Journal of Information and Data Management, 2026, 17:1, doi: 10.5753/jidm.2026.5738

Evaluating Data Drift Detection and Its Effects on Machine Learning System Performance

Lucas Helfstein Rocha [**Universidade de São Paulo** | ***lucashelfs@ime.usp.br***]
Kelly Rosa Braghetto [**Universidade de São Paulo** | ***kellyrb@ime.usp.br***]

Institute of Mathematics and Statistics, University of São Paulo, R. do Matão, 1010, Butantã, São Paulo, SP, 05508-090, Brazil.

Received: 29 June 2025 • **Published:** 13 March 2026

Abstract Software systems incorporating machine learning (ML) components are being increasingly deployed across various domains. Unlike traditional systems, ML systems are highly dependent on the quality of their input data, making their performance susceptible to changes in that data. This work explores the potential for improving ML systems by actively monitoring data flow and retraining models in response to drift detection. We begin by evaluating several widely used statistical and distance-based methods for detecting data drift, highlighting their advantages and limitations. Subsequently, we present experimental results using these methods on datasets exhibiting concept drift from the literature, as well as synthetic datasets with data drift. Our findings demonstrate how these techniques can enhance the robustness of ML systems, offering automatic adaptation regardless of the type of drift encountered.

Keywords: Machine Learning Systems, Data Drift Detection

1 Introduction

Software systems incorporating machine learning components are increasingly being adopted across a wide range of domains. Unlike traditional software systems, where performance is primarily determined by static code, the effectiveness of machine learning systems depends on the characteristics of their input data. Once deployed in a production environment, these systems may encounter input data with distributions that differ, sometimes substantially, from those used during model training [Gama and Castillo, 2006]. Such variations can lead to unforeseen behaviors, potentially resulting in critical failures. Therefore, when developing software that integrates machine learning, it is essential to establish well-defined operational requirements for model deployment, monitoring, and retraining to ensure consistent and reliable system performance [Sculley *et al*., 2015].

This work builds upon the study by Helfstein and Braghetto [2024], which applied data drift detection techniques to enhance classifier performance in the presence of concept drift. In that study, the Hellinger Distance (HD), the Jensen-Shannon (JS) Divergence, and the Kolmogorov-Smirnov (KS) Test were employed for drift detection. Expanding on this foundation, we specifically examine the Hellinger Distance Drift Detection Method [Ditzler and Polikar, 2011] and leverage its adaptive thresholding strategy to develop a novel detection approach based on the JS Divergence. Additionally, this article extends the previous analysis by evaluating these techniques on synthetic datasets with introduced data drifts, providing a more in-depth discussion of their performance in this new setting.

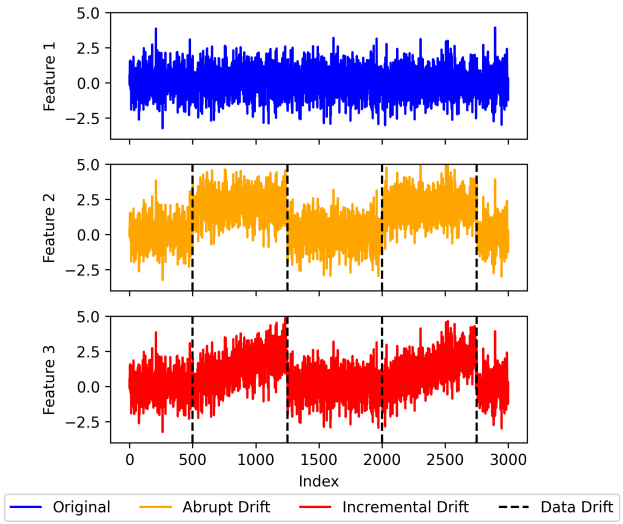
2 Data Drift

In machine learning systems, a common component is the classifier. In a classification task, a model learns a function f that maps input variables ($\mathcal{X}$), representing the feature space, to discrete output labels ($\mathcal{Y}$). After the model is trained and the system is deployed, it is subjected to two different types of deviations: *data drifts* and *concept drifts*.

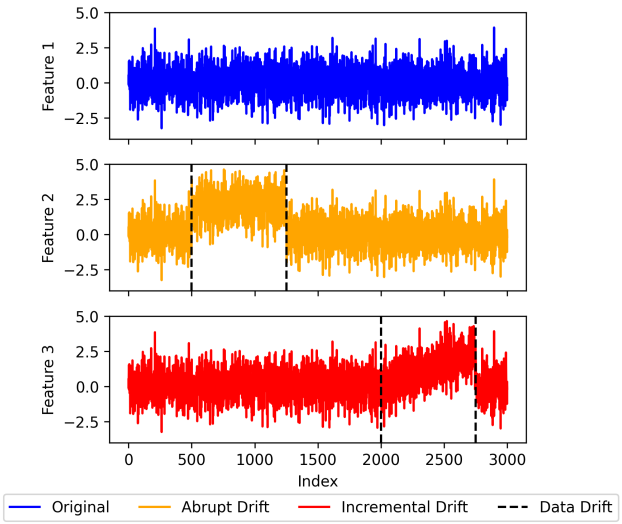
Data drift occurs when the distribution of data input to the system in production becomes distant from that of the data $\mathcal{D}$ used in model training, making it possible to detect a significant difference between them. Concept drift occurs when the relationship between the training attributes and the model classes changes [Webb *et al*., 2016]; that is, the attributes that determined one class start to determine another. In a general way, a drift occurs whenever one of the probability distributions $P(\mathcal{X})$, $P(\mathcal{X}|\mathcal{Y})$ or $P(\mathcal{Y}|\mathcal{X})$ changes over time. Moreover, drifts may occur with different velocities and patterns (e.g., abrupt, gradual, incremental, reoccurring) [Souza *et al*., 2020].

Figure 1a illustrates the effect of abrupt and incremental drifts on a data stream of 3,000 instances, where all features initially follow the same normal distribution. Drifts were introduced at 1,500 points and occur in parallel within two distinct drift windows. In contrast, Figure 1b presents a switching drift scenario, where changes affect only Feature 2 in the first drift window and Feature 3 in the second.

To detect drifts, it is necessary to monitor the extent of the divergence between the distributions of the sets. In this work, we focus on two categories of drift detection techniques very commonly used in practice: *statistical* and *distance-based* methods.



(a) Recurring parallel drift with two drift windows



(b) Switching drifts with alternated drift windows

Figure 1. Visualization of different drift patterns: (a) recurring parallel drift and (b) switching drifts.

2.1 Statistical Methods

Statistical methods typically use a formal statistical hypothesis test to compare the distributions of two datasets (e.g., current data vs. reference data). They generate a statistical measure (e.g., p-value) indicating the likelihood that the two distributions are the same. A low p-value suggests that the distributions are significantly different, indicating drift. They can be sensitive to sample size and assumptions about the underlying distributions. When data scarcity is not a concern, nonparametric methods (which do not assume any specific data distribution) are more often used [Dasu *et al.*, 2006]. Common nonparametric tests include the Wilcoxon, Kolmogorov-Smirnov, and multinomial tests.

2.1.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) statistical test can be used to test whether two samples came from the same distribution. Assuming F and G are the empirical distribution functions of the two samples and the sample sizes are n and m, respectively, the Kolmogorov–Smirnov statistic $KS(F, G)$ is defined as follows [Hodges Jr, 1958]:

$$KS(F, G) = \sup_x |F(x) - G(x)| \tag{1}$$

The decision rule is to reject the hypothesis at the significance level α if $KS(F, G) > c(\alpha)\sqrt{\frac{n+m}{n\times m}}$, where $c(\alpha)$ is given in the Kolmogorov table [Hodges Jr, 1958].

2.1.2 Multiple Univariate Kolmogorov-Smirnov Test

While the univariate KS Test assesses the difference between the empirical distribution of two datasets in one dimension, the Multiple Univariate KS Test does this across multiple dimensions simultaneously. However, when multiple hypotheses are tested, the probability of observing a rare event increases, increasing the likelihood of incorrectly rejecting a null hypothesis. According to Rabanser *et al.* [2019], a conservative aggregation method can be used to reduce the probability of this type of error, the Bonferroni correction [Bland and Altman, 1995], to adjust the significance level for multiple comparisons.

Applying it to the Multiple Univariate KS Test for d distributions, the decision rule is to reject the hypothesis at the significance level α if $\min_{k=1,2,\ldots,d} KS(F_k, G_k) > c(\frac{\alpha}{d})\sqrt{\frac{n+m}{n\times m}}$, where $KS(F_k, G_k)$ is the KS Test for the empirical distribution functions F and G of the k-th dimension, whose respective sample sizes are n and m. Note that Bonferroni correction is applied by testing the hypothesis at a significance level of α/d.

2.2 Distance-Based Methods

Distance-based methods offer a more direct measure of the distance or dissimilarity between two probability distributions than statistical methods. Moreover, they do not rely on specific distributional assumptions. They provide a numerical value indicating the degree of difference: the larger the distance, the greater the difference, indicating drift.

2.2.1 Kullback–Leibler Divergence

For two discrete probability distributions P and Q, the Kullback–Leibler (KL) Divergence (or relative entropy) $KL(P||Q)$ from Q to P is defined as [Dasu *et al.*, 2006]:

$$KL(P||Q) = \sum_{x\in\mathcal{X}} P(x)log(\frac{P(x)}{Q(x)}) \tag{2}$$

2.2.2 Jensen–Shannon Divergence

The Jensen–Shannon (JS) Divergence is a method based on KL Divergence but with an improvement: it is symmetric. For two discrete probability distributions P and Q, the Jensen–Shannon Divergence $JS(P||Q)$ is defined as:

$$JS(P||Q) = \frac{1}{2}KL(P||\frac{(P+Q)}{2}) + \frac{1}{2}KL(Q||\frac{(P+Q)}{2}) \tag{3}$$

2.2.3 Hellinger Distance

For two discrete probability distributions $P(p_1, p_2, \ldots, p_n)$ and $Q(q_1, q_2, \ldots, q_n)$, the Hellinger distance $H(P, Q)$ is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{4}$$

2.3 Drift Detection Methods

Choosing a distance function to measure change is only one aspect of the drift detection problem. Another crucial aspect is determining the statistical significance of the observed change [Dasu *et al*., 2006]. This involves specifying a null hypothesis (i.e., that no change has occurred) and assessing how likely it is that the observed measurement could occur under this hypothesis. Some drift detection approaches, such as HDDDM, monitor the magnitude of the change in some distance metric between a new distribution and a baseline distribution, which can or cannot be updated every time a drift is detected. This way, these methods can be used to detect changes in data distribution over time, which can help maintain the performance of machine learning models in dynamic environments.

The following sections define some methods that use this approach and are evaluated in this work. The first one, HDDDM, was presented by Ditzler and Polikar [2011]. The second and third methods, JSDDM and KSDDM, are the new methods proposed in this work. All of these approaches assume an incremental learning setting, where new datasets are presented in batches over time; they are not based on the classifier's performance (i.e., they rely only on raw features); and they are classifier-free.

It is important to note that, in terms of computational cost, these techniques are comparable. Each method derives empirical distributions from the same input data. The computation of drift is linear with respect to the number of bins, which is dictated by the batch size.

2.3.1 Hellinger Distance Drift Detection Method (HDDDM)

Ditzler and Polikar [2011] proposed a drift detection algorithm that relies only on the raw data features to estimate whether drift is present in a supervised incremental learning scenario, using the Hellinger distance as a measure to infer whether drift is present between two batches of training data using an adaptive threshold.

The Hellinger Distance-based Drift Detection Method (HDDDM) proposed by Ditzler and Polikar [2011] assumes that data arrives in batches, with dataset $\mathcal{D}_t$ becoming available at time t. The following steps summarize the method, assuming as initialization $\lambda = 1$, $\mathcal{D}_\lambda = \mathcal{D}_1$, and for $t = 2, 3, \ldots$:

1. Generate histograms P from $\mathcal{D}_t$ and Q from $\mathcal{D}_\lambda$, each one with $b = \lfloor \sqrt{|\mathcal{D}_t|} \rfloor$ bins.
2. Calculate the Hellinger distance $\delta_H(t)$ between P and Q and its difference from $\delta_H(t-1)$:

$$\delta_H(t) = \frac{1}{d} \sum_{k=1}^{d} \sqrt{\sum_{i=1}^{b} \left(\frac{P_{i,k}}{\sum_{j=1}^{b} P_{j,k}} - \frac{Q_{i,k}}{\sum_{j=1}^{b} Q_{j,k}} \right)^2}$$
$$\epsilon(t) = \delta_H(t) - \delta_H(t-1) \tag{5}$$

   where d is the dimensionality of the data, and $P_{i,k}$($Q_{i,k}$) is the frequency count in bin i of the histogram corresponding to P (Q) of feature k.
3. Update the adaptative threshold $\hat{\epsilon}$ and the standard deviation $\hat{\rho}$:

$$\hat{\epsilon} = \frac{1}{t-\lambda-1} \sum_{i=\lambda}^{t-1} |\epsilon(i)| \qquad \hat{\rho} = \sqrt{\frac{\sum_{i=\lambda}^{t-1} (|\epsilon(i)| - \hat{\epsilon})^2}{t-\lambda-1}} \tag{6}$$

4. Compute the actual threshold $\beta(t) = \hat{\epsilon} + \gamma\hat{\rho}$, where γ is some positive constant, indicating how many standard deviations of change around the mean indicate drift.
5. Determine if drift occurred:
   Drift is present if $|\epsilon(t)| > \beta(t)$. In this case, it is necessary reset $\mathcal{D}_\lambda$ by making $\mathcal{D}_\lambda = \mathcal{D}_t$ and $\lambda = t$.
   When drift is not present, $\mathcal{D}_\lambda$ must be expanded to include $\mathcal{D}_t$: $\mathcal{D}_\lambda = \{\mathcal{D}_\lambda, \mathcal{D}_t\}$.

HDDDM signalizes a drift when the magnitude of the change ($|\epsilon(t)|$) is significantly greater than the average of the change since the last detected change ($\hat{\epsilon}$). The significance is controlled by γ and the standard deviation of the divergence differences ($\hat{\rho}$).

2.3.2 Jensen-Shannon Drift Detection Method (JSDDM)

To create a drift detection method based on the JS Divergence, we adapted the HDDDM method by replacing the Hellinger Distance in Equation 5 in Step 2 of the method with the JS Divergence defined in Equation 3, as shown in the following:

$$\delta_{JS}(t) = \frac{1}{d} \sum_{k=1}^{d} JS(P_k || Q_k) \qquad \epsilon(t) = \delta_{JS}(t) - \delta_{JS}(t-1) \tag{7}$$

where d is the data dimensionality, and P_k(Q_k) is the distribution histogram of feature k.

2.3.3 Kolmogorov-Smirnov Drift Detection Method (KSDDM)

In order to detect data drifts using the Kolmogorov-Smirnov (KS) Test, we devised the following simple algorithm, which assumes that data arrives in batches, with dataset $\mathcal{D}_t$ becoming available at time t, $\lambda = 1$, $\mathcal{D}_\lambda = \mathcal{D}_1$, and for $t = 2, 3, \ldots$:

1. Generate the empirical distribution functions F_k from $\mathcal{D}_t$ and G_k from $\mathcal{D}_\lambda$ for each feature k in data.

2. Obtain the minimum of the KS Test of all features' empirical distributions:

$$\delta_{KS}(t) = \min_{k=1,2,\ldots,d} \{KS(F_k, G_k)\} \quad (8)$$

where F_k (G_k) is the empirical distribution function of feature k in $\mathcal{D}_t$ ($\mathcal{D}_\lambda$).

3. Determine if drift occurred:
Drift is present at the significance level α if $\delta_{KS}(t)$ is greater than $c(\frac{\alpha}{d})\sqrt{\frac{n+m}{n\times m}}$, where $m = |\mathcal{D}_t|$, $n = |\mathcal{D}_\lambda|$, d is the dimensionality of the data and $c(\frac{\alpha}{d})$ is given in the Kolmogorov table.
If drift is detected, it is necessary to reset $\mathcal{D}_\lambda$ by making $\mathcal{D}_\lambda = \mathcal{D}_t$ and $\lambda = t$.
When drift is not present, $\mathcal{D}_\lambda$ must be expanded to include $\mathcal{D}_t$: $\mathcal{D}_\lambda = \{\mathcal{D}_\lambda, \mathcal{D}_t\}$.

3 Datasets Used in the Experiments

In our experiments, we used datasets cited by Lu *et al*. [2018], a literature review on learning under concept drift. The objective of utilizing these datasets is to examine how data drift detection methods respond in a concept drift scenario. Specifically, we employ two datasets in which the distributions remain stable while only the concept drifts. Additionally, we used synthetic datasets containing data drift to further analyze the behavior of the proposed techniques.

3.1 Insects Datasets

The Insects Datasets [Souza *et al*., 2020] is derived from a real-world streaming application that uses optical sensors to capture data and recognize flying insect species in real time using a Smart Trap. The temperature is the main environmental factor influencing the insect behavior in the trap and, consequently, the data captured. Observations were ordered over time based on temperature patterns, and examples were uniformly sampled within each temperature to isolate temperature-induced drifts. Then, data were split into 11 datasets with 33 features, showcasing different patterns of change (incremental, abrupt, gradual, reoccurring), balanced and unbalanced classes, and class distribution changes.

3.2 SEA Datasets

The Streaming Ensemble Algorithm (SEA) synthetic dataset [Street and Kim, 2001] simulates a data stream with abrupt drift. Each observation consists of three features, with only the first two being relevant. The target is binary and positive if the sum of the relevant features exceeds a specific threshold. Concept drift is introduced by switching the threshold (i.e. changing the classification function).

We used an implementation of the SEA dataset provided by River, a library to build online machine-learning models. In the library, there are four different variants to be chosen as threshold configuration: (0) threshold = 8, (1) threshold = 9, (2) threshold = 7, and (3) threshold = 9.5. Using them, we built two different experimental setups:

1. **SEA**: a stream that starts with variant 0, has variant 3 in the middle, then returns to variant 0, simulating a deviation in concept for a period;
2. **MULTISEA**: a scenario that has 12500 items from each of the four variants.

3.3 STAGGER Datasets

The STAGGER synthetic datasets introduced by Schlimmer and Granger [1986], like SEA, simulate data with abrupt concept drift. In STAGGER, objects are described by three features – size (small, medium, and large), shape (circle, square, and triangle), and color (red, blue, and green) – and the target is a boolean value given by a function f. The River library provides three variants for f: (0) True if the size is small and the color is red, (1) True if the color is green or the shape is a circle, and (2) True if the size is medium or large. Changing f causes concept drift. Using River, we built two experimental setups:

1. **STAGGER**: a stream that starts with variant 0, has variant 1 in the middle, then returns to variant 0, simulating a deviation in concept for a period;
2. **MULTISTAGGER**: a scenario with 16,666 items from each of the three variants.

3.4 Electricity

The Electricity Pricing dataset [Harries, 1999] is used in our experiment to simulate the concept drifting and class imbalance environment, which originally contains 45, 312 samples drawn from 7 May 1996 to 5 December 1998 with one sample for every half an hour from the electricity market in New South Wales, Australia.

3.5 Magic Gamma Telescope

The Magic Gamma Telescope dataset is Monte Carlo generated to simulate high-energy gamma particle registration in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The images captured by the telescope allow the discrimination of primary gammas (signal) from hadronic showers caused by cosmic rays (background) [Bock, 2007]. The original dataset has little drift; hence, it has been modified by sorting the feature *fConc1* in ascending order and generating incremental batches with meaningful data drift, as described in [Ditzler and Polikar, 2011].

3.6 Synthetic Datasets

To enable a more precise and rigorous evaluation of data drift detection techniques, this study also utilizes controlled synthetic datasets, allowing for accurate oversight of drift occurrences and assessment of detection performance. The datasets represent a random binary classification problem and were generated using an implementation of the work by Guyon [2003], available in the scikit-learn Python module.

The datasets used in the experiments exhibit drift patterns similar to those described in Section 2. Each dataset consists of 80,000 entries with five features, of which 20,000 entries contain drift. The experiments consider scenarios with

recurring abrupt and recurring incremental drifts, occurring either in parallel or in switching configurations, with drift windows evenly distributed throughout the dataset. The variations include SYN, a synthetic dataset without data drifts; SYN-PA, which contains parallel abrupt drifts; SYN-PI, with parallel incremental drifts; SYN-SA, featuring switching abrupt drifts; and SYN-SI, incorporating switching incremental drifts. The distributions of the SYN-PA and SYN-SI datasets are visualized through heatmaps presented in Section 4.

Detailed information regarding the data generation process, including drift configurations, drift rates, attribute structure, and noise control, is fully documented in the public repository associated with this work [1]. This documentation ensures the reproducibility of the experiments by providing transparent descriptions of the dataset creation process.

4 Analysis of Drift Detection

The following sections present visualizations of the detected drifts and the performance metrics for the detection techniques. To evaluate these techniques, the datasets were segmented into batches of 1,000, 1,500, 2,000, and 2,500 instances. These batch sizes were selected to balance dataset size constraints while ensuring a sufficient number of batches for a robust evaluation of the techniques both in datasets with data and concept drifts. Additionally, this segmentation allows for an analysis of how each technique's sensitivity varies with batch size.

For the synthetic datasets, the experiments comprised 5 dataset variations combined with 4 different batch sizes, resulting in a total of 20 configurations. For the real-world datasets focused on concept drift scenarios, there were 16 datasets also evaluated across 4 batch sizes, leading to 64 additional experiment configurations.

The following drift detection techniques were tested: Base (no drift detection), KS95 (KSDDM with $\alpha = 5\%$), KS90 (KSDDM with $\alpha = 10\%$), HD (HDDDM with $\gamma = 1$, as specified in [Ditzler and Polikar, 2011]), and JS (JSDDM with $\gamma = 1$). The $\gamma = 1$ value used in both HDDDM and JSDDM follows the original specification from Ditzler and Polikar [2011], and it was maintained in JSDDM to ensure consistency for comparative purposes since JSDDM is a direct adaptation of HDDDM using a different distance measure. Similarly, the α values of 5% (KS95) and 10% (KS90) correspond to standard confidence thresholds widely adopted in statistical hypothesis testing, representing conventional trade-offs between sensitivity and specificity. These parameters, along with the batch sizes selected, were empirically defined based on preliminary experiments aimed at maintaining classifier performance and a sufficient number of batches for meaningful drift analysis. While this approach aligns with typical practices in empirical machine learning research, we acknowledge that automated parameter tuning or sensitivity analysis could provide further robustness and remains an important direction for future work.

[1] `https://github.com/lucashelfs/EmpiricalAnalysisOfDataDrift/blob/main/synthetic_datasets_documentation.md`

4.1 Visualization of the Detected Data Drifts

Figure 2 illustrates the behavior of drift detection techniques on the MULTISEA dataset, which exhibits concept drift. Figures 3a and 4a provide examples of drift detection in datasets with synthetic data drift. In these figures, each point represents a detected drift in a given batch, with colors distinguishing the detection techniques. The vertical dashed lines indicate the locations of known drifts (concept or data drifts, depending on the dataset). At the top of each plot, the affected features and the corresponding batch intervals in which data drifts occur are indicated.

On the concept drift scenario (Figure 2), even though data drifts are being detected, they are not being detected specifically at the datasets' concept-changing points. For the case of the data drift scenarios (Figures 3a and 4a), the techniques are more prone to detect the drifts near the regions where drifts were added, behaving expectedly.

Heatmap plots can be used to visualize the drifts of all the features individually, as shown in Figures 3b and 4b. This type of plotting offers valuable insights into the drift patterns detected from the batches and the reference set. On these plots, darker color shades indicate a greater degree of divergence between specific batches and the reference batch.

In the incremental scenario depicted in Figure 3a, the KS test effectively detects changes within the drift interval, whereas JS and HD exhibit a slight delay in identifying variations in the feature distribution as data drift. Furthermore, Figure 3b illustrates how less significant differences are smoothed as the reference dataset expands, thereby enhancing the technique's adaptability to new data.

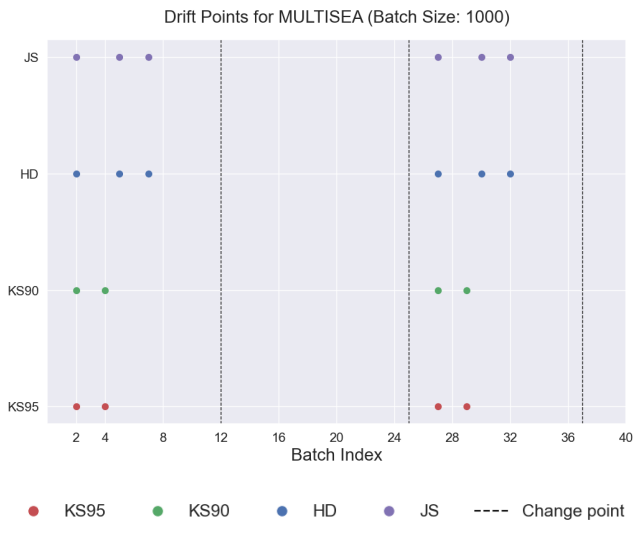
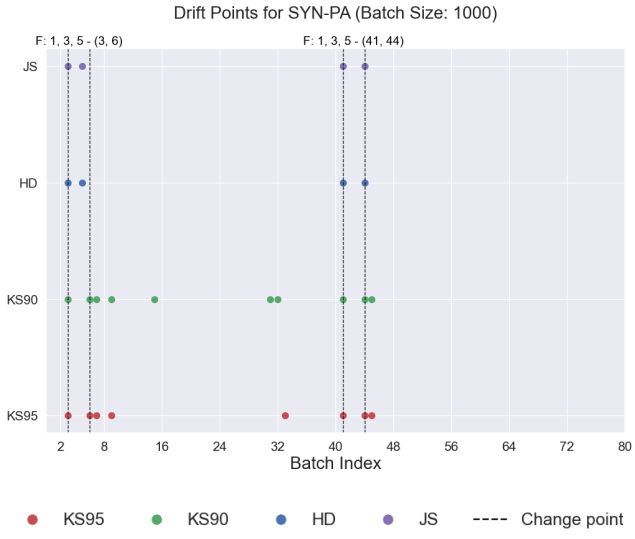
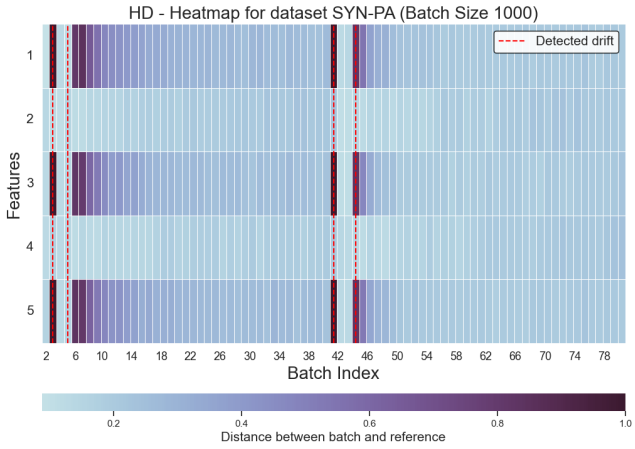


Figure 2. Detected drifts for the MULTISEA dataset (Batch size: 1,000).

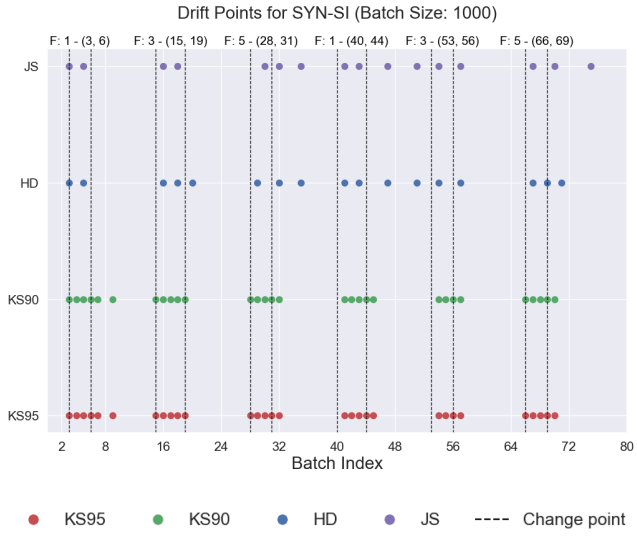


(a) Detected drifts

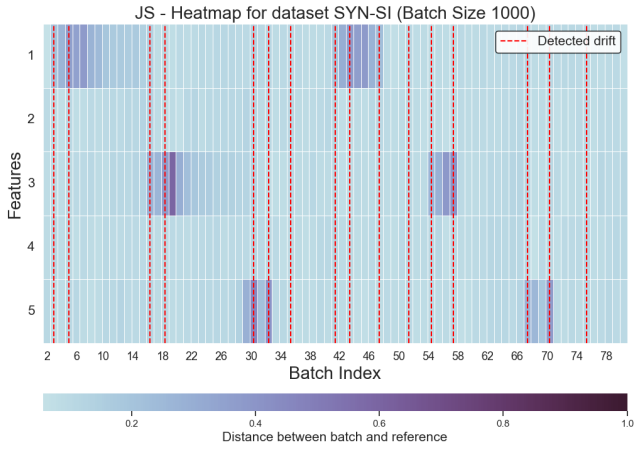


(b) Heatmap for the drifts detected using HDDM

Figure 3. Detected drifts and heatmap visualization for the synthetic dataset with parallel abrupt drifts (Batch size: 1,000).



(a) Detected drifts



(b) Heatmap for the drifts detected using JSDDM

Figure 4. Detected drifts and heatmap visualization for the synthetic dataset with switching incremental drifts (Batch size: 1,000).

4.2 Metrics

To evaluate the performance of the drift detection techniques on the new synthetic datasets, the batches containing drift in the feature distributions can be used to assess whether the techniques are detecting drift accurately. The evaluation framework defines a true positive (TP) as a correctly identified drift in a batch. A false positive (FP) occurs when drift is detected, but none is actually present. True negatives (TN) correspond to cases where no drift is detected, and none occurs. False negatives (FN) represent batches where drift is present but goes undetected.

This can be interpreted in Figures 3a and 4a, where the dots within the intervals between the dashed lines represent true positives (TP), while the dots outside these intervals represent false positives (FP). Batches that do not contain detections within the dashed lines are classified as false negatives (FN), while batches outside the dashed lines that also lack drift detections are likewise considered false negatives (FN).

Table 1 presents the performance metrics of the drift detection techniques when applied on the synthetic datasets variations with a batch size of 1,000 and 2,500. The tables include the metrics precision (P), recall (R), and F1-score (F1) calculated as presented in Equation 9.

$$P = \frac{TP}{TP+FP} \quad R = \frac{TP}{TP+FN} \quad \text{F1} = \frac{2PR}{P+R} \tag{9}$$

The results in Table 1 reveal how different drift patterns influence detection performance and, consequently, the retraining strategy of a classifier. Our primary interest lies in achieving higher recall values, as recall provides insight into the proportion of correctly identified instances within batches containing drift. Upon examination, it is evident that recall values tend to be higher for larger batch sizes, suggesting that smaller batch sizes may lead to less precise and assertive drift detection. Regarding precision, a lower precision score implies the potential for over-detection, which could introduce instability in a continuous learning system by triggering unnecessary retraining.

The techniques tend to detect drift less frequently in the case of abrupt drifts. This may be due to the inherent design of the drift detection methods, which may not be as responsive to the sudden changes characteristic of abrupt drift patterns. In contrast, for incremental drifts, the techniques can

Table 1. Precision (P), Recall (R), and F1-score (F1) of Drift Detection techniques for batch sizes 1,000 and 2,500.

Dataset	Tech.	Batch Size = 1,000			Batch Size = 2,500		
		P	R	F1	P	R	F1
SYN-PA	KS95	0.5	0.5	0.5	0.571	1.0	0.727
	KS90	0.4	0.5	0.444	0.571	1.0	0.727
	HD	1.0	0.5	0.667	1.0	1.0	1.0
	JS	1.0	0.5	0.667	1.0	1.0	1.0
SYN-PI	KS95	0.667	1.0	0.8	0.571	1.0	0.727
	KS90	0.571	1.0	0.727	0.571	1.0	0.727
	HD	1.0	0.5	0.667	1.0	0.5	0.667
	JS	1.0	0.5	0.667	1.0	0.5	0.667
SYN-SA	KS95	0.714	0.577	0.638	0.706	0.857	0.774
	KS90	0.714	0.577	0.638	0.706	0.857	0.774
	HD	0.688	0.423	0.524	1.0	0.286	0.444
	JS	0.688	0.423	0.524	1.0	0.286	0.444
SYN-SI	KS95	0.8	0.923	0.857	0.722	0.929	0.813
	KS90	0.8	0.923	0.857	0.722	0.929	0.813
	HD	0.588	0.385	0.465	0.6	0.429	0.5
	JS	0.562	0.346	0.429	0.6	0.429	0.5

more consistently detect the drift over time. This intermittent detection of abrupt drifts could be attributed to the gradual nature of drift detection algorithms, which are often tuned to perform better with slower, incremental shifts in data.

From the perspective of the techniques, KS-based methods (KS95, KS90) generally performed robustly across datasets, maintaining high recall and improving precision with larger batch sizes. This suggests they provide reliable drift detection, though careful tuning may be needed to balance them, particularly for switching drifts. HDDDM and JS-DDM, which rely on distance-based measures, show varying effectiveness depending on the dataset. While they struggle in terms of recall in certain incremental scenarios (e.g., SYN-SA at batch 2,500), they maintain reasonable precision in other cases, which helps reduce unnecessary retraining. These results suggest that no single technique is universally superior; rather, selection should be guided by the dataset's drift characteristics and the trade-off between detection sensitivity and retraining stability.

5 Using Drift Detection to Improve ML System's Performance

This study aims to provide an empirical demonstration of how statistical and distance-based methods for detecting data drift in input streams can be used to improve the performance metrics of a machine learning algorithm subjected to concept drifts and data drifts. We used the datasets described in Section 3 to (re)train classifiers and evaluate their performance metrics. The experiments were implemented in Python, using the `scikit-learn`, `scipy`, `river`, and `Menelaus` libraries [2]. All experiments were executed on a machine equipped with an Apple M3 Max chip and 48 GB of RAM.

5.1 An Approach for Using Detected Drifts to Improve a Classifier

The classifiers were evaluated using a prequential (test-then-train) approach, wherein a classifier is reset upon the detection of a drift by the used technique, an approach similar to the one used by Souza *et al*. [2020]. We used the algorithm described in the following for creating a Naive Bayes classifier and evolving it using a drift detection technique; the algorithm also creates a baseline classifier to compare the performances:

1. **Input**: Labeled data batches and a drift detection technique.
2. Use batch 1 to train a Naive Bayes classifier C_B – the baseline model.
3. Use batch 1 to train a Naive Bayes classifier C_D – the model that benefits from drift detection.
4. Set batch 1 as the reference batch.
5. **From batch 2 onwards:**
   (a) Store the predictions of both classifiers C_B and C_D for the current batch.
   (b) Check for drift between the reference set and the current batch using the drift detection technique.
   (c) Update C_B classifier with the current batch.
   (d) **If no drift is detected:**
      - Update C_D classifier with the current batch.
      - Update the reference set by merging it with the current batch.
   (e) **If drift is detected:**
      - Set the reference set to the current batch.
      - Reset classifier C_D training only with the new reference set.
6. **At the end of all batches:** Compute the performance metrics.

5.2 Experimental Results and Discussion

The algorithm described in Section 5.1 was implemented to compare the drift detection techniques specified in Section 2. We evaluated these techniques using the datasets described in Section 3 and measured various classification performance metrics for each method.

Given the diversity of datasets used in this study, the metrics most appropriate for evaluating classifier performance are the Area Under the Curve (AUC) and the F1 score. Detailed performance metrics for batch sizes of 1000, 1500, 2000, and 2500 are available in the repository tables[3]. In these tables, values in bold represent the highest F1 score for each dataset, while italics indicate the highest AUC. The column *Drift* reports the number of drifts detected by each technique. The *Base* column contains the results for the classifier C_B, which does not undergo retraining at any point.

To summarize the overall performance, Figures 5 and 6 present the mean F1 and AUC scores, respectively, aggregated across all datasets and batch sizes. These plots indicate

[2] `scikit-learn.org/`, `scipy.org/`, `riverml.xyz/`, `pypi.org/project/menelaus/`

[3] `https://github.com/lucashelfs/EmpiricalAnalysisOfDataDrift/blob/main/experimental_results_tables.md`

that all drift detection methods outperform the *Base* classifier, which is not retrained throughout the stream. This reinforces the importance of employing drift detection mechanisms to trigger model updates. Notably, the KSDDM techniques, particularly with a p-value of 0.05, achieved the highest average F1 and AUC scores. The HDDDM and JSDDM methods produced similar results to each other but were outperformed by the KSDDM variants in this aggregated analysis. It is worth noting that different choices for threshold parameters might influence this outcome, and exploring this remains an interesting direction for future work. Furthermore, for certain datasets, the absolute performance of the Naive Bayes classifier was relatively low across all methods. In such cases, the best-performing values—for example, an F1 score of 0.464—should be interpreted with caution. While these values represent the highest result within that experimental setup, they do not necessarily reflect satisfactory performance in absolute terms. This underscores the importance of considering dataset complexity and inherent task difficulty when interpreting comparative results. A higher value remains a valid indicator of relative performance but does not imply strong predictive performance overall.

The use of smaller batch sizes generally led to better results in terms of both F1 and AUC. In terms of the number of detected drifts relative to the total number of batches, the KS90 technique triggered more classifier resets than the other methods, closely followed by KS95. In contrast, HDDDM and JSDDM yielded similar results to each other, producing substantially fewer resets compared to the KS-based methods.

The KS Test is highly sensitive to data shifts, as discussed in Sections 4.1 and 4.2. This sensitivity often results in the excessive detection of drifts, which can lead to frequent retraining and, consequently, an overfitted classifier. While this behavior may yield favorable F1 and AUC scores, it can negatively impact the model's ability to generalize to unseen data. Conversely, the adaptive threshold mechanisms employed by HDDDM and JSDDM make these methods more conservative over time, which proves advantageous for datasets with more stable distributions, such as MULTISTAGGER. In this case, the drifts detected and the consequent retraining cycles led to F1 and AUC metrics that surpassed those achieved by the KSDDM approach. It is also worth noting that the substitution of the Hellinger Distance with the Jensen–Shannon Divergence in HDDDM—resulting in the JSDDM method—did not yield improvements over HDDDM in the evaluated scenarios. It is possible that alternative configurations of the thresholds could lead to different results, and exploring this remains a relevant direction for future work.

For the SYN datasets, the concepts exhibit greater stability, leading to more consistent and accurate predictions by the classifier. This stability, coupled with the behavior of the drift detection techniques, results in a retraining strategy that is neither excessive nor insufficient, thereby improving the overall robustness of the system. Moreover, for the SYN datasets with injected drifts, the best performance was observed when employing the KS-based techniques for drift monitoring.

The experimental results demonstrate that the analyzed drift detection techniques contribute meaningfully to improving system robustness, even in the presence of concept drift. Despite using a prequential evaluation framework, which incrementally trains the classifier on past data and tests it on the current batch, the strategy of resetting the classifier upon drift detection was found to improve performance when drifts occurred.

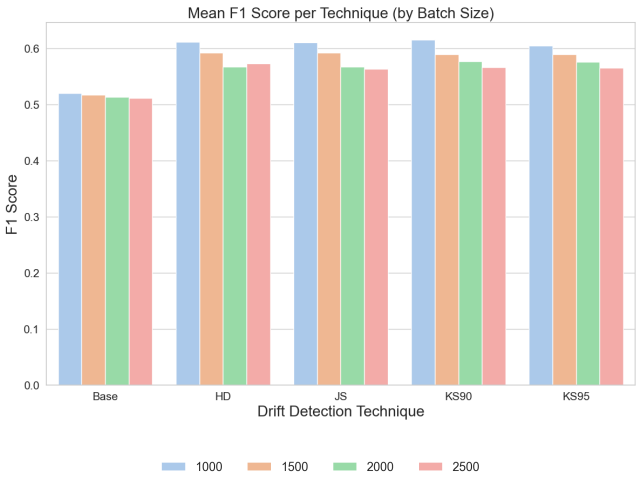


Figure 5. Mean F1 Score for all batch sizes.

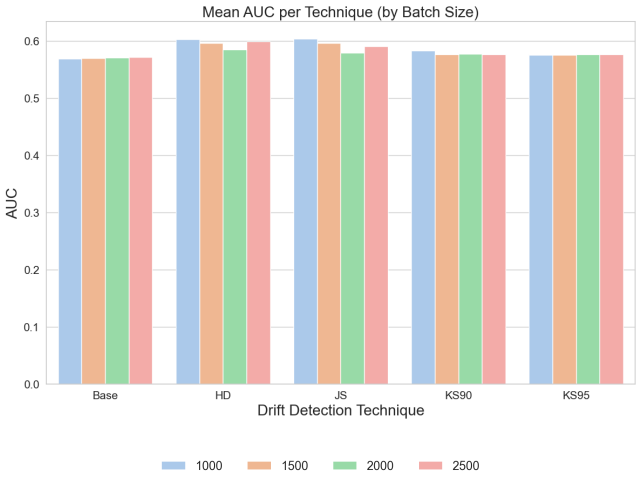


Figure 6. Mean AUC for all batch sizes.

Figure 7 presents the execution time as a function of batch size, corroborating the theoretical expectation that the computational complexity of the drift detection algorithms scales linearly with the number of bins rather than with the batch size itself. As larger batches result in fewer processed batches over the stream, a consistent reduction in drift detection time is observed across most datasets. This behavior is particularly evident in synthetic datasets with controlled drift dynamics. However, datasets characterized by high drift frequencies or significant class imbalance exhibit deviations from this pattern, likely due to the computational overhead associated with frequent drift detections and repeated model resets. These findings reinforce the scalability

of histogram-based drift detectors in typical data stream scenarios. A detailed analysis is available in the project repository[4]. While this runtime analysis provides an initial perspective on computational costs, further work is required to investigate how these factors impact large-scale streaming systems and whether algorithmic optimizations can mitigate these costs.

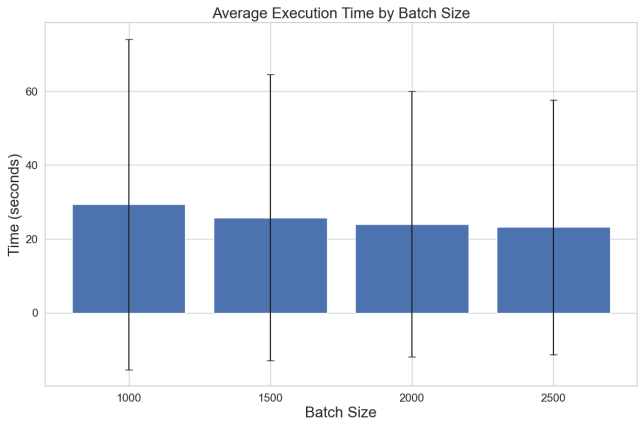


Figure 7. Average execution time for drift detection as a function of batch size, aggregated across all datasets.

6 Related Work

Several works approached the use of statistical and distance-based methods to detect drifts. For example, Dasu *et al*. [2006] presented a method based on the KL Divergence that uses a bootstrapping approach to establish the statistical significance of the distances with demonstrated statistical guarantees. An empirical evaluation (on real and synthetic data) was performed to show the accuracy of the approach. Pérez-Cruz [2008] proposed a method for estimating the KL Divergence between continuous densities. They showed that the divergence can be either estimated using the empirical cdf or k-nearest-neighbors density estimation.

Rabanser *et al*. [2019] investigated shift detection through the lens of statistical two-sample testing. In particular, they presented an empirical investigation on image datasets of how dimensionality reduction and two-sample testing might be combined in a practical pipeline for detecting distribution shifts in real-life ML systems.

Souza *et al*. [2020] discussed the challenges faced by the stream learning community concerning the reduced number of real-world data and the lack of a benchmark to evaluate adaptive classifiers and drift detectors. To mitigate these issues, the authors created 11 new datasets based on data collected by an optical sensor that measures the flying behavior of insects and evaluated several learning techniques on these datasets. They also provided a new public repository with datasets from real problems.

The novelty of the present work, when compared to these related works, lies in presenting variations of established drift detection techniques and demonstrating their effectiveness in scenarios involving both data and concept drifts.

7 Threats to the Validity of the Study

This study presents some limitations that should be considered when interpreting the results.

First, the drift detection techniques evaluated here are designed to detect changes in the input distribution $P(\mathcal{X})$, rather than changes in the conditional distribution $P(\mathcal{Y}|\mathcal{X})$, which characterizes concept drift. This was a deliberate decision to limit the scope, given the complexity of concept drift detection methods and the constraints of the article. Nevertheless, integrating data drift detection with concept drift detection remains an important direction for future work.

Second, all experiments were conducted using Naive Bayes classifiers. This choice allowed for clearer observation of the behavior of drift detection methods in controlled settings, particularly with the synthetic datasets. However, this restricts the generalizability of the results to more complex models. Evaluating the interaction of drift detection techniques with models such as decision trees, ensemble methods, and neural networks is a valuable area for further research.

Third, the evaluation metrics were limited to classifier performance measures, specifically AUC and F1-score. Due to the batch-based nature of the experiments, standard drift detection metrics—such as detection delay, false positive rate, and true detection rate—were not employed. While this choice aligns with the experimental design, incorporating these metrics could provide a more comprehensive assessment of the detectors' effectiveness.

Additionally, this study did not include statistical significance tests to formally assess whether observed differences between detection methods are meaningful. Memory consumption and scalability were also not evaluated—factors that are critical for deploying drift detection techniques in real-world streaming applications. These aspects are recognized as important and represent valuable directions for future research.

8 Conclusion

This study demonstrated the effectiveness of statistical and distance-based methods for detecting drift in data inputs and adapting classifiers. We evaluated JSDDM, a variation of the HDDDM method, and introduced a novel approach based on the Kolmogorov-Smirnov Test (KSDDM), analyzing their performance across different drift scenarios.

Through our experiments with datasets from various fields exhibiting concept drift and a synthetic dataset containing data drifts, we found that while the detection methods did not always immediately signal concept drift, they effectively identified data drifts, prompting necessary classifier resets. These results highlight the importance of considering data drifts in enhancing the robustness of machine learning systems.

[4] `https://github.com/lucashelfs/EmpiricalAnalysisOfDataDrift/tree/main/execution_time_analysis`

The effectiveness of each drift detection method varied significantly across datasets, which points to the need for a better understanding of dataset characteristics before comparing detection methods. In this regard, synthetic datasets provided valuable control over data and concept changes, facilitating comparative experiments and enabling deeper insights into detection performance.

Batch size played a crucial role in our experiments. Smaller batches yielded the best classifier performance when handling concept-drifted datasets, while larger batches slightly improved the performance metrics of drift detection techniques. This suggests that these methods can be effectively applied in real-world streaming scenarios, with batch sizes adjusted to meet specific application requirements.

Our experimental results demonstrate that the analyzed drift detection techniques contribute to improving the system's robustness, even in the presence of concept drift or data drift. Despite using prequential evaluation, which trains the classifier incrementally on past data and tests it on the current batch, the strategy of resetting the classifier was found to improve performance when drifts occurred.

Funding

This research was funded by grants #2023/00779-0 and #2023/18026-8, São Paulo Research Foundation (FAPESP), and CNPq proc. 420623/2023-0.

Availability of data and materials

The source code, datasets and experiment results analyzed on the experiments are available at `https://github.com/lucashelfs/EmpiricalAnalysisOfDataDrift`.

References

Bland, J. M. and Altman, D. G. (1995). Multiple significance tests: the Bonferroni method. *Bmj*, 310(6973):170.

Bock, R. (2007). MAGIC Gamma Telescope. https://doi.org/10.24432/C52C8B.

Dasu, T., Krishnan, S., Venkatasubramanian, S., and Yi, K. (2006). An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Symposium on the Interface of Statistics, Computing Science, and Applications (Interface)*.

Ditzler, G. and Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*, pages 41–48.

Gama, J. and Castillo, G. (2006). Learning with local drift detection. In *Advanced Data Mining and Applications: Second International Conference, ADMA 2006, Xi'an, China, August 14-16, 2006 Proceedings 2*, pages 42–55.

Guyon, I. (2003). Design of experiments of the nips 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction and feature selection*, volume 253, page 40.

Harries, M. (1999). Splice-2 comparative evaluation: electricity pricing. Technical report, The University of New South Wales, Sydney.

Helfstein, L. and Braghetto, K. R. (2024). An empirical analysis of data drift detection techniques in machine learning systems. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 40–52. SBC.

Hodges Jr, J. (1958). The significance probability of the Smirnov two-sample test. *Arkiv för matematik*, 3(5):469–486.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363.

Pérez-Cruz, F. (2008). Kullback-Leibler divergence estimation of continuous distributions. In *2008 IEEE international symposium on information theory*, pages 1666–1670.

Rabanser, S., Günnemann, S., and Lipton, Z. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32.

Schlimmer, J. C. and Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1:317–354.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 2015-Janua:2503–2511.

Souza, V. M. A., Reis, D. M., Maletzke, A. G., and Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34:1805–1858. DOI: 10.1007/s10618-020-00698-5.

Street, W. N. and Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., and Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994.