaptive windows; in our setup, its change points are correctly aligned with the drift interval (cycles 80–120), but these signals reflect distributional change, not anomalous behavior. Instead, drift detectors trigger contextual adaptation steps. For example, one might pause anomaly scoring, collect a clean window of data after a drift for retraining, or resume detection. Future work may explore pipelines in which ADWIN change signals automatically initiate a retraining process on a filtered subset of non-anomalous data [15,18,19].

5.3. Implications for Real-World Deployment. Predictive maintenance systems cannot rely on static unsupervised detectors in environments subject to slow, predictable degradation. The lag between the drift onset and retraining, in addition to the confounding effect of drifted normal points with false anomalies, underscores the need for real-time drift-aware mechanisms. As Figure 4 shows, no detector maintained a usable performance after the drift onset (see Figure 5 for the drift window), highlighting the operational risk of naive deployments. The same collapse repeats across Units 1–3 (Table 4). Possible approaches include the following:

- **Ensemble detectors:** combine multiple anomaly detectors trained over different time windows and weigh them based on the drift confidence.
- **Drift-gated retraining:** uses drift detector signals to temporarily suspend anomaly scoring, gather a verified normal reference window (e.g., from maintenance cycles), and retrain detectors on clean data.
- **Feature weighting:** dynamically adjusts feature importance; sensors known to drift (such as temperature readings) can be downweighted in the anomaly score once drift is detected.
- **Hybrid physics-AI models:** incorporate physics-based models or digital twins to anchor the expected sensor behavior and augment them with data-driven detectors to isolate anomalies above physical thresholds.

5.4. Limitations and Future Directions. This study focuses on FD001 (single operating condition) with additive

Gaussian anomaly injections of a fixed magnitude across sensors 1–5 and a multiplicative drift from cycles 80 to 120. These choices stabilize the benchmark but limit generalizability: real faults often follow multi sensor context-dependent patterns. Future work should extend the protocol to FD002–FD004 (multiple operating regimes) and other datasets, incorporate richer anomaly forms (e.g., engineered fault signatures and correlated patterns), and evaluate deployment constraints at the edge by reporting per-cycle latency, memory footprint for sliding windows, and retraining cost, along with detection delay and false-alarm rate. Benchmarking under resource-limited hardware clarifies the trade-offs between faster recovery and constrained computing.

Beyond static or periodically retrained detectors, future studies should focus on systems that can adapt as objects change. This means moving toward hybrid frameworks that react to drift the moment they appear, rather than after the fact. One promising approach is drift-triggered ensembles, which update or replace their models when sensing a shift. The other is online learning methods that gently reshape decision boundaries as new data flows. There is also a growing interest in physics-aware deep networks — for instance, long short-term memory (LSTM) autoencoders paired with digital twin residuals — that can model complex behaviors without losing interpretability. Testing these adaptive ideas on multi regime datasets such as FD002–FD004 under real-time conditions would reveal how well they balance quick responsiveness with reliability and efficiency at the edge.

6. CONCLUSION

We presented a comprehensive benchmark of unsupervised anomaly detectors under concept drift in a streaming turbofan engine scenario. By injecting drift and anomalies into NASA's C-MAPSS FD001 data, we show that static detectors degrade sharply (IF: pre-drift F1 = 0.857 to mid-drift F1 = 0.140), naive retraining fails, and streaming drift detectors (ADWIN) cannot serve as direct anomaly proxies (F1 = 0). These findings highlight the need for drift-aware anomaly-detection strategies that separate evolving normal signals from genuine fault events. We have publicly released our code and drift simulation protocol to encourage replication, extension, and innovation.

AFFILIATIONS AND AUTHOR DETAILS

Undergradute Author and Corresponding Author

Rakan M. AlKhulaif – *Department of Applied Computer Science, King Saud University, Riyadh 11451, Saudi Arabia;*
0009-0008-4131-3325
Email: 442105575@student.ksu.edu.sa

ACKNOWLEDGEMENTS

This study did not receive any specific funding.

CONFLICTS OF INTEREST

The author declares no conflicts of interest.

REFERENCES

(1) Chandola, V., Banerjee, A. & Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **41**, 15 (2009).
(2) Gama, J. et al. A survey on concept drift adaptation. *ACM Comput. Surv.* **46**, 44 (2014).
(3) Lu, J. et al. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **31**, 2346–2363 (2019).
(4) Widmer, G. & Kubat, M. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.* **23**, 69–101 (1996).
(5) Hundman, K. et al. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding Proc. KDD 387–395 (2018).
(6) Malhotra, P. et al. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148v2* (2016).
(7) Gama, J., Sebastião, R. & Rodrigues, P. On evaluating stream learning algorithms. *Mach. Learn.* **90**, 317–346 (2013).
(8) Brzezinski, D. & Stefanowski, J. Reacting to different types of concept drift: The accuracy updated ensemble. *IEEE Trans. Neural Netw. Learn. Syst.* **25**, 81–94 (2013).
(9) Domingos, P. & Hulten, G. Mining high-speed data streams. *Proc. KDD* **2000**, 71–80.
(10) Gama, J., Castillo, G. & Rodrigues, P. Incremental learning of decision trees. *Proc. ACM SAC* **2006**, 1105–1110.
(11) Liu, F. T., Ting, K. M. & Zhou, Z.-H. Isolation Forest. *Proc. IEEE Int. Conf. Data Mining* **2008**, 413–422.
(12) Schölkopf, B. et al. Estimating the support of a high-dimensional distribution. *Neural Comput.* **13**, 1443–1471 (2001).
(13) Bifet, A. & Gavaldà, R. Learning from time-changing data with adaptive windowing. *Proc. SIAM Int. Conf. Data Mining* **2007**, 443–454.
(14) Saxena, A., Goebel, K., Saha, B. & Eklund, N. Damage propagation modelling for aircraft engine run-to-failure simulation. *Proc. PHM 2008*, 1–9 (2008).
(15) Gama, J., Medas, P., Castillo, G. & Rodrigues, P. Learning with drift detection. *SBIA* **2004**, 286–295.
(16) Kifer, D., Ben-David, S. & Gehrke, J. Detecting change in data streams. *Proc. VLDB* **2004**, 180–191.
(17) Baena-García, M. et al. Early drift detection method. *KDD Work. Knowl. Discov. Data Streams* **2006**, 77–86.
(18) Basseville, M. & Nikiforov, I. V. *Detection of Abrupt Changes: Theory and Application*. (Prentice Hall, 1993).
(19) Page, E. S. Continuous inspection schemes. *Biometrika* **41**, 100–115 (1954).
(20) Breunig, M. M. et al. LOF: Identifying density-based local outliers. *Proc. ACM SIGMOD* **2000**, 93–104.
(21) Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. KDD* **1996**, 226–231.
(22) Lai, G., Chang, W.-C., Yang, Y. & Liu, H. Modeling long- and short-term temporal patterns with deep neural networks for time series forecasting. *Proc. SIGIR* **2018**, 95–104.
(23) Pedregosa, F. et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
(24) Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
(25) Zhao, Y., Nasrullah, Z. & Li, Z. PyOD: A Python toolbox for scalable outlier detection. *J. Mach. Learn. Res.* **20**, 1–7 (2019).