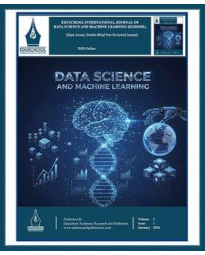
EDUSCHOOL INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344



Data-Centric AI and MLOPS: Drift Detection and Model Monitoring

V Sakhtipriyanka

Assistant Professor, Department of Computer Science, Kongunadu Arts and Science College, Coimbatore, India

Article information

Received: 23rd March 2026
Received in revised form: 27th April 2026
Accepted: 29th May 2026
Available online: 18th July 2026

Volume: 1
Issue: 2
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0007

Abstract

Machine learning systems in production fail in ways that training metrics rarely predict. Once a model serves live traffic, the data it receives drifts away from the distribution it learned, and accuracy degrades silently. This paper examines the shift from a model-centric view, where the dataset is fixed and architectures are tuned, toward a data-centric practice in which data quality, validation, and continuous monitoring become first-class engineering concerns. The discussion is organized around the machine learning operations (MLOps) lifecycle, covering continuous integration, delivery, and training, together with feature stores and model registries that make pipelines reproducible. A taxonomy of distribution shift is presented, separating covariate shift, prior or label shift, and concept drift, each with distinct detection needs. Detection methods are reviewed across two settings: batch statistics such as the population stability index, Kullback-Leibler and Jensen-Shannon divergence, the Kolmogorov-Smirnov test, and maximum mean discrepancy, and streaming detectors such as ADWIN and the drift detection method. Illustrative metrics show detection delay and false-alarm trade-offs across these methods, alongside a monitored stability index crossing an alert threshold that triggers retraining. Retraining strategies, governance, and current tooling are discussed. The aim is a practical reference for teams that must keep deployed models reliable as their input data changes.

Keywords:- Data-Centric AI, MLOps, Concept Drift, Covariate Shift, Model Monitoring, Population Stability Index, ADWIN, Retraining.

.

I. INTRODUCTION

A trained model that scores well on a held-out test set is only the beginning of a production system. The larger and more error-prone part of the system is the surrounding infrastructure for data collection, feature extraction, serving, and monitoring. Sculley and colleagues described this gap as hidden technical debt, noting that only a small fraction of a real-world machine learning system is the model code itself [1]. The remainder consists of configuration, data dependencies, and glue code whose failure modes are subtle and expensive.

Once deployed, a model operates under an assumption that is almost never true for long: that the data it receives at inference time follows the same distribution as the data it was trained on. Customer behavior changes, sensors age, upstream pipelines are modified, and adversaries adapt. The result is distribution shift, and its

practical consequence is a quiet decline in predictive quality that no offline metric will reveal until labels arrive, sometimes weeks later [2], [3].

Two responses have shaped recent practice. The first is data-centric AI, an emphasis on systematically improving the quality and consistency of data rather than only refining model architecture [4]. The second is machine learning operations, or MLOps, which adapts software engineering discipline to the lifecycle of models, adding continuous training to the familiar continuous integration and delivery [5], [6]. Drift detection and model monitoring sit at the intersection of these two ideas: they convert the abstract goal of reliable data into concrete signals that an operations team can act on.

This paper provides a structured account of that intersection. Section II reviews data-centric AI and the MLOps lifecycle. Section III sets out a taxonomy of drift. Section IV surveys detection and monitoring methods for both batch and streaming settings. Section V discusses retraining strategies, governance, and illustrative results. Section VI reviews tooling and open challenges, and Section VII concludes. Figure references and two summary tables support the discussion. Reported numbers are illustrative and are not drawn from a single deployed study.

II. DATA-CENTRIC AI AND THE MLOPS LIFECYCLE

For much of the past decade, progress was measured by holding a benchmark dataset constant and searching for better architectures. Data-centric AI inverts that emphasis. It treats the model as relatively fixed and asks how labeling consistency, coverage of rare cases, and removal of corrupted records change downstream accuracy [4]. In many applied settings, cleaning a noisy label set yields a larger gain than swapping one network for another, and the gain is easier to sustain.

MLOps gives this view an operational backbone. The lifecycle in Fig. 1 runs from data ingestion and validation, through a feature store and versioned datasets, to training, a model registry, and serving. What distinguishes it from a one-off training script is the closed loop: monitoring observes live data and predictions, drift detection raises alerts, and a retraining trigger feeds a continuous training pipeline that produces a new candidate model. Continuous integration tests code and data schemas, continuous delivery promotes validated models, and continuous training automates the refresh [5], [6].

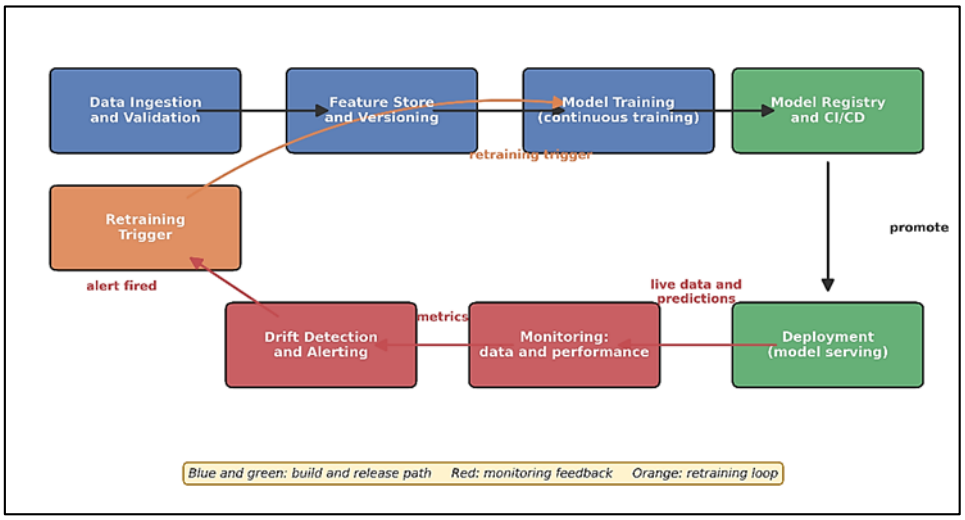


Fig. 1: MLOps lifecycle with a closed monitoring and retraining loop. Drift signals from production feed continuous training.

Several components make the loop reproducible. A feature store provides consistent feature computation for both training and serving, removing the skew that arises when offline and online code diverge [7]. A model registry records versions, lineage, and stage transitions, so that any served prediction can be traced to a specific model and dataset. Data validation tools encode expectations about schema, ranges, and distributions, and flag records that violate them before they reach the model [8], [9].

Quality of the engineering itself can be scored. Breck and colleagues proposed a rubric for production readiness, the ML Test Score, covering tests for features and data, model development, infrastructure, and monitoring [10]. Teams that adopt such rubrics tend to detect problems earlier, because the rubric forces explicit checks that are otherwise skipped under delivery pressure.

III. TYPES OF DRIFT

Distribution shift is not a single phenomenon. Let X denote inputs and y the target. Training assumes a joint distribution P(X, y); production may present a different joint distribution. Decomposing P(X, y) as P(y | X) P(X) or as P(X | y) P(y) clarifies which part has moved, and this distinction drives both detection and remedy [2], [11]. Fig. 2 organizes the categories.

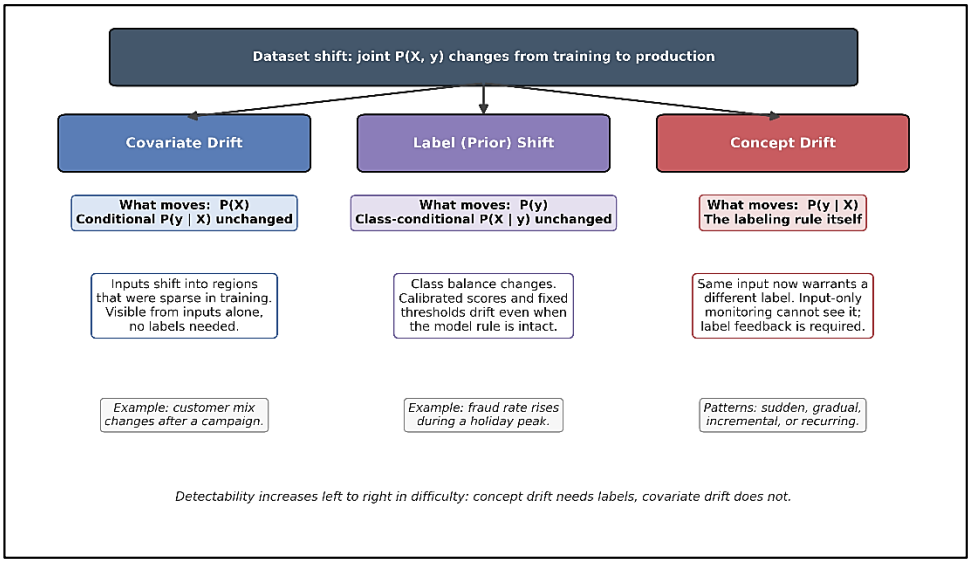


Fig. 2: Taxonomy of distribution shift, separating covariate shift, prior or label shift, and concept drift.

A. Covariate (Data) Drift

Covariate drift means P(X) changes while the conditional P(y | X) stays the same. The relationship the model learned is still valid, but it is queried in regions of input space that were sparse during training. A credit model trained before a change in customer demographics is a typical case. Covariate drift is the most directly observable form, because input features are available immediately, without waiting for labels [3], [12].

B. Prior (Label) Shift

Prior shift means the marginal P(y) changes while P(X | y) holds. The mix of classes seen in production differs from training, which distorts calibrated probabilities and any fixed decision threshold. Fraud rates that rise during a holiday season, or disease prevalence that changes across regions, produce prior shift. Detection often relies on estimating the label distribution from model outputs when true labels are delayed [11].

C. Concept Drift

Concept drift is the change of P(y | X) itself: the target relationship moves. The same input now warrants a different prediction. Gama and colleagues classify concept drift by its temporal pattern as sudden, gradual, incremental, or recurring [2]. Sudden drift follows a discrete event such as a policy change; gradual drift blends old and new concepts over time; recurring drift returns to a previous regime, as with seasonal patterns. Because concept drift alters the labeling rule, input-only monitoring cannot detect it, and label feedback or a proxy signal becomes necessary.

Table 1. Categories of distribution shift and their detection needs.

Drift type	What changes	Observable from inputs only	Typical remedy
Covariate (data)	P(X)	Yes	Reweighting, retraining
Prior (label) shift	P(y)	Partly (via outputs)	Threshold or prior correction
Concept drift	P(y | X)	No, needs labels	Retraining on recent data
Recurring concept	P(y | X), seasonal	No	Model ensembles, context flags

IV. DETECTION AND MONITORING METHODS

Detection methods divide into two families. Batch methods compare a current window of data against a fixed reference, usually a training or recent baseline sample. Streaming methods process examples one at a time and maintain an online estimate of change. The choice depends on latency budget, label availability, and the cost of a false alarm [13].

A. Batch Statistical Tests

The population stability index (PSI) bins a feature and sums the weighted log-ratio of bin proportions between reference and current samples. A common rule treats PSI below 0.10 as stable, 0.10 to 0.25 as moderate, and above 0.25 as a strong shift that warrants action. PSI is inexpensive and widely used in credit risk, though it is sensitive to bin choice. Kullback-Leibler divergence measures the information lost when the current distribution approximates the reference, and its symmetric, bounded variant, the Jensen-Shannon divergence, is preferred for monitoring because it stays finite when bins are empty [14].

For continuous features, the two-sample Kolmogorov-Smirnov test compares empirical cumulative distributions and returns a statistic and p-value, making it convenient for automated alerting. For high-dimensional or multivariate data, maximum mean discrepancy (MMD) compares distributions in a reproducing kernel Hilbert space and detects shifts that per-feature tests miss [15]. Rabanser and colleagues evaluated such tests for detecting shift and recommend combining a dimensionality reduction step with a statistical test, reporting that no single method dominates across shift types [13].

B. Streaming Detectors

When data arrives as a stream and rapid response matters, online detectors are used. The Drift Detection Method (DDM) tracks the online error rate of a classifier and signals a warning, then a drift, when the error rises by a set number of standard deviations above its minimum [16]. ADWIN maintains a variable-length window and automatically drops older sub-windows when their mean differs significantly from recent data, giving change detection with formal guarantees and no fixed window size [17]. These methods need a label or error signal, so they suit settings where feedback is fast.

C. Monitoring Data Quality and Performance

Drift detection is one layer of a broader monitoring stack. Data quality monitoring checks schema conformance, missing-value rates, and range violations at ingestion, catching pipeline faults that masquerade as drift [8], [9]. Performance monitoring tracks accuracy, precision, recall, calibration, and business metrics once labels are available, and uses proxy signals such as prediction confidence distributions while labels are pending. Fig. 3 shows a monitored stability index for a transaction-amount feature that stays low for several weeks, then climbs after an upstream change and crosses the alert threshold, triggering retraining.

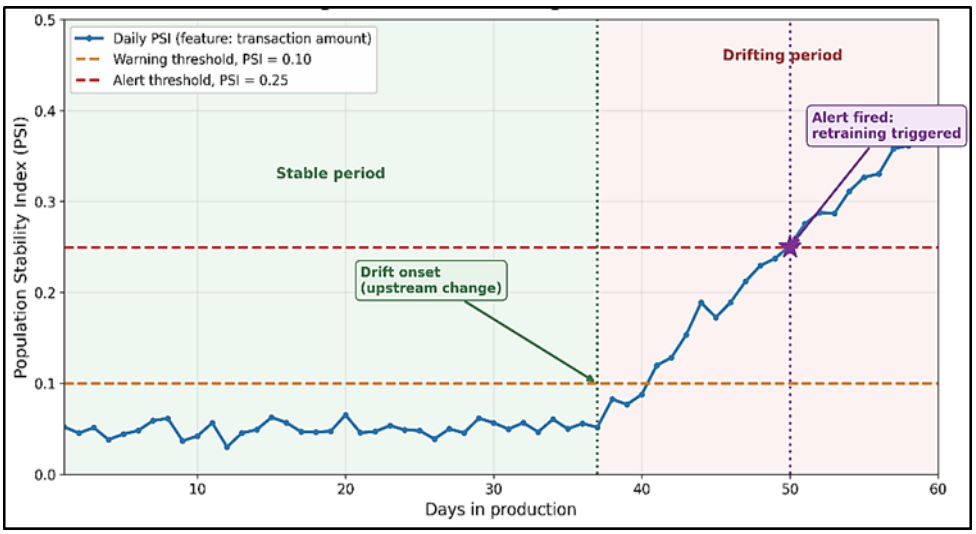


Fig. 3: Monitored population stability index for one feature crossing warning and alert thresholds, triggering a retraining event.

Method selection involves trade-offs between how quickly a true shift is detected and how often the monitor raises a false alarm. Fig. 4 gives illustrative values for mean detection delay and false-alarm rate across six methods. Streaming detectors such as ADWIN react with shorter delay than batch tests on the same stream,

while a poorly tuned detector can trade that speed for more spurious alerts. Table 2 summarizes the operating characteristics.

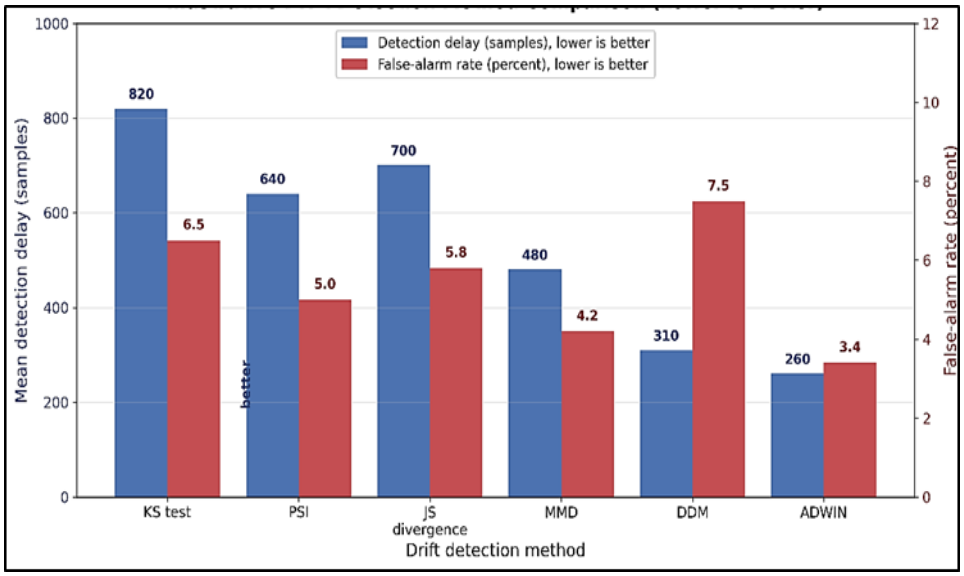


Fig. 4: Illustrative comparison of drift-detection methods by mean detection delay and false-alarm rate.

Table 2. Comparison of common drift-detection methods.

Method	Setting	Data type	Needs labels	Main use
PSI	Batch	Binned	No	Feature drift in credit risk
KL / JS divergence	Batch	Distributions	No	Distribution distance
KS test	Batch	Continuous	No	Per-feature shift test
MMD	Batch	Multivariate	No	High-dimensional shift
DDM	Streaming	Error stream	Yes	Online error monitoring
ADWIN	Streaming	Error / value stream	Yes	Adaptive windowed change

V. RETRAINING, GOVERNANCE, AND ILLUSTRATIVE RESULTS

An alert is useful only if a defined response follows. Retraining strategies fall along a spectrum. Scheduled retraining refreshes the model at fixed intervals regardless of drift, which is simple but wastes computation when data is stable and lags when it is not. Triggered retraining fires when a monitored metric crosses a threshold, as in Fig. 3, aligning cost with need. Online or incremental learning updates the model continuously from the stream, which suits fast-moving data but risks instability if noisy batches are absorbed without guardrails [2], [18].

A retraining decision should weigh the expected accuracy recovery against the cost and risk of deploying a new model. In the illustrative scenario of Fig. 3, accuracy had fallen by roughly nine points from a baseline of 0.91 by the time the alert fired, and triggered retraining on the most recent labeled window restored accuracy to within two points of the baseline. A new candidate is validated against a held-out recent set and a canary slice of live traffic before full promotion, so that a regression is caught before it reaches all users.

Governance turns these mechanics into a repeatable, auditable process. A model registry records which model is live, who approved it, and which dataset trained it. Rollback procedures let an operator revert to a prior version within minutes. For regulated domains, documentation artifacts such as model cards and datasheets record intended use, evaluation slices, and known limitations [19], [20]. Access controls and approval gates separate the act of training from the act of promotion, reducing the chance that an untested model reaches production.

Fairness and stability deserve continued monitoring after deployment, not only at launch. A model that was balanced across groups at training time can drift into disparate error rates as subpopulation distributions change, so slice-level metrics belong in the monitoring dashboard alongside aggregate accuracy [21]. Tying drift

alerts to these slices helps a team notice when degradation concentrates in a sensitive group rather than spreading evenly.

VI. TOOLING AND OPEN CHALLENGES

A practical ecosystem has formed around these ideas. Open-source libraries provide drift tests and monitoring reports, including Evidently for data and prediction drift dashboards [22] and the Alibi Detect library for outlier and drift detection with MMD and related tests [23]. Data validation frameworks such as TensorFlow Data Validation and Great Expectations encode schema and distribution checks in the pipeline [8], [24]. Experiment tracking and registry tools record runs, parameters, and model lineage. Streaming detectors are available through libraries for online learning, where ADWIN and DDM are standard components.

Open problems remain. Detecting concept drift without labels is fundamentally hard, since the labeling rule is exactly what is hidden; proxy signals help but cannot fully substitute for ground truth. Setting thresholds is a balance that depends on the cost of a missed shift versus a false alarm, and good values are domain-specific. High-dimensional and unstructured data, such as text and images, complicate per-feature tests and push monitoring toward learned representations and embedding-space distances [13], [15]. Finally, monitoring infrastructure itself must be maintained, lest the monitors become another source of silent failure [1], [25].

VII. CONCLUSION

Reliable machine learning in production depends less on a single strong model than on the discipline that keeps that model aligned with changing data. The data-centric view and the MLOps lifecycle together reframe deployment as an ongoing process of validation, monitoring, and retraining rather than a one-time release. Drift, separated into covariate, prior, and concept categories, calls for different detection tools, from batch statistics such as PSI, the Kolmogorov-Smirnov test, and maximum mean discrepancy, to streaming detectors such as ADWIN and DDM. Coupled with data-quality checks, performance tracking, governance, and a clear retraining policy, these tools let teams notice degradation early and respond in a controlled way. The illustrative results here show the shape of the problem; the engineering value lies in wiring these signals into an automated, auditable loop that holds a deployed model to the standard its users expect.

REFERENCES

[1] 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 Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 28, pp. 2503–2511, 2015.

[2] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Comput. Surveys, vol. 46, no. 4, pp. 1–37, 2014.

[3] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodriguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," Pattern Recognit., vol. 45, no. 1, pp. 521–530, 2012.

[4] A. Ng, "A chat with Andrew on MLOps: From model-centric to data-centric AI," DeepLearning.AI, 2021. [Online]. Available: https://www.deeplearning.ai/

[5] D. Sculley *et al*., "MLOps: Continuous delivery and automation pipelines in machine learning," Google Cloud Architecture Center, 2020. [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[6] D. Kreuzberger, N. Kuhl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023.

[7] L. E. Orr, A. Sanyal, X. Ling, K. Goel, and M. Leszczynski, "Managing ML pipelines: Feature stores and the coming wave of embedding ecosystems," Proc. VLDB Endowment, vol. 14, no. 12, pp. 3178–3181, 2021.

[8] E. Breck, N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data validation for machine learning," in Proc. Conf. Mach. Learn. Syst. (MLSys), 2019.

[9] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," ACM SIGMOD Rec., vol. 47, no. 2, pp. 17–28, 2018.

[10] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML Test Score: A rubric for ML production readiness and technical debt reduction," in Proc. IEEE Int. Conf. Big Data, pp. 1123–1132, 2017.

[11] A. Storkey, "When training and test sets are different: Characterizing learning transfer," in *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, pp. 3–28, 2009.

[12] M. Sugiyama, M. Krauledat, and K.-R. Muller, "Covariate shift adaptation by importance weighted cross validation," J. Mach. Learn. Res., vol. 8, pp. 985–1005, 2007.

[13] S. Rabanser, S. Gunnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 32, pp. 1396–1408, 2019.

[14] J. Lin, "Divergence measures based on the Shannon entropy," IEEE Trans. Inf. Theory, vol. 37, no. 1, pp. 145–151, 1991.

[15] [15] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Scholkopf, and A. Smola, "A kernel two-sample test," J. Mach. Learn. Res., vol. 13, pp. 723–773, 2012.

[16] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in Proc. Brazilian Symp. Artif. Intell. (SBIA), LNCS 3171, pp. 286–295, 2004.
[17] 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.
[18] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," J. Mach. Learn. Res., vol. 11, pp. 1601–1604, 2010.
[19] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in Proc. ACM Conf. Fairness, Accountability, Transparency (FAT*), pp. 220–229, 2019.
[20] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daume III, and K. Crawford, "Datasheets for datasets," Commun. ACM, vol. 64, no. 12, pp. 86–92, 2021.
[21] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 29, pp. 3315–3323, 2016.
[22] Evidently AI, "Evidently: An open-source framework to evaluate, test, and monitor ML models in production," 2024. [Online]. Available: https://github.com/evidentlyai/evidently
[23] A. Van Looveren, J. Klaise, G. Vacanti, O. Cobb, A. Scillitoe, R. Samoilescu, and A. Athorne, "Alibi Detect: Algorithms for outlier, adversarial and drift detection," J. Open Source Softw., vol. 9, no. 97, p. 5742, 2024.
[24] Great Expectations, "Great Expectations: Data quality testing for data pipelines," 2024. [Online]. Available: https://github.com/great-expectations/great_expectations
[25] D. Sculley, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high-interest credit card of technical debt," in NeurIPS Workshop on Software Engineering for Machine Learning, 2014.